Cognitive Sciences EPrint Archive (CogPrints) 4104, (2005)
http://cogprints.org/4104/

Pattern Recognition with Slow Feature Analysis

Pietro Berkes
Institute for Theoretical Biology
Humboldt University Berlin
Invalidenstraße 43, D-10115 Berlin, Germany
p.berkes@biologie.hu-berlin.de
http://itb.biologie.hu-berlin.de/~berkes

Abstract

Slow feature analysis (SFA) is a new unsupervised algorithm to learn nonlinear functions that extract slowly varying signals out of the input data. In this paper we describe its application to pattern recognition. In this context in order to be slowly varying the functions learned by SFA need to respond similarly to the patterns belonging to the same class. We prove that, given input patterns belonging to C non-overlapping classes and a large enough function space, the optimal solution consists of $C-1$ output signals that are constant for each individual class. As a consequence, their output provides a feature space suitable to perform classification with simple methods, such as Gaussian classifiers. We then show as an example the application of SFA to the MNIST handwritten digits database. The performance of SFA is comparable to that of other established algorithms. Finally, we suggest some possible extensions to the proposed method. Our approach is in particular attractive because for a given input signal and a fixed function space it has no parameters, it is easy to implement and apply, and it has low memory requirements and high speed during recognition. SFA finds the global solution (within the considered function space) in a single iteration without convergence issues. Moreover, the proposed method is completely problem-independent.

Keywords: slow feature analysis, pattern recognition, digit recognition, unsupervised feature extraction

1 Introduction

Slow feature analysis (SFA) is a new unsupervised algorithm to learn nonlinear functions that extract slowly varying signals from time series (Wiskott and Sejnowski, 2002). SFA was originally conceived as a way to learn salient features of time series in a way invariant to frequent transformations (see also Wiskott, 1998). Such a representation would of course be ideal to perform classification in pattern recognition problems. Most such problems, however, do not have a temporal structure, and it is thus necessary to reformulate the algorithm. The basic idea is to construct a large set of small time series with only two elements chosen from patterns that belong to the same class (Fig. 1a). In order to be slowly varying, the functions learned by SFA will need to respond similarly to both elements of the time series (Fig. 1b), and therefore ignore the transformation between the individual patterns. As a consequence, patterns corresponding to the same class will cluster in the feature space formed by the output signals of the slowest functions, making it suitable to perform classification with simple techniques such as Gaussian classifiers. It is possible to show that in the ideal case the output of the functions is constant for all patterns of a given class, and that the number of relevant functions is small (Sect. 3). Notice that this approach does not use any a priori knowledge of the problem. SFA simply extracts the information about relevant features and common transformations by comparing pairs of patterns.

In the next section we introduce the SFA algorithm. In Section 3 we adapt it to the pattern recognition problem and consider the optimal output signal. In the last section we apply the proposed method to an handwritten digit recognition problem and discuss the results.

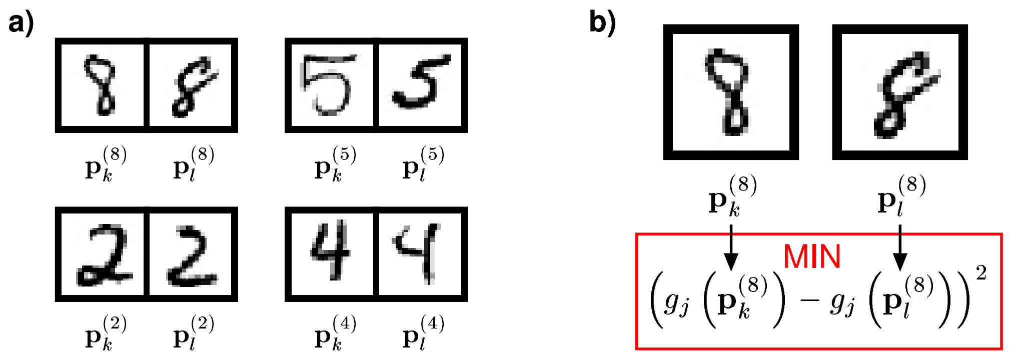
a)
$\mathbf{p}_k^{(8)}$ $\mathbf{p}_l^{(8)}$
$\mathbf{p}_k^{(5)}$ $\mathbf{p}_l^{(5)}$
$\mathbf{p}_k^{(2)}$ $\mathbf{p}_l^{(2)}$
$\mathbf{p}_k^{(4)}$ $\mathbf{p}_l^{(4)}$
b)
$\mathbf{p}_k^{(8)}$ $\mathbf{p}_l^{(8)}$
MIN
$\left(g_j\left(\mathbf{p}_k^{(8)}\right) - g_j\left(\mathbf{p}_l^{(8)}\right)\right)^2$

Figure 1 SFA for pattern recognition The input to SFA for pattern recognition is formed by small time series consisting of pairs of patterns that belong to the same class. **(a)** Sample time series in a digit recognition problem. **(b)** In order to be slowly varying the functions learned by SFA must respond similarly to both elements of the time series, as defined in Eq. (29).

2 Slow feature analysis

2.1 Problem statement

As originally defined in (Wiskott and Sejnowski, 2002), SFA solves the following problem: given a multidimensional time series $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))^T$, $t \in [t_0, t_1]$, find a set of real-valued functions $g_1(\mathbf{x}), \ldots, g_M(\mathbf{x})$ lying in a function space $\mathcal{F}$ such that for the output signals $y_j(t) := g_j(\mathbf{x}(t))$

$$\Delta(y_j) := \langle \dot{y}_j^2 \rangle_t \quad \text{is minimal} \tag{1}$$

under the constraints

$$\begin{aligned} \langle y_j \rangle_t &= 0 \quad \text{(zero mean)}\,, & (2)\\ \langle y_j^2 \rangle_t &= 1 \quad \text{(unit variance)}\,, & (3)\\ \forall i < j, \quad \langle y_i y_j \rangle_t &= 0 \quad \text{(decorrelation and order)}\,, & (4) \end{aligned}$$

with $\langle . \rangle_t$ and $\dot{y}$ indicating time averaging and the time derivative of y, respectively.

