Journal of Machine Learning Research 14 (2013) 3683-3719 Submitted 3/13; Revised 10/13; Published 12/13

How to Solve Classification and Regression Problems on High-Dimensional Data with a Supervised Extension of Slow Feature Analysis

Alberto N. Escalante-B. ALBERTO.ESCALANTE@INI.RUB.DE
Laurenz Wiskott LAURENZ.WISKOTT@INI.RUB.DE
Institut für Neuroinformatik
Ruhr-Universität Bochum
Bochum D-44801, Germany

Editor: David Dunson

Abstract

Supervised learning from high-dimensional data, for example, multimedia data, is a challenging task. We propose an extension of slow feature analysis (SFA) for *supervised* dimensionality reduction called graph-based SFA (GSFA). The algorithm extracts a label-predictive low-dimensional set of features that can be post-processed by typical supervised algorithms to generate the final label or class estimation. GSFA is trained with a so-called training graph, in which the vertices are the samples and the edges represent similarities of the corresponding labels. A new weighted SFA optimization problem is introduced, generalizing the notion of slowness from sequences of samples to such training graphs. We show that GSFA computes an optimal solution to this problem in the considered function space and propose several types of 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 were evaluated experimentally with a subproblem of face detection on photographs. The method proposed is promising particularly when linear models are insufficient as well as when feature selection is difficult.

Keywords: slow feature analysis, feature extraction, classification, regression, pattern recognition, training graphs, nonlinear dimensionality reduction, supervised learning, implicitly supervised, high-dimensional data, image analysis

1. Introduction

Supervised learning from high-dimensional data has important applications in areas such as multimedia processing, human-computer interfaces, industrial quality control, speech processing, robotics, bioinformatics, image understanding, and medicine. Despite constant improvements in computational resources and learning algorithms, supervised processing, for example, for regression or classification, of high-dimensional data is still a challenge largely due to insufficient data and several phenomena referred to as the curse of dimensionality. This limits the practical applicability of supervised learning.

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, this approach is inherently suboptimal.

Supervised dimensionality reduction is more appropriate in this case. Its goal is to compute a low-dimensional set of features from the high-dimensional input samples that contains 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.

Slow feature analysis (SFA) (Wiskott, 1998; Wiskott and Sejnowski, 2002) is an unsupervised learning algorithm inspired by the visual system and based on the slowness principle. SFA has been applied to classification in various ways. Franzius et al. (2008) 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 was 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 was relatively small, making identity a feature that changes slowly. This result confirms that SFA is capable of extracting categorical information. Klampfl and Maass (2010) introduced a particular Markov chain to generate a sequence used to train SFA for classification. The transition probability between samples from different object identities was proportional to a small parameter a. The authors showed that in the limit $a \to 0$ (i.e., only intra-class transitions), the features learned by SFA are equivalent to the features learned by Fisher discriminant analysis (FDA). The equivalence of the discrimination capability of SFA and FDA in some setups was already known (compare Berkes, 2005a, and Berkes, 2005b) but had not been rigorously shown before. In the two papers by Berkes, hand-written digits from the MNIST database were recognized. Several mini-sequences of two samples from the same digit were used to train SFA. The same approach was 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) proposed an elaborate system for human action recognition, in which the difference between delta values of different training signals was amplified and used for discrimination.

SFA has been used to solve regression problems as well. Franzius et al. (2008) used standard SFA to learn the position of animated fish from images with homogeneous background. The same training sequence used for learning fish identities was used, thus the fish position changed continuously over time. However, a different supervised post-processing step was employed consisting of linear regression coupled with a nonlinear transformation.

In this article, we introduce a supervised extension of SFA called graph-based SFA (GSFA) specifically designed for supervised dimensionality reduction. We show that GSFA computes the slowest features possible according to the GSFA optimization problem, a weighted extension of the

SFA problem, generalizing the concept of signal slowness. The objective function of the GSFA 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 at least in 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) was conceived from the beginning as a nonlinear algorithm without resorting to kernels (although there exist versions 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 leaves room for better optimization towards the labels.

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 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 node-weights, and uses function spaces.

There is also a strong connection between GSFA and LPP. In Section 7.1 we show 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 (Sprekeler, 2011; Rehn, 2013), being basically LPP on nonlinearly expanded data, is also closely connected to GSFA.

One advantage of GSFA over many 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.

Given a large number of high-dimensional labeled data, supervised learning algorithms can often not be applied due to prohibitive computational requirements. In such cases we propose the following general scheme based on GSFA/SFA, illustrated in Figure 1 (left):

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 are highly predictive of the labels. Hierarchical processing (Section 2.4) is an efficient divide-and-conquer approach for high-dimensional data with SFA and GSFA.

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 data set 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, node weights specifying *a priori* sample probabilities, and edge weights indicating desired output similarities, as derived from the labels. Details are given in Section 3. 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 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 1 (right), 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 $O(N)$ in the number of samples N and $O(I^3)$ in the number of dimensions I (possibly after a nonlinear expansion). Hierarchical processing greatly reduces the latter complexity down to $O(I)$. 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.

In the next sections, we first recall the standard SFA optimization problem and algorithm. Then, we introduce the GSFA optimization problem, which incorporates the information contained in a training graph, and propose the GSFA algorithm, which solves this optimization problem. We recall how classification problems have been addressed with SFA and propose a training graph for doing this task with GSFA. Afterwards, we propose various graph structures for regression problems offering great computational efficiency and good accuracy. Thereafter, we experimentally evaluate and compare the performance of four training graphs to other common supervised methods (e.g., PCA+SVM) w.r.t. a particular regression problem closely related to face detection using real photographs. A discussion section concludes the article.

2. Standard SFA

In this section, we begin by introducing the slowness principle, which has inspired SFA. Afterwards, we recall the SFA optimization problem and the algorithm itself. We conclude the section with a brief introduction to hierarchical processing with SFA.

2.1 The Slowness Principle and SFA

Perception plays a crucial role in the interaction of animals or humans with their environment. Although processing of sensory information appears to be done straightforwardly by the nervous

