Efficient update of the covariance matrix inverse in iterated linear discriminant analysis

Jan Salmen, Marc Schlipsing, Christian Igel

Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany

Abstract

For fast classification under real-time constraints, as required in many image-based pattern recognition applications, linear discriminant functions are a good choice. Linear discriminant analysis (LDA) computes such discriminant functions in a space spanned by real-valued features extracted from the input. The accuracy of the trained classifier crucially depends on these features, its time complexity on their number. As the number of available features is immense in most real-world problems, it becomes essential to use meta-heuristics for feature selection and/or feature optimization. These methods typically involve iterated training of a classifier after substitutions or modifications of features. Therefore, we derive an efficient incremental update formula for LDA discriminant functions for the substitution of features. It scales linearly in the number of altered features and quadratically in the overall number of features, while completely retraining scales cubically in the number of features. The update rule allows for efficient feature selection and optimization with any meta-heuristic that is based on iteratively modifying existing solutions. The proposed method was tested on an artificial benchmark problem as well as on a real-world problem. Results show that significant time savings during training are achieved while numerical stability is maintained.

Keywords: LDA, Feature selection, Feature optimization, Meta-heuristics, Covariance matrix update

Email addresses: Jan.Salmen@neuroinformatik.rub.de (Jan Salmen), Marc.Schlipsing@neuroinformatik.rub.de (Marc Schlipsing), Christian.Igel@neuroinformatik.rub.de (Christian Igel)

1. Introduction

Machine learning algorithms are standard tools for pattern recognition in image processing. As execution speed matters for many computer vision applications, often simple classifiers with low computational complexity are used in practice. Linear discriminant analysis (LDA) is a simple, but still state-of-the-art method to train such classifiers that can be sufficient for many real-world applications. Of course, classification rates crucially depend on the representation of the input patterns. Therefore, a set of good features which capture the "dominant non-linearities" [2] has to be found.

When facing image-based pattern recognition problems, an immense number of features is available. As an exhaustive search is impractical even for moderate sizes [8], it becomes essential to use heuristics: either *feature selection* methods to find an optimal subset from a large pool or *feature optimization* methods to construct such a set iteratively. Meta-heuristics such as evolutionary algorithms, tabu search, simulated annealing etc. have been successfully applied for these tasks. These and other [8] popular approaches have in common that features are iteratively substituted or modified, followed by training and evaluation of a classifier. Canonical meta-heuristics for selection or optimization repeat the loop:

1. Generate candidate set of features based on existing set(s).
2. Train classifier using these features.
3. Evaluate trained classifier.
4. If termination criterion not met, go to 1.

In this cycle, most computational costs typically result from training the classifier, even if it is a simple one. Training a classifier with LDA involves calculating the inverse of the shared covariance matrix, which requires $\Omega(n^2)$ operations where n is the dimensionality of the input space (i.e., number of features).

In this paper, we propose an efficient method to speed up iterated LDA training for such scenarios. Instead of recalculating the inverse for every change made in the feature set, we derive a formula based on the *Woodbury matrix identity* for incremental update of the covariance matrix inverse. The complexity for iterated training is reduced to $\Theta(n^2)$, the asymptotic lower bound, when a single feature is altered. This increase in efficiency allows to analyze larger feature sets more accurately (e.g., by considering more iterations or larger populations in population-based meta-heuristics).

This article is organized as follows. In the ongoing section, we introduce the basic concept of LDA. Then, we give an overview of related work on iterative LDA and distinguish the new approach presented here from others. In section 4, the efficient incremental update is presented, in section 5 its application to iterated training during feature selection is shown. Our experiments are described in section 6, finally the results are discussed.