Equation (1) introduces a measure of the temporal variation of a signal (the Δ-*value* of a signal) equal to the mean of the squared derivative of the signal. This quantity is large for quickly-varying signals and zero for constant signals. The zero-mean constraint (2) is present for convenience only, so that (3) and (4) take a simple form. Constraint (3) means that each signal should carry some information and avoids the trivial solution $g_j(\mathbf{x}) = 0$. Alternatively, one could drop this constraint and divide the right side of (1) by the variance $\langle y_j^2 \rangle_t$. Constraint (4) forces different signals to be uncorrelated and thus to code for different aspects of the input. It also induces an order, the first output signal being the slowest one, the second being the second slowest, etc. .

2.2 The linear case

Consider first the linear case

$$g_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x}\,, \tag{5}$$

for some input $\mathbf{x}$ and weight vectors $\mathbf{w}_j$. In the following we assume $\mathbf{x}$ to have zero mean (i.e. $\langle \mathbf{x} \rangle_t = \mathbf{0}$) without loss of generality. This implies that Constraint (2) is always fulfilled, since

$$\langle y_j \rangle_t = \langle \mathbf{w}_j^T \mathbf{x} \rangle_t = \mathbf{w}_j^T \langle \mathbf{x} \rangle_t = 0\,. \tag{6}$$

We can rewrite Equations (1), (3) and (4) as

$$\begin{aligned}\Delta(y_j) &= \langle \dot{y_j}^2 \rangle_t & (7)\\ &= \langle (\mathbf{w}_j^T \dot{\mathbf{x}})^2 \rangle_t & (8)\\ &= \mathbf{w}_j^T \langle \dot{\mathbf{x}}\dot{\mathbf{x}}^T \rangle_t \mathbf{w}_j & (9)\\ &=: \mathbf{w}_j^T \mathbf{A} \mathbf{w}_j & (10)\end{aligned}$$

and

$$\begin{aligned}\langle y_i y_j \rangle_t &= \langle (\mathbf{w}_i^T \mathbf{x})(\mathbf{w}_j^T \mathbf{x}) \rangle_t & (11)\\ &= \mathbf{w}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle_t \mathbf{w}_j & (12)\\ &=: \mathbf{w}_i^T \mathbf{B} \mathbf{w}_j \,. & (13)\end{aligned}$$

If we integrate Constraint (3) in the objective function (1), as suggested in the previous section, we obtain:

$$\Delta(y_j) \underset{(1,3)}{=} \frac{\langle \dot{y_j}^2 \rangle_t}{\langle y_j^2 \rangle_t} \underset{(10,13)}{=} \frac{\mathbf{w}_j^T \mathbf{A} \mathbf{w}_j}{\mathbf{w}_j^T \mathbf{B} \mathbf{w}_j} \,. \tag{14}$$

It is known from linear algebra that the weight vectors $\mathbf{w}_j$ that minimize this equation correspond to the eigenvectors of the generalized eigenvalue problem

$$\mathbf{A}\mathbf{W} = \mathbf{B}\mathbf{W}\mathbf{\Lambda} \,, \tag{15}$$

where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)$ is the matrix of the generalized eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of the generalized eigenvalues $\lambda_1, \ldots, \lambda_N$ (see e.g. Gantmacher, 1959, chap. 10.7, Theorem 8, 10 and 11). The vectors $\mathbf{w}_j$ can be normalized such that $\mathbf{w}_i^T \mathbf{B} \mathbf{w}_j = \delta_{ij}$, which implies that Constraints (3) and (4) are fulfilled:

$$\begin{aligned}\langle y_j^2 \rangle_t &= \mathbf{w}_j^T \mathbf{B} \mathbf{w}_j = 1 \,, & (16)\\ \langle y_i y_j \rangle_t &= \mathbf{w}_i^T \mathbf{B} \mathbf{w}_j = 0 \,. & (17)\end{aligned}$$

Note that, by substituting Equation (15) into Equation (14) one obtains

$$\Delta(y_j) = \lambda_j \,, \tag{18}$$

so that by sorting the eigenvectors by increasing eigenvalues we induce an order where the most slowly varying signals have lowest indices (i.e. $\Delta(y_1) \leq \Delta(y_2) \leq \ldots \leq \Delta(y_N)$), as required by Constraint (4).

2.3 The general case

In the more general case of a finite dimensional function space $\mathcal{F}$, consider a basis $h_1, \ldots, h_M$ of $\mathcal{F}$. For example, in the standard case where $\mathcal{F}$ is the space of all polynomials of degree d, the basis will include all monomials up to order d.

Defining the *expanded input*

$$\mathbf{h}(\mathbf{x}) := (h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})) \,, \tag{19}$$

every function $g \in \mathcal{F}$ can be expressed as

$$g(\mathbf{x}) = \sum_{k=1}^{M} w_k h_k(\mathbf{x}) = \mathbf{w}^T \mathbf{h}(\mathbf{x}) \,. \tag{20}$$

This leads us back to the linear case if we assume that $\mathbf{h}(\mathbf{x})$ has zero mean (again, without loss of generality), which can be easily obtained in practice by subtracting the mean over time $\langle \mathbf{h}(\mathbf{x}) \rangle_t =: \mathbf{h}_0$ from the expanded input signal.

For example, in the case of 3 input dimensions and polynomials of degree 2 we have

$$\mathbf{h}(\mathbf{x}) = (x_1^2,\, x_1x_2,\, x_1x_3,\, x_2^2,\, x_2x_3,\, x_3^2,\, x_1,\, x_2,\, x_3) - \mathbf{h}_0 \tag{21}$$

and

$$\begin{aligned} g(\mathbf{x}) \underset{(20)}{=} & w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_1 x_3 + w_4 x_2^2 + w_5 x_2 x_3 + w_6 x_3^2 \\ & + w_7 x_1 + w_2 x_2 + w_8 x_3 - \mathbf{h}_0^T \mathbf{x} \end{aligned} \tag{22}$$

Every polynomial of degree 2 in the 3 input variables can then be expressed by an appropriate choice of the weights w_i in (22).

