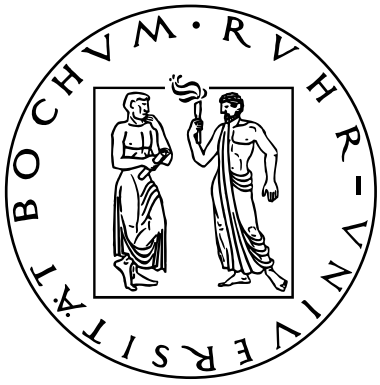
Extensions of Hierarchical Slow Feature Analysis for Efficient Classification and Regression on High-Dimensional Data



Dissertation

Submitted to the Faculty of Electrical
Engineering and Information Technology
at the
Ruhr University Bochum

for the
Degree of Doktor-Ingenieur

by
Alberto Nicolás Escalante Bañuelos

Bochum, Germany, July, 2017

Alberto Nicolás Escalante Bañuelos
Place of birth: Durango, Dgo., Mexico
Email: alberto.escalante@ini.rub.de
alberto.nicolas.escalante@gmail.com

Thesis advisor and first referee	Prof. Dr. Laurenz Wiskott Ruhr University Bochum, Germany
Second referee	PD Dr. Rolf Würtz Ruhr University Bochum, Germany
Thesis submission: Thesis defense:	January 18, 2017 May 3, 2017

Abstract

This thesis develops new extensions to slow feature analysis (SFA) that solve supervised learning problems (e.g., classification and regression) on high-dimensional data (e.g., images) in an efficient, accurate, and principled way. This type of problems has been addressed by convolutional neural networks (CNNs) in the last decade with excellent results. However, additional approaches would be valuable, specially those that are conceptually novel and whose design can be justified theoretically.

SFA is an algorithm originally designed for unsupervised learning that extracts slow (i.e., temporally stable) features. Advantages of SFA include a strong theoretical foundation and that it might be intimately connected to learning in biological systems. One can apply SFA to high-dimensional data if it is implemented hierarchically, a technique called hierarchical SFA (HSFA). The extensions to SFA listed in the following allow the construction and training of deep HSFA networks, yielding competitive accuracy and efficiency.

Graph-based SFA (GSFA) is a supervised extension to SFA that introduces the concept of *training graph*, a structure in which the vertices are samples (e.g., images) and edges represent transitions between pairs of samples. Edges have weights that can be interpreted as desired output similarities of the corresponding samples. Compared to SFA, GSFA solves a more general optimization problem and considers many more transitions. Information about label (or class) similarities is encoded in the graph by the strength of the edge weights. Many training graphs are proposed to handle regression and classification problems. The efficacy of GSFA is demonstrated on a subproblem of face detection.

The *exact label learning* (ELL) method allows to compute training graphs where the slowest feature(s) one could extract would be equal to the label(s), if the feature space were unrestricted. In contrast to previously proposed graphs, the edge weights of the resulting ELL graphs are set precisely as needed, improving the label estimation accuracy. Moreover, ELL allows to learn multiple labels simultaneously using a single network, which is more efficient than learning the labels separately and often results in more robust features.

Hierarchical information-preserving GSFA (HiGSFA) improves the amount of label information propagated from the input to the top node in hierarchical GSFA (HGSFA). HiGSFA computes two types of features: slow features that maximize slowness, as usual, and reconstructive features that minimize an input reconstruction error, following an information-preservation goal. HiGSFA is evaluated on the problem of age estimation (along with gender and race) from facial photographs, where it yields a mean average error of 3.50 years, outperforming current state-of-the-art systems.

Among the proposed extensions, HiGSFA is the most promising. HiGSFA incorporates the other extensions and yields the best results, making this approach competitive, scalable, and robust. Moreover, HiGSFA is a versatile algorithm, allowing new technical applications and further principled extensions.

Kurzfassung der Dissertation

In dieser Doktorarbeit werden neue Erweiterungen des *Slow Feature Analysis* (SFA) Algorithmus vorgestellt. Diese lösen effizient und genau überwachte Lernprobleme des maschinellen Lernens (machine learning), wie Klassifikation und Regression, auf hochdimensionalen Daten (z.B. Bilder). Probleme dieser Klasse wurden in jüngerer Zeit hauptsächlich mit *Convolutional Neural Networks* behandelt, ein Ansatz der hervorragende Ergebnisse erzielt. Dennoch sind neue Lösungsansätze wünschenswert, insbesondere wenn sie auf neuen Konzepten basieren und Designentscheidungen auf fundierten theoretischen Grundlagen beruhen.

SFA ist ein Algorithmus der ursprünglich für das unüberwachte Lernen von langsamen (d.h. zeitlich stabilen) Merkmalen entwickelt wurde. Vorteile von SFA beinhalten ein im Detail ausgearbeitetes theoretisches Fundament sowie eine Verbindung zu Lernprozessen in biologischen Systemen. Eine hierarchische Implementierung von SFA (*hierachical SFA*, kurz: HSFA) erlaubt es, SFA auf hochdimensionale Daten anzuwenden. Die im folgenden genannten SFA-Varianten erlauben es, tiefe hierarchische SFA Netzwerke zu erstellen und zu trainieren. Diese erreichen Ergebnisse vergleichbar zu anderen Methoden bzgl. Genauigkeit und Effizienz.

Graph-based SFA (GSFA) ist eine SFA Variante für überwachtes Lernen, welche das Konzept des *training graph* einführt: eine Struktur in der die Knoten des Graphen die zu lernenden Datenpunkte und Kanten die Verbindungen zwischen diesen Datenpunkten repräsentieren. Die Gewichte der Kanten können als die gewünschte Ähnlichkeit zwischen zwei verbundenen Datenpunkten interpretiert werden. Im Vergleich zu SFA löst GSFA ein allgemeineres Optimierungsproblem und berücksichtigt eine wesentlich höhere Anzahl von Verbindungen zwischen einzelnen Datenpunkten. Die Information der Klassenähnlichkeit ist durch die Stärke der Verbindungsgewichte repräsentiert. In dieser Arbeit werden verschiedene Trainingsgraphen zum Lösen von Regressions- und Klassifikationsproblemen vorgestellt. Die Leistungsfähigkeit von GSFA wird anhand eines Unterproblems der Gesichtsdetektion demonstriert.

Der *exact label learning* (ELL) Algorithmus erlaubt es Graphen zu trainieren, für welche die ermittelten langsamen Merkmale den Klassenzugehörigkeiten entsprechen, falls der Merkmalsraum als unbeschränkt vorausgesetzt wird. Im Gegensatz zu den bisher vorgestellten Graphen werden die Verbindungsgewichte von ELL daher exakt vordefiniert, wodurch die Klassifikationsleistung des Netzwerks verbessert wird. Darüber hinaus erlaubt ELL das Lernen mehrerer Klassen mit nur einem einzigen Netzwerk. Dies ist nicht nur effizienter als ein seperates Netzwerk für jede Klasse zu lernen, sondern führt außerdem zu robusteren Merkmalen.

Hierachical information-preserving GSFA (HiGSFA) vergrößert den durch das Netzwerk propagierenden Anteil der Information über die Klassenzugehörigkeit. HiGSFA extrahiert dazu zwei verschiedene Arten von Merkmalen: Langsame Merkmale, welche wie bisher die Langsamkeit maximieren, sowie Rekonstruktionsmerkmale, welche den Rekonstruktionsfehler minimieren und somit der Informationserhaltung dienen. Zur Evaluation der HiGSFA werden die Probleme der Alters- und Geschlechtsbestimmung und der ethnischen Zuordnung anhand von Porträtfotos herangezogen. Dabei erreicht HiGSFA im Falle des Ersteren einen durchschnittlichen Fehler von 3,5 Jahren und übertrifft mit diesem Ergebnis den bisherigen Stand der Technik.

Von den vorgestellten Varianten ist HiGSFA am vielversprechendsten. HiGSFA integriert die anderen aufgezählten Varianten und erzielt die besten Ergebnisse, was diesen Ansatz konkurrenzfähig, skalierbar und robust macht. Darüber hinaus bietet HiGSFA durch seine Vielseitigkeit die Möglichkeit neuer technischer Anwendungen sowie die Option auf weitere Varianten.

To my mother
María del Rosario Bañuelos Castañeda
and my father
Dr. Evodio Escalante Betancourt
with all my love and admiration.

Acknowledgements

Doing a PhD at the Institut für Neuroinformatik was a very enriching and joyful experience. It allowed me to learn from many great people in a professional, stimulating, and interdisciplinary environment. I would like to express my gratitude to all people at the institute who helped me to accomplish this PhD.

First of all, my deepest thank to my advisor *Prof. Dr. Laurenz Wiskott*, who trusted me with his most precious algorithm (SFA) and was an ideal mentor from a scientific as well as a personal standpoint. I have learned so much from him, directly and indirectly, and benefited from his analytical skills, mathematical intuition, and thoughtful questions. He encouraged in me a stronger desire for more fundamental approaches and formal scientific methods. I also thank him for having given me the chance to join the amazing field of machine learning.

I am especially grateful to *PD Dr. Rolf Würtz* for his great advice on different matters and for helping me during the PhD in several ways, including the acquisition of private databases, accessing publications, allowing me to use the face rendering software, and even providing me with a work place when my regular office needed renovation.

Several people helped me to improve this manuscript and previous publications that later gave rise to it. These people were fundamental for the quality of the text and for helping me to improve my writing skills: *Dr. Fabian Schönfeld, Mathias Tuma, Jonas Lins, Björn Weghenkel, Jan Melchior, Jun.-Prof. Dr. Tobias Glasmachers, Dr. Varun Kompella, Merlin Schüler*, and *Mathis Richter*. Thanks a lot guys for your valuable comments and suggestions.

Thanks to *Arno Berg*, who always provided me promptly with all hardware and software resources needed for this research. I also appreciate the work of the secretaries of the institute, especially *Frau Wille* and *Frau Schmidt*, who kindly helped me with complex paperwork and formalities.

I would like to also thank several friends that I had the pleasure to meet in Bochum, who kept me motivated to write this dissertation and helped me clear up my mind after exhausting work: *Kamal, Magdalena, Paulo, Diego, Emelyn, Osvaldo, Elvira, Dulce, Janaí, Grisell, Caroline, Luis, Alicia, Nathalie, Winona, Silke, Ricardo, Alexandra, Florian, Christian, Jens, Attila, Hafize, Denisa*, and *Mirela*. Also I would like to thank Mexican friends, who supported me from the distance: *Uriel, Javier*, and *Carina*. Thanks also to several outstanding people not named explicitly here.

This work was jointly supported by the *German Academic Exchange Service* (DAAD) and the *National Council of Science and Technology of Mexico* (CONACYT) through a scholarship. The *Research School* also supported this work through a yearly financial allowance and useful workshops.

Thanks!

Contents

Chapter 1

Introduction

Can deep networks be constructed and trained in a principled yet efficient and effective way? The field of machine learning has recently gained a lot of attention thanks to the exciting comeback of neural networks in the form of deep learning, i.e., neural networks that consist of several layers and are typically trained using backpropagation or other gradient-based algorithms (Bengio, 2009). Convolutional networks (CNNs) (e.g., LeCun et al., 1998; Krizhevsky et al., 2012) are currently the most successful type of deep architectures, providing superhuman performance in some applications, such as digit and traffic sign recognition (Schmidhuber, 2015). In spite of the popularity of CNNs, several questions regarding their theoretical foundations are still open, since their development has been frequently driven by performance or by technical ideas rather than by a scientific understanding (Zeiler and Fergus, 2014). This dissertation pursues a different approach to build and train deep networks, by exploring the idea of designing learning algorithms based on a few simple but strong principles. The fundamental principle employed in this work is the slowness principle (e.g., Hinton, 1989), which mandates the extraction of temporally stable features and is realized by means of the slow feature analysis (SFA) algorithm (Wiskott, 1998; Wiskott and Sejnowski, 2002).

Machine learning is concerned with the development and analysis of learning algorithms, that is, algorithms that adapt to their input data to accomplish a particular objective (Murphy, 2012). Supervised learning is a sub-field of machine learning where the training samples are accompanied by a (ground-truth) label. The objective is then to predict the label of new samples as accurately as possible according to a given error measure referred to as loss function. Labels are frequently categorical (classes) or numerical. In the first case the objective is classification and in the second case regression. Depending on the problem at hand, different loss functions are appropriate, for instance, a classification error or a mean squared error.

An important application of supervised learning is supervised image analysis, where the data samples are images and the labels encode some aspect of them. For example, labels may refer to the identity of the main object or sub-

ject in the image (object recognition, face recognition), or the numerical value of a particular attribute (e.g., object position, pose, age). Computer vision researchers have proposed many advanced algorithms for supervised learning from images (e.g., Szeliski, 2010; Fu et al., 2010). However, computer vision algorithms are frequently optimized for concrete problems and datasets, and tend to be problem specific. Thus, more general approaches are desirable.

1.1 Principled Supervised Learning with SFA

In the context of animal perception, a key observation is that individual sensory inputs change relatively quickly compared to information derived from the sensory data that encodes useful higher-level aspects of the environment (e.g., the position of a moth changes slower than the quickly changing neural activations in the retina of a frog observing it). The slowness principle is based on this observation and requires, consequently, the extraction of slow features. It has probably first been formulated by Hinton (1989). The first closed-form algorithm for computing slow features has been developed by Wiskott and is referred to as slow feature analysis (SFA) (Wiskott, 1998; Wiskott and Sejnowski, 2002) (see Chapter 2).

It has been shown that SFA learns responses similar to those of specific types of neurons in the primate visual system (e.g. Berkes and Wiskott, 2005) and hippocampus (e.g. Franzius et al., 2007) when trained in a purely unsupervised manner. SFA has many possible applications, particularly for the extraction of hidden driving forces, learning of invariant features, and blind source separation. Although SFA is originally unsupervised, it also has applications for supervised learning (see Section 2.6), making it a versatile and remarkably problem-independent algorithm.

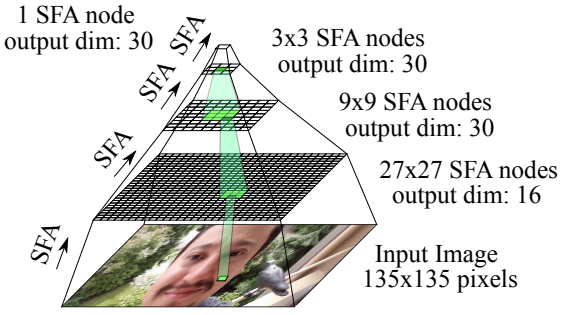


Figure 1.1: A hierarchical SFA network with 4 layers and no receptive field overlap used for gender and age estimation from artificial face images (2D data). Nodes have a fan-in of 5x5 pixels in layer 1 and 3x3 nodes in the next layers. The terms receptive field, overlap, and fan-in are defined in Section 2.4.1. The input to one node in layers 1, 2 and 3 is highlighted.

[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

Due to their size, images are frequently considered high-dimensional. The direct application of SFA on high-dimensional data, including images, is less efficient. However, one can resort to a divide-and-conquer approach for the computation of slow features called hierarchical SFA (HSFA) to efficiently deal with such high-dimensional data (e.g., Franzius et al., 2011). HSFA results in a multi-layer hierarchical network, where each layer is composed of various possibly independent instances of SFA, called SFA nodes (see Figure 1.1). The input image information is propagated from the bottom to the top layer in a feed-forward manner. Careful network design allows training and execution with linear complexity w.r.t. the input dimensionality and number of samples (scalable learning).

This thesis investigates methods for using SFA and HSFA for the solution of classification and regression problems. In an abstract sense, the approach consists in transforming a supervised learning problem into an unsupervised learning one, where SFA or a variant of it can be applied. Various extensions to SFA and HSFA are proposed, which yield improved performance in terms of feature slowness, label estimation accuracy, and generalization to unseen data. These extensions (see Section 1.7) include graph-based SFA (GSFA), exact label learning (ELL) and hierarchical information-preserving GSFA (HiGSFA), with HiGSFA being the most promising extension.

HiGSFA encompasses the principles and heuristics used by the other extensions, see Table 1.1, inheriting most advantages of SFA and resulting in a general and efficient learning algorithm with a strong mathematical foundation. HiGSFA is capable of extracting useful features from the raw pixel data that increase in complexity, selectivity, and abstraction level as the data is processed through the network. The label-predictive information is concentrated in a few output features that are mostly invariant to other aspects of the input data. Note that these properties are similar to those of other neural networks. However, HiGSFA is in fact considerably different from existing methods based on gradient descent, including the highly successful CNNs: The training method of HiGSFA is bottom-up, it uses other types of nonlinearities, is not convolutional, uses neither backpropagation nor max pooling, and the features of each node fulfill a local optimality criteria. Therefore, HiGSFA is a promising new approach.

The possible applications of this research to image analysis are extensive, ranging from demographics analysis, quality control, image-based diagnosis, and driver assistance to image tagging. A strong label estimation accuracy is crucial for these applications. Therefore, more general and accurate algorithms are valuable not only for machine learning and computer vision researchers but for developers of image analysis systems across multiple disciplines.

1.2 General Objective

The general objective of this thesis is to analyze the viability of using or extending SFA to solve supervised learning tasks on images via *supervised* feature

Principle or heuristic	Implemented through	Described in
Slowness principle	SFA	Chapter 2
Exploitation of label similarities through slowness	GSFA, training graphs for supervised learning, ELL method	Chapters 3 and 4
Spatial localization of features, divide and conquer approach	Hierarchical processing	Section 2.4 and Chapter 5
Nonlinearities that are robust to outliers	Normalized or saturating nonlinear expansions	Sections 3.5.1 and 5.5.4
Information preservation	Minimization of a reconstruction error, PCA, HiGSFA	Chapter 5
Multiple information channels	Multi-label learning combining efficient training graphs, HiGSFA	Chapters 4 and 5

Table 1.1: Principles, heuristics, and ideas considered in this thesis and the base methods or algorithms used to exploit them.

extraction (and not simply unsupervised feature extraction), and in particular such tasks that involve face images, such as the estimation of age, gender, race, scale and pose.

In order to achieve this goal, new methods are developed to exploit the label information, improve the label estimation accuracy, and minimize the computational requirements. To develop such methods, different problems of increasing difficulty level are addressed, ranging from gender and age estimation on artificial face images generated using 3D models to problems on real photographs, such as face detection, digit and traffic sign recognition, as well as age, gender, and race estimation.

1.3 Questions Addressed

This work attempts to contribute to the solution to the following questions:

- Is it possible to build competitive supervised learning algorithms based on a small set of simple but strong principles? What kinds of principles or heuristics are most valuable in theory and practice? Can such a general approach outperform problem-specific algorithms?
- Is it possible to obtain good label estimation accuracies with SFA on non-artificial data other than the MNIST dataset for handwritten digit recognition[1]? What attributes can be estimated from face photographs? Can SFA be used for face recognition?
- How can one use SFA to solve classification and regression problems effectively? Can one transform a supervised learning problem into an equiva-

[1]When this PhD project began, digit recognition was apparently the only supervised learning application on *real data* in which SFA had provided competitive results (Berkes, 2005a).

lent slow feature extraction problem?

- Regarding hierarchical processing, what are the concrete gains, computational and otherwise, of using HSFA? What are the main limitations and can they be overcome?
- From a practical point of view there is a trade-off between efficiency and model complexity. Can one design layer and network structures that are efficient and still yield useful features? What types of nonlinear expansions provide rich feature spaces and good generalization?

1.4 Hypotheses

The main hypothesis of this work maintains that it is possible to use a small set of principles and heuristics to develop general supervised learning algorithms with good efficiency and label estimation accuracy. That is, a few principles and heuristics can be useful to learn a mapping from samples to labels in a problem-independent way and still obtain good performance.

The hypothesis above does not contradict "no-free-lunch" theorems, because the goal is not to solve all possible problems, but only those arising from real-world applications where labels have a natural interpretation. Therefore, different datasets and labels may be constrained by, for example, physical and biological restrictions and might share similar intrinsic regularities at a high level.

Based on several theoretical and experimental results (e.g., Franzius et al., 2007; Schoenfeld and Wiskott, 2015), this thesis also assumes that the slowness principle is a fundamental learning principle well suitable for the task above.

Even though HSFA is not globally optimal, in practice it has allowed to extract useful features with good results under much less computational requirements than the direct application of SFA, and is thus particularly attractive for handling high-dimensional data. Therefore, it is also hypothesized that feature extraction with HSFA (and extensions) can be robust enough to overcome (to some degree) variations present in real-life face images, such as different poses, subject identity, hair styles, and lighting conditions.

1.5 Scope and Limitations

The scope of this work has been restricted as follows. According to the hypotheses above, all developed algorithms should extend or be related to SFA.

The experiments of this thesis are limited to images. However, the proposed extensions may also be applied to other types of data if they can be vectorized, such as audio fragments and voxels. For simplicity, and due to the availability of large databases, the focus is on face images, except for the classification algorithms, which are also tested on more classical databases (i.e., digit and traffic

sign recognition) because face recognition may strongly benefit from problem-specific methods (e.g., explicit detection of fiducial points). However, the extensions are application independent and may be applied to different problems and types of images besides the ones mentioned above.

One application area of SFA is reinforcement learning (Wilbert, 2012), where SFA is used as a general feature extraction algorithm. Although this work does not address reinforcement learning directly, the proposed extensions might also be beneficial in this area.

Many implementations for deep learning have employed graphics cards or special hardware (Schmidhuber, 2015). To avoid premature optimization, the proposed extensions do not yet exploit this type of resources, but they benefit from conventional multi-core architectures (multithreading) and have been optimized at the mathematical and algorithmic level.

1.6 Methods

The approach uses methods frequently employed in machine learning: linear algebra, spectral analysis, algorithm analysis and design, loss functions, asymptotic computational and memory complexity, and basic probability, calculus, and information theory.

Although this work mainly belongs to the field of machine learning, it profits from ideas originating from two other fields. From neuroscience this work adopts various heuristics and principles inspired by properties of the mammal visual system and hippocampus, namely hierarchical processing and feature locality. From theoretical biology this work borrows SFA and the slowness principle.

The use of concepts from neuroscience and theoretical biology does not lessen the rigor of the approach from a machine learning perspective, because this work does not attempt to simulate or replicate any specifics of the brain or to provide biologically-feasible algorithms (though they are not discarded). Moreover, the slowness principle (implemented via HSFA) is not adopted blindly but rather critically assessed and modified for the challenges of supervised learning.

1.7 Contributions

The most representative precursor of this research is the work of Franzius et al. (2011) on object recognition and pose estimation with HSFA, in which HSFA is trained in an unsupervised fashion on random sequences of objects (either artificial fish or clusters of spheres) where the pose and identity of the objects changes slowly. The extracted slow features are post-processed by a supervised algorithm to compute the final label estimations. Their system is able to successfully estimate identity, position, size, and one or more angles of the object.

Some shortcomings of the work above are the artificial nature of the input images, the small number of objects used, the plain background, and that 512 slow features are used in the supervised step (indicating poor feature selectivity).

Moreover, for classification, the slow features have to be post-processed with Fisher discriminant analysis.

This thesis advances the approach above by introducing new extensions to SFA and HSFA that explicitly use the label information to provide a more compact feature representation (less than 10 features frequently suffice for best accuracy), yield higher label-estimation accuracy, and can be applied to real photographs with (to some extent) cluttered backgrounds. The extensions are especially beneficial when used hierarchically and applied to high-dimensional data. A separate chapter is devoted to each one of the three major contributions of this thesis, which are briefly outlined below.

1.7.1 Graph-Based SFA (GSFA)

GSFA is the first proposed extension to SFA. It has been developed to solve classification and regression problems, a goal that is achieved through features that are more label predictive. The training data of SFA is a temporal sequence of samples, where transitions occur between consecutive samples. GSFA generalizes and replaces such training data by a graph structure called training graph, where each vertex is a sample, the edges connect arbitrary pairs of samples, and the vertices and edges are weighted. The label information (or most of it) is encoded in this structure by making connections between samples with similar labels stronger than those between samples with dissimilar labels. The structure of the training graphs is crucial. Given the same samples, different connections allow GSFA to become sensitive to specific labels (e.g., object position, size, class) and invariant to other aspects of the data.

Several training graphs for GSFA are proposed, such as the serial training graph, which is useful for regression. The serial graph has $\mathcal{O}(N^2)$ transitions (where N is the number of samples), that is, many more than the $N-1$ transitions considered by standard SFA. However, training GSFA (+ serial graph) has the same asymptotic computational complexity as training SFA.

GSFA is evaluated on a subproblem of the face detection problem consisting in the estimation of the horizontal position of a face (x-pos) contained in an image patch. The images originate from several databases to increase image variability and are normalized to grayscale and 128×128 pixels.

The x-pos problem is successfully tackled with an 11-layer hierarchical GSFA (HGSFA) network. The features learned with HGSFA allow for more accurate label estimations than those computed with HSFA. Moreover, the label information is concentrated in considerably fewer features than in previous approaches (e.g., only 5 features are necessary to achieve top label estimation accuracy, indicating that HGSFA indeed finds the slowest hidden parameters of the data).

1.7.2 Exact Label Learning (ELL)

The ELL method is motivated by the following question: Can one construct a training graph such that the slowest feature computed by GSFA is equal to

the label we want to learn? Pre-defined graphs, such as the serial one, provide promising results for the solution of regression problems. However, such graphs take only the rank of the labels into account (i.e., the position or index of the samples when ordered by increasing label) and not their exact value. The consideration of the exact numerical value of the labels makes it possible to improve the estimation accuracy.

The ELL method removes the dependency on a final regression step that maps the slow features to labels, providing an end-to-end method (from raw pixels to label estimation). The ELL method allows us to construct a training graph, such that the first optimal free response (i.e., the slowest possible feature that can be extracted from a given training graph without being restricted by the training samples or the feature space) is equal to a normalized version of the label. More precisely, under unrestricted conditions, when GSFA or HGSFA are trained with an ELL graph, the solution to the regression problem is given by the slowest feature extracted. It is only necessary to correct its sign globally (since slowness is invariant to feature polarity) and reverse the label normalization (since labels do not necessarily have unit variance and zero mean in contrast to slow features).

One standalone contribution (part of ELL) is a method for the analysis of training graphs. With such a method we can compute the optimal free responses of a given graph, resulting in an alternative representation of the graph that shows an important aspect of its structure.

The strongest advantage of the ELL method is that it allows us to learn multiple labels simultaneously (e.g., face position, size, pose, age, gender). Furthermore, the theory of the ELL method also shows us how to combine pre-defined graphs to obtain more accurate and efficient estimations.

1.7.3 Hierarchical Information-Preserving GSFA (HiGSFA)

Experiments and a theoretical analysis of hierarchical GSFA (HGSFA) show the advantages of hierarchical processing with GSFA, including efficiency, good label estimations, and a rich feature space. However, it is shown that HGSFA suffers from a drawback that I term here *unnecessary information loss*, where nodes of the network discard information that is not slow locally, but that would have resulted in slow features when combined with information from other nodes higher in the network. The goal of HiGSFA is to counteract this drawback by adopting a secondary optimization objective called information preservation, which complements the slowness-maximization objective. In practice, information preservation is implemented as the minimization of a reconstruction error. Although this combination of objectives might appear counter-intuitive, experiments show that HiGSFA outperforms HGSFA in feature slowness, label estimation accuracy, and input reconstruction with the same asymptotic computational complexity.

HiGSFA is evaluated on the challenging problem of age estimation from facial photographs of the MORPH-II database. The input images are nor-

malized to gray-scale and size 96×96. A 10-layer HiGSFA network is used. Best performance is obtained when not only age, but also gender and race labels are learned simultaneously, which is possible by combining three different pre-defined graphs, one for each label. The approach yields a mean absolute error (MAE) of 3.50 years, resulting in state-of-the-art performance for this task and an improvement over an MAE of 3.63 years using a multi-scale convolutional neural network (Yi et al., 2015) and 3.92 years using bio-inspired features+rKCCA+support vector machine (Guo and Mu, 2014).

Among the extensions proposed in this thesis, HiGSFA is the most promising one, because it encompasses all the principles and heuristics considered by the other proposed extensions and provides the most accurate results. It constitutes a powerful, scalable, principled, and end-to-end method for supervised dimensionality reduction and feature extraction.

1.8 Thesis Structure

The remainder of the thesis is structured as follows. First, SFA and several existing applications of it are introduced in Chapter 2. Then, the GSFA algorithm and optimization problem are proposed in Chapter 3. Afterwards, the ELL method is proposed in Chapter 4, including a method for the analysis of training graphs and the computation of compact discriminative features. In Chapter 5, the advantages and drawbacks of hierarchical processing with HSFA and HGSFA are analyzed, and the HiGSFA algorithm is proposed. The thesis is closed in Chapter 6 with a general discussion. In addition, the software framework used to implement and evaluate all the extensions, called Cuicuilco, is briefly described in Appendix A.

Chapter 2

Standard SFA

In this chapter, the general learning principle behind slow feature analysis (SFA) called the *slowness principle* is introduced. Furthermore, the SFA algorithm and its optimization problem are reviewed, and the concepts of supervised learning with SFA and hierarchical SFA networks (HSFA) are briefly described. Additionally, several concrete applications of SFA are described[1].

2.1 The Slowness Principle and SFA

The nervous system of humans (and mammals in general) appears to process sensory information (e.g., visual stimuli) in a simple and straightforward way. However, this computation is a complex task of crucial importance for any interaction with the environment. Consider the visual perception of a driver observing pedestrians walking on the street: As the car moves forward, the activations of the individual receptors in his retina typically change quickly, and are especially sensitive to eye movement and variations in the position of objects and pedestrians in his field of view. However, all pertinent information, including the position and posture of the pedestrians, can easily be distinguished by the driver. In general, useful high-level information contained in the perception of the environment frequently changes on a much slower scale than the inputs arriving at individual receptors. This observation inspires the slowness principle, which explicitly requires the extraction of slow features.

This principle has probably first been formulated by Hinton (1989), and online learning rules have been developed by Földiák (1991) and Mitchison (1991). The first closed-form algorithm has been developed by Wiskott and is referred to as slow feature analysis (SFA) (Wiskott, 1998; Wiskott and Sejnowski, 2002).

The concise formulation of the SFA optimization problem (see Section 2.2) permits an extended mathematical treatment that facilitates a deep analytical understanding of its properties (Wiskott, 2003a; Franzius et al., 2007; Sprekeler and Wiskott, 2011). The SFA algorithm (see Section 2.3) is guaranteed to

[1]This chapter is in part an edited version of (Escalante-B. and Wiskott, 2013); the information here has been updated to include newer extensions to SFA.

find an optimal solution within the considered function space (e.g., all linear or quadratic functions). It was initially developed for learning invariances in a model of the primate visual system (Wiskott and Sejnowski, 2002; Franzius et al., 2011). Subsequently, Berkes and Wiskott (2005) used it for learning responses similar to those of complex cells in primary visual cortex, and Franzius et al. (2007) for learning responses similar to place cells in the hippocampus. In recent years, the number of technical applications of SFA has been continually increasing (Escalante-B. and Wiskott, 2012); some of them are briefly described in Section 2.6.

2.2 Standard SFA Optimization Problem

The SFA optimization problem can be stated as follows (Wiskott, 1998; Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005): Given an I-dimensional input signal $\mathbf{x}(t) = (x_1(t), \ldots, x_I(t))^T$, where $t \in \mathbb{R}$, and an output dimensionality J, find an instantaneous vectorial function $\mathbf{g} : \mathbb{R}^I \to \mathbb{R}^J$ within a function space $\mathcal{F}$, that is, $\mathbf{g}(\mathbf{x}(t)) = (g_1(\mathbf{x}(t)), \ldots, g_J(\mathbf{x}(t)))^T$, such that for each component $y_j(t) \stackrel{\text{def}}{=} g_j(\mathbf{x}(t))$ of the output signal $\mathbf{y}(t) \stackrel{\text{def}}{=} \mathbf{g}(\mathbf{x}(t))$, for $1 \le j \le J$, the objective function

$$\Delta(y_j) \stackrel{\text{def}}{=} \langle \dot{y}_j(t)^2 \rangle_t \text{ is minimal} \quad \textbf{(delta value)} \tag{1}$$

under the constraints

$$\langle y_j(t) \rangle_t = 0 \textbf{ (zero mean),} \tag{2}$$

$$\langle y_j(t)^2 \rangle_t = 1 \textbf{ (unit variance),} \tag{3}$$

$$\langle y_j(t) y_{j'}(t) \rangle_t = 0, \forall j' < j \textbf{ (decorrelation and order).} \tag{4}$$

The delta value $\Delta(y_j)$ is defined as the time average ($\langle \cdot \rangle_t$) of the squared derivative of y_j and is therefore a measure of the slowness (or rather fastness) of the signal. The constraints (2–4) require that the output signals are normalized, not constant, and represent different aspects of the input signal. The problem can be solved iteratively beginning with y_1 (the slowest extracted feature) and finishing with y_J. An algorithm for extracting y_1 to y_J is described in the next section. Due to constraint (4), the delta values are ordered, that is, $\Delta(y_1) \le \Delta(y_2) \le \cdots \le \Delta(y_J)$. See Figure 2.1 for an illustrative example.

The term instantaneous is used to emphasize that the function is only sensitive to the current input sample and is (conditionally) independent of previous and future samples and the time variable (even though the original input signal $\mathbf{x}(t)$ is a function of time). This rules out averaging the data over time to compute slow features.

In practice, the function $\mathbf{g}$ is usually restricted to a finite-dimensional space $\mathcal{F}$, for example, to all polynomial functions of degree 1, 2, or 3. Highly complex function spaces $\mathcal{F}$ should be avoided because they result in overfitting. In extreme cases one obtains features such as those in Figure 2.1 (right) even when the hidden parameters implicit in the input data lack such an ideal structure.

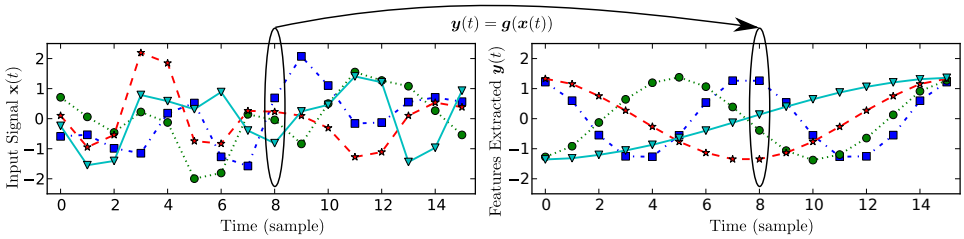


Figure 2.1: Illustrative example of feature extraction from a 10-dimensional discrete-time input signal. Four arbitrary components of the input (left) and the four slowest outputs (right) are shown. Notice that feature extraction is an instantaneous operation, even though the outputs are slow over time. This example was designed such that the features extracted are the slowest ones theoretically possible independently of the data and feature space (optimal free responses). In particular, in continuous time the slowest possible feature is a normalized half-period of a cosine function. Such a feature is similar to the discrete-time feature plotted in light blue. The next optimal free responses are cosine functions of increasing frequency. The input has been generated by linearly shuffling different optimal free responses, an operation that is easily reversed by linear SFA.

[Figure reproduced from (Escalante-B. and Wiskott, 2012), with permission.]

This overfitting problem is then evident when one extracts features from test data that lack such a structure, see Figure 2.2. An unrestricted function space is, however, useful for theoretical analyses (e.g., Wiskott, 2003a) because it is general and mathematically convenient.

2.3 Standard Linear SFA Algorithm

The SFA algorithm is typically nonlinear. Although kernelized versions have been proposed (Bray and Martinez, 2003; Vollgraf and Obermayer, 2006; Böhmer et al., 2012), it is usually implemented more directly with a nonlinear expansion of the input data followed by linear SFA (linear in the expanded space).

In this section, the standard linear SFA algorithm (Wiskott and Sejnowski, 2002) is recalled, in which $\mathcal{F}$ is the space of all linear functions. Discrete time, $t \in \mathbb{N}$, is used for the application of the algorithm to real data. The objective function and the constraints are thus adapted to discrete time. The input is then a single training signal (i.e., a sequence of N samples) $\mathbf{x}(t)$, where $1 \leq t \leq N$, and the time derivative of $\mathbf{x}(t)$ is usually approximated by a sequence of differences of consecutive samples: $\dot{\mathbf{x}}(t) \stackrel{\text{def}}{\approx} \mathbf{x}(t+1) - \mathbf{x}(t)$, for $1 \leq t \leq N-1$. Thus, it is assumed without lost of generality that $\Delta t = 1$.

The output components take the form $g_j(\mathbf{x}) = \mathbf{w}_j^T(\mathbf{x} - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}} \stackrel{\text{def}}{=}$

$\frac{1}{N}\sum_{t=1}^{N}\mathbf{x}(t)$ is the average sample, which is subtracted to ensure that the output has zero-mean to conform with (2). Thus, in the linear case, the SFA problem is reduced to finding an optimal set of weight vectors $\{\mathbf{w}_j\}$ under constraints (2–4) and can be solved by linear algebra methods, as shown below.

The SFA algorithm computes the second-order statistics of the training data. Such information is ideally described by the covariance matrix of the data, but since the number of samples is finite, the covariance matrix is approximated by the sample covariance matrix

$$\mathbf{C} = \frac{1}{N-1}\sum_{t=1}^{N}(\mathbf{x}(t)-\bar{\mathbf{x}})(\mathbf{x}(t)-\bar{\mathbf{x}})^T\,,$$

and the derivative second-moment matrix is approximated as

$$\dot{\mathbf{C}} = \frac{1}{N-1}\sum_{t=1}^{N-1}(\mathbf{x}(t+1)-\mathbf{x}(t))(\mathbf{x}(t+1)-\mathbf{x}(t))^T\,.$$

Then, a sphered signal $\mathbf{z} \stackrel{\text{def}}{=} \mathbf{S}^T\mathbf{x}$ is computed, such that $\mathbf{S}^T\mathbf{C}\mathbf{S} = \mathbf{I}$ for a sphering matrix $\mathbf{S}$, i.e., the sample covariance matrix of $\mathbf{z}$ is the identity matrix. Afterwards, the J directions of least variance in the derivative signal $\dot{\mathbf{z}} \stackrel{\text{def}}{=} \mathbf{S}^T\dot{\mathbf{x}}$ of the sphered data are found and represented by an $I \times J$ rotation matrix $\mathbf{R}$. Such directions are found by solving $\mathbf{R}^T\dot{\mathbf{C}}_z\mathbf{R} = \mathbf{\Lambda}$ for $\mathbf{R}$ and $\mathbf{\Lambda}$ (i.e., the eigenvalues and orthonormal eigenvectors of $\dot{\mathbf{C}}_z$), where $\dot{\mathbf{C}}_z \stackrel{\text{def}}{=} \langle\dot{\mathbf{z}}\dot{\mathbf{z}}^T\rangle_t$ and $\mathbf{\Lambda}$ is a diagonal matrix with diagonal elements $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_J$. Finally the algorithm returns the weight matrix $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_J)$, defined as $\mathbf{W} = \mathbf{S}\mathbf{R}$, the extracted features $\mathbf{y} = \mathbf{W}^T(\mathbf{x}-\bar{\mathbf{x}})$, and $\Delta(y_j) = \lambda_j$, for $1 \le j \le J$.

The linear SFA algorithm is guaranteed to find an optimal solution to the optimization problem (1–4) in the linear function space, e.g., the first component extracted is the slowest possible linear feature. A more detailed description of the linear SFA algorithm is provided by Wiskott and Sejnowski (2002).

The complexity of the linear SFA algorithm is $\mathcal{O}(NI^2 + I^3)$ where N is the number of samples and I is the input dimensionality (possibly after a nonlinear expansion), thus for high-dimensional data standard SFA is not feasible[2]. The term NI^2 is due to the computation of the covariance and second moment matrices, whereas the term I^3 is due to the eigenvalue decompositions. In practice, the complexity of eigenvalue decomposition is generally accepted as $\mathcal{O}(I^3)$, however, this is not a tight bound. It has been suggested (e.g., Poloni, 2015) that the problem might be reduced in theory to matrix multiplication, yielding a tighter complexity $\mathcal{O}(I^\alpha)$, for some $2 < \alpha < 2.376$. In practice, the total running time of SFA is comparable to that of PCA, even though SFA also takes into account the temporal structure of the data.

[2]The problem is still feasible if N is small enough so that one might apply singular value decomposition methods. However, a small number of samples $N < I$ usually results in pronounced overfitting.

2.4 Hierarchical SFA (HSFA)

High-level information that occurs in real-life applications can usually be encoded as different types of high-dimensional data, including images, video, and voxels. However, as explained above, the direct application of SFA on high-dimensional data is inefficient.

Hierarchical processing (e.g., Franzius et al., 2011) is an efficient divide-and-conquer strategy for the extraction of slow features when the original data is high dimensional. For example, if the input dimension I is large, computing $\mathrm{SFA}(\mathbf{x}(t))$ would be infeasible[3]. One effective solution is to divide the input data spatially into k lower-dimensional signals $\mathbf{x}^{(1)}(t), \ldots, \mathbf{x}^{(k)}(t)$ of dimensionality $I' \stackrel{\text{def}}{=} I/k$. Then, one can extract local slow features $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$ from each signal: $\mathbf{y}^{(1)}(t) \stackrel{\text{def}}{=} \mathrm{SFA}^{(1)}(\mathbf{x}^{(1)}(t))$, $\mathbf{y}^{(2)}(t) \stackrel{\text{def}}{=} \mathrm{SFA}^{(2)}(\mathbf{x}^{(2)}(t))$, $\ldots, \mathbf{y}^{(k)}(t) \stackrel{\text{def}}{=} \mathrm{SFA}^{(k)}(\mathbf{x}^{(k)}(t))$. A concrete instance of SFA trained with a particular training data is denoted by $\mathrm{SFA}^{(\cdot)}$. Different SFA instances are also referred to as SFA nodes, especially in the context of hierarchical SFA networks described as directed graphs. The nodes above are called local because their input is only a fraction of the original input. Each of them extracts $J' < I'$ slow features (i.e., the output dimensionality of the nodes must be smaller than their input dimensionality). Afterwards, a new SFA node $\mathrm{SFA}^{(\mathrm{top})}$ in a new layer extracts global slow features from the concatenation of the local slow features previously computed: $\mathbf{y}^{(\mathrm{top})}(t) \stackrel{\text{def}}{=} \mathrm{SFA}^{(\mathrm{top})}\left(\mathbf{y}^{(1)}(t)|\cdots|\mathbf{y}^{(k)}(t)\right)$, where $\cdot|\cdot$ is the concatenation operation in space (not in time). A proper choice of J' and k can ensure that the computation of $\mathbf{y}^{(\mathrm{top})}(t)$ is feasible.

If the input dimensionality I' of the local nodes $\mathrm{SFA}^{(1)}, \ldots, \mathrm{SFA}^{(k)}$ is still too large, one can repeat the strategy above to each one of these nodes. Following such an approach recursively results in a multi-layer hierarchical network (e.g., see Figure 5.3 on page 100). Due to information loss before the top node and the change of the feature space, hierarchical SFA does not guarantee globally optimal slow features anymore. However it has shown to be effective in many practical experiments, in part because low-level features are spatially localized in most real data.

Interestingly, hierarchical processing can also be seen as a regularization method, as shown in Figure 2.2, leading to better generalization. An additional advantage is that the nonlinearity accumulates across layers, so that even when using simple expansions the network as a whole can realize a complex nonlinearity (Escalante-B. and Wiskott, 2011).

2.4.1 Previous Work on HSFA and Terminology

The first example of HSFA was given in the paper that first introduced the SFA algorithm (Wiskott, 1998), where it was used as a model of the visual system

[3]Even linear SFA becomes infeasible if I is sufficiently large. Therefore, the usefulness of hierarchical processing is not limited to nonlinear SFA.

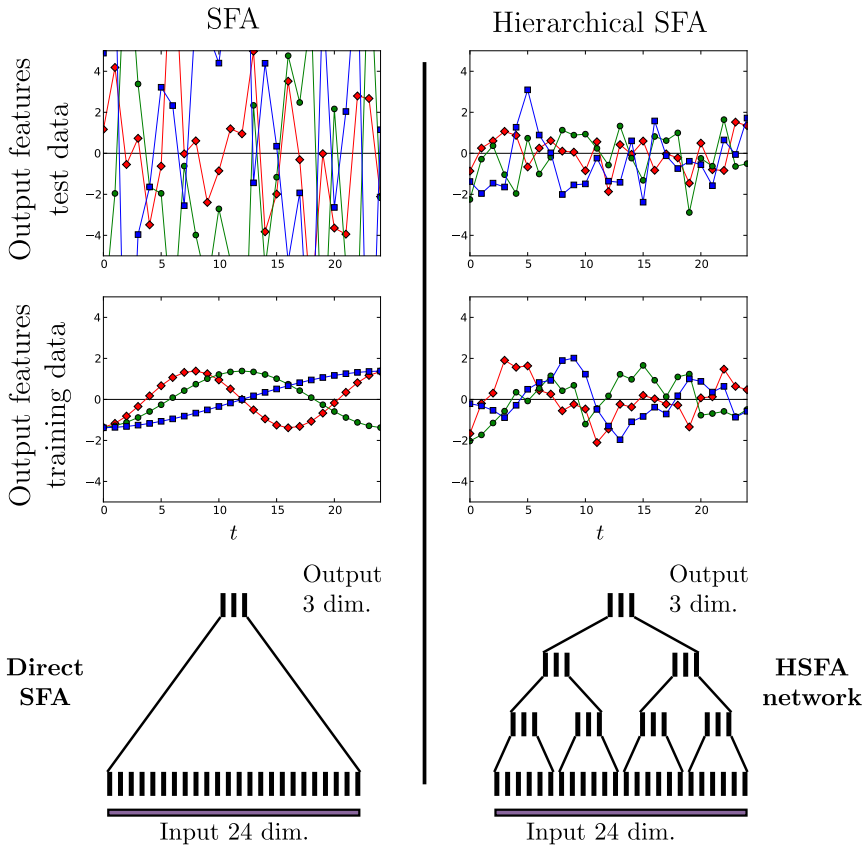


Figure 2.2: Example of how hierarchical SFA (HSFA) is more robust against overfitting than direct SFA. Useless data consisting of 25 random i.i.d. samples is processed by linear SFA and linear HSFA. Both algorithms reduce the dimensionality from 24 to 3 dimensions. Even though the training data is random, the direct application of SFA extracts the slowest features theoretically possible (optimal free responses), which is possible due to the number of dimensions and samples, permitting extreme overfitting. However, it fails to provide consistent features for test data (e.g., standard deviations $\sigma_{\text{training}} = 1.0$ vs. $\sigma_{\text{test}} = 6.5$), indicating lack of generalization. In contrast, HSFA extracts much more consistent features (e.g., standard deviations $\sigma_{\text{training}} = 1.0$ vs. $\sigma_{\text{test}} = 1.18$) resulting in less overfitting. Counter-intuitively, this result holds even though the HSFA network has $7 \times 6 \times 3 = 126$ free parameters, many more than the $24 \times 3 = 72$ free parameters of direct SFA.

[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

for learning invariant representations.

Franzius et al. (2007) have used HSFA to learn invariant features from the simulated view of a rat that moves inside a box. In conjunction with a sparseness post-processing step, the extracted features are similar to the responses of place cells in the hippocampus. Various contributions have continued this biologically-

inspired approach (e.g. Schoenfeld and Wiskott, 2015).

Other systems have especially profited from its computational efficiency compared to direct SFA. Franzius et al. (2011) have used HSFA for object recognition from images and to estimate pose parameters of single objects moving and rotating over a plain background. Escalante-B. and Wiskott (2013) have used a hierarchical graph-based SFA (HGSFA) network with 11 layers to accurately find the horizontal position of faces in photographs, which is a subproblem of face detection. Further experiments include the estimation of the vertical position, size, and in-plane angle of faces.

The structure of HSFA networks usually follows the structure of the data. For instance, networks for data in one dimension (e.g., audio data represented as fixed length vectors) typically have a one-dimensional structure (e.g., Figure 5.3), and networks for data in two dimensions (e.g., images) have a two-dimensional structure (e.g., Figure 1.1). This idea extends to voxel data in three dimensions, and beyond.

For simplicity, the input data is referred to as layer 0. Important parameters that define the structure of a network include: (a) The *output dimensionality* of a node. (b) The *fan-in* of a node, which is the number of nodes (or data elements) in a previous layer that feed into it. (c) The *receptive field* of a node, which refers to all the elements of the input data that directly or indirectly (i.e., via other nodes) provide input to the node. (d) The *stride* of a layer, which tells how far apart the inputs to adjacent nodes in a layer are. If the stride is smaller than the fan-in, then at least one node in the previous layer will feed two or more nodes in the current layer. This is called *receptive field overlap*. The analysis of hierarchical networks, their advantages, and their limitations is detailed in Chapter 5.

2.5 SFA for Supervised Learning

SFA, as described so far, learns features that are implicit in the input signal. It extracts the slowly varying aspects and becomes less sensitive or even fully invariant to quickly changing aspects. However, in many learning problems the input data does not take the form of a temporal sequence (i.e., a time series), and the features to be learned are provided explicitly as labels along with the input samples, resulting in a supervised learning problem.

Multiple approaches allow us to apply SFA in such cases. As an example, consider a number of face images for which the age of the person is known and the task is to automatically extract the age given the images only. A large part of the problem can be solved by SFA if one simply orders the images by increasing age and presents this to SFA. With this approach, age becomes the most slowly varying feature and SFA will naturally extract it—although not age directly but rather a feature that is monotonically related to it. In such an approach, a final step mapping the first SFA output(s) to the real age label is therefore required. This method for using SFA to solve regression problems is

called *sample reordering* and is further addressed in Section 3.4.1.

If the labels indicate classes, such as subject identity, one can order the images by class and proceed as above. However, it is more efficient to give up the linear temporal sequence and instead use the graph structures employed by *graph-based SFA* (GSFA). For classification, one would connect all samples of the same identity with each other and make no connection between samples of different identities. In this case the graph separates into fully connected subgraphs. The GSFA objective (see Chapter 3) takes into account all connections and minimizes the output differences between connected samples. The few first ideal GSFA outputs would therefore be constant for any given identity and different for different identities. Again, a final supervised step would be required to map GSFA output values to identity labels. A formalization and in-depth study of this idea is presented in Chapter 3. GSFA can be used to learn (categorical) class labels as well as numerical labels, thus being useful for both classification and regression.

2.6 Technical Applications of SFA

Although SFA was developed for computational neuroscience, it has turned out to be a versatile algorithm for a range of technical applications. In this section, several examples of current applications of SFA (and its extensions) are described. The goal is to provide a broad overview of the current practical problems addressed with SFA rather than to describe the concrete systems or how SFA is used in them. Technical applications of SFA have been previously summarized by Escalante-B. and Wiskott (2012), and this text improves on this previously published review by adding new applications that have appeared afterwards.

2.6.1 General-Purpose Feature Extraction

Given some temporal data, it is possible to apply SFA with the hope that the extracted slow features contain useful information that might solve or simplify a particular task.

An early application of SFA is the *estimation of driving forces* (Wiskott, 2003b). A driving force is a parameter that changes over time and has a direct influence on the underlying, possibly complex, dynamics of a system. The assumption is that such driving forces might change much slower than the signals generated by the dynamic system. Therefore, such driving forces may be found by SFA. To facilitate this, the input sequence $\mathbf{x}(t)$ can be pre-processed by concatenating k consecutive input samples: $\mathbf{x}^{\mathrm{emb}}(t) \stackrel{\mathrm{def}}{=} \big(\mathbf{x}(t), \mathbf{x}(t+1), \ldots, \mathbf{x}(t+k-1)\big)$, a procedure that is called time embedding. The embedded input $\mathbf{x}^{\mathrm{emb}}(t)$ is then provided to SFA. An example of this approach is the estimation of the parameters of a tent map (Wiskott, 2003b). The approach might also be applied to real data, for example, to allow the decoding of frequency and amplitude modulated

signals (i.e., FM/AM radio signals).

Blind source separation is the problem of identifying and separating unknown source signals from a set of unknown (possibly nonlinear) mixtures of them. A common example is the *cocktail party problem*, in which several people speak simultaneously and a listener is trying to follow one of the conversations. If the mixture is instantaneous, Sprekeler et al. (2014) have realized that a good heuristic indicates that the slowest source is frequently slower than transformations of sources and their mixtures. Therefore, the slowest feature extracted from the mixture would be strongly related to the slowest source (or a monotonous transformation of it). Such a feature and its transformations can be removed from the mixture, and the procedure can be repeated to extract the next source.

The applications that have been mentioned in this section are specific examples of unsupervised feature extraction, where the features of interest are well defined. However, SFA can also be applied to less well defined situations.

In Dähne et al. (2011) linear SFA has been applied to *EEG data* recorded with 63 electrodes during an auditory discrimination task. Fisher discriminant analysis is then applied to the extracted features to discriminate between two auditory stimuli. The system is therefore able to extract the human auditory percept from the EEG recording. Such an approach is of interest for the area of brain-computer interfaces. Unspecific feature extraction with SFA has been also used by Höfer et al. (2010) for the classification of *humanoid robot postures*.

2.6.2 Applications of SFA for Dimensionality Reduction

Feature extraction using SFA on high-dimensional input has been employed in a system by Legenstein et al. (2010), where the input is a sequence of 155×155 pixel images showing two objects and one out of two animated fish. For one of the fish one object is a *target* and the other a *distractor*, and for the other fish the role of the objects is interchanged. The direction in which the fish swim is controlled by a reinforcement learning system that has to learn to recognize the type of fish present and lead it to its target while avoiding the distractor. However, reinforcement learning is generally notoriously slow on high-dimensional input, and it is therefore mandatory to first reduce the original dimensionality. In this particular example SFA was able to reduce the dimensionality from 24,025 features to 32 while still preserving all information necessary to solve the task.

2.6.3 Applications of SFA for Classification

In the applications above, SFA has been used in a purely unsupervised manner. We now consider extensions for supervised learning, first for classification and later (Section 2.6.4) for regression.

The first example of classification with SFA is the classification of *handwritten digits* from the MNIST database (Berkes, 2005a). Random pairs of training samples from the same class are connected to build mini-sequences of length two. SFA with polynomials of degree 3 is then trained on the collection of these

mini-sequences. A Gaussian classifier is applied to the nine slowest extracted features to do the final classification. An error rate of 1.5% is obtained, which is close to the 0.95% achieved by LeNet-5 (LeCun et al., 1998), a hierarchical special-purpose architecture for digit recognition. The same approach has also been applied to *human gesture recognition* (Koch et al., 2010) and a similar approach to *monocular road segmentation* (Kuhnl et al., 2011).

In the experimental research of this dissertation, the more general formulation of SFA with graph structures, GSFA, has been applied to the German *traffic sign recognition* benchmark (GTSRB) proposed by Stallkamp et al. (2011), where the goal is to classify photographs of traffic signs taken from a car driving along German roads, which is of major interest for the development of driver assistance systems.

The database consists of 26,640 training and 12,569 test images of 43 different types of traffic signs. The position of the signs in the images is available, as well as precomputed HOG-features (histograms of oriented gradients). The input data is described more extensively in Section 3.5.1, and Figure 4.5 on page 86 illustrates the actual traffic signs considered. The GSFA system above ranked 8th place out of 24 groups in the GTSRB online competition with a performance of 96.4% , whereas human performance was 98.8% and the best algorithm achieved 99.0% recognition rate.