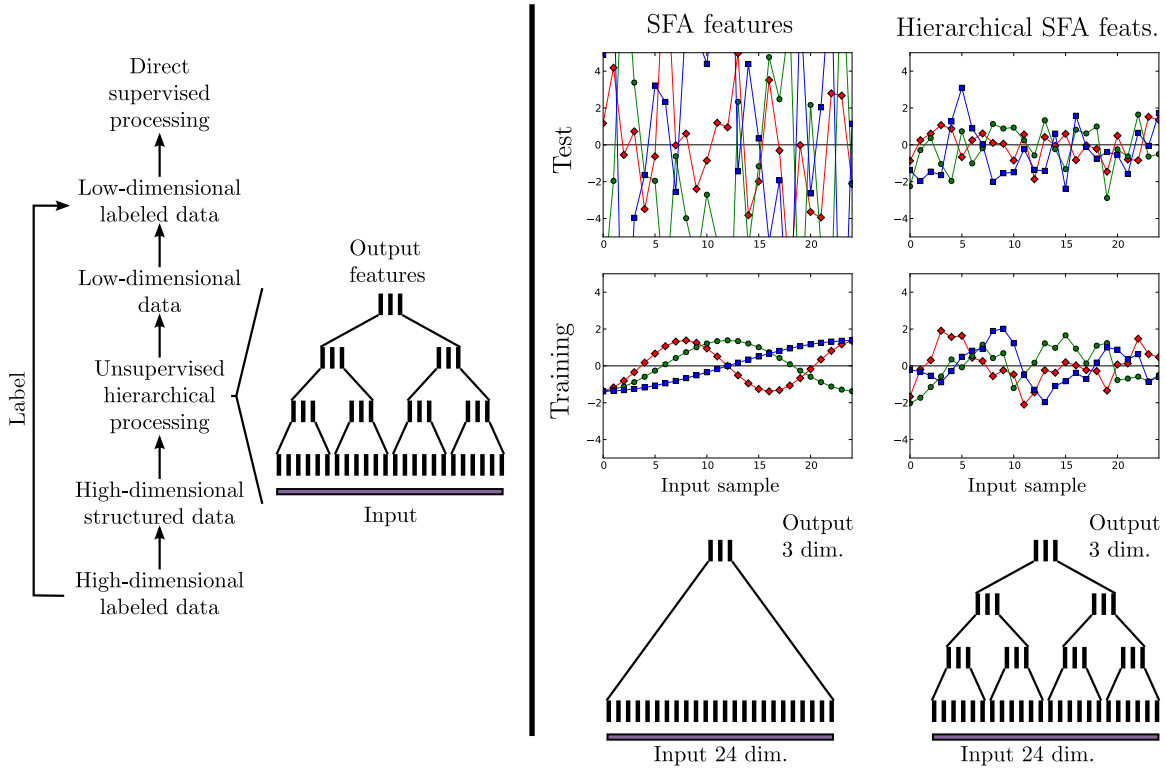


Figure 1: (Left) Transformation of a supervised learning problem on high-dimensional data into a supervised learning problem on low-dimensional data by means of unsupervised hierarchical processing on structured data, that is, without labels. This construction allows the solution of supervised learning problems on high-dimensional data when the dimensionality and number of samples make the direct application of many conventional supervised algorithms infeasible. (Right) Example of how hierarchical SFA (HSFA) is more robust against overfitting than standard 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 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. Several points even fall outside the plotted area. 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. Counterintuitively, this result holds even though the HSFA network used has $7 \times 6 \times 3 = 126$ free parameters, many more than the $24 \times 3 = 72$ free parameters of direct SFA.

system, it is a complex computational task. As an example, consider the visual perception of a driver watching pedestrians walking on the street. As the car advances, his receptor responses

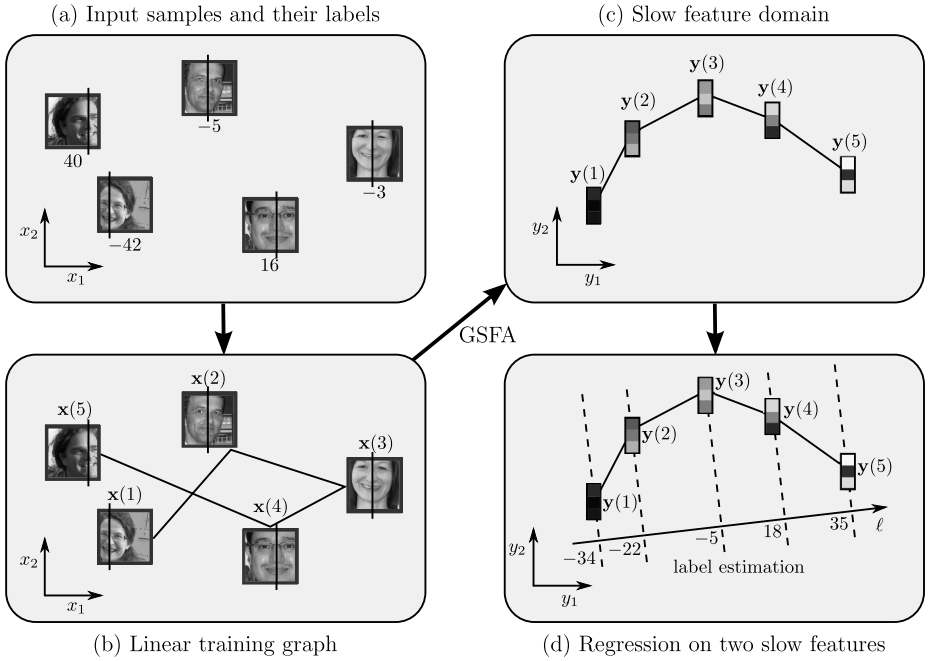


Figure 2: Illustration of the application of GSFA to solve a regression problem. (a) The input samples are 128×128-pixel images with labels indicating the horizontal position of the center of the face. (b) A training graph is constructed using the label information. In this example, only images with 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.

typically change quite quickly, and are especially sensitive to the eye movement and to variations in the position or pose of the pedestrians. However, a lot of information, including the position and identity of the pedestrians, can still be distinguished. Relevant abstract information derived from the perception of the environment typically changes on a time scale much slower than the individual sensory inputs. 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 were developed shortly after 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 also permits an extended mathematical treatment so that its properties are well understood analytically (Wiskott, 2003; Franzius et al., 2007; Sprekeler and Wiskott, 2011). SFA has the advantage that it is guaranteed to find an optimal solution within the considered function space. It was initially developed for learning invariances in a model of the primate visual system

(Wiskott and Sejnowski, 2002; Franzius et al., 2011). Berkes and Wiskott (2005) subsequently used it for learning complex-cell receptive fields and Franzius et al. (2007) for place cells in the hippocampus. Recently, researchers have begun using SFA for various technical applications (see Escalante-B. and Wiskott, 2012, for a review).

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 $t \in \mathbb{R}$, 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

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

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) assure that the output signals are normalized, not constant, and represent different features of the input signal. The problem can be solved iteratively beginning with y_1 (the slowest feature extracted) and finishing with y_J (an algorithm 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 3 for an illustrative example.

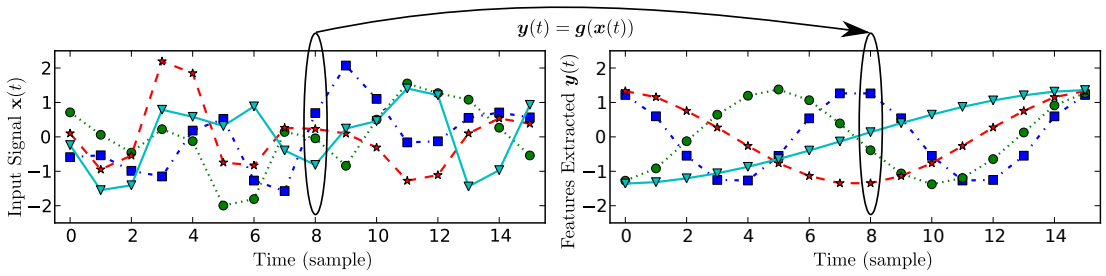


Figure 3: Illustrative example of feature extraction from a 10-dimensional (discrete time) input signal. Four arbitrary components of the input (left) and the four slowest outputs (right) are shown. Notice that feature extraction is an instantaneous operation, even though the outputs are slow over time. This example was designed such that the features extracted are the slowest ones theoretically possible.

In practice, the function $\mathbf{g}$ is usually restricted to a finite-dimensional space $\mathcal{F}$, for example, to all quadratic or linear functions. 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 3 (right) even when the hidden parameters of the input data lack such a precise structure. The problem is then evident when one extracts unstructured features from test data, see Figure 1 (right). An unrestricted function space is, however, useful for various theoretical analyses (e.g., Wiskott, 2003) because of its generality and mathematical convenience.

2.3 Standard Linear SFA Algorithm

The SFA algorithm is typically nonlinear. Even though 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 in the expanded space. In this section, we recall the standard linear SFA algorithm (Wiskott and Sejnowski, 2002), 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. Also the objective function and the constraints are 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 $1 \leq t \leq N-1$.

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, so that the output has zero-mean to conform with (2). Thus, in the linear case, the SFA problem reduces to finding an optimal set of weight vectors $\{\mathbf{w}_j\}$ under the constraints above, and it can be solved by linear algebra methods, see below.

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 $\langle \dot{\mathbf{x}}\dot{\mathbf{x}}^T \rangle_t$ 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 $\mathbf{S}$. Afterwards, the J directions of least variance in the derivative signal $\dot{\mathbf{z}}$ are found and represented by an $I \times J$ rotation matrix $\mathbf{R}$, such that $\mathbf{R}^T\dot{\mathbf{C}}_z\mathbf{R} = \mathbf{\Lambda}$, 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 $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_J$. 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 features extracted $\mathbf{y} = \mathbf{W}^T(\mathbf{x} - \bar{\mathbf{x}})$, and $\Delta(y_j) = \lambda_j$, for $1 \leq j \leq J$. The linear SFA algorithm is guaranteed to find an optimal solution to the optimization problem (1–4) in the linear function space, for example, 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 described above is $O(NI^2 + I^3)$ 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.[1] In practice, it has a speed comparable to PCA, even though SFA also takes into account the temporal structure of the data.