2.4 The SFA algorithm

We can now formulate the Slow Feature Analysis (SFA) algorithm (cf. Wiskott and Sejnowski, 2002):

Nonlinear expansion: Expand the input data and compute the mean over time $\mathbf{h}_0 := \langle \mathbf{h}(\mathbf{x}) \rangle_t$ to obtain the expanded signal

$$\mathbf{z} := \mathbf{h}(\mathbf{x}) - \mathbf{h}_0 \tag{23}$$
$$= (h_1(\mathbf{x}), \dots, h_M(\mathbf{x})) - \mathbf{h}_0 \,. \tag{24}$$

Slow feature extraction: Solve the generalized eigenvalue problem

$$\mathbf{AW} = \mathbf{BW\Lambda} \tag{25}$$
$$\text{with} \quad \mathbf{A} := \langle \dot{\mathbf{z}} \dot{\mathbf{z}}^T \rangle_t \tag{26}$$
$$\text{and} \quad \mathbf{B} := \langle \mathbf{z}\mathbf{z}^T \rangle_t \,. \tag{27}$$

The K eigenvectors $\mathbf{w}_1, \dots, \mathbf{w}_K$ ($K \leq M$) corresponding to the smallest generalized eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_K$ define the nonlinear input-output functions $g_1(\mathbf{x}), \dots, g_K(\mathbf{x}) \in \mathcal{F}$:

$$g_j(\mathbf{x}) \quad := \quad \mathbf{w}_j^T (\mathbf{h}(\mathbf{x}) - \mathbf{h}_0) \tag{28}$$

which satisfy Constraints (2)-(4) and minimize (1).

In other words to solve the optimization problem (1) it is sufficient to compute the covariance matrix of the signals and that of their derivatives in the expanded space and then solve the generalized eigenvalue problem (25).

3 SFA for pattern recognition

The pattern recognition problem can be summarized as follow. Given C distinct classes $c_1, \dots, c_C$ and for each class c_m a set of P_m patterns $\mathbf{p}_1^{(m)}, \dots, \mathbf{p}_{P_m}^{(m)}$ we are requested to learn the mapping $c(\cdot)$ between a pattern $\mathbf{p}_j^{(m)}$ and its class $c(\mathbf{p}_j^{(m)}) = c_m$. We define $P := \sum_{m=1}^{C} P_m$ to be the total number of patterns.

In general in a pattern recognition problem the input data does not have a temporal structure and it is thus necessary to reformulate the definition of the SFA algorithm. Intuitively, we want to obtain a set of functions that respond similarly to patterns belonging to the same class. The basic idea is to consider time series of just two patterns $(\mathbf{p}_k^{(m)}, \mathbf{p}_l^{(m)})$, where k and l are two distinct indices in a class c_m (Fig. 1).

Rewriting Equation (1) using the mean over all possible pairs we obtain

$$\Delta(y_j) = a \cdot \sum_{m=1}^{C} \sum_{\substack{k,l=1 \\ k<l}}^{P_m} \left(g_j(\mathbf{p}_k^{(m)}) - g_j(\mathbf{p}_l^{(m)}) \right)^2 , \tag{29}$$

where the normalization constant a equals one over the number of all possible pairs, i.e.

$$a = \frac{1}{\sum_{m=1}^{C} \binom{P_m}{2}} \,. \tag{30}$$

We reformulate Constraints (2)–(4) by substituting the average over time with the average over all patterns, such that the learned functions are going to have zero mean, unit variance and be decorrelated when applied to the whole training data.

$$\frac{1}{P}\sum_{m=1}^{C}\sum_{k=1}^{P_m} g_j(\mathbf{p}_k^{(m)}) = 0 \quad \text{(zero mean)}\,, \tag{31}$$

$$\frac{1}{P}\sum_{m=1}^{C}\sum_{k=1}^{P_m} g_j(\mathbf{p}_k^{(m)})^2 = 1 \quad \text{(unit variance)}\,, \tag{32}$$

$$\forall i < j, \quad \frac{1}{P}\sum_{m=1}^{C}\sum_{k=1}^{P_m} g_i(\mathbf{p}_k^{(m)}) g_j(\mathbf{p}_k^{(m)}) = 0 \quad \text{(decorrelation and order)}\,. \tag{33}$$

Comparing Equations (1–4) with Equations (29, 31–33) it is possible to see that to respect the new formulation, matrix **A** (Eq. 26) must be computed using the derivatives of all possible pairs of patterns while matrix **B** (Eq. 27) using all training patterns just once. Since the total number of pairs increases very fast with the number of patterns it is sometimes necessary to approximate **A** with a random subset thereof.

It is clear by (29) that the optimal output signal for the slowest functions consists of a signal that is constant for all patterns belonging to the same class, in which case the objective function is zero[1], as illustrated in Figure 2a. Signals of this kind can be fully represented by a C-dimensional vector, where each component contains the output value for one of the classes. The zero-mean constraint (Eq. 31) eliminates one degree of freedom, such that in the end all possible optimal signals span a $(C-1)$-dimensional space. Constraints (32) and (33) force successive output signals to build an orthogonal basis of this space. In a simulation it is therefore possible to extract at most $(C-1)$ such signals. We are considering here the interesting case of an unsupervised algorithm for which in the ideal case (i.e. when the function space $\mathcal{F}$ is large enough) the output signals are known in advance. The feature space is very small ($C-1$ dimensions) and consists of C sets of superimposing points (one for each class). In actual applications, of course, the response to the patterns belonging to one class is not exactly constant, but tends to be narrowly distributed around a constant value. (Fig. 2b). The representation in the $(C-1)$-dimensional feature space will thus consist of C clusters (Fig. 2c). Classification can be thus performed with simple methods, such as Gaussian classifiers.