A more complex system has been proposed by Zhang and Tao (2012) for *human action recognition*. In this case, SFA is applied to cuboids (voxels), that is, subsequences within localized regions extracted from video sequences of subjects performing various actions, such as walking, jogging, hand clapping, etc. Three supervised learning strategies are proposed, all of which exploit the fact that SFA learns features that vary slowly for sequences of the particular type of action used for training. Extracting features during testing from a different type of action typically produces outputs that change much faster than sequences of the same action used for training. The results show that SFA achieves comparable or even better performance than previous methods.

The approach above has been extended for automatic *detection of violence in videos* by Wang et al. (2012). Optical flow fields are used to compute dense trajectories of interest points. Then, cuboids are extracted at locations defined by these dense trajectories instead of at fixed spatial locations. Slow features are extracted from these cuboids using a variant of SFA called *discriminative SFA*. Afterwards, *accumulated squared derivative* (ASD) feature vectors are constructed using the Δ values of all cuboids, and a linear SVM is applied to these vectors. Human action recognition might also be used to *recognize prisoner's activities*, which has been described by Rubia and Manimala (2013).

The problem of *human action recognition* has also been addressed by Sun et al. (2014) using an architecture that combines aspects of hierarchical SFA and convolutional neural networks (CNNs). In a first layer, linear SFA with weight sharing is trained on cuboids. This step is analogous to the convolution operation of CNNs. Afterwards, two types of max-pooling are applied: 1) The

maximum output over all J slow features is computed for each possible space-time location. 2) Standard max-pooling is applied spatially to 2×2 elements. A second layer consists of linear SFA with weight sharing. Finally, the features are processed by a quantization and classification method.

2.6.4 Applications of SFA for Regression

SFA, and particularly GSFA, have also been applied to regression problems with excellent results. This section covers two of the applications in the field of facial image processing that I have developed for this research; age and gender estimation from artificial images and face detection.

Age and Gender Estimation

Successful human-computer interaction benefits from knowing basic information about the interacting subjects, such as their approximate age, gender, and general mood. Gender is a parameter that can be estimated relatively well (e.g., Jia and Cristianini, 2015). However, age estimation is a challenging problem, since aging only results in subtle changes in the face appearance compared to other variables, and because aging is influenced by several factors (see Fu et al., 2010).

A four-layer hierarchical GSFA network has been proposed by Escalante-B. and Wiskott (2010). This network (Fig. 1.1) estimates age and gender from frontal, static, 135×135-pixel face images of artificial subjects that have been created with special software for 3D face modeling and rendering (FaceGen SDK, Singular Inversions Inc., 2008). To estimate age, the training and test images vary from 16 to 65 years, while gender, racial background, and identity are chosen randomly. To estimate gender, a similar database is created and the same algorithm is applied. This is possible because gender is represented as a continuous variable within the software from -3 (very feminine) to $+3$ (very masculine) rather than a binary one (male vs. female). See Figure 2.3 for an illustration of the images and learned filters. Two training graphs are used, where two images are connected if they have either similar age or gender label (serial graph, see Section 3.4.3). After the GSFA network is trained, the first three outputs are used for training a Gaussian classifier to estimate either the age or gender group. This yields *a posteriori* group probabilities from which an expectation value can be computed as the final estimate.

In both cases, good performance has been achieved (on test data), with a root mean square error (RMSE) of 3.8 years for age and 0.33 units for gender, compared to a chance level of 13.8 years and 1.73 units, respectively. As expected, age estimation is less accurate than gender estimation (relative to chance level). Interestingly, best performance is achieved with a linear SFA network, outperforming various (more complex) nonlinear networks. One reason might be that the number of training images is insufficient (only 4140/10,800 images for age/gender) to train a nonlinear network, but it might also be that

the rendering software uses a too simple model for age and gender.

The recent release of large publicly available databases with age/gender labels allows the estimation on real photographs; such an application is addressed in Chapter 5.

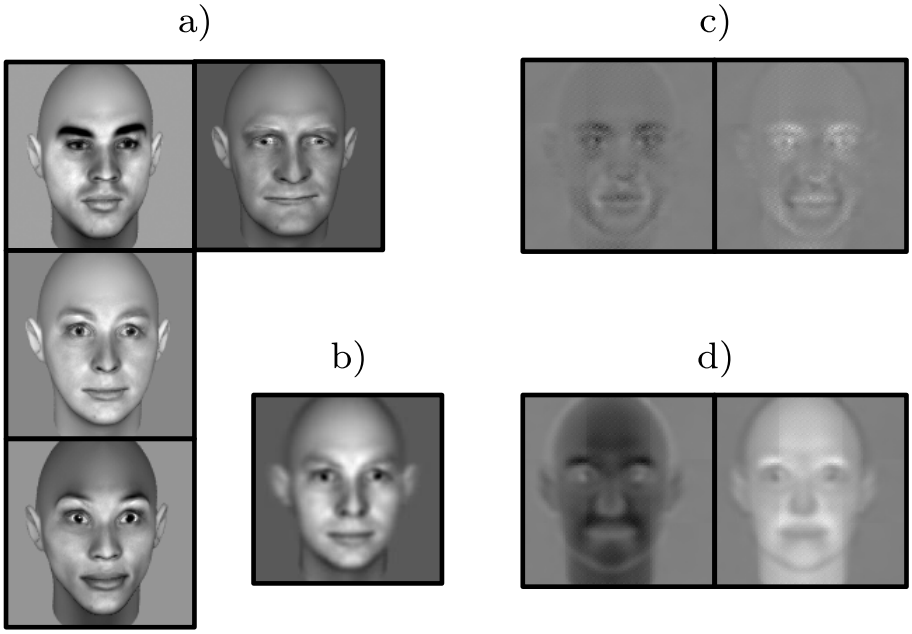


Figure 2.3: (a) Sample images used for age estimation, (b) average image, (c) image variation that specifically activates the slowest feature computed by the network for age estimation (right, its negative), (d) image variation that specifically activates the slowest feature computed by the network for gender estimation (right, its negative). The images used for age and gender estimation have a slightly different head angle. Notice how the image variation for gender resembles a masculine face, whereas its negative resembles a feminine one and is lighter. Several studies on sexual dimorphism in humans have found that adult females are lighter than adult males (see Madrigal and Kelly, 2007), which has probably been taken into account by the face model and "rediscovered" by SFA. [Figure reproduced from (Escalante-B. and Wiskott, 2012), with permission.]

Face Detection

Systems for face detection from images have become very popular due to the increase in computing power and specialized algorithms capable of running on cameras, smartphones, and other portable devices. Face detection is typically used to initialize face tracking and is a necessary step to normalize the images, for example, before age and gender can be estimated. In spite of its progress, face detection is still a challenge under extreme conditions, in particular in the presence of facial hair, strong or unusual lighting conditions, face occlusions, poor image quality and image artifacts.

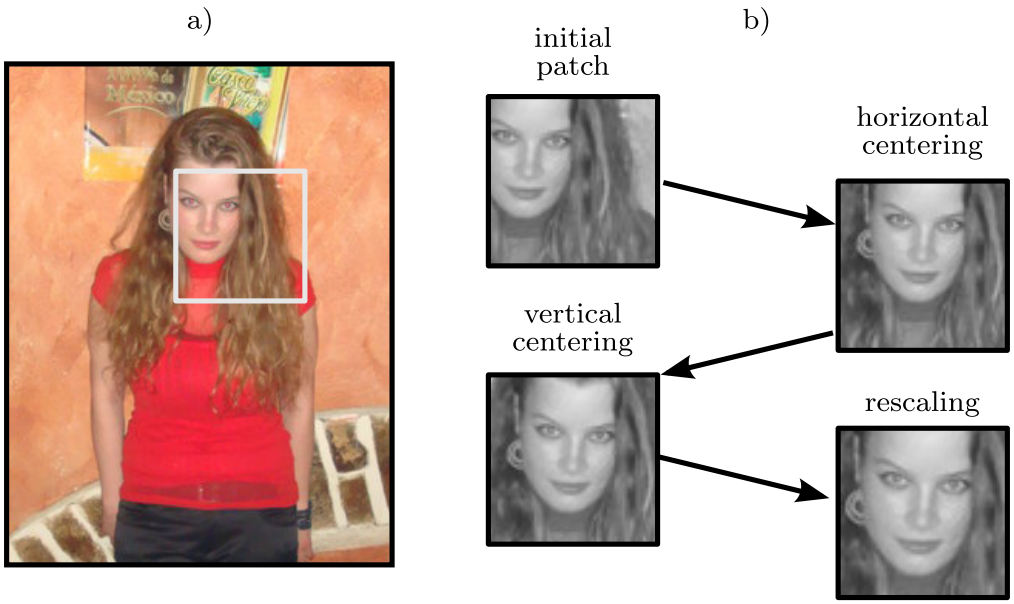


Figure 2.4: (a) A test image showing a single image patch containing a face. (b) Refinement of the location and size of the face patch: initial patch and patch after horizontal centering, vertical centering, and rescaling. Image used with kind permission of the subject.
[Figure reproduced from (Escalante-B. and Wiskott, 2012), with permission.]

It has been described above how a hierarchical SFA network can be used to estimate a continuous parameter such as age or gender from a facial image (either using sample reordering or GSFA with an appropriate training graph). Such approaches can also be used to learn other parameters such as x-position, y-position, or scale. Using three separate networks, image patches potentially containing a face can be centered at the face (see Figure 2.4) as follows: 1) Estimate x-position of the face and center the patch horizontally. 2) Estimate y-position of the face and center the patch vertically. 3) Estimate scale of the face and resize the patch. A fourth network can be trained to estimate the quality of the normalization and to indicate whether a face is present at all. To improve detection, this process is repeated three times successively, leading each time to more accurate and reliable localization. Finally, eye positions can be determined around their average position within normalized faces using an eye specific SFA network. To detect multiple faces, an image is first tiled with overlap into many candidate regions of different sizes, and the algorithm above is applied to each region separately. Thus, most candidate regions typically do not contain a face at all, which is the reason why the fourth network is necessary to decide on the presence of a face.

In this example application, 40,000 frontal face photographs from different sources are used for training. The networks are improved versions of the ones

used for age and gender estimation, and have been redesigned with a 9- or 11-layer structure characterized by a very small fan-in in all layers, which is particularly useful to reduce overfitting.

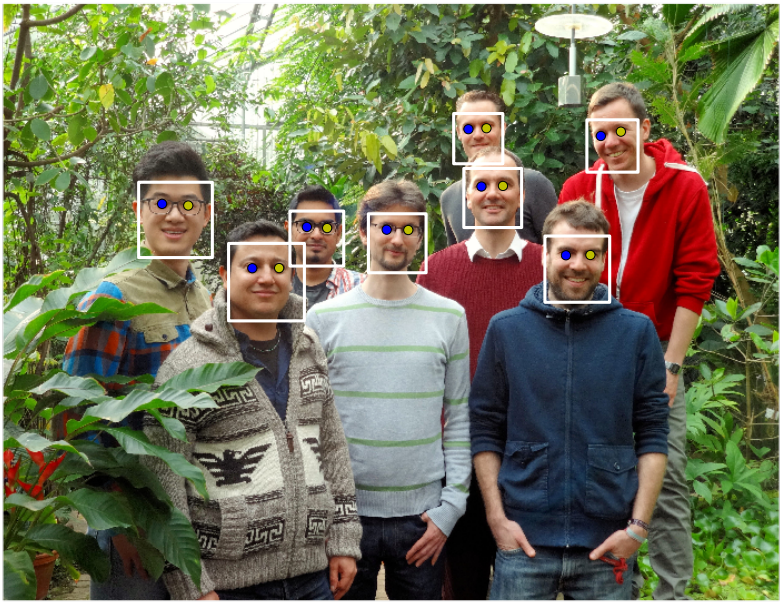


Figure 2.5: Example of face detection using the developed system. The original image is the current group photograph of the Theory of Neural Systems Chair. Detected faces are delimited by a white bounding box. The detected left and right eyes are marked with a yellow and a blue circle, respectively. This image produces no false detections and all faces are detected even though one of them is rotated and another slightly occluded. For display purposes the original color image is shown. However, detection only uses grayscale information. The plot was created with with the `FaceDetectUpdated.py` program (source code available upon request). This program is a newer and improved version of the software that had been evaluated by Mohamed and Mahdi (2010).

The face detection system has a competitive performance on different image databases, as reported by Mohamed and Mahdi (2010). According to evaluations I performed on the system using different databases, it yields a detection rate on grayscale photographs from 71.5% to 99.5% depending on the difficulty of the test database. Figure 2.5 shows a sample run of the system.

2.7 Discussion of SFA and its Applications

The applications reviewed in this chapter demonstrate the versatility of SFA and that it can be applied to a broad range of problems in machine learning and computer vision. The robustness and flexibility of SFA are two factors that make it a useful general purpose preprocessing tool for feature extraction and dimensionality reduction. The algorithm is computationally efficient (though not enough for high-dimensional data), easy to implement and straightforward to use, since it is basically parameter free. In standard SFA, the only choice is that of the function space used, i.e., the type of nonlinearity. In hierarchical SFA, one also has to define a network structure and decide on the number of SFA outputs passed to the next layer.

The theory of slow feature extraction does not end with standard SFA. Three extensions of SFA have already been mentioned and are proposed in this work, namely: graph-based SFA (GSFA), exact label learning (ELL), and hierarchical information-preserving GSFA (HiGSFA). These extensions have been designed for supervised learning and are consequently more powerful than standard SFA for this class of problems, yielding higher label estimation accuracies and classification rates. The next three chapters address each of these extensions, present additional applications, and describe the results. The focus of the next chapter is thus on GSFA.

Chapter 3

Graph-Based SFA

This chapter introduces an extension to SFA, called graph-based SFA (GSFA), that has been explicitly designed for supervised learning. GSFA is able to extract a compact set of features that can be post-processed by typical supervised algorithms to generate the final label or class estimation with good accuracy[1].

As explained in the preceding chapter, the training data used by SFA is a multi-dimensional time series (i.e., a sequence of samples) with transitions only between temporally adjacent samples. In GSFA, the training data is specified by a graph structure called training graph, where the vertices are the samples and the edges represent transitions. It is possible to include transitions between arbitrary pairs of samples. Moreover, both vertices and edges are weighted, and the edge weights are typically related to the similarities of the corresponding labels.

It is shown that GSFA computes an optimal solution to the GSFA problem (in the considered function space), a weighted version of the SFA optimization problem that generalizes the notion of slowness from sequences of samples to training graphs. GSFA can be used to solve classification and regression problems by choosing appropriate training graphs. For classification, the most straightforward graph yields features equivalent to those of (nonlinear) Fisher discriminant analysis. Emphasis is on regression, where four different graphs are proposed. GSFA is a promising algorithm in combination with the proposed graphs, particularly for tackling image analysis problems where linear models are insufficient as well as when feature selection is difficult.

The remainder of the chapter is organized as follows. The next section presents the general motivation behind GSFA. Then, the GSFA optimization problem is introduced and the GSFA algorithm is proposed. Afterwards, a classification method based on SFA is recalled, and a training graph for solving this task with GSFA is proposed. Moreover, various training graphs useful to solve regression problems are proposed, offering great computational efficiency and good accuracy. Thereafter, the performance of four training graphs is evalu-

[1]This chapter is mostly an edited version of (Escalante-B. and Wiskott, 2013), which has been published in the Journal of Machine Learning Research (JMLR).

ated experimentally and compared to other common supervised methods (e.g., PCA+SVM) w.r.t. a regression problem closely related to face detection from real photographs. A discussion section concludes the chapter.

3.1 Introduction

Supervised learning from high-dimensional data has applications in many areas, such as image analysis, human-computer interfaces, product quality control, and robotics. However, supervised learning of high-dimensional data is in general still a challenge due to insufficient training data and several phenomena referred to as the curse of dimensionality. These limitations reduce the practical applicability of supervised learning for high-dimensional data even when using state-of-the-art algorithms and lots of computational power.

Unsupervised dimensionality reduction, including algorithms such as principal component analysis (PCA) or locality preserving projections (LPP, He and Niyogi, 2003), can be used to attenuate these problems. After dimensionality reduction, typical supervised learning algorithms can be applied. Frequent benefits include a lower computational cost and better robustness against overfitting. However, since the final goal is to solve a supervised learning problem, such an approach is inherently suboptimal.

Supervised dimensionality reduction is a more appropriate technique in this case. Its goal is to compute a low-dimensional set of features from the high-dimensional input samples that contain predictive information about the labels (Rish et al., 2008). One advantage is that dimensions irrelevant for the label estimation can be discarded, resulting in a more compact representation and more accurate label estimations. Different supervised algorithms can then be applied to the low-dimensional data. A widely known algorithm for supervised dimensionality reduction is Fisher discriminant analysis (FDA) (Fisher, 1936). Sugiyama (2006) proposed local FDA (LFDA), an adaptation of FDA with a discriminative objective function that also preserves the local structure of the input data. Later, Sugiyama et al. (2010) proposed semi-supervised LFDA (SELF) bridging LFDA and PCA and allowing the combination of labeled and unlabeled data. Tang and Zhong (2007) introduced pairwise constraints-guided feature projection (PCGFP), where two types of constraints are allowed. Must-link constraints denote that a pair of samples should be mapped closely in the low-dimensional space, while cannot-link constraints require that the samples are mapped far apart. Later, Zhang et al. (2007) proposed semi-supervised dimensionality reduction (SSDR), which is similar to PCGFP and also supports semi-supervised learning.

The previous chapter mentions different applications of direct SFA where the goal is the solution of classification problems. We now focus on the methods employed to address such applications. Franzius et al. (2011) have extracted the identity of animated fish invariant to pose (including a rotation angle and the fish position) with SFA. A long sequence of fish images is rendered from

3D models in which the pose of the fish changed following a Brownian motion, and in which the probability of randomly changing the fish identity is relatively small, making identity a feature that changes slowly. The experimental results confirm that SFA is capable of extracting categorical information. Klampfl and Maass (2010) have introduced a particular Markov chain to generate a sequence used to train SFA for classification. The transition probability between samples from different object identities is proportional to a small parameter a. The authors show that in the limit $a \to 0$ (i.e., only intra-class transitions), the features learned by SFA are equivalent to those learned by Fisher discriminant analysis (FDA). The equivalence of the discrimination capability of SFA and FDA in some setups had already been known (compare Berkes, 2005a, and Berkes, 2005b) but had not been rigorously shown before. In these two papers by Berkes, hand-written digits from the MNIST database are recognized. The method consists in constructing several mini-sequences of two samples from the same digit and using them to train SFA. The same approach has been also applied more recently to human gesture recognition by Koch et al. (2010) and a similar approach to monocular road segmentation by Kuhnl et al. (2011). Zhang and Tao (2012) has addressed human action recognition, in which the difference between delta values of different training signals is amplified and used for discrimination.

Standard SFA has been used to solve regression problems as well. Franzius et al. (2011) have used SFA to learn the position of animated fish from images with plain background. They use the same training sequence for learning fish identities, thus the fish position changed continuously over time. However, a different supervised post-processing step is employed consisting of linear regression coupled with a nonlinear transformation.

3.1.1 Connection of GSFA with Other Algorithms

The objective function of the GSFA optimization problem is a weighted sum of squared output differences and is therefore similar to the underlying objective functions of, for example, FDA, LFDA, SELF, PCGFP, and SSDR. However, in general the optimization problem solved by GSFA differs from these in at least one of the following elements: a) the concrete coefficients of the objective function, b) the constraints, or c) the feature space considered. Although nonlinear or kernelized versions of the algorithms above can be defined, one has to overcome the difficulty of finding a good nonlinearity or kernel. In contrast, SFA and GSFA have been conceived from the beginning as nonlinear algorithms without resorting to kernels (although there exist versions of SFA with a kernel: Bray and Martinez, 2003; Vollgraf and Obermayer, 2006; Böhmer et al., 2012), with linear SFA being just a less used special case. Another difference to various algorithms above is that SFA (and GSFA) does not explicitly attempt to preserve the spatial structure of the input data. Instead, it preserves the similarity structure provided, which is responsible for a better optimization towards the labels.

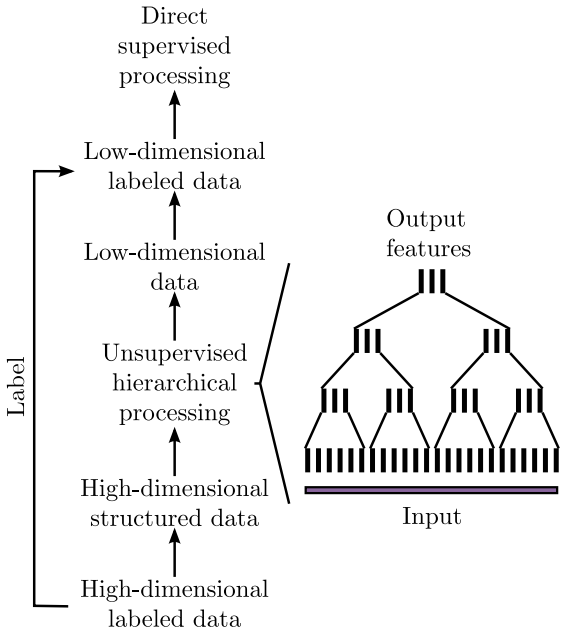


Figure 3.1: Transformation of a supervised learning problem on high-dimensional data into a supervised learning problem on low-dimensional data by means of unsupervised hierarchical processing on structured data in the form of a training graph, that is, without labels. This construction allows the solution of supervised learning problems on high-dimensional data when the dimensionality and number of samples make the direct application of many conventional supervised algorithms infeasible.
[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

Besides the similarities and differences outlined above, GSFA is strongly connected to some algorithms in specific cases. For instance, features equivalent to those of FDA can be obtained if a particular training graph is given to linear GSFA. There is also a close relation between SFA and Laplacian eigenmaps (LE), which has been studied by Sprekeler (2011). GSFA and LE basically have the same objective function, but in general GSFA uses different edge-weight (adjacency) matrices, has different normalization constraints, supports vertex weights, and uses function spaces.

There is also a strong connection between GSFA and LPP. In Section 3.6.2, it is described how to use GSFA to extract LPP features and *vice versa*. This is a remarkable connection because GSFA and LPP originate from different backgrounds and are typically used for related but different goals. Generalized SFA (genSFA) (Sprekeler, 2011; Rehn, 2013), being basically LPP on nonlinearly expanded data, is also closely connected to GSFA.

One advantage of GSFA compared to many other algorithms for supervised dimensionality reduction is that it is designed for both classification and regression (using appropriate training graphs), whereas other algorithms typically focus on classification only.

3.1.2 General Approach behind GSFA

Given a large number of high-dimensional labeled samples, supervised learning algorithms can often not be applied due to prohibitive computational requirements. In such cases, the following general scheme based on hierarchical GSFA/SFA, illustrated in Figure 3.1, is proposed:

1. Transform the labeled data to structured data, where the label information is implicitly encoded in the connections between the data points (samples). This permits using unsupervised learning algorithms, such as SFA, or its extension GSFA.
2. Use hierarchical processing to reduce the dimensionality, resulting in low-dimensional data with component similarities strongly dependent on the graph connectivity. Since the label information is encoded in the graph connectivity, the low-dimensional data is highly predictive of the labels.
3. Convert the (low-dimensional) data back to labeled data by combining the low-dimensional data points with the original labels or classes. This now constitutes a dataset suitable for standard supervised learning methods, because the dimensionality has become manageable.
4. Use standard supervised learning methods on the low-dimensional labeled data to estimate the labels. The unsupervised hierarchical network plus the supervised direct method together constitute the classifier or regression architecture.

In the case of GSFA, the structured training data is called training graph, a weighted graph that has vertices representing the samples, vertex weights specifying *a priori* sample probabilities, and edge weights indicating desired output similarities, as derived from the labels. Details are given in Section 3.2. This structure permits us to extend SFA to extract features from the data points that tend to reflect similarity relationships between their labels without the need to reproduce the labels themselves. A concrete example of the application of the method to a regression problem is illustrated in Figure 3.2. Various important advantages of GSFA are inherited from SFA:

- It allows hierarchical processing, which has various remarkable properties, as described in Section 2.4. One of them, illustrated in Figure 2.2, is that the local application of SFA/GSFA to lower-dimensional data chunks typically results in less overfitting than non-hierarchical SFA/GSFA.
- SFA has a complexity of $\mathcal{O}(N)$ in the number of samples N and $\mathcal{O}(I^3)$ in the number of dimensions I (possibly after a nonlinear expansion), see Section 2.3. Hierarchical processing greatly reduces the latter complexity down to $\mathcal{O}(I)$. In practice, processing 100,000 samples of 10,000-dimensional input data can be done in less than three hours by using hierarchical SFA/GSFA without resorting to parallelization or GPU computing.

- Typically no expensive parameter search is required. The SFA and GSFA algorithms themselves are almost parameter free. Only the nonlinear expansion has to be defined. In hierarchical SFA, the structure of the network has several parameters, but the choice is usually not critical.

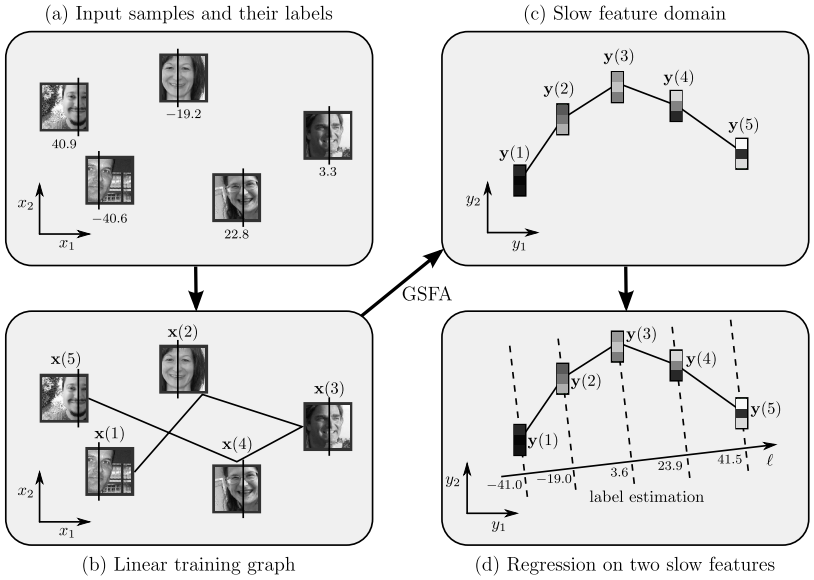


Figure 3.2: Illustration of the use of GSFA to solve a regression problem. (a) The input samples are 128×128-pixel images with labels that indicate the horizontal position of the center of the face. (b) A training graph is constructed using the label information. In this example, only images with most similar labels are connected resulting in a linear graph. (c) The data dimensionality is reduced with GSFA, yielding in this case 3-dimensional feature vectors plotted in the first two dimensions. (d) The application of standard regression methods to the slow features (e.g., linear regression) generates the label estimates. In theory, the labels can be estimated from y_1 alone. In practice, performance is usually improved by using not one, but a few slow features.
[Based on a figure from (Escalante-B. and Wiskott, 2013).]

The next section outlines various high-level properties of GSFA and then presents the specifics of the algorithm.

3.2 Training Graphs and Graph-Based SFA

This section formalizes the concept of training graph and proposes: (1) the GSFA optimization problem, (2) the GSFA algorithm, and (3) a probabilistic model for the generation of training data, connecting SFA and GSFA.

3.2.1 Organization of the Training Samples in a Graph

Learning from a single time series (i.e., a sequence of samples), as in standard SFA, is motivated from biology, because the input data is assumed to originate from sensory perception. In a more technical and supervised learning setting, the training data needs not be a time series but may be a set of independent samples. However, one can use the labels to induce structure. For instance, face images may come from different persons and different sources but can still be ordered by, say, age. If one arranges such images by increasing age in a sequence, they would form a linear structure that could be used for training much like a regular time series.

One central contribution of this chapter is the consideration of a more complex structure for training SFA called training graph. In the example above, one can then introduce a weighted edge between any pair of face images according to some similarity measure based on age (or other criteria such as gender, race, or mimic expression), with high similarity resulting in large edge weights. The original SFA objective then needs to be adapted such that samples connected by large edge weights yield similar output values.

In mathematical terms, the training data is represented as a training graph $G = (\mathbf{V},\mathbf{E})$ (illustrated in Figure 3.3) with a set $\mathbf{V}$ of vertices $\mathbf{x}(n)$ (each vertex[2] being a sample), and a set $\mathbf{E}$ of edges $(\mathbf{x}(n), \mathbf{x}(n'))$, which are pairs of samples, with $1 \leq n,n' \leq N$. The index n (or n') substitutes the time variable t. The edges are undirected and have symmetric weights

$$\gamma_{n,n'} = \gamma_{n',n} \tag{5}$$

that indicate the similarity between the connected vertices; also each vertex $\mathbf{x}(n)$ has an associated weight $v_n > 0$, which can be used to reflect its importance, frequency, or reliability. For instance, a sample occurring frequently in an observed phenomenon should have a larger weight than a rare sample. This representation includes the standard time series as a special case in which the graph has a linear structure and all vertex and edge weights are identical (Figure 3.3.b). How exactly edge weights are derived from label values will be elaborated later.

3.2.2 GSFA Optimization Problem

The concept of slowness, originally conceived for sequences of samples, is extended here to data that has been structured in a training graph. It is assumed that the edge weights have already been fixed (see Sections 3.3 and 3.4 for how to choose them). The generalized optimization problem can then be formalized as follows: For $1 \leq j \leq J$, find features $y_j(n) \stackrel{\text{def}}{=} g_j(\mathbf{x}(n))$, where $1 \leq n \leq N$,

[2]The terms vertex and node can be used interchangeably in graph theory. However, for clarity, in this work the term vertex is reserved for the elements of a training graph and the term node is reserved for the elements of a hierarchical network.

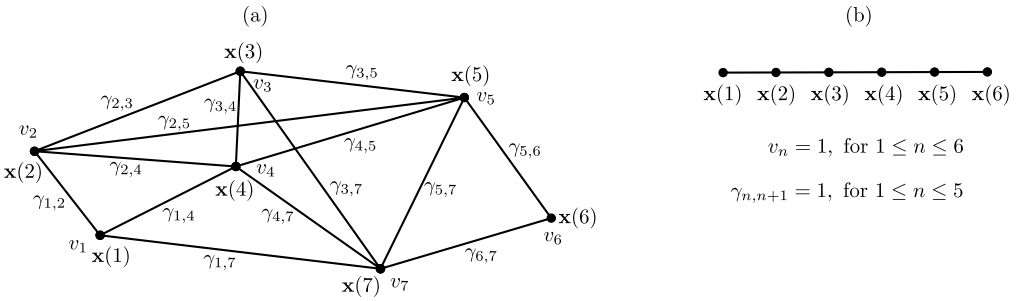


Figure 3.3: (a) Example of a training graph with $N = 7$ vertices. (b) A regular sample sequence (time series), which can be used to train SFA. This sequence is represented here as a linear graph that can be used with GSFA. If labels are available and the samples have been reordered by increasing/decreasing label (e.g., instead of having been ordered by time), it is called *sample reordering* graph.

[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

J is the number of output features, and g_j is a function belonging to a feature space $\mathcal{F}$, such that the objective function

$$\Delta_j \stackrel{\text{def}}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} (y_j(n') - y_j(n))^2 \text{ is minimal} \quad \textbf{(weighted delta value)} \tag{6}$$

under the constraints

$$\frac{1}{Q} \sum_n v_n y_j(n) = 0 \ \textbf{(weighted zero mean)}, \tag{7}$$

$$\frac{1}{Q} \sum_n v_n (y_j(n))^2 = 1 \ \textbf{(weighted unit variance)}, \text{ and} \tag{8}$$

$$\frac{1}{Q} \sum_n v_n y_j(n) y_{j'}(n) = 0, \text{ for } j' < j \ \textbf{(weighted decorrelation)}, \tag{9}$$

with

$$R \stackrel{\text{def}}{=} \sum_{n,n'} \gamma_{n,n'} \,, \tag{10}$$

$$Q \stackrel{\text{def}}{=} \sum_n v_n \,. \tag{11}$$

Compared to the original SFA problem (1–4), the vertex weights generalize the normalization constraints, whereas the edge weights extend the objective function to penalize the difference between the outputs of arbitrary pairs of samples. As in SFA, frequent choices for $\mathcal{F}$ are all linear or quadratic transfor-

mations of the inputs. Of course, the factor $1/R$ in the objective function is not essential for the minimization problem. Likewise, the factor $1/Q$ can be dropped from (7–9). These factors, however, provide invariance to the scale of the edge weights as well as to the scale of the vertex weights, and serve a normalization purpose. More concretely, $\Delta_j \geq 0$ for training graphs with non-negative edge weights, and $\Delta_j \leq 4$ if restriction (32) is also fulfilled (this is explained in Section 3.2.5), for $1 \leq j \leq J$. These are basically the same bounds as in standard SFA ($0 < \Delta \leq 4$).

By definition (Section 3.2.1), training graphs are undirected and have symmetric edge weights. This does not cause any loss of generality and is justified by the GSFA optimization problem above. Its objective function (6) is insensitive to the direction of an edge because the sign of the output difference cancels out during the computation of Δ_j. It therefore makes no difference whether we choose $\gamma_{n,n'} = 2$ and $\gamma_{n',n} = 0$ or $\gamma_{n,n'} = \gamma_{n',n} = 1$, for instance. Note also that $\gamma_{n,n}$ multiplies with zero in (6) and only enters into the calculation of R. The variables $\gamma_{n,n}$ are kept only for mathematical convenience, but are generally set to zero.

3.2.3 Linear Graph-Based SFA Algorithm (Linear GSFA)

Similarly to the standard linear SFA algorithm, which solves the standard SFA problem in the linear function space, here an extension that computes an optimal solution to the GSFA problem within the same space is proposed. Let the vertices $\mathbf{V} = \{\mathbf{x}(1), \ldots, \mathbf{x}(N)\}$ be the input samples with weights $\{v_1, \ldots, v_N\}$ and the edges $\mathbf{E}$ be a set of edges $(\mathbf{x}(n), \mathbf{x}(n'))$ with edge weights $\gamma_{n,n'}$. To simplify notation, zero edge weights $\gamma_{n,n'} = 0$ are defined for non-existing edges $(\mathbf{x}(n), \mathbf{x}(n')) \notin \mathbf{E}$. The linear GSFA algorithm differs from standard SFA only in the computation of the matrices $\mathbf{C}$ and $\dot{\mathbf{C}}$, which now take into account the neighborhood structure (samples, edges, and weights) specified by the training graph.

The sample covariance matrix $\mathbf{C_G}$ is defined as:

$$\mathbf{C_G} \stackrel{\text{def}}{=} \frac{1}{Q} \sum_n v_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T = \frac{1}{Q} \sum_n \left(v_n \mathbf{x}(n)(\mathbf{x}(n))^T \right) - \hat{\mathbf{x}}\hat{\mathbf{x}}^T , \quad (12)$$

where

$$\hat{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{Q} \sum_n v_n \mathbf{x}(n) \quad (13)$$

is the weighted average of all samples. The derivative second-moment matrix $\dot{\mathbf{C}}_\mathbf{G}$ is defined as:

$$\dot{\mathbf{C}}_\mathbf{G} \stackrel{\text{def}}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} \big(\mathbf{x}(n') - \mathbf{x}(n)\big)\big(\mathbf{x}(n') - \mathbf{x}(n)\big)^T . \quad (14)$$

Given these matrices, the computation of $\mathbf{W}$ and the rest of the algorithm proceeds as in the standard algorithm (Section 2.3). Thus, a sphering matrix $\mathbf{S}$

and a rotation matrix $\mathbf{R}$ are computed with

$$\mathbf{S}^T\mathbf{C_G}\mathbf{S} = \mathbf{I}, \text{ and} \tag{15}$$

$$\mathbf{R}^T\mathbf{S}^T\dot{\mathbf{C}}_\mathbf{G}\mathbf{S}\mathbf{R} = \mathbf{\Lambda}, \tag{16}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with diagonal elements $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_J$. Finally the algorithm returns $\Delta(y_1), \ldots, \Delta(y_J)$, $\mathbf{W}$ and $\mathbf{y}(n)$, where

$$\mathbf{W} = \mathbf{S}\mathbf{R}, \text{ and} \tag{17}$$

$$\mathbf{y}(n) = \mathbf{W}^T(\mathbf{x}(n) - \hat{\mathbf{x}}). \tag{18}$$

3.2.4 Correctness of the Graph-Based SFA Algorithm

It can be proven that the GSFA algorithm indeed solves the optimization problem (6–9) with a proof similar to that of the optimality of the standard SFA algorithm (Wiskott and Sejnowski, 2002), as follows.

For simplicity, assume that $\mathbf{C_G}$ and $\dot{\mathbf{C}}_\mathbf{G}$ have full rank. The weighted zero mean constraint (7) holds trivially for any $\mathbf{W}$, because

$$\sum_n v_n \mathbf{y}(n) \overset{(18)}{=} \sum_n v_n \mathbf{W}^T(\mathbf{x}(n) - \hat{\mathbf{x}}) \tag{19}$$

$$= \mathbf{W}^T \left(\sum_n v_n \mathbf{x}(n) - \sum_{n'} v_{n'} \hat{\mathbf{x}} \right) \tag{20}$$

$$\overset{(11,13)}{=} \mathbf{W}^T (Q\,\hat{\mathbf{x}} - Q\,\hat{\mathbf{x}}) = \mathbf{0}. \tag{21}$$

One can also find

$$\mathbf{I} = \mathbf{R}^T\mathbf{I}\mathbf{R} \quad \text{(since } \mathbf{R} \text{ is a rotation matrix)}, \tag{22}$$

$$\overset{(15)}{=} \mathbf{R}^T(\mathbf{S}^T\mathbf{C_G}\mathbf{S})\mathbf{R}, \tag{23}$$

$$\overset{(17)}{=} \mathbf{W}^T\mathbf{C_G}\mathbf{W}, \tag{24}$$

$$\overset{(12)}{=} \mathbf{W}^T \frac{1}{Q} \sum_n v_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T \mathbf{W}, \tag{25}$$

$$\overset{(18)}{=} \frac{1}{Q} \sum_n v_n \mathbf{y}(n)(\mathbf{y}(n))^T, \tag{26}$$

which is equivalent to the normalization constraints (8) and (9).

Now, let us consider the objective function

$$\Delta_j \overset{(6)}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} \big(y_j(n') - y_j(n)\big)^2 \tag{27}$$

$$\overset{(14)}{=} \mathbf{w}_j^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{w}_j \tag{28}$$

$$\overset{(17)}{=} \mathbf{r}_j^T \mathbf{S}^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{S} \mathbf{r}_j \tag{29}$$

$$\overset{(16)}{=} \lambda_j \,, \tag{30}$$

where $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_J)$. The algorithm finds a rotation matrix $\mathbf{R}$ solving (16) and yielding increasing λs. It can be seen (cf. Adali and Haykin, 2010, Section 4.2.3) that this $\mathbf{R}$ also achieves the minimization of Δ_j, for $j = 1, \ldots, J$, hence, fulfilling (6).

3.2.5 Probabilistic Interpretation of Training Graphs

There is an intuitive explanation of the relationship between GSFA and standard SFA. Readers less interested in this theoretical excursion can safely skip it. This section is inspired in part by the Markov chain introduced by Klampfl and Maass (2010).

Given a training graph, it is shown below how to construct a Markov chain $\mathcal{M}$ for the generation of input data such that training standard SFA with such data yields the same features as GSFA does with the graph. Contrary to the graph introduced by Klampfl and Maass (2010), the formulation here is not restricted to classification, accounting for any training graph irrespective of its purpose, and there is one state per sample rather than one state per class. In order for the equivalence of GSFA and SFA to hold, the vertex weights $\tilde{v}_n$ and edge weights $\tilde{\gamma}_{n,n'}$ of the graph must fulfill the following *normalization restrictions*:

$$\sum_n \tilde{v}_n = 1 \,, \tag{31}$$

$$\sum_{n'} \tilde{\gamma}_{n,n'}/\tilde{v}_n = 1 \quad \forall n \,, \tag{32}$$

$$\overset{(5)}{\Longleftrightarrow} \quad \sum_{n'} \tilde{\gamma}_{n',n}/\tilde{v}_n = 1 \quad \forall n \,, \tag{33}$$

$$\sum_{n,n'} \tilde{\gamma}_{n,n'} \overset{(31,32)}{=} 1 \,, \tag{34}$$

where it is assumed that not only $\tilde{v}_n > 0$ but also $\tilde{\gamma}_{n,n'} \geq 0$, for all n and n'. Restrictions (31) and (34) can always be assumed without loss of generality, because they can be achieved by a constant scaling of the weights (i.e., $\tilde{v}_n \overset{\text{def}}{=} v_n/Q$, $\tilde{\gamma}_{n,n'} \overset{\text{def}}{=} \gamma_{n,n'}/R$) without affecting the outputs generated by GSFA. Restriction (32) is fundamental because it limits the graph connectivity, and indicates (after multiplying with $\tilde{v}_n$) that each vertex weight should be equal to the sum of the weights of all edges originating from such a vertex.

The Markov chain is then a sequence $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3, \ldots$ of random variables that can assume states that correspond to different input samples. $\mathbf{Z}_1$ is drawn from the initial distribution $\boldsymbol{p}_1$, which is equal to the stationary distribution $\boldsymbol{\pi}$, where

$$\pi_n = \boldsymbol{p}_1(n) \overset{\text{def}}{=} \Pr(\mathbf{Z}_1 = \mathbf{x}(n)) \overset{\text{def}}{=} \tilde{v}_n \,, \tag{35}$$

and the transition probabilities are given by

$$P_{nn'} \stackrel{\text{def}}{=} \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n') | \mathbf{Z}_t = \mathbf{x}(n)) \stackrel{\text{def}}{=} (1-\epsilon)\tilde{\gamma}_{n,n'}/\tilde{v}_n + \epsilon\tilde{v}_{n'} \underset{\lim_{\epsilon\to 0}}{=} \tilde{\gamma}_{n,n'}/\tilde{v}_n \,, \tag{36}$$

$$\stackrel{(35)}{\Longrightarrow} \quad \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n'), \mathbf{Z}_t = \mathbf{x}(n)) \;=\; (1-\epsilon)\tilde{\gamma}_{n,n'} + \epsilon\tilde{v}_n\tilde{v}_{n'} \underset{\lim_{\epsilon\to 0}}{=} \tilde{\gamma}_{n,n'} \,, \tag{37}$$

(for $\mathbf{Z}_t$ stationary) with $0 < \epsilon \ll 1$. Due to the ϵ-term all states of the Markov chain can transition to all other states including themselves, which makes the Markov chain irreducible and aperiodic, and therefore ergodic. Thus, the stationary distribution is unique and the Markov chain converges to it. The normalization restrictions (31), (32), and (34) ensure the normalization of (35), (36), and (37), respectively.

It is easy to see that $\boldsymbol{\pi} = \{\tilde{v}_n\}_{n=1}^N$ is indeed a stationary distribution, since for $\mathbf{p}_t(n) = \tilde{v}_n$

$$\mathbf{p}_{t+1}(n) \;=\; \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n)) \;=\; \sum_{n'} \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n) | \mathbf{Z}_t = \mathbf{x}(n')) \Pr(\mathbf{Z}_t = \mathbf{x}(n')) \tag{38}$$

$$\stackrel{(35,36)}{=} \sum_{n'} \left((1-\epsilon)\left(\tilde{\gamma}_{n',n}/\tilde{v}_{n'}\right) + \epsilon\tilde{v}_n\right)\tilde{v}_{n'} \tag{39}$$

$$\stackrel{(31,33)}{=} (1-\epsilon)\tilde{v}_n + \epsilon\tilde{v}_n \;=\; \tilde{v}_n \;=\; \mathbf{p}_t(n) \,. \tag{40}$$

The time average of the input sequence is

$$\boldsymbol{\mu}_Z \stackrel{\text{def}}{=} \langle \mathbf{Z}_t \rangle_t \tag{41}$$

$$= \langle \mathbf{Z} \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{42}$$

$$\stackrel{(40)}{=} \sum_n \tilde{v}_n \mathbf{x}(n) \tag{43}$$

$$\stackrel{(13)}{=} \hat{\mathbf{x}} \,, \tag{44}$$

and the covariance matrix is

$$\mathbf{C} \stackrel{\text{def}}{=} \langle (\mathbf{Z}_t - \boldsymbol{\mu}_Z)(\mathbf{Z}_t - \boldsymbol{\mu}_Z)^T \rangle_t \tag{45}$$

$$\stackrel{(44)}{=} \langle (\mathbf{Z} - \hat{\mathbf{x}})(\mathbf{Z} - \hat{\mathbf{x}})^T \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{46}$$

$$\stackrel{(40)}{=} \sum_n \tilde{v}_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T \tag{47}$$

$$\stackrel{(12)}{=} \mathbf{C}_{\mathbf{G}} \,, \tag{48}$$

whereas the derivative second-moment matrix is

$$\dot{\mathbf{C}} \stackrel{\text{def}}{=} \langle \dot{\mathbf{Z}}_t \dot{\mathbf{Z}}_t^T \rangle_t \tag{49}$$

$$= \langle \dot{\mathbf{Z}} \dot{\mathbf{Z}}^T \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{50}$$

$$\stackrel{(37)}{=} \sum_{n,n'} \left((1-\epsilon)\tilde{\gamma}_{n,n'} + \epsilon \tilde{v}_n \tilde{v}_{n'} \right) (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T , \tag{51}$$

where $\dot{\mathbf{Z}}_t \stackrel{\text{def}}{=} \mathbf{Z}_{t+1} - \mathbf{Z}_t$. Notice that $\lim_{\epsilon \to 0} \dot{\mathbf{C}} \stackrel{(51)}{=} \tilde{\gamma}_{n,n'} (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T \stackrel{(14)}{=} \dot{\mathbf{C}}_{\mathbf{G}}$. Therefore, if a graph fulfills the normalization restrictions (31–34), GSFA yields the same features as standard SFA on the sequence generated by the Markov chain, in the limit $\epsilon \to 0$.

Probabilistic interpretation of a connected graph. The vanishing parameter ϵ guarantees transitions between arbitrary samples. If the graph is connected, one can assume that $\epsilon = 0$ and simplify the equations above. For connected graphs, GSFA yields the same features as standard SFA trained on a sequence generated by using the graph as a Markov chain with transition probabilities $\gamma_{n,n'}/R$. Thus, one can use SFA trained on a single sequence (in the limit of infinite length) to emulate GSFA. However, such an emulation may be more computationally expensive than directly using GSFA.

3.2.6 Construction of Training Graphs

It is possible, in principle, to construct training graphs with arbitrary connections and weights. However, when the goal is to solve a supervised learning task, the graph edges should implicitly integrate the label information taking into account whether the goal is classification or regression. Each case is addressed separately in the next two sections. The proposed training graphs have been implemented in Python (source code available upon request) and their usefulness on real-world data has been verified (Escalante-B. and Wiskott, 2010; Mohamed and Mahdi, 2010; Escalante-B. and Wiskott, 2012).

3.3 Classification with GSFA

This section shows how to use GSFA to profit from the label information and solve classification tasks more efficiently and accurately than with standard SFA.

3.3.1 Clustered Training Graph

To generate features useful for classification, the use of a *clustered training graph* is proposed. This graph is presented below and illustrated in Figure 3.4. Assume there are C identities/classes, and for each particular identity $c = 1, \ldots, C$ there are N_c samples $\mathbf{x}^c(n)$, where $n = 1, \ldots, N_c$, making a total of $N = \sum_c N_c$ samples. The clustered training graph is defined as a graph $G = (\mathbf{V}, \mathbf{E})$ with

vertices $\mathbf{V} = \{\mathbf{x}^c(n)\}$, and edges $\mathbf{E} = \{(\mathbf{x}^c(n), \mathbf{x}^c(n'))\}$ for $c = 1, \ldots, C$, and $n, n' = 1, \ldots, N_c$. Thus all pairs of samples of the same identity are connected, while samples of different identity are not connected. Vertex weights are identical and equal to one, that is, $\forall c, n : \; v_n^c = 1$. In contrast, edge weights, $\gamma_{n,n'}^c = 1/N_c \;\; \forall n, n'$, depend on the cluster size[3]. Otherwise identities with a large N_c would be over-represented because the number of edges in the complete subgraph for identity c grows quadratically with N_c. These weights directly fulfill the normalization restriction (32). As usual, a trivial scaling of the vertex and edge weights suffices to fulfill restrictions (31) and (34), allowing the probabilistic interpretation of the graph. The optimization problem associated with this graph explicitly demands that samples from the same object identity should be typically mapped to similar outputs.

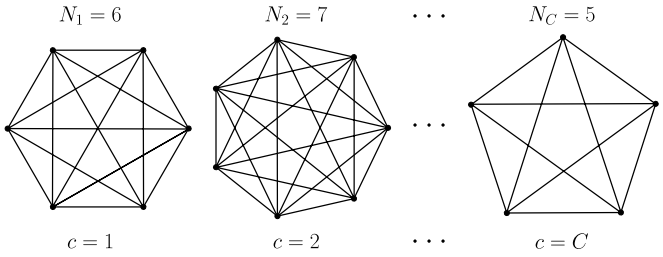


Figure 3.4: Illustration of a *clustered* training graph used for a classification problem with C classes. Each vertex represents a sample, and edges represent transitions. The N_c samples belonging to a class $c \in \{1, \ldots, C\}$ are connected, constituting a fully connected subgraph. Samples of different classes are not connected. Vertex weights are identical and equal to one, whereas edge weights depend on the cluster size as $\gamma_{n,n'} = 1/N_c$, where $\mathbf{x}(n)$ and $\mathbf{x}(n')$ belong to class c and $n \neq n'$. Self-loops (connecting each sample with itself) can be safely discarded without altering the computed features and are not displayed here.
[Based on a figure from (Escalante-B. and Wiskott, 2013).]

3.3.2 Efficient Learning Using the Clustered Graph

At first sight, the large number of edges, $\sum_c N_c(N_c+1)/2$, seems to introduce a computational burden. It is shown here that this is not the case if one exploits the symmetry of the graph. From (12), the sample covariance matrix of this

[3]These vertex and edge weights assume that the classification of all samples is equally important. In the alternative case that classification over every cluster is equally important, one can set $v_n^c = 1/N_c$ and $\forall n, n' : \; \gamma_{n,n'}^c = (1/N_c)^2$ instead.

graph using the vertex weights $v_n^c = 1$ is (notice the definition of Π^c and $\hat{\mathbf{x}}^c$):

$$\mathbf{C}_{\text{clus}} \overset{(12)}{=} \frac{1}{Q}\Big[\underbrace{\sum_c \sum_{n=1}^{N_c} \mathbf{x}^c(n)(\mathbf{x}^c(n))^T}_{\overset{\text{def}}{=}\Pi^c} - Q\Big(\underbrace{\frac{1}{Q}\sum_c \overbrace{\sum_{n=1}^{N_c} \mathbf{x}^c(n)}^{\overset{\text{def}}{=}N_c\hat{\mathbf{x}}^c}}_{\overset{(13)}{=}\hat{\mathbf{x}}}\Big)\cdot \Big(\frac{1}{Q}\sum_c \sum_{n=1}^{N_c} \mathbf{x}^c(n)\Big)^T\Big], \tag{52}$$

$$= \frac{1}{Q}\Big(\sum_c \Pi^c - Q\hat{\mathbf{x}}\hat{\mathbf{x}}^T\Big), \tag{53}$$

where $Q \overset{(11)}{=} \sum_c \sum_{n=1}^{N_c} 1 = \sum_c N_c = N$.

From (14), the derivative second-moment matrix of the clustered training graph using edge weights $\gamma_{n,n'}^c = 1/N_c$ is:

$$\dot{\mathbf{C}}_{\text{clus}} \overset{(14)}{=} \frac{1}{R}\sum_c \frac{1}{N_c} \sum_{n,n'=1}^{N_c} (\mathbf{x}^c(n') - \mathbf{x}^c(n))(\mathbf{x}^c(n') - \mathbf{x}^c(n))^T, \tag{54}$$

$$= \frac{1}{R}\sum_c \frac{1}{N_c} \sum_{n,n'=1}^{N_c} \big(\mathbf{x}^c(n')(\mathbf{x}^c(n'))^T + \mathbf{x}^c(n)(\mathbf{x}^c(n))^T - \mathbf{x}^c(n')(\mathbf{x}^c(n))^T - \mathbf{x}^c(n)(\mathbf{x}^c(n'))^T\big), \tag{55}$$

$$\overset{(52)}{=} \frac{1}{R}\sum_c \frac{1}{N_c}\Big(N_c \sum_{n=1}^{N_c} \mathbf{x}^c(n)(\mathbf{x}^c(n))^T + N_c \sum_{n'=1}^{N_c} \mathbf{x}^c(n')(\mathbf{x}^c(n'))^T - 2N_c\hat{\mathbf{x}}^c(N_c\hat{\mathbf{x}}^c)^T\Big), \tag{56}$$

$$\overset{(52)}{=} \frac{2}{R}\sum_c \big(\Pi^c - N_c\hat{\mathbf{x}}^c(\hat{\mathbf{x}}^c)^T\big), \tag{57}$$

where $R \overset{(10)}{=} \sum_c \sum_{n,n'} \gamma_{n,n'}^c = \sum_c \sum_{n,n'} 1/N_c = \sum_c (N_c)^2/N_c = \sum_c N_c = N$.

The complexity of computing $\mathbf{C}_{\text{clus}}$ using (52) or (53) is the same, namely $\mathcal{O}(\sum_c N_c)$ operations. However, the complexity of computing $\dot{\mathbf{C}}_{\text{clus}}$ can be reduced from $\mathcal{O}(\sum_c N_c^2)$ operations directly using (54) to $\mathcal{O}(\sum_c N_c)$ operations using (57). This algebraic simplification allows us to compute $\dot{\mathbf{C}}_{\text{clus}}$ with a complexity linear in N (and N_c), resulting in an important speedup since, depending on the application, N_c might be larger than 100 and sometimes even $N_c > 1000$.

Interestingly, one can show that the features learned by GSFA on this graph are equivalent to those learned by FDA (see Section 3.6).

3.3.3 Supervised Step for Classification Problems

Consistent with FDA, the theory of SFA using an unrestricted function space (optimal free responses) predicts that, for this type of problem, the first $C-1$ slow features extracted are orthogonal step functions, and are piece-wise constant for samples from the same identity (Berkes, 2005a). This closely approximates what has been observed empirically, which can be informally described as features that are approximately constant for samples of the same identity, with moderate noise.

When the features extracted are close to the theoretical predictions (e.g., their Δ-values are small), their structure is simple enough that one can use even a modest supervised step after SFA, such as a nearest centroid or a Gaussian classifier (in which a Gaussian distribution is fitted to each class) on $C-1$ slow features or fewer.

Previous (undocumented) experimental results of my research project have shown that Gaussian classifiers provide better robustness than the nearest centroid classifier when enough training data is available. While a more powerful classification method, such as an SVM, might also be used, experiments carried out have only found a small increase in estimation accuracy at the cost of longer training times.

