Journal of Machine Learning Research 17 (2016) 1-36 Submitted 6/15; Revised 3/16; Published 8/16

Theoretical Analysis of the Optimal Free Responses of Graph-Based SFA for the Design of Training Graphs

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 Bochum
Bochum D-44801, Germany

Editor: Zhuowen Tu

Abstract

Slow feature analysis (SFA) is an unsupervised learning algorithm that extracts slowly varying features from a multi-dimensional time series. Graph-based SFA (GSFA) is an extension to SFA for supervised learning that can be used to successfully solve regression problems if combined with a simple supervised post-processing step on a small number of slow features. The objective function of GSFA minimizes the squared output differences between pairs of samples specified by the edges of a structure called training graph. The edges of current training graphs, however, are derived only from the relative order of the labels. Exploiting the exact numerical value of the labels enables further improvements in label estimation accuracy.

In this article, we propose the exact label learning (ELL) method to create a more precise training graph that encodes the desired labels explicitly and allows GSFA to extract a normalized version of them directly (i.e., without supervised post-processing). The ELL method is used for three tasks: (1) We estimate gender from artificial images of human faces (regression) and show the advantage of coding additional labels, particularly skin color. (2) We analyze two existing graphs for regression. (3) We extract *compact* discriminative features to classify traffic sign images. When the number of output features is limited, such compact features provide a higher classification rate compared to a graph that generates features equivalent to those of nonlinear Fisher discriminant analysis. The method is versatile, directly supports multiple labels, and provides higher accuracy compared to current graphs for the problems considered.

Keywords: slow feature analysis, nonlinear regression, image analysis, pattern recognition, many classes

1. Introduction

The slowness principle is one of the learning paradigms that might explain the self-organization of neurons in the brain to extract invariant representations (e.g. Franzius et al., 2007). This principle operates on an abstract level of information processing and postulates that perceived information relevant to a subject typically changes much slower than individual sensory components—e.g., the position of a moth changes slower than the quickly changing neural activations in the retina of a frog observing it.

The slowness principle was probably first formulated by Hinton (1989), and early online learning rules were developed by Földiák (1991) and Mitchison (1991). The first closed-form algorithm is referred to as slow feature analysis (SFA, Wiskott, 1998; Wiskott and Sejnowski, 2002). Given a multi-dimensional time series (i.e., a sequence of samples), SFA finds an instantaneous mapping from the input samples to output features that change as slowly as possible within a given feature space. The objective function of SFA requires the minimization of the average squared differences of consecutive output values. Thus, SFA is especially useful for extracting slowly changing hidden parameters of the data.

Although SFA is unsupervised, it has also been used to solve supervised learning tasks, where it operates as a dimensionality reduction algorithm that is complemented by a supervised algorithm on a small number of slow features. This is motivated by the idea that samples originating at a similar time (e.g., consecutive samples) are likely to have similar labels due to physical and biological constraints on the subject and the environment. Thus, the temporal arrangement of the samples provides a weak form of supervised information.

Recently, an extension of SFA for classification and regression, called graph-based SFA (GSFA, Escalante-B. and Wiskott, 2013), has been proposed, which explicitly exploits the available labels. The training data of GSFA are organized in a graph structure called training graph, in which the vertices are the samples and the edge weights represent similarities between the corresponding labels, where each sample typically has several connections. The objective function of GSFA is similar to that of SFA, except that the pairs of samples need not be temporally consecutive, are weighted, and are indicated by the edges of the graph.

Typically, GSFA is more effective than SFA at extracting a set of features that tend to concentrate the label information, allowing accurate prediction of the labels from a few features, and implicitly solving the supervised learning problem. The resulting (low-dimensional) output features can then be easily post-processed by standard supervised algorithms, such as a regression method based on a Gaussian classifier (Escalante-B. and Wiskott, 2012) or ordinary least squares, to generate the final label estimation.

The main type of application addressed in this article is the solution of regression problems on high-dimensional data with hierarchical GSFA (HGSFA, see Section 2). Regression problems can be solved with GSFA using pre-defined graphs (e.g., a serial graph explained in Section 5.1). However, the structure of pre-defined graphs only takes into account the rank of the labels and not their exact value, a simplification that might decrease the estimation accuracy.

In this article, we focus on the analysis and design of training graphs. We develop a new approach, called exact label learning (ELL), for solving regression problems with GSFA based on the construction of a special training graph, in which the slowest feature extracted is already a label estimation, up to an affine transformation (Figure 1.c). The resulting system learns a nonlinear mapping from the input data (e.g., the pixels or features) to label estimations, where the features extracted by the layers of the GSFA network increase in complexity, abstraction level, and invariance as the data is propagated from the bottom to the top layer.

To develop the ELL method, we first study the slowest possible features that can be extracted by GSFA from a given graph when the feature space is unrestricted. Such features are called optimal free responses. After we express the optimal free responses of GSFA in a closed form, we develop a theoretical method for the converse operation: from a set of free

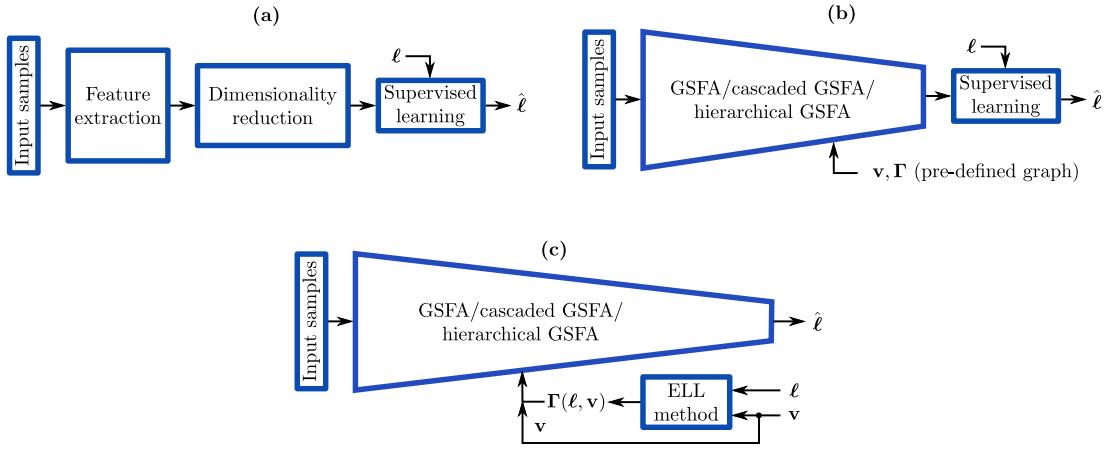

Figure 1: Three approaches for solving supervised learning problems. (a) A classic approach. (b) Previous approach using GSFA with a pre-defined training graph, which is defined by the input samples, node weights $\mathbf{v}$, and edge weights $\mathbf{\Gamma}$. The samples are assumed to be ordered by increasing label. (c) Our proposal, which consists of a single GSFA architecture that is trained with a specially constructed graph $\mathbf{\Gamma}(\boldsymbol{\ell}, \mathbf{v})$. The first slow feature (with a global sign adjustment) directly provides the label estimation. If the label $\boldsymbol{\ell}$ does not have weighted zero mean and weighted unit variance, a final affine transformation (scaling and offset) should be included.

responses we design the corresponding training graph. The method allows the creation of a graph in which the slowest possible feature is the label to be learned. Moreover, one can learn *multiple labels* simultaneously (e.g., object position, average color, shape, and size), and balance their importance by setting the value of certain parameters.

We show analytically that the serial graph is similar to the ELL graph in terms of the first optimal free responses, and that when only one label is learned the former may substitute the later reasonably well with faster training time. In addition, we outline in the discussion a few extensions to the ELL method towards improving its efficiency and allowing the combination of training graphs.

The remainder of the article is organized as follows: In the next section, we describe the context of the ELL method and review previous work. In Section 3, we review GSFA. In Section 4, we propose the ELL method. In Section 5, we provide 3 applications: (1) We solve a regression problem on gender estimation from artificial images, validating the method. (2) We analyze efficient pre-defined training graphs for regression. (3) We use the ELL method in a non-conventional way to design a training graph for the extraction of compact features for classification, yielding improved performance when the number of features preserved is between $\log_2(C)$ and $C - 2$, where C is the number of classes. For applications (1) and (3) the accuracy is evaluated experimentally; the first one uses HGSFA and the latter one uses direct (i.e., non-hierarchical) GSFA. Section 6 closes the article with a discussion.

2. Related Work

There are several approaches to solve regression problems on high-dimensional data and reduce the expensive computational requirements typically associated with this type of problems. A classic approach consists of feature extraction, (unsupervised) dimensionality reduction (DR), and an explicit supervised step (Figure 1.a). A different approach first uses GSFA for *supervised* DR. Supervised DR might result in higher accuracy than unsupervised DR. A small number of slow features extracted by GSFA are post-processed with a conventional classification or regression algorithm (Escalante-B. and Wiskott, 2013), see Figure 1.b. In such an approach, the supervised learning problem is mostly solved by GSFA implicitly, because it identifies and concentrates the label-predictive information in a few features.

For high-dimensional data, the direct application of SFA and GSFA is computationally too expensive, but one can resort to hierarchical processing (e.g., Franzius et al., 2011), which is an efficient divide-and-conquer strategy for the extraction of slow features[1] (e.g., Figure 2). For example, if the input dimension I is large, one can divide input data spatially into k lower-dimensional signals $\mathbf{x}^{(1)}(t), \ldots, \mathbf{x}^{(k)}(t)$ of dimensionality $I' \stackrel{\text{def}}{=} I/k$. Then, one can extract slow features $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$ from each lower-dimensional signal: $\mathbf{y}^{(1)}(t) \stackrel{\text{def}}{=} \text{SFA}^{(1)}(\mathbf{x}^{(1)}(t))$, $\mathbf{y}^{(2)}(t) \stackrel{\text{def}}{=} \text{SFA}^{(2)}(\mathbf{x}^{(2)}(t))$, $\ldots, \mathbf{y}^{(k)}(t) \stackrel{\text{def}}{=} \text{SFA}^{(k)}(\mathbf{x}^{(k)}(t))$. A concrete instance of SFA trained with a particular subset of the training data is denoted as $\text{SFA}^{(\cdot)}$. Different SFA instances are also referred to as SFA nodes, especially in the context of hierarchical SFA networks structured as directed graphs. The nodes above are called local because their input is only a local subset of the original input. Each of them extracts J' slow features. Afterwards, another SFA node $\text{SFA}^{(\text{top})}$ in an additional layer extracts global slow features from the concatenation of the local slow features computed previously: $\mathbf{y}^{(\text{top})}(t) \stackrel{\text{def}}{=} \text{SFA}^{(\text{top})}\left(\mathbf{y}^{(1)}(t)|\cdots|\mathbf{y}^{(k)}(t)\right)$, where $\cdot|\cdot$ is the concatenation operation in space (not in time). A proper choice of J' and k ensures that the computation of $\mathbf{y}^{(\text{top})}(t)$ is feasible.

If the input dimensionality I' of the local nodes $\text{SFA}^{(1)}, \ldots, \text{SFA}^{(k)}$ is still too large, one can repeat the strategy above to each of these nodes. Following such an approach recursively results in a multi-layer hierarchical network. Due to information loss before the top node and the change of the feature space, hierarchical SFA does not guarantee globally optimal slow features anymore. However, it has been shown to be effective in many practical experiments, in part because low-level features are spatially localized in most real-world data. Hierarchical GSFA (HGSFA) offers an excellent computational complexity compared to direct GSFA that can be as good as linear w.r.t. the number of samples and the input dimensionality, depending on the network architecture (Escalante-B. and Wiskott, 2016).

