Cognitive Sciences EPrint Archive (CogPrints) 8966, (2013)
http://cogprints.org/8966/

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

Theory of Neural Systems
Institut für Neuroinformatik
Ruhr-University of Bochum
Bochum D-44801, Germany

Abstract

Supervised learning from high-dimensional data, e.g., 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, high-dimensional data, implicitly supervised, 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, e.g., 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, frequently resulting in lower performance and higher computational requirements.

Unsupervised dimensionality reduction, including algorithms such as PCA or 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 limited.

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 and Sejnowski, 2002) is an unsupervised learning algorithm inspired by the visual system and based on the slowness principle. Various methods have allowed the application of SFA to classification. 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. Later, Klampfl and Maass (2010) studied the use of a Markov chain to generate a sequence used to train SFA. The transition probability between samples from different object identities was proportional to a small parameter a. The authors showed that in the limit $a \to 0$, 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 (e.g., 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). Measuring and amplifying the difference between delta values for different training signals, an elaborated system for human action recognition was proposed by Zhang and Tao (2012).

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. In this particular setup, the position of the fish was changed at a relatively slow pace. SFA was thus trained in a completely unsupervised way. A number of extracted slow features were then post-processed with 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 a generalized concept of signal slowness formalized by a weighted SFA optimization problem described below.

The objective function of this optimization problem is a weighted sum of squared output differences, and is therefore similar to the underlying objective functions of, for example, FDA, LFDA, SELF, PCGFP, and SSDR. However, in general the optimization problem solved by GSFA differs 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 kernelized versions of the algorithms above can be defined, one has to overcome the difficulty of finding a good kernel. In contrast, SFA (and GSFA) was conceived from the beginning as a nonlinear algorithm without resorting to kernels, with linear SFA being just a less used special case. Another difference with various algorithms above is that SFA (and GSFA) does not explicitly attempt to preserve the structure of the data. Interestingly, features equivalent to those of FDA can be obtained if particular training data is given to GSFA.

There is a close relation between SFA and Laplacian eigenmaps (LE), which has been studied by Sprekeler (2011). GSFA and LE 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.

One advantage of GSFA is that it is designed for classification and regression, while most algorithms for supervised dimensionality reduction, including the ones above, focus on classification only.

The central idea behind GSFA is to encode the label information implicitly in the structure of the input data, as some type of similarity matrix called edge-weight matrix, to indirectly solve the supervised learning problem, rather than performing an explicit fit to the labels. The full input data, called training graph, then takes the form of a weighted graph with vertices representing the data points (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 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. An illustration of the method is provided in Figure 1. This has several advantages that originate from SFA:

- The problem of dimensionality reduction is technically formulated as an implicitly supervised learning problem, because the labels are not given explicitly as a target value but implicitly within the structure of the input data. This easily permits hierarchical processing, which is a divide-and-conquer approach to computing slow features and can be seen as a form of regularization. A smaller total number of weights frequently results in less overfitting (compared to non-hierarchical SFA).

- SFA has a complexity of $\mathcal{O}(N)$ in the number of samples N and $\mathcal{O}(I^3)$ in the number of dimensions I (possibly after a nonlinear expansion). Hierarchical processing greatly reduces the later complexity down to $\mathcal{O}(I \log I)$. In practice, processing 100,000 samples of 10,000-dimensional input data can be done in less than three hours by using hierarchical SFA without resorting to parallelization or GPU computing.
- The SFA algorithm itself is 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. Typically no expensive parameter search is required.

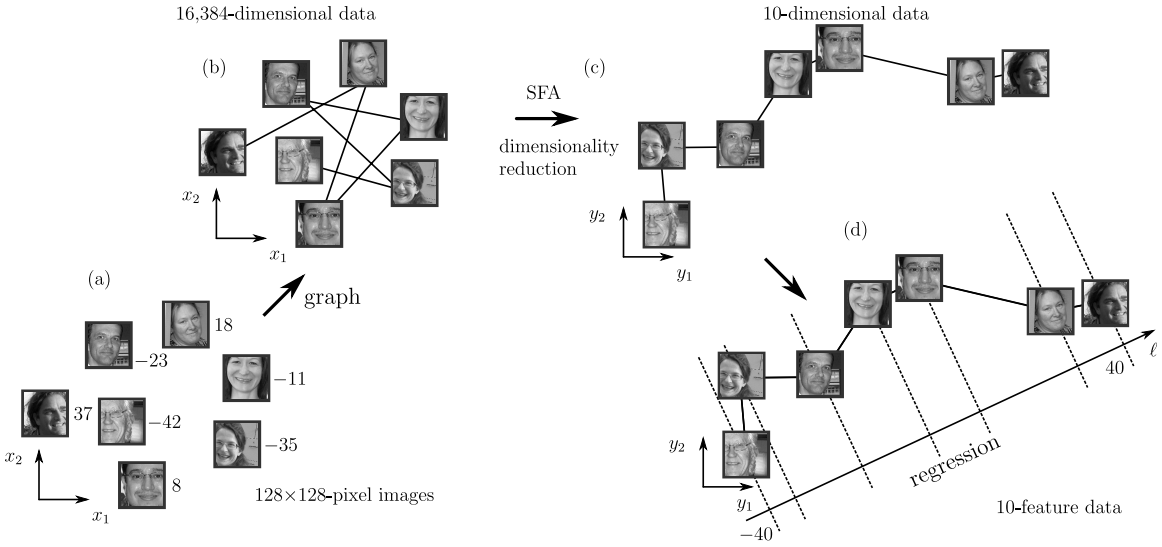

Figure 1: Illustration of the use of GSFA to solve a regression problem. An input sample is an image represented as a 16384-dimensional vector, $\mathbf{x}$, and yields a 2-dimensional feature vector $\mathbf{y}$. (a) Original images and their labels indicating the horizontal position of the face center. (b) A graph with appropriate pairwise connections between the samples is created. Here only images with most similar labels are connected resulting in a linear graph. (c) Example of the low-dimensional GSFA output. Notice how the first slow component, y_1, is strongly related to the original label. (d) The result of regression, thus, ideally depends only on y_1.