1. 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

To reduce the complexity of SFA, a divide-and-conquer strategy to extract slow features is usually effective (e.g., Franzius et al., 2011). For instance, one can spatially divide the data into lower-dimensional blocks of dimension $I' \ll I$ and extract $J' < I'$ local slow features separately with different instances of SFA, the so-called SFA nodes. Then, one uses another SFA node in a next layer to extract global slow features from the local slow features. Since each SFA node performs dimensionality reduction, the input dimension of the top SFA node is much less than I. This strategy can be repeated iteratively until the input dimensionality at each node is small enough, resulting in a multi-layer hierarchical network. Due to information loss before the top node, this does not guarantee optimal global 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 1 (right), 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).

3. Graph-Based SFA (GSFA)

In this section, we first present a generalized representation of the training data used by SFA called training graph. Afterwards, we propose the GSFA optimization problem, which is defined in terms of the nodes, edges and weights of such a graph. Then, we present the GSFA algorithm and a probabilistic model for the generation of training data, connecting SFA and GSFA.

3.1 Organization of the Training Samples in a Graph

Learning from a single (multi-dimensional) 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 need 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 these images in a sequence of increasing age, they would form a linear structure that could be used for training much like a regular time series.

The central contribution of this work 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 4) with a set $\mathbf{V}$ of vertices $\mathbf{x}(n)$ (each vertex/node 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 node and edge weights are identical (Figure 4.b). How exactly edge weights are derived from label values will be elaborated later.

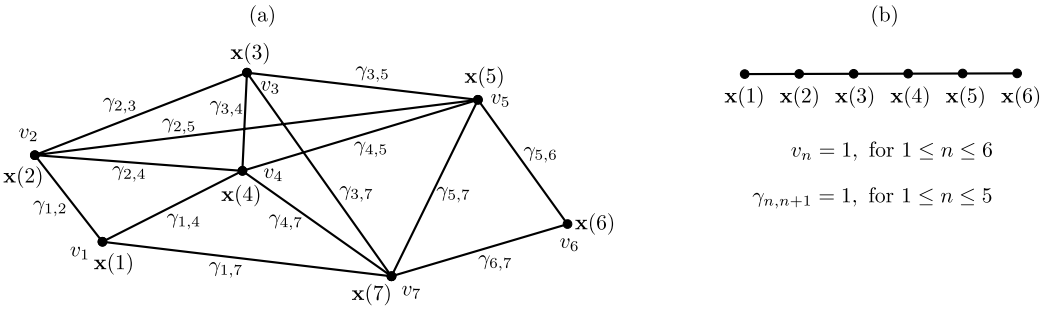


Figure 4: (a) Example of a training graph with $N = 7$ vertices. (b) A regular sample sequence (time-series) represented as a linear graph suitable for GSFA.

3.2 GSFA Optimization Problem

We extend the concept of slowness, originally conceived for sequences of samples, to data structured in a training graph making use of its associated edge weights. The generalized optimization problem can then be formalized as follows. For $1 \le j \le J$, find features $y_j(n)$, where $1 \le n \le N$, such that the objective function

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

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, 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. 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 node weights, and serve a normalization purpose.

By definition (see Section 3.1), 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. We 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.

3.3 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 we propose an extension that computes an optimal solution to the GSFA problem within the same space. 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 the set of edges $(\mathbf{x}(n), \mathbf{x}(n'))$ with edge weights $\gamma_{n,n'}$. To simplify notation we introduce zero edge weights $\gamma_{n,n'} = 0$ for non-existing edges $(\mathbf{x}(n), \mathbf{x}(n')) \notin \mathbf{E}$. The linear GSFA algorithm differs from the standard version only in the computation of the matrices $\mathbf{C}$ and $\dot{\mathbf{C}}$, which now take into account the neighbourhood structure (samples, edges, and weights) specified by the training graph.

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

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

where

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

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

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

Given these matrices, the computation of $\mathbf{W}$ is the same 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 $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_J$. Finally the algorithm returns $\Delta(y_1), \ldots, \Delta(y_J)$, $\mathbf{W}$ and $\mathbf{y}(n)$, where

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

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

3.4 Correctness of the Graph-Based SFA Algorithm

We now prove that the GSFA algorithm indeed solves the optimization problem (6–9). This proof is similar to the optimality proof of the standard SFA algorithm (Wiskott and Sejnowski, 2002). 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

$$\begin{aligned}\sum_n v_n \mathbf{y}(n) &\overset{(18)}{=} \sum_n v_n \mathbf{W}^T (\mathbf{x}(n) - \hat{\mathbf{x}}) \\ &= \mathbf{W}^T \left(\sum_n v_n \mathbf{x}(n) - \sum_{n'} v_{n'} \hat{\mathbf{x}} \right) \\ &\overset{(13,11)}{=} \mathbf{W}^T (Q\hat{\mathbf{x}} - Q\hat{\mathbf{x}}) = \mathbf{0}.\end{aligned}$$

We also find

$$\begin{aligned}\mathbf{I} &= \mathbf{R}^T \mathbf{I} \mathbf{R} \quad \text{(since } \mathbf{R} \text{ is a rotation matrix)}, \\ &\overset{(15)}{=} \mathbf{R}^T (\mathbf{S}^T \mathbf{C_G} \mathbf{S}) \mathbf{R}, \\ &\overset{(17)}{=} \mathbf{W}^T \mathbf{C_G} \mathbf{W}, \\ &\overset{(12)}{=} \mathbf{W}^T \frac{1}{Q} \sum_n v_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T \mathbf{W}, \\ &\overset{(18)}{=} \frac{1}{Q} \sum_n v_n \mathbf{y}(n) (\mathbf{y}(n))^T,\end{aligned}$$

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

Now, let us consider the objective function

$$\begin{aligned}\Delta_j &\overset{(6)}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} \big(y_j(n') - y_j(n)\big)^2 \\ &\overset{(14)}{=} \mathbf{w}_j^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{w}_j \\ &\overset{(17)}{=} \mathbf{r}_j^T \mathbf{S}^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{S} \mathbf{r}_j \\ &\overset{(16)}{=} \lambda_j,\end{aligned}$$

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).

3.5 Probabilistic Interpretation of Training Graphs

In this section, we give an intuition for 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, we construct below 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 fulfil the following *normalization restrictions*:

$$\sum_n \tilde{v}_n \;=\; 1\,, \tag{19}$$

$$\sum_{n'} \tilde{\gamma}_{n,n'}/\tilde{v}_n \;=\; 1 \quad \forall n\,, \tag{20}$$

$$\overset{(5)}{\Longleftrightarrow} \quad \sum_{n'} \tilde{\gamma}_{n',n}/\tilde{v}_n \;=\; 1 \quad \forall n\,, \tag{21}$$

$$\sum_{n,n'} \tilde{\gamma}_{n,n'} \overset{(20,19)}{=} 1\,. \tag{22}$$

Restrictions (19) and (22) can always be assumed without loss of generality, because they can be achieved by a constant scaling of the weights (i.e., $\tilde{v}_n \leftarrow \tilde{v}_n/Q$, $\tilde{\gamma}_{n,n'} \leftarrow \tilde{\gamma}_{n,n'}/R$) without affecting the outputs generated by GSFA. Restriction (20) 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

$$\pi_n = \boldsymbol{p}_1(n) \overset{\text{def}}{=} \Pr(\mathbf{Z}_1 = \mathbf{x}(n)) \overset{\text{def}}{=} \tilde{v}_n\,, \tag{23}$$

and the transition probabilities are given by

$$P_{nn'} \overset{\text{def}}{=} \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n') | \mathbf{Z}_t = \mathbf{x}(n)) \;\overset{\text{def}}{=}\; (1-\varepsilon)\tilde{\gamma}_{n,n'}/\tilde{v}_n + \varepsilon\tilde{v}_{n'} \underset{\lim_{\varepsilon\to 0}}{=} \tilde{\gamma}_{n,n'}/\tilde{v}_n\,, \tag{24}$$

$$\overset{(23)}{\Longrightarrow} \quad \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n'), \mathbf{Z}_t = \mathbf{x}(n)) \;=\; (1-\varepsilon)\tilde{\gamma}_{n,n'} + \varepsilon\tilde{v}_n\tilde{v}_{n'} \underset{\lim_{\varepsilon\to 0}}{=} \tilde{\gamma}_{n,n'}\,, \tag{25}$$

(for $\mathbf{Z}_t$ stationary) with $0 < \varepsilon \ll 1$. 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 (19), (20), and (22) ensure the normalization of (23), (24), and (25), respectively.