For a given input and a fixed function space, the approach proposed above has no parameters (with the exception of the number of derivatives used to estimate **A** if the number of patterns per class is too high), which is an important advantage with respect to other algorithms that need to be fine-tuned to the considered problem. Moreover, since SFA is based on an eigenvector problem, it finds the global solution in a single iteration and has no convergence problems, in contrast for example to algorithms based on gradient descent that might get trapped in local minima. On the other hand, SFA suffers from the curse of dimensionality, since the size of the covariance matrices **A** and **B** that have to be stored during training grows rapidly with increasing dimensionality of the considered function space, which limits the number of input components (see also Sect. 4.3).

4 Example application

4.1 Methods

We illustrate our method by its application to a digit recognition problem. We consider the MNIST digit database[2], which consists of a standardized and freely available set of 70,000 handwritten digits, divided into a training set (60,000 digits) and a test set (10,000 digits). Each pattern consists of an handwritten digit of size 28×28 pixels (some examples are shown in Fig. 1). Several established pattern recognition methods have been applied to this database by LeCun et al. (1998). Their paper provides a standard reference work to benchmark new algorithms.

[1]In the ideal case one would obtain the optimal signal also by just showing C sequences, each consisting of all patterns belonging to one class. In general, however, if SFA cannot produce a perfectly constant solution, it could find and exploit some structure in the particular patterns succession used during training which would in turn create some artificial structures in the output signals. For this reason one obtains better results by presenting the patterns in all possible combinations as mentioned above.

[2]Available online at `http://yann.lecun.com/exdb/mnist/`, see also Additional material.

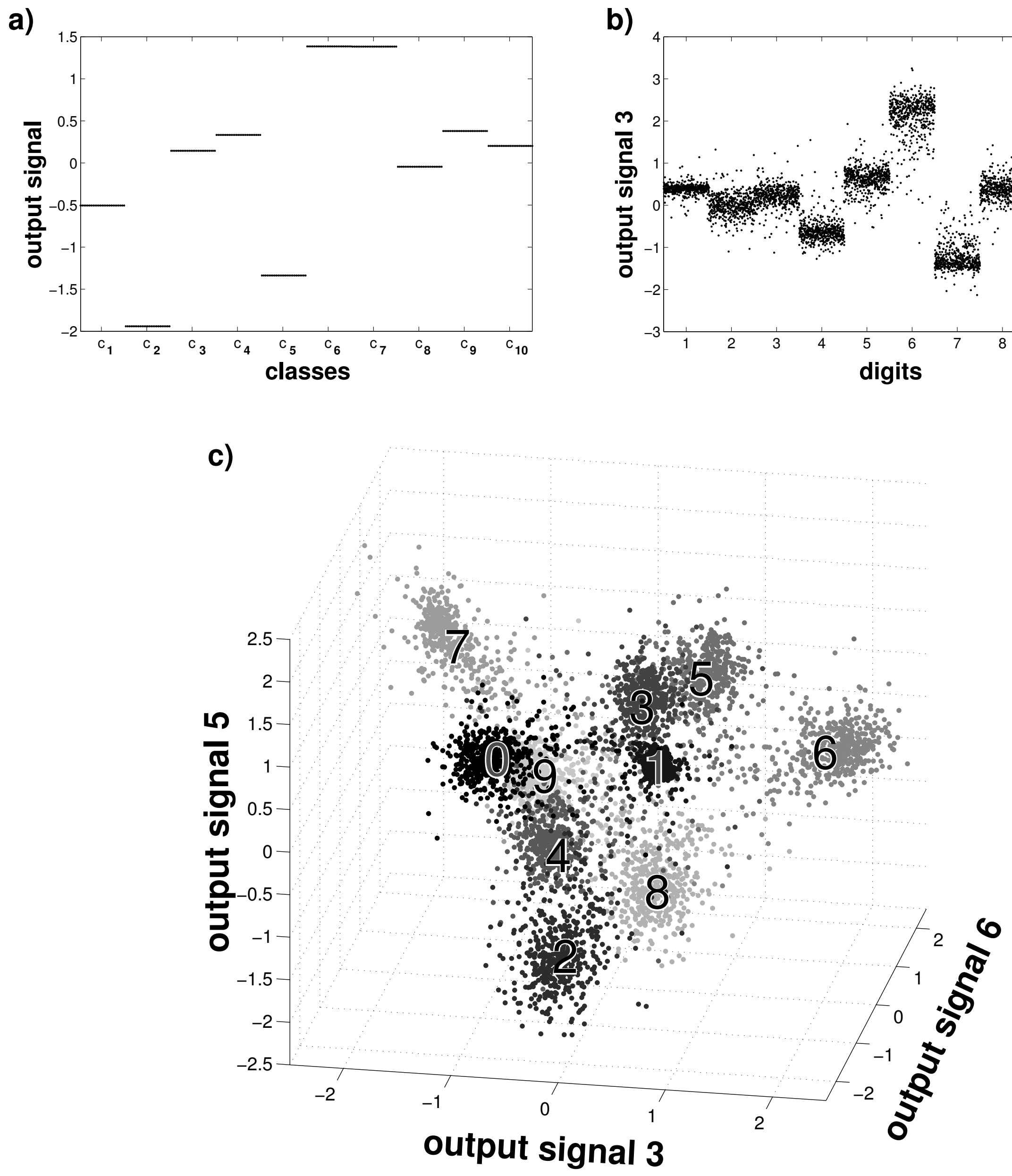

Figure 2 Output signal **(a)** In the ideal case (i.e. when the considered function space is large enough) the optimal output signal is constant for all patterns of a given class. **(b)** Output signal of one of the functions learned in the best among the simulations of Sect. 4, performed on the MNIST database using polynomials of degree 3 and 35 input dimensions. In this plot the function was applied to 500 test digits for each class. Its output for a specific class is narrowly distributed around a constant value. **(c)** Feature space representation given by 3 output signals in the same simulation as in (b). Each class forms a distinct cluster of points. (A color version of this figure can be found at the end of the paper.)