In the next sections, we first recall the standard SFA optimization problem and algorithm. Then, we introduce a weighted SFA optimization problem that accounts for 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

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 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 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 (Escalante-B. and Wiskott, 2012).

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 $\mathbf{g} : \mathbb{R}^I \to \mathbb{R}^J$, i.e., $\mathbf{g}(\mathbf{x}(t)) = (g_1(\mathbf{x}(t)), \ldots, g_J(\mathbf{x}(t)))^T$, within a function space $\mathcal{F}$ such that for each component $y_j(t) \stackrel{\text{def}}{=} g_j(\mathbf{x}(t))$ of the output signal $\mathbf{y}(t) \stackrel{\text{def}}{=} \mathbf{g}(\mathbf{x}(t))$, for $1 \le j \le J$, the objective function

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

under the constraints

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

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

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

The delta value $\Delta(y_j)$ is defined as the time average ($\langle\cdot\rangle_t$) of the squared derivative of y_j and is therefore a measure of the slowness (or rather fastness) of the signal. The constraints (2–4) 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 subsection). Due to constraint (4), the delta values are ordered, i.e., $\Delta(y_1) \leq \Delta(y_2) \leq \cdots \leq \Delta(y_J)$. See Figure 2 for an illustrative example.

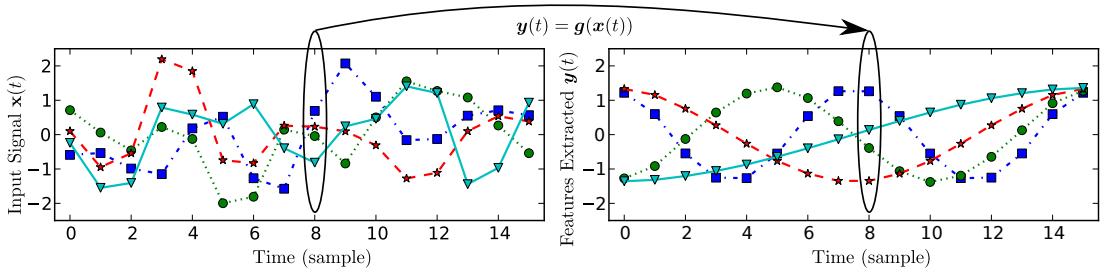

Figure 2: Illustrative example of feature extraction from a 10-dimensional (discrete time) input signal. Four arbitrary components of the input (left) and the four slowest outputs (right) are shown. Notice that feature extraction is an instantaneous operation, even though the outputs are slow over time. 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 $\mathcal{F}$, e.g., to all quadratic or linear functions. In theoretical analyses (e.g., Wiskott, 2003) it is common to use an unrestricted function space.

2.3 Standard linear SFA algorithm

The SFA algorithm is typically nonlinear. Even though a kernelized version has been proposed (Bray and Martinez, 2003; Vollgraf and Obermayer, 2006), 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 $\langle(\mathbf{x}-\mu_x)(\mathbf{x}-\mu_x)^T\rangle_t$ is approximated by the sample covariance matrix

$$\mathbf{C} = \frac{1}{N-1}\sum_{t=1}^{N}(\mathbf{x}(t)-\bar{\mathbf{x}})(\mathbf{x}(t)-\bar{\mathbf{x}})^T\,, \tag{5}$$

and the derivative second-moment matrix $\langle\dot{\mathbf{x}}\dot{\mathbf{x}}^T\rangle_t$ is approximated as

$$\dot{\mathbf{C}} = \frac{1}{N-1}\sum_{t=1}^{N-1}(\mathbf{x}(t+1)-\mathbf{x}(t))(\mathbf{x}(t+1)-\mathbf{x}(t))^T. \tag{6}$$

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 $\Delta(y_j)$, for $1 \leq j \leq J$. The linear SFA algorithm is guaranteed to find an optimal solution to the optimization problem in the linear function space (Wiskott and Sejnowski, 2002), e.g., the first component extracted is the slowest possible linear feature.

The complexity of the linear SFA algorithm described above is $\mathcal{O}(NI^2 + I^3)$ where N is the number of samples and I is the input dimensionality (possibly after a nonlinear expansion), thus for high-dimensional data standard SFA is not feasible[1]. In practice, it has a speed comparable to PCA, even though SFA also takes into account the temporal structure of the data.

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 because the number of parameters to be learned is typically smaller than if a single SFA node with a huge input is used, 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).

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.

3. Graph-Based SFA (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 3) 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

$$\gamma_{n,n'} = \gamma_{n',n} \tag{7}$$

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 3.b). How exactly edge weights are derived from label values will be elaborated later on.

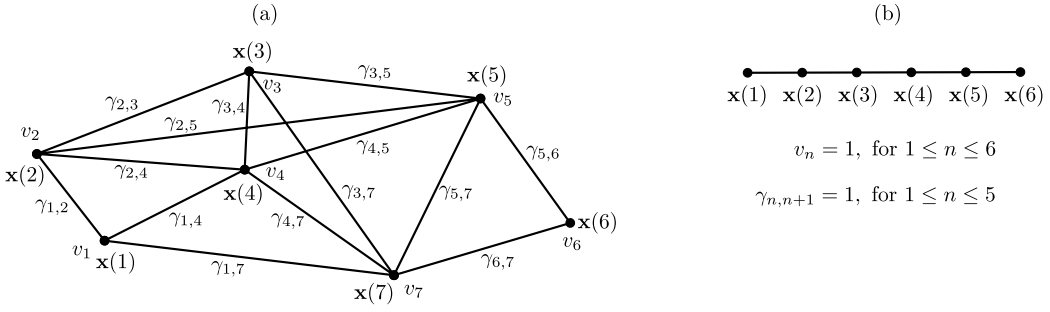

Figure 3: (a) Example of a training graph with $N = 7$ vertices. (b) A regular sample sequence (time-series) represented as a linear graph suitable for GSFA.