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

$$\begin{aligned}
\mathbf{p}_{t+1}(n) \;=\; \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n)) \;&=\; \sum_{n'} \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n) | \mathbf{Z}_t = \mathbf{x}(n')) \Pr(\mathbf{Z}_t = \mathbf{x}(n')) \\
&\overset{(23,24)}{=} \sum_{n'} \left((1-\varepsilon)\left(\tilde{\gamma}_{n',n}/\tilde{v}_{n'}\right) + \varepsilon\tilde{v}_n \right) \tilde{v}_{n'} \\
&\overset{(21,19)}{=} (1-\varepsilon)\tilde{v}_n + \varepsilon\tilde{v}_n \;=\; \tilde{v}_n \;=\; \mathbf{p}_t(n)\,.
\end{aligned} \tag{26}$$

The time average of the input sequence is

$$\begin{aligned}\boldsymbol{\mu}_Z &\stackrel{\text{def}}{=} \langle \mathbf{Z}_t \rangle_t \\ &= \langle \mathbf{Z} \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \\ &\stackrel{(26)}{=} \sum_n \tilde{v}_n \mathbf{x}(n) \\ &\stackrel{(13)}{=} \hat{\mathbf{x}}, \end{aligned} \tag{27}$$

and the covariance matrix is

$$\begin{aligned}\mathbf{C} &\stackrel{\text{def}}{=} \langle (\mathbf{Z}_t - \boldsymbol{\mu}_Z)(\mathbf{Z}_t - \boldsymbol{\mu}_Z)^T \rangle_t \\ &\stackrel{(27)}{=} \langle (\mathbf{Z} - \hat{\mathbf{x}})(\mathbf{Z} - \hat{\mathbf{x}})^T \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \\ &\stackrel{(26)}{=} \sum_n \tilde{v}_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T \\ &\stackrel{(12)}{=} \mathbf{C}_{\mathbf{G}}, \end{aligned}$$

whereas the derivative covariance matrix is

$$\begin{aligned}\dot{\mathbf{C}} &\stackrel{\text{def}}{=} \langle \dot{\mathbf{Z}}_t \dot{\mathbf{Z}}_t^T \rangle_t \\ &= \langle \dot{\mathbf{Z}} \dot{\mathbf{Z}}^T \rangle_{\boldsymbol{\pi}} \quad \text{(since } \mathcal{M} \text{ is ergodic)} \\ &\stackrel{(25)}{=} \sum_{n,n'} \left((1-\varepsilon)\tilde{\gamma}_{n,n'} + \varepsilon \tilde{v}_n \tilde{v}_{n'} \right) (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T, \end{aligned} \tag{28}$$

where $\dot{\mathbf{Z}}_t \stackrel{\text{def}}{=} \mathbf{Z}_{t+1} - \mathbf{Z}_t$. Notice that $\lim_{\varepsilon \to 0} \dot{\mathbf{C}} \stackrel{(28)}{=} \tilde{\gamma}_{n,n'} (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T \stackrel{(14)}{=} \dot{\mathbf{C}}_{\mathbf{G}}$. Therefore, if a graph fulfils the normalization restrictions (19)–(22), GSFA yields the same features as standard SFA on the sequence generated by the Markov chain, in the limit $\varepsilon \to 0$.

3.6 Construction of Training Graphs

One can, in principle, construct training graphs with arbitrary connections and weights. However, when the goal is to solve a supervised learning task, the graph created should implicitly integrate the label information. An appropriate structure of the training graphs depends on whether the goal is classification or regression. In the next sections, we describe each case separately. We have previously implemented the proposed training graphs, and we have tested and verified their usefulness on real-world data (Escalante-B. and Wiskott, 2010; Mohamed and Mahdi, 2010; Stallkamp et al., 2011; Escalante-B. and Wiskott, 2012).

4. Classification with SFA

In this section, we show how to use GSFA to profit from the label information and solve classification tasks more efficiently and accurately than with standard SFA.

4.1 Clustered Training Graph

To generate features useful for classification, we propose the use of a *clustered training graph* presented below (Figure 5). Assume there are S identities/classes, and for each particular identity $s = 1,\ldots,S$ there are N_s samples $\mathbf{x}^s(n)$, where $n = 1,\ldots,N_s$, making a total of $N = \sum_s N_s$ samples. We define the clustered training graph as a graph $G = (\mathbf{V},\mathbf{E})$ with vertices $\mathbf{V} = \{\mathbf{x}^s(n)\}$, and edges $\mathbf{E} = \{(\mathbf{x}^s(n),\mathbf{x}^s(n'))\}$ for $s = 1,\ldots,S$, and $n,n' = 1,\ldots,N_s$. Thus all pairs of samples of the same identity are connected, while samples of different identity are not connected. Node weights are identical and equal to one, that is, $\forall s,n:\ v_n^s = 1$. In contrast, edge weights, $\gamma^s_{n,n'} = 1/N_s \quad \forall n,n'$, depend on the cluster size.[2] Otherwise identities with a large N_s would be over-represented because the number of edges in the complete subgraph for identity s grows quadratically with N_s. These weights directly fulfil the normalization restriction (20). As usual, a trivial scaling of the node and edge weights suffices to fulfil restrictions (19) and (22), allowing the probabilistic interpretation of the graph. The optimization problem associated to this graph explicitly demands that samples from the same object identity should be typically mapped to similar outputs.

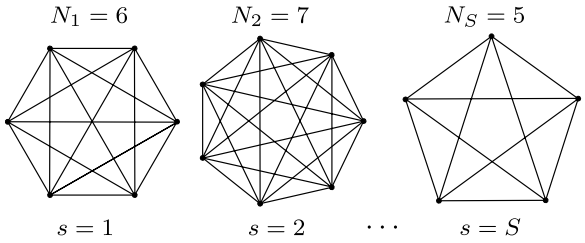


Figure 5: Illustration of a *clustered* training graph used for a classification problem. All samples belonging to the same object identity form fully connected subgraphs. Thus, for S identities there are S complete subgraphs. Self-loops not shown.

4.2 Efficient Learning Using the Clustered Training Graph

At first sight, the large number of edges, $\sum_s N_s(N_s+1)/2$, seems to introduce a computational burden. Here we show that this is not the case if one exploits the symmetry of the clustered training graph. From (12), the sample covariance matrix of this graph using the node weights $v_n^s = 1$ is (notice the definition of Π^s and $\hat{\mathbf{x}}^s$):

$$\mathbf{C}_{\text{clus}} \overset{(12)}{=} \frac{1}{Q}\left(\underbrace{\sum_s \sum_{n=1}^{N_s} \mathbf{x}^s(n)(\mathbf{x}^s(n))^T}_{\overset{\text{def}}{=}\Pi^s} - Q\Big(\underbrace{\frac{1}{Q}\sum_s \overbrace{\sum_{n=1}^{N_s} \mathbf{x}^s(n)}^{\overset{\text{def}}{=}N_s\hat{\mathbf{x}}^s}}_{\overset{(13)}{=}\hat{\mathbf{x}}} \Big)\Big(\frac{1}{Q}\sum_s \sum_{n=1}^{N_s} \mathbf{x}^s(n) \Big)^T \right), \tag{29}$$

$$= \frac{1}{Q}\Big(\sum_s \Pi^s - Q\hat{\mathbf{x}}(\hat{\mathbf{x}})^T \Big), \tag{30}$$

2. These node 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 $v_n^s = 1/N_s$ and $\forall n,n':\ \gamma^s_{n,n'} = (1/N_s)^2$ instead.

