Mach Learn (2012) 89:67–86
DOI 10.1007/s10994-012-5300-0

Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis

Wendelin Böhmer · Steffen Grünewälder · Hannes Nickisch · Klaus Obermayer

Received: 31 October 2011 / Accepted: 22 May 2012 / Published online: 13 June 2012

Abstract Without non-linear basis functions many problems can not be solved by linear algorithms. This article proposes a method to automatically construct such basis functions with *slow feature analysis* (SFA). Non-linear optimization of this unsupervised learning method generates an orthogonal basis on the unknown latent space for a given time series. In contrast to methods like PCA, SFA is thus well suited for techniques that make direct use of the latent space. Real-world time series can be complex, and current SFA algorithms are either not powerful enough or tend to over-fit. We make use of the *kernel trick* in combination with *sparsification* to develop a kernelized SFA algorithm which provides a powerful function class for large data sets. Sparsity is achieved by a novel *matching pursuit* approach that can be applied to other tasks as well. For small data sets, however, the kernel SFA approach leads to over-fitting and numerical instabilities. To enforce a stable solution, we introduce *regularization* to the SFA objective. We hypothesize that *our algorithm generates a feature space that resembles a Fourier basis in the unknown space of latent variables underlying a given real-world time series*. We evaluate this hypothesis at the example of a *vowel classification* task in comparison to *sparse kernel PCA*. Our results show excellent classification accuracy and demonstrate the superiority of kernel SFA over kernel PCA in encoding latent variables.

Editors: Dimitrios Gunopulos, Donato Malerba, and Michalis Vazirgiannis.

W. Böhmer (✉) · K. Obermayer
Neural Information Processing Group, Technische Universität Berlin, Berlin, Germany
e-mail: wendelin@ni.tu-berlin.de

K. Obermayer
e-mail: klaus.obermayer@tu-berlin.de

S. Grünewälder
Centre for Computational Statistics and Machine Learning, University College London, London, UK
e-mail: steffen@cs.ucl.ac.uk

H. Nickisch
Philips Research Laboratories, Hamburg, Germany
e-mail: hannes.nickisch@philips.com

Keywords Time series · Latent variables · Unsupervised learning · Slow feature analysis · Sparse kernel methods · Linear classification

1 Introduction

This article is concerned with the automatic construction of non-linear basis functions for linear algorithms. This is of particular importance if the original space of inputs $\mathcal{X} \subseteq \mathbb{R}^l$ can not support an adequate linear solution.

New algorithms are often initially formulated using a linear function class $\mathcal{F}_\mathcal{X} := \{f(\boldsymbol{x}) = \sum_{i=1}^{l} w_i x_i | \boldsymbol{w} \in \mathbb{R}^l\}$ of possible solutions $f : \mathcal{X} \to \mathbb{R}$. Examples include *linear regression* and the discrimination function for *linear classification*. We adopt the point of view that the desired solutions are actually defined on the domain of an unknown low dimensional space of latent variables Θ, which is embedded in the high dimensional space of sensor observations $\boldsymbol{x} \in \mathcal{X}$. The restriction to linear functions $f \in \mathcal{F}_\mathcal{X}$, however, can prevent a suitable solution because (a) Θ might be non-linearly embedded in $\mathcal{X}$ and (b) the true solution f^* might be non-linear in Θ. This can be compensated by the introduction of a feature space $\Phi := \{\boldsymbol{\phi}(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{X}\}$ with mappings $\phi_i : \mathcal{X} \to \mathbb{R}, \forall i \in \{1, \ldots, p\}$, such that:

- $\phi_i(\cdot), \forall i \in \{1, \ldots, p\}$, is non-linear in $\mathcal{X}$ to encode Θ rather than $\mathcal{X}$.
- $\{\phi_i\}_{i=1}^{p}$ constitutes a well behaving functional basis in Θ, e.g. a Fourier basis.
- Dimensionality p is as low as possible to reduce the number of training samples required to reliably estimate $f \in \mathcal{F}_\Phi := \{f(\boldsymbol{x}) = \sum_{i=1}^{p} w_i \phi_i(\boldsymbol{x}) | \boldsymbol{w} \in \mathbb{R}^p\}$.

This article outlines an approach to automatically construct such a feature space Φ for a given time series. The critical assumption is that there exist an unknown low dimensional space of latent variables Θ, that generated the observed data $\boldsymbol{x} \in \mathcal{X}$ by some unknown stochastic process $\Theta \to \mathcal{X}$. It is important to understand that we do not talk about the true underlying *cause* of the data, but a description Θ that suffices to generate the training samples $\{\boldsymbol{x}_t\}_{t=1}^{n} \subset \mathcal{X}$. Additionally, we restrict ourselves here to the most general encoding of Θ, i.e. we assume no additional information about labels or classes of the training samples.[1] This is achieved by an unsupervised learning principle called *slow feature analysis* (SFA, Wiskott and Sejnowski 2002). SFA aims for temporally coherent features out of high dimensional and/or delayed sensor measurements. Given an infinite time series and an unrestricted function class, the learned features will be a Fourier basis[2] in Θ (Wiskott 2003; Franzius et al. 2007). Although there have been numerous studies highlighting its resemblance to biological sensor processing (Wiskott and Sejnowski 2002; Berkes and Wiskott 2005; Franzius et al. 2007), the method has not yet found its way in the engineering community that focuses on the same problems. One of the reasons is undoubtedly the lack of an easily operated non-linear extension that is powerful enough to generate a Fourier basis.

The major contribution of this article is to provide such an extension in the form of a kernel SFA algorithm and to demonstrate its application at the example of linear classification. An approach on kernelizing SFA has previously been made by Bray and Martinez (2002) and is reported to work well with a large image data set. Small training sets, however, lead to numerical instabilities in any kernel SFA algorithm. Our goal is to provide an algorithm

[1] Extensions that include such information are discussed in Sect. 6.

[2] This holds only for *factorizing spaces* Θ, i.e. independent boundary conditions for each dimension. However, more general statements can be derived for any *ergodic* Markov chain.

that can be applied to both of the cases above and to document the pitfalls that arise in its application on real-world time series. This algorithm has previously been presented in a conference paper, which has been the basis for this article (Böhmer et al. 2011). Beyond the scope of the original paper, we provide evidence for its main hypothesis and evaluate linear classification algorithms in the feature spaces generated by kernel SFA and kernel PCA (Schölkopf et al. 1997).