3.2 Weighted SFA 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

$$\Delta_j \stackrel{\text{def}}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} (y_j(n') - y_j(n))^2 \text{ is minimal} \quad \textbf{(weighted delta value)} \tag{8}$$

under the constraints

$$\frac{1}{Q} \sum_n v_n y_j(n) = 0 \textbf{ (weighted zero mean)}, \tag{9}$$

$$\frac{1}{Q} \sum_n v_n (y_j(n))^2 = 1 \textbf{ (weighted unit variance)}, \text{ and} \tag{10}$$

$$\frac{1}{Q} \sum_n v_n y_j(n) y_{j'}(n) = 0, \text{ for } j' < j \textbf{ (weighted decorrelation)}, \tag{11}$$

with

$$R \stackrel{\text{def}}{=} \sum_{n,n'} \gamma_{n,n'}, \tag{12}$$

$$Q \stackrel{\text{def}}{=} \sum_n v_n. \tag{13}$$

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, but it is useful for normalization purposes. Likewise, the factor $1/Q$ can be dropped from (9) and (11).

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 weighted optimization problem above. Its objective function (8) 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 (8) 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 weighted SFA 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:

$$\mathbf{C_G} \stackrel{\text{def}}{=} \frac{1}{Q}\sum_n v_n(\mathbf{x}(n)-\hat{\mathbf{x}})(\mathbf{x}(n)-\hat{\mathbf{x}})^T = \frac{1}{Q}\sum_n \left(v_n\mathbf{x}(n)(\mathbf{x}(n))^T\right) - \hat{\mathbf{x}}(\hat{\mathbf{x}})^T\,, \tag{14}$$

where

$$\hat{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{Q}\sum_n v_n\mathbf{x}(n) \tag{15}$$

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

$$\dot{\mathbf{C}}_\mathbf{G} \stackrel{\text{def}}{=} \frac{1}{R}\sum_{n,n'} \gamma_{n,n'}\big(\mathbf{x}(n')-\mathbf{x}(n)\big)\big(\mathbf{x}(n')-\mathbf{x}(n)\big)^T\,. \tag{16}$$

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

$$\mathbf{S}^T\mathbf{C_G}\mathbf{S} = \mathbf{I}\,, \text{ and} \tag{17}$$

$$\mathbf{R}^T\mathbf{S}^T\dot{\mathbf{C}}_\mathbf{G}\mathbf{S}\mathbf{R} = \mathbf{\Lambda}\,, \tag{18}$$

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

$$\mathbf{W} = \mathbf{S}\mathbf{R}\,, \text{ and} \tag{19}$$

$$\mathbf{y}(n) = \mathbf{W}^T(\mathbf{x}(n)-\hat{\mathbf{x}})\,. \tag{20}$$

3.4 Correctness of the graph-based SFA algorithm

We now prove that the GSFA algorithm indeed solves the optimization problem (8–11). 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 (9) holds trivially for any $\mathbf{W}$, because

$$\sum_n v_n\mathbf{y}(n) \stackrel{(20)}{=} \sum_n v_n\mathbf{W}^T(\mathbf{x}(n)-\hat{\mathbf{x}}) \tag{21}$$

$$= \mathbf{W}^T\left(\sum_n v_n\mathbf{x}(n) - \sum_{n'} v_{n'}\hat{\mathbf{x}}\right) \tag{22}$$

$$\stackrel{(15,13)}{=} \mathbf{W}^T\left(Q\,\hat{\mathbf{x}} - Q\,\hat{\mathbf{x}}\right) = \mathbf{0}\,. \tag{23}$$

We also find

$$\begin{aligned}
\mathbf{I} &= \mathbf{R}^T\mathbf{I}\mathbf{R} \quad \text{(since } \mathbf{R} \text{ is a rotation matrix)}, && (24)\\
&\overset{(17)}{=} \mathbf{R}^T(\mathbf{S}^T\mathbf{C_G}\mathbf{S})\mathbf{R}\,, && (25)\\
&\overset{(19)}{=} \mathbf{W}^T\mathbf{C_G}\mathbf{W}\,, && (26)\\
&\overset{(14)}{=} \mathbf{W}^T\frac{1}{Q}\sum_n v_n(\mathbf{x}(n)-\hat{\mathbf{x}})(\mathbf{x}(n)-\hat{\mathbf{x}})^T\mathbf{W}\,, && (27)\\
&\overset{(20)}{=} \frac{1}{Q}\sum_n v_n\mathbf{y}(n)(\mathbf{y}(n))^T\,, && (28)
\end{aligned}$$

which is equivalent to the normalization constraints (10) and (11).

Now, let us consider the objective function

$$\begin{aligned}
\Delta_j &\overset{(8)}{=} \frac{1}{R}\sum_{n,n'}\gamma_{n,n'}\big(y_j(n')-y_j(n)\big)^2 && (29)\\
&\overset{(16)}{=} \mathbf{w}_j^T\dot{\mathbf{C}}_\mathbf{G}\mathbf{w}_j && (30)\\
&\overset{(19)}{=} \mathbf{r}_j^T\mathbf{S}^T\dot{\mathbf{C}}_\mathbf{G}\mathbf{S}\mathbf{r}_j && (31)\\
&\overset{(18)}{=} \lambda_j\,, && (32)
\end{aligned}$$

where $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_J)$. The algorithm finds a rotation matrix $\mathbf{R}$ solving (18) and yielding increasing λs. It can be seen (c.f. 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 (8).

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. Given a training graph, we construct below a Markov chain $\mathcal{M}$ for the generation of input data such that the application of standard SFA to the input data yields the same features as GSFA does for the graph.