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

From (14), the derivative covariance matrix of the clustered training graph using edge weights $\gamma^s_{n,n'} = 1/N_s$ is:

$$\dot{\mathbf{C}}_{\text{clus}} \overset{(14)}{=} \frac{1}{R} \sum_s \frac{1}{N_s} \sum_{n,n'=1}^{N_s} (\mathbf{x}^s(n') - \mathbf{x}^s(n))(\mathbf{x}^s(n') - \mathbf{x}^s(n))^T , \tag{31}$$

$$= \frac{1}{R} \sum_s \frac{1}{N_s} \sum_{n,n'=1}^{N_s} \left(\mathbf{x}^s(n')(\mathbf{x}^s(n'))^T + \mathbf{x}^s(n)(\mathbf{x}^s(n))^T - \mathbf{x}^s(n')(\mathbf{x}^s(n))^T - \mathbf{x}^s(n)(\mathbf{x}^s(n'))^T \right),$$

$$\overset{(29)}{=} \frac{1}{R} \sum_s \frac{1}{N_s} \left(N_s \sum_{n=1}^{N_s} \mathbf{x}^s(n)(\mathbf{x}^s(n))^T + N_s \sum_{n'=1}^{N_s} \mathbf{x}^s(n')(\mathbf{x}^s(n'))^T - 2N_s \hat{\mathbf{x}}^s (N_s \hat{\mathbf{x}}^s)^T \right),$$

$$\overset{(29)}{=} \frac{2}{R} \sum_s \left(\Pi^s - N_s \hat{\mathbf{x}}^s (\hat{\mathbf{x}}^s)^T \right), \tag{32}$$

where $R \overset{(10)}{=} \sum_s \sum_{n,n'} \gamma^s_{n,n'} = \sum_s \sum_{n,n'} 1/N_s = \sum_s (N_s)^2/N_s = \sum_s N_s = N$.

The complexity of computing $\mathbf{C}_{\text{clus}}$ using (29) or (30) is the same, namely $O(\sum_s N_s)$ (vector) operations. However, the complexity of computing $\dot{\mathbf{C}}_{\text{clus}}$ can be reduced from $O(\sum_s N_s^2)$ operations directly using (31) to $O(\sum_s N_s)$ operations using (32). This algebraic simplification allows us to compute $\dot{\mathbf{C}}_{\text{clus}}$ with a complexity linear in N (and N_s), which constitutes an important speedup since, depending on the application, N_s might be larger than 100 and sometimes even $N_s > 1000$.

Interestingly, one can show that the features learned by GSFA on this graph are equivalent to those learned by FDA (see Section 7).

4.3 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 $S-1$ 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 $S-1$ slow features or less. We suggest the use of a Gaussian classifier because in practice we have obtained better robustness when enough training data is available. While a more powerful classification method, such as an SVM, might also be used, we have found only a small increase in performance at the cost of longer training times.

5. Regression with SFA

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, $\hat{\ell}$, and their ground-truth values, ℓ. We assume here that the loss function is an increasing function of $|\hat{\ell} - \ell|$ (e.g., contrary to periodic functions useful to compare angular values, or arbitrary functions of $\hat{\ell}$ and ℓ).

Regression problems can be address 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, we propose four SFA-based methods that explicitly use available labels. 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* (Sections 5.1–5.4). The selection of the explicit regression step for post-processing is discussed in Section 5.5.

5.1 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 data is 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 6).

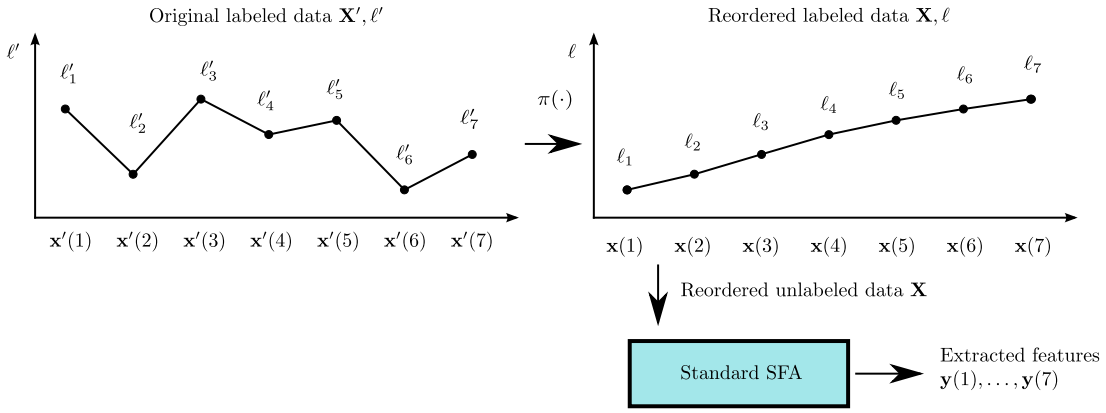


Figure 6: Sample reordering approach. Standard SFA is trained with a reordered sample sequence, in which the hidden labels are increasing.

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 also fulfils the normalization conditions). Clearly, SFA could only extract this information if the samples indeed intrinsically 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 $O(N)$ operations. Since this method only requires standard SFA and is the most straightforward to implement, we recommend its use for first experiments. If more robust outputs are desired, the methods below based on GSFA are more appropriate.

5.2 Sliding Window Training Graph

This is an improvement over the method 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 7.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. For the experiments in this article, we employ a *mirrored* sliding window with edge weights

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

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

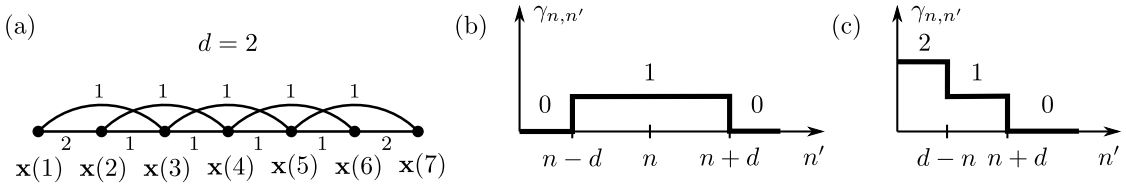


Figure 7: (a) A mirrored square sliding window training graph with a half-width of $d = 2$. Each vertex is thus adjacent to at most 4 other vertices. (b) Illustration of the edge weights of an intermediate node $\mathbf{x}(n)$ for an arbitrary window half-width d. (c) Edge weights for a node $\mathbf{x}(n)$ close to the left extreme ($n < d$). Notice that the sum of the edge weights is also approximately $2d$ for extreme nodes.

GSFA is guaranteed to find functions that minimize (6) within the function space considered. However, whether such a solution is suitable for regression largely depends on how one has defined the weights of the training graph. For instance, if there is a sample with a large node 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. This 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, this is a good solution in terms of the optimization problem but not a good one to solve the regression problem at hand, because the samples with small values are hard to discriminate. We refer to such solutions 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. A necessary condition is that the graph is connected because, as discussed in Section 4, for disconnected graphs GSFA has a strong tendency to produce a representation suitable for classification rather than regression. After various experiments, we have found useful to enforce the normalization restriction (20) at least approximately (after node and edge weights have been normalized). This ensures that the samples are connected sufficiently strongly to the other ones, relative to their own node weight. Of course, one should not resort to self-loops $\gamma_{n,n} \neq 0$ to trivially fulfil the restriction.