Although formulated as a linear algorithm, SFA was originally intended to be applied on the space of polynomials, e.g. quadratic (Wiskott and Sejnowski 2002) or cubic (Berkes and Wiskott 2005). The polynomial expansion of potentially high dimensional data, however, spans an impractically large space of coefficients. Hierarchical application of quadratic SFA has been proposed to solve this problem (Wiskott and Sejnowski 2002). Although proven to work in complex tasks (Franzius et al. 2007), this approach involves a multitude of hyper-parameters and no easy way to counteract inevitable over-fitting is known so far. It appears biologically plausible, but is definitely not easy to operate.

A powerful alternative to polynomial expansions are *kernel methods*. Here the considered feature maps $\phi : \mathcal{X} \to \mathbb{R}$ are elements of a *reproducing kernel Hilbert space* $\mathcal{H}$. For many optimization problems a *representer theorem* holds in line with Wahba (1990), if a regularization term is used. Such theorems guarantee that the optimal solution for a given training set exists within the span of *kernel functions*, which are parametrized by training samples. Depending on the kernel, continuous functions can be approximated arbitrarily well in $\mathcal{H}$ and a mapping $\phi \in \mathcal{H}$ is thus very powerful (Shawe-Taylor and Cristianini 2004).

There are, however, fundamental drawbacks in a kernel approach to SFA. First, choosing feature mappings from a powerful Hilbert space is naturally prone to over-fitting. More to the point, kernel SFA shows *numerical instabilities* due to it's unit variance constraint (see Sects. 3 and 5). This tendency has been analytically shown for the related *kernel canonical correlation analysis* (Fukumizu et al. 2007). We introduce a regularization term for the SFA objective to enforce a stable solution. Secondly, kernel SFA is based on a *kernel matrix* of size $\mathcal{O}(n^2)$, where n is the number of training samples. This is not feasible for large training sets. Our approach approximates the optimal solution by projecting into a sparse subset of the data. The choice of this subset, however, is a crucial decision.

The question how *many* samples should be selected can only be answered empirically. We compare two state-of-the-art sparse subset selection algorithms that approach this problem very differently: (1) A fast *online algorithm* (Csató and Opper 2002) that must recompute the whole solution to change the subset's size. (2) A computational costly *matching pursuit approach* to sparse kernel PCA (Smola and Schölkopf 2000) that incrementally augments the selected subset. To obtain a method that is *both* fast and incremental we derive a novel matching pursuit approach to the first algorithm.

Bray and Martinez (2002) have previously introduced a *kernel SFA* algorithm that incorporates a simplistic sparsity scheme. Instead of the well-established framework of Wiskott and Sejnowski (2002), they utilize the cost function of Stone (2001) based on long and short term variances without explicit constraints. Due to a high level of sparsity, their approach does not require function regularization. We show that the same holds for our algorithm if the sparse subset is only a small fraction of the training data. However, for larger fractions additional regularization becomes inevitable.

Theoretic predictions by Wiskott (2003) and Franzius et al. (2007) suggests that *regularized sparse kernel SFA features will resemble a Fourier basis in the space of latent variables Θ for a given real-world time series*. To verify this hypothesis, we perform a *vowel classification task* on spoken words. None of the tested *linear classification* algorithms (Bishop 2006) were able to solve the task without non-linear basis functions. Our work extends

Berkes (2005), who used non-linear SFA to classify images of hand written digits. His work was based on artificial time series rather that real-world data, though. Our results show excellent classification accuracy of more than 97 % and superior encoding of latent variables by *kernel SFA* in comparison with *kernel PCA* (Schölkopf et al. 1997). To the best of our knowledge, this work and its predecessor (Böhmer et al. 2011) are the first attempt to apply non-linear SFA as a pre-processing for audio data.

In the following section, we first review a variety of linear classification algorithms, which we will use to compare the constructed feature spaces Φ. In Sect. 3 we formulate the general SFA optimization problem and derive a *regularized sparse kernel SFA algorithm*. In Sect. 4 the sparse subset selection is introduced and a novel matching pursuit algorithm derived. Section 5 evaluates both the SFA algorithm and the generated feature space on a vowel classification task, followed by a discussion of our method and possible extensions in Sect. 6.

2 Linear classification algorithms

Classification aims to assign to each test sample $\boldsymbol{x}^* \in \mathcal{X}$ a class $c^* \in \mathcal{C}$, based on a given training set of samples $\{\boldsymbol{x}_t\}_{t=1}^n$ and their assigned classes $\{c_t\}_{t=1}^n$. In the case of two classes $\mathcal{C} = \{-1, +1\}$, *linear classification* aims for a *discrimination function* $f \in \mathcal{F}_\Phi = \{f(\boldsymbol{x}) = \sum_{i=1}^p w_i \phi_i(\boldsymbol{x}) | \boldsymbol{w} \in \mathbb{R}^p\}$ with $\text{sign}(f(\boldsymbol{x}_t)) = c_t, \forall t \in \{1, \ldots, n\}$. All of the below algorithms are excellently discussed by Bishop (2006).

2.1 Least squares classification/LDA/FDA

The most straight forward approach to classification is to encode each class label with a 1-of-$|\mathcal{C}|$ binary label vector[3] and to perform *least-squares* (LS) regression of $\boldsymbol{g} \in \mathcal{F}_\Phi^{|\mathcal{C}|}$ against this multivariate target (Bishop 2006). The function $g_i \in \mathcal{F}_\Phi$ with the highest output on a test sample yields the corresponding class. For example, in the two-class case $\mathcal{C} = \{-1, +1\}$, the *discrimination function* $f \in \mathcal{F}_\Phi$ that separates the classes is given by $f(\boldsymbol{x}) = g_2(\boldsymbol{x}) - g_1(\boldsymbol{x})$. The LS solution is found by minimizing

$$\min_{\boldsymbol{g} \in \mathcal{F}_\Phi^{|\mathcal{C}|}} \sum_{i=1}^{|\mathcal{C}|} \sum_{t=1}^{n} \big(g_i(\boldsymbol{x}_t) - T_{it}\big)^2. \tag{1}$$

As each linear function $g_i \in \mathcal{F}_\Phi$ is characterized by a vector $\boldsymbol{w}_i \in \mathbb{R}^p$, the above cost function has the unique[4] analytical solution

$$\boldsymbol{w}_i = \mathbf{C}^{-1} \boldsymbol{\mu}_i, \quad \text{where } C_{ij} := \frac{1}{n} \sum_{t=1}^{n} \phi_i(\boldsymbol{x}_t) \phi_j(\boldsymbol{x}_t) \quad \text{and} \quad \boldsymbol{\mu}_i := \frac{1}{n} \sum_{t=1}^{n} T_{it} \boldsymbol{\phi}(\boldsymbol{x}_t). \tag{2}$$

The terms $\boldsymbol{\mu}_i \in \mathbb{R}^p$ and $\mathbf{C} \in \mathbb{R}^{p \times p}$ are the empirical estimates of a weighted ith class mean and the second moment correlation matrix in feature space Φ. Notice that in *zero mean* and

[3] Here the ith class label is a $|\mathcal{C}|$ dimensional vector which is zero everywhere except for the ith entry, which is one. The result is a matrix $\mathbf{T}$ with $T_{it} = 1$ if $c_t = i$ and 0 otherwise.