In order for this equivalence to hold, the vertex weights $\tilde{v}_n$ and edge weights $\tilde{\gamma}_{n,n'}$ of the graph must fulfil the following *normalization restrictions*:

$$\begin{aligned}
\sum_n \tilde{v}_n &= 1\,, && (33)\\
\sum_{n'} \tilde{\gamma}_{n,n'}/\tilde{v}_n &= 1 \quad \forall n\,, && (34)\\
\overset{(7)}{\Longleftrightarrow} \quad \sum_{n'} \tilde{\gamma}_{n',n}/\tilde{v}_n &= 1 \quad \forall n\,, && (35)\\
\sum_{n,n'} \tilde{\gamma}_{n,n'} &\overset{(34,33)}{=} 1\,. && (36)
\end{aligned}$$

Restrictions (33) and (36) can 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. The fundamental restriction is (34), which 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 stationary distribution

$$\boldsymbol{\pi} \;=\; \boldsymbol{p}_1 \;\stackrel{\text{def}}{=}\; \Pr(\mathbf{Z}_1 = \mathbf{x}(n)) \;\stackrel{\text{def}}{=}\; \tilde{v}_n\,, \tag{37}$$

and the transition probabilities are given by

$$P_{nn'} \;\stackrel{\text{def}}{=}\; \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n') | \mathbf{Z}_t = \mathbf{x}(n)) \;\stackrel{\text{def}}{=}\; (1-\epsilon)\tilde{\gamma}_{n,n'}/\tilde{v}_n + \epsilon\tilde{v}_{n'} \underset{\lim_{\epsilon\to 0}}{=} \tilde{\gamma}_{n,n'}/\tilde{v}_n\,, \tag{38}$$

$$\stackrel{(37)}{\Longrightarrow} \quad \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n'), \mathbf{Z}_t = \mathbf{x}(n)) \;=\; (1-\epsilon)\tilde{\gamma}_{n,n'} + \epsilon\tilde{v}_n\tilde{v}_{n'} \underset{\lim_{\epsilon\to 0}}{=} \tilde{\gamma}_{n,n'}\,, \tag{39}$$

(for $\mathbf{Z}_t$ stationary) with $0 < \epsilon \ll 1$. Due to the ϵ-term all states of the Markov chain can transition to all other states including themselves, which makes the Markov chain irreducible and aperiodic, and therefore ergodic. Thus, the stationary distribution is unique and the Markov chain converges to it. The normalization restrictions (33), (34), and (36) ensure the normalization of (37), (38), and (39), respectively.

It is easy to see that $\boldsymbol{\pi} = \tilde{v}_n$ is indeed a stationary distribution, since for $\mathbf{p}_t = \tilde{v}_n$

$$\mathbf{p}_{t+1} \;=\; \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n)) \;=\; \sum_{n'} \Pr(\mathbf{Z}_{t+1} = \mathbf{x}(n) | \mathbf{Z}_t = \mathbf{x}(n')) \Pr(\mathbf{Z}_t = \mathbf{x}(n')) \tag{40}$$

$$\stackrel{(37,38)}{=} \sum_{n'} \left((1-\epsilon)\left(\tilde{\gamma}_{n',n}/\tilde{v}_{n'}\right) + \epsilon\tilde{v}_n\right)\tilde{v}_{n'} \tag{41}$$

$$\stackrel{(35,33)}{=} (1-\epsilon)\tilde{v}_n + \epsilon\tilde{v}_n \;=\; \tilde{v}_n \;=\; \mathbf{p}_t\,. \tag{42}$$

The time average of the input sequence is

$$\boldsymbol{\mu}_Z \;\stackrel{\text{def}}{=}\; \langle \mathbf{Z}_t \rangle_t \tag{43}$$

$$=\; \langle \mathbf{Z} \rangle_\pi \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{44}$$

$$\stackrel{(42)}{=} \sum_n \tilde{v}_n \mathbf{x}(n) \tag{45}$$

$$\stackrel{(15)}{=} \hat{\mathbf{x}}\,, \tag{46}$$

and the covariance matrix is

$$\mathbf{C} \;\stackrel{\text{def}}{=}\; \langle (\mathbf{Z} - \boldsymbol{\mu}_Z)(\mathbf{Z} - \boldsymbol{\mu}_Z)^T \rangle_t \tag{47}$$

$$\stackrel{(46)}{=} \langle (\mathbf{Z} - \hat{\mathbf{x}})(\mathbf{Z} - \hat{\mathbf{x}})^T \rangle_\pi \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{48}$$

$$\stackrel{(42)}{=} \sum_n \tilde{v}_n (\mathbf{x}(n) - \hat{\mathbf{x}})(\mathbf{x}(n) - \hat{\mathbf{x}})^T \tag{49}$$

$$\stackrel{(14)}{=} \mathbf{C}_\mathbf{G}\,, \tag{50}$$

whereas the derivative covariance matrix is

$$\dot{\mathbf{C}} \stackrel{\text{def}}{=} \langle \dot{\mathbf{Z}}_t \dot{\mathbf{Z}}_t^T \rangle_t \tag{51}$$

$$= \langle \dot{\mathbf{Z}} \dot{\mathbf{Z}}^T \rangle_\pi \quad \text{(since } \mathcal{M} \text{ is ergodic)} \tag{52}$$

$$\stackrel{(39)}{=} \sum_{n,n'} \left((1-\epsilon)\tilde{\gamma}_{n,n'} + \epsilon \tilde{v}_n \tilde{v}_{n'} \right) (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T , \tag{53}$$

where $\dot{\mathbf{Z}}_t \stackrel{\text{def}}{=} \mathbf{Z}_{t+1} - \mathbf{Z}_t$. Notice that $\lim_{\epsilon \to 0} \dot{\mathbf{C}} \stackrel{(53)}{=} \tilde{\gamma}_{n,n'} (\mathbf{x}(n') - \mathbf{x}(n))(\mathbf{x}(n') - \mathbf{x}(n))^T \stackrel{(16)}{=} \dot{\mathbf{C}}_{\mathbf{G}}$. Therefore, if a graph fulfils the normalization restrictions (33)–(36), GSFA yields the same features as standard SFA on the sequence generated by the Markov chain, in the limit $\epsilon \to 0$.