The improved continuity of the features appears to also benefit performance after the supervised step. This is the reason why we make the node weights of the first and last groups of samples in the serial training graph weaker, the intra-group connections of the first and last groups of samples in the mixed graph stronger, and introduced mirroring for the square sliding window graph.

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

5.3 Serial Training Graph

The *serial* training graph is similar to the clustered training graph used for classification in terms of construction and efficiency. It results from discretizing the original labels $\boldsymbol{\ell}$ into a relatively small set of discrete labels of size L, namely $\{\ell_1, \ldots, \ell_L\}$, 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 this 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 neighbouring group). The only existing connections are inter-group connections, no intra-group connections are present.

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, we assume 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 we use the same edge weight for all connections: $\forall n, n', l:\ \gamma_{n,n'}^{l,l+1} = 1$. 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 8). The reason for the different weights in the first and last groups is to improve feature quality by enforcing the normalization restriction (20) (after node and edge weight normalization). Notice that since any two vertices of the same group are adjacent to exactly the same neighbours, they are likely to be mapped to similar outputs by GSFA.

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. Unsurprisingly, the structure of the graph can be exploited to train GSFA efficiently. Similarly to the clustered training graph, 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

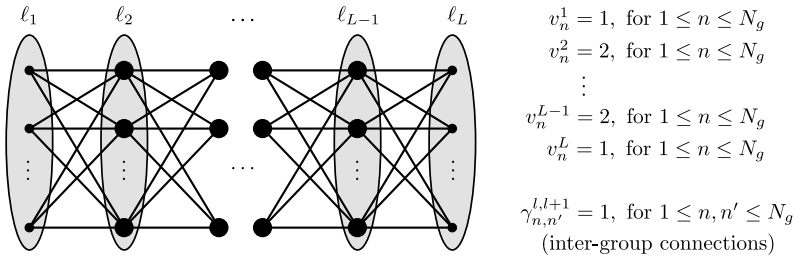


Figure 8: Illustration of a serial training graph with L discrete labels. Even though the original labels of two samples might differ, they will be grouped together if they have the same discrete label. In the figure, a bigger node represents a sample with a larger weight, and the ovals represent the groups.

as:

$$\hat{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{Q}\sum_n \left(\mathbf{x}^1(n) + \mathbf{x}^L(n) + 2\sum_{l=2}^{L-1}\mathbf{x}^l(n)\right) = \frac{1}{2(L-1)}\left(\hat{\mathbf{x}}^1 + \hat{\mathbf{x}}^L + 2\sum_{l=2}^{L-1}\hat{\mathbf{x}}^l\right). \tag{33}$$

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

$$\begin{aligned}\mathbf{C}_{\text{ser}} &\stackrel{(12,33)}{=} \frac{1}{Q}\left(\sum_n \mathbf{x}^1(n)(\mathbf{x}^1(n))^T + 2\sum_{l=2}^{L-1}\sum_n \mathbf{x}^l(n)(\mathbf{x}^l(n))^T + \sum_n \mathbf{x}^L(n)(\mathbf{x}^L(n))^T - Q\hat{\mathbf{x}}(\hat{\mathbf{x}})^T\right)\\ &= \frac{1}{Q}\left(\Pi^1 + \Pi^L + 2\sum_{l=2}^{L-1}\Pi^l - Q\hat{\mathbf{x}}'(\hat{\mathbf{x}}')^T\right).\end{aligned}$$

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

$$\dot{\mathbf{C}}_{\text{ser}} \stackrel{(14)}{=} \frac{1}{R}\sum_{l=1}^{L-1}\sum_{n,n'}(\mathbf{x}^{l+1}(n') - \mathbf{x}^l(n))(\mathbf{x}^{l+1}(n') - \mathbf{x}^l(n))^T \tag{34}$$

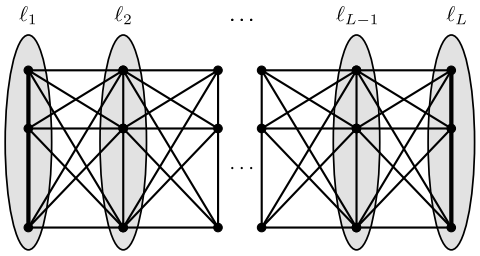
$$\begin{aligned}&= \frac{1}{R}\sum_{l=1}^{L-1}\sum_{n,n'}\left(\mathbf{x}^{l+1}(n')(\mathbf{x}^{l+1}(n'))^T + \mathbf{x}^l(n)(\mathbf{x}^l(n))^T - \mathbf{x}^l(n)(\mathbf{x}^{l+1}(n'))^T - \mathbf{x}^{l+1}(n')(\mathbf{x}^l(n))^T\right)\\ &= \frac{1}{R}\sum_{l=1}^{L-1}\left(\sum_{n'}\left(\Pi^{l+1} + \Pi^l\right) - \left(\sum_n \mathbf{x}^l(n)\right)\left(\sum_{n'}\mathbf{x}^{l+1}(n')\right)^T - \left(\sum_{n'}\mathbf{x}^{l+1}(n')\right)\left(\sum_n \mathbf{x}^l(n)\right)^T\right)\\ &= \frac{N_g}{R}\sum_{l=1}^{L-1}\left(\Pi^{l+1} + \Pi^l - N_g\hat{\mathbf{x}}^l(\hat{\mathbf{x}}^{l+1})^T - N_g\hat{\mathbf{x}}^{l+1}(\hat{\mathbf{x}}^l)^T\right).\end{aligned} \tag{35}$$

By using (35) instead of (34), 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 $O(L(N_g)^2)$ to only $O(N)$ 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, we obtain 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 samples).

5.4 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. 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. However, in some cases, particularly for small numbers of training samples, additional intra-group connections might indeed improve robustness. We thus conceived the *mixed* training graph (Figure 9), which is a combination of the serial and clustered training graph and fulfils the consistency restriction (20). In the mixed training graph, all nodes and edges 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).



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

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

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

$$(1 \le n, n' \le N_g)$$

Figure 9: Illustration of the mixed training graph. Samples having the same label are fully connected (intra-group connections, represented with vertical edges) and 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 as represented by thick lines.

5.5 Supervised Step for Regression Problems

There are at least three approaches to implement the supervised step on top of 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

$$\ell \stackrel{\text{def}}{=} \sum_{l=1}^{\tilde{L}} \tilde{\ell}_l \cdot \mathbf{P}(C_{\tilde{\ell}_l}|\mathbf{y}) \tag{36}$$

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, such as the Mean Average Error (MAE), can be addressed in a similar way.

We have tested these three approaches in combination with supervised algorithms such as linear regression, and classifiers such as nearest neighbour, nearest centroid, Gaussian classifier, and SVMs. We recommend using the soft labels computed from the class probabilities estimated by a Gaussian classifier because in most of our experiments this method has provided best performance and robustness. Of course, other classifiers providing class probabilities could also be used.

6. Experimental Evaluation of the Methods Proposed

In this section, we evaluate the performance of the supervised learning methods based on SFA presented above. We consider two concrete image analysis problems using real photograph databases, the first one for classification and the second one for regression.

6.1 Classification

For classification, we have proposed the clustered training graph. As mentioned in Section 4.2, when this graph is used, the outputs of GSFA are equivalent to those of FDA. Since FDA has been used and evaluated exhaustively, here we only verify that our implementation of GSFA generates the expected results when trained with such a graph.

The German Traffic Sign Recognition Benchmark (Stallkamp et al., 2011) was chosen for the experimental test. This was a competition with 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 was necessary because the true position of the signs was included as annotations, making this a pure classification task and ideal for our test. We participated in the online version of the competition, where 26,640 labeled images were provided for training and 12,569 images without label for evaluation (classification rate was computed by the organisers, 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 the experiments done here, a compact expansion that only doubles the data dimension was employed, $\mathbf{x}^T \mapsto \mathbf{x}^T, (|\mathbf{x}|^{0.8})^T$, where the absolute value and exponent 0.8 are computed component-wise. We refer to this expansion as $0.8Exp$. Previously, Escalante-B. and Wiskott (2011) 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.