2. Linear discriminant analysis

Linear discriminant analysis is based on a *maximum a posteriori* estimate of the class membership. Let the classification problem be described by two random variables X and Y with joint probability distribution on $\mathbb{R}^n \times \{c_1, \ldots, c_m\}$, where $\mathbb{R}^n$ is the feature space and $\{c_1, \ldots, c_m\}$ a set of m class labels. Let $P(Y = c_k \,|\, X = x)$ denote the probability that feature vector x belongs to class c_k. For two classes a pattern x is assigned to class c_1 or c_2 depending on whether the log-ratio

$$\ln \frac{P(Y = c_1 \,|\, X = x)}{P(Y = c_2 \,|\, X = x)} = \ln \frac{p(X = x \,|\, Y = c_1)}{p(X = x \,|\, Y = c_2)} + \ln \frac{P(Y = c_1)}{P(Y = c_2)} \tag{1}$$

is positive or negative. In LDA the classification rule is derived under the assumption that the class densities $p(X \,|\, Y)$ are multi-variate Gaussians having a common covariance matrix Σ (i.e., $p(X \,|\, Y = c_k) \sim \mathcal{N}(\mu_k, \Sigma)$), and the log-ratio above reduces to

$$\ln \frac{P(Y = c_1)}{P(Y = c_2)} - \frac{1}{2}(\mu_1 + \mu_2)^{\mathrm{T}} \Sigma^{-1} (\mu_1 - \mu_2) + x^{\mathrm{T}} \Sigma^{-1} (\mu_1 - \mu_2). \tag{2}$$

Given training data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \in (\mathbb{R}^n \times \{c_1, c_2\})^\ell$, the class priors are estimated by $P(Y = c_k) = \ell_k / \ell$, the class centers by $\mu_k = \frac{1}{\ell_k} \sum_{(x,y) \in S^{(k)}} x$, and the covariance matrix Σ by

$$\Sigma = \frac{1}{\ell - 2} \sum_{k=1}^{2} \sum_{(x,y) \in S^{(k)}} (x - \mu_k)(x - \mu_k)^{\mathrm{T}} \tag{3}$$

with $S^{(k)} = \{(x, y) \,|\, (x, y) \in S \wedge y = c_k\}$ and $\ell_k = |S^{(k)}|$.

Despite its simplicity, LDA gives surprisingly good results in practice [5]. Provided appropriate features, state-of-the-art results in many image-based object recognition tasks [1, 20, 10, 14, 16] are reported.

3. Related work

Linear discriminant analysis as presented in the previous section relies on a training data set S to compute a classification rule. For classification tasks in computer vision, S typically contains feature vectors $x_1, \ldots, x_l \in \mathbb{R}^n$ computed by a function $f : \mathbb{R}^m \to \mathbb{R}^n$ from a set $R = \{r_1, \ldots, r_l\}$ containing raw image data (w.l.o.g. we assume $r_i \in \mathbb{R}^m$ for $i = 1, \ldots, l$).

In many real-world applications, the available data R may change over time. This occurs for example in *online* classification scenarios, where a classifier is trained once on initially available data S_0 computed from R_0. During runtime, new information can be collected continuously, therefore new data sets $R_1, R_2, \ldots, R_n$ becoming available. It is a crucial issue to adapt the classifier online according to that new knowledge.

But also in solely *offline* scenarios, efficient handling of changing training data plays an important role. Employing meta-heuristics during training (e.g., feature selection, feature optimization, cross-validation) requires iterated training on different but interrelated training sets $S_0, S_1, \ldots, S_n$. For real-world data, the naïve solution—complete retraining in each step—is often impossible due to time and memory restrictions.

Therefore, efficient update rules for suchlike scenarios have been studied. The solution to a concrete problem at hand strongly depends on the type of changes that are assumed from R_i to R_{i+1}, thereby from S_i to S_{i+1}, and on the constraints on f. In general, changes that can occur may affect the number of training samples $|R|$, the number of classes, and the representation of training samples in S due to changing f.