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 4). 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, i.e., $\forall s, n : \; v_n^s = 1$. In contrast, edge weights, $\gamma_{n,n'}^s = 1/N_s \quad \forall n, n'$, 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 fulfil the normalization restriction (34). The optimization problem associated to this graph explicitly demands that samples from the same object identity should be typically mapped to similar outputs.

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 $\forall n, n' : \; \gamma_{n,n'}^s = (1/N_s)^2$ instead.

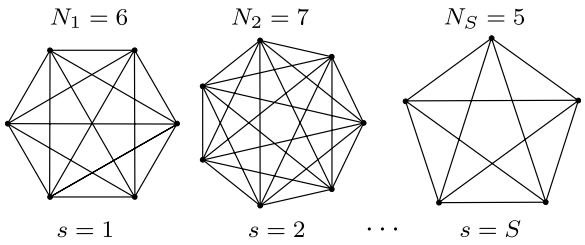

Figure 4: Illustration of a *clustered* training graph used for a classification problem. All samples belonging to the same object identity form fully connected subgraphs. Thus, for S identities there are S complete subgraphs. Self-loops not shown.

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 (14), the sample covariance matrix of this graph using the node weights $v_n^s = 1$ is (notice the definition of Π^s and $\hat{\mathbf{x}}^s$):

$$\mathbf{C}_{\text{clus}} \overset{(14)}{=} \frac{1}{Q}\Bigg(\underbrace{\sum_s \sum_{n=1}^{N_s} \mathbf{x}^s(n)(\mathbf{x}^s(n))^T}_{\overset{\text{def}}{=} \Pi^s} - Q\Big(\underbrace{\frac{1}{Q}\sum_s \overbrace{\sum_{n=1}^{N_s} \mathbf{x}^s(n)}^{\overset{\text{def}}{=} N_s\hat{\mathbf{x}}^s}}_{\overset{(15)}{=}\hat{\mathbf{x}}} \Big)\Big(\frac{1}{Q}\sum_s \sum_{n=1}^{N_s} \mathbf{x}^s(n) \Big)^T \Bigg), \quad (54)$$

$$= \frac{1}{Q}\Big(\sum_s \Pi^s - Q\hat{\mathbf{x}}(\hat{\mathbf{x}})^T \Big), \quad (55)$$

where $Q \overset{(13)}{=} \sum_s \sum_{n=1}^{N_s} 1 = \sum_s N_s = N$.

From (16), the derivative covariance matrix of the clustered training graph using edge weights $\gamma_{n,n'}^s = 1/N_s$ is:

$$\dot{\mathbf{C}}_{\text{clus}} \overset{(16)}{=} \frac{1}{R}\sum_s \frac{1}{N_s} \sum_{n,n'=1}^{N_s} (\mathbf{x}^s(n') - \mathbf{x}^s(n))(\mathbf{x}^s(n') - \mathbf{x}^s(n))^T, \quad (56)$$

$$= \frac{1}{R}\sum_s \frac{1}{N_s} \sum_{n,n'=1}^{N_s} \Big(\mathbf{x}^s(n')(\mathbf{x}^s(n'))^T + \mathbf{x}^s(n)(\mathbf{x}^s(n))^T - \mathbf{x}^s(n')(\mathbf{x}^s(n))^T - \mathbf{x}^s(n)(\mathbf{x}^s(n'))^T \Big), \quad (57)$$

$$\overset{(54)}{=} \frac{1}{R}\sum_s \frac{1}{N_s} \left(N_s \sum_{n=1}^{N_s} \mathbf{x}^s(n)(\mathbf{x}^s(n))^T + N_s \sum_{n'=1}^{N_s} \mathbf{x}^s(n')(\mathbf{x}^s(n'))^T - 2N_s\hat{\mathbf{x}}^s(N_s\hat{\mathbf{x}}^s)^T \right), \quad (58)$$

$$\overset{(54)}{=} \frac{2}{R}\sum_s \left(\Pi^s - N_s\hat{\mathbf{x}}^s(\hat{\mathbf{x}}^s)^T \right), \quad (59)$$

where $R \overset{(12)}{=} \sum_s \sum_{n,n'} \gamma^s_{n,n'} = \sum_s \sum_{n,n'} 1/N_s = \sum_s (N_s)^2/N_s = \sum_s N_s = N$.

The complexity of computing $\mathbf{C}_{\text{clus}}$ using (54) or (55) is the same, namely $\mathcal{O}(\sum_s N_s)$ (vector) operations. However, the complexity of computing $\dot{\mathbf{C}}_{\text{clus}}$ can be reduced from $\mathcal{O}(\sum_s N_s^2)$ operations directly using (56) to $\mathcal{O}(\sum_s N_s)$ operations using (59). 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 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 5).