Our method, complemented by a Gaussian classifier on 42 slow features, achieved a recognition rate of 96.4% on test data.[3] This, as expected, was similar to the reported performance of various methods based on FDA participating in the same competition. For comparison, human performance was 98.81%, and a convolutional neural network gave top performance with a 98.98% recognition rate.

6.2 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 we proposed previously (see Mohamed and Mahdi, 2010). In our system, face detection is decomposed 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, we explain this regression problem, the algorithms evaluated, and the results in more detail.

6.2.1 PROBLEM AND DATA SET DESCRIPTION

To increase image variability and improve generalization, face images from several databases were 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.), 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 were automatically pre-processed through a pose-normalization and a pose-reintroduction step. In the first step, each image was converted to greyscale and pose-normalized using annotated facial points so that the face is centered,[4] has a fixed eye-mouth-triangle area, and the resulting pose-normalized image has a resolution of 256×192 pixels. In the second step, horizontal and vertical displacements were 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 were 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) was done using bicubic interpolation. At last, the images were cropped to 128×128 pixels.

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.

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

4. 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 **LE**, **RE** and **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.

Figure 10: 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).

The pose-normalized images were randomly split in three data sets of $30,000$, $20,000$ and $9,000$ images. The first data set 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 data set were duplicated, resulting in two pose-reintroduced images per input image, that is, a single input image exclusively belongs to one of the three data sets, appearing twice in it with two different poses. Hence, the final size of the data sets is $60,000$, $40,000$ and $18,000$ pre-processed images, respectively.

6.2.2 DIMENSIONALITY-REDUCTION METHODS EVALUATED

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. We circumvent this 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 without distinction). The nodes in the HSFA networks first expand the data using the $0.8Exp$ expansion function (see Section 6.1) and then apply SFA/GSFA to it, except for the nodes in the first layer in which additionally PCA is applied before the expansion preserving 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 1. In contrast to other works (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 were used because in previous experiments with similar data they showed good performance at a smaller computational cost.

The following dimensionality-reduction methods were 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).

Layer	size	node fan-in	output dim. per HSFA node	output dim. per HPCA node
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 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) with such a shape, located in the preceding layer.

- 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 we made a small selection thereof. We chose HPCA 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, we believe that HPCA (combined with various supervised algorithms) is a fair point of comparison, and a good representative among 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 11.

6.2.3 SUPERVISED POST-PROCESSING ALGORITHMS CONSIDERED

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

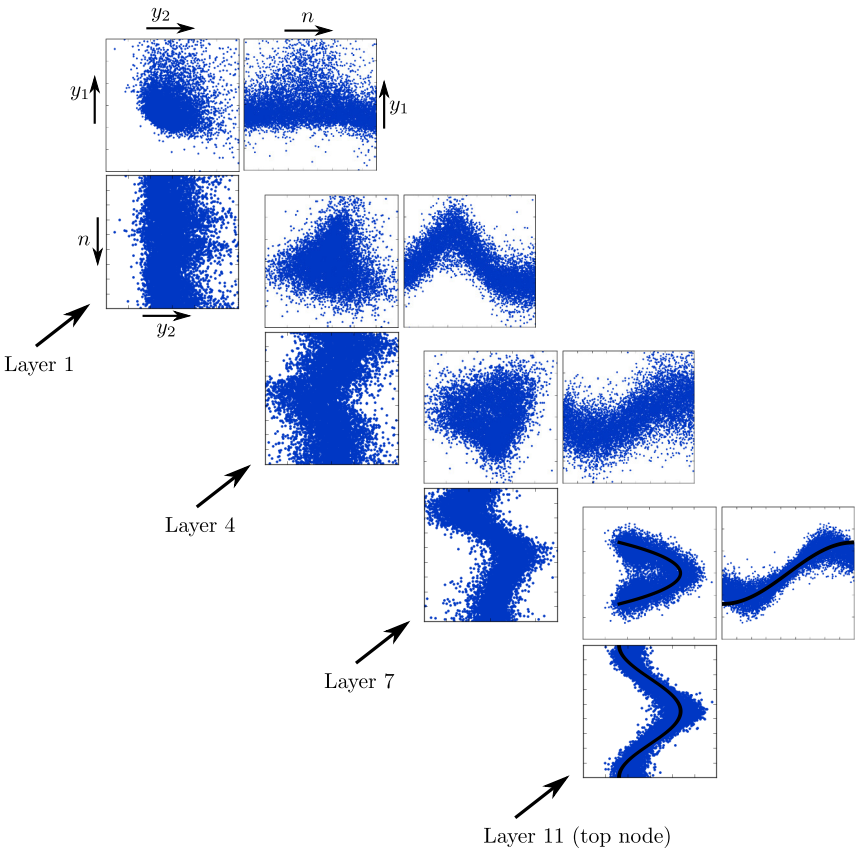


Figure 11: Evolution of the slow features extracted from test data after layers 1, 4, 7 and 11 of a GSFA network trained with the serial training graph. A central node was selected from each layer, and three plots are provided, that is, y_2 vs y_1, n vs y_1, and y_2 vs n. Hierarchical processing results in progressively slower features as one moves from the first to the top layer. 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, 2003). Notice how the features evolve from being mostly unstructured in the first layer to being similar to the free responses at the top node, indicating success at finding the hidden parameter changing most slowly for these data (i.e., the horizontal position of the faces).

- A nearest centroid classifier (NCC).
- Labels estimated using (36) 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 data set were grouped in 50 equally large classes according to their horizontal displacement x, $-45 \leq x \leq 45$.

6.2.4 RESULTS

We evaluated all the 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 was measured on test data and reported in terms of the RMSE. The labels estimated depend on three 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 were determined for each combination of algorithms using a single trial, but the RMSEs reported here were averaged over 5 trials.

The results are presented in Table 2, and 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 data set than when using HPCA.

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

Table 2: Performance (RMSE) of the dimensionality reduction algorithms measured in pixels 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 deviation of 0.21 pixels. Clearly, linear regression benefited from all the SFA and PCA features available.

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.

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 the chance level for HPCA. The SVMs were consistently better, but the error can be further reduced by 4% to 23% by using Soft GC, the soft labels derived from the Gaussian classifier.

Regarding the training graphs, we 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 one 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.

7. Discussion

In this paper, we propose the graph-based SFA (GSFA) optimization problem, an extension of the standard SFA optimization problem that takes into account the information contained 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 to such a graph.

We also propose the GSFA algorithm, an implicitly supervised extension of the (unsupervised) SFA algorithm, and prove that GSFA solves the new optimization problem in the function space considered. 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.

We call GSFA implicitly supervised because the labels themselves are never provided to it, 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 or regression are introduced in this paper. We have designed them aiming at a balance between speed 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 4 and 5). First, determining 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 neighbour algorithms. Second, as we have shown, linear algebra can be used to optimize the computation of $\dot{\mathbf{C}}$, 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 specially 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 paper, shown in Figure 1 (left), 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 we suggest 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. These 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.

7.1 Related Optimization Problems and Algorithms

Recently, Böhmer et al. (2012) introduced regularized sparse kernel SFA. The algorithm was applied to solve a classification problem by reducing the data dimensionality. In the discussion section, various extensions similar to the GSFA optimization problem were 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_s 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) are less appropriate in our view because larger classes are overrepresented by the quadratic number of edges $N_s(N_s+1)/2$ 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

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

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 us 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 node 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.

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. Such a framework is useful for comparing linear SFA and GSFA to other algorithms from a mathematical point of view, regardless of their typical usage and application areas.

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):

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

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 $\mathbf{D} = \text{diag}(1/2, 1, 1, \ldots, 1, 1/2)$ being slightly different from the identity and the equivalence would only be approximate.

Sprekeler (2011) studied the relation between SFA and LE and proposed the combination of the neighbourhood relation used by LE and the functional nature of SFA. The resulting algorithm, called generalized SFA, computes a functional approximation to LE with a feature space spanned by a set of basis functions. Linear generalized SFA is equivalent to linear LPP (Rehn, 2013), and in general it can be regarded as LPP on the data expanded by such basis functions.