3.4 Regression with SFA and GSFA

The objective in regression problems is to learn a mapping from samples to labels providing the best estimation as measured by a loss function, for example, the root mean squared error (RMSE) between the estimated labels, $\hat{\ell}_1, \ldots, \hat{\ell}_N$, and their ground-truth values, $\ell_1, \ldots, \ell_N$. It is assumed here that the loss function is an increasing function of the absolute estimation error $|\hat{\ell} - \ell|$ (e.g., contrary to periodic functions, useful to compare angular values, or arbitrary functions of $\hat{\ell}$ and ℓ).

Regression problems can be addressed with SFA through multiple methods. The fundamental idea is to treat labels as the value of a hidden slow parameter that we want to learn. In general, SFA will not extract the label values exactly. However, optimization for slowness implies that samples with similar label values are typically mapped to similar output values. After SFA reduces the dimensionality of the data, a complementary explicit regression step on a few features solves the original regression problem.

In this section, four SFA-based methods that explicitly use available labels are proposed. The first method is called *sample reordering* and employs standard SFA, whereas the remaining ones employ GSFA with three different training graphs called *sliding window*, *serial*, and *mixed* training graphs (Sections 3.4.1–3.4.4). The selection of the explicit regression step for post-processing is discussed in Section 3.4.5.

3.4.1 Sample Reordering

Let $\mathbf{X}' = (\mathbf{x}'(1), \ldots, \mathbf{x}'(N))$ be a sequence of N data samples with labels $\boldsymbol{\ell}' = (\ell'_1, \ldots, \ell'_N)$. The samples are reordered by means of a permutation $\pi(\cdot)$ in such a way that the labels become monotonically increasing. The reordered samples are $\mathbf{X} = (\mathbf{x}(1), \ldots, \mathbf{x}(N))$, where $\mathbf{x}(n) = \mathbf{x}'(\pi(n))$, and their labels are $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_N)$ with $\ell_l \leq \ell_{l+1}$. Afterwards the sequence $\mathbf{X}$ is used to train standard SFA using the regular single-sequence method (Figure 3.5).

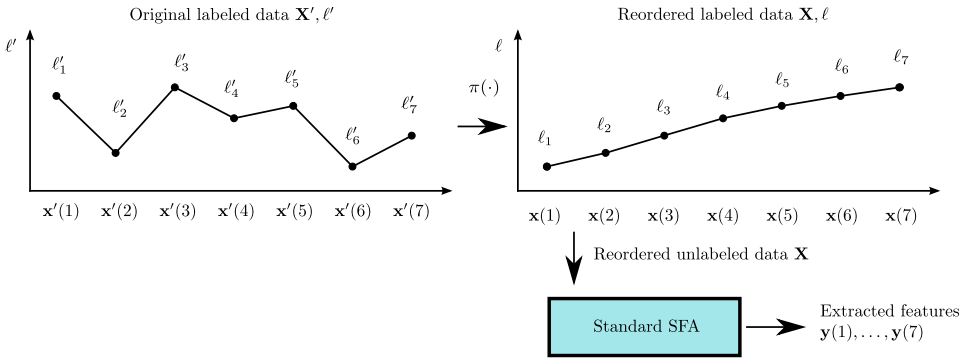


Figure 3.5: Sample reordering approach. Standard SFA is trained with a reordered sample sequence, in which the hidden labels are increasing.
[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

Since the ordered label values only increase, they change very slowly and should be found by SFA (or actually some increasing/decreasing function of the labels that fulfills the normalization conditions). Clearly, SFA can extract this information only if the samples indeed contain information about the labels such that it is possible to extract the labels from them. Due to limitations of the feature space considered, insufficient data, noise, etc., one typically obtains noisy and distorted versions of the predicted signals.

In this basic approach, the computation of the covariance matrices takes $\mathcal{O}(N)$ operations. Since this method only requires standard SFA and is the most straightforward to implement, its use is recommended for first experiments. If more robust outputs are desired, the methods below based on GSFA are more appropriate.

3.4.2 Sliding Window Training Graph

This graph is an improvement over the reordering graph above in which GSFA facilitates the consideration of more connections. Starting from the reordered sequence $\mathbf{X}$ as defined above, a training graph is constructed, in which each sample $\mathbf{x}(n)$ is connected to its d closest samples to the left and to the right in the order given by $\mathbf{X}$. Thus, $\mathbf{x}(n)$ is connected to the samples $\mathbf{x}(n-d), \ldots, \mathbf{x}(n-1)$, $\mathbf{x}(n+1), \ldots, \mathbf{x}(n+d)$ (Figure 3.6.a). In this graph, the vertex weights are constant, that is, $v_n = 1$, and the edge weights typically depend on the distance of

the samples involved, that is, $\forall n,n' :\ \gamma_{n,n'} = f(|n' - n|)$, for some function $f(\cdot)$ that specifies the shape of a "weight window". The simplest case is a square weight window defined by $\gamma_{n,n'} = 1$ if $|n' - n| \leq d$ and $\gamma_{n,n'} = 0$ otherwise. The experiments described later in this chapter employ a *mirrored* sliding window graph with edge weights

$$\gamma_{n,n'} = \begin{cases} 2, & \text{if } n + n' \leq d + 1 \text{ or } n + n' \geq 2N - 1\,, \\ 1, & \text{if } |n' - n| \leq d,\ n + n' > d + 1 \text{ and } n + n' < 2N - 1\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{58}$$

These weights compensate the limited connectivity of the few first and last samples (which are connected to at least d and at most $2d-1$ samples) in contrast to intermediate samples (connected to $2d$ samples). Preliminary experiments suggest that such compensation slightly improves the quality of the extracted features, as explained below.

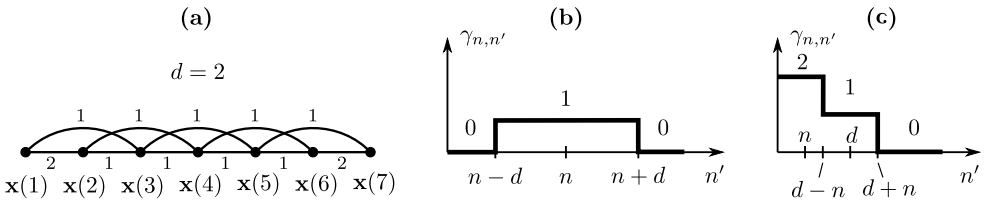


Figure 3.6: (a) A mirrored square sliding window training graph with a half-width of $d = 2$. Each vertex is thus adjacent to at most 4 other vertices. (b) Illustration of the edge weights of an intermediate vertex $\mathbf{x}(n)$ for an arbitrary window half-width d. (c) Edge weights for a vertex $\mathbf{x}(n)$ close to the left extreme ($n < d$). Notice that the sum of the edge weights is also approximately $2d$ for extreme vertices.

[Based on a figure from (Escalante-B. and Wiskott, 2013).]

GSFA is guaranteed to find functions that minimize (6) within the function space considered. However, how good such a solution is for the regression problem largely depends on how one has defined the weights of the training graph. For instance, if there is a sample with a large vertex weight that has only weak connections to the other samples, an optimal but undesired solution might assign a high positive value to that single sample and negative small values to all other samples. Such output values can satisfy the zero mean and unit variance constraint while yielding a small Δ-value, because the large differences in output value only occur at the weak connections. Thus, such a solution is good in terms of the GSFA optimization problem but not good w.r.t. the regression problem at hand, because the samples with small values are hard to discriminate. In what follows, this type of solutions is referred to as pathological. Pathological solutions have certain similarities to the features obtained for classification, which are approximately constant for each cluster (class) but discontinuous among them.

The occurrence of pathological solutions depends on the concrete data samples, feature space, and training graph. To avoid pathological solutions, a necessary condition is that the graph is connected because, as discussed in Section 3.3, for disconnected graphs GSFA has a strong tendency to produce a representation suitable for classification rather than regression. Various experiments conducted for this PhD project have shown that it is useful to enforce the normalization restriction (32) at least approximately (after vertex and edge weights have been normalized). Such a restriction ensures that the samples are connected strongly enough to other samples, relative to their own vertex weight. Of course, one should not resort to self-loops (i.e., $\gamma_{n,n} \neq 0$) to trivially fulfill the restriction.

In the sliding window training graph with arbitrary window, the computation of $\mathbf{C_G}$ and $\dot{\mathbf{C}}_\mathbf{G}$ requires $\mathcal{O}(dN)$ operations. If the window is square (mirrored or not), the computation can be improved to $\mathcal{O}(N)$ operations by using accumulators for sums and products and reusing intermediate results. While larger d implies more connections, which is generally good, connecting too distant samples is undesired. The selection of d is not crucial and done empirically.

3.4.3 Serial Training Graph

The *serial* training graph is similar to the clustered training graph (used for classification) in terms of its structure and efficiency. It results from discretizing the original labels $\boldsymbol{\ell}$ into a relatively small set of discrete labels $\{\ell_1, \ldots, \ell_L\}$ of size L, where $\ell_1 < \ell_2 < \cdots < \ell_L$. As described below, faster training is achieved if L is small, for example, $3 \leq L \ll N$.

In the serial graph, the vertices are grouped according to their discrete labels. Every sample in the group with label ℓ_l is connected to every sample in the groups with label ℓ_{l+1} and ℓ_{l-1} (except the samples in the first and last groups, which can only be connected to one neighboring group). The only existing connections are inter-group connections, no intra-group connections are present (a graph with intra-group connections is discussed later).

The samples used for training are denoted by $\mathbf{x}^l(n)$, where the index l ($1 \leq l \leq L$) denotes the group (discrete label) and n ($1 \leq n \leq N_l$) denotes the sample within such a group. For simplicity, it is assumed here that all groups have the same number N_g of samples: $\forall l:\ N_l = N_g$. Thus, the total number of samples is $N = LN_g$. The vertex weight of $\mathbf{x}^l(n)$ is denoted by v_n^l, where $v_n^l = 1$ for $l \in \{1,L\}$ and $v_n^l = 2$ for $1 < l < L$. The edge weight of the edge $(\mathbf{x}^l(n), \mathbf{x}^{l+1}(n'))$ is denoted by $\gamma_{n,n'}^{l,l+1}$, and all connections have the same edge weight: $\forall n,n',l:\ \gamma_{n,n'}^{l,l+1} = 1$. Thus, all edges have a weight of 1, and all samples are assigned a weight of 2 except for the samples in the first and last groups, which have a weight of 1 (Figure 3.7). The reason for the different weights in the first and last groups is to improve feature quality by enforcing the normalization restriction (32) (after vertex and edge weight normalization). Notice that since any two vertices of the same group are adjacent to exactly the same neighbors, they are likely to be mapped to similar outputs by GSFA.

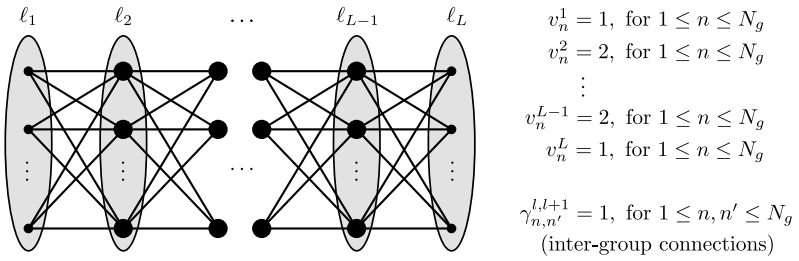


Figure 3.7: Illustration of a serial training graph with L discrete label values $\{\ell_1, \ldots, \ell_L\}$. Even though the original labels of two samples might differ, they will be grouped together if they have the same discrete label. Each dot represents a sample, edges represent connections, and ovals represent groups of samples. The samples of groups with discrete labels ℓ_2 to ℓ_{L-1} have a weight of 2, whereas the samples of extreme groups with labels ℓ_1 or ℓ_L have a weight of 1 (sample weights are represented by bigger/smaller dots). The weight of all edges is 1.
[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

The sum of vertex weights is $Q \overset{(11)}{=} N_g + 2N_g(L-2) + N_g = 2N_g(L-1)$ and the sum of edge weights is $R \overset{(10)}{=} (L-1)(N_g)^2$, which is also the number of connections considered. Not surprisingly, the structure of the graph can be exploited to train GSFA efficiently. Similarly to the clustered training graph, one can define the average of the samples from the group l as $\hat{\mathbf{x}}^l \overset{\text{def}}{=} \sum_n \mathbf{x}^l(n)/N_g$, the sum of the products of samples from group l as $\Pi^l = \sum_n x^l(n)(x^l(n))^T$, and the weighted sample average as:

$$\hat{\mathbf{x}} \overset{\text{def}}{=} \frac{1}{Q} \sum_n \left(\mathbf{x}^1(n) + \mathbf{x}^L(n) + 2 \sum_{l=2}^{L-1} \mathbf{x}^l(n) \right) \tag{59}$$

$$= \frac{1}{2(L-1)} \left(\hat{\mathbf{x}}^1 + \hat{\mathbf{x}}^L + 2 \sum_{l=2}^{L-1} \hat{\mathbf{x}}^l \right) . \tag{60}$$

From (12), the sample covariance matrix accounting for the weights v_n^l of the serial training graph is:

$$\mathbf{C}_{\text{ser}} \overset{(12,59)}{=} \frac{1}{Q} \left(\sum_n \mathbf{x}^1(n)(\mathbf{x}^1(n))^T + 2 \sum_{l=2}^{L-1} \sum_n \mathbf{x}^l(n)(\mathbf{x}^l(n))^T + \sum_n \mathbf{x}^L(n)(\mathbf{x}^L(n))^T - Q\hat{\mathbf{x}}(\hat{\mathbf{x}})^T \right) \tag{61}$$

$$= \frac{1}{Q} \left(\Pi^1 + \Pi^L + 2 \sum_{l=2}^{L-1} \Pi^l - Q\hat{\mathbf{x}}'(\hat{\mathbf{x}}')^T \right) . \tag{62}$$

From (14), the matrix $\dot{\mathbf{C}}_{\mathbf{G}}$ computed using the edges $\gamma_{n,n'}^{l,l+1}$ defined above is:

$$\dot{\mathbf{C}}_{\text{ser}} \overset{(14)}{=} \frac{1}{R}\sum_{l=1}^{L-1}\sum_{n,n'}\left(\mathbf{x}^{l+1}(n') - \mathbf{x}^{l}(n)\right)\left(\mathbf{x}^{l+1}(n') - \mathbf{x}^{l}(n)\right)^T \tag{63}$$

$$= \frac{1}{R}\sum_{l=1}^{L-1}\sum_{n,n'}\Big(\mathbf{x}^{l+1}(n')(\mathbf{x}^{l+1}(n'))^T + \mathbf{x}^{l}(n)(\mathbf{x}^{l}(n))^T - \mathbf{x}^{l}(n)(\mathbf{x}^{l+1}(n'))^T - \mathbf{x}^{l+1}(n')(\mathbf{x}^{l}(n))^T\Big) \tag{64}$$

$$= \frac{1}{R}\sum_{l=1}^{L-1}\left(\sum_{n'}\left(\Pi^{l+1} + \Pi^{l}\right) - \Big(\sum_{n}\mathbf{x}^{l}(n)\Big)\Big(\sum_{n'}\mathbf{x}^{l+1}(n')\Big)^T - \Big(\sum_{n'}\mathbf{x}^{l+1}(n')\Big)\Big(\sum_{n}\mathbf{x}^{l}(n)\Big)^T\right) \tag{65}$$

$$= \frac{N_g}{R}\sum_{l=1}^{L-1}\left(\Pi^{l} + \Pi^{l+1} - N_g\hat{\mathbf{x}}^{l}(\hat{\mathbf{x}}^{l+1})^T - N_g\hat{\mathbf{x}}^{l+1}(\hat{\mathbf{x}}^{l})^T\right). \tag{66}$$

By using (66) instead of (63), the slowest step in the computation of the covariance matrices, which is the computation of $\dot{\mathbf{C}}_{\text{ser}}$, can be reduced in complexity from $\mathcal{O}(L(N_g)^2)$ to only $\mathcal{O}(N)$ operations ($N = LN_g$), which is of the same order as the computation of $\mathbf{C}_{\text{ser}}$. Thus, for the same number of samples N, one obtains a larger speed-up for larger group sizes.

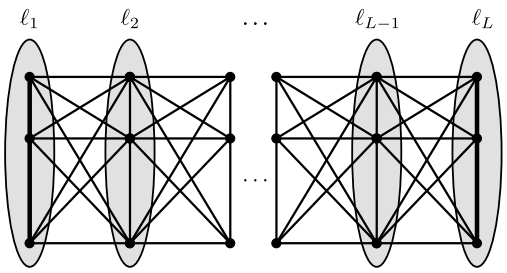
Discretization introduces some type of quantization error. While a large number of discrete labels L results in a smaller quantization error, having too many of them is undesired because fewer edges would be considered, which would increase the number of samples needed to reduce the overall error. For example, in the extreme case of $N_g = 1$ and $L = N$, this method does not bring any benefit because it is almost equivalent to the sample reordering approach (differing only due to the smaller weights of the first and last sample).

3.4.4 Mixed Training Graph

The serial training graph does not have intra-group connections, and therefore the output differences of samples with the same label are not explicitly being minimized. In some cases, particularly for small numbers of training samples, additional intra-group connections might indeed improve robustness. However, one argument against intra-group connections is that if two vertices are adjacent to the same set of vertices, their corresponding samples are already likely to be mapped to similar outputs. Clearly, these are reasonable arguments both in favor and against intra-group connections. To resolve whether these connections are indeed useful, it will be resorted in this chapter to concrete experiments. However, this question is addressed more theoretically in Section 4.4.2, where an analytic method is used to compute a different representation of the graphs.

The *mixed* training graph (Figure 3.8) has been conceived to test the effect

of having both inter- and intra-group connections. This graph is a combination of the serial and clustered training graph that fulfills the consistency restriction (32). All vertices and edges of this graph have a weight of 1, except for the intra-group edges in the first and last groups, which have a weight of 2. As expected, the computation of the covariance matrices can also be done efficiently for this training graph (details omitted).



$v_n^l = 1$, for $1 \le n \le N_g$ and $1 \le l \le L$

For $1 \le n, n' \le N_g$:

$\left.\begin{array}{l} \gamma_{n,n'}^{1,1} = 2 \\ \gamma_{n,n'}^{l,l} = 1, \text{ for } 2 \le l \le L-1 \\ \gamma_{n,n'}^{L,L} = 2 \end{array}\right\}$ (intra-group connections)

$\gamma_{n,n'}^{l,l+1} = 1$, for $1 \le l \le L-1$ (inter-group connections)

Figure 3.8: Illustration of the mixed training graph. Each group of samples having the same discrete label is fully connected (intra-group connections, represented with vertical edges) and also all samples of adjacent groups are connected (inter-group connections). All vertex and edge weights are equal to 1 except for the intra-group edge weights of the first and last groups, which are equal to 2 and represented by thick lines.
[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

3.4.5 Supervised Step for Regression Problems

There are at least three approaches to implement the supervised step on top of (G)SFA to learn a mapping from slow features to the labels. The first one is to use a method such as linear or nonlinear regression. The second one is to discretize the original labels to a small discrete set $\{\tilde{\ell}_1, \ldots, \tilde{\ell}_{\tilde{L}}\}$ (which might be different from the discrete set used by the training graphs). The discrete labels are then treated as classes, and a classifier is trained to predict them from the slow features. One can then output the predicted class as the estimated label. Of course, an error due to the discretization of the labels is unavoidable. The third approach improves on the second one by using a classifier that also estimates class membership probabilities. Let $\mathbf{P}(C_{\tilde{\ell}_l}|\mathbf{y})$ be the estimated class probability that the input sample $\mathbf{x}$ with slow features $\mathbf{y} = \mathbf{g}(\mathbf{x})$ belongs to the group with (discretized) label $\tilde{\ell}_l$. Class probabilities can be used to provide a more robust estimation of a soft (continuous) label ℓ, better suited to the particular loss function. For instance, one can use

$$\ell \stackrel{\text{def}}{=} \sum_{l=1}^{\tilde{L}} \tilde{\ell}_l \cdot \mathbf{P}(C_{\tilde{\ell}_l}|\mathbf{y}) \tag{67}$$

if the loss function is the RMSE, where the slow features $\mathbf{y}$ might be extracted using any of the four SFA-based methods for regression above. Other loss functions can be addressed in a similar way. For instance, the optimal label estimation when using the mean average error (MAE) is not the mean value but the median, computed according to the estimated distribution.

During this research, these three approaches have been tested and different classifiers have been used, such as nearest neighbor, nearest centroid, Gaussian classifier, and SVMs. The results of such experiments suggest that the most convenient approach is to use the soft labels computed from the class probabilities estimated by a Gaussian classifier because in most of the experiments this method has provided best performance and robustness. Of course, other classifiers providing class probabilities could also be used.

3.5 Experimental Evaluation of the Graphs

In this section, the performance of the supervised learning methods based on SFA (sample reordering) and GSFA presented above (using various graphs) is evaluated. Two concrete machine learning problems on image analysis are considered, both of them on real photograph databases, the first one for classification and the second one for regression.

3.5.1 Classification

The clustered training graph has been proposed in this work to solve classification problems. As mentioned in Section 3.3.2, when this graph is used, the outputs of GSFA are equivalent to those of FDA. Since FDA has been used and evaluated exhaustively, here it is only verified that the implementation of GSFA (as part of the Cuicuilco framework, explained in Appendix A) generates the expected results when trained with such a graph.

The German Traffic Sign Recognition Benchmark (Stallkamp et al., 2011) is used for the experimental test. This is a competition that had the goal of classifying photographs of 43 different traffic signs taken on German roads under uncontrolled conditions with variations in lighting, sign size, and distance. No detection step is necessary because the true position of the signs is included as annotations, making this a pure classification task and ideal for our test. More specifically, I participated in the online version of the competition, where 26,640 labeled images have been provided for training and 12,569 images without label for evaluation (classification rates were computed by the organizers, who had ground-truth data).

Two-layer nonlinear cascaded (non-hierarchical) SFA was employed. To achieve good performance, the choice of the nonlinear expansion function is crucial. If it is too simple (e.g., low-dimensional), it does not solve the problem; if it is too complex (e.g., high-dimensional), it might overfit to the training data and not generalize well to test data. In all experiments done in this chapter, a

compact expansion that only doubles the data dimension has been employed, which has been called 0.8Exp,

$$0.8\text{Exp}: \quad \mathbf{x}^T \mapsto \mathbf{x}^T, (|\mathbf{x}|^{0.8})^T, \tag{68}$$

where the absolute value and exponent 0.8 are computed component-wise. This expansion has been proposed by Escalante-B. and Wiskott (2011), who have reported that it offers good generalization and competitive performance in SFA networks, presumably due to its robustness to outliers and certain properties regarding the approximation of higher frequency harmonics.

The network above, complemented by a Gaussian classifier on 42 slow features, has achieved a recognition rate of 96.4% on test data[4]. This, as expected, was similar to the reported performance of various methods based on FDA participating in the same competition. For comparison, human performance is 98.81%, and a convolutional neural network has given top performance with a 98.98% recognition rate.

3.5.2 Regression

The remaining training graphs have all been designed for regression problems and were evaluated with the problem of estimating the horizontal position of a face in frontal face photographs, an important regression problem because it can be used as a component of a face detection system, as shown in Figure 2.4. For a slightly more detailed description of such a system see the evaluation of Mohamed and Mahdi (2010). As previously mentioned, it is proposed to decompose face detection into the problems of the estimation of the horizontal position of a face, its vertical position, and its size. Afterwards, face detection is refined by locating each eye more accurately with the same approach applied now to the eyes instead of to the face centers. Below, this regression problem is explained, the algorithms are evaluated, and the results are presented in more detail.

Problem and Dataset Description

To increase image variability and improve generalization, face images from several databases are used, namely 1,521 images from BioID (Jesorsky et al., 2001), 9,030 from CAS-PEAL (Gao et al., 2008), 5,479 from Caltech (Fink et al., 2003), 9,113 from FaceTracer (Kumar et al., 2008), and 39,328 from FRGC (Phillips et al., 2005) making a total of 64,471 images, which have been automatically preprocessed through a pose-normalization and a pose-reintroduction step. In the

[4]Interestingly, GSFA did not provide best performance directly on the pixel data, but on precomputed HOG features. Ideally, pre-processing is not needed if GSFA has an unrestricted feature space. In practice, knowing a good low-dimensional set of features for the particular data is beneficial. Applying GSFA to such features, as commonly done with other machine learning algorithms, can reduce overfitting.

first step, each image is converted to grayscale and pose-normalized using annotated facial points so that the face is centered[5], has a fixed eyes-mouth-triangle area, and the resulting pose-normalized image has a resolution of 256×192 pixels. In the second step, horizontal and vertical displacements are re-introduced, as well as scalings, so that the center of the face deviates horizontally at most ±45 pixels from the center of the image. The vertical position and the size of the face are randomized, so that vertically the face center deviates at most ±20 pixels, and the smallest faces are half the size of the largest faces (a ratio of at most 1 to 4 in area). Interpolation (e.g., needed for scaling and sub-pixel displacements) is done using bicubic interpolation. At last, the images are cropped to 128×128 pixels (see Figure 3.9).

Figure 3.9: Example of a pose-normalized image (left) and various images after pose was reintroduced, illustrating the final range of vertical and horizontal displacements, as well as the face sizes (right).
[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

Given a pre-processed input image, as described above, with a face at position (x, y) w.r.t. the image center and size z, the regression problem is then to estimate the x-coordinate of the center of the face. The range of the variables x, y and z is bounded to a box, so that one does not have to consider extremely small faces, for example. To assure a robust estimation for new images, invariance to a large number of factors is needed, including the vertical position of the face, its size, the expression and identity of the subject, his or her accessories, clothing, hair style, the lighting conditions, and the background.

The pose-normalized images are randomly split in three datasets of 30,000, 20,000 and 9,000 images. The first dataset was used to train the dimensionality reduction method, the second one to train the supervised post-processing step, and the last one for testing. To further exploit the images available, the pose-normalized images of each dataset are duplicated, resulting in two pose-reintroduced images per input image. A single input image exclusively belongs to one of the three datasets, appearing twice in it with two different poses. Hence, the final size of the datasets is 60,000, 40,000 and 18,000 pre-processed images, respectively.

[5]The center of a face was defined here as $\frac{1}{4}\mathbf{LE} + \frac{1}{4}\mathbf{RE} + \frac{1}{2}\mathbf{M}$, where $\mathbf{LE}$, $\mathbf{RE}$ and $\mathbf{M}$ are the coordinates of the centers of the left eye, right eye and mouth, respectively. Thus, the face center is the midpoint between the mouth and the midpoint of the eyes.

Dimensionality-Reduction Methods

The resolution of the images and their number make it less practical to directly apply SFA and the majority of supervised methods, such as an SVM, and unsupervised methods, such as PCA/ICA/LLE, to the raw images. This problem is circumvented by using three efficient dimensionality reduction methods, and by applying supervised processing on the lower-dimensional features extracted. The first two methods are efficient hierarchical implementations of SFA and GSFA (referred to as HSFA in this chapter without distinction). The nodes in the HSFA networks first expand the data using the 0.8Exp expansion function (see Section 3.5.1) and then apply SFA/GSFA to it, except for the nodes in the first layer in which additionally PCA is applied before the expansion to preserve 13 out of 16 principal components. For comparison, we use a third method, a hierarchical implementation of PCA (HPCA), in which all nodes do pure PCA. The structure of the hierarchies for the HSFA and PCA networks is described in Table 3.1. In contrast to other HSFA networks (e.g., Franzius et al., 2007), weight-sharing was not used at all, improving feature specificity at the lowest layers. The input to the nodes (fan-in) comes mostly from two nodes in the previous layer. This small fan-in reduces the computational cost because the input dimensionality is minimized. This also results in networks with a large number of layers potentiating the accumulation of non-linearity across the network. Non-overlapping receptive fields are used because such a small fan-in showed good performance and a smaller computational cost in previous experiments of this research using similar data.

The following dimensionality-reduction methods are evaluated (one based on SFA, four based on GSFA, and one based on PCA).

- SFA using sample reordering (reordering).
- GSFA with a mirrored sliding window graph with $d = 32$ (MSW32).
- GSFA with a mirrored sliding window graph with $d = 64$ (MSW64).
- GSFA with a serial training graph with $L = 50$ groups of $N_g = 600$ images (serial).
- GSFA with a mixed graph and the same number of groups and images (mixed).
- A hierarchical implementation of PCA (HPCA).

It is impossible to compare GSFA against all the dimensionality reduction and supervised learning algorithms available, and therefore a small subset of them has been selected. HPCA has been chosen for efficiency reasons and because it is likely to be a good dimensionality reduction algorithm for the problem at hand since principal components code well the coarse structure of the image including the silhouette of the subjects, allowing for a good estimation of the position of the face. Thus, HPCA (combined with various supervised algorithms) appears to be a fair point of comparison, and a good representative among

Layer	size	node fan-in	output dimensionality of the nodes HSFA	HPCA
0 (input image)	128×128 pixels	—	—	—
1	32×32 nodes	4×4	13	13
2	16×32 nodes	2×1	20	20
3	16×16 nodes	1×2	35	35
4	8×16 nodes	2×1	60	60
5	8×8 nodes	1×2	60	100
6	4×8 nodes	2×1	60	120
7	4×4 nodes	1×2	60	120
8	2×4 nodes	2×1	60	120
9	2×2 nodes	1×2	60	120
10	1×2 nodes	2×1	60	120
11 (top node)	1×1 nodes	1×2	60	120

Table 3.1: Structure of the SFA and PCA deep hierarchical networks. The networks only differ in the type of processing done by each node and in the number of features preserved. For HSFA an upper bound of 60 features was set, whereas for HPCA at most 120 features were preserved. A node with a fan-in of $a \times b$ is driven by a rectangular array of nodes (or pixels for the first layer) that has such a shape and is located in the preceding layer.

generic machine learning algorithms for this problem. For the data employed, 120 HPCA features at the top node explain 88% of the data variance, suggesting that HPCA is indeed a good approximation to PCA in this case.

The evolution across the hierarchical network of the two slowest features extracted by HSFA is illustrated in Figure 3.10.

Supervised Post-Processing Algorithms

On top of the dimensionality reduction methods, the following supervised post-processing algorithms are employed.

- A nearest centroid classifier (NCC).
- Labels estimated using (67) and the class membership probabilities given by a Gaussian classifier (Soft GC).
- A multi-class (one-versus-one) SVM (Chang and Lin, 2011) with a Gaussian radial basis kernel, and grid search for model selection.
- Linear regression (LR).

To train the classifiers, the images of the second dataset are grouped in 50 equally large classes depending on their horizontal displacement x, where $-45 \leq x \leq 45$. The classifiers are then trained with these 50 virtual classes.

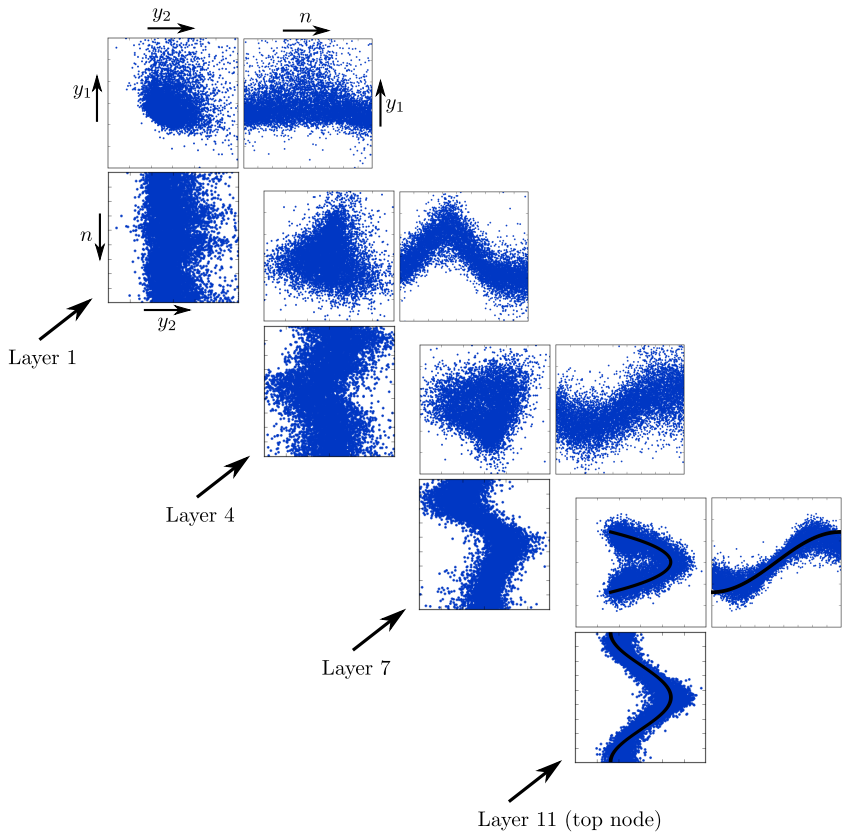


Figure 3.10: Evolution of the slow features extracted from test data after layers 1, 4, 7 and 11 of a GSFA network trained with the serial training graph. A central node has been selected from each layer and the first two features are shown in three plots, that is, y_2 vs y_1, n vs y_1, and y_2 vs n. Hierarchical processing results in progressively slower features as the data is processed from the first to the last layer (i.e., the top node). The solid line in the plots of the top node represents the optimal free responses, which are the slowest possible features one can obtain using an unrestricted mapping, as predicted theoretically (Wiskott, 2003a). Notice how the features evolve from being mostly unstructured in the first layer to being similar to the optimal free responses at the top node, indicating success at finding the hidden parameter changing most slowly for this data (i.e., the horizontal position of the faces).

[Figure reproduced from (Escalante-B. and Wiskott, 2013).]

Results

The evaluation comprises all combinations of a dimensionality reduction method (reordering, MSW32, MSW64, serial, mixed and HPCA) and a supervised post-processing algorithm (NCC, Soft GC, SVM, LR). Their performance is measured on test data and reported in terms of the RMSE. The estimated labels depend on a few parameters: the number of features passed to the supervised post-

processing algorithm, and the parameters C and γ in the case of the SVM. These parameters are optimized for each combination of algorithms using a single trial per tested parameter value, but the RMSEs reported here are averaged over 5 trials.

The results are presented in Table 3.2 and are analyzed focusing on four aspects: the dimensionality-reduction method, the number of features used, the supervised methods, and the training graphs. For any choice of the post-processing algorithm and training graph, GSFA resulted in an RMSE 5% to 13% smaller than when using the basic reordering of samples employing standard SFA. In turn, reordering resulted in an RMSE at least 10% better for this dataset than when using HPCA.

Dim. red. method	**NCC** (RMSE)	# of feat.	**Soft GC** (RMSE)	# of feat.	**SVM** (RMSE)	# of feat.	**LR** (RMSE)	# of feat.	Wall time
Reord.	6.16	6	5.63	4	6.00	14	10.23	60	51
MSW32	5.78	5	5.25	4	5.52	18	9.74	60	49
MSW64	5.69	5	5.15	4	5.38	18	9.69	60	62
Serial	**5.58**	4	**5.03**	5	**5.23**	15	9.68	60	62
Mixed	5.63	4	5.12	4	5.40	19	**9.54**	60	55
HPCA	29.68	118	6.17	54	8.09	50	19.24	120	18

Table 3.2: Performance (RMSE, measured in pixels) of the dimensionality reduction algorithms in combination with various supervised algorithms for the post-processing step. The RMSE at chance level is 25.98 pixels. Each entry reports the best performance achievable using a different number of features and parameters in the post-processing step. Largest standard error of the mean is 0.21 pixels. Clearly, linear regression benefits from all SFA and PCA features available. Running times (in minutes) are reported using a newer version of the software and a computer with 24 Xeon X7542 cores @ 2.67GHz.

Taking a look at the number of features used by each supervised post-processing algorithm, one can observe that considerably fewer HSFA features are used than HPCA features (e.g., 5 vs. 54 for Soft GC). This can be explained because PCA is sensitive to many factors that are irrelevant to solve the regression problem, such as the vertical position of the face, its scale, the background, lighting, etc. Thus, the information that encodes the horizontal position of a face is mixed with other information and distributed over many principal components, whereas it is more concentrated in the slowest components of SFA/GSFA.

If one focuses on the post-processing methods, one can observe that linear regression performed poorly, confirming that a linear supervised step is too weak, particularly when the dimensionality reduction is also linear (e.g., HPCA). The nearest centroid classifier did modestly for HSFA, but even worse than chance level for HPCA. The SVMs are consistently better, but the error can be further reduced by 4% to 23% by using Soft GC, the soft labels derived from a Gaussian classifier.

Regarding the training graphs, I expected that the sliding window graphs, MSW32 and MSW64, would be more accurate than the serial and mixed graphs, even when using a square window, because the labels are not discretized. Surprisingly, the mixed and serial graphs were the most accurate ones. This might be explained in part by the larger number of connections in these graphs. Still, MSW32 and MSW64 were better than the reordering approach, the wider window being superior. The RMSE of the serial graph was smaller than the one of the mixed graph by less than 2% (for Soft GC), making it uncertain for statistical reasons which of these graphs is better for this problem. A larger number of trials, or even better, a more detailed mathematical analysis of the graphs might be necessary to determine which one is better (see Section 4.4.2).

3.6 Discussion of GSFA

In this chapter, the graph-based SFA (GSFA) optimization problem has been proposed, an extension to the standard SFA optimization problem that takes into account label and sample information specified in a structure called training graph, in which the vertices are the training samples and the edges represent connections between samples. Edge weights allow the specification of desired output similarities and can be derived from label or class information. The GSFA optimization problem generalizes the notion of slowness defined originally for a plain sequence of samples.

The GSFA algorithm has also been proposed, which is an implicitly supervised extension of the (unsupervised) SFA algorithm, and it is shown that GSFA solves the new optimization problem in the considered function space. The main goal of GSFA is to solve supervised learning problems by reducing the dimensionality of the data to a few very label-predictive features.

The term "implicitly supervised" has been used to qualify GSFA to emphasize that the labels themselves are never provided to GSFA, but only the training graphs, which encode the labels through their structure. While the construction of the graph is a supervised operation, GSFA works in an unsupervised fashion on structured data. Hence, GSFA does not search for a fit to the labels explicitly but instead fully concentrates on the generation of slow features according to the topology defined by the graph.

Several training graphs for classification and regression have been introduced in this chapter. They have been designed aiming at a balance between computational requirements and accuracy. These graphs offer a significant advantage in terms of speed, for example, over other similarity matrices typically used with LPP. Conceptually, such a speed-up can be traced back to two factors that originate from the highly regular structure of the graphs (Sections 3.3 and 3.4). (1) The determination of the edges and edge weights is a trivial operation because they are derived from the labels in a simple manner. In contrast, this operation can be quite expensive if the connections are computed using nearest neighbor algorithms. (2) Linear algebra can be used to optimize the computation

of $\dot{\mathbf{C}}$ (as already shown), which is needed during the training phase. The resulting complexity for training is linear in the number of samples, even though the number of connections considered is quadratic. The experimental results demonstrate that the larger number of connections considered by GSFA indeed provides a more robust learning than standard SFA, making it superior to SFA in supervised learning settings.

When solving a concrete supervised learning problem, the features extracted by unsupervised dimensionality reduction algorithms are often suboptimal. For instance, PCA does not yield good features for age estimation from adult face photographs because features revealing age (e.g., skin textures) have higher spatial frequencies and do not belong to the main principal components. Supervised dimensionality reduction algorithms, including GSFA, are especially promising when one does not know a good set of features for the particular data and problem at hand and one wants to improve performance by generating features adapted to the specific data and labels.

One central idea of this chapter, shown in Figure 3.1, is the following. If one has a large number of high-dimensional labeled data, supervised learning algorithms can often not be applied due to high computational requirements. In such cases, it is suggested to transform the labeled data to structured data, where the label information is implicitly encoded in the connections between data points. Then, unsupervised learning algorithms, such as SFA, or its implicitly supervised extension GSFA, can be used. This permits hierarchical processing for dimensionality reduction, an operation that is frequently more difficult with supervised learning algorithms. The resulting low-dimensional data has an intrinsic structure that reflects the graph topology. This data can then be transformed back to labeled data by adding the labels, and standard supervised learning algorithms can be applied to solve the original supervised learning problem.

3.6.1 Related Optimization Problems

Recently, Böhmer et al. (2012) introduced regularized sparse kernel SFA. The algorithm is used to solve a classification problem by reducing the data dimensionality. In the discussion section, various extensions similar to the GSFA optimization problem are briefly presented without empirical evaluation. For classification, the authors propose an objective function equivalent to (6), with edge weights $\gamma_{n,n'} = \delta_{c_n c_{n'}}$, where c_n and $c_{n'}$ are the classes of the respective samples, and $\delta_{c_n c_{n'}}$ is the Kronecker delta. If all classes $C_1, \ldots, C_S$ are equally represented by N_c samples, such edge weights are equivalent to those specified by the clustered graph. However, if N_s is not the same for all classes, the binary edge weights (either 1 or 0, as given by the Kronecker delta) seem to be less appropriate because larger classes are overrepresented by the quadratic number of edges $N_c(N_c+1)/2$ for class s. The authors also consider transitions with variable weights. For this purpose, they use an importance measure $p(\mathbf{x}_{t+1}, \mathbf{x}_t) \geq 0$ with high values for transitions within the desired subspace and propose the

objective function

$$\min\ s'(y_i) \stackrel{\text{def}}{=} \frac{1}{n-1}\sum_{t=1}^{n-1}\frac{(y_i(t+1)-y_i(t))^2}{p(\mathbf{x}_{t+1},\mathbf{x}_t)}\,, \tag{69}$$

with $y_i(t) \stackrel{\text{def}}{=} \phi_i(\mathbf{x}(t))$. This accounts for arbitrary edge weights $\gamma_{n,n+1}$ in a linear graph, which could be easily generalized to arbitrary graphs. It is not clear to me why the importance measure p has been introduced as a quotient instead of as a factor. The authors also propose an importance measure $q(\mathbf{x})$ for the samples, which plays exactly the same role as the vertex weights $\{v_n\}_n$. The unit variance and decorrelation constraints are adapted to account for $q(\mathbf{x})$ and become fully equivalent to constraints (8–9) of GSFA. The remaining zero-mean constraint was not explicitly adapted.

3.6.2 Conversions Between GSFA and Similar Algorithms

Zhang et al. (2009) propose a framework for the systematic analysis of several dimensionality reduction algorithms, such as LLE, LE, LPP, PCA and LDA, to name just a few. From a mathematical point of view, SFA, LPP, genSFA and GSFA belong to the same family of optimization problems (regardless of their typical usage and application areas) and can be solved via generalized eigenvalues. Conversions between these four algorithms are possible.

LPP and SFA. The authors show that LPP is a linearization of LE. In turn, linear SFA can be seen as a special case of LPP, where the weight matrix has a special form (see Sprekeler, 2011). Consider the following LPP optimization problem (He and Niyogi, 2003):

$$\arg\min_{\substack{\mathbf{y}\\ \mathbf{y}^T D\mathbf{y}=1}} \frac{1}{2}\sum_{i,j}(y_i-y_j)^2 W_{ij} = \mathbf{y}^T\mathbf{L}\mathbf{y}\,, \tag{70}$$

where $\mathbf{W}$ is a symmetric weight matrix, $\mathbf{D}$ is a diagonal matrix with diagonal $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij}$, $\mathbf{L} \stackrel{\text{def}}{=} \mathbf{D}-\mathbf{W}$ is the Laplacian matrix, and the output features $\mathbf{y} \stackrel{\text{def}}{=} \mathbf{a}^T\mathbf{x}$ are linear in the input. The equivalence to linear SFA is achieved if one sets $\mathbf{W}$ as $W_{ij} = \frac{1}{2}(\delta_{i,1}\delta_{j,1} + \delta_{i,N}\delta_{j,N} + \delta_{j,i+1} + \delta_{i,j+1})$. The objective function then becomes the same as in SFA, and $\mathbf{D}$ becomes the identity matrix, yielding the same restrictions as in SFA. The zero-mean constraint is implicit and achieved by discarding a constant solution with eigenvalue zero. Notice that leaving out the terms $\delta_{i,1}\delta_{j,1} + \delta_{i,N}\delta_{j,N}$ results in $\mathbf{D} = \mathrm{diag}(1/2, 1, 1, \ldots, 1, 1/2)$ being slightly different from the identity matrix and the equivalence above would only be approximate.

LPP and Generalized SFA. Sprekeler (2011) studied the relation between SFA and LE and proposed the combination of the neighborhood relation used by

LE and the functional nature of SFA. The resulting algorithm, called generalized SFA (genSFA), computes a functional approximation to LE with a feature space spanned by a set of basis functions. Linear genSFA is equivalent to linear LPP (Rehn, 2013), and in general it can be regarded as LPP on the data expanded by such basis functions.

LPP and GSFA. Also the strong connection between LPP and linear GSFA is evident from (70). In this case, the elements D_{ii} play the role of the vertex weights v_i, and the elements W_{ij} play the role of the edge weights $\gamma_{i,j}$. One difference between LPP and GSFA is that the additional normalization factors Q and R of GSFA provide invariance to the scale of the edge and vertex weights specifying a particular feature and objective function normalization. A second difference is that for GSFA the vertex weights, fundamental for feature normalization, can be chosen independently from the edge weights (unless one explicitly enforces (32)), whereas for LPP the diagonal elements $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij}$ are derived from the edge weights.

Now it is shown how one can easily compute LPP features using GSFA. Given a similarity matrix W_{ij} and diagonal matrix $\mathbf{D}$ with diagonal elements $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij}$, one solves a GSFA problem defined over a training graph with the same samples, edge weights $\gamma_{i,j} = W_{ij}$, and vertex weights $v_i = D_{ii}$. The optimization problem solved by GSFA is then equivalent to the LPP problem, except for the scale of the objective function and the features. If the features extracted from a particular sample are denoted by $\mathbf{y}_{\text{GSFA}}$, one can match the feature scales of LPP simply by applying a scaling $\mathbf{y} = Q^{-1/2}\mathbf{y}_{\text{GSFA}}$, where $Q \stackrel{(11)}{=} \sum_i v_i$.

It is also possible to use LPP to compute GSFA features. Given a training graph with edge weights $\gamma_{i,j}$ and vertex weights v_i, we can define the following similarity matrix:

$$W_{ij} = \begin{cases} 2\gamma_{i,j}/R, & \text{for } i \neq j, \text{ with } R \text{ as defined in (10)}, \\ v_i/Q - \sum_{j' \neq i} W_{ij'}, & \text{for } i = j, \text{ with } Q \text{ as defined in (11)}. \end{cases} \tag{71}$$

The similarity matrix $\mathbf{W}$ above ensures the same objective function as in GSFA, and also that $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij} = v_i/Q$, resulting in the same constraints. In practice, using self-loops $W_{ii} \neq 0$ is unusual for LPP, but they are useful in this case.

In spite of being closely related mathematically, SFA, LPP, genSFA and GSFA originate from different backgrounds and were first motivated with different goals in mind. Therefore, they usually have different applications and are trained with different similarity matrices, resulting in features with different properties. For instance, LPP originates from the field of manifold learning and transitions are typically defined from input similarities (nearest neighbors). SFA originates from unsupervised learning and transitions are typically defined by the temporal ordering of the samples. genSFA typically combines input sim-

ilarities with class information, although more recently an approach uses only the label information (Rehn and Sprekeler, 2014). In GSFA, transitions typically reflect label similarities. For SFA and GSFA the use of input similarities might be disadvantageous, because it might compromise the invariance of the extracted features, which is one of the central goals.

3.6.3 Remarks on Classification with GSFA

Both, classification and regression can be solved with GSFA. The results show that a few slow features allow for an accurate solution to the supervised learning problem, requiring only a small post-processing step that maps the features to labels or classes.

For classification problems, the clustered training graph is proposed, which yields features having the discriminative capability of FDA. The results of the implementation of this graph confirm the expectations from theory. Training with the clustered graph is equivalent to considering all transitions between samples of the same identity and no transition between different identities. The computation, however, is more efficient than the direct method of Berkes (2005a), where a large number of transitions have to be considered explicitly.

The Markov chain generated through the probabilistic interpretation of this graph is equal to the Markov chain defined by Klampfl and Maass (2010). These two Markov chains are parameterized by vanishing parameters ϵ and a, respectively. The parameter a influences the probability $P_{ij} = aN_j/N$ of transitioning from a class c_i to a different class c_j, where N_j is the number of samples of class c_j and N is the total number of samples. Thus, in the limit $a \to 0$ all transitions lie within the same class. However, the Markov chain and ϵ (see Equation 36) are introduced here for analytical purposes only. In practice, GSFA directly uses the graph structure, which is deterministic and free of ϵ. This avoids the less efficient training of SFA with a very long sequence of samples generated with the Markov chain, as done by Klampfl and Maass (2010). (Even though the set of samples is finite, as the parameter a approaches 0, an infinite sequence is required to properly capture the data statistics of all identities).

Klampfl and Maass (2010) have proven that if $a \to 0$ the features learned by SFA from the data generated are equivalent to the features learned by FDA. From the equality of the two Markov chains above, the features extracted by GSFA turn out to be also equivalent to those of FDA. Thus, the features extracted with GSFA from the clustered graph are not better or worse than those extracted with FDA. However, this equivalence is an interesting result because it allows a different interpretation of FDA from the point of view of the extraction of slow signals. Moreover, advances in generic methods for SFA, such as hierarchical network architectures or robust nonlinearities, result in improved classification rates over standard FDA.

It is possible to design other training graphs for classification without the equivalence to FDA, for example, by using non-constant sample weights, or by incorporating input similarity information or other criteria in the edge-weight

matrix. This idea has been explored by Rehn (2013) using genSFA, where various adjacency graphs (i.e., edge weights) were proposed for classification inspired by the theory of manifold learning. Instead of using full-class connectivity, only samples within the same class among the first nearest neighbors are connected. Less susceptibility to outliers and better performance have been reported.

3.6.4 Remarks on Regression with GSFA

To solve regression problems, I have proposed three training graphs for GSFA that resulted in a reduction of the RMSE of up to 11% over the basic reordering approach using standard SFA. Such an improvement is caused by the higher number of similarity relations considered even though the same number of training samples is used.

Extensions of SFA for regression (i.e., GSFA) have first been used by Escalante-B. and Wiskott (2010) to estimate age and gender from frontal static face images of artificial subjects, created with special software for 3D face modeling and rendering, see Figure 2.3. Gender estimation is treated as a regression problem, because the software represents gender as a continuous variable (e.g., -1.0=typical masculine face, 0.0=neutral, 1.0=typical feminine face). Early versions of the mixed and serial training graphs were employed. Only three extracted features were passed to an explicit regression step based on a Gaussian classifier (Section 3.4.5). In both cases, good performance has been achieved, with an RMSE of 3.8 years for age and 0.33 units for gender on test data, compared to a chance level of 13.8 years and 1.73 units, respectively.

With a system similar to the one presented here, I participated in a face detection competition successfully (Mohamed and Mahdi, 2010). As described in Section 2.6.4, the x-position, y-position and scale of a face within an image are estimated using three separate SFA networks, one for each parameter, to pose-normalize the faces within image patches. A fourth network is trained to estimate the quality of the normalization, again as a continuous parameter, and to indicate whether a face is present at all. These four networks together were used to detect faces. Performance of the resulting face detection system on various image databases was competitive and yielded a detection rate on grayscale photographs within the range from 71.5% to 99.5% depending on the difficulty of the test images. An increase in the image variability in the training data is expected to result in further improvements.

The serial and mixed training graphs have resulted in the best accuracy for the experiments in this chapter, with the serial one being slightly more accurate but not to a significant level. These graphs have the advantage over the sliding window graph that they are also suitable for cases in which the labels are discrete from the beginning, which occurs frequently due to finite resolution in measurements or due to discrete phenomena (e.g., when learning the number of red blood cells in a sample, or a distance in pixels).

Since the edge weights supported by GSFA can be chosen arbitrarily, it is tempting to use a complete weight matrix continuous in the labels (e.g., $\gamma_{n,n'} =$

$\frac{1}{|\ell_{n'}-\ell_n|+k}$, for $k > 0$, or $\gamma_{n,n'} = \exp(-\frac{(\ell_{n'}-\ell_n)^2}{\sigma^2})$). However, this might affect the training time markedly. Moreover, one should be aware that imbalances in the connectivity of the samples might result in pathological solutions or features less useful than expected.

The emphasis of this chapter has been put on supervised dimensionality reduction towards the estimation of a single label. However, one can estimate two or more labels simultaneously using appropriate training graphs. In general, using such graphs might reduce the performance of the method compared to the separate estimation of the labels. However, if the labels are intrinsically related, performance might actually improve.

The features extracted by GSFA strongly depend on the labels, even though label information is only provided implicitly by the graph connectivity. Ideally, the slowest feature extracted is a monotonic function of the hidden label, and the remaining features are harmonics of increasing frequency of the first one. In practice, noisy and distorted versions of these features are found, but still providing an approximate, redundant, and concentrated coding of the labels. Their simple structure permits the use of simple supervised algorithms for the post-processing step saving time and computer resources. For the estimation of the x-position of faces, all the nonlinear post-processing algorithms, including the nearest centroid classifier, provided good accuracy. Although a Gaussian classifier is a less powerful classifier than an SVM, the estimation based on the class membership probabilities of the Gaussian classifier (Soft GC) is more accurate because it reduces the effect of miss-classifications.

3.6.5 Other Considerations

Locality preserving projections and GSFA come from very different backgrounds. On the one hand, LPP is motivated from unsupervised learning and was conceived as a linear algorithm. The similarity matrices used are typically derived from the training samples, for example, using a heat kernel function. Later, weight matrices accounting for label information have been proposed, particularly for classification. On the other hand GSFA is motivated from supervised learning, and was conceived as an extension of SFA designed for supervised non-linear dimensionality reduction specifically targeting regression and classification problems. Although the motivation behind LPP and GSFA, as well as their typical applications, are different, these algorithms are strongly connected. Therefore, in future work it might be worth not only to unify their formalism, but also the conceptual roots that have inspired them.

Although supervised learning is less biologically plausible, GSFA being implicitly supervised is still closely connected to feasible unsupervised biological models through the probabilistic interpretation of the graph. (If one ensures that the graph fulfills the normalization restrictions, the Markov chain described in Section 3.2.5 can be constructed, and learning with GSFA and such graph becomes equivalent to learning with standard (unsupervised) SFA if it is trained with a sequence generated by the Markov chain.) From this perspective, GSFA

uses the graph information to simplify a learning procedure that could also be done in an unsupervised fashion.

This research shows that GSFA is better for supervised learning than SFA, and suggests that it is a very interesting and promising algorithm. Of course, specialized algorithms might outperform GSFA for particular tasks. For instance, algorithms for face detection can outperform the system presented here, but a fundamental advantage of GSFA is that it is general purpose. Moreover, various improvements to GSFA are possible. Some of them are discussed in Chapters 4 and 5, which will increase its performance and provide accuracies similar or even better than special-purpose algorithms.

One limitation of hierarchical processing with GSFA or SFA (i.e., HSFA) is that the features should be spatially localized in the input data. For instance, if one randomly shuffles the pixels in the input image, performance would decrease considerably. This limits the applicability of HSFA for scenarios with heterogeneous independent sources of data, but makes it well suited, for example, for images.