We perform SFA on spaces of polynomials of a given degree d, whose corresponding expansion functions include all monomials up to order d and have $D = \binom{N+d}{d} - 1$ output dimensions[3], where N is the number of variables, i.e. the input dimension. In this very large space we have to compute the two covariance matrices $\mathbf{A}$ and $\mathbf{B}$ which have each D^2 component. It is clear that the problem quickly becomes intractable because of the high memory requirements. For this reason, the input dimensionality N is first reduced by principal component analysis (PCA) from $28 \times 28 = 784$ to a smaller number of dimensions (from 10 to 140).

On the preprocessed data we then apply SFA. We compute the covariance matrix $\mathbf{B}$ using all training patterns, and we approximate $\mathbf{A}$ with 1 million derivatives (100,000 derivatives for each digit class), chosen at random without repetition from the set of all derivatives. As explained in Section 3, since we have 10 classes only the 9 slowest signals should be relevant. This is confirmed by the Δ-values of the learned functions (Eq. 29), which increase abruptly from function g_9 to function g_{10} (Fig. 3). The increase factor $\Delta(y_{10})/\Delta(y_9)$ in our simulations was on average 3.7 . For this reason, we only need to learn the 9 slowest functions.

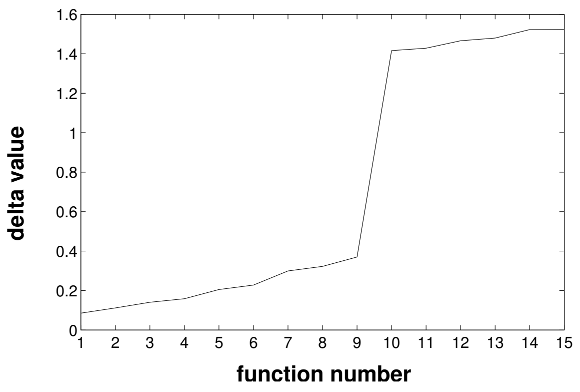

Figure 3 Δ-values Δ-values (Eq. 29) of the first 15 functions in the simulation with polynomials of degree 3 and 35 input dimensions. As explained in Sect. 3, since there are 10 classes only the 9 slowest signals should be relevant. This is confirmed by the abrupt increase in temporal variability at function number 10.

For classification, we apply the 9 functions to the training data of each class separately and fit a Gaussian distribution to their output by computing mean and covariance matrix. In this way we obtain 10 Gaussian distributions, each of which represents the probability $P(\mathbf{y}|c_m)$ that an output vector $\mathbf{y}$ belongs to one of the classes c_m. We then consider the test digits and compute for each of them the probability to belong to each of the classes

$$P(c_m|\mathbf{y}) = \frac{P(\mathbf{y}|c_m)P(c_m)}{\sum_{j=1}^{C} P(\mathbf{y}|c_j)}, \tag{34}$$

where $P(c_m) = P_m/P$ is the probability of occurring of class c_m. Finally we assign the test digit to the class with the highest probability.

4.2 Results

Figure 4 shows the training and test errors for experiments performed with polynomials of degree from 2 to 5 and number of input dimensions from 10 to 140. With polynomials of second degree the explosion in the dimensionality of the expanded space with increasing number of input dimensions is relatively restricted, such that it is possible to perform simulations with up to 140 dimensions, which explain 94% of the total input variance. The test error settles down quickly around 2%, with a minimum at 100 dimensions (1.95%). Simulations performed with higher order polynomials have to rely on a smaller number of input dimensions, but since the function space gets larger and includes new nonlinearities, one obtains a remarkable improvement in performance. Given a fixed dimensionality, the error rate decreases monotonically with increasing degree of the polynomials (Fig. 4).

[3]The number of monomials of degree i is equal to the number of combinations of length i on N elements with repetition, i.e. $\binom{N+i-1}{i}$. The number of output dimensions is thus $\sum_{i=0}^{d}\binom{N+i-1}{i} - 1$, where we subtracted the constant term $i = 0$ which

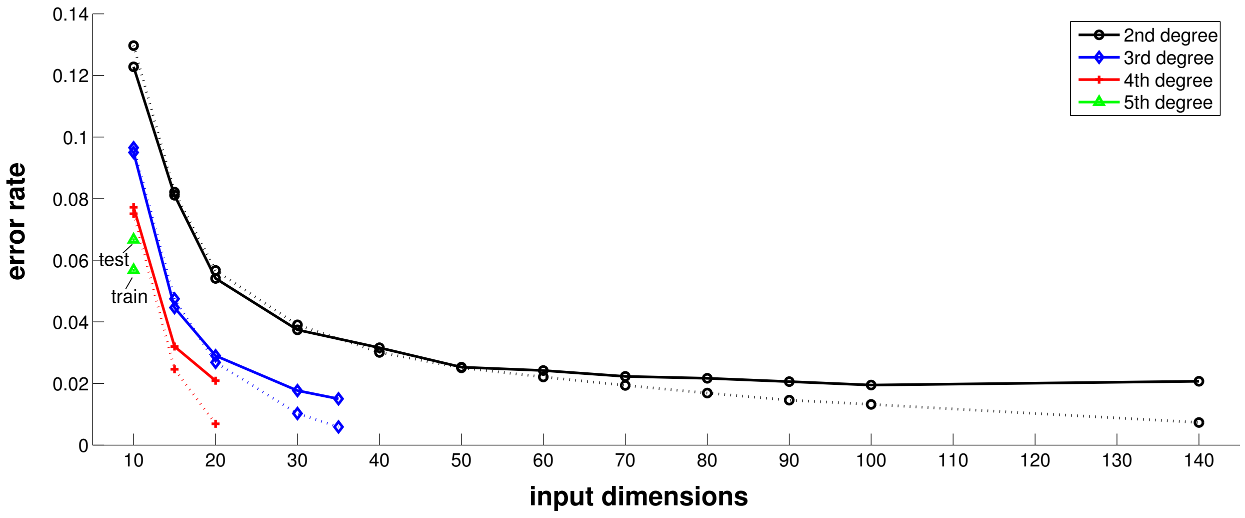

Figure 4 Test and training errors Error rates for simulations performed with polynomials of degree 2 to 5 and number of input dimensions from 10 to 140. Dotted and solid lines represent training and test error rate, respectively. For polynomials of degree 5, the error rate for test and training data are indicated by a label.