[4] The solution is unique if $\mathbf{C}$ is invertible. If not, a *Moore-Penrose pseudoinverse* of $\mathbf{C}$ can still yield acceptable results.

unit variance feature spaces, which we will compare in Sect. 5, the computation reduces to the estimation of the class means $\boldsymbol{\mu}_i$. In general, however, the algorithm has a computational complexity of $\mathcal{O}(p^2n + p^3)$.

If Φ has zero mean w.r.t. all samples and the classes are equally distributed, $\mathbf{C}$ is the estimate of the covariance matrix and the solution is equivalent to a Gaussian estimate of the *class probability densities* with the restriction that all classes share the same covariance (Bishop 2006). This approach to derive the discrimination function out of Gaussian density estimates is also called *linear discrimination analysis* (LDA). With small modifications to the target weights (Duda and Hart 1973), the solution of (2) becomes also equivalent to *Fisher discrimination analysis* (FDA, Fisher 1936). As the empirical differences in our experiments were marginal, we restricted ourselves in Sect. 5 to LDA. Note, however, that due to the strong assumption of equal covariances, these algorithms are known to perform poorly if this assumption is not met.

2.2 Perceptron classification

The algorithm that started the field of *artificial neural networks* (Rosenblatt 1962) can classify two-class problems with $\mathcal{C} = \{-1, +1\}$ by assigning a signum function onto the output of the discrimination function $f \in \mathcal{F}_\Phi$, i.e. $y(\boldsymbol{x}_t) = \operatorname{sign}(f(\boldsymbol{x}_t))$. At sample $\boldsymbol{x}_t$, the online version of this algorithm updates the weight vector $\boldsymbol{w} \in \mathbb{R}^p$ of $f(\boldsymbol{x}_t) = \sum_{i=1}^p w_i \phi_i(\boldsymbol{x}_t)$ by $\Delta w_i^{(t)} := -\phi_i(\boldsymbol{x}_t)c_t$, but *only* if the prediction $y(\boldsymbol{x}_t)$ was incorrect. The update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \Delta \boldsymbol{w}^{(t)}$ with learning rate $0 < \alpha \leq 1$ is repeated until all training samples are classified correctly. The algorithm is guaranteed to converge for linear separable problems (Rosenblatt 1962).

We do not expect linearly separable problems and thus devised a batch algorithm with a decaying learning rate, i.e. $\alpha_i := 1/i$ at iteration i until convergence.

$$w_j^{(i+1)} := w_j^{(i)} - \frac{\alpha_i}{\sum_{t=1}^n d_t^{(i)}} \sum_{t=1}^n d_t^{(i)} c_t \phi_j(\boldsymbol{x}_t), \quad \text{where } d_t^{(i)} := \begin{cases} 1, & \text{if } y(\boldsymbol{x}_t) \neq c_t \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Convergence criteria that stay comparable over multiple feature space sizes are hard to formulate and so we ran the algorithm for 100 iterations, which appeared to be sufficient for convergence in all feature spaces Φ. The computational complexity for i iterations of the above algorithm is $\mathcal{O}(pin)$.

2.3 Logistic regression

Rather than a *signum* in the perceptron algorithm, the *logistic regression* algorithm applies the *logistic sigmoid* function $\sigma(a) := (1 + \exp(-a))^{-1}$ onto the discrimination function $f \in \mathcal{F}_\Phi$ to approximate the probability $P(c_t = 1|\boldsymbol{x}_t) \approx y(\boldsymbol{x}_t) := \sigma(f(\boldsymbol{x}_t))$. By encoding the classes $\mathcal{C} := \{0, 1\}$, one can express the likelihood function in the two-classes case (Bishop 2006) for a given discriminant function $f(\cdot)$ as

$$p(c_1, \ldots, c_n | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{t=1}^n \big(y(\boldsymbol{x}_t)\big)^{c_t} \big(1 - y(\boldsymbol{x}_t)\big)^{1-c_t}. \tag{4}$$

This likelihood can be maximized by the *Newton-Raphson scheme* (Fletcher 1987). The method requires the calculation of gradient and Hessian of the log-likelihood. As the sigmoid $\sigma(\cdot)$ is nonlinear, the Hessian $\mathbf{H}^{(i)}$ is no longer independent of the discrimination

function $f^{(i)} \in \mathcal{F}_\Phi$ and has to be recomputed every iteration i. The resulting algorithm is called *iterative reweighted least squares* (Rubin 1983):

$$w_j^{(i+1)} := w_j^{(i)} - \sum_{k=1}^{p}\sum_{t=1}^{n}\left(\mathbf{H}^{(i)}\right)^{-1}_{jk}\phi_k(\boldsymbol{x}_t)\left(y^{(i)}(\boldsymbol{x}_t) - c_t\right),$$

$$\text{where } H_{jk}^{(i)} = \sum_{t=1}^{n}\phi_j(\boldsymbol{x}_t)y^{(i)}(\boldsymbol{x}_t)\left(1 - y^{(i)}(\boldsymbol{x}_t)\right)\phi_k(\boldsymbol{x}_t). \tag{5}$$

The algorithm showed clear signs of over-fitting in large feature spaces. Regularization[5] of the Hessian $\mathbf{H}^{(i)} \leftarrow \mathbf{H}^{(i)} + \eta\mathbf{I}$ has proven very effective in compensating for this problem (see Sect. 5.4). As the perceptron algorithm, we ran the algorithm for 100 iterations, which appeared to be sufficient for convergence. With i iterations the computational complexity is $\mathcal{O}(p^2in + p^3i)$.