The ELL method allows the creation of graphs useful to train direct, cascaded, and hierarchical GSFA. Cascaded GSFA refers to the application of several consecutive passes of GSFA (thus, it is equivalent to HGSFA with only one GSFA node per layer) and is a useful approach when the features obtained through direct GSFA are not slow enough for a

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

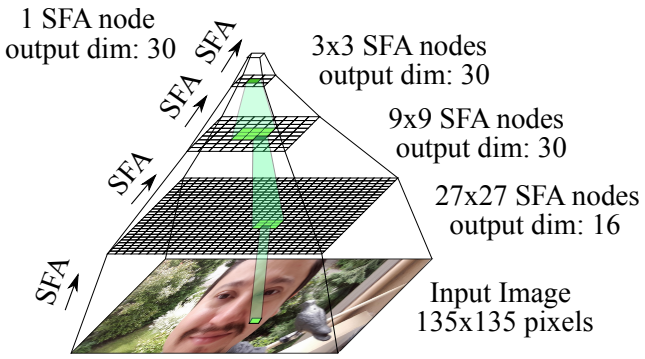

Figure 2: An example hierarchical SFA network for 2D data with 4 layers and no receptive field overlap that has been used for gender and age estimation from artificial face images (Escalante-B. and Wiskott, 2010). The SFA nodes can be easily replaced by GSFA nodes to construct an HGSFA network. The input to one node in layers 1, 2 and 3 is highlighted.

particular application. The main advantage of cascaded over direct GSFA is that the feature space of the cascade may be more complex without making the individual nodes/layers/steps more complex. The theoretical part of this article concentrates on GSFA for simplicity, but we implicitly assume that it will be implemented in practice as HGSFA and applied to high-dimensional data.

There is a close relation between GSFA, generalized SFA (genSFA, Sprekeler, 2011; also see Rehn and Sprekeler, 2014) and locality preserving projections (LPP, He and Niyogi, 2003), sharing very similar objective functions and constraints, even though they originate from different backgrounds and were developed with different goals and applications in mind. Two differences are that in GSFA the vertex weights are independent of the edge weights and that GSFA is invariant to the scale of the weights, providing a normalized objective function $0 \leq \Delta \leq 4$ (for consistent graphs with non-negative edge weights). There are also differences in the similarity matrices used in practice for these algorithms. LPP originates from the field of manifold learning and its similarity matrices have been frequently computed using nearest neighbors of the input samples, reflecting input similarities. genSFA has mostly been used for classification and its similarity matrices are typically computed using input similarity information (nearest neighbors) with transitions restricted to samples of the same class. One can consider genSFA as LPP applied on the nonlinearly expanded data. GSFA originates from unsupervised learning and learning of invariances but is motivated by supervised learning, and training graphs for classification and regression have been used. Such graphs are computed using only the label information. The goal is to provide sensitivity to the label information and invariance to any other factor. Therefore, GSFA does not intend to preserve the manifold structure of the input data. It is possible to use GSFA to compute LPP features, and vice versa. The results of this article might thus also be relevant for researchers using LPP and genSFA.

Various efficient training graphs for classification (clustered graph) and regression (e.g., serial, mixed, sliding window graphs) have been proposed (Escalante-B. and Wiskott, 2013). These graphs have been pre-defined with efficiency in mind; although the number of edges contained in them is $\mathcal{O}(N^2)$, where N is the number of samples, their structure makes the training complexity linear w.r.t. N. This is possible due to algebraic simplifications in the training method that avoid the explicit representation of the graph as an $N \times N$ matrix.

Wiskott (2003) has studied the optimal free responses of SFA, i.e., the slowest possible features that can be extracted by SFA when there is no restriction regarding the training data or feature space. For SFA, he has computed the optimal free responses in continuous time by using variational calculus (also see Franzius et al., 2007). In this article, we use a different method for GSFA based on linear algebra to cope with the discrete nature of the index n that takes the place of time.

Due to the close connection between GSFA and LPP, the method proposed in Section 4.1 for computing the free responses of GSFA is closely connected to Laplacian eigenmaps (LE, Belkin and Niyogi, 2003). LE can be interpreted as a relaxation of LPP where the output features belong to an unrestricted feature space instead of being linear transformations of the inputs. Equivalently, LPP is a linearization of LE (Zhang et al., 2009).

Most regression methods have the shortcoming of a prohibitive computational cost if the data is high dimensional. Unsupervised DR can be useful to reduce the complexity, but the final accuracy may be suboptimal. Instead of direct regression, one could attempt to do hierarchical regression by training several regression nodes on low-dimensional data chunks and then combining their outputs on higher layers, similarly as HGSFA is built to create a network of GSFA nodes. However, it is likely that the labels are not extractable at the lowest levels of the network (except with a noticeable error). Therefore, most information would be lost after the first layer, making the next layers unable to recover the labels accurately. GSFA extracts several slow features that do not need to be related to the labels in any simple way, allowing more label information to reach the top layers in HGSFA, even if the labels cannot be extracted in the first layers.

The features extracted by GSFA nodes in an HGSFA network have smaller receptive fields in the first layers that increase in size, as well as in complexity and selectivity, as one approaches the top of the network. This property is also present in other neural networks, such as the highly successful convolutional neural networks (CNNs). However, HGSFA and CNNs differ considerably: the training method of HGSFA is bottom-up, uses other types of nonlinearities, is not convolutional (although one can make some layers convolutional via weight sharing), and uses neither backpropagation nor max pooling. Moreover, in HGSFA the features of each node fulfill a local optimality criterion (i.e., slowness).

3. Review of Graph-Based SFA (GSFA)

In this section, we recall the GSFA optimization problem and the GSFA algorithm. For a more detailed presentation of GSFA we refer to Escalante-B. and Wiskott (2013).

3.1 Training Graphs and the GSFA Problem

GSFA is trained with a so-called training graph, in which the vertices are the samples and the edges between two samples may represent or be related to the similarity of their labels.

In mathematical terms, the training data is represented as a training graph $G = (\mathbf{V}, \mathbf{E})$ (illustrated in Figure 3.a) with a set $\mathbf{V} = \{\mathbf{x}(n)\}_n$ of vertices, each vertex being a sample (i.e., a vector of length I), 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') replaces the time variable t used by SFA. The edges are directed but typically have symmetric weights $\mathbf{\Gamma}^T = \mathbf{\Gamma} = \{\gamma_{n,n'}\}_{n',n}$; weights $v_n > 0$ are associated with the vertices $\mathbf{x}(n)$ and can be used to reflect their importance, frequency, or reliability. This representation includes the standard time series of SFA as a special case in which the graph has a linear structure (see Figure 3.b).

In order to solve classification problems with GSFA features, one should use training graphs that favor connections between samples from the same class by means of larger edge weights compared to those of different classes. When one is interested in regression problems, the training graphs should favor connections between samples with similar labels.

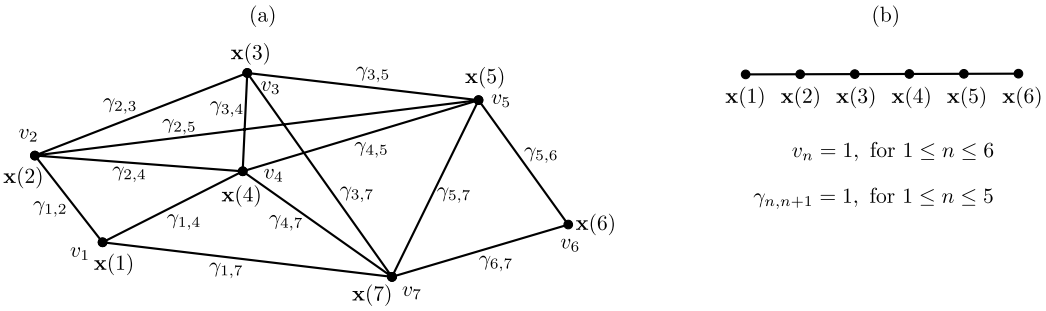

Figure 3: (a) Example of a training graph with $N = 7$ vertices. (b) A regular sample sequence (time series), which can be used to train SFA. This sequence is represented here as a linear graph that can be used with GSFA. If labels are available and the samples have been reordered by increasing/decreasing label (e.g., instead of having been ordered by time), the graph is called *sample reordering* graph. (Figure from Escalante-B. and Wiskott, 2013).

The concept of slowness has been originally defined for sequences of samples, but it has been generalized for GSFA to training graphs. The general goal of GSFA is to extract features that fulfill certain normalization restrictions and minimize the sum of the weighted squared output differences of all connected samples. More formally, the GSFA optimization problem (Escalante-B. and Wiskott, 2013) can be stated as follows: For $1 \leq j \leq J$, where J is the number of output features, find features $y_j(n) \stackrel{\text{def}}{=} g_j(\mathbf{x}(n))$, where $1 \leq n \leq N$, N is the number of samples, and g_j is a function belonging to a feature space $\mathcal{F}$ (frequent choices for $\mathcal{F}$ are all linear or quadratic transformations of the inputs), such that the objective function (weighted delta value)

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

under the constraints

$$\frac{1}{Q}\sum_n v_n y_j(n) = 0\,, \tag{2}$$

$$\frac{1}{Q}\sum_n v_n (y_j(n))^2 = 1\,, \text{ and} \tag{3}$$

$$\frac{1}{Q}\sum_n v_n y_j(n) y_{j'}(n) = 0\,, \text{ for } j' < j\,, \tag{4}$$

$$\text{with } Q \stackrel{\text{def}}{=} \sum_n v_n \text{ and } R \stackrel{\text{def}}{=} \sum_{n,n'} \gamma_{n,n'}\,. \tag{5}$$

The objective function penalizes the squared output differences between arbitrary pairs of samples using the edge weights as weighting factors. The feature $y_1(n)$, for $1 \le n \le N$, is the slowest one, $y_2(n)$ is the second slowest, and so on. Constraints (2)–(4) are called weighted zero mean, weighted unit variance, and weighted decorrelation, respectively. They are similar to the normalization constraints of SFA, except for the inclusion of vertex weights. The factors $1/R$ and $1/Q$ are not essential for the optimization problem, but they provide invariance to the scale of the edge weights as well as to the scale of the vertex weights, and serve a normalization purpose.

We write vectors and matrices in bold type. For instance, $\mathbf{y}_j$ is the j-th feature vector of size N, $y_j(n)$ is the j-th feature of sample n, and $\mathbf{x}(n)$ is the n-th input sample of size I.

3.2 Linear GSFA Algorithm

The linear GSFA algorithm is similar to standard SFA (Wiskott and Sejnowski, 2002) and only differs in the computation of the matrices $\mathbf{C}$ and $\dot{\mathbf{C}}$, which in GSFA takes into account the neighborhood structure specified by the training graph (samples, edges, and weights). The sample covariance matrix $\mathbf{C_G}$ is defined as:

$$\mathbf{C_G} \stackrel{\text{def}}{=} \frac{1}{Q}\sum_n v_n (\mathbf{x}(n) - \tilde{\mathbf{x}})(\mathbf{x}(n) - \tilde{\mathbf{x}})^T\,,$$

where $\mathbf{x}(n)$ and v_n denote an input sample and its weight, respectively, and

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

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

$$\dot{\mathbf{C}}_\mathbf{G} \stackrel{\text{def}}{=} \frac{1}{R}\sum_{n,n'} \gamma_{n,n'} \big(\mathbf{x}(n') - \mathbf{x}(n)\big)\big(\mathbf{x}(n') - \mathbf{x}(n)\big)^T\,, \tag{6}$$

where edge weights $\gamma_{n,n'}$ are defined as 0 if the graph does not have an edge $(\mathbf{x}(n), \mathbf{x}(n'))$.