The best performance in our simulations is achieved with polynomials of degree 3 and 35 input dimensions, with an error rate of 1.5% on test data. Figure 2b shows the output of one of the functions learned in this simulation when applied to 500 test digits for each class. For each individual class the signal is narrowly distributed around a constant value, and approximates the optimal solution shown in Figure 2a. Some of the digit classes (e.g. 1, 2, and 3) have similar output values and cannot be distinguished by looking at this signal alone. However, other functions represent those classes in different ways and considering all signals together it is possible to separate them. For example, Figure 2c shows the feature space representation given by 3 output signals. It is possible to see that each class forms a distinct cluster. Considering the whole 9-dimensional feature space improves the separation further. Figure 5 shows all 146 digits that were misclassified out of the 10,000 test patterns. Some of them seem genuinely ambiguous, while others would have been classified correctly by a human. The reduction of the input dimensionality by PCA is partly responsible for these error, since it erases some of the details and makes some patterns difficult to recognize, as illustrated in Figure 6.

Table 1 compares the error rates on test data of different algorithms presented in (LeCun et al., 1998) with that of SFA. The error rate of SFA is comparable to but does not outperform that of the most elaborate algorithms. Yet its performance is remarkable considering the simplicity of the method and the fact that it has no a priori knowledge on the problem, in contrast for example to the LeNet-5 algorithm which has been designed specifically for handwritten character recognition. In addition, for recognition, our method has to store and compute only 9 functions and has thus small memory requirements and a high recognition speed.

The comparison with the Tangent Distance (TD) algorithm (Simard et al., 1993; LeCun et al., 1998; Simard et al., 2000) is particularly interesting. TD is a nearest-neighbor classifier where the distance between two patterns is not computed with the Euclidean norm but with a metric that is made insensitive to local transformations of the patterns which have to be specified a priori. In the case of digit recognition, they might include for example local translation, rotation, scaling, stretching, and thickening or thinning of the image. This method, as most memory-based recognition systems, has very high memory and computational requirements. One can interpret SFA as an algorithm that learns a nonlinear transformation of the input such that the Euclidean distance between patterns belonging to the same class gets as small as possible. A

is fixed by the zero-mean constraint (31). The formula indicated above is then easily proved by induction.

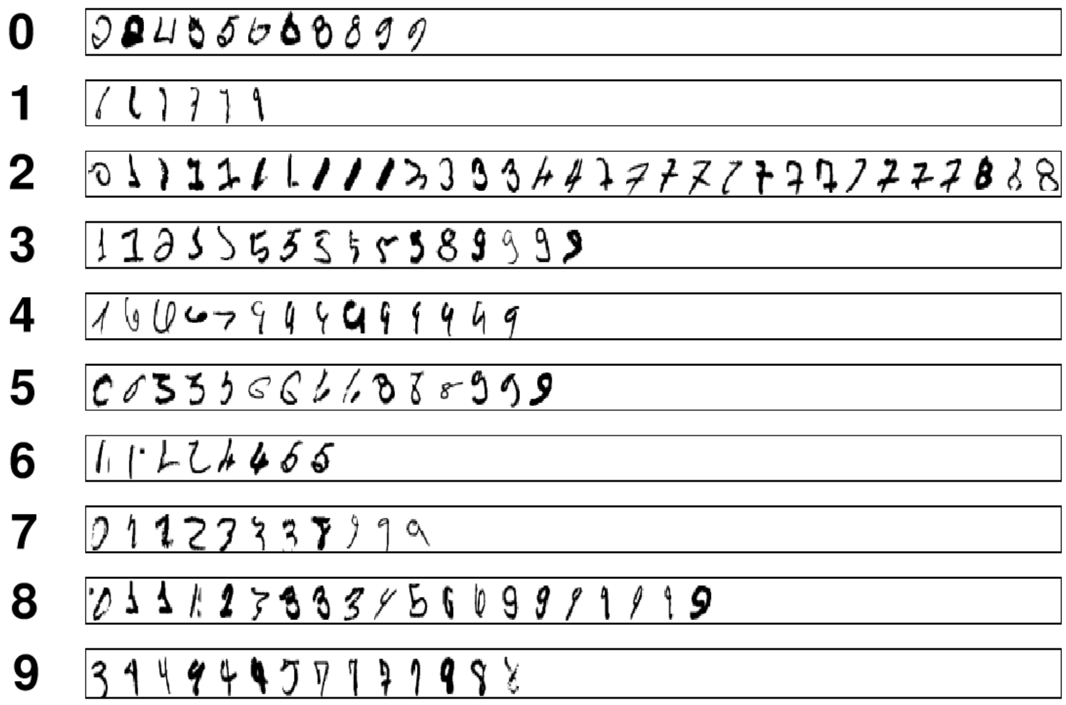

Figure 5 Classification errors This figure shows all the 146 digits that were misclassified out of the 10,000 test patterns in the best among our simulations performed with polynomials of degree 3 and 35 input dimensions. The patterns in the first line were classified as 0, the ones in the second line as 1, etc.

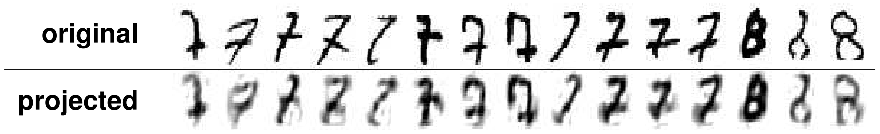

Figure 6 Dimensionality reduction The reduction by PCA of the number of input dimensions might be responsible for some of the classification errors. This figure shows some of the test patterns that have been misclassified as 2. In the top row we plot the original patterns from the MNIST database. In the bottom row we plot the same patterns after a projection onto the first 35 principal components. Due to dimensionality reduction some of the patterns become more ambiguous.