The special case $R_0 \subseteq R_1 \subseteq \ldots$ and $S_0 \subseteq S_1 \subseteq \ldots$, thus continuously adding training data while keeping f fixed, was studied by Pang et al. [12] who propose an incremental LDA for sequential and chunk updates, Kim et al. [9] who present an approximation based on sufficient spanning set, and Huang et al. [6].

Tapp and Kemsley [17] study update formulas for more efficient cross-validation in LDA. They showed how the covariance matrix can be updated when single training examples are exchanged, thus $|R_i \cap R_{i+1}| = |R_i| - 1$ while the feature calculation f is unchanged.

In this work, we consider a different scenario. We assume that the raw training samples do not change ($R_i = R_{i+1}$), but their representation S_i varies due to variable feature extraction, thus f is assumed to be altered. A method handling this efficiently can be used to continuously update a classifier while

feature calculation is altered (e.g., during selection or optimization). More formally, we deal with the following task: Perform efficient training of an LDA classifier in step i given

- new training data S_i,
- training data S_{i-1},
- an LDA classifier trained for S_{i-1},
- and $f_i \approx f_{i-1}$.

The notation $f_i \approx f_{i-1}$ indicates that the representation of the training examples changes only in few dimensions from S_{i-1} to S_i. In the next section, we present an efficient update rule for the case that only the representation in one dimension is changing. This update can be repeated as often as required when more changes happen.

The formula we present here allows for an efficient *rank-two* update to adapt the inverse of the covariance matrix. Similar *rank-one* updates have been proposed for Kalman filtering [4] and recently in the domain of meta-heuristics for variable metric evolution strategies [15].

4. Efficient incremental update

Given training data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \in (\mathbb{R}^n \times \{c_1, \ldots\})^\ell$, let $\hat{S} = \{(\hat{x}_1, y_1), \ldots, (\hat{x}_\ell, y_\ell)\} \in (\mathbb{R}^n \times \{c_1, \ldots\})^\ell$ be the training data which results from substituting or modifying a single feature. That is, if the ith feature is altered, for any $k = 1, \ldots, l$ the feature vectors x_k and $\hat{x}_k$ may only differ in their ith component while all other components are equal.

Let Σ denote a common covariance matrix as used by LDA trained on S and $\hat{\Sigma}$ the updated matrix resulting from changes in one input dimension. We assume that iterative training of a classifier with LDA is performed and that the inverse matrix Σ^{-1} is known. Then, the calculation of $\hat{\Sigma}^{-1}$ can be done more efficiently, compared to explicit inversion, by an update based on the decomposition

$$\hat{\Sigma}^{-1} = (\Sigma + G)^{-1} \tag{4}$$

and application of the *Woodbury matrix identity*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U \left(C^{-1} + VA^{-1}U\right)^{-1} VA^{-1}. \tag{5}$$

Here,

$$G = \hat{\Sigma} - \Sigma = \begin{pmatrix} 0 & \cdots & g_{1\,i} & \cdots & 0 \\ 0 & \cdots & \vdots & \cdots & 0 \\ g_{i\,1} & \cdots & g_{i\,i} & \cdots & g_{i\,n} \\ 0 & \cdots & \vdots & \cdots & 0 \\ 0 & \cdots & g_{n\,i} & \cdots & 0 \end{pmatrix} \tag{6}$$

is a matrix with rank two.

The crucial point in order to benefit from eq. 5 is to bring G to a form $G = UCV$ where the matrix C has low dimensionality. This can be achieved by choosing

$$U = \begin{pmatrix} g_{1\,i} & 0 \\ g_{2\,i} & 0 \\ \vdots & \vdots \\ g_{(i-1)\,i} & 0 \\ g_{i\,i} & 1 \\ g_{(i+1)\,i} & 0 \\ \vdots & \vdots \\ g_{n\,i} & 0 \end{pmatrix}, \tag{7}$$

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{8}$$

and

$$V = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ g_{1\,i} & g_{2\,i} & \cdots & g_{(i-1)\,i} & 0 & g_{(i+1)\,i} & \cdots & g_{n\,i} \end{pmatrix}. \tag{9}$$

Now $\hat{\Sigma}^{-1}$ can be computed as

$$\hat{\Sigma}^{-1} = (\Sigma + UCV)^{-1} = \Sigma^{-1} - \Sigma^{-1}U \left(C^{-1} + V\Sigma^{-1}U\right)^{-1} V\Sigma^{-1}. \tag{10}$$

Note that here $C^{-1} = C$.