3 Slow feature analysis

Let $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X}$ be a sequence of n observations. The goal of *slow feature analysis* (SFA) is to find a set of mappings $\phi_i : \mathcal{X} \to \mathbb{R}$, $i \in \{1, \ldots, p\}$, such that the values $\phi_i(\boldsymbol{x}_t)$ change slowly over time (Wiskott and Sejnowski 2002). A mapping ϕ_i's change over time is measured by the *discrete temporal derivative* $\dot{\phi}_i(\boldsymbol{x}_t) := \phi_i(\boldsymbol{x}_t) - \phi_i(\boldsymbol{x}_{t-1})$. The SFA objective (called *slowness*, (6)) is to minimize the squared mean of this derivative, where $\mathbb{E}_t[\cdot]$ is the sample mean[6] over all available indices t:

$$\min s(\phi_i) := \mathbb{E}_t\left[\dot{\phi}_i^2(\boldsymbol{x}_t)\right] \quad \text{(Slowness)}. \tag{6}$$

To avoid trivial solutions as well as to deal with mixed sensor observations in different scales, the mappings are forced to change uniformly, i.e. to exhibit *unit variance* ((7) and (8)). *Decorrelation* ensures every mapping to extract unique information (9). The last degree of freedom is eliminated by demanding *order* (10), leading to the following constraints:

$$\mathbb{E}_t\left[\phi_i(\boldsymbol{x}_t)\right] = 0 \quad \text{(Zero Mean)} \tag{7}$$

$$\mathbb{E}_t\left[\phi_i^2(\boldsymbol{x}_t)\right] = 1 \quad \text{(Unit Variance)} \tag{8}$$

$$\mathbb{E}_t\left[\phi_i(\boldsymbol{x}_t)\phi_j(\boldsymbol{x}_t)\right] = 0, \quad \forall j \neq i \quad \text{(Decorrelation)} \tag{9}$$

$$\forall j > i : s(\phi_i) \leq s(\phi_j) \quad \text{(Order)}. \tag{10}$$

The principle of slowness, although not the above definition, has been used very early in the context of neural networks (Földiák 1991; Becker and Hinton 1992). Recent variations of SFA differ either in the objective (Bray and Martinez 2002) or the constraints (Einhäuser et al. 2005; Wyss et al. 2006). For some simplified cases, given an infinite time series and unrestricted function class, it can be analytically shown that SFA solutions converge to trigonometric polynomials w.r.t. the underlying latent variables (Wiskott 2003;

[5]This regularization approach is equivalent to *weight decay*, i.e. an additional regularization term $\eta\|\boldsymbol{w}\|_2^2$ in the log-likelihood objective.

[6]The samples are not i.i.d. and must be drawn by an *ergodic* Markov chain in order for the empirical mean to converge in the limit (Meyn and Tweedie 1993).

Franzius et al. 2007). In reality those conditions are never met and one requires a function class that can be adjusted to the data set at hand.

3.1 Kernel SFA

Let the considered mappings $\phi_i : \mathcal{X} \to \mathbb{R}$ be elements of a *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}$, with corresponding positive semi-definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The *reproducing property* of those kernels allows $\forall \phi \in \mathcal{H} : \phi(\boldsymbol{x}) = \langle \phi, \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$, in particular $\langle \kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{x}') \rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$. The *representer theorem* ensures[7] the solution to be in the span[8] of *support functions*, parametrized by the training data (Wahba 1990), i.e. $\phi = \sum_{t=1}^{n} a_t \kappa(\cdot, \boldsymbol{x}_t)$. Together those two relationships set the basis for the *kernel trick* (see e.g. Shawe-Taylor and Cristianini 2004).

The zero mean constraint can be achieved by centering all involved *support functions* $\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^{n}$ in $\mathcal{H}$ (for details see Sect. 3.2). Afterwards, the combined kernel SFA (K-SFA) optimization problem for all p mappings $\boldsymbol{\phi}(\cdot) \in \mathcal{H}^p$ is

$$\min_{\boldsymbol{\phi} \in \mathcal{H}^p} \sum_{i=1}^{p} \mathbb{E}_t\big[\dot{\phi}_i^2(\boldsymbol{x}_t)\big], \quad \text{s.t. } \mathbb{E}_t\big[\boldsymbol{\phi}(\boldsymbol{x}_t)\boldsymbol{\phi}(\boldsymbol{x}_t)^\top\big] = \mathbf{I}. \tag{11}$$

Through application of the kernel trick, the problem can be reformulated as

$$\min_{\mathbf{A} \in \mathbb{R}^{n \times p}} \frac{1}{n-1} \mathbf{tr}\big(\mathbf{A}^\top \mathbf{K} \mathbf{D} \mathbf{D}^\top \mathbf{K}^\top \mathbf{A}\big) \qquad \text{s.t. } \frac{1}{n} \mathbf{A}^\top \mathbf{K} \mathbf{K}^\top \mathbf{A} = \mathbf{I}, \tag{12}$$

where $K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the *kernel matrix* and $\mathbf{D} \in \mathbb{R}^{n \times n-1}$ the *temporal derivation matrix*[9] with all zero entries except $\forall t \in \{1, \ldots, n-1\} : D_{t,t} = -1$ and $D_{t+1,t} = 1$.

Sparse kernel SFA If one assumes the feature mappings within the span of another set of data $\{\boldsymbol{z}_i\}_{i=1}^{m} \subset \mathcal{X}$ (e.g. a sparse subset of the training data, often called *support vectors*), the *sparse kernel matrix* $\mathbf{K} \in \mathbb{R}^{m \times n}$ is defined as $K_{ij} = \kappa(\boldsymbol{z}_i, \boldsymbol{x}_j)$ instead. The resulting algorithm will be called *sparse kernel SFA*. Note that the representer theorem no longer applies and therefore the solution merely approximates the optimal mappings in $\mathcal{H}$. Both optimization problems have identical solutions if $\forall t \in \{1, \ldots, n\} : \kappa(\cdot, \boldsymbol{x}_t) \in \mathbf{span}(\{\kappa(\cdot, \boldsymbol{z}_i)\}_{i=1}^{m})$, e.g. $\{\boldsymbol{z}_i\}_{i=1}^{n} = \{\boldsymbol{x}_t\}_{t=1}^{n}$.

Regularized sparse kernel SFA The Hilbert spaces corresponding to some of the most popular kernels are equivalent to an infinite dimensional space of continuous functions (Shawe-Taylor and Cristianini 2004). One example is the *Gaussian kernel* $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$. Depending on hyper-parameter σ and data distribution, this can obviously lead to over-fitting. Less obvious, however, is the tendency of kernel SFA to become numerically unstable for large σ, i.e. to violate the unit variance constraint. Fukumizu et al. (2007) have shown this analytically for the related *kernel canonical correlation analysis*. Note that both problems do not affect sufficiently sparse solutions, as sparsity reduces the

[7] Technically this holds only when the solution is regularized in $\mathcal{H}$—which we will do later.