Given these matrices, a sphering matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$ are computed with

$$\mathbf{S}^T \mathbf{C_G} \mathbf{S} = \mathbf{I}\,, \text{ and}$$
$$\mathbf{R}^T \mathbf{S}^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{S} \mathbf{R} = \mathbf{\Lambda}\,,$$

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

$$\begin{aligned} \mathbf{W} &= \mathbf{S}\mathbf{R}\,, \text{ and} \\ \mathbf{y}(n) &= \mathbf{W}^T(\mathbf{x}(n) - \tilde{\mathbf{x}})\,. \end{aligned} \tag{7}$$

It has been shown that the GSFA algorithm presented above indeed solves the optimization problem (1)–(4) in the linear function space. The proof is similar to the corresponding proof of standard linear SFA (Wiskott and Sejnowski, 2002).

The choice of the training graph $\mathbf{\Gamma}$ is important because it defines the types of features to be extracted. Figure 5 shows a serial graph useful for regression, whereas Figure 9 shows a clustered graph useful for classification.

3.3 Probabilistic interpretation of a graph

Interestingly, if the graph is connected and the following *consistency* restriction is fulfilled

$$\forall n : v_n/Q = \sum_{n'} \gamma_{n,n'}/R\,, \tag{8}$$

then GSFA yields the same features as standard SFA trained on a sequence generated by using the graph as a Markov chain with transition probabilities $\gamma_{n,n'}/R$ (see Klampfl and Maass, 2010; Escalante-B. and Wiskott, 2013). Thus, one can use SFA to emulate GSFA. However, depending on the training graph chosen, emulating GSFA with SFA may be more expensive computationally.

3.4 GSFA Optimization Problem in Matrix Notation

In order to apply linear algebra methods to analyze GSFA, we use matrix notation. In what follows we assume that the edge weights are symmetric[2] ($\mathbf{\Gamma} = \mathbf{\Gamma}^T$) and that the consistency restriction (8) is fulfilled. This restriction can also be written as

$$\mathbf{v} \overset{(8)}{=} \frac{Q}{R}\mathbf{\Gamma}\mathbf{1}\,, \tag{9}$$

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

If $\mathbf{y}$ is a feasible solution (i.e., satisfying (2) and (3)) and the graph fulfills the consistency restriction (8), the weighted delta value (1) can be simplified as follows,

$$\begin{aligned} \Delta_{\mathbf{y}} &\overset{(1)}{=} \frac{1}{R}\sum_{n,n'} \gamma_{n,n'}(y(n') - y(n))^2 \\ &= \frac{1}{R}\Big(\sum_{n'}(y(n'))^2 \sum_{n} \gamma_{n,n'} + \sum_{n}(y(n))^2 \sum_{n'} \gamma_{n,n'} - 2\sum_{n,n'} \gamma_{n,n'} y(n') y(n)\Big) \\ &\overset{(8)}{=} \frac{1}{R}\Big(\sum_{n'}(y(n'))^2 \frac{R}{Q} v(n') + \sum_{n}(y(n))^2 \frac{R}{Q} v(n) - 2\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\Big) \\ &\overset{(3)}{=} 2 - \frac{2}{R}\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\,. \end{aligned} \tag{10}$$

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

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

$$\Delta_j \stackrel{(1,3,8)}{=} 2 - \frac{2}{R}\mathbf{y}_j^T \mathbf{\Gamma} \mathbf{y}_j \tag{11}$$

subject to:

$$\mathbf{v}^T \mathbf{y}_j \stackrel{(2)}{=} 0 \tag{12}$$

$$\mathbf{y}_j^T \text{Diag}(\mathbf{v}) \mathbf{y}_j \stackrel{(3)}{=} Q \tag{13}$$

$$\mathbf{y}_j^T \text{Diag}(\mathbf{v}) \mathbf{y}_{j'} \stackrel{(4)}{=} 0, \text{ for } j' < j\,, \tag{14}$$

where

$$Q \stackrel{(5.a)}{=} \mathbf{1}^T \mathbf{v}\,, \tag{15}$$

$$R \stackrel{(5.b)}{=} \mathbf{1}^T \mathbf{\Gamma} \mathbf{1}\,, \tag{16}$$

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

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

4. Explicit Label Learning for Regression Problems

In this section, we propose the ELL method. First, we compute the optimal free responses of GSFA given any training graph. Then, we show how to construct a graph useful to learn any particular label or multiple labels. Afterwards, we show how to convert graphs with negative edge weights into graphs with non-negative weights only (such a method is useful to allow the probabilistic interpretation of the graph and to guarantee that the Δ values of all features lie between 0 and 4). Then, we motivate the use of auxiliary labels to improve learning. Finally, we analyze the computational complexity of the ELL method.

4.1 Optimal Free Responses of GSFA

In this section, we calculate the slowest possible solutions (optimal free responses) to the GSFA problem (11)–(14) that one could find if the feature space were unlimited. As we will see, the optimal free responses together with their corresponding Δ values, provide an alternative representation of the training graph and are a useful tool to understand its structure.

We use the Lagrange multiplier method to find critical points $\mathbf{y}$ that are candidates for the optimal free responses. For the moment, we ignore the weighted decorrelation constraint (14) to solve for the first optimal free response, but we consider the remaining responses later. The method of Wiskott (2003) and Franzius et al. (2007) for computing optimal free responses of SFA relies on a continuous time variable t and cannot be applied to GSFA due the discrete index n. Due to the close relationship between GSFA and LPP, the approach below is strongly related to Laplacian Eigenmaps (Belkin and Niyogi, 2003). Let

$$L \stackrel{\text{def}}{=} \big(2 - \frac{2}{R}\mathbf{y}^T \mathbf{\Gamma} \mathbf{y}\big) + \alpha \mathbf{v}^T \mathbf{y} + \beta\big(\mathbf{y}^T \text{Diag}(\mathbf{v}) \mathbf{y} - Q\big) \tag{17}$$

be a Lagrangian corresponding to the objective function (11), under the constraints (12) and (13). A signal $\mathbf{y}$ is a critical point if the partial derivatives of L with respect to α, β, and $y(n)$, for $1 \le n \le N$, are simultaneously zero:

$$\partial L/\partial \alpha \overset{(17)}{=} \mathbf{v}^T\mathbf{y} \overset{!}{=} 0\,, \tag{18}$$

$$\partial L/\partial \beta \overset{(17)}{=} \mathbf{y}^T\mathrm{Diag}(\mathbf{v})\mathbf{y} - Q \overset{!}{=} 0\,, \text{ and} \tag{19}$$

$$\partial L/\partial \mathbf{y} \overset{(17)}{=} -\frac{4}{R}\mathbf{\Gamma}\mathbf{y} + \alpha\mathbf{v} + 2\beta\mathrm{Diag}(\mathbf{v})\mathbf{y} \overset{!}{=} \mathbf{0}\,, \tag{20}$$

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

Equations (18) and (19) merely require that the output $\mathbf{y}$ has weighted zero mean and weighted unit variance, respectively. Multiplying (20) with $\mathbf{1}^T$ from the left and taking into account that $\mathbf{1}^T\mathrm{Diag}(\mathbf{v}) = \mathbf{v}^T$, $\mathbf{1}^T\mathbf{v} \overset{(15)}{=} Q$,$\mathbf{1}^T\mathbf{\Gamma} \overset{(9)}{=} \frac{R}{Q}\mathbf{v}^T$, and $Q > 0$ results in

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

implying $\alpha = 0$ due to (18). Therefore, (20) can be simplified to:

$$\begin{aligned}
& \Big(-\frac{4}{R}\mathbf{\Gamma} + 2\beta\mathrm{Diag}(\mathbf{v})\Big)\mathbf{y} = \mathbf{0}\,, \\
\Leftrightarrow \quad & \mathrm{Diag}(\mathbf{v}^{-1/2})\Big(\frac{4}{R}\mathbf{\Gamma} - 2\beta\mathrm{Diag}(\mathbf{v})\Big)\mathrm{Diag}(\mathbf{v}^{-1/2})\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y} = \mathbf{0}\,, \\
\Leftrightarrow \quad & \Big(\frac{4}{R}\mathrm{Diag}(\mathbf{v}^{-1/2})\,\mathbf{\Gamma}\,\mathrm{Diag}(\mathbf{v}^{-1/2}) - 2\beta\mathbf{I}\Big)\big(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}\big) = \mathbf{0}\,, \\
\Leftrightarrow \quad & \Big(\mathrm{Diag}(\mathbf{v}^{-1/2})\,\mathbf{\Gamma}\,\mathrm{Diag}(\mathbf{v}^{-1/2}) - \frac{R\beta}{2}\mathbf{I}\Big)\big(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}\big) = \mathbf{0}\,,
\end{aligned} \tag{21}$$

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

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

$$\mathbf{M} \overset{\text{def}}{=} \mathrm{Diag}(\mathbf{v}^{-1/2})\,\mathbf{\Gamma}\,\mathrm{Diag}(\mathbf{v}^{-1/2})\,. \tag{22}$$

The corresponding eigenvalue is denoted

$$\lambda = \frac{R\beta}{2}\,. \tag{23}$$

We denote the (orthogonal) eigenvectors of matrix $\mathbf{M}$ as $\mathbf{u}_j$ with $\mathbf{u}_j^T\mathbf{u}_j = 1$. Each eigenvector $\mathbf{u}_j$ gives rise to a critical point $\mathbf{y}_j \overset{\text{def}}{=} Q^{1/2}\mathrm{Diag}(\mathbf{v}^{-1/2})\mathbf{u}_j$ as long as also the weighted normalization constraints (12) and (13) are satisfied by $\mathbf{y}_j$. The slowest possible solution is the critical point $\mathbf{y}_j$ with the smallest Δ-value. As we show below, the Δ-value of a critical point $\mathbf{y}_j$ is directly related to the eigenvalue λ_j of the eigenvector $\mathbf{u}_j = Q^{-1/2}\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}_j$ of $\mathbf{M}$ and can be computed as follows.

$$\begin{aligned}
\Delta_{\mathbf{y}_j} &\overset{(11)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T \mathbf{\Gamma} \mathbf{y}_j \\
&\overset{(22)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T \mathrm{Diag}(\mathbf{v}^{1/2})\, \mathbf{M}\, \left(\mathrm{Diag}(\mathbf{v}^{1/2})\mathbf{y}_j\right) \\
&\overset{(23)}{=} 2 - \frac{2}{R}(\mathbf{y}_j)^T \mathrm{Diag}(\mathbf{v}^{1/2}) \lambda_j \mathrm{Diag}(\mathbf{v}^{1/2}) \mathbf{y}_j \\
&\overset{(13)}{=} 2 - \frac{2Q}{R}\lambda_j \,. \qquad (24)
\end{aligned}$$

Thus, the slowest solution is the critical point $\mathbf{y}_j$ with the largest eigenvalue λ_j. The remaining optimal free responses can now be addressed. They are given by the remaining critical points, where their corresponding eigenvalue defines their order, from largest to smallest. The weighted decorrelation condition (14) is fulfilled automatically due to the orthogonality of the eigenvectors: $\mathbf{u}_j^T \mathbf{u}_{j'} = 0 \Leftrightarrow \frac{1}{Q}\mathbf{y}_j^T \mathrm{Diag}(v)\mathbf{y}_{j'} = 0$ (follows from the definition of $\mathbf{y}_j$ above).