This is more efficient than calculating the inverse directly: $\Sigma^{-1}U$ and $V\Sigma^{-1}$ are ordinary matrix products with $2n$ elements each. The only inverse

left to compute, $(C^{-1} + V\Sigma^{-1}U)^{-1}$, is a 2×2 matrix and therefore can simply be calculated as

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}. \tag{11}$$

Multiplying the resulting matrices on the right side of eq. 10 and merging has complexity of $\Theta(n^2)$.

The total complexity for this incremental update is $\Theta(n^2)$. As the update step can be repeated multiple times, the resulting complexity is $\Theta(Nn^2)$ when changes in N dimensions occur.

5. Application to iterative training with LDA

To benefit from the presented update rule during feature selection or optimization using a meta-heuristic that performs iterated adaptations is straight-forward: The covariance matrix has to be inverted only once directly to initialize the algorithm. All further changes to the feature set can be adopted by the incremental update. For the substitution of one feature resulting in new input values in dimension i, the following steps are performed:

1. Calculate new class means $\hat{\mu}_{ki}$
2. Calculate new covariance matrix entries $\hat{c}_{1\,i}, \ldots, \hat{c}_{n\,i}$ and $\hat{c}_{i\,1}, \ldots, \hat{c}_{i\,n}$
3. Calculate values $g_{1\,i}, \ldots, g_{n\,i}$ and $g_{i\,1}, \ldots, g_{i\,n}$ as in eq. 6
4. Build U and V according to eq. 7 and eq. 9
5. Compute $\hat{\Sigma}^{-1}$ using the update rule from eq.10

If more than one feature is substituted, these steps can be repeated as often as required.

After exchanging features, the classifier can be retrained by LDA based on class means $\hat{\mu}_k$ and the covariance matrix $\hat{\Sigma}^{-1}$ updated this way. This retraining is simply done (cf. eq. 2) by calculating the new weight vector

$$\hat{w} = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2) \tag{12}$$

and the constant

$$\hat{b} = -\frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_2)^{\mathrm{T}}\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2). \tag{13}$$

6. Experiments

The goal of our experiments is to study two aspects:

Numerical stability. The numerical stability regarding the incrementally updated inverse $\hat{\Sigma}^{-1}$ can be evaluated based on the terms $I_l = \hat{\Sigma}^{-1}\hat{\Sigma}$ and $I_r = \hat{\Sigma}\hat{\Sigma}^{-1}$. Both, I_l and I_r, should result in the identity matrix I. The *max-norm* $\|A\|_\infty$ defined by

$$\|A\|_\infty = \max\{|a_{ij}|\} \tag{14}$$

was used to determine the largest deviation

$$e = \max\{\|I - I_l\|_\infty, \|I - I_r\|_\infty\} \tag{15}$$

at the end of the training process (e.g., when the meta-heuristic has terminated).

Time measurements. In order to show that the theoretical results really lead to an improvement in practice, we compared the runtime of meta-heuristics using the proposed incremental update to the basically same algorithms performing complete retraining in every iteration.

We studied feature selection on an artificial test problem (in sec. 6.1) and feature optimization for a real-world classification problem (in sec. 6.2).