Although GSFA makes a more efficient use of the available samples than SFA, it can still overfit in part because these algorithms lack an explicit regularization parameter. Hence, for a small number of samples data randomization techniques are useful. Interestingly, certain expansions and hierarchical processing can be seen as implicit regularization measures. In fact, less overfitting compared to standard SFA is one of the central advantages of using HSFA. A second central advantage of HSFA is that HSFA networks can be trained in a feed-forward manner layer by layer, resulting in a very efficient training. Moreover, the nonlinearities of different layers accumulate, and the features at the top node of the network can be highly nonlinear w.r.t. the input, potentially spanning a rich feature space.

Most of this work was originally motivated by the need to improve generalization of current SFA networks. Of course, if the amount of training data and computation time were unrestricted, overfitting would be negligible, and all SFA training methods would approximately converge to the same features and provide similar performance. For finite data and resources, the results demonstrate that GSFA does provide better performance than SFA (reordering method) using the same amount of training data. Another interpretation is that GSFA demands less training data to achieve the same performance, thus, indeed contributing to the original goal of improving generalization.

Chapter 4

Exact Label Learning: Theoretical Analysis of the Optimal Free Responses of Graph-Based SFA for the Design of Training Graphs

The previous chapter introduced graph-based SFA (GSFA), a *supervised* extension of SFA that can be used to solve regression problems if followed by a simple post-processing step (e.g., a classification or regression algorithm, such as a Gaussian classifier or ordinary least squares). Training graphs are a key concept of GSFA. They define the value of the vertex and edge weights. Edge weights allow the specification of arbitrary connections between the training samples and define an output similarity objective. Various training graphs for regression have been proposed, namely, the reordering, serial, mixed, and sliding window graphs. Results show that the serial graph is particularly promising due to its efficiency and the label estimation accuracy. One disadvantage of the graphs mentioned above, however, is that the samples are connected by edges that depend only on the *rank* of the corresponding labels. Exploiting the exact label values allows further improvements in estimation accuracy.

In this chapter, I propose the exact label learning (ELL) method[1]. The ELL method constructs a graph that encodes the desired label accurately and allows GSFA to extract a normalized version of it directly. The ELL method is used for three tasks: (1) Gender estimation from artificial images of human faces (regression), which shows the advantage of encoding additional labels, particularly skin color. (2) The analysis of two existing graphs for regression. (3) The ELL method is used in a non-conventional way to extract *compact* discriminative

[1]This chapter is an edited version of (Escalante-B. and Wiskott, 2016b), which has been published in the Journal of Machine Learning Research (JMLR).

features and classify traffic sign images. When the number of output features is fewer than $C-1$, compact discriminative features yield a higher classification rate than a training graph that generates features equivalent to those of nonlinear Fisher discriminant analysis, where C is the number of classes. The method is versatile, directly supports multiple labels, and provides higher accuracy compared to previously proposed graphs for the problems considered.

The next section provides a high-level introduction to the ELL method. Then, the GSFA optimization problem is expressed in matrix notation to simplify its analysis. Afterwards, the ELL method is proposed. Thereafter, the method is validated and applied to three different tasks. Finally, the chapter is closed with a discussion.

4.1 Introduction

Supervised learning problems on simple data (e.g., low-dimensional and/or artificial data) can usually be solved by the direct application of conventional supervised learning algorithms, e.g., an SVM applied directly to the pixel data. However, such methods are mostly inefficient or ineffective for real-world high-dimensional data. A classical approach to deal with high-dimensional data is to apply feature extraction followed by (unsupervised) dimensionality reduction (DR) and an explicit supervised learning step (Figure 4.1.a). In Chapter 3, a new approach has been proposed that uses GSFA for *supervised* DR and then post-processes a small number of slow features with a conventional classification or regression algorithm (Figure 4.1.b). An advantage of this approach is that the supervised learning problem is mostly solved by GSFA implicitly, because it usually identifies and compresses the label-predictive information in a small number of features. Therefore, an advanced post-processing step is not crucial, and a simple mapping from slow features to labels may suffice.

From now on, we will refer to graphs with a fixed shape that only takes into account the rank of the labels and not their exact value as *pre-defined* graphs. These include all the graphs proposed in Chapter 3. The number of edges represented by pre-defined graphs is $\mathcal{O}(N^2)$, where N is the number of samples. However, their structure allows quite efficient training methods that have linear complexity w.r.t. N.

GSFA trained with pre-defined graphs provides much better label estimations than standard SFA. However, the edge weights of pre-defined graphs are defined in a somewhat primitive manner. The exact value of the labels can be exploited to carefully define the weights of the training graph, allowing further improvements in estimation accuracy. For instance, the serial graph is suboptimal, since edge weights are either 0 or 1 and the graph incurs in a form of label quantization error. This chapter focuses on the analysis and design of training graphs. A new approach for solving regression problems with GSFA is proposed. Such an approach is based on the construction of a special training graph, in

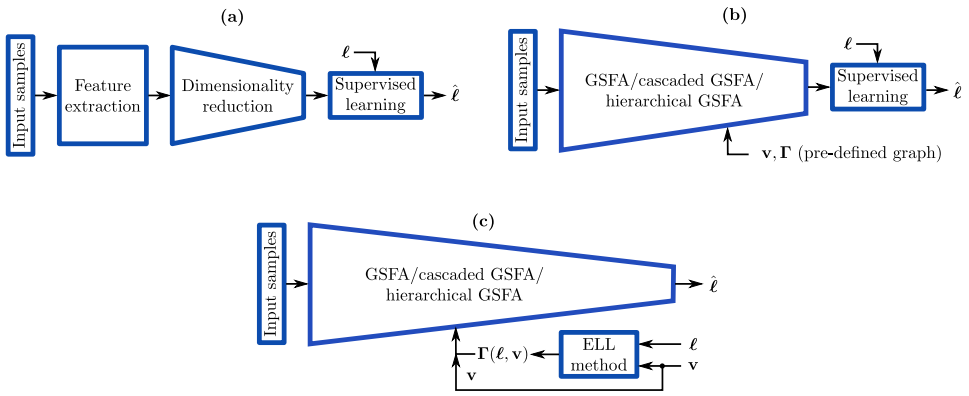


Figure 4.1: Three approaches for solving supervised learning problems. $\boldsymbol{\ell}$ denotes ground-truth labels, $\hat{\boldsymbol{\ell}}$ are label estimations, $\mathbf{v}$ are the vertex weights, and $\boldsymbol{\Gamma}$ is a training graph. (a) A classical approach. (b) Previous approach using GSFA with a pre-defined training graph. The samples are assumed to be ordered by increasing label. (c) The approach proposed here, which consists of a single GSFA architecture that is trained with a specially constructed graph $\boldsymbol{\Gamma}(\boldsymbol{\ell}, \mathbf{v})$, i.e., an explicit function of $\boldsymbol{\ell}$ and $\mathbf{v}$. The first slow feature (with a global sign adjustment) directly provides the label estimation. If the label $\boldsymbol{\ell}$ does not have weighted zero mean and weighted unit variance, a final affine transformation (scaling and offset) should be included.
[Based on a figure from (Escalante-B. and Wiskott, 2016b)].

which the slowest feature extracted is already a label estimation, up to an affine transformation (Figure 4.1.c).

A graph designed with the proposed ELL method can be used to train GSFA, cascaded GSFA, and hierarchical GSFA (HGSFA). In fact, the main application area of the ELL method is the solution of regression problems on high-dimensional data with HGSFA. The resulting system efficiently learns a nonlinear mapping from the input data (e.g., the pixels or features) to label estimations.

The first step to develop this exact label learning (ELL) method, is to analyze the slowest possible features that can be extracted by GSFA from a given graph when the feature space is unrestricted. These features have also been called optimal free responses and have been computed for SFA in continuous time by Wiskott (2003a) using variational calculus. For GSFA, a different method must be used, because the data has a discrete graph structure and is no longer a continuous function of time.

After the optimal free responses of GSFA have been expressed in a closed form, a theoretical method for the converse operation is developed: From a set of free responses, the corresponding training graph is computed. The method allows the creation of a graph in which the slowest possible feature is the label to be learned. Moreover, one can learn *multiple labels* simultaneously (e.g.,

object position, average color, shape, and size), and balance their importance by setting the value of certain parameters. This property can be exploited to learn *auxiliary labels*, which provide a redundant encoding of the original labels that can potentially increase the estimation accuracy.

A theoretical graph-analysis method is used to analyze the serial graph and ELL graphs. It is shown that these graphs are similar in terms of the first optimal free responses, and that when *only one* label is learned the serial graph may substitute the ELL graph reasonably well with faster training time. However, the chapter's discussion outlines a few extensions to the ELL method towards improving its efficiency.

The ELL method is useful for practical applications, since it provides higher accuracy than pre-defined graphs, particularly when multiple labels are learned. Moreover, the theoretical analysis behind the ELL method provides insights to a deeper understanding of GSFA. While in general the ELL method results in a higher complexity compared to GSFA trained with an efficient pre-defined graph, it is still computationally viable for some datasets without resorting to specialized hardware, multi-threading, or parallelization.

To simplify the analysis of training graphs and the presentation of the ELL method, the next section expresses the GSFA optimization problem in a more compact form.

4.2 GSFA Optimization Problem in Matrix Notation

In order to apply linear algebra methods to analyze GSFA, matrix notation is used. In what follows it is assumed that the edge weights are symmetric[2] ($\mathbf{\Gamma} = \mathbf{\Gamma}^T$) and that the consistency restriction (32) is fulfilled. This restriction can also be written as

$$\mathbf{v} \overset{(32)}{=} \frac{Q}{R}\mathbf{\Gamma 1}\,, \tag{72}$$

where $\mathbf{1}$ is a vector of ones of length N.

If $\mathbf{y}$ is a feasible solution (i.e., satisfying (7) and (8)) and the graph fulfills the consistency restriction (72), the weighted delta value (6) can be simplified

[2] An asymmetric edge-weight matrix $\mathbf{\Gamma}$ can be converted into a symmetric one $\mathbf{\Gamma}' \overset{\text{def}}{=} \frac{\mathbf{\Gamma}+\mathbf{\Gamma}^T}{2}$ without altering the solution to the optimization problem.

as follows,

$$\Delta_{\mathbf{y}} \overset{(6)}{=} \frac{1}{R}\sum_{n,n'}\gamma_{n,n'}(y(n')-y(n))^2 \tag{73}$$

$$= \frac{1}{R}\Big(\sum_{n'}(y(n'))^2\sum_{n}\gamma_{n,n'}+\sum_{n}(y(n))^2\sum_{n'}\gamma_{n,n'}-2\sum_{n,n'}\gamma_{n,n'}y(n')y(n)\Big) \tag{74}$$

$$\overset{(72)}{=} \frac{1}{R}\Big(\sum_{n'}(y(n'))^2\frac{R}{Q}v(n')+\sum_{n}(y(n))^2\frac{R}{Q}v(n)-2\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\Big) \tag{75}$$

$$\overset{(8)}{=} 2-\frac{2}{R}\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\,. \tag{76}$$

The optimization problem can then be stated as follows: For $1 \le j \le J$, find vectors $\mathbf{y}_j$ of length N, with $y_j(n) \overset{\text{def}}{=} g_j(\mathbf{x}(n))$ and $g_j \in \mathcal{F}$, minimizing

$$\Delta_j \overset{(6,8,72)}{=} 2-\frac{2}{R}\mathbf{y}_j^T\mathbf{\Gamma}\mathbf{y}_j \tag{77}$$

subject to:

$$\mathbf{v}^T\mathbf{y}_j \overset{(7)}{=} 0 \tag{78}$$

$$\mathbf{y}_j^T\mathrm{Diag}(\mathbf{v})\mathbf{y}_j \overset{(8)}{=} Q \tag{79}$$

$$\mathbf{y}_j^T\mathrm{Diag}(\mathbf{v})\mathbf{y}_{j'} \overset{(9)}{=} 0,\ \text{for } j'<j\,, \tag{80}$$

where

$$Q \overset{(11)}{=} \mathbf{1}^T\mathbf{v}\,, \tag{81}$$

$$R \overset{(10)}{=} \mathbf{1}^T\mathbf{\Gamma}\mathbf{1}\,, \tag{82}$$

and $\mathrm{Diag}(\mathbf{v})$ denotes a diagonal matrix with diagonal $\mathbf{v}$.

The use of matrix notation facilitates the study of GSFA and the development of the ELL method in the next section.

4.3 Explicit Label Learning for Regression Problems

This section formally proposes the ELL method and is structured as follows. First, the optimal free responses of GSFA are computed for any given training graph. Then, it is shown how to construct a graph useful to learn any particular label or multiple labels. Afterwards, a method is given to convert graphs with negative edge weights into graphs with non-negative weights only (such a method is useful to allow the probabilistic interpretation of the graph and to guarantee that the Δ values of all features lie between 0 and 4). Thereafter, the use of auxiliary labels to improve learning is motivated. Finally, the computational complexity of the ELL method is analyzed.

4.3.1 Optimal Free Responses of GSFA

This section presents a method to calculate the slowest possible solutions (optimal free responses) to the GSFA problem (77)–(80) that one could find if the feature space were unlimited. As we will see, the optimal free responses together with their corresponding Δ values, provide an alternative representation of the training graph and are a useful tool to understand its structure.

The Lagrange multiplier method is used to find critical points $\mathbf{y}$ that are candidates for the optimal free responses. For the moment, the weighted decorrelation constraint (80) is ignored to solve for the first optimal free response, but the remaining responses are considered later. Due to the close relationship between GSFA and LPP, the approach below is strongly related to Laplacian Eigenmaps (Belkin and Niyogi, 2003). Let

$$L \stackrel{\text{def}}{=} \big(2 - \frac{2}{R}\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\big) + \alpha\mathbf{v}^T\mathbf{y} + \beta\big(\mathbf{y}^T\mathrm{Diag}(\mathbf{v})\mathbf{y} - Q\big) \tag{83}$$

be a Lagrangian corresponding to the objective function (77), under the constraints (78) and (79). A signal $\mathbf{y}$ is a critical point if the partial derivatives of L with respect to α, β, and $y(n)$, for $1 \le n \le N$, are simultaneously zero:

$$\partial L/\partial\alpha \stackrel{(83)}{=} \mathbf{v}^T\mathbf{y} \stackrel{!}{=} 0\,, \tag{84}$$

$$\partial L/\partial\beta \stackrel{(83)}{=} \mathbf{y}^T\mathrm{Diag}(\mathbf{v})\mathbf{y} - Q \stackrel{!}{=} 0\,, \text{ and} \tag{85}$$

$$\partial L/\partial\mathbf{y} \stackrel{(83)}{=} -\frac{4}{R}\mathbf{\Gamma}\mathbf{y} + \alpha\mathbf{v} + 2\beta\mathrm{Diag}(\mathbf{v})\mathbf{y} \stackrel{!}{=} \mathbf{0}\,, \tag{86}$$

where $\mathbf{0}$ is a vector of zeros.

Equations (84) and (85) merely require that the output $\mathbf{y}$ has weighted zero mean and weighted unit variance, respectively. Multiplying (86) with $\mathbf{1}^T$ from the left and taking into account that $\mathbf{1}^T\mathrm{Diag}(\mathbf{v}) = \mathbf{v}^T$, $\mathbf{1}^T\mathbf{v} \stackrel{(81)}{=} Q$, $\mathbf{1}^T\mathbf{\Gamma} \stackrel{(72)}{=} \frac{R}{Q}\mathbf{v}^T$, and $Q > 0$ results in:

$$-\frac{4}{R}\Big(\frac{R}{Q}\mathbf{v}^T\Big)\mathbf{y} + \alpha Q + 2\beta\mathbf{v}^T\mathbf{y} = 0\,, \tag{87}$$

implying $\alpha = 0$ due to (84). Therefore, (86) can be simplified to:

$$\Big(-\frac{4}{R}\mathbf{\Gamma} + 2\beta\mathrm{Diag}(\mathbf{v})\Big)\mathbf{y} = \mathbf{0}\,, \tag{88}$$

$$\Leftrightarrow \quad \mathrm{Diag}(\mathbf{v}^{-1/2})\Big(\frac{4}{R}\mathbf{\Gamma} - 2\beta\mathrm{Diag}(\mathbf{v})\Big)\mathrm{Diag}(\mathbf{v}^{-1/2})\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y} = \mathbf{0}\,, \tag{89}$$

$$\Leftrightarrow \quad \Big(\frac{4}{R}\mathrm{Diag}(\mathbf{v}^{-1/2})\,\mathbf{\Gamma}\,\mathrm{Diag}(\mathbf{v}^{-1/2}) - 2\beta\mathbf{I}\Big)\big(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}\big) = \mathbf{0}\,, \tag{90}$$

$$\Leftrightarrow \quad \Big(\mathrm{Diag}(\mathbf{v}^{-1/2})\,\mathbf{\Gamma}\,\mathrm{Diag}(\mathbf{v}^{-1/2}) - \frac{R\beta}{2}\mathbf{I}\Big)\big(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}\big) = \mathbf{0}\,, \tag{91}$$

where $\mathbf{v}^{1/2}$ is defined as the element-wise square root of the elements of $\mathbf{v}$, and $\mathbf{v}^{-1/2}$ is defined similarly (as usual, weights v_j are required to be strictly positive).

In a few words, $\mathbf{y}$ is a critical point if it fulfills the weighted normalization constraints and the vector $\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}$ is an eigenvector of the matrix $\mathbf{M}$ defined as

$$\mathbf{M} \stackrel{\mathrm{def}}{=} \mathrm{Diag}(\mathbf{v}^{-1/2})\, \mathbf{\Gamma}\, \mathrm{Diag}(\mathbf{v}^{-1/2})\,. \tag{92}$$

The corresponding eigenvalue is denoted

$$\lambda = \frac{R\beta}{2}\,. \tag{93}$$

The (orthogonal) eigenvectors of matrix $\mathbf{M}$ are denoted by $\mathbf{u}_j$ with $\mathbf{u}_j^T\mathbf{u}_j = 1$. Each eigenvector $\mathbf{u}_j$ gives rise to a critical point $\mathbf{y}_j \stackrel{\mathrm{def}}{=} Q^{1/2}\mathrm{Diag}(\mathbf{v}^{-1/2})\mathbf{u}_j$ as long as also the weighted normalization constraints (78) and (79) are satisfied by $\mathbf{y}_j$. Constraint (79) is fulfilled by any $\mathbf{y}_j$: $\mathbf{y}_j^T\mathrm{Diag}(\mathbf{v})\mathbf{y}_j = Q\mathbf{u}_j^T\mathrm{Diag}(\mathbf{v}^{-1/2})\mathrm{Diag}(\mathbf{v})\mathrm{Diag}(\mathbf{v}^{-1/2})\mathbf{u}_j = Q\mathbf{u}_j^T\mathrm{Diag}(\mathbf{1})\mathbf{u}_j = Q$. Most $\mathbf{y}_j$ fulfill (78) except for one denoted by $\mathbf{y}_0 \stackrel{\mathrm{def}}{=} \mathbf{1}$ (a vector of ones). This constant and infeasible feature plays the same role as in SFA. The slowest possible solution is the critical point $\mathbf{y}_{j>0}$ with the smallest Δ-value. As shown below, the Δ-value of a critical point $\mathbf{y}_j$ is directly related to the eigenvalue λ_j of the eigenvector $\mathbf{u}_j = Q^{-1/2}\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}_j$ of $\mathbf{M}$ and can be computed as follows.

$$\begin{aligned}
\Delta_{\mathbf{y}_j} &\stackrel{(77)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T\mathbf{\Gamma}\mathbf{y}_j && (94)\\
&\stackrel{(92)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T\mathrm{Diag}(\mathbf{v}^{1/2})\,\mathbf{M}\,\left(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}_j\right) && (95)\\
&\stackrel{(93)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T\mathrm{Diag}(\mathbf{v}^{1/2})\lambda_j\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}_j && (96)\\
&\stackrel{(79)}{=} 2 - \frac{2Q}{R}\lambda_j\,. && (97)
\end{aligned}$$

Thus, the slowest solution is the critical point $\mathbf{y}_j$ with the largest eigenvalue λ_j. The remaining optimal free responses can now be addressed. They are given by the remaining critical points, where their corresponding eigenvalue defines their order, from largest to smallest. The weighted decorrelation condition (80) is fulfilled automatically due to the orthogonality of the eigenvectors: $\mathbf{u}_j^T\mathbf{u}_{j'} = 0 \Leftrightarrow \frac{1}{Q}\mathbf{y}_j^T\mathrm{Diag}(v)\mathbf{y}_{j'} = 0$ (follows from the definition of $\mathbf{y}_j$ above).

One special case is when an eigenvalue has multiplicities. This means that two or more optimal free responses have the same Δ value, which is in fact the same Δ value of any rotation of such free responses. Therefore, optimal free responses with the same Δ value are not uniquely defined and any rotation of them is equivalent.

4.3.2 Design of a Training Graph for Learning One or Multiple Labels

Given a set of samples $\{\mathbf{x}(1), \ldots, \mathbf{x}(N)\}$ with label $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_N)$, it is now shown how to generate a training graph such that the slowest feature that could be extracted by GSFA is equal to a normalized version of the label. Notice that this problem (determining the structure of a training graph, or more concretely, its edge-weight matrix $\mathbf{\Gamma}$, having a particular optimal solution) differs considerably from the original GSFA problem of finding an optimal solution given a training graph and a feature space. The approach can be extended to multiple labels per sample. To distinguish them, the index $1 \leq j \leq L$ is introduced, making $\boldsymbol{\ell}_j$ denote the j-th label. The L labels can then be expressed as an affine transformation of the first L free responses, as described below.

Vertex-weights v_n indicate *a priori* likelihood information about the samples, and are thus assumed to be given and strictly positive. If this information is absent, one may set the vertex weights constant, e.g. $\mathbf{v} = \frac{1}{N}\mathbf{1}$.

Due to the normalization constraints, the outputs generated by GSFA must have weighted zero mean (78) and weighted unit variance (79). Therefore, to learn a single label $\boldsymbol{\ell}$ one can normalize it as follows: Let $\mu_{\boldsymbol{\ell}} = \frac{1}{Q}\mathbf{v}^T\boldsymbol{\ell}$ be the weighted label average and $\sigma^2_{\boldsymbol{\ell}} = \frac{1}{Q}(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1})^T\mathrm{Diag}(\mathbf{v})(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1})$ be the weighted label variance. Then, the normalized label is computed as

$$\tilde{\boldsymbol{\ell}} = \frac{1}{\sigma_{\boldsymbol{\ell}}}(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1})\,. \tag{98}$$

Hence, it is trivial to convert a normalized label into a non-normalized label and *vice versa*.

In order for the construction to work when samples have multiple labels, one must first weight decorrelate them. To decorrelate two labels $\boldsymbol{\ell}_{j'}$ and $\boldsymbol{\ell}_j$, with $j' > j$, one can project $\boldsymbol{\ell}_j$ out of $\boldsymbol{\ell}_{j'}$; $\ell^{\mathrm{dec}}_{j'}(n) = \ell_{j'}(n) - \frac{1}{Q}\big(\boldsymbol{\ell}^T_{j'}\mathrm{Diag}(\mathbf{v})\boldsymbol{\ell}_j\big)\ell_j(n)$, which is an invertible affine operation.

From now on, we assume that the labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_L$ have been decorrelated and normalized. The goal is then to compute edge weights $\gamma_{n,n'}$ such that the j-th optimal free response is equal to $\boldsymbol{\ell}_j$ (with arbitrary polarity).

Define

$$\mathbf{\Gamma}^{\mathrm{ELL}} \stackrel{\mathrm{def}}{=} \mathrm{Diag}(\mathbf{v}^{1/2})\,\mathbf{M}^{\mathrm{ELL}}\,\mathrm{Diag}(\mathbf{v}^{1/2})\,, \tag{99}$$

where

$$\mathbf{M}^{\mathrm{ELL}} \stackrel{\mathrm{def}}{=} \sum_{j=0}^{N-1} \lambda_j \mathbf{u}^{\mathrm{ELL}}_j (\mathbf{u}^{\mathrm{ELL}}_j)^T\,. \tag{100}$$

If $L < N-1$ one can set $\lambda_{j>L} = 0$. The matrix $\mathbf{\Gamma}^{\mathrm{ELL}}$ is symmetric by construction. The eigenvectors and eigenvalues of $\mathbf{M}^{\mathrm{ELL}}$, which are explicit in its eigenvector decomposition (100), directly define the matrix $\mathbf{\Gamma}^{\mathrm{ELL}}$ and determine the optimal free responses of the resulting graph. Concretely, for each $j \geq 1$ one sets $\mathbf{u}^{\mathrm{ELL}}_j$ according to the desired label $\boldsymbol{\ell}_j$ (ignore $\mathbf{u}^{\mathrm{ELL}}_0$ and λ_0 for the time

being).

$$\mathbf{u}_j^{\text{ELL}} = Q^{-1/2}\text{Diag}(\mathbf{v}^{1/2})\boldsymbol{\ell}_j\,,\ \text{for } j \geq 1 \tag{101}$$

Notice that the weighted decorrelation of the labels translates directly into the orthogonality of the corresponding eigenvectors, that is

$$\frac{1}{Q}(\boldsymbol{\ell}_j)^T\text{Diag}(\mathbf{v})\boldsymbol{\ell}_{j'} \overset{(80)}{=} 0 \quad \overset{(101)}{\Leftrightarrow} \quad (\mathbf{u}_j^{\text{ELL}})^T\mathbf{u}_{j'}^{\text{ELL}} = 0 \tag{102}$$

Once the eigenvectors are computed one must decide which eigenvalues they should have. Alternatively, one can decide which Δ values are given to the labels, because $\Delta_{\boldsymbol{\ell}_j}$ and λ_j are directly related: $\lambda_j \overset{(97)}{=} \frac{R}{2Q}(2 - \Delta_{\boldsymbol{\ell}_j})$.

Larger eigenvalues (equivalent to smaller Δ values) might result in higher accuracy for the corresponding label. In order to ease the choice of the eigenvalues, some hints and intuitions are provided. (a) In general, important labels should have larger eigenvalues than less important ones. (b) The global scale of the eigenvalues $\lambda_{j>0}$ is irrelevant, only their relative scales matter. For convenience one can scale them such that $\sum \lambda_{j>0} = 1$. (c) If two labels are similarly important, their eigenvalues should also be similar.

For example, if one only wants to learn a single label $\boldsymbol{\ell}_1$ with a delta value $\Delta_{\boldsymbol{\ell}_1} = 0$, one can set $\mathbf{u}_1^{\text{ELL}} = Q^{-1/2}\text{Diag}(\mathbf{v}^{1/2})\boldsymbol{\ell}_1$, $\lambda_1 = 1$, and the eigenvalues $\lambda_{j>1}$ to zero. These particular eigenvalues and $\Delta_{\boldsymbol{\ell}_1} = 0$ imply that $Q \overset{(97)}{=} R$, but the eigenvalues could be scaled by a positive factor, e.g., $\lambda_1 = 2, \lambda_{j>0} = 0$ implies $Q \overset{(97)}{=} R/2$. If $\boldsymbol{\ell}_1$ takes only two possible values (e.g., -1 and 1), the resulting graph will be disconnected and contain two clusters. Otherwise, the resulting graph will be connected, and the condition $\Delta_{\boldsymbol{\ell}_1} = 0$ necessarily implies that some of the resulting edge weights will be negative, a condition that is circumvented in Section 4.3.3.

The analysis of Section 4.3.1, which is used by the ELL method requires that the graph fulfills the consistency restriction (72). The remaining eigenvector is set as

$$\mathbf{u}_0^{\text{ELL}} = Q^{-1/2}\mathbf{v}^{1/2}\,, \tag{103}$$

with eigenvalue $\lambda_0 = R/Q$. This ensures that $(\mathbf{u}_0^{\text{ELL}})^T\mathbf{u}_0^{\text{ELL}} = 1$ and (72) is fulfilled, as follows.

$$\begin{aligned}
\boldsymbol{\Gamma}^{\text{ELL}}\mathbf{1} &\overset{(99,100)}{=} \text{Diag}(\mathbf{v}^{1/2})\big(\sum \lambda_j \mathbf{u}_j^{\text{ELL}}(\mathbf{u}_j^{\text{ELL}})^T\big)\text{Diag}(\mathbf{v}^{1/2})\mathbf{1} && (104)\\
&\overset{(103)}{=} \text{Diag}(\mathbf{v}^{1/2})\big(\sum \lambda_j \mathbf{u}_j^{\text{ELL}}(\mathbf{u}_j^{\text{ELL}})^T\big)\mathbf{u}_0^{\text{ELL}}Q^{1/2} && (105)\\
&\overset{(103)}{=} \text{Diag}(\mathbf{v}^{1/2})\lambda_0\mathbf{u}_0^{\text{ELL}}Q^{1/2} && (106)\\
&= (R/Q)\mathbf{v}\,. && (107)
\end{aligned}$$

The assignment of $\mathbf{u}_0^{\text{ELL}}$ and λ_0 above also ensures that $\mathbf{1}^T\mathbf{\Gamma}^{\text{ELL}}\mathbf{1} \overset{(81,107)}{=} R$. The free pseudo-response $\boldsymbol{\ell}_0 \overset{(101)}{=} \mathbf{1}$ corresponding to $\mathbf{u}_0^{\text{ELL}}$ fulfills equations (79) and (80) but not (78). Therefore, $\boldsymbol{\ell}_0$ is not a feasible solution, but it has similar properties to the optimal free responses. The introduction of $\mathbf{u}_0^{\text{ELL}}$ does not reduce the generality of the labels $\boldsymbol{\ell}_{j>0}$ that can be learned; orthogonality between $\mathbf{u}_0^{\text{ELL}}$ and $\mathbf{u}_{j>0}^{\text{ELL}}$ is equivalent to (78), i.e., the weighted zero mean of $\boldsymbol{\ell}_{j>0}$, a condition that is required anyway for any feasible solution: $(\mathbf{u}_0^{\text{ELL}})^T\mathbf{u}_{j>0}^{\text{ELL}} = 0 \overset{(102)}{\Leftrightarrow} (Q^{-1/2}\mathbf{v}^{1/2})^T Q^{-1/2}\text{Diag}(\mathbf{v}^{1/2})\boldsymbol{\ell}_{j>0} = Q^{-1}\mathbf{v}^T\boldsymbol{\ell}_{j>0} = 0$.

Although only L free responses are explicitly defined, $N - L - 1$ additional optimal free responses are defined implicitly with an eigenvalue of 0, corresponding to $\Delta = 2.0$. This Δ value has a particular meaning, because as proved in the next paragraph, it is the Δ value of unit-variance zero-mean i.i.d. noise for certain graphs.

Expected Weighted Δ Value of a Noise Feature

Let $\mathbf{y}$ be a noise feature randomly sampled from a zero-mean unit-variance distribution $\mathcal{D}$, i.e., $y(n) \leftarrow \mathcal{D}(0,1)$. On average, $\mathbf{y}$ fulfills the weighted normalization constraints (78) and (79), as can be seen as follows.

$$\text{(78):} \qquad \langle \mathbf{v}^T\mathbf{y}\rangle_{\mathcal{D}} = \mathbf{v}^T\langle\mathbf{y}\rangle_{\mathcal{D}} = 0\,, \tag{108}$$

$$\text{(79):} \qquad \langle \mathbf{y}^T\text{Diag}(\mathbf{v})\mathbf{y}\rangle_{\mathcal{D}} = \langle\sum_n v_n y(n)^2\rangle_{\mathcal{D}} = \sum_n v_n\langle y(n)^2\rangle_{\mathcal{D}} = Q\,, \tag{109}$$

where $\langle\cdot\rangle_{\mathcal{D}}$ denotes expected value when sampling over $\mathcal{D}$. The expected delta value can be computed as

$$\langle\Delta_{\mathbf{y}}\rangle_{\mathcal{D}} \overset{(6)}{=} \frac{1}{R}\sum_{n,n'}\gamma_{n,n'}\langle(y(n') - y(n))^2\rangle_{\mathcal{D}} \tag{110}$$

$$= \frac{1}{R}\Big(\sum_{\substack{n,n',\\ n\neq n'}}\gamma_{n,n'}\langle(y(n') - y(n))^2\rangle_{\mathcal{D}} + \sum_n \gamma_{n,n}\langle(y(n) - y(n))^2\rangle_{\mathcal{D}}\Big) \tag{111}$$

$$= \frac{1}{R}\Big(\sum_{\substack{n,n',\\ n\neq n'}}\gamma_{n,n'}\big(\langle y(n')^2\rangle_{\mathcal{D}} + \langle y(n)^2\rangle_{\mathcal{D}} - 2\langle y(n')\rangle_{\mathcal{D}}\langle y(n)\rangle_{\mathcal{D}}\big) + 0\Big) \tag{112}$$

$$= \frac{1}{R}\sum_{\substack{n,n',\\ n\neq n'}}\gamma_{n,n'}(1 + 1 - 0) \tag{113}$$

$$= \frac{2}{R}\big(\sum_{n,n'}\gamma_{n,n'} - \sum_n \gamma_{n,n}\big) \overset{(10)}{=} \frac{2(R - \sum_n \gamma_{n,n})}{R}\,. \tag{114}$$

Therefore, if the graph has no self-loops (i.e., $\forall n : \gamma_{n,n} = 0$), the expected Δ value $\langle\Delta_{\mathbf{y}}\rangle_{\mathcal{D}}$ of a noise feature $\mathbf{y}$ is 2.0. The self-loops of a graph (e.g., one

constructed using the ELL method) can be removed (i.e., their weight be set to zero). This does not change the free responses, only the scale of the Δ values is modified due to the change in R. The consistency restriction might be broken, though.

4.3.3 Elimination of Negative Edge Weights

From the objective function (6), it is obvious that a positive edge weight connecting two samples expresses that those samples should be mapped close to each other in feature space. In contrast, a negative edge weight expresses that two samples should be mapped as far apart as possible, thus encoding output dissimilarities. Nevertheless, the weighted unit variance constraint still applies, so the solutions are not unbounded.

If the edge weights are non-negative, the smallest possible Δ value is $\Delta = 0$. However, if negative edge weights are allowed, some feasible features might have $\Delta < 0$. A feature with $\Delta < 0$ would appear to be "slower" than the infeasible constant feature $\mathbf{y} = \mathbf{1}$ with $\Delta = 0$, contradicting the intuitive interpretation of slowness. Moreover, negative edge weights hinder the probabilistic interpretation of the graph (see Section 3.2.5), because some of the transition probabilities $\gamma_{n,n'}/R$ of the resulting Markov chain would be negative.

Training graphs constructed using the ELL method might include negative edge weights, which would result in the disadvantages described above. Therefore, in this section, an additional step is added to the ELL method to ensure that the training graph has non-negative edge weights. More concretely, it is shown how to transform a training graph with strictly positive vertex weights v_n and arbitrary edge weights $\mathbf{\Gamma}$ (positive and negative) into a graph with the same vertex weights and only non-negative edge weights $\mathbf{\Gamma}'$. The optimization problem defined by $\mathbf{\Gamma}'$ is equivalent to the original optimization problem in terms of its solutions and their order. Only the value of the objective function is linearly changed (or, more precisely, changed by an affine function).

Assume that $\forall n : v_n > 0$, and that there is at least one element $\gamma_{n,n'} < 0$. Let

$$c \stackrel{\text{def}}{=} \max_{n,n'} \frac{-\gamma_{n,n'}}{v_n v_{n'}} \,. \tag{115}$$

The new edge weights $\mathbf{\Gamma}'$ are defined as

$$\mathbf{\Gamma}' \stackrel{\text{def}}{=} \frac{1}{1 + cQ^2/R} (\mathbf{\Gamma} + c\mathbf{v}\mathbf{v}^T). \tag{116}$$

The properties of $\mathbf{\Gamma}'$ and their relation to those of $\mathbf{\Gamma}$ are as follows:

1. All elements of $\mathbf{\Gamma}'$ are greater or equal to zero, as desired. (Follows from (115), which implies $\gamma_{n,n'} + c v_n v_{n'} \geq 0$.)
2. Symmetry is preserved by (116). Clearly $\mathbf{\Gamma}'$ is symmetric if and only if $\mathbf{\Gamma}$ is symmetric.

3. The sum of edge-weights is preserved:

$$R' \overset{(82)}{=} \mathbf{1}^T \mathbf{\Gamma}' \mathbf{1} \overset{(116)}{=} \frac{R + cQ^2}{1 + cQ^2/R} = R\,. \tag{117}$$

4. Fulfillment of the graph consistency restriction (72) is preserved:

$$\begin{aligned}
\mathbf{1}^T \mathbf{\Gamma} \overset{(72)}{=} R/Q\mathbf{v}^T \quad \Rightarrow \quad \mathbf{1}^T \mathbf{\Gamma}' &\overset{(116)}{=} \frac{1}{1 + cQ^2/R}(\mathbf{1}^T \mathbf{\Gamma} + c\mathbf{1}^T \mathbf{v}\mathbf{v}^T) && (118)\\
&\overset{(72)}{=} \frac{1}{1 + cQ^2/R}(R/Q\mathbf{v}^T + cQ\mathbf{v}^T) && (119)\\
&= \frac{R/Q}{1 + cQ^2/R}(1 + cQ^2/R)\mathbf{v}^T && (120)\\
&= R/Q\mathbf{v}^T\,. && (121)
\end{aligned}$$

5. $\mathbf{\Gamma}$ and $\mathbf{\Gamma}'$ define equivalent optimization problems. Let $\mathbf{y}$ be a feasible solution. The constraints of the optimization problem are independent of $\mathbf{\Gamma}'$, and only the objective function is modified as follows:

$$\begin{aligned}
\Delta'_{\mathbf{y}} &\overset{(76)}{=} 2 - \frac{2}{R'}\mathbf{y}^T \mathbf{\Gamma}' \mathbf{y} && (122)\\
&\overset{(116,117)}{=} 2 - \frac{2}{R(1 + cQ^2/R)}\left(\mathbf{y}^T \mathbf{\Gamma} \mathbf{y} + c\mathbf{y}^T \mathbf{v}\mathbf{v}^T \mathbf{y}\right) && (123)\\
&\overset{(78)}{=} 2 - \frac{2}{R(1 + cQ^2/R)}\mathbf{y}^T \mathbf{\Gamma} \mathbf{y} && (124)\\
&\overset{(76)}{=} 2 - \frac{2}{R(1 + cQ^2/R)}\frac{R}{2}(2 - \Delta_{\mathbf{y}}) && (125)\\
&= \frac{1}{(1 + cQ^2/R)}\left(\Delta_{\mathbf{y}} + \frac{2cQ^2}{R}\right)\,. && (126)
\end{aligned}$$

   Therefore, the objective function is only modified by a positive scaling factor and a constant positive offset, proving that the optimal free solutions to the training graph remain stable, as well as their order.

6. In particular, a feature $\mathbf{y}$ with $\Delta_{\mathbf{y}} = 2$ preserves its delta value, i.e. $\Delta_{\mathbf{y}} = 2 \overset{(125)}{\Leftrightarrow} \Delta'_{\mathbf{y}} = 2$.

4.3.4 Auxiliary Labels for Boosting Estimation Accuracy

It is possible to provide additional *auxiliary* labels derived from the original one $\boldsymbol{\ell}_1$ as a means to improve the estimation accuracy when GSFA is applied repeatedly (e.g., cascaded or in a convergent hierarchical GSFA network).

Consider two GSFA nodes, one stacked on top of the other. If the first GSFA node is not able to extract $\boldsymbol{\ell}_1$ accurately, it might still be capable of approximating labels $\boldsymbol{\ell}_k = f_k(\boldsymbol{\ell}_1)$, for $2 \leq k \leq K$, where the functions $f_k(\cdot)$ are nonlinear. Since these features are derived from the original label $\boldsymbol{\ell}_1$, they contain a certain

amount of information about it. When multiple labels are learned, the output features are likely to contain (or more precisely, approximate) linear combinations of labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_k$ providing a redundant encoding of $\boldsymbol{\ell}_1$. These features are likely to be easier to disentangle by the second node to approximate the original label $\boldsymbol{\ell}_1$ more accurately.

The functions f_k can be defined arbitrarily, one simple choice is to use

$$\ell_k(n) = \cos\left(\frac{\ell_1(n) - \min(\boldsymbol{\ell}_1)}{\max(\boldsymbol{\ell}_1) - \min(\boldsymbol{\ell}_1)}\pi k\right), \text{ for } 2 \le k \le K\,, \tag{127}$$

where $\max(\boldsymbol{\ell}_1)$ is the largest label value, and $\min(\boldsymbol{\ell}_1)$ is the smallest one. As usual, it is assumed that labels $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_K$ are weight decorrelated and normalized before the ELL method is applied.

The eigenvalues corresponding to the auxiliary labels must be set smaller than those of the original label. Otherwise, the slowest features might be more similar to the auxiliary labels than to the original one. From now on, the term *target* labels will be used to refer to the original and auxiliary labels, if present.

The use of auxiliary labels can be justified from information theory. Assume that the samples have been ordered by increasing label $\boldsymbol{\ell}_1$. This implies that for $\boldsymbol{\ell}_k$ the argument of the cosine function ranges from 0 to $k\pi$. Thus $\boldsymbol{\ell}_2$ describes 1 oscillation, $\boldsymbol{\ell}_3$ describes 1.5 oscillations, etc. In this sense, the auxiliary labels are "higher-frequency" versions of $\boldsymbol{\ell}_1$. Notice that $\boldsymbol{\ell}_2$ contains almost all the information about $\boldsymbol{\ell}_1$ except for 1 bit. That is, $I(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2) = H(\boldsymbol{\ell}_1) - 1$, where I is mutual information and H is entropy. Similarly, $\boldsymbol{\ell}_4$ loses 2 bits of information about $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_8$ loses 3 bits, and so on. Thus, auxiliary labels contain a large amount of information about $\boldsymbol{\ell}_1$.

Moreover, the use of auxiliary labels supports the goal that samples $\mathbf{x}(n)$ and $\mathbf{x}(n')$ with similar labels $\ell_1(n)$ and $\ell_1(n')$ should have similar output features $y_j(n)$ and $y_j(n')$ on average, for $1 \le j \le J$. This property is desirable not only for the slowest feature $\mathbf{y}_1$. The reason auxiliary labels favor the appearance of this property among other features is the "smoothness" of the auxiliary labels in terms of $\boldsymbol{\ell}_1$ (i.e., how fast they change w.r.t. $\boldsymbol{\ell}_1$). Notice that $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_J$ are ordered by decreasing smoothness.

Interestingly, in regular SFA (or GSFA trained with the reordering graph) particular auxiliary labels are included automatically (however, they are not identical to those defined in (127)). The slowest free response is a half period of a cosine function—approximately $\sqrt{2}\cos(\pi n/(N-1))$—, and the subsequent free responses are the higher-frequency harmonics of the first one (see Section 4.4.2, particularly Figure 4.4).

4.3.5 Computational Complexity of the ELL Method

The main drawback of ELL is its computational cost compared to efficient predefined training graphs, a disadvantage that is more marked for large N. One can analyze the efficiency of explicit label learning by considering its two main parts: The construction of the training graph and training GSFA with it.

The graph construction requires $\mathcal{O}(L^2N + LN^2)$ operations. The term L^2N is due to the transformation of L target labels into eigenvectors, which might require a decorrelation step on L N-dimensional vectors. The term LN^2 is due to the computation of $\mathbf{M}$, which involves L outer vector products $\mathbf{u}_j\mathbf{u}_j^T$.

When GSFA is trained, three computations are particularly expensive. (1) The computation of $\mathbf{C_G}$, which takes $\mathcal{O}(NI^2)$ operations. (2) The computation of $\dot{\mathbf{C}}_\mathbf{G}$, which can be expressed as $\dot{\mathbf{C}}_\mathbf{G} = \frac{2}{Q}\mathbf{X}\mathrm{Diag}(\mathbf{v})\mathbf{X}^T - \frac{2}{R}\mathbf{X}\mathbf{\Gamma}\mathbf{X}^T$, where $\mathbf{X} = \big(\mathbf{x}_1, \ldots, \mathbf{x}_N\big)$, taking $\mathcal{O}(N^2I + NI^2)$ operations. (3) The solution to the generalized eigenvalue problem, which requires $\mathcal{O}(I^3)$ operations. Therefore, in general, training GSFA requires $\mathcal{O}(NI^2 + N^2I + I^3)$ operations. Typically $N > I$ to avoid overfitting, so the computation of $\dot{\mathbf{C}}_\mathbf{G}$ is the most expensive part.

However, when an efficient pre-defined graph (e.g., the serial graph) is used instead of an ELL graph, it is possible to avoid the explicit graph construction and compute $\dot{\mathbf{C}}_\mathbf{G}$ with optimized algorithms that take into account the regular structure of the graph. In this way, efficient pre-defined graphs allow the computation of $\dot{\mathbf{C}}_\mathbf{G}$ in $\mathcal{O}(NI^2)$ operations, which is equivalent to the complexity of standard SFA on N I-dimensional samples. Moreover, if the number of edges $N_e \leq N(N+1)/2$ is small, one can use (14) to compute $\dot{\mathbf{C}}_\mathbf{G}$ in $\mathcal{O}(N_eI^2)$ operations. Therefore, for these two special cases, training GSFA takes $\mathcal{O}(NI^2 + I^3)$ and $\mathcal{O}((N_e+N)I^2+I^3)$ operations, respectively. In Section 4.5.4 the complexity of the ELL method is further discussed and in Section 4.5.5 a few approaches to improve it are proposed.

4.4 Applications of Explicit Label Learning

This section presents three applications of the proposed method. The first one illustrates how to solve a regression problem with GSFA explicitly, learning a direct mapping from images to labels (see Figure 4.1.c). The second application shows the analysis of two pre-defined graphs by computing their optimal free responses. In the third application, the ELL method is used in a new way to learn compact discriminative labels for classification.

4.4.1 Explicit Estimation of Gender with GSFA

This section addresses the problem of gender estimation from artificial face images, which is treated here as a regression problem, because the gender parameter is defined as a real value by the face modeling software (FaceGen SDK, Singular Inversions Inc., 2008). This application has been described in Section 2.6.4 but is now addressed in more detail and with a newer method.

Input data. The input data consists of 12,000 64×64 grayscale images. Each image is generated using a new subject identity, where the gender is explicitly specified, and the rest of the parameters of the faces (e.g., age, racial composi-

tion) are random. The average pixel intensity of each image is normalized by multiplying the pixel values by an appropriate factor to eliminate skin color as a cue for gender estimation. The resulting images show subjects with a fixed pose, no hair or accessories, and the illumination is fixed, as well as the average pixel intensity and the background color (black). See Figure 4.2 for some sample images. To specify the gender parameter, 60 different values are used $(-3, -2.9, \ldots, 2.9)$.

-3 very masculine | -1.5 masculine | 0 neutral | 1.5 feminine | 2.9 very feminine

Figure 4.2: Examples of normalized images, showing different values of the gender parameter.
[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

The images are randomly split into a training and a test set. The training set consists of 10,800 images, 180 images for each gender value, whereas the test set consists of 1,200 images, 20 images for each gender value.

Besides the gender label, also a second "color" label is considered, which is the average pixel intensity of the image *before* normalization. Due to normalization, this label cannot be computed directly, but it can be estimated from other cues, such as the subject's apparent race and face size. In the following experiment only the gender label is considered, but afterwards both labels (gender and color) are used simultaneously.

Network used. For efficiency reasons, hierarchical GSFA (HGSFA) is used, which is implemented by an 8-layer HGSFA network with the structure described in Table 4.1. The nodes of the network have non-overlapping receptive fields and use the 0.8*Expo* expansion function defined in (68), which can be stated as: $(x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n, |x_1|^{0.8}, \ldots, |x_n|^{0.8})$, followed by linear GSFA. The nodes of the first layer include a PCA pre-processing step that preserves 50 out of 64 components.

Training graphs for gender estimation. Several training graphs are constructed using the ELL method described in Sections 4.3.2–4.3.4. These graphs are denoted ELL^g-L, where L is the total number of target labels considered, with $L \in \{1, 10, 20, 30, 40, 50\}$, and the superscript g stands for gender (later c will be used for color and g,c for gender and color). The first target label $\ell_1(n)$ is the gender parameter, where $1 \le n \le 10{,}800$. The remaining $L-1$ labels are auxiliary and computed using (127). For comparison purposes, the serial and reordering training graphs are also evaluated.

layer	number of nodes	node's receptive field (pixels)	input dim per node	expanded dim per node	output dim per node
1	8×8	8×8	64	100	40
2	4×8	16×8	80	160	40
3	4×4	16×16	80	160	40
4	2×4	32×16	80	160	40
5	2×2	32×32	80	160	40
6	1×2	64×32	80	160	40
7	1×1	64×64	80	160	40
8	1×1	64×64	40	80	6

Table 4.1: Structure of the GSFA hierarchical network. The inputs to the nodes in the first layer are 8×8-pixel patches. The input to the node in layer 8 is the output of the node in layer 7. The inputs to all other nodes come from two nodes in the previous layer that are contiguous either vertically or horizontally.

Label estimations. Three mappings from the slowest features to the label estimation $\hat{\ell}$ are tested. The first mapping (only available for the ELL graphs) is an affine transformation $\hat{\ell} = \pm y_1 \sigma_{\boldsymbol{\ell}} + \mu_{\boldsymbol{\ell}}$, where $\mu_{\boldsymbol{\ell}}$ and $\sigma_{\boldsymbol{\ell}}$ have been computed previously for label normalization (98). Since the sign of y_1 is arbitrary, it is globally adjusted to fit the labels best. The second method is linear regression (LR, ordinary least squares). For these two methods, final label estimation $\hat{\ell}$ is clipped to the valid label range $[-3, 2.9]$. The third mapping is the soft GC method (see Section 3.4.5), which provides a soft estimation based on the class probabilities estimated by a Gaussian classifier, in this case trained using 60 classes.

Results. Table 4.2 (left) shows the label estimation errors (RMSE) when gender is estimated. Unless otherwise stated, all results have been averaged over 10 runs. Depending on the mapping, the ELL^g-10 and ELL^g-40 graphs outperform the rest. This supports the intuition that auxiliary labels are useful. 50 target labels perform worse than 40, probably in part because the output dimensionality of the intermediate nodes in the network is 40. Without the final clipping step LR is clearly more accurate than the affine mapping (experiment not shown), but both methods have similar accuracy if clipping is enabled. For all graphs, the explicitly supervised soft GC method provided better accuracy than the affine mapping, although the difference is smaller than one might have expected.

For comparison, the serial graph results in RMSEs of 0.351 (soft GC, 1F) and 0.349 (soft GC, 3F), whereas the reordering graph results in RMSEs of 0.353 (soft GC, 1F) and 0.347 (soft GC, 3F). The accuracy of these two graphs appears to be similar; however, in more complex experiments the serial graph has typically been more accurate (e.g. see Section 3.5.2). The ELL^g-40 graph is, therefore, slightly more accurate than the serial and reordering graphs but 25

Graph ELL^g-L				
L	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)
1	0.376	0.380	0.364	0.365
10	**0.364**	**0.365**	0.353	0.356
20	0.372	0.374	0.356	0.357
30	0.367	0.368	0.350	0.349
40	0.368	0.367	**0.346**	**0.345**
50	0.376	0.375	0.351	0.350

Graph ELL^{g,c}-L				
L	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)
2×1	**0.298**	**0.299**	**0.289**	0.284
2×5	0.349	0.350	0.343	**0.277**
2×10	0.423	0.426	0.410	0.288
2×15	0.473	0.478	0.453	0.291
2×20	0.508	0.514	0.479	0.292
2×25	0.535	0.543	0.499	0.294

Table 4.2: *Gender* estimation errors (RMSE) using various graphs and either one (1F) or three (3F) features. For the linear regression (LR) mapping, the label is estimated as $\hat{\ell}_1 = ay_1 + b$, with a and b fitted to the training data. Chance level (RMSE) is 1.731 if one uses the constant estimation $\hat{\ell}_1 = -0.05$. All errors have been computed on test data and averaged over 10 runs. Estimation errors using training graphs for gender estimation only (left) and using training graphs for simultaneous estimation of gender and color (right).

times slower, taking about 250 min for training instead of about 10 min (single thread).

Simultaneous learning of gender and color. A graph that encodes gender and color simultaneously is constructed to learn labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_L$, where $\boldsymbol{\ell}_1$ is the gender label, $\boldsymbol{\ell}_2$ is the color label, $\boldsymbol{\ell}_3, \boldsymbol{\ell}_5, \ldots, \boldsymbol{\ell}_{L-1}$ are derived from $\boldsymbol{\ell}_1$, and $\boldsymbol{\ell}_4, \boldsymbol{\ell}_6, \ldots, \boldsymbol{\ell}_L$ are derived from $\boldsymbol{\ell}_2$. Each set of labels is computed using (127) similarly to the auxiliary labels for gender only but starting from either the original gender or color labels. The chosen eigenvalues decrease linearly and add to one. The resulting graphs are denoted ELL^{g,c}-L, where L is the total number of target labels, with $L = 2 \times d$, for $d \in \{1, 5, 10, 15, 20, 25\}$, and $2(d-1)$ is the number of auxiliary labels used for gender and color.

The effect of encoding gender and color simultaneously on gender estimation is shown in Table 4.2, right (compare to Table 4.2, left).

The ELL^{g,c}-L graphs yield significantly higher accuracy than the ELL^g-L graphs (an MAE as small as 0.277 vs. 0.345). The results on color estimation using the ELL^{g,c}-L graphs are shown in Table 4.3, right (compare to Table 4.3, left). The slowest extracted feature represents mostly gender. However, it must also contain color information since it allows color estimation better than chance level. When 3 features are preserved, the ELL^{g,c}-L graphs yield higher accuracy than the ELL^c-L graphs. Similar experimental results have been reported, e.g. by Guo and Mu (2014), who have shown that age estimation improves when gender and race labels are also considered.

Learning label transformations. An additional experiment is performed to verify that the method can learn other labels implicitly described by the

Graph $\mathbf{ELL}^{c}$-L				
L	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)
1	2.000	1.987	1.971	1.979
10	**1.969**	**1.958**	1.905	1.922
20	2.006	1.999	1.914	1.922
30	1.991	1.989	1.877	1.889
40	1.990	1.990	**1.864**	**1.867**
50	1.997	1.997	1.865	1.871

Graph $\mathbf{ELL}^{g,c}$-L				
L	LR (1F)	soft GC (1F)	LR (3F)	soft GC (3F)
2×1	4.247	4.291	1.393	1.221
2×5	3.606	3.614	**1.239**	1.210
2×10	3.214	3.185	1.337	1.180
2×15	2.978	2.945	1.429	1.158
2×20	2.828	2.802	1.501	1.141
2×25	**2.718**	**2.700**	1.582	**1.140**

Table 4.3: *Color* estimation errors (RMSE) using various graphs and either one (1F) or three (3F) features. Chance level is 7.447. All results have been computed on test data and averaged over 10 runs. (Left) Error using training graphs that encode only color. (Right) Error using training graphs that simultaneously encode gender and color.

data. This is mostly useful to test the integrity of the method. More precisely, GSFA is used to learn labels $(\boldsymbol{\ell}_1)^2$ and $(\boldsymbol{\ell}_1)^3$, which are distorted versions of the original gender label $\boldsymbol{\ell}_1$. The graphs constructed for this purpose are denoted $\text{ELL}^{g}\text{-}40^{(\boldsymbol{\ell}_1)^2}$ and $\text{ELL}^{g}\text{-}40^{(\boldsymbol{\ell}_1)^3}$, respectively. Both of them include 39 auxiliary labels besides the main distorted label. To better approximate the target labels, more complex nonlinearities are used in some of the nodes of the hierarchical networks. The $(\boldsymbol{\ell}_1)^2$ network is identical to the $\boldsymbol{\ell}_1$ network, except that in the top node the quadratic expansion is used instead of the 0.8Expo expansion. Similarly, the $(\boldsymbol{\ell}_1)^3$ network uses the quadratic expansion in the 7th layer, and the 6th-degree polynomial expansion in the top node. In both networks, the output dimension of the node in the 7th layer is set to 3 to avoid overfitting due to the expansion in the 8th layer.