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

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

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

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

Due to the normalization constraints, the outputs generated by GSFA must have weighted zero mean (12) and weighted unit variance (13). Therefore, to learn a single label $\boldsymbol{\ell}$ we normalize it as follows: Let $\mu_{\boldsymbol{\ell}} = \frac{1}{Q}\mathbf{v}^T\boldsymbol{\ell}$ be the weighted label average and $\sigma^2_{\boldsymbol{\ell}} = \frac{1}{Q}(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1})^T \mathrm{Diag}(\mathbf{v})(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1})$ be the weighted label variance. Then, the normalized label is computed as

$$\tilde{\boldsymbol{\ell}} = \frac{1}{\sigma_{\boldsymbol{\ell}}}(\boldsymbol{\ell} - \mu_{\boldsymbol{\ell}}\mathbf{1}) \,. \qquad (25)$$

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

In order for the construction to work when samples have multiple labels, we must weight decorrelate them first. To decorrelate two labels $\boldsymbol{\ell}_{j'}$ and $\boldsymbol{\ell}_j$, with $j' > j$, one can project $\boldsymbol{\ell}_j$ out of $\boldsymbol{\ell}_{j'}$; $\ell^{\mathrm{dec}}_{j'}(n) = \ell_{j'}(n) - \frac{1}{Q}\left(\boldsymbol{\ell}_{j'}^T \mathrm{Diag}(\mathbf{v})\boldsymbol{\ell}_j\right)\ell_j(n)$, which is an invertible linear operation.

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

Define

$$\mathbf{\Gamma}^{\mathrm{ELL}} \stackrel{\mathrm{def}}{=} \mathrm{Diag}(\mathbf{v}^{1/2})\, \mathbf{M}^{\mathrm{ELL}}\, \mathrm{Diag}(\mathbf{v}^{1/2})\,, \tag{26}$$

where

$$\mathbf{M}^{\mathrm{ELL}} \stackrel{\mathrm{def}}{=} \sum_{j=0}^{N-1} \lambda_j \mathbf{u}_j^{\mathrm{ELL}} (\mathbf{u}_j^{\mathrm{ELL}})^T\,. \tag{27}$$

If $L < N-1$ one can set $\lambda_{j>L} = 0$. The matrix $\mathbf{\Gamma}^{\mathrm{ELL}}$ is symmetric by construction. The eigenvectors and eigenvalues of $\mathbf{M}^{\mathrm{ELL}}$, which are explicit in its eigenvector decomposition (27), directly define the matrix $\mathbf{\Gamma}^{\mathrm{ELL}}$ and determine the optimal free responses of the resulting graph. Concretely, for each $j \geq 1$ one sets $\mathbf{u}_j^{\mathrm{ELL}}$ according to the desired label $\boldsymbol{\ell}_j$ (ignore $\mathbf{u}_0^{\mathrm{ELL}}$ and λ_0 for the time being).

$$\mathbf{u}_j^{\mathrm{ELL}} = Q^{-1/2} \mathrm{Diag}(\mathbf{v}^{1/2}) \boldsymbol{\ell}_j, \text{ for } j \geq 1 \tag{28}$$

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

$$\frac{1}{Q} (\boldsymbol{\ell}_j)^T \mathrm{Diag}(\mathbf{v}) \boldsymbol{\ell}_{j'} \stackrel{(14)}{=} 0 \quad \stackrel{(28)}{\Leftrightarrow} \quad (\mathbf{u}_j^{\mathrm{ELL}})^T \mathbf{u}_{j'}^{\mathrm{ELL}} = 0 \tag{29}$$

Once the eigenvectors are computed we must decide which eigenvalues we want to give them. Alternatively, we can decide which Δ values we give to the labels, because $\Delta_{\boldsymbol{\ell}_j}$ and λ_j are directly related: $\lambda_j \stackrel{(24)}{=} \frac{R}{2Q}(2 - \Delta_{\boldsymbol{\ell}_j})$.

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

For example, if one only wants to learn a single label $\boldsymbol{\ell}_1$ with a delta value $\Delta_{\boldsymbol{\ell}_1} = 0$, one can set $\mathbf{u}_1^{\mathrm{ELL}} = Q^{-1/2} \mathrm{Diag}(\mathbf{v}^{1/2}) \boldsymbol{\ell}_1$, $\lambda_1 = 1$, and the eigenvalues $\lambda_{j>1}$ to zero. If $\boldsymbol{\ell}_1$ takes only two possible values (e.g., -1 and 1), the resulting graph will be disconnected and contain two clusters. Otherwise, the resulting graph will be connected, and the condition $\Delta_{\boldsymbol{\ell}_1} = 0$ necessarily implies that some of the resulting edge weights will be negative, a condition that we deal with in Section 4.3.

The analysis of Section 4.1, which is used by the ELL method requires that the graph fulfills the consistency restriction (9). We set the remaining eigenvector

$$\mathbf{u}_0^{\mathrm{ELL}} = Q^{-1/2} \mathbf{v}^{1/2}\,, \tag{30}$$

with eigenvalue $\lambda_0 = R/Q$. This ensures that $(\mathbf{u}_0^{\text{ELL}})^T \mathbf{u}_0^{\text{ELL}} = 1$ and (9) is fulfilled, as follows.

$$\begin{aligned}
\mathbf{\Gamma}^{\text{ELL}}\mathbf{1} &\overset{(26,27)}{=} \text{Diag}(\mathbf{v}^{1/2})\big(\sum \lambda_j \mathbf{u}_j^{\text{ELL}}(\mathbf{u}_j^{\text{ELL}})^T\big)\text{Diag}(\mathbf{v}^{1/2})\mathbf{1} \\
&\overset{(30)}{=} \text{Diag}(\mathbf{v}^{1/2})\big(\sum \lambda_j \mathbf{u}_j^{\text{ELL}}(\mathbf{u}_j^{\text{ELL}})^T\big)\mathbf{u}_0^{\text{ELL}} Q^{1/2} \\
&\overset{(30)}{=} \text{Diag}(\mathbf{v}^{1/2})\lambda_0 \mathbf{u}_0^{\text{ELL}} Q^{1/2} \\
&= (R/Q)\mathbf{v}\,. \qquad (31)
\end{aligned}$$

The assignment of $\mathbf{u}_0^{\text{ELL}}$ and λ_0 above also ensures that $\mathbf{1}^T\mathbf{\Gamma}^{\text{ELL}}\mathbf{1} \overset{(15,31)}{=} R$. The free pseudo-response $\boldsymbol{\ell}_0 \overset{(28)}{=} \mathbf{1}$ corresponding to $\mathbf{u}_0^{\text{ELL}}$ fulfills equations (13) and (14) but not (12). Therefore, $\boldsymbol{\ell}_0$ is not a feasible solution, but it has similar properties to the optimal free responses. The introduction of $\mathbf{u}_0^{\text{ELL}}$ does not reduce the generality of the labels $\boldsymbol{\ell}_{j>0}$ that can be learned; orthogonality between $\mathbf{u}_0^{\text{ELL}}$ and $\mathbf{u}_{j>0}^{\text{ELL}}$ is equivalent to (12), i.e., the weighted zero mean of $\boldsymbol{\ell}_{j>0}$, a condition that is required anyway for any feasible solution: $(\mathbf{u}_0^{\text{ELL}})^T\mathbf{u}_{j>0}^{\text{ELL}} = 0 \overset{(29)}{\Leftrightarrow} (Q^{-1/2}\mathbf{v}^{1/2})^T Q^{-1/2}\text{Diag}(\mathbf{v}^{1/2})\boldsymbol{\ell}_{j>0} = Q^{-1}\mathbf{v}^T\boldsymbol{\ell}_{j>0} = 0$.

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

4.2.1 Expected Weighted Δ Value of a Noise Feature

Let $\mathbf{y}$ be a noise feature randomly sampled from a zero-mean unit-variance distribution $\mathcal{D}$, i.e., $y(n) \leftarrow \mathcal{D}(0,1)$. On average, $\mathbf{y}$ fulfills the weighted normalization constraints (12) and (13), as can be seen as follows.

$$\text{(12):} \qquad \langle \mathbf{v}^T\mathbf{y}\rangle_{\mathcal{D}} = \mathbf{v}^T\langle\mathbf{y}\rangle_{\mathcal{D}} = 0\,, \qquad (32)$$

$$\text{(13):} \qquad \langle \mathbf{y}^T\text{Diag}(\mathbf{v})\mathbf{y}\rangle_{\mathcal{D}} = \langle\sum_n v_n y(n)^2\rangle_{\mathcal{D}} = \sum_n v_n\langle y(n)^2\rangle_{\mathcal{D}} = Q\,, \qquad (33)$$

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

$$\begin{aligned}
\langle\Delta_{\mathbf{y}}\rangle_{\mathcal{D}} &\overset{(1)}{=} \frac{1}{R}\sum_{n,n'}\gamma_{n,n'}\langle(y(n') - y(n))^2\rangle_{\mathcal{D}} \\
&= \frac{1}{R}\Big(\sum_{n,n',n\neq n'}\gamma_{n,n'}\langle(y(n') - y(n))^2\rangle_{\mathcal{D}} + \sum_n \gamma_{n,n}\langle(y(n) - y(n))^2\rangle_{\mathcal{D}}\Big) \\
&= \frac{1}{R}\Big(\sum_{n,n',n\neq n'}\gamma_{n,n'}\big(\langle y(n')^2\rangle_{\mathcal{D}} + \langle y(n)^2\rangle_{\mathcal{D}} - 2\langle y(n')\rangle_{\mathcal{D}}\langle y(n)\rangle_{\mathcal{D}}\big) + 0\Big) \\
&= \frac{1}{R}\sum_{n,n',n\neq n'}\gamma_{n,n'}(1 + 1 - 0) \\
&= \frac{2}{R}\big(\sum_{n,n'}\gamma_{n,n'} - \sum_n \gamma_{n,n}\big) \overset{(5)}{=} \frac{2(R - \sum_n \gamma_{n,n})}{R}\,. \qquad (34)
\end{aligned}$$

Therefore, if the graph has no self-loops (i.e., $\forall n : \gamma_{n,n} = 0$), the expected Δ value $\langle \Delta_{\mathbf{y}} \rangle_{\mathcal{D}}$ of a noise feature $\mathbf{y}$ is 2.0. The self-loops of a graph (e.g., one constructed using the ELL method) can be removed (i.e., their weight be set to zero). This does not change the free responses, only the scale of the Δ values is modified due to the change in R. The consistency restriction might be broken, though.

4.3 Elimination of Negative Edge Weights

From the objective function (1), it is obvious that a positive edge weight connecting two samples expresses that those samples should be mapped close to each other in feature space. In contrast, a negative edge weight expresses that two samples should be mapped as far apart as possible, thus encoding output dissimilarities. Nevertheless, the weighted unit variance constraint still applies, so the solutions are not unbounded.

If the edge weights are non-negative, the smallest possible Δ value is $\Delta = 0$. However, if negative edge weights are allowed, some feasible features might have $\Delta < 0$. A feature with $\Delta < 0$ would appear to be "slower" than the infeasible constant feature $\mathbf{y} = \mathbf{1}$ with $\Delta = 0$, contradicting the intuitive interpretation of slowness. Moreover, negative edge weights hinder the probabilistic interpretation of the graph (see Section 3.2), because some of the transition probabilities $\gamma_{n,n'}/R$ of the resulting Markov chain would be negative.