[8] Note that the solutions $\phi_i(\cdot)$ are thus *linear functions* of *inner products* in $\mathcal{H}$.

[9] In Sect. 5 we will generate feature mappings from a collection of time series, i.e. multiple words. If all words are assembled into one large time series, the transition from one word to the next can be excluded from optimization by setting the respective entry in $\mathbf{D}$ to zero.

function complexity and sparse kernel matrices $\mathbf{K}\mathbf{K}^\top$ are more robust w.r.t. eigenvalue decompositions.

One countermeasure is to introduce a *regularization term* to stabilize the sparse kernel SFA algorithm, which thereafter will be called *regularized sparse kernel SFA* (RSK-SFA). Our approach penalizes the squared Hilbert-norm of the selected functions $\|\phi_i\|^2_{\mathcal{H}}$ by a *regularization parameter* λ. Analogous to K-SFA the kernel trick can be utilized to obtain the new objective:

$$\min_{\mathbf{A}\in\mathbb{R}^{m\times p}} \frac{1}{n-1}\mathbf{tr}\big(\mathbf{A}^\top\mathbf{K}\mathbf{D}\mathbf{D}^\top\mathbf{K}^\top\mathbf{A}\big) + \lambda\mathbf{tr}\big(\mathbf{A}^\top\bar{\mathbf{K}}\mathbf{A}\big)$$
$$\text{s.t.}\frac{1}{n}\mathbf{A}^\top\mathbf{K}\mathbf{K}^\top\mathbf{A} = \mathbf{I}, \tag{13}$$

where $\bar{K}_{ij} = \kappa(\boldsymbol{z}_i, \boldsymbol{z}_j)$ is the kernel matrix of the support vectors.

3.2 The RSK-SFA algorithm

The RSK-SFA algorithm (Algorithm 1) is closely related to the linear SFA algorithm of Wiskott and Sejnowski (2002). It consists of three phases: (1) fulfilling zero mean by centering, (2) fulfilling unit variance and decorrelation by sphering and (3) minimizing the objective by rotation.

Zero mean To fulfill the zero mean constraint, one centers the *support functions* $\{g_i\}_{i=1}^m \subset \mathcal{H}$ w.r.t. the data distribution, i.e. $g_i(\cdot) := \kappa(\cdot, \boldsymbol{z}_i) - \mathbb{E}_t[\kappa(\boldsymbol{x}_t, \boldsymbol{z}_i)]\mathbf{1}_{\mathcal{H}}(\cdot)$, where $\forall \boldsymbol{x} \in \mathcal{X}$: $\mathbf{1}_{\mathcal{H}}(\boldsymbol{x}) = \langle \mathbf{1}_{\mathcal{H}}, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}} := 1$ is the constant function[10] in Hilbert space $\mathcal{H}$. Although $\forall\phi \in \mathbf{span}(\{g_i\}_{i=1}^m) : \mathbb{E}_t[\phi(\boldsymbol{x}_t)] = 0$ already holds, it is of advantage to center the support functions as well w.r.t. each other (Schölkopf et al. 1998), i.e. $\hat{g}_i := g_i - \mathbb{E}_j[g_j]$. The resulting transformation of support functions on the training data can be applied directly onto the kernel matrices $\mathbf{K}$ and $\bar{\mathbf{K}}$:

$$\hat{\mathbf{K}} := \left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{K}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right) \tag{14}$$

$$\hat{\bar{\mathbf{K}}} := \left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\bar{\mathbf{K}}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right), \tag{15}$$

where $\mathbf{1}_m$ and $\mathbf{1}_n$ are one-vectors of dimensionality m and n, respectively.

Unit variance and decorrelation Analogue to linear SFA, we first project into the normalized eigenspace of $\frac{1}{n}\hat{\mathbf{K}}\hat{\mathbf{K}}^\top =: \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$. The procedure is called *sphering* or *whitening* and fulfills the constraint in (13), invariant to further rotations $\mathbf{R} \in \mathbb{R}^{m\times p} : \mathbf{R}^\top\mathbf{R} = \mathbf{I}$.

$$\mathbf{A} := \mathbf{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{R} \quad \Rightarrow \quad \frac{1}{n}\mathbf{A}^\top\hat{\mathbf{K}}\hat{\mathbf{K}}^\top\mathbf{A} = \mathbf{R}^\top\mathbf{R} = \mathbf{I}. \tag{16}$$

Note that an inversion of the diagonal matrix $\boldsymbol{\Lambda}$ requires the removal of zero eigenvalues and corresponding eigenvectors first.

[10] Not all Hilbert spaces $\mathcal{H}$ contain $\mathbf{1}_{\mathcal{H}}$. Technically we must optimize over $\mathcal{H} \cup \{\mathbf{1}_{\mathcal{H}}\}$. This can be achieved by allowing solutions of the form $\phi_i(\cdot) = \sum_{j=1}^m A_{ji}\kappa(\cdot, \boldsymbol{z}_j) - c_i$.

Algorithm 1 Regularized sparse kernel slow feature analysis (RSK-SFA)

Input: $\mathbf{K} \in \mathbb{R}^{m\times n}, \bar{\mathbf{K}} \in \mathbb{R}^{m\times m}, p \in \mathbb{N}$,
$\lambda \in \mathbb{R}^+, \mathbf{D} \in \mathbb{R}^{n\times n-1}$

$\hat{\mathbf{K}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)$ (14)

$\hat{\bar{\mathbf{K}}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\bar{\mathbf{K}}(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)$ (15)

$\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top = \mathrm{eig}(\frac{1}{n}\hat{\mathbf{K}}\hat{\mathbf{K}}^\top)$

$(\mathbf{U}_r, \boldsymbol{\Lambda}_r) = \mathrm{remove\_zero\_eigenvalues}(\mathbf{U}, \boldsymbol{\Lambda})$

$\mathbf{B} = \frac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}$ (17)

$\mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^\top = \mathrm{eig}(\boldsymbol{\Lambda}_r^{-1/2}\mathbf{U}_r^\top\mathbf{B}\mathbf{U}_r\boldsymbol{\Lambda}_r^{-1/2})$ (17)

$(\mathbf{R}_p, \boldsymbol{\Sigma}_p) = \mathrm{keep\_lowest\_p\_eigenvalues}(\mathbf{R}, \boldsymbol{\Sigma}, p)$

$\hat{\mathbf{A}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{U}_r\boldsymbol{\Lambda}_r^{-1/2}\mathbf{R}_p$ (16) + (19)

$\hat{\boldsymbol{c}} = \frac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n$ (19)

Output: $\hat{\mathbf{A}}, \hat{\boldsymbol{c}}$

Minimization of the objective Application of (16) allows us to solve (13) with a second eigenvalue decomposition:

$$\min_{\mathbf{R}} \mathbf{tr}\left(\mathbf{R}^\top\left\{\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{B}\mathbf{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\right\}\mathbf{R}\right)$$
$$\text{s.t. } \mathbf{R}^\top\mathbf{R} = \mathbf{I} \quad \text{with } \mathbf{B} := \frac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}. \tag{17}$$

Note that $\mathbf{R}$ is composed of the eigenvectors to the p *smallest* eigenvalues of the above expression.