Also the strong connection between LPP and linear GSFA is evident from the optimization problem above. In this case, the elements D_{ii} play the role of the node 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 node weights specifying a particular feature and objective function normalization. A second difference is that for GSFA the node weights, fundamental for feature normalization, can be chosen independently from the edge weights (unless one explicitly enforces (20)), whereas for LPP the diagonal elements $D_{ii} \stackrel{\text{def}}{=} \sum_j W_{ij}$ are derived from the edge weights.

We show now 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 node 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 as $\mathbf{y}_{\text{GSFA}}$, 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 node weights v_i, we can define the following similarity matrix:

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

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.

7.2 Remarks on Classification with GSFA

Both classification and regression tasks can be solved with GSFA. We 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, we propose a clustered training graph, which yields features having the discrimination 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 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 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 generation 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 generalized SFA, 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 neighbours are connected. Less susceptibility to outliers and better performance have been reported.

7.3 Remarks on Regression with GSFA

To solve regression problems, we propose 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, an improvement caused by the higher number of similarity relations considered even though the same number of training samples is used.

First extensions of SFA for regression (i.e., GSFA) were employed 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 modelling and rendering. Gender estimation was 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, undocumented versions of the mixed and serial training graphs were employed. Only three features extracted were passed to an explicit regression step based on a Gaussian classifier (Section 5.5). In both cases, good performance was 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, we participated in a face detection competition successfully (Mohamed and Mahdi, 2010). We estimated the x-position, y-position and scale of a face within an image. Using three separate SFA networks, one for each parameter, faces can be pose-normalized. A fourth network can be 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 greyscale 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 can be expected to result in further improvements.

The serial and mixed training graphs have provided the best accuracy for the experiments in this article, 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 the 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 are arbitrary, 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 less useful features than expected.

In this work, we have focused 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. For instance, using another algorithm, the estimation of age from face images has been reported to improve when also gender and race labels are estimated (Guo and Mu, 2011). This is justified because gender and race are two factors that strongly influence the aging process.

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.

7.4 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, 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 we ensure that the graph fulfils the normalization restrictions, the Markov chain described in Section 3.5 can be constructed, and learning with GSFA and such graph becomes equivalent to learning with standard (unsupervised) SFA as long as the training sequence originates from the Markov chain. From this perspective, GSFA uses the graph information to simplify a learning procedure that could also be done unsupervised.

We do not claim that GSFA is better or worse than other supervised learning algorithms, it is actually equivalent to other algorithms under specific conditions. We only show that it is better for supervised learning than SFA, and believe 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 (not discussed here) are under development, which will increase its performance and narrow the gap to 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 samples available 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. Due to accumulation of nonlinearities across the network, the features at the top node 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 our learning system. Of course, if the amount of training data and computation time were unrestricted, overfit-

ting 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 our pursuit of generalization.

Acknowledgments

We would like to thank Mathias Tuma for his helpful suggestions on an earlier version of this work and Fabian Schönfeld and Björn Weghenkel for their valuable corrections. The comments of the anonymous reviewers are also kindly appreciated. A. Escalante is jointly supported by the German Academic Exchange Service (DAAD) and the National Council of Science and Technology of Mexico (CONACYT).

References

T. Adali and S. Haykin. *Adaptive Signal Processing: Next Generation Solutions*. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. John Wiley & Sons, 2010.

P. Berkes. Pattern recognition with slow feature analysis. Cognitive Sciences EPrint Archive (CogPrints), February 2005a. URL `http://cogprints.org/4104/`.

P. Berkes. Handwritten digit recognition with nonlinear fisher discriminant analysis. In *ICANN*, volume 3697 of *LNCS*, pages 285–287. Springer Berlin/Heidelberg, 2005b.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5(6):579–602, 2005.

W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1):67–86, 2012.

A. Bray and D. Martinez. Kernel-based extraction of slow features: Complex cells learn disparity and translation invariance from natural images. In *NIPS*, volume 15, pages 253–260, Cambridge, MA, 2003. MIT Press.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

A. N. Escalante-B. and L. Wiskott. 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*, pages 240–249, 2010.

A. N. Escalante-B. and L. Wiskott. Heuristic evaluation of expansions for Non-Linear Hierarchical Slow Feature Analysis. In *Proc. The 10th International Conference on Machine Learning and Applications*, pages 133–138, Los Alamitos, CA, USA, 2011.

A. N. Escalante-B. and L. Wiskott. Slow feature analysis: Perspectives for technical applications of a versatile learning algorithm. *Künstliche Intelligenz [Artificial Intelligence]*, 26(4):341–348, 2012.

M. Fink, R. Fergus, and A. Angelova. Caltech 10,000 web faces. URL `http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/`.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7: 179–188, 1936.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):1605–1622, 2007.

M. Franzius, N. Wilbert, and L. Wiskott. Invariant object recognition with slow feature analysis. In *Proc. of the 18th ICANN*, volume 5163 of *LNCS*, pages 961–970. Springer, 2008.

M. Franzius, N. Wilbert, and L. Wiskott. Invariant object recognition and pose estimation with slow feature analysis. *Neural Computation*, 23(9):2289–2323, 2011.

W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao. 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.

G. Guo and G. Mu. 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.

X. He and P. Niyogi. Locality Preserving Projections. In *Neural Information Processing Systems*, volume 16, pages 153–160, 2003.

G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989.

O. Jesorsky, K. J. Kirchberg, and R. Frischholz. 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.

S. Klampfl and W. Maass. 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.

P. Koch, W. Konen, and K. Hein. Gesture recognition on few training data using slow feature analysis and parametric bootstrap. In *International Joint Conference on Neural Networks*, pages 1 –8, 2010.

T. Kuhnl, F. Kummert, and J. Fritsch. Monocular road segmentation using slow feature analysis. In *Intelligent Vehicles Symposium, IEEE*, pages 800–806, june 2011.

N. Kumar, P. N. Belhumeur, and S. K. Nayar. FaceTracer: A search engine for large collections of images with faces. In *European Conference on Computer Vision (ECCV)*, pages 340–353, 2008.

G. Mitchison. Removing time variation with the anti-hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.

N. M. Mohamed and H. Mahdi. 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.

P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 947–954, Washington, DC, USA, 2005. IEEE Computer Society.

E. M. Rehn. On the slowness principle and learning in hierarchical temporal memory. Master's thesis, Bernstein Center for Computational Neuroscience, 2013.

I. Rish, G. Grabarnik, G. Cecchi, F. Pereira, and G. J. Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In *Proc. of the 25th ICML*, pages 832–839, New York, NY, USA, 2008. ACM.

H. Sprekeler. On the relation of slow feature analysis and laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

H. Sprekeler and L. Wiskott. A theory of slow feature analysis for transformation-based input signals with an application to complex cells. *Neural Computation*, 23(2):303–335, 2011.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

M. Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *Proc. of the 23rd ICML*, pages 905–912, 2006.

M. Sugiyama, T. Idé, S. Nakajima, and J. Sese. Semi-supervised local fisher discriminant analysis for dimensionality reduction. *Machine Learning*, 78(1-2):35–61, 2010.

W. Tang and S. Zhong. *Computational Methods of Feature Selection*, chapter Pairwise Constraints-Guided Dimensionality Reduction. Chapman and Hall/CRC, 2007.

R. Vollgraf and K. Obermayer. Sparse optimization for second order kernel methods. In *International Joint Conference on Neural Networks*, pages 145–152, 2006.

L. Wiskott. 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.

L. Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.

L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

D. Zhang, Z.-H. Zhou, and S. Chen. Semi-supervised dimensionality reduction. In *Proc. of the 7th SIAM International Conference on Data Mining*, 2007.

T. Zhang, D. Tao, X. Li, and J. Yang. Patch alignment for dimensionality reduction. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1299–1313, 2009.

Z. Zhang and D. Tao. Slow feature analysis for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):436 –450, 2012.