The corresponding label estimations are shown in Figure 4.3. For comparison, also the $\text{ELL}^{g}\text{-}40$ graph is included. The results prove that the ELL method can also be used to learn distortions of the main label. Admittedly, the accuracy of the estimations (expressed as a fraction of the respective chance levels) decreases even though the complexity of the feature space has been increased.

4.4.2 Analysis of Pre-Defined Training Graphs

In this section, the method of Section 4.3.1 is used to extract the optimal free responses of three graphs (reordering, serial and ELL-4). The optimal free responses and their Δ values (alternatively, the eigenvectors $\mathbf{u}_j$ and eigenvalues λ_j) fully characterize the properties of a training graph, and provide another representation of it that might be more useful in some contexts.

Equations (91)–(93) are used to compute optimal free responses, and (97) to compute their delta values. Therefore, these results have been obtained by solving these analytical equations numerically and are not constrained by the

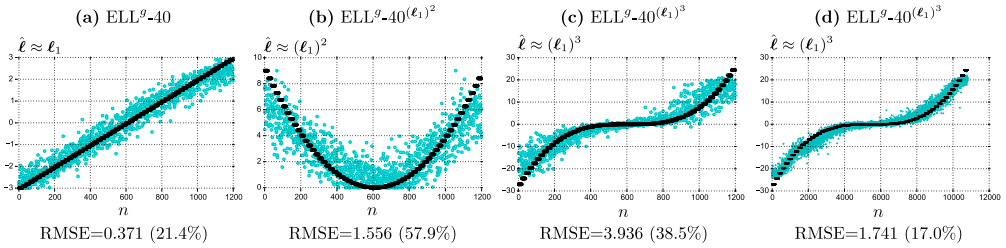


Figure 4.3: Plots (a) to (c) show the label estimations on test data (a single run) when different distorted versions of $\boldsymbol{\ell}_1$ are learned. The affine mapping is used. Therefore, the estimations are only generated from the slowest feature. Ground-truth values are shown in thicker black. The RMSE is expressed in parenthesis as a percentage of the chance level. Plot (d) is analogous to (c) but shows *training* data.

[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

input samples[3]. They are visualized in Figure 4.4, which shows an arbitrary label to be learned (top), and three different graphs that can be used for this purpose. Only $N = 30$ samples (ordered by increasing label) are used to ease visualization, but the plots behave similarly for larger N. The following three graphs are employed. 1) A reordering graph (Figure 3.3.b) that has been extended with two edge weights $\gamma_{0,0} = 1$ and $\gamma_{N-1,N-1} = 1$ to fulfill the consistency restriction (32), which is required by the method. These weights introduce a constant scaling $N/(N + 2)$ of the delta values, without any further consequence. 2) A serial graph (Section 3.4.3) with $K = 15$ groups of 2 samples each. 3) An ELL-4 graph (Sections 4.3.2–4.3.4) that is constructed with the original labels $\ell_1(n) = \ell(n)$, and 3 auxiliary labels computed using (127).

Figure 4.4 shows also that the most remarkable difference between these graphs is the number of optimal free responses with $\Delta < 2.0$, which is 14 for the reordering graph, 6 for the serial graph, and 4 for the ELL-4 graph, for the parameters above. For arbitrary parameters, the reordering, serial and ELL-L graphs have $\lfloor (N-1)/2 \rfloor$, $\lfloor (K-1)/2 \rfloor$, and, depending on the eigenvalues, up to $L \leq N - 1$ optimal free responses with $\Delta < 2.0$, respectively.

Although the graphs differ considerably in their connectivity, their first four to five optimal free responses have a somewhat similar shape. Since in all graphs the slowest free response $\mathbf{y_1}$ is increasing, a monotonic mapping would be enough to approximate the label for any of these graphs. However, the slowest response of the serial graph is constant within each group, which might lower accuracy due to a discretization error. In contrast, the ELL-4 graph has been tailored to learn a particular label, and therefore $\mathbf{y_1}$ is exactly $\boldsymbol{\ell}_1$ (the original label) except for an offset and scaling.

[3] A less elegant method is to apply GSFA to a graph consisting of random samples and the given edges. If the sample dimensionality is large enough, the extracted features will approximate the optimal free responses due to extreme overfitting.

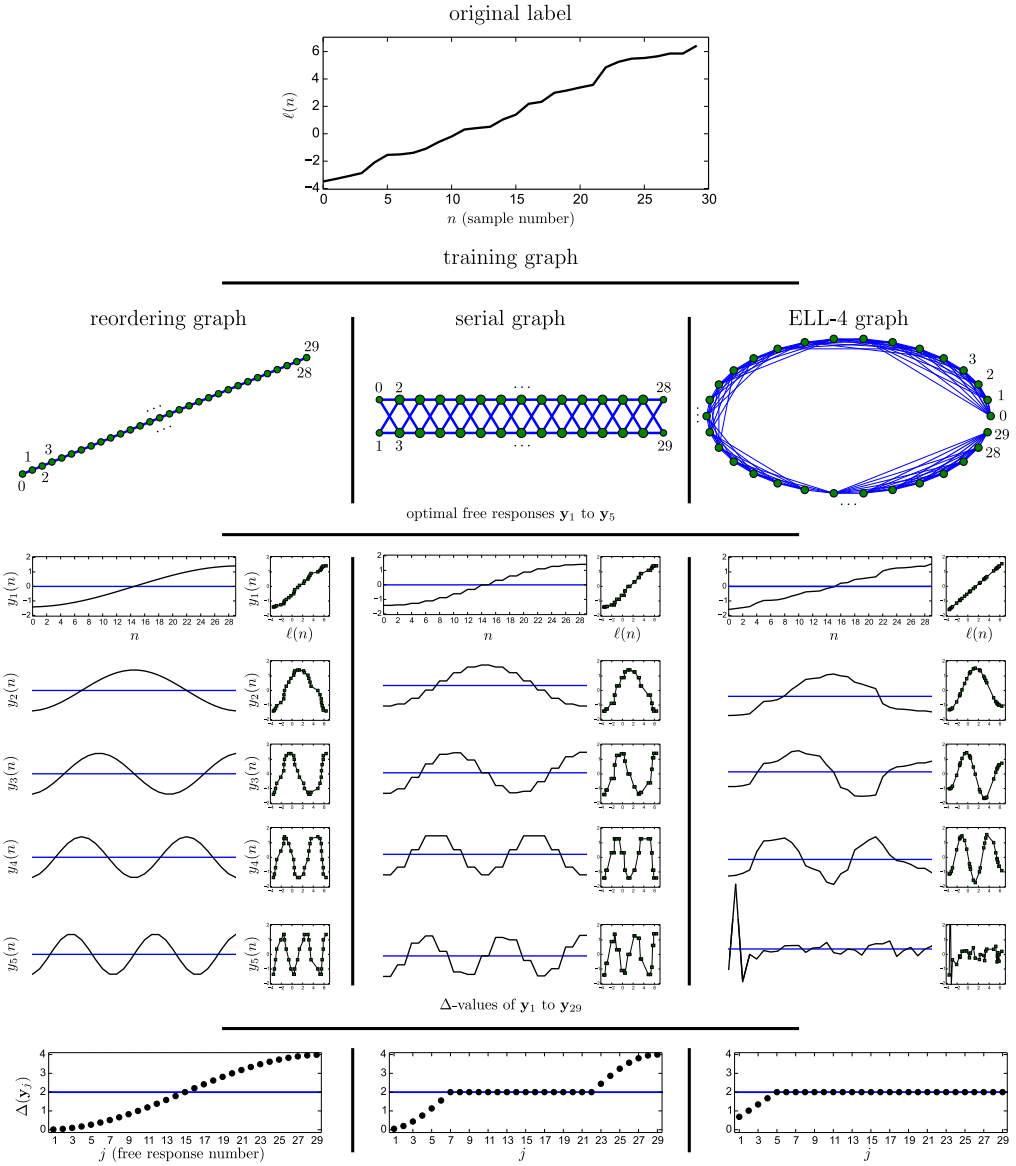


Figure 4.4: An arbitrary label $\ell(n)$ (top) and three graphs that can be used to learn it. The five slowest optimal free responses $\mathbf{y}_1$ to $\mathbf{y}_5$ of each graph are plotted, as well as the delta values of all optimal free responses. The ELL-4 graph is almost fully connected, but here only the strongest 30% of the connections are displayed. Samples have an index n from 0 to 29, and free responses have an index j from 1 to 29. The free responses are also plotted against the original label (smaller square plots). The polarity of the free responses was adjusted once to make them negative for the first sample.

[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

The analysis makes clear that the serial and ELL-4 graphs are *more selective* than the reordering graph regarding the features that they consider slow. To illustrate why this might be an advantage, consider a scaled and noisy version $\hat{\mathbf{y}}_1$ of $\boldsymbol{\ell}_1$. More concretely, $\hat{y}_1(n) = \frac{\sqrt{2}}{2}\ell_1(n) + \frac{\sqrt{2}}{2}e(n)$, where $e(n)$ is an i.i.d. zero-mean unit-variance noise signal. When the reordering graph is used, the feature $\hat{\mathbf{y}}_1$ has an average Δ-value of about 1 (i.e. $\langle\Delta_{\hat{\mathbf{y}}_1}\rangle \approx 1$), and therefore such a feature would appear to be faster than an auxiliary (127) feature $\mathbf{y}_6 = \boldsymbol{\ell}_6$, because $\Delta_{\boldsymbol{\ell}_6} \approx 0.38$. Hence, a GSFA node trained with the reordering graph would favor the extraction of $\mathbf{y}_6$ over $\hat{\mathbf{y}}_1$, even though $\hat{\mathbf{y}}_1$ is more similar to the label. In contrast, the serial and ELL-4 graphs might favor the extraction of $\hat{\mathbf{y}}_1$, because for these graphs $\Delta_{\boldsymbol{\ell}_6}$ is larger and close to 2.0.

4.4.3 Compact Discriminative Features for Classification

A well-known algorithm for supervised dimensionality reduction for classification is Fisher discriminant analysis (FDA). According to the theory of FDA, if there are C classes, $C-1$ features define a $C-1$ dimensional subspace that best separates the classes. In practice, one typically uses all these $C-1$ features, because all of them contain discriminative information and contribute to classification accuracy. The same holds for GSFA if the clustered training graph is used (GSFA+clustered), because in this case the features learned are equivalent to those of FDA (see Section 3.6.3, and Klampfl and Maass, 2010).

One can take advantage of hierarchical processing to do classification using the clustered graph (HGSFA+clustered). However, when the number of classes C is large (e.g., $C \geq 100$) it might become expensive to preserve $C-1$ features in each node, because the size of the input to subsequent nodes would be a multiple of $C-1$. Such a large dimensionality would be further increased by the expansion function, resulting in a large training complexity. For instance, consider a 2-layer nonlinear network for classification with two GSFA nodes in the first layer and one in the top layer. Suppose the first two nodes have output dimensionality $C-1 = 99$, making the input of the top node 198-dimensional, and suppose that the top node applies a quadratic expansion to its input data before linear GSFA. The expanded data would have dimensionality I' =19,701. The combination of a large sample dimensionality I' and a large number of samples N (with $N \gg I'$ to avoid overfitting) would result in considerable computational and memory costs. Therefore, if it were possible to encode the class information in the first layer more compactly, one could reduce the output dimensionality of the first-layer nodes and reduce overfitting, aiming at increasing classification accuracy.

In this section, the theory of explicit learning with multiple labels is used to compute compact features for classification using GSFA. These features are used to classify images of $C = 32$ traffic signs from the German traffic sign recognition benchmark database (Houben et al., 2013).

The images are represented as 48×48-pixel color (RGB) images (see Figure 4.5). From 43 different traffic signs only 32 of them with the most samples are used, so that the number of classes is a power of 2 and the number of sam-

ples is maximized. For the training data, the same number of samples is used for each class (traffic sign), namely 2,160 of them, making a total of 69,120 images. To reach 2,160 samples per class, images of some classes are used up to 6 times (since the database is unbalanced). The images used for training are distorted by a random rotation r of $-3.15 \leq r \leq 3.15$ degrees, horizontal and vertical translations Δ_x, Δ_y with $-1.73 \leq \Delta_x, \Delta_y \leq 1.73$ pixels, and a scaling factor s with $0.91 \leq s \leq 1.09$. The purpose of these distortions is to improve generalization and provide invariances to small misalignments. For testing, the official test data is used, which ensures that the test images originate from signs physically different from the ones used for training. The test data consists of 9,030 undistorted images.

Figure 4.5: The 32 traffic signs learned, one image per class.
[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

The employed GSFA architecture is simple (non-hierarchical): PCA is applied first to reduce the dimensionality to 120 principal components. Afterwards, quadratic GSFA is applied using different training graphs, described below. Finally, since this is a classification problem, a nearest centroid classifier is used instead of the affine mapping.

The ELL method is used to construct two training graphs with binary target labels (i.e., label values are either 1 or -1). The first one has 5 labels (compact+5) and the second one has 31 (compact+31). The target labels are defined in Table 4.4. Notice that the first 5 labels (for both graphs) suffice, in principle, to fully encode the class information, because they can be viewed as a binary representation of the class number.

For the compact+5 graph, identical eigenvalues ($\lambda_1^1 = \lambda_2^1 = \lambda_3^1 = \lambda_4^1 = \lambda_5^1 = 0.2$) are used to express equal importance of the target labels. The compact+31 graph has been included to show the effect of auxiliary labels $\boldsymbol{\ell}_6, \boldsymbol{\ell}_7, \ldots, \boldsymbol{\ell}_{31}$. For this graph, the first five eigenvalues $(\lambda_1^2, \lambda_2^2, \ldots, \lambda_5^2) = (0.056, 0.056, \ldots, 0.056)$ are identical, but the rest decrease linearly: $(\lambda_6^2, \lambda_7^2, \ldots, \lambda_{31}^2) = (0.053, 0.051, \ldots, 0.004, 0.002)$, where only three decimal places are shown. Thus, the importance

$c \to$	1	2	3	4	5	6	7	8	9	...	16	17	...	30	31	32
$\boldsymbol{\ell}_1(c)$	-1	-1	-1	-1	-1	-1	-1	-1	-1	...	-1	$+1$	...	$+1$	$+1$	$+1$
$\boldsymbol{\ell}_2(c)$	-1	-1	-1	-1	-1	-1	-1	-1	$+1$	...	$+1$	-1	...	$+1$	$+1$	$+1$
$\boldsymbol{\ell}_3(c)$	-1	-1	-1	-1	$+1$	$+1$	$+1$	$+1$	-1	...	$+1$	-1	...	$+1$	$+1$	$+1$
$\boldsymbol{\ell}_4(c)$	-1	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	...	$+1$	-1	...	-1	$+1$	$+1$
$\boldsymbol{\ell}_5(c)$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1	...	$+1$	-1	...	$+1$	-1	$+1$
$\boldsymbol{\ell}_6(c)$	-1	$+1$	$+1$	-1	$+1$	-1	-1	$+1$	$+1$	...	-1	$+1$	...	-1	-1	$+1$
$\boldsymbol{\ell}_7(c)$	-1	-1	$+1$	$+1$	$+1$	$+1$	-1	-1	$+1$	...	$+1$	$+1$	...	$+1$	-1	-1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{\ell}_{30}(c)$	-1	$+1$	-1	$+1$	$+1$	-1	$+1$	-1	-1	...	-1	-1	...	-1	$+1$	-1
$\boldsymbol{\ell}_{31}(c)$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	...	-1	-1	...	$+1$	$+1$	-1

Table 4.4: Target labels used to encode the class information, compactly expressed as a function of the class number c. The compact+5 graph is constructed with labels $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_5$, whereas the compact+31 graph with $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_{31}$. The first five labels can be seen as the original ones and the rest as auxiliary.

given to the auxiliary labels decreases from $\boldsymbol{\ell}_6$ to $\boldsymbol{\ell}_{31}$. In both graphs, the eigenvalues have been scaled to make their sum equal to 1.

The number of classes, $C = 2^5$, has been specifically chosen, because powers of two make it simple to obtain binary labels with a weighted zero mean, weighted unit variance, and weighted decorrelation, as follows. The first five original labels can be computed as $\boldsymbol{\ell}_j(c) = 2(\frac{c-1}{2^{5-j}} \mod 2) - 1$, where $1 \le c \le C$ is the class number, the division is integer division and "mod" is the modulo operation (i.e., an image n of class c is assigned a label $\boldsymbol{\ell}_j(c)$). The auxiliary labels are computed as the product of two or more labels $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_5$, possibly multiplied by a factor -1 to make the label assigned to the first class negative. More concretely, $\boldsymbol{\ell}_6$ is the product of all original labels, $\boldsymbol{\ell}_7$ to $\boldsymbol{\ell}_{11}$ are all products of four of them, $\boldsymbol{\ell}_{12}$ to $\boldsymbol{\ell}_{21}$ are all products of three, and $\boldsymbol{\ell}_{22}$ to $\boldsymbol{\ell}_{31}$ are all products of two (e.g., $\boldsymbol{\ell}_6 = \boldsymbol{\ell}_1\boldsymbol{\ell}_2\boldsymbol{\ell}_3\boldsymbol{\ell}_4\boldsymbol{\ell}_5$, $\boldsymbol{\ell}_7 = -\boldsymbol{\ell}_1\boldsymbol{\ell}_2\boldsymbol{\ell}_3\boldsymbol{\ell}_4$, $\boldsymbol{\ell}_8 = -\boldsymbol{\ell}_1\boldsymbol{\ell}_2\boldsymbol{\ell}_3\boldsymbol{\ell}_5$, $\boldsymbol{\ell}_{30} = -\boldsymbol{\ell}_3\boldsymbol{\ell}_5$, $\boldsymbol{\ell}_{31} = -\boldsymbol{\ell}_4\boldsymbol{\ell}_5$).

For both graphs, vertex weights are set to 1 (i.e., $\mathbf{v} = \mathbf{1}$). The corresponding eigenvectors are $\mathbf{u}_j \overset{(101)}{=} Q^{-1/2}\boldsymbol{\ell}_j$, where $Q \overset{(11)}{=} N \cdot 1 = 69{,}120$ (N is the number of training images). These eigenvectors are also binary and allow for a fast computation of the covariance matrix in $\mathcal{O}(LNI^2 + I^3)$ operations, where L is the number of target labels.

The classification error is plotted in Figure 4.6, where the number of slow features d given to a nearest centroid classifier ranges from 4 to 31. For comparison, the clustered graph is also evaluated. For $d = 5$ features, the compact+5 graph results in the best accuracy with an error rate of 11.67%, against 12.42% (compact+31) and 29.74% (clustered). However, the error rate of the compact+5 graph increases if one preserves more than 5 features, indicating that additional features contain little or no discriminative information. For $6 \le d \le 30$, the compact+31 graph yields clearly better accuracy than the other graphs. Inter-

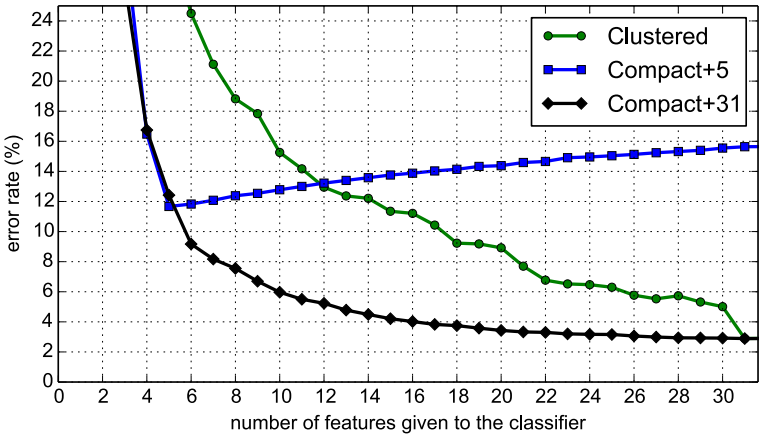


Figure 4.6: Classification error when GSFA is trained with the compact+5, compact+31, and clustered graph (FDA). This error is a function of the graph and the number of slow features d passed to the classifier. For the clustered graph, dropping even a single feature might increase the error rate significantly. For instance, the error rate of using 30 features computed with the clustered graph is worse than the error rate of 13 features computed with the compact+31 graph. Performance on 9,030 test samples, averaged over 10 runs. For $d \geq 4$ the standard error or the mean is at most 0.38%.
[Figure reproduced from (Escalante-B. and Wiskott, 2016b).]

estingly, for $d = 31 = C - 1$ features, the compact+31 and clustered graph give identical error rates of 2.89%, which is their top performance. In this case, the features extracted are *different* but contain the *same information* since they can be mapped to each other linearly. In other words, the first 31 free responses of the two graphs describe the same subspace. Any *single* optimal free response from the compact+31 graph contains 1 bit of discriminative information (which might be redundant to the others). In contrast, the first features extracted by the clustered graph might sacrifice discriminative information to minimize within-class variance (e.g., a feature $\mathbf{y}(c) = \left((\frac{C}{2})^{1/2}, -(\frac{C}{2})^{1/2}, 0, 0, \ldots, 0\right)$ has minimal (zero) within-class variance but provides little discriminative information (less than 1 bit if $C \geq 9$), because most of the time the feature takes the value 0 and otherwise only the first two classes can be identified from it). Using $d > 31$ features does not improve accuracy in any case. For comparison, the highest performance obtained for this database during the official competition is a 0.54% error rate for all 43 signs by Ciresan et al. (2012).

The method of compact discriminative classes provides more accurate label estimations if the feature space is complex enough to allow the extraction of features that approximate the binary labels. If the feature space is poor, the compact graphs might not bring any advantage over the clustered one.

4.5 Discussion of Exact Label Learning

This chapter has introduced the exact label learning (ELL) method, where the labels are used explicitly to construct training graphs. When GSFA is trained with an ELL graph, the final label estimation is just an affine transformation of the slowest extracted feature. Thus, the proposed method allows the direct solution of regression problems, without having to resort to a supervised post-processing step. In other words, given a new input sample (e.g., an input image) the first feature computed using an ELL graph directly provides an approximation of the label (or an affine transformation of it). In practice, even better results may be achieved using more than one feature and supervised post-processing.

Supervised learning problems on high-dimensional data are of great practical importance, but they frequently result in systems with large computational demands. A common approach is to apply feature extraction, dimensionality reduction, and a supervised learning algorithm. A promising alternative approach is hierarchical GSFA (HGSFA), because its complexity scales in some cases even linearly w.r.t. the input dimensionality and the number of samples. In this context, it is especially useful to train HGSFA with an ELL graph since the resulting architecture is simple and homogeneous, as shown in Figure 4.1.c.

A method to analytically compute the optimal free responses of a training graph has been proposed. This method allows us to understand the type of features that can be extracted from a training graph independently of the input and the feature space. Moreover, it has been useful to develop the ELL method, where the labels are explicitly considered to create the training graph. In the resulting ELL graph, the optimal free responses are equal to a normalized version of the labels, and if the feature space is complex enough, HGSFA will learn features that approximate (or span) the original labels.

Graphs with negative edge weights would result in negative transition probabilities, violating the probabilistic interpretation of the graph, and might yield features with negative Δ value, contradicting the notion of slowness. Therefore, it is also shown how to transform a graph to make the edge weights non-negative without altering the extracted features.

The usefulness of the ELL method has been proven by showing three types of applications that are relevant in practice: ELL regression with multiple labels, analysis of training graphs, and classification with compact discriminative features.

It is crucial to emphasize that GSFA optimizes feature slowness, which depends on the particular training graph used, and not label estimation accuracy. However, when the ELL method is used, the training graphs define a slowness objective that requires optimizing an output similarity function where the similarities are intimately related to the desired label similarities. As a consequence, the feature slowness objective and estimation accuracy objective become equivalent when the feature space $\mathcal{F}$ is unlimited. That is, the slowest possible features that can be extracted (i.e. optimal free responses) are equal to a normalized ver-

sion of the label(s). In practice, $\mathcal{F}$ is finite to allow generalization from training to test data and, if the features extracted are slow enough (i.e. close to the optimal free responses), they are also good solutions to the original regression problem. If the slowest feature extracted is not sufficiently similar to the label, one can enhance the mapping from features to labels by mapping more than one output feature, and one can boost feature slowness by including auxiliary labels in the graph construction, as explained in Section 4.5.1.

It is important to underline that the ELL method is not equivalent to linear regression from the data to the (weight decorrelated and normalized) target labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_L$. Any feasible feature vector $\tilde{\mathbf{y}}$ can be decomposed in terms of the optimal free responses $\mathbf{y}_1, \ldots, \mathbf{y}_{N-1}$ as $\tilde{\mathbf{y}} = \sum_{j=1}^{N-1} \alpha_j \mathbf{y}_j$, with $\boldsymbol{\alpha}^T \boldsymbol{\alpha} = 1$ to ensure weighted unit variance. The ELL method ensures that the first L optimal free responses $\mathbf{y}_1, \ldots, \mathbf{y}_L$ are equal to the target labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_L$ and have Δ values $\Delta_1, \ldots, \Delta_L$. The remaining free responses are defined implicitly and have $\Delta_{L<j<N} = 2$. The Δ value of $\tilde{\mathbf{y}}$ can be expressed as $\Delta_{\tilde{\mathbf{y}}} = \sum_{j=1}^{N-1} (\alpha_j)^2 \Delta_j$. Let $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_J$ be concrete output features of GSFA for particular data using an ELL graph. The features $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_J$ are *ordered by slowness*, and $\tilde{\mathbf{y}}_j$ does not necessarily approximate $\mathbf{y}_j$. In particular, $\tilde{\mathbf{y}}_1$ is the slowest possible feature *in the feature space*, and it may be a linear combination of the free responses that is uncorrelated with $\mathbf{y}_1 = \boldsymbol{\ell}_1$ if $\mathbf{y}_1$ cannot be approximated in the feature space (although this extreme case is less likely). In contrast, if one used linear regression, each one of the target labels would be approximated separately (i.e., $\tilde{\mathbf{y}}_j$ would approximate $\mathbf{y}_j$, regardless of the quality of the approximation). This would be mostly disadvantageous when used hierarchically. For instance, if a node in a network has output dimensionality $J < L$ (this scenario is frequent in the lower layers of the network), it is more preferable to preserve the J slowest extractable features than the (eventually poor) linear approximations of $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_J$.

The proposed ELL method explores the limits of HGSFA and is valuable as a theoretical tool for the analysis and design of training graphs. However, the results show that with certain adaptations (e.g., the use of supervised post-processing) it is also sufficiently robust to be applied to practical computer vision and machine learning tasks (although it is generally more costly).

4.5.1 Multiple and Auxiliary Labels

ELL allows learning multiple labels simultaneously, for instance to encode different aspects of the input at once (e.g., object color, size, shape, orientation). The use of multiple labels has been inspired by biological systems, where complementary information channels have been observed and appear to improve feature robustness, for example, under incomplete information (Krüger et al., 2013). Learning gender and color simultaneously yielded clearly smaller estimation errors than when these labels were estimated separately (Section 4.4.1). This shows that multiple label learning is not only theoretically possible, but

that encoding complementary information channels might boost accuracy in practice. For instance, an automatic system for face image processing might benefit from the simultaneous extraction of the subject's identity, age, gender, race, pose, and expression.

One application of multiple labels is learning auxiliary labels derived from the original one (e.g. "higher-frequency" transformations of it). The results show that encoding auxiliary labels improves accuracy (Section 4.4.1). Such a technique is particularly relevant for cascaded or convergent hierarchical GSFA networks, where the outputs of some GSFA nodes feed other nodes. The use of auxiliary labels has been justified based on the fact that these labels contain substantial information about the original label (Section 4.3.4). For instance, as explained before, the first auxiliary label $\boldsymbol{\ell}_2$ only lacks one bit of information about the original label $\boldsymbol{\ell}_1$. Therefore, even if $\boldsymbol{\ell}_1$ does not belong to the feature space of a node, the auxiliary labels might be (approximately) extracted, preserving information about $\boldsymbol{\ell}_1$. A GSFA node (or any supervised learning algorithm) higher in the hierarchy may then be able to approximate $\boldsymbol{\ell}_1$ more accurately by making use of the information carried by the auxiliary labels. Additionally, auxiliary labels have been also justified by a smoothness heuristic, where samples n and n' having similar labels $\ell_1(n)$ and $\ell_1(n')$ should have similar output features $y_j(n)$ and $y_j(n')$, for $1 \leq j \leq J$. Without auxiliary labels only the first output feature would have this property, and the remaining features might vary quickly w.r.t. the original label.

4.5.2 Application of the ELL Method

The experiments on gender (and skin-color) estimation from artificial face images demonstrate that the ELL method also works in practice when used hierarchically.

The experiments of Section 4.4.1 and the analytical results of Section 4.4.2 show the strength of the serial graph when only a single label is available. In this case, the ELL graph provided marginally better estimations than the serial graph (an RMSE of 0.345 with the ELL^g-40 graph vs. 0.349 with the serial graph, in both cases using 3 features, the soft GC post-processing method, and averaging over 10 runs), but the computation time was 25 times longer. The difference between these RMSEs is small but statistically significant (the difference is 1.8 times the sum of the standard errors of the means).

Although the shape of the slowest feature extracted with the serial graph may be less similar to the label, a monotonic transformation of the slowest feature learned by a nonlinear supervised step (e.g. soft GC) may suffice to approximate it.

However, the results suggest that if two or more (intrinsically connected) labels are available, the accuracy of using ELL graphs further increases. Efficient pre-defined graphs are not available in this case. In the gender estimation experiment, the RMSE was improved to 0.277 by jointly learning gender and skin color (ELL^{g,c}-(2×5) graph, 3 features, soft GC). This is much better than the

serial graph. Hence, a particularly promising application for the ELL method is multiple label learning.

Various methods for mapping the slowest feature to a label were tested. The affine mapping method is interesting from a theoretical point of view. However, as one would expect, the soft GC method, which is nonlinear and supervised, provides better accuracy on test data. Therefore, the latter might be preferred in practical applications. Moreover, in this scenario, supervised post-processing methods might be computationally inexpensive, because their input is frequently low-dimensional (e.g., 1 to 3 slow features were used for gender estimation).

4.5.3 Classification with ELL

Although ELL was originally designed for regression, it has been shown that it can also be useful for classification when particular labels are learned. The experiment on traffic sign classification shows the benefit of using compact discriminative features, implemented here by learning multiple binary labels. The resulting system has a much smaller classification error than the clustered graph (equivalent to nonlinear FDA) when the number of output dimensions is fewer than $C-1$, where C is the number of classes. The compactness of the feature set can be useful to do classification with many classes. This is particularly beneficial for hierarchical GSFA because fewer features have to be propagated by the network, which might also reduce overfitting. Although ideally $\log_2(C)$ binary target labels suffice for perfect classification, the experiments show that additional target labels via auxiliary labels improve classification accuracy in practice.

Interestingly, the clustered graph for C classes (equivalent to FDA) and the compact+$(C-1)$ graph are equivalent if the latter is constructed with constant positive eigenvalues $\lambda_1 = \cdots = \lambda_{C-1} = 1/(C-1)$. The reason for this equivalence is that this compact+$(C-1)$ graph would only have within-class transitions, because transitions between different classes cancel out each other. Therefore, the clustered graph can be seen as a special case of the compact+$(C-1)$ graph, with maximum label redundancy ($C-1$ target labels) and giving equal importance (eigenvalues) to all of them.

For simplicity the employed target labels are binary, but it is also possible to use C-valued labels. For instance, the first label can be the class number, and additional labels can be random permutations of this assignment (label decorrelation and normalization still apply). Ideally, these labels might result in an even more compact representation, because a single optimal free response encodes the class information.

Contrary to many approaches for classification based on LPP, the goal of the ELL method is strictly focused on learning the label information while being invariant to any other aspect of the data. Learning the input manifold is not a goal of this dissertation, and this is the main reason (besides scalability and robustness) why nearest neighbors were not used to compute training graphs. However, as shown by the (regression) experiments on simultaneous gender and

color estimation, learning specific additional labels can also be useful to better disentangle the discriminative information.

4.5.4 Efficiency of ELL

Section 4.3.5 explains that the complexity of creating an ELL graph is $\mathcal{O}(L^2N + LN^2)$ operations, where N is the number of samples and $1 \leq L \leq N-1$ is the number of target labels. The complexity of training a single GSFA node with an ELL graph is $\mathcal{O}(IN^2 + I^2N + I^3)$ operations, where I is the input dimensionality (possibly after a nonlinear expansion), and $N > I$ is the number of samples. For comparison, the serial graph has a complexity of $\mathcal{O}(I^2N + I^3)$. Thus, the main limitation of using ELL graphs is the training complexity when N is large. However, this might not be a big disadvantage for the following reasons:
(1) The complexity of the ELL method is comparable to the complexity of LPP. Similarity matrices in LPP are typically computed using nearest neighbors. In practice, the complexity of computing these matrices is similar to $\mathcal{O}(IN^2)$ (He, 2005), and the remaining steps of LPP have complexity $\mathcal{O}(I^2N + I^3)$ if the number of edges is linear w.r.t. N.
(2) The experiment on the estimation of gender shows that it is feasible to apply the ELL method to 10,800 64× 64 images in 250 min (*single thread*, Intel Core i7-870 2.93GHz, 16 GByte RAM). This might be fast enough for some real-world applications.
(3) The ELL method is of theoretical interest in any case, allowing the analysis of training graphs and providing insights for the design of better hand-crafted graphs.

In case better efficiency is still necessary, a few extensions to the ELL method are outlined in Section 4.5.5, two of them trading accuracy for speed.

4.5.5 Extensions of ELL

The following extensions to the ELL method are possible (and may be combined):

(1) Graph trimming. One might compute a sparse approximation of an ELL graph with significantly fewer than $\mathcal{O}(N^2)$ edges. For example, one might delete a fraction of the edges having the smallest weights or a random selection of all the edges. If the resulting number of edges is small, this can be much faster than training using the whole graph.

(2) Sample grouping. Another method first groups the input samples according to their labels, resulting in K groups of N/K samples each. The ELL method is then applied to the average labels of the groups to compute a reduced graph with K vertices and $\approx K^2$ edges. If the largest number of labels L is used, i.e. $L = K-1$, the reduced graph can be constructed in $\mathcal{O}(K^3)$ operations[4]. Af-

[4]In (Escalante-B. and Wiskott, 2016b) the maximum number of labels and the complexity of creating the reduced graph have been stated incorrectly by as $L = I$ and $\mathcal{O}(IK^2 + I^2K)$, respectively.

terwards, one can derive a specialized method to train GSFA using the reduced graph. Such a method considers the transitions between all pairs of samples of two connected groups in the same way as the serial graph. This avoids the explicit computation of the full edge-weight matrix of size $N \times N$. The training complexity would then be $\mathcal{O}(I^2N + I^2K^2 + I^3)$ using $\mathcal{O}(I^2K + NI)$ memory. An interesting value for K is $K = \sqrt{N}$, which divides the training data in $\sqrt{N}$ groups of $\sqrt{N}$ samples each, resulting in $\mathcal{O}(I^2N + I^3)$ operations. The term I^2K in the memory complexity might be large, but one can sacrifice some performance to reduce memory usage, resulting in $\mathcal{O}(I^2N + I^2KN + I^3)$ operations and $\mathcal{O}(I^2 + NI)$ memory.

(3) Combination of graphs. Under some conditions, training graphs can be meaningfully combined: Consider two training graphs that fulfill the consistency restriction (72) and share the same vertices (samples) $\mathbf{x}(n)$ and vertex weights $v(n)$. Let $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ be the corresponding edge weight matrices, and $0 < \alpha < 1$ be a weighting factor. The combined graph has the same vertices and vertex weights, but a combined edge weight matrix $\mathbf{\Gamma}_c \stackrel{\text{def}}{=} \alpha\mathbf{\Gamma}_1 + (1-\alpha)\mathbf{\Gamma}_2$. Assume that $\mathbf{1}^T\mathbf{\Gamma}_1\mathbf{1} = \mathbf{1}^T\mathbf{\Gamma}_2\mathbf{1} = R$ (otherwise the edge weights of one graph can be scaled). Since the vertices and vertex weights of all three graphs are identical, a feature $\mathbf{y}$ that fulfills constraints (78) and (79) for one of the graphs also fulfills these constraints for the remaining two graphs. Let $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1}$ and $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2}$ be the Δ value of $\mathbf{y}$ for the original graphs. Then, $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_c} \stackrel{(76)}{=} \alpha\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1} + (1-\alpha)\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2}$. This implies that if a feature $\mathbf{y}$ has a Δ-value smaller than an arbitrary constant β (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1} < \beta$) and it is not larger than β in the second one (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2} \leq \beta$), it can be warranted that it will be also smaller than β in the combined graph (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_c} < \beta$) for any $0 < \alpha < 1$. This property may be useful to create graphs with optimal free responses that span various labels. In practice, one typically uses $\beta = 2.0$ to apply this extension.

The third extension can be used to combine ELL and/or pre-defined graphs without distinction. The combination of graphs can be used to learn two or more labels simultaneously, which can yield higher label estimation accuracy than learning the labels separately, as shown in the color and gender experiments (Tables 4.2 and 4.3). In fact, this extension will be used in Chapter 5 to combine three efficient pre-defined graphs for face image analysis: two clustered graphs for classification of race and gender, and a serial graph for the estimation of age. The accuracy for age estimation on the MORPH-II database using the combined graph (and an improved version of HGSFA) is a mean average error (MAE) of 3.50 years, which is more accurate than the current state-of-the-art systems for this database.

Chapter 5

Hierarchical Information-Preserving GSFA (HiGSFA)

The previous two chapters have employed graph-based SFA (GSFA) to solve supervised learning problems; in Chapter 3, this task has been solved using predefined graphs, and in Chapter 4, using ELL-graphs. In both cases, hierarchical GSFA (HGSFA) allows the extraction of slow features from high-dimensional data more efficiently than direct GSFA. Moreover, HGSFA can span a more complex feature space than direct GSFA due to the composition of the nonlinearities of the nodes of the network, allows the extraction of slower features, and yields more accurate label estimations.

HGSFA is a promising algorithm for the solution of supervised learning problems on real-world high-dimensional data. However, in this chapter it is shown that HGSFA has a shortcoming: the nodes of the network discard part of the information useful for slowness maximization and label estimation prematurely before it reaches higher nodes. This shortcoming is called unnecessary information loss and it results in suboptimal global slowness and less accurate estimations, because the global feature space is underexploited.

To counteract unnecessary information loss, in this chapter an extension called *hierarchical information-preserving GSFA* (HiGSFA) is proposed, in which information preservation complements the slowness-maximization goal. To evaluate the efficacy of the extension, a 10-layer HiGSFA network is built to estimate human age from facial photographs of the MORPH-II database. The current state-of-the-art performance is improved by HiGSFA achieving a mean absolute error (MAE) of 3.50 years. HiGSFA and HGSFA support multiple-labels and offer a rich feature space, feed-forward training, and linear complexity in the number of samples and dimensions. However, HiGSFA outperforms HGSFA in terms of feature slowness, estimation accuracy and input reconstruction, giving rise to a more promising hierarchical supervised-learning approach.

The main contributions of this chapter are the following: (1) It is proven that HGSFA networks can provide linear complexity w.r.t. the number of training samples and their dimensionality. (2) Two fundamental shortcomings of

HGSFA are revealed: unnecessary information loss and poor input reconstruction. (3) The concept of information preservation is introduced to HGSFA networks, giving rise to the HiGSFA algorithm, which counteracts the two shortcomings above. (4) The improved capabilities of HiGSFA over HGSFA are verified empirically, and (5) the problem of age estimation is solved with state-of-the-art accuracy.

The remainder of the chapter is organized as follows. The main ideas behind HiGSFA are introduced, and related work is briefly discussed. Then, a deeper analysis of the advantages and limitations of HGSFA is presented. Afterwards, the HiGSFA algorithm is proposed, and it is evaluated experimentally. A discussion section concludes the chapter.

5.1 Introduction

A common problem in machine learning is the high computational cost of most algorithms when the data is high-dimensional. This is also the case when GSFA is applied to the data directly (i.e., direct GSFA), because it has cubic complexity w.r.t. the number of dimensions. However, processing high-dimensional data is still practical if one resorts to hierarchical GSFA (HGSFA), see Section 2.4. Additional advantages of HGSFA over direct GSFA include lower memory complexity, a more complex global nonlinear feature space spanned by the network, and less overfitting due to the local extraction of slow features, see Figure 2.2.

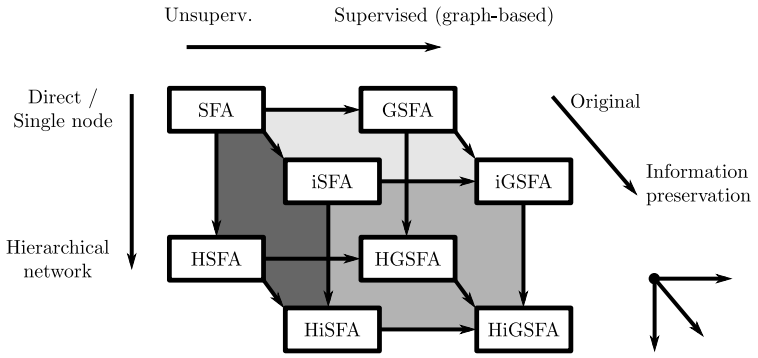


Figure 5.1: Illustration of three different extensions to SFA (graph-based, hierarchical, information-preserving). Each extension is represented by a different direction. The combination of extensions results in 8 different versions of SFA. This chapter proposes information preservation, which is used in iSFA, iGSFA, HiSFA, and HiGSFA, the latter being the most promising version of SFA.

This chapter shows that HGSFA suffers from a shortcoming: The GSFA nodes (the separate instances of GSFA that process the low-dimensional data chunks in the network) may prematurely discard features that are not slow at a local level but that would have been useful to improve global slowness (i.e., the slowness of the final output features) if combined in subsequent nodes with

other features originating in other nodes. This drawback, which is referred to as unnecessary information loss, leads to an under-exploited feature space, i.e., the global feature space contains slower features than those actually extracted by HGSFA.

To reduce unnecessary information loss in HGSFA, I suggest to complement slowness with information preservation (i.e., maximization of mutual information between the input data and the output features). For simplicity and efficiency, this idea is implemented here as the minimization of a reconstruction error. The resulting network that considers both optimization goals is called *hierarchical information-preserving GSFA* (HiGSFA), and the algorithm constituting each node of the network is called *information-preserving GSFA* (iGSFA). The feature vector computed by iGSFA has two parts with different types of components: (1) a slow part, which is a linear transformation of the (nonlinear) features computed using GSFA, and (2) an input-reconstructive part computed using PCA.

The iGSFA algorithm employed by HiGSFA reduces the redundancy between the slow and reconstructive parts; the reconstructive part does not directly approximate the input data but only a version of them where the slow part has been projected out, called residual data. This ensures that both parts are decorrelated. Moreover, iGSFA also ensures that the scale of the slow components is compatible with the scale of the reconstructive components. This enables meaningful processing by PCA in subsequent layers. Different versions of SFA with and without information preservation are shown in Figure 5.1. The full list of principles and heuristics behind HiGSFA is presented in Table 1.1.

The experiments show the advantages of HiGSFA over HGSFA: (1) slower features, (2) better generalization to unseen data, (3) much better input reconstruction (see Figure 5.2), and (4) improved accuracy for the supervised learning problem. Furthermore, the computational and memory requirements of HiGSFA have the same asymptotic order as those of HGSFA.

5.2 Related work

HiGSFA is the main extension to SFA proposed in this chapter and the most specialized algorithm of this dissertation. HiGSFA belongs to supervised dimensionality reduction (supervised DR), where existing algorithms include: Fisher discriminant analysis (FDA) (Fisher, 1936), local FDA (LFDA) (Sugiyama, 2006), pairwise constraints-guided feature projection (PCGFP) (Tang and Zhong, 2007), semi-supervised dimensionality reduction (SSDR) (Zhang et al., 2007), and semi-supervised LFDA (SELF) (Sugiyama et al., 2010).

Previous extensions to SFA include extended SFA (xSFA) (Sprekeler et al., 2010), generalized SFA (genSFA) (Sprekeler, 2011) and graph-based SFA (GSFA, Chapter 3) (Escalante-B. and Wiskott, 2010, 2012, 2013). HiGSFA extends hierarchical GSFA (HGSFA) by adding information preservation.

Figure 5.2: (a) An image from a private database after pose normalization. (b) The same image fully pre-processed (i.e., after pose normalization and face sampling, 96×96 pixels). Linear reconstructions on 75 features extracted with either (c) PCA, (d) HiGSFA or (e) HGSFA. (f) Average over all pre-processed images of the MORPH-II database. Notice that the reconstruction using HiGSFA features is most similar to that of PCA, whereas the reconstruction using HGSFA features is most similar to the average image.

5.3 Advantages and Limitations of Hierarchical Processing by HSFA and HGSFA Networks

This section analyzes HSFA networks in terms of their advantages, particularly their computational complexity, and their limitations, particularly unnecessary information loss. The focus of this section is on HSFA, but HGSFA is covered by extension and specifically addressed later. The central goal behind HiSFA and HiGSFA is to overcome two shortcomings of HSFA and HGSFA. Therefore, the analysis below of limitations of HSFA is crucial, since it justifies the extensions with information preservation proposed in Section 5.4.

5.3.1 Advantages of HSFA and HGSFA Networks

The central advantages of hierarchical processing—compared to direct SFA—have been mentioned in Section 2.4: (1) It reduces overfitting and can be seen as a regularization method, as illustrated in Figure 2.2. (2) The nonlinearity of the nodes accumulates in a compositional manner while the data traverses the nodes of the network, so that even when using simple expansions the network as a whole may describe a highly nonlinear feature space. (3) HSFA has better computational efficiency than SFA.

Some remarks about these advantages are pertinent: Advantage (1) is explained by the fact that the input dimensionality to the nodes of the hierarchical network is much smaller than the original input dimensionality, whereas the

number of samples remains unchanged. Thus, individual nodes are less susceptible to overfitting. As a consequently, the gap in generalization performance between HSFA and direct SFA increases when polynomial expansions are involved, because the large dimensionality of the expanded data in direct SFA translates into stronger overfitting.

Advantage (2) is valuable in practice, because most real-life problems are nonlinear (e.g., all problems involving translation invariance and inhomogeneous backgrounds, since a simple linear mask cannot selectively ignore parts of the input depending on the current object position). A complex feature space may be necessary to extract the slowest hidden parameters and solve the supervised problem with good accuracy.

Advantage (3) is addressed more precisely by recalling the computational complexity of SFA and GSFA in the following paragraphs. This complexity can then be compared with the complexity of a particular HSFA network (Section 5.3.2). The focus is on the training complexity rather than on the complexity of feature extraction, because the former can be considerable in practice, whereas the latter is relatively lightweight in both HSFA and direct SFA. Following standard notation of algorithm complexity, computational (time) complexity is denoted by T (e.g., T_{SFA}), and memory (space) complexity is denoted by S (e.g., S_{SFA}). As stated in Section 2.3, training linear SFA has a time (computational) complexity

$$T_{\mathrm{SFA}}(N, I) = \mathcal{O}(NI^2 + I^3)\,, \tag{128}$$

where N is the number of samples and I is the input dimensionality (possibly after a nonlinear expansion). The same complexity holds for GSFA if one uses an efficient training graph (e.g., the serial graphs or clustered graph, see Sections 3.3 and 3.4), otherwise (for arbitrary graphs) it can be as large as

$$T_{\mathrm{GSFA}}(N, I) = \mathcal{O}(N^2I^2 + I^3)\,. \tag{129}$$

For large I (i.e., high-dimensional data) direct SFA and direct GSFA are therefore inefficient. However, their complexity can be reduced by using HSFA and HGSFA. The exact resulting complexity of HSFA and HGSFA depends on the structure and parameters of the hierarchical network. It is proven below that it can be linear in I and N for certain networks.

5.3.2 Complexity of a Quadratic HSFA Network

Although existing systems based on HSFA have resorted to hierarchical processing for efficiency reasons, apparently its actual asymptotic complexity has not yet been formally established. In this section, the computational complexity of a concrete quadratic HSFA (QHSFA) network is computed, see Figure 5.3. This network has L layers and operates on data with a 1D structure (i.e., vectors). All nodes of the network perform quadratic SFA (QSFA, i.e., a quadratic expansion followed by linear SFA). The receptive field, fan-in, and stride of the

nodes in layer 1 is k input values, where k is a fixed parameter. In the rest of the layers, the fan-in and stride of the nodes is 2 nodes. Assume that the total input dimensionality is I and that each node reduces the dimensionality of the data from k input components to $k/2$ output components.

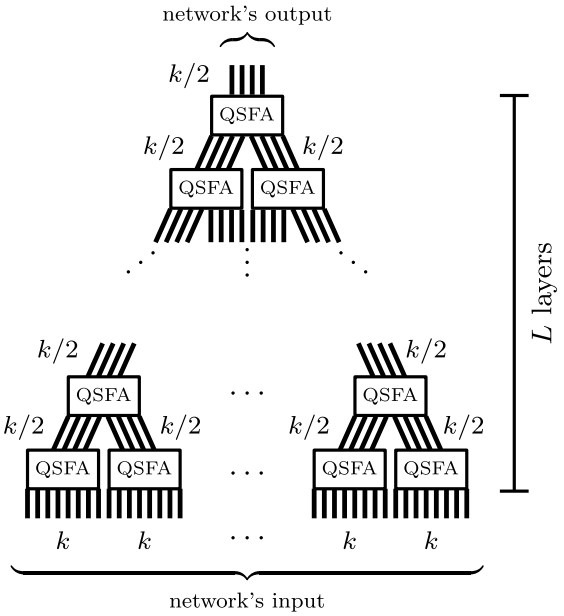


Figure 5.3: Example of a 1D QHSFA network with binary fan-ins and no overlap. Each node performs quadratic SFA and reduces the dimensionality from k to $k/2$ components. A small fan-in results in a network with a large number L of layers and is, thus, useful to build deep networks.

From the structure described above, it follows that the receptive fields of the nodes are non-overlapping, the network's output has $k/2$ components, the number of layers L is related to I and k:

$$I \;=\; k2^{L-1}\,, \tag{130}$$

and the total number of nodes in the network is

$$M \;=\; 2^L - 1\,. \tag{131}$$

The input to each QSFA node is k dimensional. The quadratic expansion increases the number of dimensions to $k(k+3)/2$ components. Afterwards, linear SFA reduces the dimensionality to $k/2$. From the information above, one can compute the complexity of training a single (nonlinear) node:

$$T_{\mathrm{QSFA}}(N,k) \overset{(128)}{=} \mathcal{O}(N(k(k+3)/2)^2 + (k(k+3)/2)^3) \tag{132}$$

$$= \mathcal{O}(Nk^4 + k^6)\,. \tag{133}$$

The number of nodes is $M \overset{(130,131)}{=} \mathcal{O}(I/k)$. Therefore, the complexity of training the whole network is

$$T_{\text{QHSFA}}(N,I,k) \overset{(133)}{=} \mathcal{O}((Nk^4 + k^6)I/k) = \mathcal{O}(INk^3 + Ik^5)\,. \tag{134}$$

Thus, the complexity of the complete QHSFA network above is linear w.r.t. the input dimension I, whereas the complexity of direct QSFA is

$$T_{\text{QSFA}}(N,I) \overset{(133)}{=} \mathcal{O}(NI^4 + I^6)\,, \tag{135}$$

which is linear w.r.t. I^6. Therefore, the QHSFA network is computationally much more efficient than direct QSFA.

Since each layer in the QHSFA network is quadratic, in general the output features of layer l can be written as polynomials of degree 2^l on the input values. In particular, the output features of the whole network are polynomials of degree 2^L. However, the actual feature space spanned by the network does not include all polynomials of this degree but only a subset of them due to the restricted connectivity of the network. In contrast, direct QSFA only contains quadratic polynomials (although all of them).

One could try to train direct SFA on data expanded by a polynomial expansion of degree 2^L (to encompass the feature space of QHSFA), but the complexity would be prohibitive: The expanded dimensionality would be $\sum_{d=0}^{2^L} \binom{d+I-1}{I-1}$. Thus, the last term of the summation is $\binom{2^L+I-1}{I-1} = \frac{(2^L+I-1)!}{2^L!(I-1)!}$. Assuming this term dominates the others, letting $d \overset{\text{def}}{=} 2^L$, and assuming $1 \ll k \ll d \ll I$, one can use Stirling's formula ($n! \approx \sqrt{2\pi n}(n/e)^n$) as well as other simplifications to crudely approximate the expanded dimensionality as $\frac{1}{\sqrt{2\pi d}}(e(1+k/2))^d$ (k enters into the equation, because $k \overset{(130)}{=} I/2^{L-1}$). Substituting $d \overset{(130)}{=} 2I/k$ yields $\frac{1}{2\sqrt{\pi I/k}}(e(1+k/2))^{2I/k}$ dimensions. Thus, the complexity of training SFA with an expansion of degree 2^L would be approximately $\mathcal{O}(N(e(1+k/2))^{4I/k} + (e(1+k/2))^{6I/k})$, being even less feasible than direct QSFA.

The memory (space) complexity of linear SFA is

$$S_{\text{SFA}}(N,I) = \mathcal{O}(I^2 + NI)\,, \tag{136}$$

where the term NI is due to the input data. One can reduce this complexity by using HSFA and by training the nodes separately, one at a time, independently of whether an expansion is used or not. For instance, the memory complexity of direct QSFA is

$$S_{\text{QSFA}}(N,I) = \mathcal{O}(I^4 + NI)\,, \tag{137}$$

whereas the memory complexity of the QHSFA network is only

$$S_{\text{QHSFA}}(N,I,k) = \mathcal{O}(k^4 + NI)\,. \tag{138}$$

The excellent computational and memory complexity of the QHSFA network is not exclusive to this simple architecture. It is possible to design more sophisticated networks and preserve a similar computational complexity. For example, the network that is proposed in Section 5.5 has overlapping receptive fields, larger fan-ins, and a 2D structure, but its complexity is also linear in I and N (concrete analysis not provided).

5.3.3 Limitations of HSFA and HGSFA Networks

In spite of the remarkable advantages of HSFA and HGSFA networks, they also have some shortcomings in their current form. The analysis below focuses on HSFA, but it applies to all networks in which the nodes have only one criterion for DR, namely, slowness maximization, including HGSFA. Besides the slowness maximization objective, no other limitation is imposed to the nodes; they might be linear or nonlinear, include additive noise, clipping, various passes of SFA, etc.