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

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

$$c \stackrel{\text{def}}{=} \max_{n,n'} \frac{-\gamma_{n,n'}}{v_n v_{n'}}\,. \tag{35}$$

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

$$\mathbf{\Gamma'} \stackrel{\text{def}}{=} \frac{1}{1 + cQ^2/R}(\mathbf{\Gamma} + c\mathbf{v}\mathbf{v}^T). \tag{36}$$

Now, we show the properties of $\mathbf{\Gamma'}$ compared to those of $\mathbf{\Gamma}$:

1. All elements of $\mathbf{\Gamma'}$ are greater or equal to zero, as desired. (Follows from (35), which implies $\gamma_{n,n'} + cv_n v_{n'} \geq 0$.)

2. Symmetry is preserved by (36). Clearly $\mathbf{\Gamma'}$ is symmetric if and only if $\mathbf{\Gamma}$ is symmetric.

3. The sum of edge-weights is preserved:

$$R' \stackrel{(16)}{=} \mathbf{1}^T \mathbf{\Gamma'} \mathbf{1} \stackrel{(36)}{=} \frac{R + cQ^2}{1 + cQ^2/R} = R\,. \tag{37}$$

4. Fulfillment of the graph consistency restriction (9) is preserved:

$$\mathbf{1}^T\mathbf{\Gamma} \overset{(9)}{=} R/Q\mathbf{v}^T \qquad \Rightarrow \qquad \begin{aligned}\mathbf{1}^T\mathbf{\Gamma}' &\overset{(36)}{=} \frac{1}{1+cQ^2/R}(\mathbf{1}^T\mathbf{\Gamma} + c\mathbf{1}^T\mathbf{v}\mathbf{v}^T)\\ &\overset{(9)}{=} \frac{1}{1+cQ^2/R}(R/Q\mathbf{v}^T + cQ\mathbf{v}^T)\\ &= \frac{R/Q}{1+cQ^2/R}(1+cQ^2/R)\mathbf{v}^T\\ &= R/Q\mathbf{v}^T\,.\end{aligned}$$

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

$$\begin{aligned}\Delta'_{\mathbf{y}} &\overset{(10)}{=} 2 - \frac{2}{R'}\mathbf{y}^T\mathbf{\Gamma}'\mathbf{y}\\ &\overset{(36,37)}{=} 2 - \frac{2}{R(1+cQ^2/R)}\big(\mathbf{y}^T\mathbf{\Gamma}\mathbf{y} + c\mathbf{y}^T\mathbf{v}\mathbf{v}^T\mathbf{y}\big)\\ &\overset{(12)}{=} 2 - \frac{2}{R(1+cQ^2/R)}\mathbf{y}^T\mathbf{\Gamma}\mathbf{y}\\ &\overset{(10)}{=} 2 - \frac{2}{R(1+cQ^2/R)}\frac{R}{2}(2-\Delta_{\mathbf{y}}) \qquad (38)\\ &= \frac{1}{(1+cQ^2/R)}\left(\Delta_{\mathbf{y}} + \frac{2cQ^2}{R}\right)\,. \qquad (39)\end{aligned}$$

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

6. In particular, a feature $\mathbf{y}$ with $\Delta_{\mathbf{y}} = 2$ preserves its delta value, i.e. $\Delta_{\mathbf{y}} = 2 \overset{(38)}{\Leftrightarrow} \Delta'_{\mathbf{y}} = 2$.

4.4 Auxiliary Labels for Boosting Estimation Accuracy

It is possible to provide additional *auxiliary* labels derived from the original one $\boldsymbol{\ell}_1$ to improve the estimation accuracy when GSFA is applied repeatedly (e.g., cascaded or in a convergent hierarchical GSFA network). Consider two GSFA nodes, one stacked on top of the other. If the first GSFA node is not be able to extract $\boldsymbol{\ell}_1$ accurately, it might still be capable of approximating labels $\boldsymbol{\ell}_k = f_k(\boldsymbol{\ell}_1)$, for $2 \le k \le K$, where the functions $f_k(\cdot)$ are nonlinear. Since these features are derived from the original label $\boldsymbol{\ell}_1$, they contain a certain amount of information about it. In this case, the output features are likely to contain linear combinations of the labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_k$ providing a redundant coding of $\boldsymbol{\ell}_1$. These features are likely to be easier to disentangle by the second node to better approximate the original label $\boldsymbol{\ell}_1$. Therefore, we suggest to explicitly promote the appearance of these features by learning also auxiliary labels.

The functions f_k can be defined arbitrarily, we suggest to use

$$\ell_k(n) = \cos\Big(\frac{\ell_1(n) - \min(\boldsymbol{\ell}_1)}{\max(\boldsymbol{\ell}_1) - \min(\boldsymbol{\ell}_1)}\pi k\Big)\,, \quad \text{for } 2 \le k \le K\,, \qquad (40)$$

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

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

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

Moreover, the use of auxiliary labels supports the goal that samples $\mathbf{x}(n)$ and $\mathbf{x}(n')$ with similar labels $\ell_1(n)$ and $\ell_1(n')$ should have similar output features $y_j(n)$ and $y_j(n')$ on average, for $1 \le j \le J$, and not only the slowest features $y_1(n)$ and $y_1(n')$. This is a result of the "smoothness" of the auxiliary labels in terms of $\boldsymbol{\ell}_1$ (i.e., how fast they change w.r.t. $\boldsymbol{\ell}_1$). Notice that $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_J$ would be ordered by decreasing smoothness.

Interestingly, in regular SFA (or GSFA trained with the reordering graph) the inclusion of auxiliary labels occurs automatically. The slowest free response is a half period of a cosine function, and the subsequent free responses are the higher-frequency harmonics of the first one (see Section 5.2, particularly Figure 7).

4.5 Computational Complexity of Explicit Label Learning

The main drawback of ELL is its computational efficiency compared to efficient pre-defined training graphs, which is more marked for large N. We analyze the efficiency of explicit label learning by considering its two main parts: The construction of the training graph and training GSFA with it.

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

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

However, when an efficient pre-defined graph (e.g., the serial graph) is used instead of an ELL graph, it is possible to avoid the explicit graph construction and compute $\dot{\mathbf{C}}_\mathbf{G}$ with optimized algorithms that take into account the regular structure of the graph. In this way,

efficient pre-defined graphs allow the computation of $\dot{\mathbf{C}}_{\mathbf{G}}$ in $\mathcal{O}(NI^2)$ operations, which is equivalent to the complexity of standard SFA on N I-dimensional samples. Moreover, if the number of edges $N_e \leq N(N+1)/2$ is small, one can use (6) to compute $\dot{\mathbf{C}}_{\mathbf{G}}$ in $\mathcal{O}(N_e I^2)$ operations. Therefore, for these two special cases, training GSFA takes $\mathcal{O}(NI^2 + I^3)$ and $\mathcal{O}((N_e+N)I^2+I^3)$ operations, respectively. In Section 6.4 we further discuss the complexity of the ELL method and in Section 6.5 we propose a few approaches to improve it.

5. Applications of Explicit Label Learning

This section we present three applications of the proposed method. The first one illustrates how to solve a regression problem with GSFA explicitly, learning a direct mapping from images to labels (see Figure 1.c). The second application shows the analysis of two pre-defined graphs by computing their optimal free responses. In the third application, the ELL method is used in a new way to learn compact discriminative labels for classification.

5.1 Explicit Estimation of Gender with GSFA

We consider the problem of gender estimation from artificial face images, which is treated here as a regression problem, because the gender parameter is defined as a real value by the face modeling software (FaceGen SDK, Singular Inversions Inc., 2008).

Input data. The input data are 12,000 64×64 grayscale images. Each image is generated using a new subject identity, where the gender is explicitly specified, and the rest of the parameters of the faces (e.g., age, racial composition) are random. The average pixel intensity of each image is normalized by multiplying the pixel values by an appropriate factor to eliminate skin color as a cue for gender estimation. The resulting images show subjects with a fixed pose, no hair or accessories, and the illumination is fixed, as well as the average pixel intensity and the background color (black). See Figure 4 for some sample images. To specify the gender parameter, 60 different values are used $(-3, -2.9, \ldots, 2.9)$.

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

Figure 4: Example of the normalized images used, showing different values of the gender parameter.

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

Besides the gender label, also a second "color" label is considered, which is the average pixel intensity of the image *before* normalization. Due to normalization, this label cannot be

computed directly, but it can be estimated from other cues, such as the subject's apparent race and face size. In the following experiment only the gender label is considered, but afterwards both labels (gender and color) are used simultaneously.

Network used. For efficiency reasons, hierarchical GSFA (HGSFA) is used, We tested an 8-layer HGSFA network with the structure described in Table 1. The nodes of the network have non-overlapping receptive fields and are composed of an expansion function $(x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_n, |x_1|^{0.8}, \ldots, |x_n|^{0.8})$ followed by linear GSFA. This expansion only doubles the data dimensionality and is called 0.8Expo (Escalante-B. and Wiskott, 2011). The nodes of the first layer include a PCA pre-processing step that preserves 50 out of 64 components.

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

Table 1: Structure of the GSFA hierarchical network. The inputs to the nodes in the first layer are 8×8-pixel patches. The input to the node in layer 8 is the output of the node in layer 7. The inputs to all other nodes come from two nodes in the previous layer that are contiguous either vertically or horizontally.

Training graphs for gender estimation. We construct several training graphs using the ELL method described in Sections 4.2–4.4. These graphs are denoted ELL^g-L, where L is the total number of target labels considered, with $L \in \{1, 10, 20, 30, 40, 50\}$, and the superscript g stands for gender (later c will be used for color and g, c for gender and color). The first target label $\ell_1(n)$ is the gender parameter, where $1 \le n \le$ 10,800. The remaining $L - 1$ labels are auxiliary and computed using (40). For comparison purposes, the serial and reordering training graphs are also evaluated.

Serial Training Graph. To evaluate the ELL method, we also consider the serial graph (Escalante-B. and Wiskott, 2013), which is an efficient pre-defined graph for regression with one label. The serial training graph, shown in Figure 4, is constructed by discretizing the original label $\boldsymbol{\ell}$ into a relatively small set of K discrete label values, namely $\{\ell_1, \ldots, \ell_K\}$, where $\ell_1 < \ell_2 < \cdots < \ell_K$. Afterwards, the samples are divided into K groups of size N/K sharing the same discrete labels. Edges connect all pairs of samples from consecutive groups with discrete labels ℓ_k and ℓ_{k+1}, for $1 \le k \le K - 1$. Thus, connections are only inter-group, and intra-group connections are absent. Notice that since any two vertices of the same group are adjacent to exactly the same neighbors, they are likely to be mapped to similar outputs by HGSFA. Following HGSFA a complementary explicit regression step on a few features solves the original regression problem. There are several efficient graphs

for regression besides the serial graph. We employ the serial graph in this article, because it has consistently given good results for various regression problems.

The serial graph allows efficient training in linear time w.r.t. N, whereas the number of connections considered is $\mathcal{O}(N^2)$ if the number of groups is constant.