6.1. Artificial feature selection example

For our first experiment we consider the classical example by Trunk [18][8]. In this binary classification problem, n-dimensional feature vectors $x \in \mathbb{R}^n$ have to be assigned to one of two classes. Both classes are equally likely *a priori*. The likelihood of observing $x \in \mathbb{R}^n$ given the class $y \in \{\pm 1\}$ is a Gaussian

$$p(x \mid y) = \frac{1}{\sqrt{2\pi}} e^{-\|x - ym\|^2/2} \tag{16}$$

with mean ym and unit covariance matrix. The vector m has components $m_i = \sqrt{1/i}$ for $i = 1, \ldots, n$.

The Bayes optimal decision rule assigns an input x to class 1 if $x^{\mathrm{T}}m > 0$ and to class 0 otherwise. The Bayes error is given by $P_e\left(\sqrt{\sum_{i=1}^{n} 1/i}\right)$, where the auxiliary function P_e is defined as $P_e(r) = \int_r^\infty \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \mathrm{d}z$.

Since $\sum_{i=1}^{n} 1/i$ diverges for increasing n it is easy to see that increasing the number of features reduces the error probability, which converges to zero for growing n [18].

A simple Bayes plug-in classifier (with prior knowledge that the class-conditional densities are Gaussians with unit covariance matrix and that the problem is symmetric) given a training sample $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ converges to the Bayes optimal decision rule for $\ell \to \infty$ for any dimensionality n. However, Trunk showed that for fixed ℓ the error probability converges to chance level for $n \to \infty$.

6.1.1. Experimental Setup

We considered the problem of selecting an optimal subset $\mathcal{S}$ of fixed size out of a feature pool $\mathcal{P}$. For our experiments we chose $|\mathcal{S}| = 100$ and $|\mathcal{P}| = 2000$.

In total, $\ell = 6000$ training samples were generated randomly according to eq. 16 using stratified sampling. These examples were partitioned into three sets to allow for cross-validation, thus resulting in 1000 samples per class in each set.

In our experiments, we tried to keep the algorithm as simple as possible. Therefore, a stochastic hill-climbing strategy was used to perform feature selection. This meta-heuristic considers one candidate solution, the feature set $\mathcal{C}$, which is iteratively improved by evaluating possible substitutions. The cross-validation error of the linear classifier was computed in every iteration after modifying the solution. If this error increased, all changes were reversed, otherwise held.

The set $\mathcal{C}$ was initialized with $n = 100$ random features from $\mathcal{P}$. In each of the $g_{\max} = 10^6$ generations, features from $\mathcal{C}$ were randomly replaced by others from $\mathcal{P} \backslash \mathcal{C}$. The number of features N to exchange was determined in every iteration g according to a Poisson distribution with expectation $\lambda = |\mathcal{C}| \cdot 10^{-1-\frac{g}{g_{\max}}}$.

We performed 40 trials where $\mathcal{P}$ was randomly initialized every time. Our *C++* implementation was based on the *Shark* open-source machine learning library [7]. The *OpenMP* standard[1] was used to benefit from parallel program execution on the test PC[2].

[1] `http://openmp.org`

[2] Intel Core 2 Duo CPU E6300 with 3 GB RAM

6.1.2. Results

Analysis of the performed trials showed the following results:

Numerical stability. The largest error (cf. eq. 15) measured at the end of a trial was $9 \cdot 10^{-15}$. The mean error in all trials was $6 \cdot 10^{-15}$. These deviations have practically no influence on the performance of the classifier.

Time measurements. The mean runtime for feature selection was 63 min. The reference implementation without the incremental update was slower by a factor of 40.

The test problem also allows to evaluate and verify the convergence of the meta-heuristic used. The histogram in figure 1 shows how often each of the best 200 features was selected for the final set in the performed trials. The most important features have been reliably found by the heuristic used. Thus, our simple feature selection algorithm performed reasonably well.