Solution After the above calculations the i'th RSK-SFA solution is

$$\phi_i(\boldsymbol{x}) = \sum_{j=1}^{m} A_{ji}\left\langle\hat{g}_j, \kappa(\cdot, \boldsymbol{x})\right\rangle_{\mathcal{H}}. \tag{18}$$

Grouping the kernel functions of all support vectors together in a column vector, i.e. $\boldsymbol{k}(\boldsymbol{x}) = [\kappa(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{z}_m, \boldsymbol{x})]^\top$, the combined solution $\boldsymbol{\phi}(\cdot) \in \mathcal{H}^p$ can be expressed more compactly:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\mathbf{A}}^\top\boldsymbol{k}(x) - \hat{\boldsymbol{c}}$$
$$\text{with } \hat{\mathbf{A}} := \left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{A} \text{ and } \hat{\boldsymbol{c}} := \frac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n. \tag{19}$$

The computational complexity is $\mathcal{O}(m^2 n)$. For illustrative purposes, Algorithm 1 exhibits a memory complexity of $\mathcal{O}(mn)$. An online calculation of $\hat{\mathbf{K}}\hat{\mathbf{K}}^\top$ and $\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top$ reduces this to $\mathcal{O}(m^2)$.

4 Sparse subset selection

Representer theorems guarantee that the target function $\phi^* \in \mathcal{H}$ for training set $\{\boldsymbol{x}_t\}_{t=1}^n$ can be found within $\mathrm{span}(\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^n)$. For sparse K-SFA, however, no such guarantee exists.

The quality of such a sparse approximation depends exclusively on the set of support vectors $\{\boldsymbol{z}_i\}_{i=1}^m$.

Without restriction on ϕ^*, it is straight forward to select a subset of the training data, indicated by an *index vector*[11] $\boldsymbol{i} \in \mathbb{N}^m$ with $\{\boldsymbol{z}_j\}_{j=1}^m := \{\boldsymbol{x}_{\boldsymbol{i}_j}\}_{j=1}^m$, that minimizes the *approximation error*

$$\epsilon_t^{\boldsymbol{i}} := \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \left\| \kappa(\cdot, \boldsymbol{x}_t) - \sum_{j=1}^m \alpha_j \kappa(\cdot, \boldsymbol{x}_{\boldsymbol{i}_j}) \right\|_{\mathcal{H}}^2 = K_{tt} - \boldsymbol{K}_{t\boldsymbol{i}} (\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1} \boldsymbol{K}_{\boldsymbol{i}t}$$

for all training samples $\boldsymbol{x}_t$, where $K_{tj} = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_j)$ is the full kernel matrix. Finding an optimal subset is an NP hard combinatorial problem, but there exist several greedy approximations to it.

Online maximization of the affine hull A widely used algorithm (Csató and Opper 2002), which we will call *online maximization of the affine hull* (online MAH) in the absence of a generally accepted name, iterates through the data in an online fashion. At time t, sample $\boldsymbol{x}_t$ is added to the selected subset if $\epsilon_t^{\boldsymbol{i}}$ is larger than some given threshold η. Exploitation of the *matrix inversion lemma* (MIL) allows an online algorithm with computational complexity $\mathcal{O}(m^2 n)$ and memory complexity $\mathcal{O}(m^2)$. The downside of this approach is the unpredictable dependence of the final subset size m on hyper-parameter η. Changing the subset size requires therefore a complete re-computation with larger η. The resulting subset size is not predictable, although monotonically dependent on η.

Matching pursuit for sparse kernel PCA This handicap is addressed by *matching pursuit* methods (Mallat and Zhang 1993). Applied on kernels, some criterion selects the best fitting sample, followed by an orthogonalization of all remaining candidate support functions in Hilbert space $\mathcal{H}$. A resulting sequence of m selected samples therefore contains all sequences up to length m as well. The batch algorithm of Smola and Schölkopf (2000) chooses the sample $\boldsymbol{x}_j$ that minimizes[12] $\mathbb{E}_t[\epsilon_t^{\boldsymbol{i}\cup j}]$. It was shown later that this algorithm performs sparse PCA in $\mathcal{H}$ (Hussain and Shawe-Taylor 2008). The algorithm, which we will call in the following *matching pursuit for sparse kernel PCA* (MP KPCA), has a computational complexity of $\mathcal{O}(n^2 m)$ and a memory complexity of $\mathcal{O}(n^2)$. In practice it is therefore not applicable to large data sets.

4.1 Matching pursuit for online MAH

The ability to change the size of the selected subset without re-computation is a powerful property of MP KPCA. Run-time and memory consumption, however, make this algorithm infeasible for most applications. To extend the desired property to the fast online algorithm, we derive a novel *matching pursuit for online MAH* algorithm (MP MAH). Online MAH selects samples with approximation errors that exceed the threshold η and therefore forces the supremum norm L_∞ of all samples below η. This is analogous to a successive selection of the *worst* approximated sample, until the approximation error of all samples drops below η. The matching pursuit approach therefore minimizes the supremum norm L_∞ of the

[11] Let "**:**" denote the index vector of all available indices. See Algorithm 2.

[12] $\epsilon_t^{\boldsymbol{i}}$ is non-negative and MP KPCA thus minimizes the L_1 norm of approximation errors.