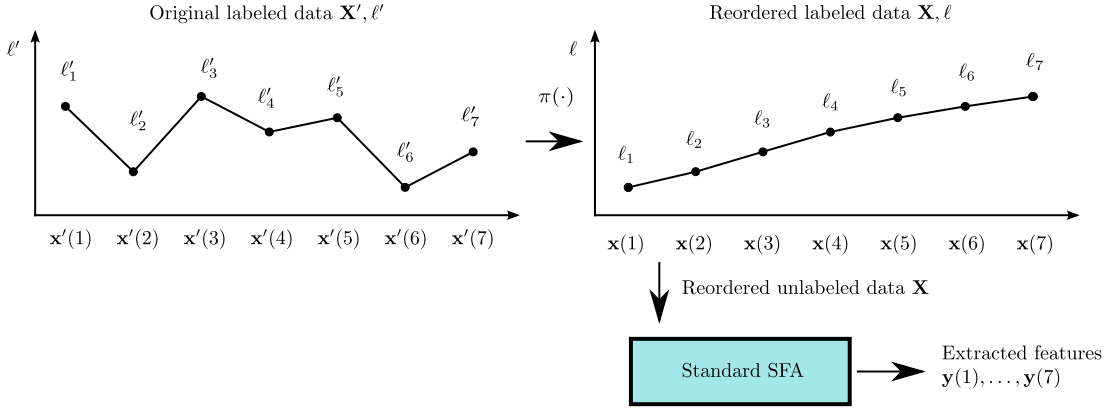

Figure 5: Sample reordering approach. Standard SFA is trained with a reordered sample sequence, in which the hidden labels are increasing.

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 $\mathcal{O}(N)$ 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 this graph, 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 6.a). In this graph, the vertex weights are constant, i.e., $v_n = 1$, and the edge weights 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. In practice, we

employ $\gamma_{n,n'} = 2$ if $(n + n' \leq d + 1$ or $n + n' \geq 2N - 1)$, $\gamma_{n,n'} = 1$ if $|n' - n| \leq d$ but not$(n + n' \leq d + 1$ or $n + n' \geq 2N - 1)$, and $\gamma_{n,n'} = 0$ otherwise. These weights avoid *pathological solutions* that sometimes occur if only weights 0 and 1 were used.

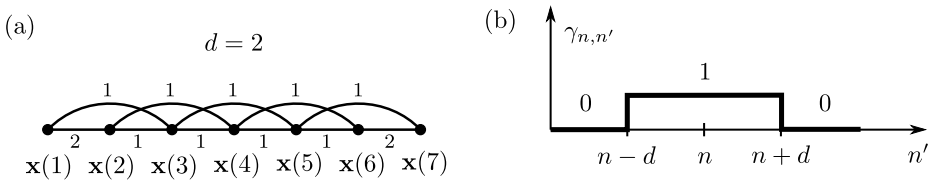

Figure 6: (a) A square sliding window training graph with a half-width of $d = 2$. Each vertex is thus adjacent to at most 4 other vertices. (b) Illustration of the weights of an intermediate node n for a square weight window with a half-width d.

Pathological solutions can occur when a sample is weakly connected to its neighbours. SFA might map such sample to outputs with a large magnitude (to achieve unit variance) and allow for a large difference between its outputs and the outputs of the neighbouring samples (since it is not strongly connected to them). From the normalization constraints of the optimization problem, this results in most samples producing small outputs with almost zero amplitude. Thus, in pathological solutions a large amount of the information about the labels is lost. In practice, these solutions can be prevented, for example, by enforcing the normalization restriction (34) and limiting self-loops on the training graph.

In the sliding window training graph, the computation of $\mathbf{C_G}$ and $\dot{\mathbf{C}}_\mathbf{G}$ requires $\mathcal{O}(dN)$ operations. If the window is square, the computation can be improved to $\mathcal{O}(N)$ operations by using accumulators for sums and products and reusing intermediate results. While larger d implies more connections, 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, e.g., $3 \leq L \ll N$.

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 $\gamma^{l,l+1}_{n,n'}$, and we use the same edge weight for all connections: $\forall n, n', l: \ \gamma^{l,l+1}_{n,n'} = 1$. 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 7). The reason for the different weights on the first and last groups is to avoid pathological solutions and to enforce the normalization restriction (34). 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.

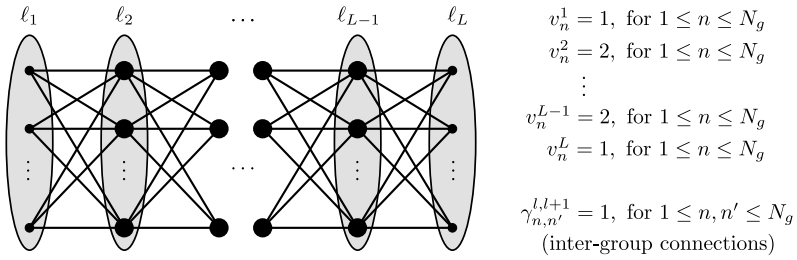

Figure 7: Illustration of a serial training graph with L discrete labels. Even though the original labels of two samples might differ, they will be grouped together if they have the same discrete label. In the figure, a bigger node represents a sample with a larger weight, and the ovals represent the groups.

The sum of vertex weights is $Q \overset{(13)}{=} N_g + 2N_g(L-2) + N_g = 2N_g(L-1)$ and the sum of edge weights is $R \overset{(12)}{=} (L-1)(N_g)^2$, 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 as:

$$\hat{\mathbf{x}} \overset{\text{def}}{=} \frac{1}{Q} \sum_n \left(\mathbf{x}^1(n) + \mathbf{x}^L(n) + 2 \sum_{l=2}^{L-1} \mathbf{x}^l(n) \right) = \frac{1}{2(L-1)} \left(\hat{\mathbf{x}}^1 + \hat{\mathbf{x}}^L + 2 \sum_{l=2}^{L-1} \hat{\mathbf{x}}^l \right) . \quad (60)$$

From (14), the sample covariance matrix accounting for the weights v^l_n of the serial training graph is:

$$\mathbf{C}_{\text{ser}} \overset{(14)}{=} \frac{1}{Q} \left(\sum_n \mathbf{x}^1(n)(\mathbf{x}^1(n))^T + 2 \sum_{l=2}^{L-1} \sum_n \mathbf{x}^l(n)(\mathbf{x}^l(n))^T + \sum_n \mathbf{x}^L(n)(\mathbf{x}^L(n))^T - Q\hat{\mathbf{x}}(\hat{\mathbf{x}})^T \right) \quad (61)$$