METHOD	% ERRORS
Linear classifier	12.0
K-Nearest-Neighbors	5.0
1000 Radial Basis Functions, linear classifier	3.6
Best Back-Propagation NN	2.95
(3 layers with 500 and 150 hidden units)	
Reduced Set SVM (5 deg. polynomials)	1.0
LeNet-1 (16×16 input)	1.7
LeNet-5	0.95
Tangent Distance (16×16 input)	1.1
Slow Feature Analysis	**1.5**
(3 deg. polynomials, 35 input dim)	

Table 1 Performance comparison Error rates on test data of various algorithms. All error rates are taken from (LeCun et al., 1998).

natural objective function for this would be, defining $\mathbf{g}$ as the vector of all learned functions $\mathbf{g} = (g_1, \dots g_K)$:

$$a \cdot \sum_{m=1}^{C} \sum_{\substack{k,l=1 \\ k<l}}^{P_m} \left\| \mathbf{g}(\mathbf{p}_k^{(m)}) - \mathbf{g}(\mathbf{p}_l^{(m)}) \right\|^2 \tag{35}$$

$$= a \cdot \sum_{m=1}^{C} \sum_{\substack{k,l=1 \\ k<l}}^{P_m} \left[\sum_{j=1}^{K} \left(g_j(\mathbf{p}_k^{(m)}) - g_j(\mathbf{p}_l^{(m)}) \right)^2 \right] \tag{36}$$

$$= \sum_{j=1}^{K} \left[a \cdot \sum_{m=1}^{C} \sum_{\substack{k,l=1 \\ k<l}}^{P_m} \left(g_j(\mathbf{p}_k^{(m)}) - g_j(\mathbf{p}_l^{(m)}) \right)^2 \right] \tag{37}$$

$$\underset{(29)}{=} \sum_{j=1}^{K} \Delta(y_j)\,, \tag{38}$$

which has to be minimized under Constraints (2–4). For a given K, the objective function (38) and that of SFA for pattern recognition (29) are identical, except that in the former case the average of the temporal variation over all K learned functions is minimized, while in the latter the functions are optimized one after the other inducing an order. The set of functions that minimizes the SFA objective function (Eq. 29) also minimizes Equation (38). Furthermore, all other solutions to (38) are orthogonal transformations of this special one (a sketch of the proof is given in Appendix A). The solution found by SFA is particular in that it is independent from the choice of K, in the sense that the components are ordered such that it is possible to compute the global solution first and then reduce the number of dimensions afterwards as needed.

In conclusion, while the TD algorithm keeps the input representation fixed and applies a special metric, SFA transforms the input space such that the TD-metric gets similar to the Euclidean one. As already mentioned, the new representation has only few dimensions (since K in Eq. 38 can be set to $K = C - 1$, see Sect. 3), which decreases memory and computational requirements dramatically, and can be easily learned from the input data without a priori knowledge on the transformations.

4.3 Extensions

In this example application we tried to keep the pattern recognition system as basic as possible in order to show the simplicity and effectiveness of SFA. Of course, a number of standard techniques could be applied to improve its performance. A trivial way to improve the performance would be to use a computer with more memory (our system had 1.5GByte RAM) and use a higher number of input dimensions and/or polynomials of higher degree. This approach would however rapidly reach its limits, since the dimensionality of the matrices **A** and **B** grows exponentially, as discussed above. An important improvement to our method would be to find an algorithm that performs a preliminary reduction of the dimensionality of the expanded space by discarding directions of high temporal variation. Note that it is theoretically not possible to reduce the dimensionality of the expanded space by PCA or by any other method that does not take into account the derivatives of the expanded signal, as suggested in (Bray and Martinez, 2002) and (Hashimoto, 2003), since there is in general no simple relation between the spatial and the temporal statistics of the expanded signal (in the case of PCA the solution would even depend on the scaling of the basis functions). If we consider for example the simulation with polynomials of degree 3 and 35 input dimensions and reduce the dimensionality of the expanded space by PCA down to one half (which is still an optimistic scenario since in general the reduction has to be more drastic), the error rate shows a substantial increase from 1.5% to 2.5%. A viable solution if the length of the data set is small (which is not the case in our example application nor usually in real-life problems) is to use the standard *kernel trick* and compute the covariance matrices on the temporal dimensions instead of in space (see (Müller et al., 2001; Bray and Martinez, 2002)). In this case it is even possible to use an infinite-dimensional function space if it permits a kernel function (i.e. if it satisfies Mercer's Condition, see (Burges, 1998)).

Other standard machine learning methods that have been applied to improve the performance of pattern recognition algorithms, like for example a more problem-specific preprocessing of the patterns, boosting techniques, or mixture of experts with other algorithms (Bishop, 1995; LeCun et al., 1998) might be applied in our case as well. Some of the simulations in (LeCun et al., 1998) were performed on a training set artificially expanded by applying some distortions to the original data, which improved the error rate. Such a strategy might be particularly effective with SFA since it might help to better estimate the covariance matrix **A** of the transformations between digits.

Finally, the Gaussian classifier might be substituted by some more enhanced supervised classifier. However, in this case we would not expect a particularly dramatic improvement in performance, due to the simple form taken by the representation in the feature space both in the ideal case and in simulations (Fig. 2).

5 Conclusions

In this paper we described the application of SFA to pattern recognition. The presented method is problem-independent and for a given input signal and a fixed function space it has no parameters. In an example application it yields an error rate comparable to that of other established algorithms. The learned feature space has a very small dimensionality, such that classification can be performed efficiently in terms of speed and memory requirements.

Since most pattern recognition problems do not have a temporal structure it is necessary to present the training set in the way described in Section 3. If instead the input does have a temporal structure (e.g. when classifying objects that naturally enter and leave a visual scene), it is possible to apply SFA directly on the input data. By applying more elaborated unsupervised clustering techniques for classification it should be thus possible to achieve totally unsupervised pattern recognition (i.e. without supervision from perception to classification).

6 Additional material

Source code and data to reproduce the simulations in this paper are available online at the author's homepage. The MNIST database can be found at `http://yann.lecun.com/exdb/mnist/`.