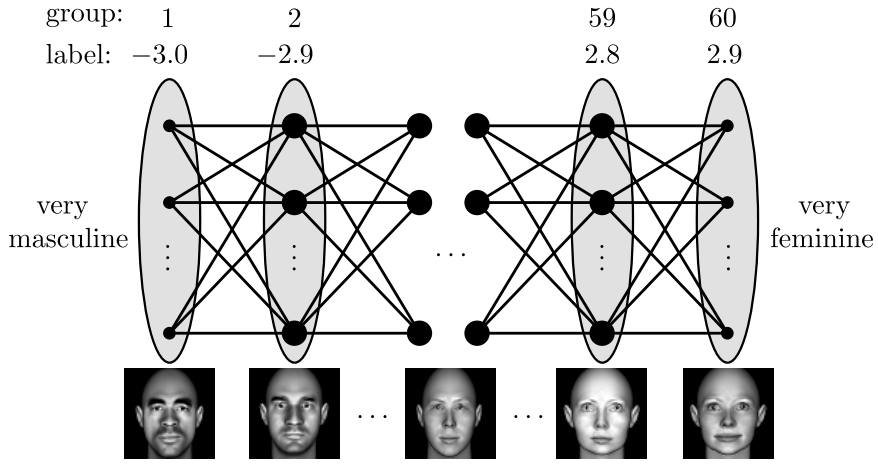

Figure 5: Illustration of a serial training graph for gender estimation. We use $K = 60$ discrete label values matching the original label values $\{\ell_1 = -3.0, \ell_2 = -2.9, \ell_3 = -2.8, \ldots, \ell_{60} = 2.9\}$. Therefore, for these data the labels were not discretized. However, in general the vertices of this graph are ordered by increasing gender value and then grouped according to their discrete label. Even if the original labels of two samples differ, they may be grouped together in the serial graph if they have the same discrete label. Each dot represents a sample, edges represent connections, and ovals represent groups of samples. The samples of groups with discrete labels ℓ_2 to ℓ_{59} have a weight of 2, whereas the samples of the first and last group with labels ℓ_1 and ℓ_{60} have a weight of 1 (sample weights represented by bigger/smaller dots). The weight of all edges is 1.

Label estimations. We used three mappings from the slowest features to the label estimation $\hat{\ell}$. The first mapping (only available for the ELL graphs) is an affine transformation $\hat{\ell} = \pm y_1 \sigma_{\boldsymbol{\ell}} + \mu_{\boldsymbol{\ell}}$, where $\mu_{\boldsymbol{\ell}}$ and $\sigma_{\boldsymbol{\ell}}$ had been previously computed for label normalization (25). Since the sign of y_1 is arbitrary, it is globally adjusted to fit the labels best. The second method is linear regression (LR, ordinary least squares). For these two methods, final label estimation $\hat{\ell}$ is clipped to the valid label range $[-3, 2.9]$. The third mapping is the soft GC method, which provides a soft estimation based on the class probabilities estimated by a Gaussian classifier, (trained with 60 classes Escalante-B. and Wiskott, 2013).

Results. Table 2 (left) shows the label estimation errors when gender is estimated. Unless otherwise stated, all results have been averaged over 10 runs. Depending on the mapping, the ELL^g-10 and ELL^g-40 graphs outperform the rest. This supports the intuition that auxiliary labels are useful. 50 target labels perform worse than 40, probably in part because the output dimensionality of the intermediate nodes in the network is 40. Without

the final clipping step LR was clearly more accurate than affine mapping (experiment not shown), but both methods have similar accuracy if clipping is enabled. For all graphs, the explicitly supervised soft GC method provided better accuracy than the affine mapping, although the difference is smaller than one might have expected.

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

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

Table 2: *Gender* estimation errors (RMSE) using various graphs and either one (1F) or three (3F) features. For the linear regression (LR) mapping, the label is estimated as $\hat{\ell}_1 = ay_1 + b$, with a and b fitted to the training data. Chance level (RMSE) is 1.731 if one uses the constant estimation $\hat{\ell}_1 = -0.05$. All errors computed on test data and averaged over 10 runs. (Left) Estimation errors using training graphs for gender estimation only. (Right) Estimation errors using training graphs for the experiment on simultaneous estimation of gender and color.

For comparison, the serial graph results in RMSEs of 0.351 (soft GC, 1F) and 0.349 (soft GC, 3F), whereas the reordering graph results in RMSEs of 0.353 (soft GC, 1F) and 0.347 (soft GC, 3F). The accuracy of these two graphs appears to be similar; however, in more complex experiments the serial graph has typically been more accurate. The ELL^{g}-40 graph is, therefore, slightly more accurate than the serial and reordering graphs but 25 times slower, taking about 250 min for training instead of about 10 min (single thread).

Simultaneous learning of gender and color. We construct a graph that encodes gender and color simultaneously, learning labels $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_L$, where $\boldsymbol{\ell}_1$ is the gender label, $\boldsymbol{\ell}_2$ is the color label, $\boldsymbol{\ell}_3, \boldsymbol{\ell}_5, \ldots, \boldsymbol{\ell}_{L-1}$ are derived from $\boldsymbol{\ell}_1$, and $\boldsymbol{\ell}_4, \boldsymbol{\ell}_6, \ldots, \boldsymbol{\ell}_L$ are derived from $\boldsymbol{\ell}_2$. Each set of labels is computed using (40) similarly to the auxiliary labels for gender only but starting from either the original gender or color labels. The chosen eigenvalues decrease linearly and add to one. The resulting graphs are denoted $\text{ELL}^{g,c}$-L, where L is the total number of target labels, with $L = 2 \times d$, for $d \in \{1, 5, 10, 15, 20, 25\}$, and $2(d-1)$ is the number of auxiliary labels used for gender and color.

The effect of coding gender and color simultaneously on gender estimation is shown in Table 2, right (compare to Table 2, left). The $\text{ELL}^{g,c}$-L graphs yield significantly higher accuracy than the ELL^{g}-L graphs (an MAE as small as 0.277 vs. 0.345). The results on color estimation using the $\text{ELL}^{g,c}$-L graphs are shown in Table 3, right (compare to Table 3, left). The slowest extracted feature represents mostly gender. However, it must also contain color information since it allows color estimation better than chance level. When 3 features are preserved, the $\text{ELL}^{g,c}$-L graphs yield higher accuracy than the ELL^{c}-L graphs. Similar

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

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

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

experimental results have been reported, e.g. by Guo and Mu (2014), who have shown that age estimation improves when gender and race labels are also considered.

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

The corresponding label estimations are shown in Figure 6. For comparison, also the ELL^g-40 graph is included. The results prove that the ELL method can also be used to learn distortions of the main label. Admittedly, the accuracy of the estimations (expressed as a fraction of the respective chance levels) decreases even though we increased the complexity of the feature space.

5.2 Analysis of Pre-Defined Training Graphs

In this section, we use the method of Section 4.1 to extract the optimal free responses of three graphs (reordering, serial and ELL-4). The optimal free responses and their Δ values (alternatively, the eigenvectors $\mathbf{u}_j$ and eigenvalues λ_j) fully characterize the properties of a training graph, and provide another representation of it that might be more useful in some contexts.

We compute optimal free responses using (21)–(23) and their delta values using (24). Therefore, these results have been obtained analytically. We plot them in Figure 7, which shows an arbitrary label to be learned (top), and three different graphs that can be used

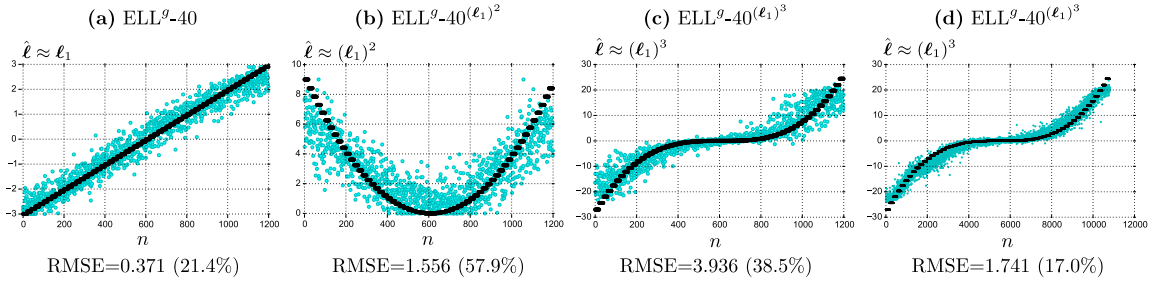

Figure 6: Plots (a) to (c) show the label estimations on test data (a single run) when different distorted versions of $\boldsymbol{\ell}_1$ are learned. The affine mapping is used. Therefore, the estimations are only generated from the slowest feature. Ground-truth values are shown in thicker black. The RMSE is expressed in parenthesis as a percentage of the chance level. Plot (d) is analogous to (c) but shows *training* data.

for this purpose. Only $N = 30$ samples (ordered by increasing label) are used to ease visualization, but the plots behave similarly for larger N. The following three graphs are employed. 1) A reordering graph (Figure 3.b) that has been extended with two edge weights $\gamma_{0,0} = 1$ and $\gamma_{N-1,N-1} = 1$ to fulfill the consistency restriction (8), which is required by the method. These weights introduce a constant scaling $N/(N+2)$ of the delta values, without any further consequence. 2) A serial graph (Section 5.1) with $K = 15$ groups of 2 samples each. 3) An ELL-4 graph (Sections 4.2–4.4) that is constructed with the original labels $\ell_1(n) = \ell(n)$, and 3 auxiliary labels computed using (40).

Figure 7 shows also that the most remarkable difference between these graphs is the number of optimal free responses with $\Delta < 2.0$, which is 14 for the reordering graph, 6 for the serial graph, and 4 for the ELL-4 graph, for the parameters above. For arbitrary parameters, the reordering, serial and ELL-L graphs have $\lfloor (N-1)/2 \rfloor$, $\lfloor (K-1)/2 \rfloor$, and, depending on the eigenvalues, up to $L \leq N-1$ optimal free responses with $\Delta < 2.0$, respectively.

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

The analysis makes clear that the serial and ELL-4 graphs are *more selective* than the reordering graph regarding the features that they consider slow. To illustrate why this might be an advantage, consider a scaled and noisy version $\hat{\mathbf{y}}_1$ of $\boldsymbol{\ell}_1$. More concretely, $\hat{y}_1(n) = \frac{\sqrt{2}}{2}\ell_1(n) + \frac{\sqrt{2}}{2}e(n)$, where $e(n)$ is an i.i.d. zero-mean unit-variance noise signal. When the reordering graph is used, the feature $\hat{\mathbf{y}}_1$ has an average Δ-value of about 1 (i.e. $\langle \Delta_{\hat{\mathbf{y}}_1} \rangle \approx 1$), and therefore such a feature would appear to be faster than an auxiliary (40) feature $\mathbf{y}_6 = \boldsymbol{\ell}_6$, because $\Delta_{\boldsymbol{\ell}_6} \approx 0.38$. Hence, a GSFA node trained with the reordering