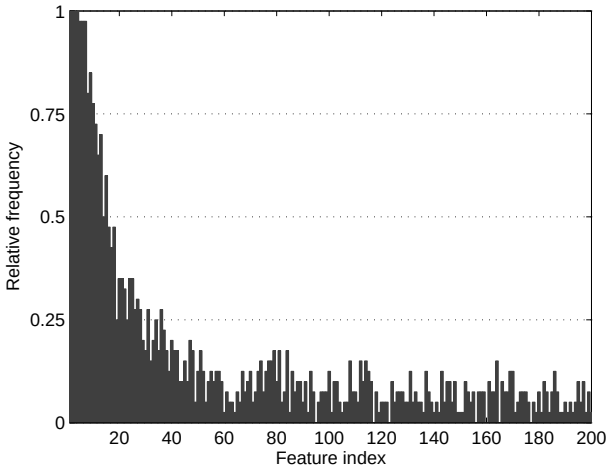

Figure 1: Relative frequency that feature $i = 1, \ldots 200$ was chosen for the final set. The smaller i the more discriminative the feature.

6.2. Real-world feature optimization example

As a real-world example, we consider the problem of optimizing a set of *Haar wavelet features* for pedestrian classification. Solving this task in real-time plays an important role for many applications, amongst others driver

assistance systems and surveillance. A classifier based on LDA can meet real-time constraints when an appropriate set of features is used.

Haar wavelet features are state-of-the-art for real-time computer vision. Their popularity is mainly based on the efficient computation using the *integral image* [19]. They were successfully applied, for instance, for object detection, classification, and tracking [19, 13, 14, 3]. Figure 2 shows examples of six basic types of Haar wavelet features. Their responses can be calculated with 6 to 9 look-ups in the integral image, independently of their absolute sizes.

Figure 2: Basic types of Haar wavelet features.

Haar wavelet features can serve as a universal basis (i.e., offer a corporate preprocessing) when different computer vision applications have to be run in parallel. We therefore assume that a small fixed number of Haar wavelet features is evaluated for every pixel in a camera image. Each of those features can be described by its width, height, and type. The goal of our experiments here is to automatically construct an optimal feature set for pedestrian detection.

6.2.1. Experimental Setup

We considered the pedestrian classification benchmark dataset introduced by Munder and Gavrila [11], where $29,400$ examples are available for training and $19,600$ for testing, all of them 18×36 pixel grayscale images. Figure 3 shows some samples from the database.

Figure 3: Samples from the pedestrian classification benchmark dataset [11].

We chose a feature set size of 7. As six different feature types are considered (see fig. 2) and the size varies from 3×3 to 16×16 pixels, more than

10^{20} different feature sets were possible. This number can not be handled by exhaustive search and employing meta-heuristics here is essential in order to find an optimal setup.

Each feature vector consists of the absolute responses. Pixels outside the image border were set to the mean image value for border handling.

We performed 5 trials of feature optimization, each consisted of 500 generations. In each generation, one new candidate solution resulted from mutating one parent solution. Mutations were realized by randomly choosing one of the seven features and either change its parameters (size and type) or substitute it by a random new one. If the 3-fold cross validation error on the training sets did not increase, the offspring solution was kept as parent for the next generation.

During optimization, we trained the classifier based on responses of the features at every second pixel (therefore, $n = 952$). As the same features are evaluated across the whole image, changing the parameters of one feature leads to changes in $N = 136$ dimensions of feature vectors.

6.2.2. Results

Analysis of the performed trials showed the following results:

Numerical stability. The largest error measured at the end of a trial was $5.2 \cdot 10^{-5}$. The mean error in all trials was $7.1 \cdot 10^{-6}$. These deviations have practically no influence on the performance of the classifier.

Time measurements. The mean runtime for feature optimization was 22.4 h. This is 60 % less than the implementation without the incremental update requires. The speed-up in this experiment is not as high as in the other experiments presented in section 6.1. This is because changing one Haar wavelet feature here influences one seventh of the dimensions in the feature vectors. Nevertheless, even in this case the gain in speed is significant and important for real-world applications.