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

From (16), the matrix $\dot{\mathbf{C}}_{\mathbf{G}}$ using the edges $\gamma_{n,n'}^{l,l+1}$ defined above is:

$$\dot{\mathbf{C}}_{\text{ser}} \overset{(16)}{=} \frac{1}{R}\sum_{l=1}^{L-1}\sum_{n,n'}(\mathbf{x}^{l+1}(n') - \mathbf{x}^{l}(n))(\mathbf{x}^{l+1}(n') - \mathbf{x}^{l}(n))^T \tag{63}$$

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

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

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

By using (66) instead of (63), the slowest step in the computation of the covariance matrices, which is the computation of $\dot{\mathbf{C}}_{\text{ser}}$, can be reduced in complexity from $\mathcal{O}(L(N_g)^2)$ to only $\mathcal{O}(N)$ operations ($N = LN_g$), which is of the same order as the computation of $\mathbf{C}_{\text{ser}}$. Thus, for the same number of samples N, 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 8), which is a combination of the serial and clustered training graph and fulfils the consistency restriction (34). 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).

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

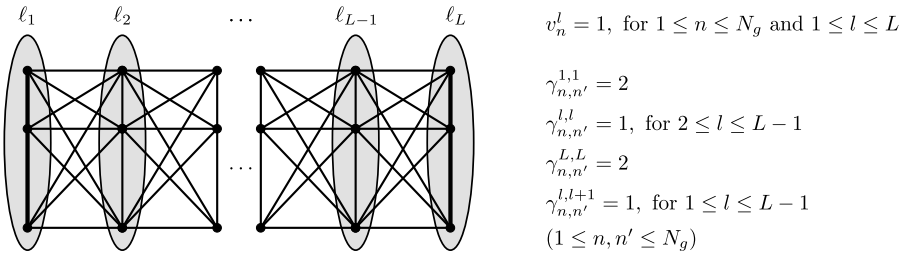

Figure 8: Illustration of the mixed training graph. Samples having the same label are fully connected (intra-group connections, represented with vertical edges) and all samples of adjacent groups are connected (inter-group connections). All vertex and edge weights are equal to 1 except for the intra-group edge weights of the first and last groups, which are equal to 2 as represented by wide edges.

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

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

if the loss function is the RMSE, where the slow features $\mathbf{y}$ might be extracted using any of the four SFA-based methods for regression above. Other loss functions, 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 certain computational properties.

Our method, complemented by a Gaussian classifier on 42 slow features, achieved a recognition rate of 96.4% on test data[3] which, 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.

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.

6.2.1 Problem and dataset 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.

Figure 9: Example of a pose-normalized image (left), and various images after pose was reintroduced illustrating the final range of vertical and horizontal displacements, as well as the face sizes (right).

To increase the usefulness of the images available, each image was pose-normalized once, but pose was reintroduced twice. This results in two post-processed images per original image having different random displacements and scalings. The post-processed images were then randomly split in three datasets (disjoint w.r.t. the original images) to perform particular sub-tasks. The first dataset, containing 30,000×2 images, was used to train the dimensionality reduction method (the factor 2 is due to the double pose-reintroduction

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.

above). The second one contains 20,000×2 images and was used to train the supervised post-processing step. The third dataset consists of 9,000×2 images and was used for testing.

6.2.2 Dimensionality-reduction methods evaluated

The resolution of the images and their number make it less practical to apply the majority of supervised methods, such as an SVM, and unsupervised methods, such as PCA/ICA/LLE, directly to the 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. 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.

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.

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 very 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 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 square sliding window graph with $d = 16$ (SW16).
- GSFA with a square sliding window graph with $d = 32$ (SW32).
- 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).

The evolution across the hierarchical network of the two slowest features extracted by HSFA is illustrated in Figure 10.

6.2.3 Supervised post-processing algorithms considered

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

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

To train the classifiers, the images of the second dataset 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, SW16, SW32, 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

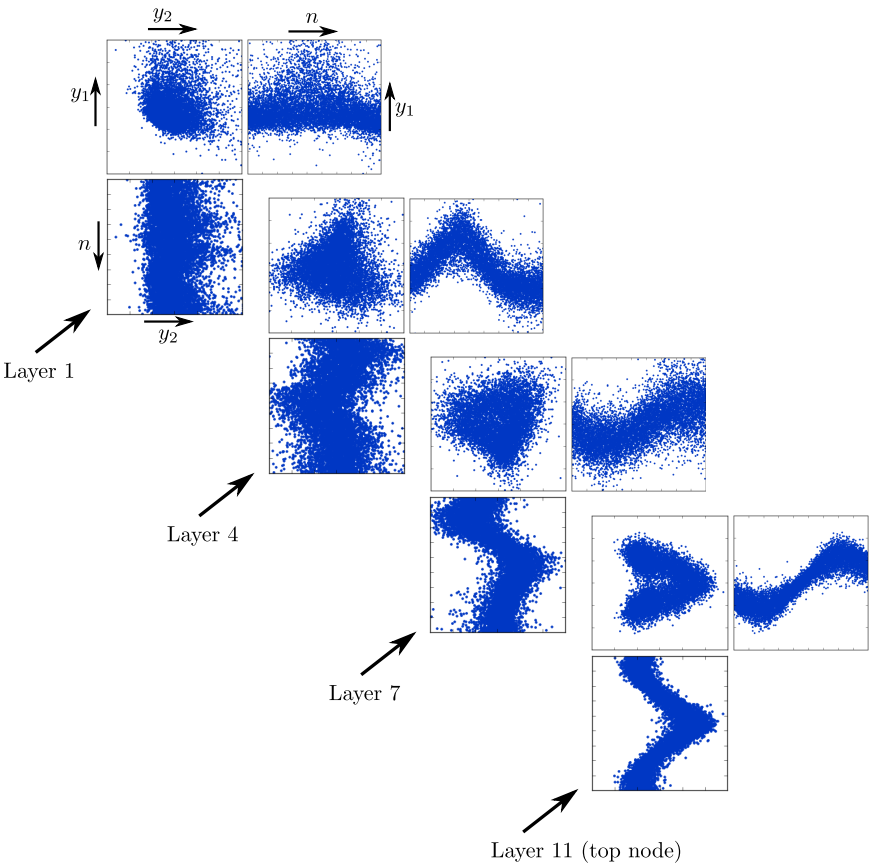

Figure 10: Evolution of the slow features extracted from test data after layers 1, 4, 7 and 11 of a GSFA network. A central node was selected from each layer, and three plots are provided. i.e., y_2 vs y_1, n vs y_1, and y_2 vs n. This network was trained with the serial training graph. Notice how slowness improves and structure appears as one moves to higher stages of the network.