7 Acknowledgments

This work has been supported by a grant from the Volkswagen Foundation. Many thanks to Laurenz Wiskott and Henning Sprekeler for useful comments on this manuscript.

Appendix

A The solutions of Equation (38)

In this appendix we prove that the functions $g_1, \ldots, g_K$ that minimize Equation (38) under Constraints (2–4) are identical up to an orthogonal transformation with the SFA solutions that minimize Equation (1) under the same constraints.

Without loss of generality we consider only the linear case

$$g_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x} \tag{39}$$

and we assume that the input vectors $\mathbf{x}$ are whitened, such that $\mathbf{B} = \langle \mathbf{x}\mathbf{x}^T \rangle_t = \mathbf{I}$, which can be obtained with an orthogonal (but in general not orthonormal) transformation. Under this conditions Constraints (2–4) simply mean that the filter $\mathbf{w}_j$ must be orthogonal (cf. Eqs. 16–17). Moreover, the generalized eigenvalue problem (15) becomes a standard eigenvalue problem

$$\mathbf{A}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}\,, \tag{40}$$

such that the SFA solutions correspond to the eigenvectors with the smallest eigenvalues.

On the other hand, Equation (38) can be written as

$$\sum_{j=1}^{K} \Delta(y_j) \underset{(10)}{=} \sum_{j=1}^{K} \mathbf{w}_j^T \mathbf{A} \mathbf{w}_j\,. \tag{41}$$

Minimizing this equation involves computing the Lagrange multipliers of (41) under the conditions

$$\mathbf{w}_i^T \mathbf{w}_j = \delta_i^j\,, \quad \forall i, j < K\,, \tag{42}$$

where δ_i^j is the Kronecker delta. The complete calculation is reported in (Bishop, 1995, Appendix E) for an equivalent PCA theorem. The solution is found to be an arbitrary orthogonal transformation of the eigenvectors of $\mathbf{A}$ corresponding to the smallest eigenvalues, i.e. an arbitrary orthogonal transformation of the special SFA solution.

References

Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press. 11, 12

Bray, A., Martinez, D., 2002. Kernel-based extraction of Slow Features: Complex cells learn disparity and translation invariance from natural images. In: NIPS 2002 proceedings. 11

Burges, C. J. C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2 (2), 121–167. 11

Gantmacher, F. R., 1959. Matrix Theory. Vol. 1. AMS Chelsea Publishing. 3

Hashimoto, W., 2003. Quadratic forms in natural images. Network: Computation in Neural Systems 14 (4), 765–788. 11

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324. 5, 8, 10, 11

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B., 2001. An Introduction to Kernel–Based Learning Algorithms. IEEE Transactions on Neural Networks 12 (2), 181–202. 11

Simard, P., LeCun, Y., Denker, J., 1993. Efficient pattern recognition using a new transformation distance. In: Hanson, S., Cowan, J., Giles, L. (Eds.), Advances in Neural Information Processing Systems. Vol. 5. Morgan Kaufmann. 8

Simard, P. Y., LeCun, Y., Denker, J. S., Victorri, B., 2000. Transformation invariance in pattern recognition – tangent distance and tangent propagation. International Journal of Imaging Systems and Technology 11 (3). 8

Wiskott, L., 1998. Learning invariance manifolds. In: Niklasson, L., Bodén, M., Ziemke, T. (Eds.), Proc. Intl. Conf. on Artificial Neural Networks, ICANN'98, Skövde. Perspectives in Neural Computing. Springer, pp. 555–560. 1

Wiskott, L., Sejnowski, T., 2002. Slow feature analysis: Unsupervised learning of invariances. Neural Computation 14 (4), 715–770. 1, 2, 4

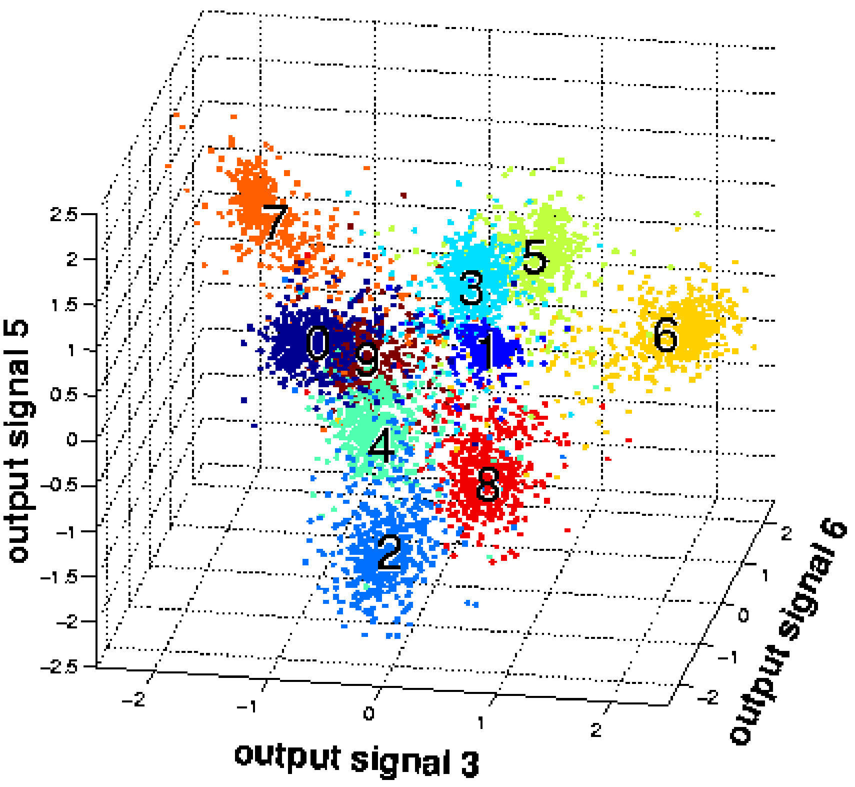

Figure 2c (color version, see page 6 for figure caption).