Algorithm 2 Matching pursuit maximization of the affine hull (MP MAH)

Input: $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X}, \kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, m \in \mathbb{N}$

$\mathbf{K} = \emptyset; \quad \mathbf{K}_1^{-1} = \emptyset;$

$\forall t \in \{1, \ldots, n\} : \epsilon_t^1 = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t)$ (see (21))

$\boldsymbol{i}_1 = \text{argmax}_t \{\epsilon_t^1\}_{t=1}^n$ (see (20))

for $j \in \{1, \ldots, m-1\}$ **do**

$\boldsymbol{\alpha}^j = [\mathbf{K}_{(j,:)}\mathbf{K}_j^{-1}, -1]^\top$ (MIL)

$\mathbf{K}_{j+1}^{-1} = \begin{bmatrix} \mathbf{K}_j^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{\boldsymbol{\alpha}^j \boldsymbol{\alpha}^{j\top}}{\epsilon_{\boldsymbol{i}_j}^j}$ (MIL)

for $t \in \{1, \ldots, n\}$ **do**

$\mathbf{K}_{(t,j)} = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_{\boldsymbol{i}_j})$

$\epsilon_t^{j+1} = \epsilon_t^j - \frac{1}{\epsilon_{\boldsymbol{i}_j}^j}(\mathbf{K}_{(t,:)}\boldsymbol{\alpha}^j)^2$ (see (21))

end for

$\boldsymbol{i}_{j+1} = \text{argmax}_t \{\epsilon_t^{j+1}\}_{t=1}^n$ (see (20))

end for

Output: $\{\boldsymbol{i}_1, \ldots, \boldsymbol{i}_m\}$

approximation error.[13] At iteration j, given the current subset $\boldsymbol{i} \in \mathbb{R}^j$,

$$\boldsymbol{i}_{j+1} := \text{argmin}_t \left\| \left[\epsilon_1^{\boldsymbol{i}\cup t}, \ldots, \epsilon_n^{\boldsymbol{i}\cup t}\right] \right\|_\infty \approx \text{argmax}_t\, \epsilon_t^{\boldsymbol{i}}. \tag{20}$$

Straight forward re-computation of the approximation error in every iteration is expensive. Using the matrix inversion lemma (MIL), this computation can be performed iteratively:

$$\epsilon_t^{\boldsymbol{i}\cup j} = \epsilon_t^{\boldsymbol{i}} - \frac{1}{\epsilon_j^{\boldsymbol{i}}}\left(K_{tj} - \boldsymbol{K}_{t\boldsymbol{i}}(\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1}\boldsymbol{K}_{\boldsymbol{i}j}\right)^2. \tag{21}$$

Algorithm 2 iterates between sample selection (20) and error update (21). The complexity is $\mathcal{O}(m^2 n)$ in time and $\mathcal{O}(mn)$ in memory (to avoid re-computations of $\mathbf{K}_{(:,\boldsymbol{i})}$).

5 Empirical validation

In this section we will evaluate the hypothesis that *RSK-SFA generates a feature space Φ which resembles a Fourier basis in the space of latent variables Θ for a given time series* at the example of a two-vowel classification task.

The true Θ is not known and we use the performance of different linear classification algorithms to measure how well RSK-SFA features encode Θ. For comparison, all algorithms are tested on feature spaces generated by RSK-SFA and *sparse kernel PCA* (Schölkopf et al. 1997). Results in Sect. 5.4 show that RSK-SFA features indeed encode Θ, whereas kernel PCA does not aim for such an encoding. We also demonstrate the importance of sparse subset selection to an efficient encoding. Last but not least, a classification accuracy of more than 97 % in a task that is not linearly solvable demonstrates the excellent performance of our approach.

[13] An exact minimization of the L_∞ norm is as expensive as the MP KPCA algorithm. However, since $\epsilon_t^{\boldsymbol{i}\cup t} = 0$, selecting the worst approximated sample $\boldsymbol{x}_t$ effectively minimizes the supremum norm L_∞.

5.1 Benchmark data sets

The "north Texas vowel database"[14] contains uncompressed audio files with English words of the form H. . . D, spoken multiple times by multiple persons (Assmann et al. 2008). The natural task is to predict the central vowels of unseen instances of a trained word. To cover both cases of small and large training sets, we selected two data sets: (i) A small set with four training and four test instances for each of the words "heed" and "head", spoken by the same person. (ii) A large data set containing all instances of "heed" and "head" for all 18 adult subjects. To apply cross-validation, we performed 20 random splits (folds) into 12 training and 6 test subjects, leading to 245 ± 10 training instances and 116 ± 10 test instances.

The spoken words are provided as mono audio streams of varying length at 48kHz, i.e. as a series of amplitude readings $\{a_1, a_2, \ldots\}$. SFA requires the space of latent variables Θ to be *embedded* in the space of observations $\mathcal{X}$, which is not the case for one-dimensional data. The problem resides in the ambiguity of the observed amplitude readings a_t, i.e. mappings $\phi_i : \mathcal{X} \to \mathbb{R}^p$ can not distinguish between two latent states which both generate the same observed amplitude a_t. However, Takens Theorem (Takens 1981; Huke 2006) guarantees an embedding of Θ as a manifold in the space $\mathcal{X} \subset \mathbb{R}^l$ of sufficiently many *time-delayed* observations. Based on the observed pattern of l time-delayed amplitude readings, nonlinear SFA and PCA can utilize the resulting one-to-one mapping of latent variables in Θ onto a manifold of observations in $\mathcal{X}$ to extract basis functions $\phi_i(\cdot)$ that encode Θ. We therefore defined our samples $\boldsymbol{x}_t := [a_{\delta t}, a_{\delta t+\epsilon}, a_{\delta t+2\epsilon}, \ldots, a_{\delta t+(l-1)\epsilon}]^\top$, which is also called *sliding window*.[15] We evaluated the parameters δ, ϵ and l empirically[16] and chose $\delta = 50$, $\epsilon = 5$ and $l = 500$. All algorithms were trained and tested with the joint[9] time series $\{\boldsymbol{x}_t\}$ of all instances of all subjects in the respective fold. Note, however, that all samples of each word-instance were exclusively used for either training or testing. Additionally, in the large data set all instances of one subject were exclusively used, too. This ensures that we really test for generalization to unseen instances of the trained words as well as to previously unseen subjects. The above procedure provided us with 3719(4108) trainings (test) samples $\boldsymbol{x}_t \in [-1, 1]^{500}$ for the small and 97060 ± 4615 (46142 ± 4614) trainings (test) samples for the large data set.

5.2 RSK-SFA performance

We start our analysis with the RSK-SFA solution for different Gaussian kernels, i.e. $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$ with varying σ. To ensure that meaningful information is extracted, we measure the *test slowness*, i.e. the slowness of the learned feature mappings applied on a previously unseen test sequence drawn from the same distribution. Small σ, however, grossly underestimate a feature's output on unseen test samples, as distances are strongly amplified. This changes the feature's slowness. For comparison we normalized all outputs to unit variance *on the test set* before measuring the test slowness.