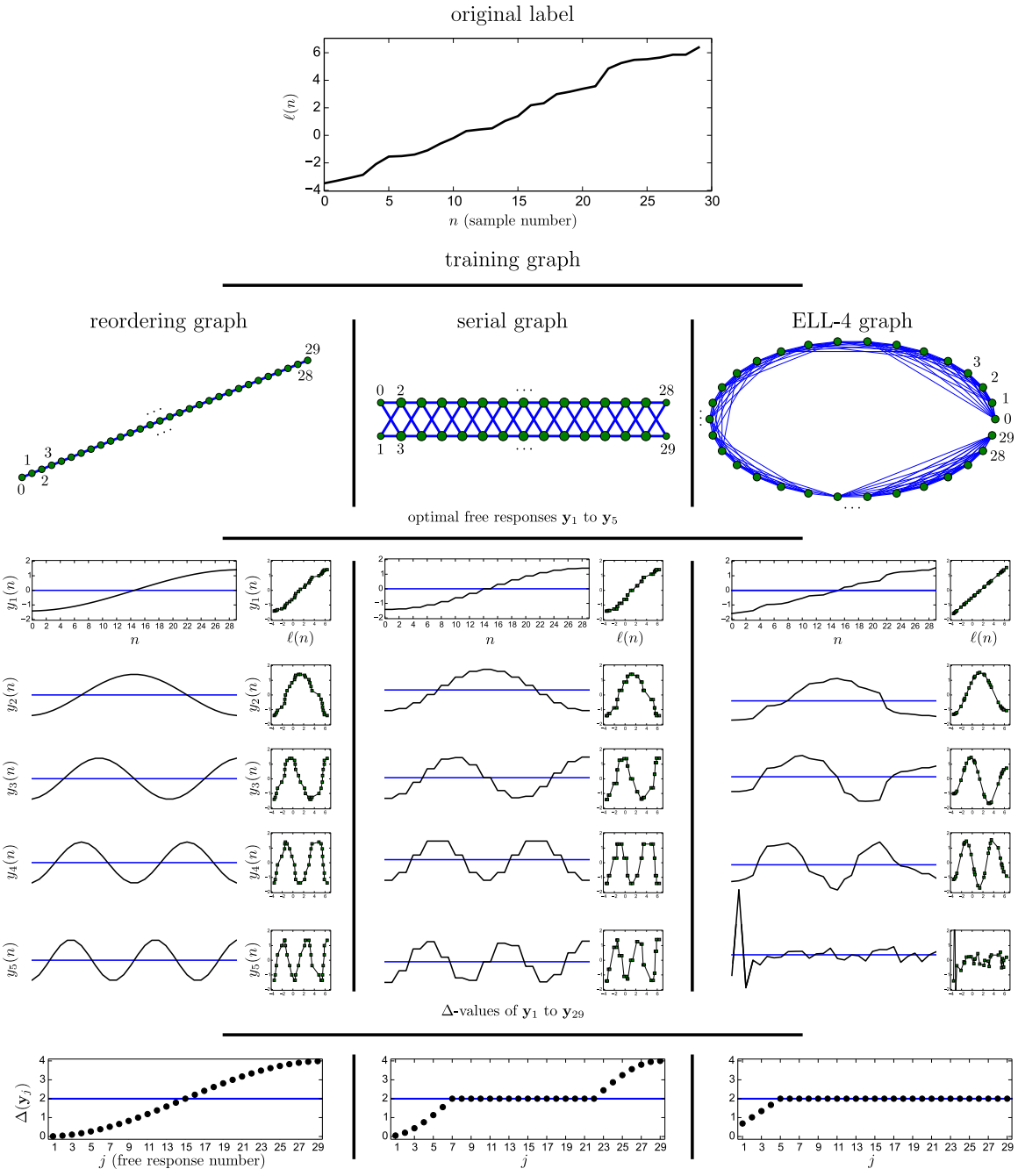

Figure 7: An arbitrary label $\ell(n)$ (top) and three graphs that can be used to learn it. The five slowest optimal free responses $\mathbf{y}_1$ to $\mathbf{y}_5$ of each graph are plotted, as well as the delta values of all optimal free responses. The ELL-4 graph is almost fully connected, but here only the strongest 30% of the connections are displayed. Samples have an index n from 0 to 29, and free responses have an index j from 1 to 29. The free responses are also plotted against the original label (smaller square plots). The polarity of the free responses was adjusted once to make them negative for the first sample.

graph would favor the extraction of $\mathbf{y}_6$ over $\hat{\mathbf{y}}_1$, even though $\hat{\mathbf{y}}_1$ is more similar to the label. In contrast, the serial and ELL-4 graphs might favor the extraction of $\hat{\mathbf{y}}_1$, because for these graphs $\Delta_{\boldsymbol{\ell}_6}$ is larger and close to 2.0.

5.3 Compact Discriminative Features for Classification

A well-known algorithm for supervised dimensionality reduction for classification is Fisher discriminant analysis (FDA). According to the theory of FDA, if there are C classes, $C-1$ features define a $C-1$ dimensional subspace that best separates the classes. In practice, one typically uses all these $C-1$ features, because all of them contain discriminative information and contribute to classification accuracy. The same holds for GSFA if the clustered training graph is used (GSFA+clustered), because in this case the features learned are equivalent to those of FDA (see Klampfl and Maass, 2010; Escalante-B. and Wiskott, 2013).

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

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

The images are represented as 48×48-pixel color (RGB) images (see Figure 8). We use only 32 out of 43 traffic signs with the most samples, so that the number of classes is a power of 2 and the number of samples is maximized. For the training data, we use the same number of samples for each class (traffic sign), namely 2,160 of them, making a total of 69,120 images. To reach 2,160 samples per class, images of some classes are used up to 6 times (since the database is unbalanced). The images used for training are distorted by a random rotation r of $-3.15 \leq r \leq 3.15$ degrees, horizontal and vertical translations Δ_x, Δ_y with $-1.73 \leq \Delta_x, \Delta_y \leq 1.73$ pixels, and a scaling factor s with $0.91 \leq s \leq 1.09$. The purpose of these distortions is to improve generalization and provide invariances to small misalignments. We use the official test data, which ensures that the test images originate from signs physically different from the ones used for training. The test data consists of 9,030 undistorted images.

We used a simple (non-hierarchical) GSFA architecture, in which PCA is applied first to reduce the dimensionality to 120 principal components. Afterwards, quadratic GSFA is

Figure 8: The 32 traffic signs learned, one image per class.

applied using different training graphs, described below. Finally, since this is a classification problem, a nearest centroid classifier is used instead of the affine mapping.

The ELL method is used to construct two training graphs with binary target labels (i.e, label values are either 1 or -1). The first one has 5 labels (compact+5) and the second one has 31 (compact+31). The target labels are defined in Table 4. Notice that the first 5 labels (for both graphs) suffice, in principle, to fully encode the class information, because they can be viewed as a binary representation of the class number.

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

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

For both graphs, we set $\mathbf{v} = \mathbf{1}$. The corresponding eigenvectors are $\mathbf{u}_j \overset{(28)}{=} Q^{-1/2}\boldsymbol{\ell}_j$, where $Q \overset{(5)}{=} N \cdot 1 = 69{,}120$ (N is the number of training images). These eigenvectors are also binary and allow for a fast computation of the covariance matrix in $\mathcal{O}(LNI^2 + I^3)$ operations, where L is the number of target labels.

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

Table 4: Target labels used to encode the class information, compactly expressed as a function of the class number c. The compact+5 graph is constructed with labels $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_5$, whereas the compact+31 graph with $\boldsymbol{\ell}_1$ to $\boldsymbol{\ell}_{31}$. The first five labels can be seen as the original ones and the rest as auxiliary.

Clustered training graph. For comparison purposes, we also consider the clustered graph (Escalante-B. and Wiskott, 2013), an efficient pre-defined graph that generates features useful for classification, see Figure 9. The optimization problem associated with this graph explicitly demands that samples from the same class should typically be mapped to similar outputs.

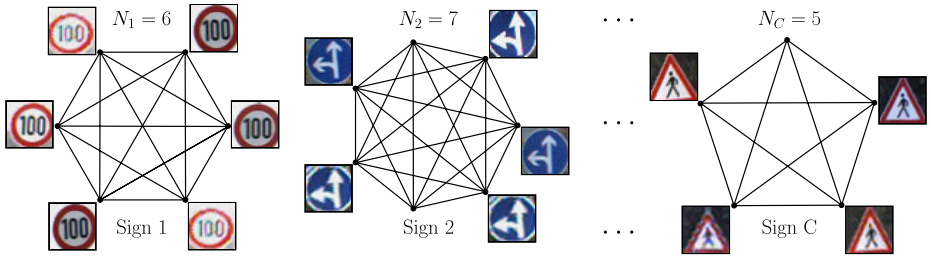

Figure 9: Illustration of a clustered training graph for classification with C classes (traffic signs). Each vertex represents a sample, and edges represent transitions. The N_c samples belonging to a class $c \in \{1, \ldots, C\}$ are connected, constituting a fully connected subgraph. Samples of different classes are not connected. Vertex weights are identical and equal to one, whereas edge weights depend on the cluster size as $\gamma_{n,n'} = 1/(N_c - 1)$, where $\mathbf{x}(n)$ and $\mathbf{x}(n')$ belong to class c and $n \neq n'$. For traffic sign recognition, we use $C = 32$ signs and $N_c = 2,160$ images per sign.

The features learned by GSFA on this graph are equivalent to those learned by Fisher discriminant analysis (FDA, see Klampfl and Maass 2010 and also compare Berkes 2005a and Berkes 2005b). This type of problem can be analyzed theoretically when the function space of SFA is unrestricted. Consistent with FDA, the first $C-1$ slow features extracted (optimal

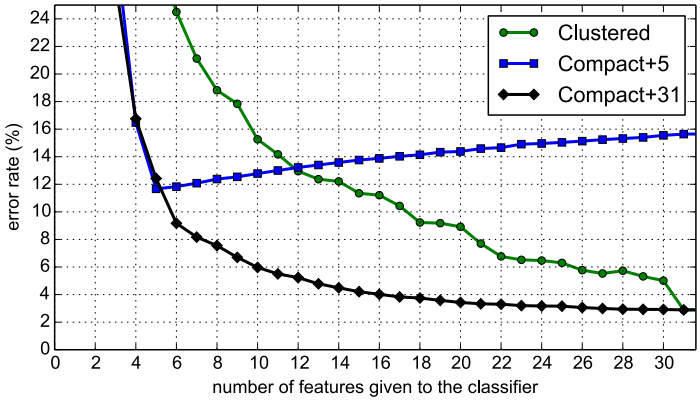

Figure 10: Classification error when GSFA is trained with the compact+5, compact+31 and clustered graph (FDA). This error is a function of the graph and the number of slow features d passed to the classifier. For the clustered graph, dropping even a single feature might increase the error rate significantly.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

free responses) are orthogonal step functions, and are piece-wise constant for samples from the same class (Berkes, 2005a).

The classification error is plotted in Figure 10, where the number of slow features d given to a nearest centroid classifier ranges from 4 to 31. For comparison, the clustered graph is also evaluated. For $d = 5$ features, the compact+5 graph results in the best accuracy with an error rate of 11.67%, against 12.42% (compact+31) and 29.74% (clustered). However, the error rate of the compact+5 graph increases if one preserves more than 5 features, indicating that additional features contain little or no discriminative information. For $6 \leq d \leq 30$, the compact+31 graph yields clearly better accuracy than the other graphs. Interestingly, for $d = 31 = C - 1$ features, the compact+31 and clustered graph give identical error rates of 2.89%, which is their top performance. In this case, the features extracted are *different* but contain the *same information* since they can be mapped to each other linearly. In other words, the first 31 free responses of both graphs describe the same subspace. Any *single* optimal free response from the compact+31 graph contains 1 bit of discriminative information (which might be redundant to the others). In contrast, the first features extracted by the clustered graph might sacrifice discriminative information to minimize within-class variance (e.g., a feature $\mathbf{y}(c) = \big((\frac{C}{2})^{1/2}, -(\frac{C}{2})^{1/2}, 0, 0, \ldots, 0\big)$ has minimal (zero) within-class variance but provides little discriminative information (less than 1 bit if $C \geq 9$), because most of the time the feature takes the value 0 and otherwise only the first two classes can be identified from it). Using $d > 31$ features does not improve accuracy in any case. For comparison, the highest performance obtained for this database during the official competition is a 0.54% error rate for all 43 signs by Ciresan et al. (2012).