It is shown here that relying only on slowness to determine which aspects of the data that are preserved results in two shortcomings: unnecessary information loss and poor input reconstruction, explained below.

(1) Unnecessary information loss. This shortcoming occurs when the nodes of the network discard dimensions of the data that are not significantly slow locally (i.e., at the node level), but which would have been useful for slowness optimization by nodes at higher levels of the network if they had been preserved and combined with other dimensions.

The following minimal theoretical experiment shows that dimensions crucial to extract global slow features are not necessarily slow locally. Consider four zero-mean, unit-variance signals: $s_1(t)$, $s_2(t)$, $s_3(t)$ and $n(t)$ that can only take binary values, either -1 or $+1$ (n stands for noise here, and t is time, or more precisely, sample number). Assume the signals are ordered by slowness ($\Delta_{s_1} < \Delta_{s_2} < \Delta_{s_3} < \Delta_n = 2.0$)[1] and these signals are statistically independent. The same holds for GSFA if the graph is consistent and has no self-loops. Let the 4-dimensional input to the network be $(x_1, x_2, x_3, x_4) \stackrel{\text{def}}{=} (s_2, s_1 n, s_3, n)$ and assume the number of samples is large enough.

The direct application of QSFA to this data would allow us to extract the slowest possible feature, namely, $x_2 x_4 = (s_1 n)n = s_1$ (or equivalently $-x_2 x_4$). However, let us assume that a 2-layer QHSFA network with 3 nodes is used, where the output of the network is: $\text{QSFA}\big(\text{QSFA}(s_2, s_1 n), \text{QSFA}(s_3, n)\big)$. Each QSFA node reduces the number of dimensions from 2 to 1. Since $\Delta_{s_2} < \Delta_{s_1 n} =$

[1] One can show that the expected Δ value of a random unit-variance i.i.d. noise feature is 2.0, see Section 4.3.2.

2.0, the first bottom node computes $\mathrm{QSFA}(s_2, s_1 n) = s_2$, and since $\Delta_{s_3} < \Delta_n = 2.0$, the second bottom node computes $\mathrm{QSFA}(s_3, n) = s_3$. The top node would then extract $\mathrm{QSFA}(s_2, s_3) = s_2$. Therefore, the network would miss the slowest feature, s_1, even though it actually belongs to the feature space spanned by the network.

The problem can be expressed in information theoretic terms:

$$I(s_1 n, s_1) = 0\,, \text{ and} \tag{139}$$

$$I(n, s_1) = 0\,, \text{ but} \tag{140}$$

$$I((s_1 n, n), s_1) = H(s_1) > 0\,, \tag{141}$$

where H is entropy, and I denotes mutual information[2]. Equations (139)–(141) show that it is impossible to *locally* rule out that a feature contains information that might yield a slow feature (in this case s_1), unless one *globally* observes the whole data available to the network. The problem above could be solved if the network could preserve s_1 and $s_1 n$ by applying other criteria besides slowness. For example, if the signals above were instead $10 s_1$ and $10 s_1 n$, one could distinguish these features based on their variance.

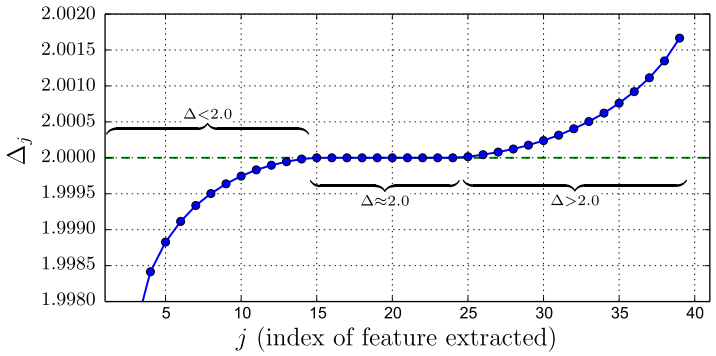


Figure 5.4: Δ values of the first 40 slow features of an HGSFA network trained for age estimation and averaged over all the nodes of the first layer ($\Delta_1 = 1.859$, $\Delta_2 = 1.981$, and $\Delta_3 = 1.995$, not shown). The training graph employed is a serial graph with 32 groups (see Section 5.5.3). Most Δ values are close to 2.0, indicating that at this early stage, where the nodes have small 6×6-pixel receptive fields, the slowest extracted features are not substantially slow.

Unnecessary information loss can also affect applications in practice. As an example, consider the problem of age estimation from human face images. Figure 5.4 shows the Δ values of the slowest features extracted by the first layer of an HGSFA network trained for age estimation. Most Δ values are

[2] $H(X)$ is the average amount of information given by instances of a random variable X. $I(X,Y)$ is the average amount of information that a random variable X gives about another random variable Y (or vice-versa). In other words, it denotes how much information is duplicated in X and Y on average. If $I(X,Y) = 0$, the variables are independent.

approximately 2.0, and only a few of them are less than 2.0. The value 2.0 is crucial; a feature with $\Delta = 2.0$ can be a transformation of the input data, a transformation of inherent noise, or a mixture of both. In fact, if two or more feasible features have the same Δ value, GSFA outputs an arbitrary rotation of them, even though one might prefer features that are transformations of the input rather than noise. Due to DR only a few features with $\Delta = 2.0$ can be preserved, and these may include "noise" features. Some of the discarded features with $\Delta \geq 2.0$ might still have contained useful information (that could have potentially resulted in slower features in a subsequent node).

One might try to preserve a large number of features to reduce information loss. However, this might be impractical because it would increase the computational cost and contradicts the goal of dimensionality reduction.

(2) Poor input reconstruction. Input reconstruction is the task of generating an (artificial) input with a given feature representation (or an approximation of it). Wilbert (2012) has studied this task for HSFA, and used it to visualize which features the network is sensitive to.

In the field of image processing, reconstruction might be relevant for image morphing and interpolation. In this work, morphing is defined as the task of finding how the input must be modified to reflect modifications introduced to the output features. For example, assume SFA was trained to extract body mass index (BMI) from facial images. Morphing would allow us to visualize how a particular person would look after losing or gaining a few kg.

Experiments have shown that input reconstruction from top-level features extracted by HSFA is a challenging task (Wilbert, 2012). The difficulty of this task has also been confirmed by several experiments conducted for this thesis using various nonlinear methods for input reconstruction, including local and global methods.

It is shown here that poor input reconstruction may not be caused by the weakness of the reconstruction algorithms employed but rather by insufficient reconstructive information in the slow features: The extracted features ideally depend only on the hidden slow parameters and are invariant to any other factor. In the BMI estimation example, the extracted features would be strongly related to the BMI and harmonic functions of it (assuming the extracted features are close to the optimal free responses predicted by theory). Thus, they would be mostly invariant to other factors, such as the identity of the person, his or her facial expression, the background, etc. Therefore, in theory only BMI information would be available for reconstruction.

In practice, residual information about the input data can still be found in the extracted features. However, one cannot rely on this information because it may be partial, making reconstructions not unique, and highly nonlinear, making it difficult to untangle it. Even the features extracted by linear SFA typically result in inaccurate reconstructions. HSFA consists on many layers of SFA nodes, potentially aggravating the problem.

One exception where reconstruction is possible is when SFA is trained with artificial data generated using only a set of slowly changing parameters. In this setup, the output features might encode generative parameters that allow input reconstruction using an appropriate reconstruction method.

The connection between the problems of unnecessary information loss and poor input reconstruction is evident if one distinguishes between two types of information: (a) information about the full input data and (b) information about the global slow parameters. Losing (a) results in poor input reconstruction, whereas losing (b) results in unnecessary information loss. Of course, (a) contains (b). Therefore, both problems originate from losing different but related types of information.

The extensions proposed in the next section counteract unnecessary information loss and poor input reconstruction.

5.4 Hierarchical Information-Preserving GSFA (HiGSFA)

This section formally proposes HiGSFA, an extension to HGSFA that counteracts the problems of unnecessary information loss and poor input reconstruction by extracting reconstructive features in addition to slow features. HiGSFA is a hierarchical implementation of iGSFA. For simplicity, the presentation below focuses on iSFA, but iSFA can be trivially extended into iGSFA and HiGSFA. Information preservation is denoted with a lowercase 'i' to prevent a name clash between iSFA and independent SFA (ISFA) (Blaschke et al., 2007).

iSFA combines two learning principles: the slowness principle and information preservation, without compromising the former in any way. Information preservation requires the maximization of the mutual information between the output features and the input data. However, for finite, discrete, and typically unique data samples, it is difficult to measure and maximize mutual information unless one assumes a specific probability model. Therefore, information preservation is implemented more practically as the minimization of a reconstruction error. A closely related concept is the explained variance, but such a term is avoided here because it is typically restricted to linear transformations.

The rest of the section presents a high-level description of iSFA, describes its construction in detail, shows how to approximate an inverse transformation, and discusses the main properties of iSFA. Finally, the extension of iSFA into iGSFA and HiGSFA is discussed.

5.4.1 Algorithm Overview (iSFA)

The goal of HiSFA is to improve feature extraction of HSFA networks at the node and global level. This goal is pursued by replacing the SFA nodes by iSFA nodes, leaving the network structure unchanged (although one can tune the network structure to achieve better accuracy, if desired).

The feature vectors computed by iSFA are composed of two parts: (1) a slow part derived from SFA features, and (2) a reconstructive part derived from principal components (PCs). Generally speaking, the slow part captures the slow aspects of the data and is basically composed of standard SFA features, except for an additional linear mixing step to be explained in Sections 5.4.2 and 5.4.4. The reconstructive part ignores the slowness criteria and focuses instead on linearly describing the input as closely as possible (disregarding the part already described by the slow part). In Section 5.5 it is shown that, although the reconstructive features are not particularly slow, they indeed contribute to global slowness maximization.

The proposed algorithm takes care of the following important considerations: (a) Given the output dimension D, it decides how many features the slow and reconstructive part should contain. (b) It minimizes the redundancy between the slow and the reconstructive part, allowing the output features to be more compact and have higher information content. (c) It corrects the amplitudes of the slow features (SFA features have unit variance) to make them compatible with the PCs and allow their processing by PCA in subsequent nodes (PCA is a rotation and projection. Thus, it preserves the amplitude of the original data in the retained directions).

5.4.2 Algorithm Description (Training Phase of iSFA)

In this section, the details of iSFA, or more precisely of its training phase, are presented. Algorithm 1 describes the whole algorithm compactly. Figure 5.5 shows the employed components.

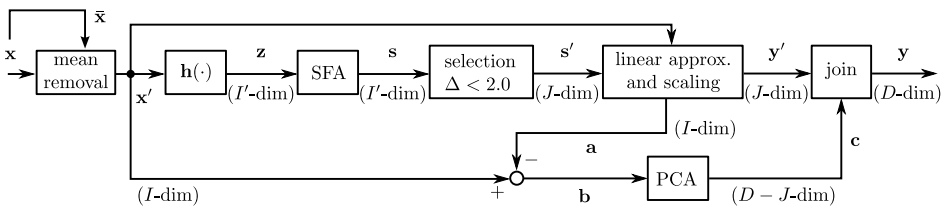


Figure 5.5: Block diagram of the iSFA node showing the components used for training and feature extraction. Signal dimensionalities are given in pharenthesis. The components and signals are explained in the text.

Let $\mathbf{X} \stackrel{\text{def}}{=} (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ be the I-dimensional training data, D the output dimensionality, $\mathbf{h}(\cdot)$ the nonlinear expansion function, and $\Delta_T \approx 2.0$ (a Δ-threshold, in practice slightly smaller than 2.0). The iSFA algorithm does the following. First, the average sample $\bar{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{N}\sum_n \mathbf{x}_n$ is removed from the N training samples resulting in the centered data $\mathbf{X}' \stackrel{\text{def}}{=} \{\mathbf{x}'_n\}$, with $\mathbf{x}'_n \stackrel{\text{def}}{=} \mathbf{x}_n - \bar{\mathbf{x}}$, where $1 \le n \le N$. Then, $\mathbf{X}'$ is expanded via $\mathbf{h}(\cdot)$, resulting in vectors $\mathbf{z}_n = \mathbf{h}(\mathbf{x}'_n)$ of dimensionality I'. Afterwards, linear SFA is trained with the expanded data $\mathbf{Z} \stackrel{\text{def}}{=} \{\mathbf{z}_n\}$, resulting in a weight matrix $\mathbf{W}_{\text{SFA}}$ and an average input vector

$\bar{\mathbf{z}}$. The slow features extracted from $\mathbf{Z}$ are $\mathbf{s}_n = \mathbf{W}_{\text{SFA}}^T(\mathbf{z}_n - \bar{\mathbf{z}})$. The first J components of $\mathbf{s}_n$ with $\Delta < \Delta_T$ and $J \leq \min(I', D)$ are denoted $\mathbf{s}'_n$. The remaining components of $\mathbf{s}_n$ are discarded.

Algorithm 1 Training phase of iSFA

Require: $D > 0$: output dimensionality

procedure TRAIN($\mathbf{X}$) ▷ $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_N})$: training samples
  $\forall n : \mathbf{x'_n} \leftarrow \mathbf{x_n} - \bar{\mathbf{x}}$ ▷ $\bar{\mathbf{x}}$: average sample
  $\forall n : \mathbf{z_n} \leftarrow \mathbf{h}(\mathbf{x'_n})$ ▷ $\mathbf{Z} = (\mathbf{z_1}, \ldots, \mathbf{z_N})$: expanded samples
  $\mathbf{W}_{\text{SFA}}, \bar{\mathbf{z}} \leftarrow$ SFA.TRAIN($\mathbf{Z}$, OUTPUT_DIM $= \min(I', D)$)
  $\forall n : \mathbf{s_n} \leftarrow {\mathbf{W}_{\text{SFA}}}^T(\mathbf{z_n} - \bar{\mathbf{z}})$
  $\forall n : \mathbf{s'_n} \leftarrow (s_{n1}, \ldots, s_{nJ})^T$ ▷ Preserve the first J features with $\Delta < \Delta_T$
  $\forall n : \mathbf{a_n} \leftarrow \mathbf{M}\mathbf{s'_n} + \mathbf{d}$ ▷ For $\mathbf{M}$ and $\mathbf{d}$, such that $\mathbf{M}\mathbf{S}' + \mathbf{b} \approx \mathbf{X}'$
  $\forall n : \mathbf{y'_n} \leftarrow \mathbf{R}\mathbf{s'_n}$ ▷ For $\mathbf{QR} = \mathbf{M}$, the QR decomposition of $\mathbf{M}$
  $\forall n : \mathbf{b_n} \leftarrow \mathbf{x'_n} - \mathbf{a_n}$
  $\mathbf{W}_{\text{PCA}} \leftarrow$ PCA.TRAIN($\mathbf{B}$, OUTPUT_DIM $= D - J$)
  $\forall n : \mathbf{c_n} \leftarrow \mathbf{W}_{\text{PCA}}^T\mathbf{b_n}$ ▷ Only $D - J$ PCs are preserved
  $\forall n : \mathbf{y_n} = \mathbf{y'_n}|\mathbf{c_n}$ ▷ Concatenation of slow and reconstructive parts
  return $\mathbf{Y} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$, $\bar{\mathbf{x}}$, $\mathbf{W}_{\text{SFA}}$, $\bar{\mathbf{z}}$, $\mathbf{W}_{\text{PCA}}$, J, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{d}$
end procedure

The J features comprised by $\mathbf{S}' \stackrel{\text{def}}{=} \{\mathbf{s}'_n\}$ have zero mean and unit variance. The next steps correct the amplitude of $\mathbf{S}'$: The centered data $\mathbf{X}'$ is approximated from $\mathbf{S}'$ linearly by using ordinary least squares to compute a matrix $\mathbf{M}$ and a vector $\mathbf{d}$, such that

$$\mathbf{A} \stackrel{\text{def}}{=} \mathbf{M}\mathbf{S}' + \mathbf{d}\mathbf{1}^T \approx \mathbf{X}', \tag{142}$$

where $\mathbf{A}$ is the approximation of the centered data given by the slow part (i.e., $\mathbf{x}'_n \approx \mathbf{a}_n \stackrel{\text{def}}{=} \mathbf{M}\mathbf{s}'_n + \mathbf{d}$) and $\mathbf{1}$ is a vector of 1s of length N. Since $\mathbf{X}'$ and $\mathbf{S}'$ are centered, $\mathbf{d}$ could be discarded because $\mathbf{d} = \mathbf{0}$. However, when GSFA is used the slow features have only *weighted* zero mean, and $\mathbf{d}$ might improve the approximation of $\mathbf{X}'$. Afterwards, the QR decomposition of $\mathbf{M}$

$$\mathbf{M} = \mathbf{Q}\mathbf{R} \tag{143}$$

is computed, where $\mathbf{Q}$ is orthonormal and $\mathbf{R}$ is upper triangular. Then, the (amplitude-corrected) slow feature part is computed as

$$\mathbf{y}'_n = \mathbf{R}\mathbf{s}'_n\,. \tag{144}$$

Section 5.4.4 justifies the mixing and scaling (144) of the slow features $\mathbf{s}'_n$.

To obtain the reconstructive part, residual data $\mathbf{b}_n \stackrel{\text{def}}{=} \mathbf{x}'_n - \mathbf{a}_n$ is computed, i.e., the data that remains after the data linearly reconstructed from $\mathbf{y}'_n$ (or $\mathbf{s}'_n$) is removed from the centered data. Afterwards, PCA is trained with $\{\mathbf{b}_n\}$,

resulting in a weight matrix $\mathbf{W}_{\text{PCA}}$. There is no bias term, because $\mathbf{b}_n$ is centered. The reconstructive part $\mathbf{c}_n$ is then defined as the first $D-J$ principal components of $\mathbf{b}_n$ and computed accordingly.

Thereafter, the slow part $\mathbf{y}'_\mathbf{n}$ (J features) and the reconstructive part $\mathbf{c_n}$ ($D-J$ features) are concatenated, resulting in the D-dimensional output features $\mathbf{y_n} \stackrel{\text{def}}{=} \mathbf{y}'_\mathbf{n}|\mathbf{c_n}$, where $|$ is vector concatenation.

Finally, the algorithm returns $\mathbf{Y} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$, $\bar{\mathbf{x}}$, $\mathbf{W}_{\text{SFA}}$, $\bar{\mathbf{z}}$, $\mathbf{W}_{\text{PCA}}$, J, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{d}$. The output features $\mathbf{Y}$ are usually computed only during feature extraction (see Algorithm 2). Still, they are kept here to simplify the understanding of the signals involved.

5.4.3 Feature Extraction by iSFA

The feature extraction algorithm is similar to the training algorithm, except that the parameters $\bar{\mathbf{x}}$, $\mathbf{W}_{\text{SFA}}$, $\bar{\mathbf{z}}$, $\mathbf{W}_{\text{PCA}}$, J, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{d}$ have already been learned from the training data. Algorithm 2 shows how a single input sample is processed, however, it can be easily and efficiently adapted to process multiple input samples by taking advantage of matrix operations.

Algorithm 2 Feature extraction with iSFA

Require: D, $\bar{\mathbf{x}}$, $\mathbf{W}_{\text{SFA}}$, $\bar{\mathbf{z}}$, $\mathbf{W}_{\text{PCA}}$, J, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{d}$

procedure EXTRACT($\mathbf{x}$) ▷ $\mathbf{x}$: a new sample
  $\mathbf{x}' \leftarrow \mathbf{x} - \bar{\mathbf{x}}$ ▷ $\bar{\mathbf{x}}$: mean of the training data
  $\mathbf{z} \leftarrow \mathbf{h}(\mathbf{x}')$
  $\mathbf{s} \leftarrow \mathbf{W}_{\text{SFA}}^T(\mathbf{z_n} - \bar{\mathbf{z}})$ ▷ Extract the first J slow features
  $\mathbf{s}' = (s_1, s_2, \ldots, s_J)$
  $\mathbf{y}' \leftarrow \mathbf{R}\mathbf{s}'$
  $\mathbf{a} \leftarrow \mathbf{Q}\mathbf{y}' + \mathbf{d}$
  $\mathbf{b} \leftarrow \mathbf{x}' - \mathbf{a}$
  $\mathbf{c} \leftarrow \mathbf{W}_{\text{PCA}}^T\mathbf{b}$
  $\mathbf{y} = \mathbf{y}'|\mathbf{c}$
  return $\mathbf{y}$
end procedure

5.4.4 Mixing and Scaling of Slow Features

In the iSFA algorithm, the J-dimensional slow features $\mathbf{s}'_n$ are transformed into the scaled $\mathbf{y}'$ features. Such a transformation is necessary to make the amplitude of the slow features compatible with the amplitude of the PCA features, so that PCA processing of the two sets of features together is possible and meaningful in the next layers.

A scaling method should ideally offer two key properties of PCA. (1) If one adds unit-variance noise to one of the output features (e.g., principal components), the variance of the reconstruction error also increases by one unit. (2) If

one adds independent noise to two or more output features simultaneously, the variance of the reconstruction error increases additively.

The *QR scaling*, used by Algorithm 1, as well as a *sensitivity-based scaling*, are explained below. Both methods ensure that the amplitude of the slow features is approximately equal to the reduction in the reconstruction error that they allow. In practice, a lower bound on the scales (not shown in the pseudocode) ensures that all features have amplitudes > 0 even if they do not contribute to reconstruction.

Consider first the QR scaling method (142)–(144). The input can be linearly approximated as $\tilde{\mathbf{x}} = \tilde{\mathbf{a}} + \tilde{\mathbf{b}} + \bar{\mathbf{x}}$, where $\tilde{\mathbf{a}} = \mathbf{Q}\mathbf{y}' + \mathbf{d}$ and $\tilde{\mathbf{b}} = \mathbf{W}_{\text{PCA}}\mathbf{c}$ (see Section 5.4.5 and recall that $\mathbf{W}_{\text{PCA}} = (\mathbf{W}_{\text{PCA}}^T)^{-1}$). Approximations are denoted here using tilded variables. The vector $\mathbf{y} = \mathbf{y}'|\mathbf{c}$ fulfills the two key properties of reconstruction of PCA above because matrix $\mathbf{Q}$ and matrix $\mathbf{W}_{\text{PCA}}$ are orthogonal, and because the rows of the two matrices are mutually orthogonal.

One small drawback is that (144) mixes the slow features. Polynomial expansion functions combined with SFA are invariant to invertible linear transformations (e.g., $\text{SFA}(\text{QExp}(\mathbf{U}\mathbf{x})) \equiv \text{SFA}(\text{QExp}(\mathbf{x}))$, where $\mathbf{U}$ is any invertible matrix and QExp is the quadratic expansion). Thus, polynomial SFA can extract the same features from $\mathbf{s}'$ or $\mathbf{y}'$. However, other expansions do not have this property. One example of them is the 0.8Exp expansion function (68), $0.8\text{Exp}(x_1, x_2, \ldots, x_I) \stackrel{\text{def}}{=} (x_1, x_2, \ldots, x_I, |x_1|^{0.8}, |x_2|^{0.8}, \ldots, |x_I|^{0.8})$. SFA is invariant to scalings of the input data if combined with 0.8Exp, but it is not invariant to their mixing, i.e., $\text{SFA}(0.8\text{Exp}(\mathbf{\Lambda}\mathbf{x})) \equiv \text{SFA}(0.8\text{Exp}(\mathbf{x}))$, where $\mathbf{\Lambda}$ is a diagonal matrix with diagonal elements $\lambda_i \neq 0$, but $\text{SFA}(0.8\text{Exp}(\mathbf{U}\mathbf{x})) \not\equiv \text{SFA}(0.8\text{Exp}(\mathbf{x}))$ in general. The 0.8Exp expansion has been motivated by a model where the slow features are noisy harmonics of increasing frequency of a hidden parameter and it should be applied to the slow features directly. Thus, mixing of the slow features would break the assumed model and might compromise slowness extraction in the next layers in practice.

Technically, feature mixing by QR scaling could be reverted in the next layer (e.g., by an additional application of linear SFA before the expansion), but such a step would add unnecessary complexity. For this reason, besides the QR scaling, a second scaling method is proposed: The *sensitivity based scaling*, which scales the slow features without mixing them, as follows.

$$\mathbf{y}' = \mathbf{\Lambda}\mathbf{s}' \,, \tag{145}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with diagonal elements $\lambda_j \stackrel{\text{def}}{=} ||\mathbf{M}_j||_2$ (the L_2-norm of the j-th column vector of $\mathbf{M}$). Therefore,

$$\mathbf{a}(t) \stackrel{(142,145)}{=} \mathbf{M}\mathbf{\Lambda}^{-1}\mathbf{y}'(t) + \mathbf{d} \,. \tag{146}$$

Clearly, the transformation (145) does not mix the slow features, it only scales them. From the two key reconstruction properties of PCA mentioned

above (adding noise of certain variance to either one or many features increases the variance of the reconstruction error by the same amount), the first one (noise on a single feature) is fulfilled, because the columns of $\mathbf{M}\mathbf{\Lambda}^{-1}$ have unit norm since $\mathbf{\Lambda}^{-1} = \mathrm{Diag}(1/\lambda_1, \ldots, 1/\lambda_J)$. The second property is not fulfilled, because $\mathbf{M}\mathbf{\Lambda}^{-1}$ is in general not orthogonal (in contrast to $\mathbf{Q}$).

At first glance, it seems like multi-view learning might be an alternative for the scaling methods used here. However, multi-view learning actually solves a different problem. Moreover, it would mix the slow and reconstructive parts and might be too expensive computationally. For instance, the system of Xia et al. (2010) has cubic complexity w.r.t. N, whereas QR scaling and sensitivity based scaling have linear complexity w.r.t. N.

5.4.5 Input Reconstruction for iSFA

One interesting property of iSFA is that the features are nonlinear w.r.t. the input data, both the slow and the reconstructive part. The slow part is nonlinear due to the expansion function. The residual data is nonlinear because it is computed using the (nonlinear) slow part and the centered data. The reconstructive part is computed using the residual data and is linear w.r.t. the residual data but nonlinear w.r.t. the input data.

Even though the computed features are all nonlinear, iSFA allows a linear approximation of the input (linear input reconstruction). In contrast, standard SFA does not have a standard input reconstruction method, although various gradient-descent and vector-quantization methods have been tried (e.g., Wilbert, 2012) with limited success.

The reconstruction algorithm is simple: $\mathbf{a}$ (the contribution of the slow part to the centered data) is approximated as $\tilde{\mathbf{a}} = \mathbf{Q}\mathbf{y}' + \mathbf{d}$. Then, $\mathbf{b}$ (the residual vector) is approximated as $\tilde{\mathbf{b}} = \mathbf{W}_{\mathrm{PCA}}\mathbf{c}$. The reconstructed sample is then $\tilde{\mathbf{x}} = \tilde{\mathbf{a}} + \tilde{\mathbf{b}} + \bar{\mathbf{x}}$. See Algorithm 3 for details.

Algorithm 3 Linear input reconstruction for iSFA

Require: D, $\bar{\mathbf{x}}$, $\mathbf{W}_{\mathrm{PCA}}$, J, $\mathbf{Q}$, $\mathbf{b}$

procedure LINEAR-RECONSTRUCTION($\mathbf{y}$)
  $\mathbf{y}' \leftarrow (y_1, \ldots, y_J)$ ▷ Slow part
  $\mathbf{c} \leftarrow (y_{J+1}, \ldots, y_D)$ ▷ Reconstructive part
  $\tilde{\mathbf{x}} \leftarrow (\mathbf{Q}\mathbf{y}' + \mathbf{d}) + \mathbf{W}_{\mathrm{PCA}}\mathbf{c} + \bar{\mathbf{x}}$ ▷ $\tilde{\mathbf{a}} + \tilde{\mathbf{b}} + \bar{\mathbf{x}}$
  return $\tilde{\mathbf{x}}$
end procedure

The linear reconstruction algorithm has interesting properties: It is shorter than the feature extraction algorithm, the nonlinear expansion $\mathbf{h}$ and $\mathbf{W}_{\mathrm{SFA}}$ are not used, and it has lower computational complexity, because it consists of only two matrix-vector multiplications and three vector additions, none of them using expanded I'-dimensional data.

Linear reconstruction for iSFA is simple and effective. However, nonlinear

reconstruction is also possible and can be implemented as follows: Assume $\mathbf{y}$ is the iSFA feature representation of a sample $\mathbf{x}$. This is denoted by $\mathbf{y} = \text{iSFA}(\mathbf{x})$. Since $\mathbf{x}$ is unknown, the reconstruction error cannot be computed directly. However, one can indirectly measure the accuracy of a particular reconstruction $\tilde{\mathbf{x}}$ by means of the feature error, which is defined here as $e_{\text{feat}} \stackrel{\text{def}}{=} ||\mathbf{y} - \text{iSFA}(\tilde{\mathbf{x}})||$. The feature error can be minimized for $\tilde{\mathbf{x}}$ using the function $\text{iSFA}(\cdot)$ as a black box and gradient descent or other generic nonlinear minimization algorithms. Frequently, such algorithms require a first approximation, which can be very conveniently provided by the linear reconstruction algorithm.

Although nonlinear reconstruction methods might result in higher reconstruction accuracy than linear methods, they are typically more expensive computationally. Moreover, in Section 5.6.4 it is explained why minimizing e_{feat} does not necessarily improve the reconstruction error unless additional aspects of the training data are considered.

5.4.6 Some Remarks on iSFA, iGSFA, and HiGSFA

Clearly, the computational complexity of iSFA is at least that of SFA, because iSFA consists of SFA and a few additional computations. However, none of the additional computations is done on the expanded I'-dimensional data but at most on I or D-dimensional data (e.g., PCA is applied to I-dimensional data, and the QR decomposition is applied to an $I \times I$-matrix. These operations have an $\mathcal{O}(NI^2 + I^3)$ and $\mathcal{O}(I^3)$ complexity, respectively). Therefore, iSFA is slightly slower than SFA but has the same complexity order. Practical experiments (Section 5.5) confirm this observation.

The presentation above focuses on iSFA. To obtain information-preserving GSFA (iGSFA) one only needs to substitute SFA by GSFA inside the iSFA algorithm and provide GSFA with the corresponding training graph during the training phase. Notice that GSFA features have weighted zero mean instead of the simple (unweighted) zero mean enforced by SFA. This difference has already been compensated by the vector $\mathbf{d}$. HiGSFA is constructed simply by connecting iGSFA nodes in a hierarchy, just as HGSFA is constructed by connecting GSFA nodes.

5.5 Experimental Evaluation of HiGSFA

In this section, HiGSFA is evaluated using the problem of age estimation from human face photographs. HiGSFA is employed instead of HiSFA to be able to explicitly use the labels to boost estimation accuracy. However, due to the close connection between these algorithms, many aspects of the evaluation also extend to HiSFA.

This section is structured as follows. First, the topic of age estimation is introduced. Then, the image pre-processing method is described. Afterwards, the training graph used to learn age, race, and gender simultaneously is presented. Finally, an HiGSFA network is described and evaluated according to

three criteria: feature slowness (compared with HGSFA), age estimation error (compared with state-of-the-art algorithms), and linear reconstruction error (compared with PCA).

5.5.1 Age Estimation and Previous Work on this Problem

The study of age estimation from photographs is relatively recent and has useful applications for human-computer interaction, group-targeted advertisement, demographics, face recognition, control of age-related policies, and security. However, age estimation is a challenging task, because different persons experience facial aging differently depending on factors such as their gender, race, nutrition, health, exposure to the weather, use of cosmetics, and operations.

Two types of age have been distinguished. The real (ground-truth) age, which is the chronological age of the person, and the apparent age, which is the age conveyed solely by the information present in the image. Clearly, an algorithm cannot determine the real age exactly, at most it might determine the apparent age.

Different methods for age estimation have been proposed. For a more comprehensive literature review see the work of Ramanathan et al. (2009) and Fu et al. (2010). Geng et al. (2007) have proposed aging pattern subspace (AGES), which is based on temporal sequences of images of individual persons, the so-called aging patterns. The images are represented using an appearance model that combines geometric and texture information. In their system, a subspace is constructed for each aging pattern. Given a new image, the subspace providing the best possible reconstruction is found. Then, the position of the image within the aging pattern is determined.

Guo et al. (2009b) have proposed the use of bio-inspired features (BIF). Their architecture consists of two layers, in which the units of the first layer compute Gabor functions inspired by simple cells, whereas the units of the second layer compute a standard-deviation operation inspired by complex cells. Then, PCA is applied to reduce the dimensionality of the data to fewer than 1,000 features. Finally, an SVM or SVR provides the final age estimate.

The first SFA architecture for age estimation is a four-layer HSFA network that has been applied to raw images without prior feature extraction (Escalante-B. and Wiskott, 2010). The input images are synthetic and created using special software for 3D-face modeling. However, the complexity of the face model was probably too simple, which allowed linear SFA (in fact linear GSFA) to achieve good performance, and left open the question of whether SFA/GSFA could also be successful on real photographs.

Race and gender are two factors that influence the accuracy of age estimation (e.g., Guo et al., 2009a; Luu et al., 2009). This idea is exploited by the system of Guo and Mu (2010), where the faces are first classified according to race and gender, and age is then estimated in the particular race/gender group. Other algorithms allow the estimation of age, race, and gender simultaneously. Guo and Mu (2011) proposed the use of kernel partial least squares regression (KPLS)

on top of the BIF features.

More recently, various methods based on canonical correlation analysis (CCA) have been proposed by Guo and Mu (2014), particularly regularized kernel CCA (rKCCA), on top of BIF features in a framework for the joint estimation of age, race, and gender, providing good accuracy.

The use of a multi-scale convolutional neural network (MCNN) trained on images decomposed as 23 48×48-pixel image patches has been proposed by Yi et al. (2015). Each patch has one out of four different scales and is centered on a particular facial landmark. This method represents the state-of-the-art accuracy before HiGSFA (see Table 5.4).

5.5.2 Image Database and Image Pre-Processing

The MORPH-II database (i.e. MORPH, Album 2) (Ricanek Jr. and Tesafaye, 2006) is a large database available for a symbolic fee and suitable for age estimation. It contains 55,134 images of about 13,000 different persons with ages ranging from 16 to 77 years. The images were taken under partially controlled conditions (e.g. frontal pose, absence of glasses, good image quality, absence of strong shadows), and include variations in head pose (e.g. tilt angle) and expression. The database includes annotations stating the age of the persons, their gender (M or F), "race": "black" (B), "white" (W), "asian" (A), "hispanic" (H), and "other" (O), and the coordinates of the eyes. The procedure used to assign the race label does not seem to be documented by the database. Most of the images are of black (77%) or white races (19%), making it probably more difficult to generalize to other races, such as asian. This database has been chosen for this research because of its large number of images.

The evaluation method used by Guo and Mu (2014) and many other works is also adopted here. In this method, the input images are partitioned in 3 disjoint sets S_1 and S_2 of 10,530 images, and S_3 of 34,074 images. The racial and gender composition of S_1 and S_2 is the same: they have about 3 times more images of males than females and the same number of white and black people. Other races are omitted. More exactly, $|MB| = |MW| = 3980$, $|FB| = |FW| = 1285$. The remaining images constitute the set S_3, which is composed as follows: $|MB| = 28\,872$, $|FB| = 3187$, $|MW| = 1$, $|FW| = 28$, $|MA| = 141$, $|MH| = 1667$, $|MO| = 44$, $|FA| = 13$, $|FH| = 102$ and $|FO| = 19$. Training and testing are done twice, using either S_1 and S_1-test $\stackrel{\text{def}}{=} S_2 + S_3$ or S_2 and S_2-test $\stackrel{\text{def}}{=} S_1 + S_3$.

The input images are pre-processed in two steps: pose normalization and face sampling (Figure 5.2). The pose-normalization step fixes the position of the eyes ensuring that: (a) the eye line is horizontal, (b) the inter-eye distance is constant, and (c) the output resolution is 256×260 pixels. After pose normalization, the face sampling step selects the head area only, enhances the contrast, and scales down the image to 96×96 pixels. Typically the chin, forehead, and some hair are visible in the resulting images.

On top of the S_1, S_2, and S_3 datasets, three extended datasets (DR, S, and T) are defined in this work: A DR-dataset is used to train HiGSFA (or the DR algorithm), an S-dataset is used to train the supervised step on top of HiGSFA (a Gaussian classifier), and a T-dataset is used for testing. The DR and S-datasets are created using the same training images (either S_1 or S_2), and the T-dataset with the corresponding test images, either S_1-test or S_2-test.

The images of the DR and S-datasets go through a distortion step during face sampling, which includes a small random translation of max ± 1.4 pixels, a rotation of max ± 2 degrees, a rescaling of $\pm 4\%$, and small fluctuations in the average color and contrast. The exact transformations are random and uniformly distributed in their corresponding intervals. Although such small distortions are frequently imperceptible, they teach HiGSFA to become invariant to small errors during image normalization and are necessary due to its feature specificity to improve generalization to test data. Other algorithms that use pre-computed features, such as BIF, or particular structures (e.g., convolutional layers, max pooling) are mostly invariant to such small transformations by construction (e.g., Guo and Mu, 2014).

Distortions allow us to increase the number of images used for training. The images of the DR-dataset are distorted 22 times, each time using a different random distortion, and those of the S-dataset 3 times, resulting in 231,660 and 31,590 images, respectively. The images of the T-dataset are not distorted and used only once.

5.5.3 Efficient Training Graphs for Learning Multiple-Labels

A main factor for the appeal of GSFA is its efficient pre-defined training graphs. Predefined graphs include a clustered graph (classification) and a serial graph (regression), both having a training complexity of $\mathcal{O}(NI^2 + I^3)$. However, up to now efficient graphs have only been defined for a single (categorical or numerical) label. In Section 4.5.5, a method for the combination of graphs has been proposed. This idea is used here to propose an efficient graph that encodes three labels and is based on three pre-defined graphs: two clustered graphs (for gender and race classification) and a serial graph for age estimation.

Clustered graphs for gender and race. The graph used for gender classification is a clustered graph (see Section 3.3) that has only two classes (male/female) of N_M and N_F samples, respectively. For the actual experiments, $N_F = 2570r$ and $N_M = 7960r$, where $r \stackrel{\text{def}}{=} 22$ is a multiplicity factor.

The graph used for race classification is similar to the graph above. Only two classes are considered by the graph (B and W), and the number of samples per class is $N_B = N_W = 5265r$.

Serial graph for age estimation. The serial graph for age estimation (see Section 3.4) is described in Figure 5.6.

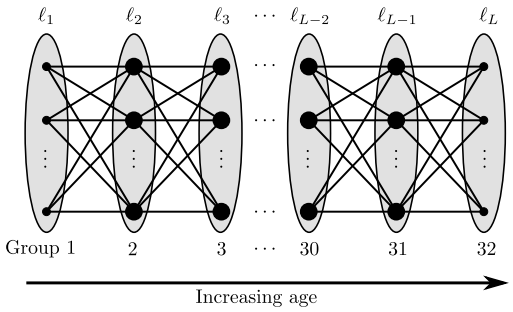


Figure 5.6: Illustration of a serial training graph for age estimation. The training images are first ordered by increasing age and then grouped into $L = 32$ groups of $N_g = 7238$ samples each. Each dot represents an image, edges represent connections, and ovals represent the groups. The images of the first and last group have weight 1 and the remaining images have weight 2 (image weights represented by smaller/bigger dots). All edges have a weight of 1.

Efficient graph for age, race, and gender estimation The method proposed in Section 4.5.5 (third extension) for the combination of graphs is used here for the first time to combine three pre-defined graphs. Such a method guarantees that the slowest optimal free responses of the combined graph span the slowest optimal free responses of the original graphs.

The method provides a compact feature representation. For example, one can combine a clustered graph for gender (M or F) estimation and another for race (B or W). The first two features of the resulting graph are then enough for gender *and* race classification. Alternatively, one could create a clustered graph with four classes (MB, MW, FB, FW), so that 3 features would be needed for classification instead of 2. However, such a representation would be impractical for larger numbers of classes. For example, if the original numbers of classes were $C_1 = 10$ and $C_2 = 12$, one would need to extract $C_1C_2 - 1 = 119$ features, whereas in the proposed graph combination, one would only need to preserve $(C_1 - 1) + (C_2 - 1) = 20$ features.

To learn age, race, and gender labels, a graph G_3 is proposed by combining a serial graph for age estimation, a clustered graph for gender, and a clustered graph for race. The serial graph has vertex weights not quite the same as in the clustered graphs, but this might not affect the accuracy of the combined graph significantly. For comparison purposes, a serial graph G_1 that only learns age is also used.

The first 4 to 7 features extracted from the S-dataset are used to train three separate Gaussian classifiers (GC), one for each label. For race only two classes are considered (B and W), and for gender only M and F. For age, the images are ordered by increasing age and partitioned in 39 classes of the same size. This hyper-parameter has been tuned independently of the number of groups in the graph, which is 32. The classes have average ages of $\{16.6, 17.6, 18.4, \ldots, 52.8,$

57.8} years. To compute these average ages, as well as to order the samples by age in the serial graph, the age of the persons is considered with a day resolution (e.g., an age may be expressed as 25.216 years). However, for the evaluation, integer ground-truth labels and integer estimates are used.

After the GC has been trained, the final age estimation (on the T-dataset) is computed using the soft GC method given by Equation (67) that exploits class membership probabilities of the GC. However, an additional step keeps the integer part of the age estimation. Equation (67) has been designed to minimize the root mean squared error (RMSE). Although it incurs in an error due to the discretization of the labels, the soft nature of the estimation has provided good accuracy and robustness to misclassifications.

5.5.4 Evaluated Algorithms

Besides HiGSFA, other algorithms are included in the evaluation for comparison purposes: HGSFA, PCA, and state-of-the-art age-estimation algorithms. iGSFA and GSFA are not used directly but indirectly in HiGSFA and HGSFA in order to take advantage of the benefits of hierarchical processing.

The structure of the HiGSFA and HGSFA networks is described in Table 5.1. In both networks, the nodes are simply an instance of iGSFA or GSFA preceded by different linear or nonlinear expansion functions, except in the first layer, where PCA is applied to the pixel data to preserve 20 out of 36 principal components prior to the expansion. The method used to scale the slow features is the sensitivity method of Section 5.4.4. The hyper-parameters have been hand-tuned to achieve best accuracy on age estimation using educated guesses, random sets S_1, S_2 and S_3 different to those used for the evaluation, and fewer image multiplicities to speed up the process.

The proposed HGSFA/HiGSFA networks are different in several aspects from SFA networks used in the literature (e.g., Franzius et al., 2007). For example, to improve feature specificity at the lowest layers, no weight sharing is used. Moreover, the input to the nodes (fan-in) originates mostly from the output of 3 nodes in the preceding layer (3×1 or 1×3). Such a small fan-in reduces the computational cost because the input dimensionality is minimized. In this case it results in networks with 10 layers, potentiating the accumulation of nonlinearities across the network.

The employed expansion functions are a mixture of different nonlinear functions on subsets of the input vectors and include: (1) The identity function $\mathrm{I}(\mathbf{x}) = \mathbf{x}$. (2) Quadratic terms $\mathrm{QT}(\mathbf{x}) \stackrel{\text{def}}{=} \{x_i x_j\}_{i,j=1}^{N}$. (3) A normalized version of QT: $\mathrm{QN}(\mathbf{x}) \stackrel{\text{def}}{=} \{\frac{1}{1+||\mathbf{x}||^2} x_i x_j\}_{i,j=1}^{N}$. (4) The terms $0.8\mathrm{ET}(\mathbf{x}) \stackrel{\text{def}}{=} \{|x_i|^{0.8}\}_{i=1}^{N}$ of the 0.8Exp expansion, which is useful to improve generalization and resistance against outliers (Escalante-B. and Wiskott, 2011). (5) The function $\textsc{max2}(\mathbf{x}) \stackrel{\text{def}}{=} \{\max(x_i, x_{i+1})\}_{i=1}^{N-1}$. The MAX2 function is proposed here motivated by an state-of-the-art algorithm for age estimation (Yi et al., 2015) that includes max pooling or a variant of it. As a concrete example of the nonlin-

layer	size	node fan-in	stride	output dim. per node	
				HGSFA	HiGSFA
0	96×96 pixels	—	—	—	—
1	31×31 nodes	6×6	3×3	14	18
2	15×31 nodes	3×1	2×1	20	27
3	15×15 nodes	1×3	1×2	27	37
4	7×15 nodes	3×1	2×1	49	66
5	7×7 nodes	1×3	1×2	60	79
6	3×7 nodes	3×1	2×1	61	88
7	3×3 nodes	1×3	1×2	65	88
8	1×3 nodes	3×1	1×1	65	93
9	1×1 nodes	1×3	—	66	95
10	1×1 nodes	1×1	—	75	75

Table 5.1: Description of the HiGSFA and HGSFA networks. Both networks have the same number of nodes and general structure, but they differ in the type of nodes and in the number of features preserved by them. Layer 0 denotes the input image, whereas layer 10 is the top node.

ear expansions employed by the HiGSFA network, the expansion of the first layer is $\mathrm{I}(x_1, \ldots, x_{18}) \,|\, 0.8\mathrm{ET}(x_1, \ldots, x_{15}) \,|\, \textsc{max}2(x_1, \ldots, x_{17}) \,|\, \mathrm{QT}(x_1, \ldots, x_{10})$, where | indicates vector concatenation. The details of the expansions used in the remaining layers are available upon request.

The parameter Δ_T of layers 3 to 10 is set to 1.96. Δ_T is not used in layers 1 and 2, and instead the number of slow features is fixed to 3 and 4, resp. The number of features given to the supervised algorithm, shown in Table 5.2, has been tuned for each DR algorithm and supervised problem.

Algorithm	Age	Race	Gender
HiGSFA (G_3)	5	6	4
HGSFA (G_3)	5	5	7
PCA	54	54	60

Table 5.2: Number of output features given to the supervised step (a Gaussian classifier).

Since the data dimensionality allows it, PCA is used directly (it was not resorted to hierarchical PCA) to provide more accurate principal components and smaller reconstruction errors.

5.5.5 Experimental Results

The results of HiGSFA, HGSFA and PCA on three evaluation criteria are presented now. Individual scores are reported as $a \pm b$, where a is the average over the test images (S_1-test and S_2-test), and b is the standard error of the mean (i.e., half the absolute difference).

Feature Slowness

The weighted Δ values of GSFA (Equation 77) are denoted here as $\Delta_j^{\mathrm{DR},G_3}$ and depend on the graph G_3, which in turn depends on the training data and the labels. To measure slowness (or rather fastness) of test data, standard Δ values are computed using the images ordered by increasing age label, $\Delta_j^{\mathrm{T,lin}} \stackrel{\mathrm{def}}{=} \frac{1}{N-1}\sum_n (y_j(n+1) - y_j(n))^2$. The last expression is equivalent to a weighted Δ value using a linear graph (Figure 3.3.b). In all cases, the features are normalized to unit variance before computing their Δ values to allow for fair comparisons in spite of the feature scaling method.

Table 5.3 shows $\Delta_{1,2,3}^{\mathrm{DR},G_3}$ (resp. $\Delta_{1,2,3}^{\mathrm{T,lin}}$), that is, the Δ values of the three slowest features extracted from the DR-dataset (resp. T-dataset) using the graph G_3 (resp. a linear graph).

	PCA	HGSFA (G_3)	HiGSFA (G_3)
$\Delta_1^{\mathrm{DR},G_3}$	—	1.23	1.17
$\Delta_2^{\mathrm{DR},G_3}$	—	1.46	1.38
$\Delta_3^{\mathrm{DR},G_3}$	—	1.56	1.53
$\Delta_1^{\mathrm{T,lin}}$	1.99	0.45	0.38
$\Delta_2^{\mathrm{T,lin}}$	1.93	1.12	0.99
$\Delta_3^{\mathrm{T,lin}}$	1.99	1.90	1.90

Table 5.3: Average delta values of the first three features extracted by PCA, HGSFA, and HiGSFA on either training or test data (smaller values are better). The first feature extracted is the most stable according to the age-ordered linear graph, indicating that this is the main feature that encodes age. For comparison, the Δ value of unit-variance i.i.d. noise is 2.0.

Table 5.3 shows that HiGSFA outperforms HGSFA in slowness maximization. The $\Delta^{\mathrm{T,lin}}$ values of the PCA features are larger, which is not surprising, because PCA does not optimize for slowness. Since Δ^{DR,G_3} and $\Delta^{\mathrm{T,lin}}$ are computed from different graphs, they should not be compared with each other. $\Delta^{\mathrm{T,lin}}$ considers transitions between images with the same or very similar ages but arbitrary race and gender. Δ^{DR,G_3} only considers transitions between images having at least one of a) the same gender, b) the same race, or c) different but consecutive age groups.

Age Estimation Error

Some real-life applications only need a coarse categorization of age in broad age groups. However, other applications benefit from a more precise estimation, making it convenient to treat age estimation as a regression problem requiring a concrete numerical estimation, usually expressed as an integer number of years.

Three metrics are used to measure age estimation accuracy: (1) the mean absolute error (MAE) (see Geng et al., 2007), which is the most frequent metric

for age estimation, (2) the root mean squared error (RMSE), which is common in Machine Learning as a loss function for regression, and (3) cumulative scores (CS) (see Geng et al., 2007), which indicate the fraction of the images that have an estimation error below a given threshold. For instance, $CS(5)$ is the fraction of estimates (e.g., expressed as a percentage) having an error of at most 5 years w.r.t. the real age. The CSs at various thresholds are provided, facilitating future comparisons with other methods. Although the RMSE is sensitive to outliers and has almost not been used in the literature on age estimation, some applications might benefit from its stronger penalization of larger estimation errors.

Algorithm	Age (MAE)	Age (RMSE)	Race (CR)	Gender (CR)
BIF+3Step (Guo and Mu, 2010)	4.45 ± 0.01	—	$98.80\% \pm 0.04$	$97.84\% \pm 0.16$
BIF+KPLS (Guo and Mu, 2011)	4.18 ± 0.03	—	$98.85\% \pm 0.05$	$98.20\% \pm 0.00$
BIF+rKCCA (Guo and Mu, 2014)	3.98 ± 0.03	—	$99.00\% \pm 0.00$	$\mathbf{98.45\% \pm 0.05}$
BIF+rKCCA+SVM (Guo and Mu, 2014)	3.92 ± 0.02	—	—	—
baseline CNN (Yi et al., 2015)	4.60 ± 0.05	—	—	—
MCNN* no align (Yi et al., 2015)	3.79 ± 0.09	—	—	—
MCNN* only age (Yi et al., 2015)	3.63 ± 0.00	—	—	—
MCNN* (Yi et al., 2015)	3.63 ± 0.09	—	$98.6\% \pm 0.05$	$97.9\% \pm 0.1$
PCA (control)	6.804 ± 0.007	8.888 ± 0.000	$96.75\% \pm 0.06$	$91.54\% \pm 0.24$
HGSFA (G_3) (control)	3.921 ± 0.018	5.148 ± 0.049	$98.60\% \pm 0.08$	$96.40\% \pm 0.12$
HiGSFA (G_1) (control)	3.605 ± 0.001	4.690 ± 0.000	—	—
HiGSFA (G_3) (proposed)	$\mathbf{3.497 \pm 0.008}$	$\mathbf{4.583 \pm 0.000}$	$\mathbf{99.15\% \pm 0.01}$	$97.70\% \pm 0.01$
Chance level	9.33	10.95	87.58%	86.73%

Table 5.4: Accuracy in years of state-of-the-art algorithms for age estimation on the MORPH-II database (test data). Classification rates (CR) for race and gender estimation are also provided. The chance level is the best possible performance when the estimation is constant. *A mistake in the evaluation protocol of MCNN (Yi et al., 2015) made their training and test data not disjoint, thus the actual accuracy might differ, see `http://www.cbsr.ia.ac.cn/users/dyi/agr.html`.

The accuracies are summarized in Table 5.4. The MAE of HGSFA is 3.921 years, which is better than that of BIF+3Step, BIF+KPLS and BIF+rKCCA, similar to BIF+rKCCA+SVM, and worse than the MCNNs (except the baseline). However, the MAE of HiGSFA is only 3.497 years, which seems to be better than all previous algorithms found in the literature. In contrast, PCA has the largest MAE, namely an MAE of 6.804 years. Detailed cumulative scores for HiGSFA and HGSFA are provided in Table 5.5.

The RMSE of HGSFA on test data is 5.148 years, HiGSFA yields an RMSE of

4.583 years, and PCA an RMSE of 8.888 years. The RMSE of other approaches does not seem to be available.

The poor accuracy of PCA for age estimation is not surprising, because principal components might lose wrinkles, skin imperfections, and other information that could reveal age in adults. Another reason is that principal components are too unstructured to be properly untangled by the soft GC method, in contrast to slow features, which have a very specific and simple structure.

Algorithm	cs(0)	cs(1)	cs(2)	cs(3)	cs(4)	cs(5)	cs(6)	cs(7)	cs(8)	cs(9)	cs(10)
HGSFA	8.80	26.42	42.41	55.80	66.38	74.86	81.31	86.37	90.12	92.93	94.96
HiGSFA	9.87	29.16	46.23	60.14	71.04	79.56	85.70	90.04	93.12	95.45	96.92

	cs(11)	cs(12)	cs(14)	cs(16)	cs(18)	cs(20)	cs(22)	cs(24)	cs(26)	cs(28)	cs(30)
HGSFA	96.43	97.42	98.67	99.34	99.69	99.85	99.92	99.97	99.97	99.98	99.99
HiGSFA	97.91	98.56	99.34	99.70	99.84	99.91	99.94	99.96	99.97	99.98	99.99

Table 5.5: Percentual cumulative scores (the larger the better) for various maximum allowed errors ranging from 0 to 30 years.

The behavior of the estimation errors of HiGSFA is plotted in Figure 5.7 as a function of the real age. On average, older persons are estimated much younger than they really are. This is in part due to the small number of older persons in the database, and because the oldest class used in the supervised step has an average of about 58 years, making this the largest age that can be estimated by the system. The MAE is surprisingly low for persons below 45 years. The most accurate estimation is an MAE of only 2.253 years for 19-year-old persons.

Reconstruction Error

A reconstruction error is a measure of how much information the output features retain from the original input. In order to compute it, a linear global model for input reconstruction is assumed here.

Let $\mathbf{X}$ be the input data and $\mathbf{Y}$ the corresponding set of extracted features. A matrix $\mathbf{D}$ and a vector $\mathbf{c}$ are learned from the DR-dataset using linear regression such that $\hat{\mathbf{X}} \stackrel{\text{def}}{=} \mathbf{D}\mathbf{Y} + \mathbf{c}\mathbf{1}^T$ approximates $\mathbf{X}$ as closely as possible, where $\mathbf{1}$ is a vector of N ones. Thus, $\hat{\mathbf{X}}$ is a matrix containing the reconstructed samples (i.e. $\hat{\mathbf{x}}_n \stackrel{\text{def}}{=} \mathbf{D}\mathbf{y}_n + \mathbf{c}$ is the reconstruction of the input $\mathbf{x}_n$ given its feature representation $\mathbf{y}_n$). Figure 5.2 shows examples of face reconstructions using features extracted by different algorithms.

Since the model is linear and global, output features are mapped to the input domain linearly using ordinary least squares. For PCA this gives the same result as the usual multiplication with the transposed projection matrix plus image average. An alternative (local) approach for HiGSFA would be to use the linear reconstruction algorithm of each node to perform reconstruction from the top of the network to the bottom, one node at a time. However, such a local approach is less accurate than the global one.