graph, GSFA was 4% to 13% more accurate than the basic reordering of samples employing standard SFA. In turn, reordering was at least 10% better than HPCA for this dataset.

Taking a look at the number of features used by each supervised post-processing algorithm, one can realize 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

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
SW16 (GSFA)	5.80	5	5.26	5	5.60	17	9.77	60
SW32 (GSFA)	5.75	5	5.25	4	5.51	18	9.73	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 benefitted from all the SFA and PCA features available.

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, SW16 and SW32, 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 a larger number of connections in these graphs, or by some type of coupling by the sample groups in them and the classes of the post-processing algorithms. Still, SW16 and SW32 were better than the reordering approach, being the wider window slightly superior. The serial graph was better than the mixed one by less than 2% (for Soft GC), making it uncertain for statistical reasons which one of them is better for this problem. A larger number of trials, or even better, a more detailed mathematical analysis of the graphs might be necessary for this purpose.

7. Discussion

In this paper, we proposed graph-based SFA (GSFA), an implicitly supervised extension of the (unsupervised) SFA algorithm. 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. This algorithm is trained with 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 the label or class information.

We call GSFA implicitly supervised because the labels themselves are never provided to the algorithm, but only the training graphs, which encode the labels through their structure.

Therefore, GSFA does not fit to the labels explicitly, but instead fully concentrates on the generation of slow features according to the topology defined by the graph.

We also propose a weighted SFA optimization problem that takes into account the additional edge and weight information contained in a training graph, and generalizes the notion of slowness from a plain sequence of samples to such a graph. Moreover, we prove that GSFA solves the new optimization problem in the function space considered.

The algorithm shares many properties and advantages with its unsupervised counterpart, allowing its hierarchical implementation and facilitating the processing of high-dimensional data even when the number of samples is large. Moreover, it can be trained very efficiently in a feed-forward manner layer by layer, and the features at the top node can be complex and highly nonlinear w.r.t. the input. The experimental results demonstrate that the larger number of connections considered by GSFA indeed provides a more robust learning than standard SFA.

Unsupervised dimensionality reduction algorithms extract features that are frequently less useful for the particular supervised problem at hand. 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 more appropriate 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 adatpted to the specific data and labels.

Both classification and regression tasks can be solved with GSFA. We showed that a few slow features allow the solution of the supervised learning problem, and require only a small post-processing step that maps the features to labels or classes.

For classification problems, a clustered training graph was proposed, yielding features having the discrimination capability of FDA. The results of the implementation of this graph confirmed the expectations from theory. Training with the clustered graph comes down to considering all transitions between samples of the same identity and no transition between different identities. This is an advantage over Berkes (2005a), where a large number of transitions have to be explicitly considered.

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. 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 inefficient 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 if 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) proved 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 and this graph are not better or worse than 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,

future advances in generic methods for SFA, such as hierarchical network architectures or robust nonlinearities, might result in improved classification rates. Of course, it is possible to design other training graphs for classification without this equivalence, e.g., by using non-constant sample weights, or by incorporating input similarity information or other criteria in the edge-weight matrix.

To solve regression problems, we proposed three training graphs for GSFA that resulted in a reduction of the RMSE of up to 11% over the basic reordering approach using standard SFA, 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 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 versions of the mixed and serial training graphs were employed. Only three extracted features are passed to an explicit regression step based on a Gaussian classifier (Section 5.5). In both cases, good performance is 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.

Using a similar approach, other parameters such as the x-position, y-position, or scale of a face within an image have been estimated (Mohamed and Mahdi, 2010). 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 from 71.5% to 99.5% depending on the difficulty of the test images.

The serial and mixed training graphs provided the best accuracy in this case, 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). We are developing a theory on the slowest features that might be extracted by arbitrary graphs. This analysis might solve the question of which one of these two graphs is better or even give rise to a new one.

The features extracted by GSFA strongly depend on the labels, even though label information is only provided implicitly. 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, providing an approximated, redundant, and very concentrated coding of the label. Their simple structure permits the use of simple supervised algorithms for the post-processing step saving time and computer resources. In the case of regression, all the nonlinear algorithms for post-processing, 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 former algorithm (Soft GC) is more accurate because it reduces the effect of miss-classifications.

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.

One limitation of hierarchical processing with GSFA or SFA (i.e., HSFA) is that the features in the data should be spatially localized. For instance, if one randomly shuffles the pixels in the input image performance would decrease considerably. This reduces the applicability of HSFA to scenarios with heterogeneous independent sources of data, but is 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 (although hierarchical processing and certain expansions can be seen as a regularization measures). Hence, for a small number of samples data randomization techniques are useful.

We do not claim that GSFA is better or worse than other supervised learning algorithms. 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 specialized algorithms.

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, generalization would be less of an issue, and all SFA training methods would approximately converge to the same features and provide similar performance. Generalization has different facets, however. For instance, one interpretation is that GSFA provides better performance than SFA (reordering method) using the same amount of training data, as shown in the results. Another interpretation is that GSFA demands less training data to achieve the same performance, thus, indeed contributing to our pursuit of generalization.

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.

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.

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, 2005.

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, 2008.

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, 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.

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.