Figure 4: Optimized feature set. The features have sizes of 16×8, 3×3, 6×5, 3×9, 4×3, 4×4, and 3×5.

The best set of features found in the trials is shown in figure 4. We evaluated the performance of the optimized classifier on the two test sets

according to the method proposed in [11]. The resulting *ROC curve* given in figure 5 illustrates all possible trade-offs between false alarm rate and correct detection rate. As the classifier considered here is extremely fast, it is very well suited for *initial detection* in a whole camera image to compute regions of interest for final pedestrian classification.

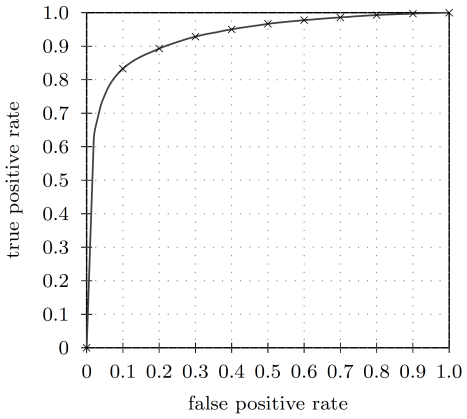

Figure 5: ROC curve of optimized classifier.

7. Conclusion

We proposed an incremental update of the covariance matrix inverse for iterated training of a classifier with linear discriminant analysis. This update reduces the complexity for iterated retraining from cubical to $\Theta(Nn^2)$ when N components of the input dimensions are changed. For $N = 1$ this is the asymptotic lower bound.

Our experiments on an artificial feature selection problem and a real-world feature optimization problem showed that the update rule allows for significant time saving during training while being numerically stable. Therefore, exploring larger feature pools more accurately becomes possible.

The problem of analyzing an immense number of features is common for many computer vision applications. Almost all meta-heuristics for LDA feature selection or LDA feature optimization can benefit from the proposed update rule.

[1] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 19(7):711–720, 1997.

[2] D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[3] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 260–267, 2006.

[4] M. S. Grewal and A.P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice-Hall, 1993.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[6] Z. Huang, K. Ding, L. Jin, and X. Gao. Writer adaptive online handwriting recognition using incremental linear discriminant analysis. In *Proceedings of the IEEE Conference on Document Analysis and Recognition*, pages 91–95, 2009.

[7] C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[8] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[9] T.-K. Kim, S.-F. Wong, B. Stenger, J. Kittler, and R. Cipolla. Incremental linear discriminant analysis using sufficient spanning set approximations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[10] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition. *Pattern Recognition Letters*, 26(2):181–191, 2005.

[11] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006.

[12] S. Pang, S. Ozawa, and N. Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(5):905–914, 2005.

[13] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International of Journal Computer Vision*, 38(1):15–33, 2000.

[14] J. Salmen, T. Suttorp, J. Edelbrunner, and C. Igel. Evolutionary optimization of wavelet feature sets for real-time pedestrian classification. In A. König, M. Köppen, N. Kasabov, A. Abraham, and C. Igel, editors, *Proceedings of the IEEE Conference on Hybrid Intelligent Systems*, pages 222–227. IEEE Press, 2007.

[15] T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 75:167–197, 2009.

[16] F. Tang and H. Tao. Fast linear discriminant analysis using binary bases. *Pattern Recognition Letters*, 28(16):2209–2218, 2007.

[17] H. S. Tapp and E. K. Kemsley. Optimizing the efficiency of cross-validation in linear discriminant analysis through selective use of the sherman-morrison-woodbury inversion formula. *Journal of Chemometrics*, 22(6):419–421, 2008.

[18] G. V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):306–307, 1979.

[19] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[20] J. Ye and Q. Li. LDA/QR: an efficient and effective dimension reduction algorithm and its theoretical foundation. *Pattern Recognition*, 37(4):851–854, 2004.