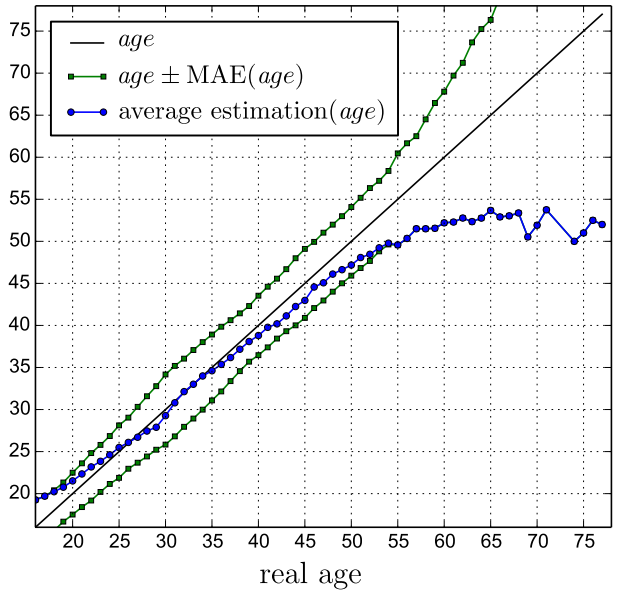


Figure 5.7: The average age estimates of HiGSFA are plotted as a function of the real age. The MAE is also computed as a function of the real age and plotted as $age \pm \mathrm{MAE}(age)$.

The normalized reconstruction error, computed from the T-dataset, is then defined as

$$e_{\mathrm{rec}} \stackrel{\mathrm{def}}{=} \frac{\sum_{n=1}^{N} ||(\mathbf{x_n} - \hat{\mathbf{x}}_\mathbf{n})||^2}{\sum_{n=1}^{N} ||(\mathbf{x_n} - \bar{\mathbf{x}})||^2}\,, \tag{147}$$

which is the ratio between the energy of the reconstruction error and the variance of the test data except for a factor $N/(N-1)$.

	Chance level	HGSFA	HiGSFA	PCA
e_{rec}	1.0	0.818	0.338	**0.201**

Table 5.6: Reconstruction errors on test data using 75 features and various algorithms. The original dimensionality is $96^2 = 9216$ components.

The reconstruction errors of HGSFA, HiGSFA and PCA using 75 features are given in Table 5.6. The largest reconstruction error results from the constant reconstruction $\bar{\mathbf{x}}$ (chance level). As expected, HGSFA does slightly better than chance level, but worse than HiGSFA, which is closer to PCA. PCA yields the best possible features for the given linear global reconstruction method, and is better than HiGSFA by 0.127. For HiGSFA, from the 75 output features, 8 of them are slow features (slow part), and the remaining 67 are reconstructive. If one uses 67 features instead of 75, PCA yields a reconstruction error of 0.211.

HiGSFA Network with HGSFA Hyper-Parameters

In the experiments above, the hyper-parameters of the HiGSFA and HGSFA networks (e.g., nonlinear expansion functions, output dimensionalities) have been tuned separately. To verify that the performance of HiGSFA is better than that of HGSFA not simply due to different hyper-parameters, the performance of an HiGSFA network is evaluated using the hyper-parameters of the HGSFA network (the only difference is the use of iGSFA nodes instead of GSFA nodes). The hyper-parameter Δ_T, not present in HGSFA, is set as in the tuned HiGSFA network. As expected, the change of hyper-parameters affected the performance of HiGSFA: The MAE increased to 3.72 years, and the RMSE increased to 4.80 years. The reconstruction error improved slightly to 0.322. Although the suboptimal hyper-parameters increased the estimation errors of HiGSFA, it was still clearly superior to HGSFA.

Sensitivity to the Delta Threshold Δ_T

The influence of Δ_T on estimation accuracy and numerical stability is evaluated by testing different values of Δ_T. For simplicity, the same Δ_T is used from layers 3 to 10 in this experiment (Δ_T is not used in layers 1 and 2, where the number of features in the slow part is constant and equal to 3 and 4 features, respectively). The performance of the algorithm as a function of Δ_T is shown in Table 5.7. The Δ_T yielding minimum MAE and used in the optimized architecture is 1.96.

Δ_T	Age MAE	Age RMSE	e_{rec}	Race CR	Gender CR	#Features L3
1.92	3.506	4.583	0.330	99.15	97.58	2.87
1.94	3.499	4.583	0.333	99.15	97.64	3.22
1.96	3.497	4.583	0.338	99.15	97.00	3.66
1.98	3.530	4.637	0.359	99.09	94.72	4.14

Table 5.7: Performance of HiGSFA on the MORPH-II database using different Δ_T (default value is $\Delta_T = 1.96$). The reported results are the age estimation errors (MAE and RMSE), the reconstruction error (e_{rec}), the percentual classification rate for race and gender, and the average length of the slow part in the features computed by the nodes of the third HiGSFA layer. All error measures have been computed on test data.

The average number of slow features in the third layer changes moderately depending on the value of Δ_T, ranging from 2.87 to 4.14 features, and the final error measures change only slightly. This shows that the parameter Δ_T is not critical and can be tuned easily.

Evaluation on the FG-NET Database

The FG-NET database (Cootes, 2004) is a small database with 1002 facial images taken under uncontrolled conditions (e.g., many are not frontal) and includes identity, race, and gender annotations. Due to its small size, it is unsuitable to evaluate HiGSFA directly. However, FG-NET is used here to investigate the capability of HiGSFA to generalize to a different *test* database. The HiGSFA (G_3) network that has been trained with images of the MORPH-II database (either with the set S_1 or S_2) is tested using images of the FG-NET database. For this experiment, images outside the original age range from 16 to 77 years are excluded.

For age estimation, the MAE is 7.32 $\pm$ 0.08 years and the RMSE is 9.51 $\pm$ 0.13 years (using 4 features for the supervised step). For gender and race estimation, the classification rates (5 features) are 80.85% $\pm$ 0.95% and 89.24% $\pm$ 1.06%, resp. The database does not include race annotations, but all inspected subjects appear to be closer to white than to black. Thus, it is assumed that all test persons have white race.

The most comparable cross-database experiment seems to be done using a system (Ni et al., 2011) trained on a large database of images from the internet and tested on FG-Net. By restricting the ages to the same 16–77 year range used above, this system achieved an MAE of approximately 8.29 years.

Alternative Evaluation Protocol

The protocol by Guo and Mu (2014) that has been used to create the training and test data used in the experiments (based on the sets S_1, S_2 and S_3) has been frequently used before for age estimation. However, it has a few disadvantages. Thus, the "leave one person out" (LOPO) (Choi et al., 2011; Huerta et al., 2015) protocol is also adopted here. For efficiency reasons, 2000 persons are left out of the training set instead of one. That is, the test data is created with the images of 2000 persons chosen at random (about 8000 images), whereas the training data is simply the remaining images (about 45 000 images).

The alternative protocol has the following properties in contrast to the one proposed by Guo and Mu (2014): (a) The distribution of the training and test images is the same on average: $|M| = 85\%$, $|F| = 15\%$, $|B| = 77\%$, $|W| = 19\%$, $|H| = 3\%$, $|A| < 1\%$, $|O| < 1\%$. This is a basic assumption in machine learning. In contrast, the original protocol has: $|M| = 76\%$, $|F| = 24\%$, $|B| = 50\%$, $|W| = 50\%$, $|O| = |A| = |H| = 0\%$ for training, and $|M| = 87\%$, $|F| = 13\%$, $|B| = 84\%$, $|W| = 12\%$, $|H| = 4\%$, $|A| < 1\%$, and $|O| < 1\%$ for testing. (b) The number of training images available is 4 times larger (about 45,000 vs 10,530 images), which might improve generalization. (c) Since realistic applications might involve age estimations from *unknown* persons, the test images are restricted to persons *not* appearing in the training data. In the protocol of Guo and Mu (2014), 44% of the test images belong to persons previously seen in training. (d) One can improve evaluation accuracy by repeating the protocol several times (here

5 times). The original protocol is repeated twice.

Note that the alternative protocol is similar but not equivalent to 5-fold cross validation: Each execution of the protocol is independent, making different test sets not necessarily disjoint, and the ratio training data/test data is larger (5.63 vs. 4.0 in 5-fold cross validation).

The alternative protocol has more training images than the original protocol. To keep efficiency manageable, each image is distorted only 4 times to create the DR-dataset (instead of 22 times) and once to create the S-dataset (instead of 3 times). Notice that all races appear in the training and test data. Therefore, during training two classes are considered: B and a virtual class R $\stackrel{\text{def}}{=}$ W+A+H+O to preserve a binary clustered graph for race estimation and balance the size of the classes. However, for fair comparisons, the classification rate is computed using images of only the B and W races. The results are shown in Table 5.8. The top MAE achieved by HiGSFA is the same for both protocols when rounded to two digits: 3.50 years (though, using different networks and training graphs).

Algorithm	Age (MAE)	Age (RMSE)	Race (CR)	Gender (CR)
BIF+3SVM †‡ (Han et al., 2013)	4.2	—	—	—
CNN (5CV)*‡ (Huerta et al., 2015)	3.88	—	—	—
HiGSFA (G_3)	3.511 ± 0.002	4.626 ± 0.026	98.80% ± 0.01	98.53 % ± 0.02
HiGSFA (age only)	$\mathbf{3.502 \pm 0.003}$	$\mathbf{4.583 \pm 0.000}$	—	—
Chance level	9.16	10.78	87.53%	80.69%

Table 5.8: Accuracy of HiGSFA on MORPH-II using the alternative protocol. Some results are written in this table as $a \pm b$, where a is the average over 5 runs (test data) and b is the standard error of the mean. *5-fold cross validation. †Uses a larger version of the database but only 10 001 training images. ‡As in the alternative protocol, it has been ensured that the persons in the training and test data are disjoint.

5.6 Discussion of HiGSFA

This chapter has proposed an extension to HGSFA, called hierarchical information-preserving GSFA (HiGSFA), that complements the slowness principle with information preservation resulting in improved global slowness, input reconstruction and label estimation accuracy compared with HGSFA.

The advantages and limitations of HSFA (and HGSFA) networks have been analyzed, particularly the phenomena of unnecessary information loss and poor input reconstruction. Unnecessary information loss occurs when a node in the network prematurely discards information that would have been useful for slowness maximization in another node higher up in the hierarchy. Poor input

reconstruction refers to the difficulty of approximating an input accurately from its feature representation. It is shown that these phenomena are the result of optimizing slowness locally, yielding suboptimal global features.

HiGSFA improves feature extraction at the local level to address these shortcomings. The feature vectors computed by iGSFA nodes of an HiGSFA network are divided in two parts: a slow and a reconstructive part. The features of the slow part follow a slowness optimization goal and are slow features transformed by a linear scaling. The features of the reconstructive part follow the principle of information preservation (i.e. maximization of mutual information between outputs and labels), which is implemented in practice as the minimization of a reconstruction error. The parameter Δ_T (Δ-threshold) allows the combination of PCA and GSFA. This parameter balances the lengths of the slow and reconstructive parts, J and $D - J$ features, respectively, where D is the output dimensionality and J is the number of features with $\Delta < \Delta_T$.

A small Δ_T results in more reconstructive features and a large Δ_T results in more slow features. In particular, when $\Delta_T < 0$, iGSFA becomes equivalent to PCA, and when $\Delta_T \geq 4.0$, iGSFA becomes equivalent to GSFA except for a linear transformation (positive edge weights and a consistent graph are assumed). Theory justifies fixing Δ_T slightly smaller than 2.0 (Section 5.3.3), resulting in some features being similar to those of GSFA and other features being similar to those of PCA (on residual data).

The method proposed in Section 4.5.5 is used for the first time to combine various efficient pre-defined training graphs into a single efficient training graph, allowing efficient and accurate learning of multiple labels.

The experimental results show that HiGSFA is better than HGSFA in terms of feature slowness, input reconstruction and age estimation accuracy. Moreover, HiGSFA offers even higher accuracies than current state-of-the-art algorithms for age estimation, including approaches based on bio-inspired features and convolutional neural networks. The improvement over state-of-the-art multi-scale CNNs is a reduction by 48.5 days ($\approx$ 1.5 months) in the average estimation error. The improvement over HGSFA is larger: 154 days ($\approx$ 5 months). This is a significant improvement technically and conceptually.

The next sections provide additional insights—mostly conceptual—into the proposed approach, the obtained results, and future work.

5.6.1 The Approach

Information preservation can be guaranteed by preserving the information contained in the data that describes the global slow features. However, it has been shown that one cannot always identify such information at a local level. Therefore, HiGSFA resorts to a reconstruction goal to preserve as much information about the local input as possible, which is likely to also include information relevant to extract the global slow features.

The features extracted by HiGSFA are better than those of HGSFA quantitatively and qualitatively. Even if unlimited training data and computational

resources were available, the features extracted by HGSFA do not necessarily converge to those of HiGSFA. In this overfitting-free scenario, information loss would only decrease partially in HGSFA, because the main cause of this problem is not overfitting but the local optimization of slowness.

Besides the approach followed by HiGSFA, the slowness principle and information preservation may be combined by optimizing a single objective function that integrates both criteria, favoring directions that are slow and have a large variance. However, I have found in previous experiments that balancing these two criteria is difficult in practice. In addition, splitting the two allows to keep SFA nonlinear and PCA linear.

HiGSFA inherits from SFA a close connection to unsupervised learning. Like GSFA can be emulated with SFA (see Section 3.2.5), HiGSFA can be emulated by training HiSFA in an unsupervised fashion with data generated from a particular Markov chain. This emulation would incur in a small error due to PCA being unaware of sample weights, which could be easily fixed by using the vertex weights as weighting factors of the samples during the computation of the covariance matrix by PCA.

The idea of complementing GSFA with information preservation in hierarchical networks can also be applied to SFA (e.g., either using HiSFA or an HiGSFA network trained with a linear graph, where the samples are ordered by time). The improved feature slowness of HiSFA over HSFA might be useful to improve simulations based on SFA for Neuroscience. Moreover, existing neural models based on the slowness principle might benefit from incorporating information preservation.

The slow and reconstructive parts of the extracted features can be seen as two information channels: The first one encodes information representing the slow parameters, and the second one encodes information representing the remaining aspects of the input. Although the information in the slow part is somewhat mixed in the age/gender/race estimation experiments, it can be further decomposed into three channels; the 3rd slowest feature is mostly related to race, the 4th one to gender, and the remaining ones to age. Therefore, HiGSFA follows two of the suggestions by Krüger et al. (2013) based on findings from Neuroscience regarding the primate visual system useful for successful computer vision, namely, hierarchical processing and information-channel separation.

The iGSFA node has been implemented in Python (including iSFA, see Appendix A). In the next few months, I plan to make the node public by integrating it into the MDP toolkit (Zito et al., 2009).

5.6.2 Network Parameters

By selecting the network structure appropriately, the computational complexity of HiGSFA (and other hierarchical versions of SFA) is linear w.r.t. the number of samples and their dimensionality, resulting in feasible training times. Training a single HiGSFA network (231 660 images of 96×96 pixels) takes only 10 hours, whereas HiSFA takes about 6 hours (including the time needed for data loading,

the supervised step, training, and testing) on a single computer (24 Xeon X7542 cores @ 2.67GHz and 128 GB of RAM) without GPU computing. However, the algorithm can also be implemented on GPUs or using distributed processing, because the nodes within a layer can be trained independently. For comparison, the system of Guo and Mu (2014) takes 24.5 hours (training and testing) on a setup with fewer images and lower resolution.

HiGSFA is more accurate than HGSFA even when the output dimensionalities and network hyper-parameters have been tuned for the latter network. However, HiGSFA yields even higher accuracies if larger output dimensionalities are used than those providing best accuracies for HGSFA. This can be explained by various factors: (a) In both networks, the input dimensionality of a single GSFA node is I' (the expanded dimension), whereas in HiGSFA the input dimensionality of a single PCA node is I, where typically $I' \gg I$. Hence, slow features may overfit more than reconstructive features for this setup. (b) In HGSFA, the features of all the nodes are attracted to the same optimal free responses, whereas in HiGSFA the reconstructive part is attracted to the (different for each node) local principal components. Thus, in HGSFA overfitting might accumulate through the layers more than in HiGSFA. (c) The HGSFA network may benefit less from more input features, because faster features might be noise-like and result in more overfitting without providing much additional information.

In the iSFA and iGSFA algorithms, the number of slow features J with $\Delta < \Delta_T$ should be smaller than the output dimensionality D, so that $D - J$ output features are reconstructive. Otherwise, the output features would not have a reconstructive part, negating the advantages of the method. This might happen if too many slow parameters, their mixtures, or higher-frequency harmonics are present in the data. One might avoid this problem by setting a smaller Δ_T, by directly controlling the number of the slow features, and by using training graphs with a small number of optimal free responses with $\Delta < 2.0$.

5.6.3 Age, Gender and Race Estimation

The problem of age estimation from adult facial photographs has been chosen for this work, because it appears to be an ideal problem to test the capabilities of HiGSFA. For age estimation, PCA is not very useful because wrinkles, skin texture and other higher-frequency features are poorly represented. Therefore, it is not obvious and even counter-intuitive that feature slowness improves by incorporating PCs in HiGSFA. Improvements on feature slowness using other supervised learning problems, such as digit and traffic sign recognition, and the estimation of gender, race, and horizontal-position from face image patches, would be less conclusive because for such problems a few PCs encode the discriminative information relatively well.

To estimate age, race, and gender simultaneously, a graph G_3 has been constructed combining three pre-defined graphs, encoding sensitivity to the particular labels, and favoring invariance to any other factor. The graphs used are a

serial graph for age estimation with 32 groups and two clustered graphs with 2 clusters each for race and gender estimation. The number of features with $\Delta < 2.0$ that can be extracted from these graphs is 15, 1 and 1, respectively (due to their particular geometry, see Section 4.4.2). Since the vertex weights of these graphs are not exactly proportional, the conditions to apply the theory of combination of graphs are not fulfilled perfectly. This results in a combined graph that has more than 17 optimal responses with $\Delta < 2.0$ instead of 17, as predicted by theory. In spite of this small discrepancy from the theoretical assumptions, the method yields the expected features and the first 5 of them contain most of the information relevant to solve these problems.

Out of the 75 features extracted in the top-most node, 8 are slow and 67 are reconstructive. The input to the supervised step is the first 4 to 7 slow features of the output, and the reconstructive part is not used. This shows that HiGSFA and HGSFA concentrate the label information in the first features. One can actually replace the iGSFA node on the top of the HiGSFA network by a regular GSFA node, so that all features are slow, without affecting the performance. The superiority in age estimation of HiGSFA over HGSFA is thus not due to the use of principal components in the final supervised step but to the higher quality of the slow features.

The performance of HiGSFA for age estimation is the highest reported on the MORPH-II database with an MAE of 3.497 years. Previous state-of-the-art results are an MAE of 3.63 years using a multi-scale CNN (Yi et al., 2015) and 3.92 using BIF+rKCCA+SVM (Guo and Mu, 2014). These results suggest that age estimation is not quite a convolutional problem; age-sensitive features may be considerably different at different facial landmarks. Multi-scale techniques for CNNs reduce this problem but do not eliminate it.

Even though the proposed system performs slightly better than state-of-the-art algorithms, this is not the main goal of this research. The specific goal is to improve HGSFA. In this sense, the central claim is that HiGSFA is better than HGSFA regarding feature slowness, input reconstruction and estimation accuracy.

5.6.4 Reconstruction from Slow Features

The experiments confirm that PCA is more accurate than HiGSFA at reconstruction using the same number of features (here 75), which was expected because PCA features are optimal when reconstruction is linear. From the 75 features extracted by HiGSFA, 67 are reconstructive (8 fewer than in PCA) and they are computed hierarchically (locally), in contrast to the PCA features, which are global. Thus, it is encouraging that the gap between PCA and HiGSFA at input reconstruction is moderate. In turn, HiGSFA is more accurate than HGSFA, because reconstruction is the secondary goal of HiGSFA, whereas HGSFA does not pursue reconstruction. The improved reconstruction capability of HiGSFA might facilitate certain applications, such as image morphing.

Since the HiGSFA network implements a nonlinear transformation, it is rea-

sonable to employ nonlinear reconstruction algorithms. Nonlinear reconstruction can provide more accurate reconstructions in theory, but in practice it is difficult to train such type of algorithms well enough to perform better on test data than the simpler global linear reconstruction algorithm. An algorithm for nonlinear reconstruction that minimizes the feature error e_{feat} has been described in Section 5.4.5. However, since the number of dimensions is reduced by the network, one can expect many samples with feature error $e_{\text{feat}} = 0$ (or e_{feat} minimal) that differ substantially from any valid input sample (e.g., not looking at all like a face). To correct this problem, one might need to consider the input distribution to limit the range of valid reconstructions.

In the context of convolutional networks, generative adversarial networks (Goodfellow et al., 2014; Alec Radford, 2015; Denton et al., 2015) have been used to generate random inputs with excellent image quality. For HiGSFA networks, such a promising approach might be useful also to do nonlinear reconstruction, if properly adapted.

5.6.5 Final Words

A hypothesis of this dissertation is that it is possible to develop successful learning algorithms based on a few simple but strong learning principles and heuristics, and this is the approach that I tried to pursue with HiGSFA. An algorithm that might be strong but cannot be understood and justified analytically would be of less interest to me.

The proposed algorithm is general purpose (e.g., it does not know anything about face geometry), but it is still capable of outperforming special-purpose state-of-the-art algorithms, at least for the age estimation problem. The results show the improved versatility and robustness of the algorithm and make it a good candidate for many other problems of computer vision on high-dimensional data, particularly those lying at the intersection of image analysis, nonlinear feature extraction, and supervised learning.

Chapter 6

Discussion

This chapter presents a general, global, and conceptual discussion. Additional method-specific topics are addressed in the discussion sections of each chapter.

This dissertation addresses the question of how to use SFA to solve supervised learning problems on high-dimensional data in an efficient and accurate manner. In order to answer this question, a family of algorithms that extend HSFA has been proposed (e.g., HGSFA, HiGSFA, ELL method). These extensions enhance the capabilities of HSFA for supervised learning, amplifying its strengths, reducing its limitations, and allowing a deeper understanding of the role of SFA in supervised learning scenarios. Preference has been given to principled methods when developing the extensions, because the goal is to develop algorithms supported by theory that can be applied to a broad range of supervised learning problems with minimal customization instead of having to develop a new method every time. The most complex and promising extension is HiGSFA, since it encompasses all techniques used in the remaining extensions and provides the strongest results. Experiments prove the efficacy of the approach: HiGSFA yields much slower features than HSFA and has a good computational complexity of the same order as HSFA. Moreover, HiGSFA yields more accurate solutions than HSFA for classification and regression, providing competitive results for several face-analysis problems (x-pos, y-pos, scale, age, gender, race). For the problem of age estimation, HiGSFA is superior to CNNs and even improves the state-of-the-art age estimation accuracy.

Supervised learning on real-life high-dimensional data is a challenge. For this type of data, existing classification and regression algorithms sometimes yield good results for specific tasks, but generally suffer from poor efficiency (e.g., have quadratic or cubic complexity w.r.t. the number of samples or dimensions) and their accuracy is usually suboptimal. In the last decade, convolutional neural networks (CNNs) have allowed excellent accuracy for several real-life supervised tasks on high-dimensional data and good efficiency with an architecture that extensively profits from parallelism in graphics cards to speed up otherwise lengthy computations. However, the development of CNNs has been typically based on technical advances and less on theoretical ideas, like Zeiler and Fergus (2014) affirm: "*... there is still little insight into the internal operation and behavior*

of these complex models (CNNs)*, or how they achieve such good performance. From a scientific standpoint, this is deeply unsatisfactory*".

Algorithms with more solid scientific foundations are desirable. Therefore, the extensions proposed in this work follow a principled approach. The most advanced extension, HiGSFA, provides a new way to construct and train deep neural networks in a principled, scalable, and accurate way. The chosen principles are mostly inspired by biological neural systems, particularly the visual system of mammals.

The main hypothesis of this dissertation (see Section 1.4 for a list of all hypotheses) is that a few simple but strong principles and heuristics, such as the slowness principle, allow the design of successful supervised learning algorithms that provide good efficiency and label-estimation accuracy. This hypothesis is supported by experimental results, including those on age estimation, where HiGSFA achieves a mean average error (MAE) of 3.50 years improving the previous state-of-the-art result of 3.63 years by a multi-scale CNN (Yi et al., 2015). HGSFA achieves an MAE of 3.92 years, which is a good accuracy but not as good as that of HiGSFA. HiGSFA (among other extensions) is well suited for large scale real-life problems since it has linear computational complexity in the number of samples and in their dimensionality. Moreover, additional challenging supervised learning problems may be handled by HiGSFA without modifying the approach, if enough training data is available, because the feature space spanned by the network is highly complex.

A second hypothesis of this work affirms that even though the slowness principle is unsupervised, it is a fundamental learning principle that can be used for supervised learning in an efficient and accurate way. This is corroborated by the reported results, particularly when the slowness principle is implemented by SFA and combined with additional principles. A more detailed discussion of this topic is left to Section 6.2.

HiGSFA relies on the following principles and heuristics: the slowness principle, divide and conquer computation, spatial localization of features, exploitation of label similarity through slowness, nonlinear expansions that are robust to outliers, information preservation, and multiple information channels. The first three principles have already been considered by HSFA before this work (Wiskott, 1998; Franzius et al., 2011). Table 1.1 shows the methods used to implement these principles and heuristics.

A third hypothesis stated in Section 1.4 focuses on real-life face images and maintains that HSFA (and its extensions) can be robust enough to extract the desired labels and overcome, to a good extent, the variations present in this type of data. This hypothesis is also verified in general using various problems on face images: age, gender, race, x-pos, y-pos, and scale estimation. HGSFA is capable of computing features mostly invariant to several image variations. The most difficult variation seems to be the lighting conditions. Invariance to extreme lighting conditions (strong unidirectional light and on-face shadows) is necessary for the subproblems comprised by face detection (x-pos, y-pos, scale).

In part, the difficulty lies in a too small number of images with extreme lighting that does not represent this type of variation well enough. However, good results are still achieved even for this extreme type of image variation.

The extensions to HSFA proposed in this thesis can be easily combined (see Figure 5.1). Each extension incorporates an additional principle and yields improved results, making the final combined system significantly more accurate than HSFA. Efficiency is only minimally affected by the extensions; for instance, the computational complexity of HiGSFA trained on a combination of three efficient pre-defined graphs has the same order of complexity as HSFA. However, feature quality is highly improved, yielding also improved feature slowness and label estimation accuracy.

Experiments show that HiGSFA and HGSFA can extract various face attributes including age, gender, race, x-pos, y-pos, and scale (experiments on the last three attributes and HiGSFA not reported due to space considerations). In all cases, no face model or landmarks were explicitly provided or constructed; only the position of the eyes and mouth was used in order to normalize the images. Good results have also been obtained for digit and traffic sign recognition. The approach is general purpose: To address a new problem or dataset, one only needs training samples and their respective labels.

The employed datasets have been limited to images and the focus has been on solving classification and regression problems. However, the proposed extensions can be applied to other types of data with minimal or small modifications, such as audio and video, as well as other problems, such as class density estimation, and other sub-fields of machine learning, such as reinforcement learning.

The remainder of the discussion is structured as follows: The proposed extensions are shortly summarized. Then, the use of the slowness principle for supervised learning is justified. Afterwards, an information analysis that inspired the different extensions to SFA is presented. Thereafter, implications of this research and negative results are outlined, and promising topics for future research are proposed. A conclusion closes the dissertation.

6.1 Proposed Extensions

The three main practical contributions of this work are the proposed extensions to SFA: (1) graph-based SFA (GSFA), (2) exact label learning (ELL), and (3) hierarchical information-preserving GSFA (HiGSFA). Each extension has been presented and detailed in a separate chapter, but they and their main advantages are shortly synthesized below.

6.1.1 Graph-Based SFA (GSFA)

The GSFA optimization problem is a natural extension to the SFA optimization problem that allows transitions between arbitrary samples as specified by the edges of a training graph. This allows us to include more information regarding

desired output similarities (SFA involves $N-1$ transitions, whereas GSFA involves up to $N(N+1)/2$ transitions). Contrary to SFA, GSFA does not involve a sequence of samples. The training graph as a whole constitutes the training data, and the generalized slowness objective given by Equation (6) is no longer temporal but depends on the graph connectivity.

An optimal solution to the GSFA optimization problem can be computed using the GSFA algorithm, which is similar to the SFA algorithm; the fundamental change is how the covariance matrix of the data and the derivative second-moment matrix are computed. The new computation accounts for each sample and edge of the training graph, as well as for their respective weights.

Compared to SFA + reordering, the quality of the features extracted by GSFA is improved, providing higher label estimation accuracy and more robustness to new data. Labels are not provided explicitly to GSFA but implicitly in the connectivity of the graph. Pre-defined graphs encode a large part of the label information: Label similarities are encoded by the strength of the edge weights. When the goal is regression, even more information can be encoded by means of an ELL graph.

6.1.2 Exact Label Learning (ELL)

The ELL method is a mathematical framework that allows us to: (1) Analyze the inherent properties of existing graphs. This method decomposes an existing graph into its optimal free responses and computes their corresponding delta values, providing an alternative representation of the graph that is useful to understand its underlying structure. (2) Design new graphs sensitive to any desired labels. The constructed ELL graph represents the label information more precisely than pre-defined graphs, typically resulting in a dense edge-weight matrix.

ELL provides excellent flexibility regarding the labels that can be encoded in the graph. One can use ELL, for example, to encode multiple labels in a single training graph, while balancing the importance of each label. The resulting ELL graph can then be used to train a single HGSFA network to learn multiple labels simultaneously. The resulting HGSFA network is more efficient than using separate HGSFA networks and provides even higher label estimation accuracies for some experiments (e.g., those reported in Table 5.4 on page 119, but not those reported in Table 5.8 on page 124). When and why exactly these label synergies occur is an open question.

6.1.3 Hierarchical Information-Preserving GSFA

Experiments with HGSFA and a conceptual analysis of how features are processed by HGSFA networks have suggested the presence of two shortcomings in HGSFA and HSFA: unnecessary information loss and poor input reconstruction. HiGSFA reduces these problems and is a *scalable* algorithm (if the network structure is appropriate) that achieves competitive results for different problems.

The key idea behind HiGSFA is to complement the slowness principle with information preservation by dividing the feature vector into two parts. The slow part contains slow components, whereas the reconstructive part contains features that allow linear reconstruction of the input. Reconstructive features allow more information to propagate to the top of the network.

A comparison of HGSFA and HiGSFA networks shows that HiGSFA inherits the advantages of HGSFA and that both span a similar feature space. However, results confirm that compared with HGSFA the features computed by HiGSFA are slower, more reliable for test data (since they overfit less), contain more label information, and result in more accurate label estimations. The experimental results show that HiGSFA is a powerful and promising algorithm for supervised learning on high-dimensional data, especially for the solution of regression problems.

A list of the datasets and supervised learning problems addressed in this work is provided in Table 6.1.

Databases	**Problems considered**
BioID (Jesorsky et al., 2001) Caltech (Fink et al., 2003) CAS-PEAL (Gao et al., 2008) FaceTracer (Kumar et al., 2008) FRGC (Phillips et al., 2005) LFW (Huang et al., 2007)	Face detection, including: estimation of the horizontal and vertical position of a face in a face patch (x-pos, y-pos), as well as the face scale and position of the eyes. Discrimination of face and non-face patches.
Images rendered using FaceGen and PovRay	Estimation of gender and average color from artificial face images.
FG-Net (Cootes, 2004) MORPH-II (Ricanek Jr. and Tesafaye, 2006)	Estimation of age, gender and race from real face photographs.
MNIST	Hand-written digit recognition.
GTSRB (Houben et al., 2013)	Traffic sign recognition. Compact discriminative features.

Table 6.1: Problems and databases addressed in this work using the proposed extensions to SFA.

6.2 Supervised Learning via Slowness Principle

In biological learning systems labels are not available (or at least they are not explicitly supplied). The slowness principle provides an excellent learning mechanism for such an unsupervised and temporal setup, because it captures the natural heuristic that fundamental aspects of the environment change slowly on average (e.g., due to physical and biological constraints).

One can do supervised learning with SFA by applying it to the raw (e.g., sensory) data. The features extracted by SFA must be post-processed by an explicit supervised learning algorithm. However, such an approach is usually

less effective, because the extracted features may be too unspecific w.r.t the particular label one wants to learn: Many aspects of the data may be slow besides the aspects related to the label. Therefore, this temporal SFA approach is suboptimal, because the label of interest cannot be guaranteed to be contained in the first few extracted features.

It is possible to obtain more label-specific features by using SFA + reordering, or even better, by using GSFA. Training graphs allow the inclusion of more label information than sample reordering (the amount of information depends on the particular graph used). Strictly speaking, GSFA does not rely on the slowness principle as originally conceived, because the data is not temporal. However, the probabilistic interpretation of a graph indicates that GSFA is equivalent to SFA if one trains SFA with an artificial sequence generated from a specially constructed Markov chain, in which similar labels translate into frequent transitions. Hence, GSFA can be seen as a computational optimization of learning in such an unsupervised and temporal setting.

This equivalence between supervised and unsupervised learning in GSFA is a remarkable property that is also present in HGSFA (and in HiGSFA if a weighted version of PCA is used). Such a property originates from SFA/the slowness principle and appears to be unique to these extensions. It is a very appealing property conceptually, because it strengthens the hypothesis of slowness as a fundamental learning principle. For instance, unsupervised HiSFA is equivalent to HiGSFA, if HiSFA is trained with appropriate data. Therefore, (unsupervised) HiSFA is more accurate than (supervised) CNNs for age estimation, if trained in an environment where the data is generated randomly using a specific Markov model (see Section 3.2.5), and equally accurate as HiGSFA.

6.3 Analysis of Information as a Criterion for Algorithm Design

Many of the contributions of this dissertation arose from an understanding of SFA as an algorithm not just for feature extraction but rather for information processing. This section explains how this view has lead to GSFA, ELL and HiGSFA.

Assume the goal is to solve a regression problem with a single target label, and consider the information available to regular SFA using the sample reordering method: The samples on their own provide little or no information regarding the target label, and they could also be associated with several other labels. Information about the specific target label is available to SFA through the order of the samples in the training sequence. That is, only label-rank information is available and the remaining *explicit* label information is lost (the samples may still contain all label information *implicitly*, but it cannot be distinguished from alternative labels).

Although label-rank information is available to the SFA + reordering method in principle, this information is not fully exploited by SFA: The slowest extracted

feature, for example, minimizes average squared differences between *consecutive* output values. Thus, strictly speaking, SFA only enforces output similarities between consecutive outputs at times t and $t+1$. Nothing is explicitly guaranteed regarding the similarities of outputs at time t and $t+k$, for $|k|>1$.

The use of training graphs in GSFA allows us to specify more transitions than in standard SFA, encoding desired output similarity information more precisely (and implicitly exploiting the label information better). Pre-defined graphs, such as the serial or sliding window graphs, are also constructed based on the label rank but make better use of the label information than standard SFA (reordering method). For example, when GSFA is trained with a serial graph, output similarity is enforced between a sample n in group l and all samples n' in groups $l-1$ and $l+1$. However, the transitions of pre-defined graphs are still somewhat imprecise, because they are specified by edge weights that usually take only the values 1 or 0, instead of varying continuously depending on the label values.

The ELL method is the next logical step to increase and maximize the label information encoded by the graph. ELL graphs make even better use of the label information by considering the exact numerical value of the labels (not only their rank) to set the weight of all edges precisely. The resulting edge weight between samples n and $n+k$ takes into account not only the corresponding pair of labels but in fact the labels of all samples.

GSFA trained with an ELL graph is aware of the complete label information, except for the offset, scale and a global sign of the label, which is unavoidable due to the constraints and objective function of GSFA. The GSFA optimization problem requires that the output features have weighted zero-mean and weighted unit variance, but is insensitive to the feature polarity. If the feature space is unrestricted and the samples are all different, the slowest feature extracted by GSFA + ELL graph is then an affine transformation of the label. This result extends to multiple label learning (though one should consider the issues of label decorrelation and possible feature combinations).

The analysis above focuses on label information that is explicitly provided for training data and how it is encoded by the sample connectivity. However, it is also possible to analyze the *implicit* label information that is encoded in the samples (the implicit label information can be measured as the mutual information between the samples and their labels). Such implicit label information is propagated by the network and is used to estimate the label of old and new samples. For age estimation, these explicit and implicit types of label information are called ground-truth age and apparent age, respectively. It has been shown that HGSFA can lose part of the implicit label information (see unnecessary information loss in Section 5.3.3). HiGSFA, as the name implies, preserves and propagates more information about the original data to higher layers. In general it is not possible to always identify all implicit label information at the local level (i.e., during learning in an individual node), and some of it may be lost by the hierarchy. Therefore, the heuristic behind HiGSFA is to preserve

as much information of the local inputs as possible, which in turn is likely to contain a large amount of implicit label information. Such additional implicit label information is responsible for the superior performance of HiGSFA over HGSFA.

One can compare the information processing in HiGSFA and CNNs, at least abstractly. The effect of information preservation in HiGSFA might be similar to the effect of unsupervised pre-training and joint supervised and unsupervised optimization in CNNs. All of these techniques result in more input information being propagated from the original images to the network outputs. During feature extraction, the information flow in HiGSFA and CNNs is similar (bottom up). However, the information flow during training is quite different: Label information in CNNs is not available to the layers of the network directly, but it is provided indirectly as an error signal that is back-propagated and used to compute local gradients. In this sense, HGSFA and HiGSFA seem to make more extensive use of the explicit label information (encoded in a training graph and available to all nodes). Still, the error signal used by backpropagation provides useful feedback from the network output to the local nodes that combines explicit and implicit label information. Currently, neither HGSFA nor HiGSFA employ backpropagation, though in principle such a method may further improve global slowness and label estimation accuracy (at the cost of local slowness in some nodes other than the top node). Therefore, one open line of research is the implementation of backpropagation for HiGSFA as supervised post-training.

6.4 Implications of this Work

This section outlines how this research might be exploited by researchers in relevant fields:

(1) HiSFA is very likely to provide better features than HSFA for unsupervised learning. Information preservation and HiGSFA were motivated in the context of supervised learning and all experiments were conducted in supervised learning settings. Although this lies out of the scope of this work, the theory suggests that unsupervised learning settings may also benefit from information preservation. Therefore, I recommend the use of HiSFA instead of HSFA in future experiments and simulations, particularly in the fields of robotics and computational neuroscience.

(2) This work confirms that if label information is available, supervised learning principles that explicitly use the label information may be more appropriate or at least complement the unsupervised ones. General principles of unsupervised learning include the slowness principle (in its original temporal form) and the preservation of input similarities (manifold learning, as in LPP). These principles are excellent for unsupervised learning and for solving problems involving feature extraction for an unspecific task, but are suboptimal for supervised learning.

(3) Consequently, the use of standard SFA or HSFA is discouraged if the goal is to solve supervised learning problems. As shown both theoretically and empirically, the approach SFA + supervised step is suboptimal w.r.t. label estimation accuracy and can be improved by using the extensions proposed in this dissertation, particularly GSFA and HiGSFA.

(4) The use of deep HSFA architectures with light-weight layers is suggested (i.e., layers that can be trained efficiently due to a small expanded dimensionality). Hierarchical SFA networks used in other works have a different structure compared with the networks used in this thesis. In general, previous networks have between 2 and 5 layers, the nonlinear expansion functions are polynomial, and the extracted features are less specific, requiring large output dimensionalities to achieve acceptable performance. The proposed networks have a large number of layers (up to 11), but these are lightweight. For example, the networks used for face detection have up to 11 layers, the output dimensionality of the nodes is at most 60 features, and the fan-ins of the nodes are 4×4 pixels in the first layer and either 1×2 or 2×1 nodes in the remaining layers. This type of network structure has been used by both HGSFA and HiGSFA as a result of optimizing the network empirically in terms of accuracy on test data and training efficiency. Different datasets have required only moderate adjustments to this network structure, and it is well suited to handle high-dimensional data and reduce overfitting.

(5) Researchers using polynomial expansions and experiencing overfitting problems are encouraged to try the 0.8Exp expansion. A key element for the success of HGSFA and HiGSFA is the choice of the nonlinear expansion functions. These functions are usually required since most interesting problems are nonlinear. The most common expansions are polynomial expansions, which provide a complex feature space; if the degree is unrestricted, this expansion constitutes a basis for all polynomials and can approximate most functions of practical relevance. Moreover, polynomial expansions are conceptually simple and mathematically tractable. However, they have the disadvantages that they are less robust against outliers and tend to overfit (Escalante-B. and Wiskott, 2011). Although resistance against outliers has not been formally addressed in this thesis, it is possible to improve it by using expansion functions that saturate or are normalized. An example is the 0.8Exp expansion (see Section 3.5.1). Although the 0.8Exp expansion is less elegant and the exponent has been tuned experimentally, it can be justified based on a computational model (Escalante-B. and Wiskott, 2011). The experiments of this dissertation show that sometimes one can obtain even better results by combining the 0.8Exp expansion with a polynomial expansion or a normalized version of it (this was exploited by the HiGSFA network for age estimation).

(6) The information analysis of SFA and its extensions consists of viewing these algorithms as methods for information processing and encoding. Such a paradigm turned out to be quite effective. Further use of this paradigm might

be useful to develop further extensions to SFA and design other algorithms in machine learning.

This work has allowed the transformation of HSFA from an algorithm that has been successful for supervised learning mostly on artificial or low-complexity datasets into an algorithm that can deal with more challenging tasks on real images (e.g., HiGSFA). The original title of this work was "Feature Extraction from Face Images with Hierarchical Slow Feature Analysis". Such an unspecific title had been chosen because at the beginning of the project it was not evident that improvements to HSFA would allow the solution of face analysis problems—among others—with good accuracies.

6.5 Negative Results

In recent years, the importance of publishing negative results has been acknowledged for ethical reasons and to facilitate future research. This section briefly describes some experiments that did not validate their hypotheses but are nevertheless valuable.

Hexagonal receptive fields did not outperform square ones. According to the heuristic of feature localization, which states that data relevant to compute useful features tends to be spatially localized, one might expect better results using circular receptive fields than using square or rectangular receptive fields (constrained to the same number of pixels). A good compromise between square and circular receptive fields are hexagonal receptive fields. They are similar to a honeycomb structure with receptive-field centers positioned in a regular lattice. Two of their advantages are that they are regular and cover the plane without overlap (though, if desired, overlap is also possible). However, (unpublished) experiments showed no significant improvement in estimation accuracy compared with square receptive fields (using HGSFA for age and gender estimation from artificial images). A theoretical explanation is that SFA can learn to ignore certain pixels of the receptive fields, if such pixels indeed do not contribute to slowness maximization, adjusting the effective shape. These results suggest that even though square/rectangular receptive fields are less 'natural' from a biological perspective and suboptimal from the feature localization heuristic, they are convenient in practice since they are easy to code and give similar accuracy to hexagonal receptive fields.

Not all methods that combine slowness and principal components are equally successful. The problem of information loss motivated the combination of the slowness principle and information preservation. This idea turned out to be quite beneficial in HiGSFA, where information preservation is implemented by computing principal components to minimize a reconstruction error. I tested at least two other approaches that yielded less accurate results. In one of them the slowness objective function was modified by introducing a bias towards the principal components in the matrix $\mathbf{R}$ of the GSFA algorithm (the larger the principal component, the 'slower' the direction appeared). In another approach

a new optimization objective was used, consisting of the weighted average of the objective functions of SFA and PCA. Both approaches suffered from poor robustness and needed additional parameters that are difficult to tune. However, HiGSFA overcomes these problems effectively by learning the slowest features using the objective function of SFA/GSFA in unaltered form and learning reconstructive features by using PCA.

First methods for semi-supervised HGSFA did not improve results. The goal of semi-supervised learning is to exploit inexpensive unlabeled data to obtain higher accuracy on test data than when only labeled data is available. This research included semi-supervised learning experiments where the horizontal position of faces was estimated using HGSFA. The unlabeled images had faces centered at random horizontal positions. These images were used to improve the accuracy of the sample covariance matrix $\mathbf{C}$, which otherwise would have been computed using only labeled samples. However, this approach did not improve the results. A more theoretical analysis of such an approach indicates that it was incorrect and equivalent to training HGSFA with an extended graph in which each of the unlabeled samples is disconnected. Thus, HGSFA learned features similar to those of classification, where all labeled samples constitute a single class, and each unlabeled sample constitutes a new class. Another approach joins all the unlabeled samples in a complete sub-graph that extends the original graph. This improved approach has provided better results than the supervised setting in some experiments, but is still suboptimal: The first optimal free response of the extended graph is binary and encodes whether the sample is labeled or not. The next few free responses encode the label for the labeled samples in the same way as the original graph and are zero for the unlabeled ones. Therefore, this improved approach is still not fully backed by the theory of optimal free responses of training graphs, indicating that in order to do semi-supervised learning with HGSFA/HiGSFA other methods need to be developed.

HGSFA is less effective for small datasets. The classification experiments addressed in this thesis are handwritten digit recognition (MNIST), traffic sign recognition (GTSRB), and gender/race recognition (MORPH-II and a synthetic database generated using FaceGen software). Besides these experiments, an experiment on object classification was carried out using a subset of a private image database of Dr. Christian Faubel. The images were taken in the local robotics lab and involve everyday objects lying on a white table, such as a tin can, hair gel, and a package of cookies. There are 30 different objects, approximately 33 52×52-pixel images per object, and each object lies in a few positions. The images were converted to grayscale to remove color as a trivial clue for classification, and HGSFA was trained on them. However, classification accuracy did not generalize well to test images due to the small number of images. This occurred even though the model complexity had been reduced. Other experiments suggest that a ratio of at least 10 to 1 in the number of samples and their dimensionality (possibly after an expansion) is usually enough

to achieve low overfitting, a ratio that was not achieved for this data. The results on object classification indicate that HGSFA (and other extensions of SFA) would profit from new methods to deal with scenarios in which the number of images is small, which occurs frequently in some fields, such as in medical image processing. Possible improvements include explicit regularization and effective methods for semi-supervised learning.

6.6 Recommendations for Future Research

A direction for future work is the implementation and analysis of the extensions to the ELL method proposed in Section 4.5.5, namely: (1) reduction in the number of edges in the ELL graph via graph trimming, (2) grouping of data samples in a pre-processing step, and (3) combination of graphs. Extensions 1 and 2 would be useful to reduce the computational complexity of the ELL method. Extension 3 has already been used in Chapter 5 for the age estimation experiment by combining a serial graph and two clustered graphs, but further ways of combining graphs are possible. For example, one may combine two regression graphs: the serial and reordering graphs. Such a graph combination might yield higher label estimation accuracy than each graph alone: The serial graph has a large number of edges and provides good generalization, whereas the reordering graph does not incur in the quantization error of the first one caused by label discretization.

The ELL method supports learning of multiple target labels and guarantees that the optimal free responses span the target labels exactly if the number of output features is at least L (i.e., the number of target labels). However, it can be impractical to preserve L features per node if L is large, particularly when using HGSFA and HiGSFA. Propagating fewer than L features may result in poor approximation of some target labels. A possible solution to this problem is to develop methods that concentrate the label information (e.g., by computing a compressed representation of the labels) to reduce the number of effective target labels. In fact, one can often compress the labels of datasets with several labels, because they are categorical and sparse (e.g., ImageNet database, Deng et al., 2009).

The advantage of learning auxiliary labels in addition to the original ones is a higher label estimation accuracy. This might seem counter-intuitive since the inclusion of auxiliary labels may influence the slowest extracted features making them less similar to the original labels. However, auxiliary labels have been justified by information theory and a smoothness heuristic (see Section 4.3.4). There are different methods to compute auxiliary labels, but a particular one has been suggested with Equation (127). Here, an interpretation of this equation is provided when the original label is a single label $\boldsymbol{\ell}$: The target (i.e., original and auxiliary) labels computed using this equation are similar to the basis used in the discrete cosine transform, if the period is $T = \max(\boldsymbol{\ell}) - \min(\boldsymbol{\ell})$. An alternative to (127) that has not been explored in this work but is neverthe-

less interesting, is to use auxiliary labels $\boldsymbol{\ell}^2$, $\boldsymbol{\ell}^3, \ldots, \boldsymbol{\ell}^L$, where the superscript is component-wise exponentiation. Such auxiliary labels would result in a different basis. Yet another possible way to compute auxiliary labels is to follow an optimization criterion and explicitly maximize estimation accuracy. The assignment of the eigenvalues could also be optimized in this way. However, apparently it is difficult to determine optimal auxiliary labels and eigenvalues analytically.

In the ELL method, several eigenvectors remain unspecified (those with indices $L+1$ to $N-1$) and their corresponding eigenvalues are implicitly set to zero. As suggested by an anonymous reviewer of the Journal of Machine Learning Research, one could use these eigenvectors and their eigenvalues to construct graphs with special structural constraints, such as a minimum and maximum number of edges per vertex.

HiGSFA has proven to be a promising algorithm for face analysis due to its good estimation accuracy in various problems (age, gender, and race estimation). One may further improve estimation accuracy by increasing the model complexity via more complex hierarchical networks. For instance, one can increase the overlap of the receptive fields and use more complex nonlinearities. More training images might improve accuracy as well. A similar effect to having more training data might be obtained by implementing true face-distortion methods and not just simple transformations of the images (i.e., small rotations, rescalings, and translations).

This work has shown that even though CNNs are one of the most successful algorithms for image analysis currently available, HiGSFA provides better accuracy for the problem of age estimation. One possible explanation is that the most useful low-level features for age estimation depend strongly on the specific face region (e.g., eye, nose, mouth, forehead) and a convolutional structure might be too unspecific to provide a single set of features that are good for most regions. However, a more general claim regarding HiGSFA and CNNs cannot be made before these algorithms are compared experimentally using additional datasets and problems. From the impressive results of CNNs in recent years, it would be surprising if CNNs do not outperform HiGSFA for most regression problems (for classification problems the current lead of CNNs over HiGSFA is clear and significant).

To further boost the accuracy of HiGSFA one may continue the paradigm of using (to some extent) general principles and heuristics to guide algorithm design and incorporate additional ones. The empirical comparison of HiGSFA and CNNs suggested above could be useful to find possible improvements to HiGSFA, perhaps also principled ones.

Multi-scale CNNs achieve good performance for age estimation (though not as good as HiGSFA). This type of networks has receptive fields centered at specific facial points and achieves an MAE of 3.63 years compared with an MAE of 3.79 years using non-aligned receptive fields (i.e., receptive fields centered not at the corresponding facial points but at their average position computed over all training images). The benefit of using receptive fields centered at specific

facial-points is that the complexity of the transformation needed to extract the labels is reduced, because the learned features can be more specific to the facial-point region, and the feature space is not wasted in learning position invariance. The application of multi-scale techniques to HiGSFA is feasible, and it is likely to improve the estimation accuracy.

Besides the research directions outlined above, there are two important practical applications: face detection and face recognition. These applications are considered in the following subsections.

6.6.1 Face Detection

The efficacy of HGSFA has been shown experimentally using the problem of estimating the horizontal position of faces in image patches. Other experiments that have been carried out but have not been documented in detail in this thesis include the estimation of the vertical position and size of faces, and the discrimination of face image patches from non-face image patches. A different network was trained for each problem and they were later combined to build the HGSFA-based system for face detection described in Section 2.6.4.

Such a system participated in a face detection competition (Mohamed and Mahdi, 2010) and defeated a system based on the Viola-Jones algorithm, which was the only competitor. While several factors were in favor of HGSFA, such as the evaluation criteria and parameter choice, the system still constitutes a promising proof of concept and demonstrates the capabilities of the approach.

The HGSFA-based face detection system already achieves quite good detection on frontal images. The challenge is to improve detection rates on uncontrolled images (i.e., increase the number of true positives and reduce the number of false negatives). Various ideas may be useful to achieve this: (1) One can use more training images with more variations. (2) One can use the ELL method and HiGSFA instead of HGSFA. (3) Different pose parameters can be estimated simultaneously using a single network (learning horizontal position, vertical position, and face scale simultaneously). (4) One can learn the screen angle of the eye line and possibly also additional angles of the head pose. (5) The face discrimination step (which distinguishes face from non-face patches) can be further improved. The current version has been trained with non-face patches centered at *random positions*. However, the real problem is to discard non-face patches that have already been *centered* by the pose normalization steps. Thus, dark regions in non-face patches might have been shifted to the positions where the eyes usually appear, for example. Therefore, the training data could be improved to account for such a centering step. Additionally, the number of non-face images could be increased.

6.6.2 Face Recognition

Multiple problems on face analysis have been addressed in this dissertation, but face recognition is a challenging problem that has not yet been addressed

explicitly. This section outlines two approaches for face recognition that use techniques proposed in this thesis.

(1) Invariant learning of subject attributes. This approach uses HiGSFA to learn features that are sensitive to several facial attributes. Matching subject identities can then be reduced to matching feature vectors. HiGSFA would be used to learn not only age, gender, and race simultaneously, but many more attributes, such as the shape and size of the face, mouth, eyebrows, nose, and ears, the distance between different keypoints, the presence and location of pimples, moles, and wrinkles, the most likely nationality or geographical location, and the body mass index.

The identity of the most similar face in feature space can be found by minimizing the L_2 norm of the feature vector differences or by using a probability model. Alternatively, one could estimate each attribute separately from the feature vector and find the face with the most similar attributes (i.e., matching would be done in attribute space instead of feature space).

Since the world population is about 7,000 millions, fewer than 33 bits of information are necessary to identify any subject uniquely ($2^{33} \approx 8.6 \times 10^9$). This gives an estimate on how many attributes should be learned. Some attributes, such as gender, provide about 1 bit of information, whereas other attributes can provide less information (e.g., presence of pimples) or more information (e.g., age). In practice, one should keep in mind that attributes are frequently not independent and compensate for it with enough redundancy.

(2) Invariant learning of subject identities. A more elegant approach is to train HiGSFA on images of different subjects, where each subject is considered as a single class. To ensure robustness to new subjects and new images, one should include several subjects in the training data and provide a large number of images per subject, including a large amount of variations in the images conditions (pose, backgrounds, lighting, etc.). For classification one typically uses the clustered graph. The optimal free responses of this graph would provide a constant and unique representation for images of any particular subject. If the features are accurately learned (no overfitting), an unseen subject would have a new representation that would be (ideally) similar to the representation of subjects with similar attributes. However, as previously explained, it can be difficult to handle such a large number of classes when hierarchical processing and the clustered graph are used, because one would need to preserve up to $C-1$ features for good recognition, where C is the number of classes or subjects in this case. Therefore, this is an ideal scenario to employ the compact graph method described in Section 4.4.3, where one can limit the number of output features to a value between $\log_2 C$ and $C-1$, if binary target labels are used.

Apparently approaches (1) and (2) are both feasible; one might need to implement them to determine which one is better in practice in terms of accuracy and efficiency. Although this would be less elegant, one might also improve accuracy through multi-stage/hierarchical face recognition. For example, in a

first step, a demographic group to which a face most likely belongs can be determined (e.g., females/adults). In a second step, the concrete subject may be identified within this group.