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

6. Discussion

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

Supervised learning problems on high-dimensional data are of great practical importance, but they frequently result in systems with large computational demands. A common approach is to apply feature extraction, dimensionality reduction, and a supervised learning algorithm. A promising alternative approach is hierarchical GSFA (HGSFA), because its complexity scales in some cases even linearly w.r.t. the input dimensionality and the number of samples. In this context, it is especially useful to train HGSFA with an ELL graph since the resulting architecture is simple and homogeneous, as shown in Figure 1.c.

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

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

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

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

they are also good solutions to the original regression problem. If the slowest feature extracted is not sufficiently similar to the label, one can enhance the mapping from features to labels by mapping more than one output feature, and one can boost feature slowness by including auxiliary labels in the graph construction, as explained in Section 6.1.

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

6.1 Multiple and Auxiliary Labels

ELL allows learning multiple labels simultaneously, for instance to encode different aspects of the input at once (e.g. object color, size, shape, orientation). The use of multiple labels has been inspired by biological systems, where complementary information channels have been observed and appear to improve feature robustness, for example, under incomplete information (Krüger et al., 2013). Learning gender and color simultaneously yielded clearly smaller estimation errors than when these labels were estimated separately (Section 5.1). This shows that multiple label learning is not only theoretically possible, but that coding complementary information channels might boost accuracy in practice. For instance, an automatic system for face image processing might benefit from the simultaneous extraction of the subject's identity, age, gender, race, pose, and expression.

One application of multiple labels is learning auxiliary labels derived from the original one (e.g. "higher-frequency" transformations of it). The results show that encoding auxiliary labels improves accuracy (Section 5.1). Such a technique is particularly relevant for cascaded or convergent hierarchical GSFA networks, where the outputs of some GSFA nodes feed other nodes. The use of auxiliary labels has been justified based on the fact that these labels contain substantial information about the original label (Section 4.4). For instance, as explained before, the first auxiliary label $\boldsymbol{\ell}_2$ only lacks one bit of information about the original label $\boldsymbol{\ell}_1$. Therefore, even if $\boldsymbol{\ell}_1$ does not belong to the feature space of a node, the auxiliary labels might be (approximately) extracted, preserving information about $\boldsymbol{\ell}_1$. A GSFA node (or any supervised learning algorithm) higher in the hierarchy may then be

able to approximate $\boldsymbol{\ell}_1$ more accurately by making use of the information carried by the auxiliary labels. Additionally, auxiliary labels have also been justified by a smoothness heuristic, where samples n and n' having similar labels $\ell_1(n)$ and $\ell_1(n')$ should have similar output features $y_j(n)$ and $y_j(n')$, for $1 \leq j \leq J$. Without auxiliary labels only the first output feature would have this property, and the remaining features might vary quickly w.r.t. the original label.

6.2 Application of the ELL Method

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

The experiments of Section 5.1 and the analytical results of Section 5.2 show the strength of the serial graph when only a single label is available. In this case, the ELL graph provided marginally better estimations than the serial graph (an RMSE of 0.345 with the ELL^g-40 graph vs. 0.349 with the serial graph, in both cases using 3 features, the soft GC post-processing method, and averaging over 10 runs), but the computation time was 25 times longer. We verified that the difference is statistically significant.

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

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

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

6.3 Classification with ELL

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

perfect classification, the experiments show that additional target labels via auxiliary labels improve classification accuracy in practice.

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

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

Contrary to many approaches for classification based on LPP, the goal of the ELL method is strictly focused on learning the label information while being invariant to any other aspect of the data. Since we do not intend to learn the input manifold, we do not use nearest neighbors to compute the training graph. However, as shown by the (regression) experiments on simultaneous gender and color estimation, learning specific additional labels can also be useful to better disentangle the discriminative information.

6.4 Efficiency of ELL

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

In case better efficiency is still necessary, we outline a few extensions to the ELL method in Section 6.5, two of them trading accuracy for speed.

6.5 Extensions of ELL

We have devised the following possible extensions (which may be combined):

(1) Graph trimming. One might compute sparse approximations of the ELL graphs with significantly less than $\mathcal{O}(N^2)$ edges. For example, one might delete a fraction of the edges having the smallest weights or a random selection of all the edges.

(2) Sample grouping. Another method first groups the input samples according to their labels, resulting in K groups of N/K samples each. The ELL method is then applied to the average labels of the groups to compute a reduced graph with K vertices and $\mathcal{O}(K^2)$ edges. If the largest number of labels L is used, i.e. $L = I$, the reduced graph can be constructed in $\mathcal{O}(IK^2 + I^2K)$ operations. Afterwards, one can derive a specialized method to train GSFA using the reduced graph. Such a method considers the transitions between all pairs of samples of two connected groups, in the same way as the serial graph. This avoids the explicit computation of the full edge-weight matrix of size $N \times N$. The training complexity would then be $\mathcal{O}(I^2N + I^2K^2 + I^3)$ using $\mathcal{O}(I^2K + NI)$ memory. An interesting value for K is $K = \sqrt{N}$, which divides the training data in $\sqrt{N}$ groups of $\sqrt{N}$ samples each, resulting in $\mathcal{O}(I^2N + I^3)$ operations. The term I^2K in the memory complexity might be large, but one can sacrifice some performance to reduce memory usage, resulting in $\mathcal{O}(I^2N + I^2KN + I^3)$ operations and $\mathcal{O}(I^2 + NI)$ memory.

(3) Combination of graphs. Under some conditions, we show how to combine training graphs meaningfully. Consider two training graphs that fulfill the consistency restriction (9) and share the same vertices (samples) $\mathbf{x}(n)$ and vertex weights $v(n)$. Let $\mathbf{\Gamma}_1$ and $\mathbf{\Gamma}_2$ be the corresponding edge weight matrices, and $0 < \alpha < 1$ be a weighting factor. The combined graph has the same vertices and node weights, but a combined edge weight matrix $\mathbf{\Gamma}_c \stackrel{\text{def}}{=} \alpha\mathbf{\Gamma}_1 + (1-\alpha)\mathbf{\Gamma}_2$. Assume that $\mathbf{1}^T\mathbf{\Gamma}_1\mathbf{1} = \mathbf{1}^T\mathbf{\Gamma}_2\mathbf{1} = R$ (otherwise the edge weights of one graph can be scaled). Since the vertices and vertex weights are identical, a feature $\mathbf{y}$ that is feasible for one of the graphs is also feasible for the other two. Let $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1}$ and $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2}$ be the Δ value of $\mathbf{y}$ for the original graphs. Then, $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_c} \stackrel{(10)}{=} \alpha\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1} + (1-\alpha)\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2}$. This implies that if a feature $\mathbf{y}$ has a Δ-value smaller than an arbitrary constant β (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_1} < \beta$) and it is not larger than β in the second one (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_2} \leq \beta$), it can be warranted that it will be also smaller than β in the combined graph (i.e., $\Delta_{\mathbf{y}}^{\mathbf{\Gamma}_c} < \beta$) for any $0 < \alpha < 1$. This property may be useful to create graphs with optimal free responses that span various labels.

The third extension can be used to combine ELL and/or pre-defined graphs without distinction. In an upcoming work, Escalante-B. and Wiskott (2016) combine three efficient pre-defined graphs for face image analysis: two clustered graphs for classification of race and gender, and a serial graph for the estimation of age. The accuracy for age estimation on the MORPH-II database using the combined graph (and an improved version of HGSFA) is a mean average error (MAE) of 3.50 years, which is more accurate than the current state-of-the-art systems for this database (Yi et al., 2015 with an MAE of 3.63 years and Guo and Mu, 2014 with an MAE of 3.92).

6.6 Future Work

In future work, we would like to explore the extensions above. For example, it seems reasonable to combine the serial and the reordering graph. The first one has a large number of edges, which provides good generalization, whereas the second one does not incur in the quantization error of the serial graph caused by its grouping of samples. Thus, the combination might improve accuracy.

Although the ELL method supports multiple labels, it might be less effective if the number of target labels L is large. However, in this case the labels are frequently categorical

and sparse. Therefore, we would like to investigate methods that concentrate the label information (e.g., by computing a compressed representation of the labels) to reduce the number of effective labels.

We have proposed a method to set the auxiliary labels, and have explained why it is meaningful to use them. However, one could choose them according to some optimization criterion, e.g., to explicitly maximize estimation accuracy. Also the assignment of the eigenvalues could be optimized. Apparently it is difficult to determine optimal auxiliary labels and their eigenvalues analytically. For classification with $C = 32$ classes, *linearly* decreasing eigenvalues (for the auxiliary labels) provided great results, but other eigenvalues might be better if C is very large.

In the ELL method, several eigenvalues were set to zero and the corresponding eigenvectors remained unspecified. As suggested by a reviewer, one could use these eigenvectors and eigenvalues to construct graphs with special structural constraints, such as a minimum and maximum number of edges per vertex.

6.7 Conclusion

Hierarchical processing and the slowness principle are two powerful brain-inspired learning principles. The strength of SFA originates from its theoretical foundations in the field of learning of invariances and the generality of the slowness principle. For practical supervised learning applications, HGSFA provides good accuracy and efficiency and still profits from strong theoretical foundations. An advantage of relying on such general principles is that the resulting algorithms are application independent and not confined to a particular problem or input feature representation. Of course, fine tuning the network parameters and the integration of problem-specific knowledge are always possible for additional performance. The proposed ELL method explores the limits of HGSFA and is valuable as a theoretical tool for the analysis and design of training graphs. However, the results show that with certain adaptations (e.g., the use of supervised post-processing) it is also sufficiently robust to be applied to practical computer vision and machine learning tasks.

References

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003.

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.

D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333 – 338, 2012. Selected Papers from IJCNN 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. How to solve classification and regression problems with an unsupervised learning algorithm based on the slowness principle. (in preparation), 2012.

A. N. Escalante-B. and L. Wiskott. How to solve classification and regression problems on high-dimensional data with a supervised extension of Slow Feature Analysis. *Journal of Machine Learning Research*, 14:3683–3719, December 2013.

A. N. Escalante-B. and L. Wiskott. Improved graph-based sfa: Information preservation complements the slowness principle. e-print arXiv:1601.03945, 01 2016.

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 and pose estimation with Slow Feature Analysis. *Neural Computation*, 23(9):2289–2323, 2011.

G. Guo and G. Mu. A framework for joint estimation of age, gender and ethnicity on a large database. *Image and Vision Computing*, 32(10):761–770, 2014. Best of Automatic Face and Gesture Recognition 2013.

X. He. *Locality Preserving Projections*. PhD thesis, Computer Science Department, The University of Chicago, Chicago, IL, USA, 2005.

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.

S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *IJCNN*, number 1288, 2013.

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.

N. Krüger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1847–1871, 2013.

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

E. M. Rehn and H. Sprekeler. Nonlinear supervised locality preserving projections for visual pattern discrimination. In *22nd International Conference on Pattern Recognition*, pages 1568–1573, 2014.

Singular Inversions Inc. FaceGen SDK. `http://www.facegen.com`, 2008.

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

L. Wiskott. Learning invariance manifolds. In *Proc. of 5th Joint Symposium on Neural Computation, San Diego, CA, USA*, volume 8, pages 196–203. Univ. of California, 1998.

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

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

D. Yi, Z. Lei, and S. Li. Age estimation by multi-scale convolutional network. In *Computer Vision – ACCV 2014*, volume 9005 of *Lecture Notes in Computer Science*, pages 144–158. 2015.

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