6.7 Conclusion

This dissertation presents various extensions to SFA that allow us to solve supervised learning problems on real-life high-dimensional data with competitive accuracy and efficiency.

SFA is an elegant algorithm backed by a strong theoretical foundation. At the same time it is versatile and has practical applications in both supervised and unsupervised learning. This work confirms that SFA and the slowness principle may be a fundamental algorithm and learning principle, respectively, and shows that they may play a more important role for supervised learning than originally thought.

The features extracted by the extensions are quantitatively and qualitatively better than those extracted by HSFA. In the particular setting of supervised learning on high-dimensional data, HSFA had been originally used as a less specific feature extraction algorithm, and it was necessary to preserve tens or hundreds of features to achieve good performance with the supervised post-processing algorithm. The output features thus contained less label information, which was more spread within them.

HSFA has been enhanced in this work by means of additional heuristics and principles, which have resulted in different extensions, most notably HiGSFA. HGSFA and HiGSFA solve the supervised learning problems almost completely on their own by computing features that concentrate the label-predictive information: Generally, fewer than 10 features yielded best label estimation for the experiments considered. HiGSFA represents a new powerful and conceptually interesting tool for supervised feature extraction. The experimental results indicate that HiGSFA outperforms HGSFA in terms of label estimation accuracy and feature slowness independently of the training graph. In turn, ELL graphs outperform pre-defined graphs, and HGSFA outperforms HSFA.

The proposed extensions are supported by a solid theoretical foundation and enable new insights regarding HSFA for supervised learning. HiGSFA potentiates the advantages of HSFA allowing (1) the extraction of slower features, (2) smaller label estimation errors, (3) better generalization to new data, and (4) more accurate input reconstruction. Still, the computational and memory complexities are minimally increased, and they are of the same order as in HSFA. Besides the current advantages of HiGSFA, it can be understood well theoretically and technically, facilitating several possible improvements, and making it a promising algorithm for further research and applications.

Bibliography

Adali, T. and Haykin, S. *Adaptive Signal Processing: Next Generation Solutions*. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. John Wiley & Sons, 2010.

Alec Radford, S. C., Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. e-print arXiv:1511.06434, November 2015.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Berkes, P. Pattern recognition with Slow Feature Analysis. Cognitive Sciences EPrint Archive (CogPrints), February 2005a. URL `http://cogprints.org/4104/`.

Berkes, P. Handwritten digit recognition with nonlinear Fisher discriminant analysis. In *ICANN*, volume 3697 of *LNCS*, pages 285–287. Springer Berlin/Heidelberg, 2005b.

Berkes, P. and Wiskott, L. Slow Feature Analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5(6):579–602, 2005.

Blaschke, T., Zito, T., and Wiskott, L. Independent Slow Feature Analysis and nonlinear blind source separation. *Neural Computation*, 19(4):994–1021, 2007.

Böhmer, W., Grünewälder, S., Nickisch, H., and Obermayer, K. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1):67–86, 2012.

Bray, A. and Martinez, D. Kernel-based extraction of slow features: Complex cells learn disparity and translation invariance from natural images. In *NIPS*, volume 15, pages 253–260, 2003.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Choi, S. E., Lee, Y. J., Lee, S. J., Park, K. R., and Kim, J. Age estimation using a hierarchical classifier based on global and local facial features. *Pattern Recognition*, 44(6):1262–1281, 2011.

Ciresan, D., Meier, U., Masci, J., and Schmidhuber, J. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012. Selected Papers from IJCNN 2011.

Cootes, T. Face and gesture recognition research network (FG-NET) aging database, 2004. URL `http://www-prima.inrialpes.fr/FGnet/`.

Dähne, S., Höhne, J., Schreuder, M., and Tangermann, M. Slow Feature Analysis - a tool for extraction of discriminating event-related potentials in brain-computer interfaces. In *ICANN*, volume 6791 of *Lecture Notes in Computer Science*, pages 36–43. 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 248–255. IEEE, 2009.

Denton, E. L., Chintala, S., Szlam, A., and Fergus, R. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems 28*, pages 1486–1494. 2015.

Escalante-B., A. N. and Wiskott, L. Gender and age estimation from synthetic face images with Hierarchical Slow Feature Analysis. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Dortmund, Germany*, pages 240–249, 2010.

Escalante-B., A. N. and Wiskott, L. Heuristic evaluation of expansions for Non-Linear Hierarchical Slow Feature Analysis. In *Proc. of the 10th International Conference on Machine Learning and Applications, Honolulu, Hawaii, USA*, pages 133–138. IEEE Computer Society, 2011.

Escalante-B., A. N. and Wiskott, L. Slow Feature Analysis: Perspectives for technical applications of a versatile learning algorithm. *Künstliche Intelligenz [Artificial Intelligence]*, 26(4):341–348, 2012.

Escalante-B., A. N. and Wiskott, L. How to solve classification and regression problems on high-dimensional data with a supervised extension of Slow Feature Analysis. *Journal of Machine Learning Research*, 14:3683–3719, December 2013.

Escalante-B., A. N. and Wiskott, L. Improved graph-based SFA: Information preservation complements the slowness principle. e-print arXiv:1601.03945, 1 2016a.

Escalante-B., A. N. and Wiskott, L. Theoretical analysis of the optimal free responses of graph-based SFA for the design of training graphs. *Journal of Machine Learning Research*, 17(157):1–36, 8 2016b.

Fink, M., Fergus, R., and Angelova, A. Caltech 10,000 web faces, 2003. URL `http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/`.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

Földiák, P. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

Franzius, M., Sprekeler, H., and Wiskott, L. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3 (8):1605–1622, 2007.

Franzius, M., Wilbert, N., and Wiskott, L. Invariant object recognition and pose estimation with slow feature analysis. *Neural Computation*, 23(9):2289–2323, 2011.

Fu, Y., Guo, G., and Huang, T. S. Age synthesis and estimation via faces: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (11):1955–1976, 2010.

Gao, W., Cao, B., Shan, S., Chen, X., Zhou, D., Zhang, X., and Zhao, D. The CAS-PEAL large-scale Chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(1):149–161, 2008.

Geng, X., Zhou, Z.-H., and Smith-Miles, K. Automatic age estimation based on facial aging patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2234–2240, 2007.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

Guo, G. and Mu, G. Human age estimation: What is the influence across race and gender? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 71–78, June 2010.

Guo, G. and Mu, G. Simultaneous dimensionality reduction and human age estimation via kernel partial least squares regression. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 657–664, 2011.

Guo, G. and Mu, G. A framework for joint estimation of age, gender and ethnicity on a large database. *Image and Vision Computing*, 32(10):761–770, 2014. ISSN 0262-8856. Best of Automatic Face and Gesture Recognition 2013.

Guo, G., Mu, G., Fu, Y., Dyer, C., and Huang, T. A study on automatic age estimation using a large database. In *IEEE 12th International Conference on Computer Vision*, pages 1986–1991, Sept 2009a.

Guo, G., Mu, G., Fu, Y., and Huang, T. S. Human age estimation using bio-inspired features. In *CVPR*, pages 112–119, 2009b.

Han, H., Otto, C., and Jain, A. Age estimation from face images: Human vs. machine performance. In *International Conference on Biometrics (ICB)*, pages 1–8, June 2013.

He, X. *Locality Preserving Projections*. PhD thesis, Computer Science Department, The University of Chicago, Chicago, IL, USA, 2005.

He, X. and Niyogi, P. Locality Preserving Projections. In *Neural Information Processing Systems*, volume 16, pages 153–160, 2003.

Hinton, G. E. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3): 185–234, 1989.

Höfer, S., Hild, M., and Kubisch, M. Using Slow Feature Analysis to extract behavioural manifolds related to humanoid robot postures. In *Tenth International Conference on Epigenetic Robotics*, pages 43–50, 2010.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

Huerta, I., Fernández, C., Segura, C., Hernando, J., and Prati, A. A deep analysis on age estimation. *Pattern Recognition Letters*, 2015.

Jesorsky, O., Kirchberg, K. J., and Frischholz, R. Robust face detection using the Hausdorff distance. In *Proc. of Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95. Springer-Verlag, 2001.

Jia, S. and Cristianini, N. Learning to classify gender from four million images. *Pattern Recognition Letters*, 58:35–41, 2015. URL `http://dblp.uni-trier.de/db/journals/prl/prl58.html`.

Klampfl, S. and Maass, W. Replacing supervised classification learning by Slow Feature Analysis in spiking neural networks. In *Proc. of NIPS 2009: Advances in Neural Information Processing Systems*, volume 22, pages 988–996. MIT Press, 2010.

Koch, P., Konen, W., and Hein, K. Gesture recognition on few training data using slow feature analysis and parametric bootstrap. In *International Joint Conference on Neural Networks*, pages 1 –8, 2010.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

Krüger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J., Rodriguez-Sanchez, A., and Wiskott, L. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1847–1871, 2013.

Kuhnl, T., Kummert, F., and Fritsch, J. Monocular road segmentation using slow feature analysis. In *Intelligent Vehicles Symposium, IEEE*, pages 800–806, june 2011.

Kumar, N., Belhumeur, P. N., and Nayar, S. K. FaceTracer: A search engine for large collections of images with faces. In *European Conference on Computer Vision (ECCV)*, pages 340–353, 2008.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

Legenstein, R., Wilbert, N., and Wiskott, L. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Comput Biol*, 6(8):e1000894, 08 2010.

Luu, K., Ricanek, K., Bui, T., and Suen, C. Age estimation using active appearance models and support vector machine regression. In *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–5, Sept 2009.

Madrigal, L. and Kelly, W. Human skin-color sexual dimorphism: a test of the sexual selection hypothesis. *American journal of physical anthropology*, 132 (3):470–482, 2007.

Mitchison, G. Removing time variation with the anti-Hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.

Mohamed, N. M. and Mahdi, H. A simple evaluation of face detection algorithms using unpublished static images. In *10th International Conference on Intelligent Systems Design and Applications*, pages 1–5, 2010.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Ni, B., Song, Z., and Yan, S. Web image and video mining towards universal and robust age estimator. *Multimedia, IEEE Transactions on*, 13(6):1217–1229, 2011.

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. Overview of the face recognition grand

challenge. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 947–954. IEEE Computer Society, 2005.

Poloni, F. "Complexity of eigenvalue decomposition" (May 15, 2017). Message posted to https://mathoverflow.net/questions/62904/complexity-of-eigenvalue-decomposition, 2015.

Ramanathan, N., Chellappa, R., and Biswas, S. Age progression in human faces: A survey. *Journal of Visual Languages and Computing*, 15:3349–3361, 2009.

Rehn, E. M. On the slowness principle and learning in hierarchical temporal memory. Master's thesis, Bernstein Center for Computational Neuroscience, 2013.

Rehn, E. M. and Sprekeler, H. Nonlinear supervised locality preserving projections for visual pattern discrimination. In *International Conference on Pattern Recognition (ICPR)*, pages 1568–1573, 2014.

Ricanek Jr., K. and Tesafaye, T. Morph: A longitudinal image database of normal adult age-progression. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, FGR '06, pages 341–345. IEEE Computer Society, 2006.

Rish, I., Grabarnik, G., Cecchi, G., Pereira, F., and Gordon, G. J. Closed-form supervised dimensionality reduction with generalized linear models. In *Proc. of the 25th ICML*, pages 832–839. ACM, 2008.

Rubia, L. B. and Manimala, K. Slow feature analysis for recognizing prisoners activities to assist jail authorities. *International Journal of Advances in Engineering and Emerging Technology*, 1:1–6, 2013.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Schoenfeld, F. and Wiskott, L. Modeling place field activity with hierarchical slow feature analysis. *Frontiers in Computational Neuroscience*, 9(51), 2015.

Singular Inversions Inc. FaceGen SDK, 2008. `http://www.facegen.com`.

Sprekeler, H. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

Sprekeler, H. and Wiskott, L. A theory of slow feature analysis for transformation-based input signals with an application to complex cells. *Neural Computation*, 23(2):303–335, 2011.

Sprekeler, H., Zito, T., and Wiskott, L. An extension of slow feature analysis for nonlinear blind source separation. Cognitive Sciences EPrint Archive (CogPrints), 2010. URL `http://cogprints.org/7056/`.

Sprekeler, H., Zito, T., and Wiskott, L. An extension of Slow Feature Analysis for nonlinear blind source separation. *Journal of Machine Learning Research*, 15:921–947, 2014. URL `http://jmlr.org/papers/v15/sprekeler14a.html`.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

Sugiyama, M. Local Fisher discriminant analysis for supervised dimensionality reduction. In *Proc. of the 23rd ICML*, pages 905–912, 2006.

Sugiyama, M., Idé, T., Nakajima, S., and Sese, J. Semi-supervised local Fisher discriminant analysis for dimensionality reduction. *Machine Learning*, 78(1-2):35–61, 2010.

Sun, L., Jia, K., Chan, T., Fang, Y., Wang, G., and Yan, S. DL-SFA: deeply-learned slow feature analysis for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, pages 2625–2632, 2014.

Szeliski, R. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Tang, W. and Zhong, S. *Computational Methods of Feature Selection*, chapter Pairwise Constraints-Guided Dimensionality Reduction. Chapman and Hall/CRC, 2007.

Vollgraf, R. and Obermayer, K. Sparse optimization for second order kernel methods. In *International Joint Conference on Neural Networks*, pages 145–152, 2006.

Wang, K., Zhang, Z., and Wang, L. Violence video detection by discriminative slow feature analysis. In *Pattern Recognition - Chinese Conference, CCPR 2012. Proceedings*, pages 137–144, 2012.

Wilbert, N. *Hierarchical Slow Feature Analysis on visual stimuli and top-down reconstruction*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät I, 2012.

Wiskott, L. Learning invariance manifolds. In *Proc. of 5th Joint Symposium on Neural Computation, San Diego, CA, USA*, volume 8, pages 196–203. Univ. of California, 1998.

Wiskott, L. Slow Feature Analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003a.

Wiskott, L. Estimating driving forces of nonstationary time series with Slow Feature Analysis. arXiv.org e-Print archive, December 2003b. URL `http://arxiv.org/abs/cond-mat/0312317/`.

Wiskott, L. and Sejnowski, T. Slow Feature Analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

Xia, T., Tao, D., Mei, T., and Zhang, Y. Multiview spectral embedding. *Trans. Sys. Man Cyber. Part B*, 40(6):1438–1446, 2010.

Yi, D., Lei, Z., and Li, S. Age estimation by multi-scale convolutional network. In *Computer Vision – ACCV 2014*, volume 9005 of *Lecture Notes in Computer Science*, pages 144–158. 2015.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

Zhang, D., Zhou, Z.-H., and Chen, S. Semi-supervised dimensionality reduction. In *Proc. of the 7th SIAM International Conference on Data Mining*, 2007.

Zhang, T., Tao, D., Li, X., and Yang, J. Patch alignment for dimensionality reduction. *IEEE Transactions on Knowledge and Data Engineering*, 21(9): 1299–1313, 2009.

Zhang, Z. and Tao, D. Slow feature analysis for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):436 –450, 2012.

Zito, T., Wilbert, N., Wiskott, L., and Berkes, P. Modular toolkit for data processing (MDP): a python data processing framework. *Frontiers in Neuroinformatics*, 2(8), 2009.

Appendix A

Cuicuilco: A Framework for Hierarchical Processing

Several experiments that use SFA or its extensions have been carried out for this research. Such experiments were necessary to validate the proposed extensions and measure their accuracy and running time. The focus of this appendix is not on such experiments or their results (this has already been described in the main body of this thesis), but on the software framework that enabled me to design and execute the experiments, which is called Cuicuilco.

The Cuicuilco framework was implemented by me from scratch in Python for the purposes of this dissertation. It encompasses all experiments described in this dissertation, including all datasets and hierarchical networks. The pronunciation of Cuicuilco is available from `http://es.forvo.com/word/cuicuilco/` and sounds similar to 'ku:i-**ku:il**-ko'. This name has been chosen in honor of the Cuicuilco pyramid (800 B.C. to 250 A.D.) located in the south of Mexico City. This pyramid is divided in a few stages, resembling hierarchical SFA networks to some extent, as shown in Figure A.1.

The Python language was used because it allows fast prototyping, includes several open-source libraries, has been optimized for scientific computing, and has low or no cost. Another reason for using Python is continuity, since it is the primary programming language in the group of Prof. Dr. Wiskott.

Cuicuilco extensively uses the Modular Toolkit for Data Processing (MDP) (Zito et al., 2009). MDP is a machine learning library that provides implementations of several algorithms, as well as components useful to coordinate the interaction of these algorithms and construct hierarchical networks. Two well known algorithms included in MDP are PCA and SFA. These algorithms are available through the `PCANode` and `SFANode` classes, respectively. The word 'node' is used in the context of MDP to denote a class name of a particular algorithm or—depending on the context—an instance of it (all learning algorithms are derived from the `Node` class). Typewriter fonts are used in this appendix to denote class names, objects, functions, and environment variables.

Certain nodes do not implement any learning algorithm but are useful

Figure A.1: Cuicuilco pyramid in Mexico City. Image copyrighted by TJ DeGroat under the Creative Commons Attribution 2.0 Generic licence (CC BY 2.0). The image has been cropped and digitally processed for display purposes (saturation and color level adjustments).

to create hierarchical networks: Switchboards (`mdp.hinet.Switchboard`) are useful to rearrange and duplicate input dimensions. Layers encapsulate the idea of arranging various nodes in parallel, where each node has access to a contiguous subset of the input dimensions. There are two types of layers (`Layer` and `CloneLayer`). A `CloneLayer` clones the node given to the layer, an operation that is equivalent to weight sharing. A particular subclass of `mdp.hinet.Switchboard` is `mdp.hinet.Rectangular2dSwitchboard`, which allows the user of the framework to connect a layer to its input, where the nodes of the layer have rectangular receptive fields. Another way to organize nodes is to put them in series, one after the other. `Flow` objects implement this idea and provide a practical way of handling sequences of nodes, where the data is propagated in a feed-forward manner.

An SFA hierarchical network can be represented in MDP as a `Flow` composed of several `switchboard` and `Layer` objects. A single (abstract) SFA layer is then represented by one or more `Layer` objects of `SFA`/`PCA`/etc. nodes. In order to train the network, one can call the `train` method of the `Flow` object or train each layer/node separately.

The Cuicuilco framework is relatively large: over 30,000 lines of code and 17 files[1]. Thus, the code is not reproduced here but is available upon request, and I plan to also publish it online soon. This appendix describes only important high-level aspects of the framework and is a first step to a detailed understanding of it, which should be followed by reading the code and its documentation.

[1]This does not include 15,000+ lines of code and 50+ files used in small experiments and plots created for this research but independent of the framework.

The remainder of the appendix describes the steps performed by Cuicuilco upon execution and the command line options. Then, the organization of layers and networks in Cuicuilco is described and a few examples are given. Finally, the modules comprised by the framework are described.

A.1 A Single Run of Cuicuilco

In rough terms, each run of Cuicuilco includes the following steps:

1. File `experimental_datasets.py` is imported and a list of all available datasets is extracted from it.
2. File `hierarchical_networks.py` is imported and a list of all available network descriptions is extracted from it.
3. A particular network description and experimental dataset are selected.
4. If ELL is activated, an ELL graph is computed.
5. The training data of the selected dataset is loaded from disk.
6. The network description is used to construct a concrete network (an MDP flow object).
7. The network is trained using a special purpose training method. Such a method has linear complexity w.r.t. the number of nodes in the network, provides the nodes with the training-graph information, and uses a node cache to reuse previously trained nodes.
8. Network post-processing operations take place (the sign of the top-node weights are adjusted, a final whitening node may be appended to the flow).
9. The supervised data is loaded from disk.
10. Features are extracted from the training data and supervised data using the trained network.
11. All enabled supervised steps are trained using the supervised data (the features extracted by the network and ground truth labels and classes).
12. The test data is loaded from disk.
13. Features are extracted from test data using the trained network.
14. Label and class estimations are computed for the training, supervised, and test data.
15. Error measures are computed (e.g., RMSE, MAE, classification rates).
16. If the graphical display is enabled, several plots are created to visualize the datasets and results.

To evaluate an algorithm, usually several runs of Cuicuilco (i.e., trials of the experiment) should be executed to ensure statistical significance (most experiments have a random component). To simplify such a time consuming task, one

can automate the runs using external scripts (e.g., written in bash or Python) that execute Cuicuilco. One can change the behavior of the code without modifying it by exploiting environment variables and command line options, which are the topics of the next section.

A.2 Environment Variables and Command Line Options

Three integer environment variables must be specified before Cuicuilco is executed: (1) `CUICUILCO_TUNING_PARAMETER` is a tuning parameter that is typically used during the construction of the experimental datasets, for example, to indicate how strongly the images are distorted, (2) `CUICUILCO_EXPERIMENT_SEED` is the main random seed used to create the datasets and train the network, and (3) `CUICUILCO_IMAGE_LOADING_NUM_PROC` indicates the number of threads used to load the images of the training, supervised, and test data. One can execute different runs of an experiment after changing the seed value and/or the tuning parameter externally.

Cuicuilco accepts several command line options. One can specify them in bash as follows: `python cuicuilco_run.py [OPTION1] [OPTION2]...`. The most useful options are described as follows.

Basic Command Line Options. These options are frequently used and specify important information.

`--EnableDisplay={1/0}` Enables the graphical interface.

`--ExperimentalDataset={ParamsRAgeFunc/ParamsRTransXYScaleFunc/ ParamsMNISTFunc/...}` Selects a particular dataset by name.

`--HierarchicalNetwork={IEVMLRecNetworkU11L_Overlap6x6L0_1Label/ voidNetwork1L/PCANetwork1L/u08expoNetworkU11L/...}` Selects a particular network by name.

`--NumFeaturesSup=`N Specifies the number of output features N used in the supervised step.

`--SleepM=`M Specifies a delay before Cuicuilco starts loading the dataset. This is useful to prevent memory and processor clogging when several instances of Cuicuilco have been started. If $M > 0$, the current Cuicuilco process is paused for M minutes; if $M = 0$, there is no delay (default); and if $M < 0$, the program joins a waiting list, sleeps until its turn is reached, and deletes itself from the list after the labels/classes have been estimated. Such a waiting list is specified by a lock file named `queue_cuicuilco.txt`.

Cache Command Line Options. Cuicuilco includes various caches to speed up similar experiments: A network cache stores trained networks and a node cache stores trained nodes.

`--CacheAvailable={1/0}` Specifies whether any type of cache is available. If they are not available, the following cache options are ignored.

`--LoadNetworkNumber=`M Loads the Mth trained network stored in cache, so that training is skipped. However, if $M < 0$ a new network is constructed and trained from the beginning.
`--AskNetworkLoading={1/0}` If this option is enabled, Cuicuilco requests the user by means of the console to enter (via stdin) the number of the network to be loaded. This option has preference over the `LoadNetworkNumber` option.
`--NetworkCacheReadDir=`*directory* Specifies the directory used to load previously trained networks.
`--NetworkCacheWriteDir=`*directory* Specifies the directory used to save trained networks.
`--NodeCacheReadDir=`*directory* Specifies the directory used to search for nodes that might have been trained previously on the same data and parameters (this option can significantly speed up network training).
`--NodeCacheWriteDir=`*directory* Specifies the directory where trained nodes are saved.

Feature Extraction/Processing Options.
`--FeatureCutOffLevel=`f Trims the feature values to the real-valued interval $[-f, f]$.
`--AddNoiseToSeenid={1/0}` Adds noise to the data used to train the supervised step. The noise amplitude is hard-coded and minuscule.

Supervised Step Options. Cuicuilco executes various supervised learning algorithms on top of the features extracted by the network. Each supervised algorithm is executed independently of the others.
`--EnableLR={1/0}` Enables linear regression as a supervised step (ordinary least squares implemented as a pseudo-inverse).
`--EnableKNN{1/0}` Enables k-nearest neighbors (kNN) as a supervised step.
`--kNN_k=`k Sets the value of k, if kNN is enabled.
`--EnableNCC={1/0}` Enables a nearest centroid classifier as a supervised step.
`--EnableGC={1/0}` Enables a Gaussian classifier as a supervised step.
`--EnableSVM={1/0}` Enables a multi-class (one against one) support vector machine as a supervised step (requires libsvm).
`--SVM_gamma=`γ Sets the value of γ, if SVM is enabled (this parameter is used by the radial basis function (RBF) kernel).
`--SVM_C=`C Sets the value of C, if SVM is enabled.

Result-Saving Options. Cuicuilco allows saving some information. This is useful to use the output features in other software, to visualize the images after all pre-processing steps have been applied, and to observe images providing best and worst label-estimation errors.
`--SaveSubimagesTraining={1/0}` Saves (a fraction of) the training images to disk (after data distortions and other operations).
`--SaveAverageSubimageTraining={1/0}` Saves the average training image to

disk (after data distortion and other operations).
`--SaveSorted_AE_GaussNewid={1/0}` Saves (a fraction of) the training images to disk ordered by the absolute error of the label estimation.
`--ExportDataToLibsvm={1/0}` Saves the output features and labels in the format used by libsvm.

Explained Variance Options. Explained variance is defined in Cuicuilco as $ev \stackrel{\text{def}}{=} 1 - re$, where $re \stackrel{\text{def}}{=} \text{var}(\hat{\mathbf{y}} - \mathbf{y})/\text{var}(\mathbf{y})$ is a normalized reconstruction error and var() denotes variance. Thus, $ev \in (-\infty, 1]$, where chance level reconstruction $\hat{\mathbf{y}} = \bar{\mathbf{y}}$ has $ev = 0$, and perfect reconstruction $\hat{\mathbf{y}} = \mathbf{y}$ has $ev = 1$. Three different reconstruction methods are supported in Cuicuilco: (1) a global linear model, (2) kNN (k nearest neighbors), and (3) the network inverse, which uses the inverse or pseudo-inverse methods of each node. The following options allow the user to compute explained variances using each method.
`--EstimateExplainedVarWithInverse={1/0}` Reconstructions are computed using `flow.inverse()`.
`--EstimateExplainedVarWithKNN_k=`k If $k > 0$, reconstructions are given by the average of the k nearest neighbors.
`--EstimateExplainedVarLinGlobal_N=`N Reconstructions are given by a linear model trained with N samples chosen randomly from the training data. If $N = -1$, all training samples are used to build the model.

Label Estimation Options. These options affect the label computation.
`--MapDaysToYears={1/0}` Divides the ground-truth labels and label estimations by 365.242 (useful to change the label units from days to years).
`--IntegerLabelEstimation={1/0}` Truncates all label estimations to integer values.
`--CumulativeScores={1/0}` Computes cumulative scores for test data. See page 119 for a definition of cumulative scores.
`--ConfusionMatrix={1/0}` Computes the confusion matrix for test data.

Exact-Label-Learning Options.
`--GraphExactLabelLearning={1/0}` Computes an ELL graph based on the available labels. If GSFA is used (including HiGSFA and HGSFA), the resulting graph is given to the nodes during training.
`--NumberTargetLabels=`N Defines the number of target labels per original label: if $N > 1$, $N - 1$ auxiliary labels are created for each original label using Equation (127) on page 77.
`--OutputInsteadOfSVM2={1/0}` This is a hack useful to test the accuracy of the ELL method without using a supervised step. If this option is enabled, the first output of the network replaces the label estimation of the method called 'SVM2' (thus, the first output skips the supervised step and is evaluated as a label estimation that appears in the entry where usually the 'SVM2' results

appear).

A complete list of options can be obtained by using the `--help` option.

A.3 Network Structure in Cuicuilco

In Cuicuilco, a hierarchical network is described abstractly by a `ParamsNetwork` object. Such a `ParamsNetwork` object is transformed later by Cuicuilco into an MDP flow object by means of the `CreateNetwork()` function defined in the `network_builder` module. After this function has been executed, the resulting flow object can be trained as usual in MDP and data can be propagated through it. See Figure A.2 for a list of all modules of Cuicuilco and Section A.7 for their description.

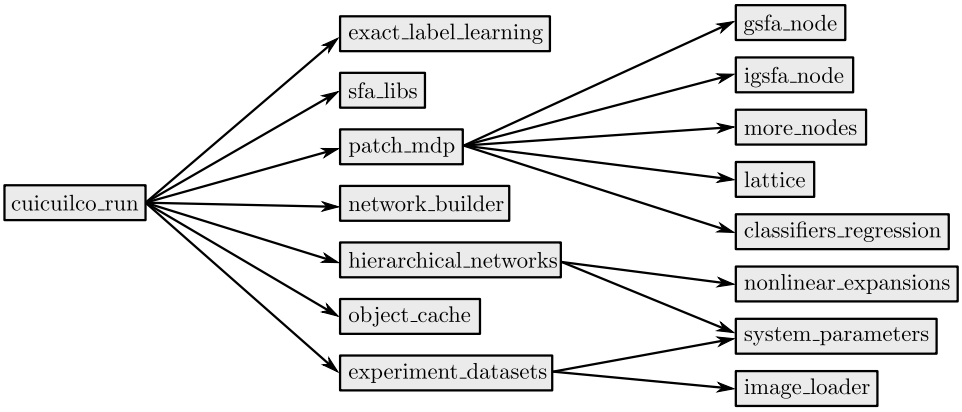


Figure A.2: Modules of Cuicuilco and their dependencies. The main module and program entry point is `cuicuilco_run`.

A `ParamsNetwork` object is quite simple; it consists mainly of a list of layers of nodes, which are accessible through the `ParamsNetwork.layers` field. Let `net` denote an object of type `ParamsNetwork`. The `k`-th layer is thus `net.layers[k]`, where `len(net.layers)` ≥ 1. For simplicity, one can also access the first 11 layers as `net.L0`, `net.L1`, ..., `net.L10`.

During network construction (i.e., execution of `CreateNetwork()`), each abstract layer of nodes may be transformed into several components: a switchboard and one or more objects of type MDP `Layer` or `CloneLayer`. Thus, the number of nodes in the resulting MDP flow is typically much larger than the number of abstract layers in Cuicuilco's network description. The structure of each layer is now addressed.

A.4 Structure of a Layer in Cuicuilco

A Cuicuilco layer is either of type `ParamsSFALayer` or `ParamsSFASuperNode`. Both types of layers are abstract (that is, they do not contain actual MDP nodes, but only a description of them). Objects of type `ParamsSFALayer` define at least one layer of nodes and at most six different layers of nodes. In contrast, objects of type `ParamsSFASuperNode` define from one to six different nodes. The classes of the MDP nodes used in these layers can be chosen at will. The reason for providing two different classes is that two types of layers are supported: hierarchical layers (specified by a `ParamsSFALayer` object) and non-hierarchical 'layers' (specified by a `ParamsSFASuperNode` object). As usual, these system classes are defined in `system_parameters.py`.

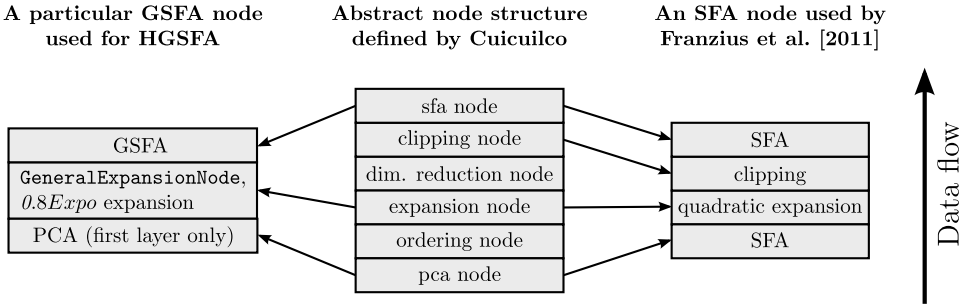


Figure A.3: Components of an abstract composite node in Cuicuilco (middle). These components provide great flexibility: Only a few of the available fields are usually necessary to represent the nodes of an arbitrary hierarchical network. A Cuicuilco Layer is composed of one or more of these abstract nodes. The figure also illustrates a typical GSFA node used by HGSFA (left) and a typical node used by Franzius et al. (2011). The arrows indicate which components of a Cuicuilco's node would be used to create such specific nodes.

The difference between these types of layers is that `ParamsSFALayer` layers are composed of one or more layers of MDP nodes and include information useful to create a switchboard needed to connect the nodes of the previous layer to the nodes of this layer, see Figure A.4. In contrast, `ParamsSFASuperNode` layers lack switchboard information and define at most six nodes connected in series. The first node processes all input dimensions, and the input to node $k+1$ is the output of node k. Thus, non-hierarchical layers are more similar to a simple list of nodes.

More concretely, hierarchical layers (`ParamsSFALayer`) define the configuration of the following elements: 1) a pseudo-invertible switchboard of type `PInvSwitchboard`, 2) a 'pca_node', 3) an 'ord_node' (an ordering node), 4) a `GeneralExpansionNode`, 5) a 'red_node' (a dimensionality reduction node), 6) a clipping function, and 7) an 'sfa_node'.

Only the switchboard and the 'sfa_node' are obligatory, the remaining nodes

```
class ParamsSFALayer(object):
  def __init__(self):
    self.name = "SFA Layer"
    #Switchboard data
    self.x_field_channels = 3
    self.y_field_channels = 3
    self.x_field_spacing = 3
    self.y_field_spacing = 3
    self.nx_value = None
    self.ny_value = None
    self.in_channel_dim = 1
    #PCA node
    self.pca_node_class = None
    self.pca_out_dim = 0.99999
    self.pca_args = {}

    #Ordering node
    self.ord_node_class = None
    self.ord_args = {}

    #General expansion node
    self.exp_funcs = None
    self.inv_use_hint = True
    self.inv_max_steady_factor=0.35
    self.inv_delta_factor = 0.6
    self.inv_min_delta = 0.0001

    #Dimensionality red. node
    self.red_node_class = None
    self.red_out_dim = 0.99999
    self.red_args = {}

    #Clip node
    self.clip_func = None
    self.clip_inv_func = None

    #SFA node
    self.sfa_node_class = None
    self.sfa_out_dim = 15
    self.sfa_args = {}

    #Other fields
    self.cloneLayer = True
    self.node_list = None
    self.layer_number = None
```

Figure A.4: Definition of the `ParamsSFALayer` class.

and functions can be omitted. The 'sfa_node' is assumed to carry out the main operation of the layer. Although their names might be misleading, 'pca_node', 'ord_node', 'red_node', and 'sfa_node' are placeholders and represent nodes of arbitrary classes. The concrete types of nodes used are defined through the following members of `ParamsSFALayer`: `pca_node_class`, `ord_node_class`, `red_node_class`, and `sfa_node_class`. Therefore, the 'sfa_node' is of type `sfa_node_class`, which could be set to `mdp.SFANode`, but it may also be set to `mdp.PCANode`, `GSFANode`, etc. If one desires to omit a node, one can set the corresponding class to `None`, e.g., `pca_node_class = None`. Furthermore, one can activate or deactivate weight sharing though the Boolean `cloneLayer` field, and one can set the output dimension of some nodes using the `pca_out_dim`, `red_out_dim`, and `sfa_out_dim` fields.

Non-hierarchical networks are defined mostly in the same way as hierarchical networks, except that they lack the fields related to the switchboard.

A.5 Examples of Network Definitions

This section shows some fragments of simplified code to exemplify the creation of two simple hierarchical networks using Cuicuilco. The construction of more complex networks works in the same way.

A.5.1 A Network that Implements the Identity Function

This section shows how to define a "network" that only has one non-hierarchical layer and implements the identity function. That is, this minimal network preserves the input unchanged. Strictly speaking it should not be called a network (since it only has one element), but it is included by extension. The layer can be defined as follows:

```
pVoidLayer = system_parameters.ParamsSFASuperNode()
pVoidLayer.pca_node_class = None
pVoidLayer.exp_funcs = [identity,]
pVoidLayer.sfa_node_class = mdp.nodes.IdentityNode
pVoidLayer.sfa_args = {}
pVoidLayer.sfa_out_dim = None
```

The definition of `pVoidLayer` does not specify any special value for the entries related to the clipping node, dim. reduction node, and ordering node, see Figure A.3. Default values ensure that these unspecified nodes are omitted. The network can then be defined in just three lines:

```
voidNetwork1L = system_parameters.ParamsNetwork()
voidNetwork1L.name = "Void 1 Layer Network"
voidNetwork1L.layers = [pVoidLayer]
```

Clearly, this minimal network is not too interesting, but one could easily activate a nonlinear expansion function (e.g., by setting `pVoidLayer.exp_funcs = [identity, QE]`), limit the output dimension (e.g., by setting `pVoidLayer.sfa_out_dim = 20`), and change the class of the 'sfa_node' (e.g., by setting `pVoidLayer.sfa_node_class = mdp.nodes.SFANode`, so that the main algorithm of the layer is SFA). Making these small changes would result in a node for nonlinear direct SFA.

A.5.2 A Simple 4-Layer HiGSFA Network

The following network is a simple HiGSFA network with 4 layers that can process 24×24 grayscale sub-images from the MNIST database (the original image resolution is 28×28-pixels, therefore, the usually black border of the images is ignored). The layers of this network are called `pSFALayerL0`, ..., `pSFALayerL3` and are defined as follows. Notice the use of `copy.deepcopy()` to reuse previously defined objects and keep the code compact:

```
##################### First layer    ############################
pSFALayerL0 = system_parameters.ParamsSFALayer()
pSFALayerL0.x_field_channels=3
pSFALayerL0.y_field_channels=3
pSFALayerL0.x_field_spacing=3
pSFALayerL0.y_field_spacing=3
pSFALayerL0.pca_node_class = mdp.nodes.PCANode
pSFALayerL0.pca_out_dim = 9
pSFALayerL0.pca_args = {}
pSFALayerL0.sfa_node_class = mdp.nodes.iGSFANode
pSFALayerL0.sfa_out_dim = 14
pSFALayerL0.sfa_args = {"pre_expansion_node_class":None,
   "expansion_funcs":[identity, unsigned_08expo], "max_comp":10,
   "max_num_samples_for_ev":None, "max_test_samples_for_ev":None,
   "offsetting_mode":"sensitivity_based_pure",
   "max_preserved_sfa":1.99}
pSFALayerL0.cloneLayer = False

##################### Second layer    ###########################
pSFALayerL1 = copy.deepcopy(pSFALayer0)
pSFALayerL1.x_field_channels=2
pSFALayerL1.y_field_channels=2
pSFALayerL1.x_field_spacing=2
pSFALayerL1.y_field_spacing=2
pSFALayerL1.sfa_out_dim = 40

##################### Third layer    ############################
pSFALayerL2 = copy.deepcopy(pSFALayer1) #Third layer
pSFALayerL2.sfa_out_dim = 60

##################### Fourth layer    ###########################
pSFALayerL3 = copy.deepcopy(pSFALayer1) #Fourth layer
pSFALayerL3.sfa_out_dim = 75
```

The 4-layer network can then be defined as follows:

```
MNISTNetwork_24x24_4L = system_parameters.ParamsNetwork()
MNISTNetwork_24x24_4L.name = "MNIST Network 4L 24x24"
MNISTNetwork_24x24_4L.layers = [pSFALayerL0,pSFALayerL1,
                                pSFALayerL2,pSFALayerL3]
```

After a network has been defined, one can easily use it in Cuicuilco by means of the `HierarchicalNetwork` command-line option (e.g., `python cuicuilco_run.py --HierarchicalNetwork=MNISTNetwork_24x24_4L --ExperimentalDataset=ParamsMNISTFunc --EnableDisplay=1`). This example

shows that Cuicuilco allows the user to create networks consisting of many layers and having a specialized structure (if needed) in a convenient way.

A.6 Definition of Experimental Datasets

The module `experimental_datasets` contains the definitions of datasets that can be loaded to perform experiments. An experimental dataset has class `ParamsSystem`. Each of them defines three datasets (training, supervised, and test), where each of these datasets is defined by two objects of classes `ParamsInput` and `ParamsDataLoading`, respectively, see Figure A.5. The `ParamsInput` object contains a high-level representation of the data and the configuration parameters that might be used to generate it, as well as ground-truth labels and classes. The `ParamsDataLoading` object contains the specific parameters needed to load the data as a numpy array. Thus, these parameters include file names, image resolution, patch size, patch position, and number of channels (e.g., 3 for RGB).

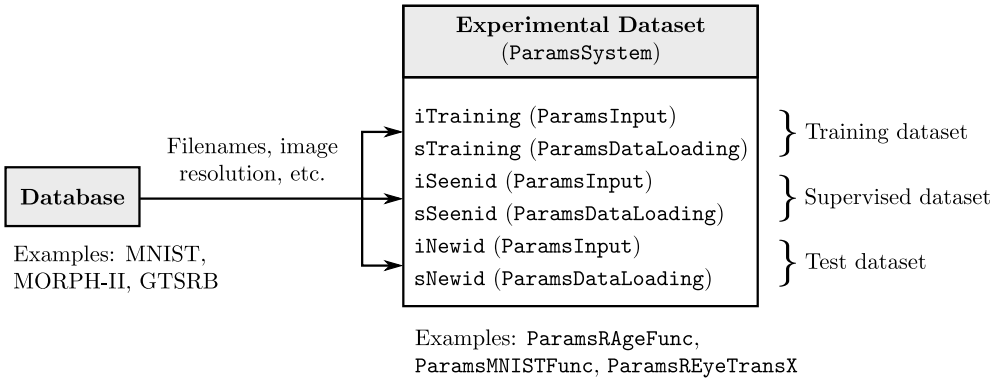


Figure A.5: Diagram of an experimental dataset described by a `ParamsSystem` object and the three datasets comprised by it. Class types are indicated in parenthesis.

The most important component of `ParamsDataLoading` objects is the `load_data()` function that is responsible for loading the data from disk and converting all the information encoded in the object into a numpy array. The details on how to implement this function are left free to the developer, but it is convenient to use the default implementation (i.e., `imageLoader.load_image_data()`) when the data samples are images.

Examples of experimental dataset objects are provided in the `experimental_datasets` module, including all definitions of the experiments carried out in this thesis. To select a particular dataset one can use the `ExperimentalDataset` command-line option.

A.7 Modules

This section is intended for future developers of the framework and describes the main source files (modules) comprised by Cuicuilco, including their main classes and methods. Figure A.2 shows the modules and their dependencies. Each module corresponds to a file with the same name and extension ".py". The modules are summarized in order of importance, as follows.

1) cuicuilco_run is the main file and entry point of the Cuicuilco framework. It implements the program flow described in Section A.1. Therefore, this module initiates the actual data loading, network training, computation of the results, and visualization.

2) system_parameters defines the fundamental classes of the framework: `ParamsNetwork`, `ParamsSFALayer`, and `ParamsSFASuperNode`, which are used to define hierarchical networks and their layers, and `ParamsSystem`, `ParamsInput`, and `ParamsDataLoading`, which are used to define experimental datasets. All network and dataset descriptions must instantiate the classes above.

3) experimental_datasets contains high-level descriptions of all experimental datasets. It can be modified by the user to define new experimental datasets. Experimental datasets are represented by an object of class `ParamsSystem` (see `system_parameters` module). `ParamsSystem` contains three separated datasets: one is used to train the network (training dataset), another is used to train the supervised steps (supervised dataset), and a third one is used for testing the whole system (test dataset), see Figure A.5. The data of all datasets is actually propagated through the network and the supervised steps. These datasets do not need to be different or disjoint, but they typically are.

Each dataset is specified by two objects of classes `ParamsDataLoading` and `ParamsInput`, respectively. `ParamsDataLoading` objects specify the low-level parameters needed to load the data (e.g., image filenames, patch/subimage sizes and their location within the image, image distortions, and the function that should be called for actually loading the data). `ParamsInput` objects contain abstract information about the data (e.g., the label and class information, as well as configuration parameters).

Examples of experimental datasets currently available are:

- `ParamsAge`, `ParamsAngle`, `ParamsGender`, and `ParamsIdentity`. These artificial datasets contain face images useful to learn age, vertical head angle, gender, and subject identity, respectively.
- `ParamsREyeTransX` and `ParamsREyeTransY`. Image patches of eyes taken from frontal face photographs useful to learn the horizontal and vertical position of the eye.

- `ParamsRTransXYScaleFunc`. Frontal face photographs extracted from several databases useful to learn x-pos, y-pos, and scale.
- `ParamsRAgeFunc`. Face photographs useful to learn age, gender and race from the MORPH-II and/or FG-Net image databases.
- `ParamsMNISTFunc`. Hand-written digits of the MNIST database.

4) hierarchical_networks contains high-level definitions of hierarchical networks as objects of type `ParamsNetwork`. `ParamsNetwork` objects have fields that specify the number of layers comprised by the network, what kind of nodes are used, what parameters are provided to the nodes during training, their output dimensionalities, nonlinearities used, etc. Examples of networks are:

- `voidNetwork1L`. A network that contains a single `IdentityNode`, thus, it implements the identity function.
- `linearPCANetworkU11L`. An 11-layer network that implements PCA hierarchically with excellent computational and memory efficiency.
- `u08expoNetworkU11L_5x5L0`. An 11-layer GSFA network, where the layers use the 0.8Exp expansion, and the receptive fields in the first layer are non-overlapping 5×5-pixel patches.
- `IEVMLRecNetworkU11L_Overlap6x6L0_1Label`. An 11 layer HiGSFA network optimized for age estimation with receptive fields of 6×6-pixels in the first layer, receptive field overlap, and specific optimized expansions in all layers.

5) gsfa_node defines the `GSFANode` (and related methods/classes). It allows efficient training of GSFA using several pre-defined graphs. The `GSFANode` uses a special class called `CovDCovMatrix` that allows the joint and optimized computation of covariance and second-moment matrices ($\mathbf{C}$ and $\dot{\mathbf{C}}$) when pre-defined graphs are used. The module also supports parallel training of the `GSFANode` whenever a *scheduler* is available (and provided as a training argument).

6) igsfa_node defines the `iGSFANode`. This node also implements the iSFA algorithm; one only needs to set the `training_mode` parameter to 'regular'.

7) more_nodes defines many nodes useful to perform general purpose experiments. Examples are:

- `HeadNode`. Selects the first k dimensions of the data.
- `PointwiseFunctionNode`. Useful to implement clipping and other transformations that are applied to each input dimension equally.
- `RandomizedMaskNode`. Removes specific dimensions of the data or replaces them by Gaussian noise.

- `RandomPermutationNode`. Randomly but consistently permutes the input dimensions.
- `SFAPCANode`. An extension to SFA inferior to iGSFA.
- `GeneralExpansionNode`. Implements general nonlinear expansions and is already included in MDP.
- `PInvSwitchboard`. A pseudo-invertible switchboard that supports receptive fields of arbitrary shape, as given by a mask, and centered at the points of a lattice).

Additionally, the module contains functions to estimate reconstruction errors using: (a) an inverse (for invertible nodes), (b) an approximated inverse using k-nearest neighbors, or (c) a linear model. Moreover, it provides functions to show a text description of the nodes comprised by a network, including their eigenvalues.

8) nonlinear_expansion implements over 500 different nonlinear transformations that can be used as expansion functions or to construct them. These transformations include the 0.8Exp expansion (`unsigned_08expo`), polynomial expansions, including the quadratic and cubic expansions (`QE` and `CE`), and different normalizations of polynomial expansions. Such normalized polynomial expansions may be more robust to outliers, and include the expansions `Q_AN_exp`, `Q_N_exp`, `Q_AE_exp`, and `Q_E_exp`, where '`A`' stands for asymmetric normalization, '`N`' for standard normalization, and '`E`' for exponential normalization:

$$\texttt{Q\_AN\_exp}(\mathbf{x}) \stackrel{\text{def}}{=} \texttt{QE}(\mathbf{x}^{\text{AN}})\,, \text{ where } x_i^{\text{AN}} \stackrel{\text{def}}{=} x_i/(1+|x_i|^{0.6})\,, \tag{148}$$

$$\texttt{Q\_N\_exp}(\mathbf{x}) \stackrel{\text{def}}{=} \texttt{QE}(\mathbf{x}^{\text{N}})\,, \text{ where } \mathbf{x}^{\text{N}} \stackrel{\text{def}}{=} \mathbf{x}/(1+||\mathbf{x}||_2^{0.6})\,, \tag{149}$$

$$\texttt{Q\_AE\_exp}(\mathbf{x}) \stackrel{\text{def}}{=} \texttt{QE}(\mathbf{x}^{\text{AE}})\,, \text{ where } x_i^{\text{AE}} \stackrel{\text{def}}{=} x_i/(1+\mathrm{e}^{0.6|x_i|})\,, \text{ and} \tag{150}$$

$$\texttt{Q\_E\_exp}(\mathbf{x}) \stackrel{\text{def}}{=} \texttt{QE}(\mathbf{x}^{\text{E}})\,, \text{ where } \mathbf{x}^{\text{E}} \stackrel{\text{def}}{=} \mathbf{x}/(1+\mathrm{e}^{0.6||\mathbf{x}||_2})\,. \tag{151}$$

9) exact_label_learning allows two tasks: (1) The computation of an ELL graph given the target labels, including methods useful to normalize and decorrelate the target labels and eliminate negative edges weights of a given graph. (2) The computation of optimal free responses of an arbitrary training graph. Additionally, various helper functions are provided, for example, to compute the values of R and Q, to explicitly compute the edge-weight matrices of pre-defined training graphs, and to test if a graph is consistent.

10) image_loader contains functions useful to load the input data from disk and pre-process them, mostly grayscale and RGB images. The main data loading function is `load_image_data`, which loads the images in parallel (multi-threading) and applies common image distortions to them

(translations, rotations, scalings, basic contrast enhancement, and additive noise). The number of threads must be specified through the environment variable `CUICUILCO_IMAGE_LOADING_NUM_PROC`. Any dataset can provide its own function to load data (by overriding the `load_data` method), but if the data samples are images, using the provided `load_image_data()` function can be convenient.

11) patch_mdp does all low level operations needed to extend MDP and make it compatible with the new features introduced by Cuicuilco. It injects the Cuicuilco nodes into MDP at run time, adds a `local_inverse` method to selected nodes (from a given output vector and an input vector, this function approximates an inverse to the output vector that is close in input space to the input vector and close in feature space to the output vector), and adds a field `list_training_params` to those nodes that allow parameters during training. The layer nodes (`Layer` and `CloneLayer`) are modified to propagate training parameters. The `Flow` class is improved, for example by providing a function `special_train_cache_scheduler_sets` that replaces the basic `train` function by supporting training parameters, per node parallelism through an explicit scheduler, and a cache mechanism at the node level to eliminate the training procedure whenever a node has been previously trained using the same data and parameters.

12) network_builder translates a high-level network description provided by a `ParamsNetwork` object (such as the networks declared in `hierarchical_networks.py`), into an MDP flow object.

13) classifiers_regressions contains functions useful to work with a Gaussian classifier (`GaussianClassifier`), such as the computation of soft labels (see Section 3.4.5) and helper functions to compute classification rates and a mean average error.

14) sfa_libs declares various basic helper functions, such as `cutoff` (a cutoff function for matrices), `cartesian_product` (computes the Cartesian product of two sets), `select_rows_from_matrix` (an operation used mostly by the switchboards), `remove_Nones` (eliminates entries equal to `None` from a list), and functions useful to compute Δ values.

15) object_cache defines a `Cache` class useful for saving objects (e.g., numpy arrays and MDP flows) to disk and restoring them. Arrays are split into several physical files if they are too large (which may otherwise result in problems in some systems). It also provides functions for computing hash values of objects (e.g., numpy arrays) in a fast and reliable way (but not collision resistant, as understood in cryptography). This module is based on code generously shared by Dr. Niko Wilbert.

16) lattice contains functions to compute the points of a lattice on the plane. This module is used by a pseudo-invertible switchboard called `PInvSwitchboard` defined in `more_nodes`.

About the Author

Author information as of January 2017.

Personal Data

Name	Alberto Nicolás Escalante Bañuelos
Place of birth	Durango Dgo., Mexico
Email	alberto.escalante@ini.rub.de, alberto.nicolas.escalante@gmail.com

Education and Professional Experience

4/2009–	**Research assistant/PhD candidate** Institut für Neuroinformatik Ruhr-University Bochum, Germany
10/2004–5/2008	**M.Sc. in Computer Science**, grade 1.1 Computer Science Department University of Saarland, Saarbrücken, Germany
11/2005–5/2008	**Visiting student** (master thesis) Chair of System Security Fakultät für Elektrotechnik und Informationstechnik Ruhr-University Bochum, Germany
9/2003–9/2004	**University lecturer.** Subject 'discrete structures' School of Engineering National Autonomous Univ. of Mexico (UNAM), Mexico
7/2001–1/2002	**Internship** (programmer and assistant) Applied Mathematics and Systems Research Institute (IIMAS), UNAM, Mexico
9/1997–8/2003	**B.E. in Computer Engineering** with honors, grade 9.8/10.0 School of Engineering, UNAM, Mexico

Publications

Journal Publications

Escalante-B., A. N. and Wiskott, L. Theoretical analysis of the optimal free responses of graph-based SFA for the design of training graphs. *Journal of Machine Learning Research*, 17(157):1–36, 8 2016b

Escalante-B., A. N. and Wiskott, L. How to solve classification and regression problems on high-dimensional data with a supervised extension of Slow Feature Analysis. *Journal of Machine Learning Research*, 14:3683–3719, December 2013

Escalante-B., A. N. and Wiskott, L. Slow Feature Analysis: Perspectives for technical applications of a versatile learning algorithm. *Künstliche Intelligenz [Artificial Intelligence]*, 26(4):341–348, 2012

Work in Progress

Escalante-B., A. N. and Wiskott, L. Improved graph-based SFA: Information preservation complements the slowness principle. e-print arXiv:1601.03945, 1 2016a

Conference Publications

Escalante-B., A. N. and Wiskott, L. Heuristic evaluation of expansions for Non-Linear Hierarchical Slow Feature Analysis. In *Proc. of the 10th International Conference on Machine Learning and Applications, Honolulu, Hawaii, USA*, pages 133–138. IEEE Computer Society, 2011

Escalante-B., A. N. and Wiskott, L. Gender and age estimation from synthetic face images with Hierarchical Slow Feature Analysis. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Dortmund, Germany*, pages 240–249, 2010

Previous Publications

Armknecht, F., Escalante B., A. N., Löhr, H., Manulis, M., and Sadeghi, A.-R. Secure multi-coupons for federated environments: Privacy-preserving and customer-friendly. In *Information Security Practice and Experience: 4th International Conference, Sydney, Australia*, pages 29–44, 2008

Escalante-B., A. N., Löhr, H., and Sadeghi, A.-R. A non-sequential unsplittable privacy-protecting multi-coupon scheme. In *INFORMATIK 2007: Informatik trifft Logistik. Band 2. Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, pages 184–188, Bremen, Germany, 2007

Chen, L., Escalante-B., A. N., Löhr, H., Manulis, M., and Sadeghi, A.-R. A privacy-protecting multi-coupon scheme with stronger protection against splitting. In *Financial Cryptography and Data Security: 11th International Conference, FC 2007, Scarborough, Trinidad and Tobago*, pages 29–44, 2007

Previous Theses

Escalante-B., A. N. Privacy-protecting multi-coupon schemes with stronger protection against splitting. Master's thesis, Computer Science Department, University of Saarland, Saarbrücken, Germany, 2007

Escalante-B., A. N. The AES encryption algorithm: Its low level implementation and optimization to improve the security mechanisms of a smart card (in Spanish). Bachelor's thesis, School of Engineering, National Autonomous University of Mexico (UNAM), 2003