[14] http://www.utdallas.edu/~assmann/KIDVOW1/North_Texas_vowel_database.html.

[15] This violates the *i.i.d. assumption* of most classification algorithms, as two successive samples are no longer independent. The classification results were excellent, however, indicating that the violation did not influence the algorithms' performance too strongly.

[16] Although the choice of embedding parameters change the resulting slowness in a nontrivial fashion, we want to point out that this change appears to be smooth and the presented shapes similar over a wide range of embedding parameters.

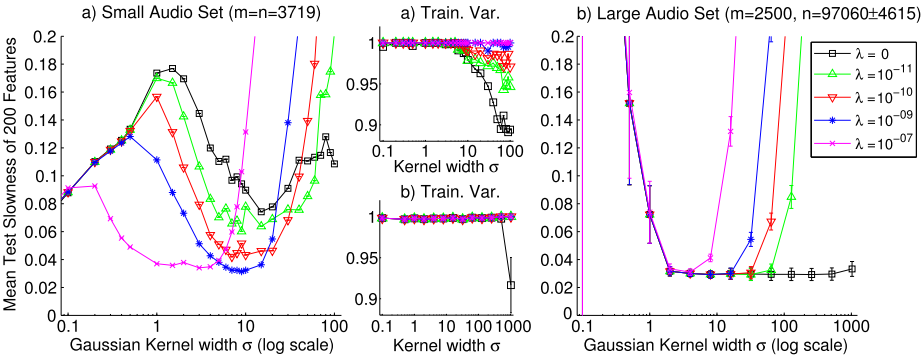

Fig. 1 Mean test slowness of 200 RSK-SFA features over varying kernel parameter σ for different regularization parameters λ: (**a**) *Small audio data set* with all $m = n = 3719$ training samples as support vectors, (**b**) *large audio data set* with $m = 2500$ support vectors selected out of $n = 97060 \pm 4615$ training samples by MP MAH. The *small plots* show the variance of the respective training output. Significant deviation from one violates the unit variance constraint and thus demonstrates numerical instability. The *legend* applies to all plots

Figure 1 shows the mean test slowness of 200 RSK-SFA features[17] on both sets for multiple kernel parameter σ and regularization parameter λ. The small data set (a) uses the complete training set as support vectors, whereas for the large data set (b) a full kernel approach is not feasible. Instead we selected a subset of size 2500 with the MP MAH algorithm (based on kernel parameter $\sigma = 5$) before training.

In the absence of significant sparseness (Fig. 1a), unregularized kernel SFA ($\lambda = 0$, equivalent to K-SFA, see (11) and (12)) shows both over-fitting and numerical instability. Over-fitting can be seen at small σ, where the features fulfill the unit variance constraint (small plot), but do not reach the minimal test slowness (main plot). The bad performance for larger σ, on the other hand, must be blamed on numerical instability, as indicated by a significantly violated unit variance constraint. Both can be counteracted by proper regularization. Although optimal regularization parameters λ are quite small and can reach computational precision, there is a wide range of kernel parameters σ for which the same minimal test slowness is reachable. E.g. in Fig. 1a, a fitting λ can be found between $\sigma = 0.5$ and $\sigma = 20$.

The more common case, depicted in Fig. 1b, is a large training set from which a small subset is selected by MP MAH. Here no regularization is necessary and unregularized sparse kernel SFA ($\lambda = 0$) learns mappings of minimal slowness in the range from $\sigma = 1$ to far beyond $\sigma = 500$. Notice that this is rendering a time consuming search for an optimal parameter σ obsolete.

5.3 Sparse subset selection

To evaluate the behavior of RSK-SFA for sparse subsets of different size, Fig. 2 plots the test slowness of the small data set for all discussed sparse subset selection algorithms in Sect. 4. As a baseline, we plotted mean and standard deviation of a random selection scheme. One

[17]Comparison to linear SFA features is omitted due to scale, e.g. test slowness was slightly above 0.5 for both data sets. RSK-SFA can therefore outperform linear SFA by more than a factor of 10, a magnitude we observed in other experiments as well.

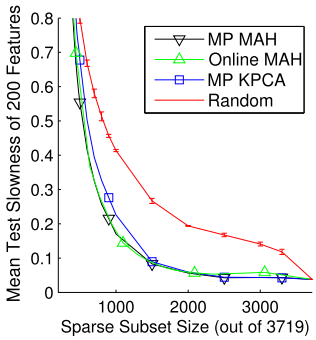

Fig. 2 Mean test slowness of 200 RSK-SFA features of the *small audio data set* ($\lambda = 10^{-7}, \sigma = 2$) over sparse subset size, selected by different algorithms. For *random selection* the mean of 10 trials is plotted with standard deviation error bars

can observe that all algorithms surpass the random selection significantly but do not differ much w.r.t. each other. As expected, the MP MAH and Online MAH algorithms perform virtually identical. The novel MP MAH, however, allows unproblematic and fast fine tuning of the selected subset's size.

5.4 Classification performance

To demonstrate that Φ encodes Θ, we compared the accuracy of linear classification algorithms (see Sect. 2) in RSK-SFA and SK-PCA feature spaces of different sizes. Sparse kernel PCA optimizes the PCA objective (maximal variance) on the same function class as RSK-SFA, i.e. with the same kernel and support vectors. Because some iterative algorithms are sensitive to scaling, we scaled the SK-PCA features to unit variance *on the test set*, to make them fully comparable with RSK-SFA features.

In the evaluated two-class problem ("head" vs. "heed"), class labels are only available for whole words. Individual samples $\boldsymbol{x}_t$, however, do not necessarily belong to a vowel, but might contain one of the consonants in the beginning or end of the word. Many samples will therefore be ambiguously labeled, which the classifier must interpret as noise. When all samples of a word are classified, the final judgment about the spoken vowel is determined by the sum over all *discrimination function* outputs $\sum_{t=1}^{n} f(\boldsymbol{x}_t)$, rather than over the individual class predictions. This is supposed to take the classifiers certainty into account and yields much better results. It is also equivalent to the empirical *temporal mean* of the discriminant function output over the whole word.[18]

This shift from training each sample to evaluating only the mean over all samples might look like a completely different problem at first. One could rightfully demand to train the classifiers on the mean feature vector $\bar{\boldsymbol{\phi}} := \mathbb{E}_t[\boldsymbol{\phi}(\boldsymbol{x}_t)]$ of each word instead of all individual samples thereof. Nonetheless, the authors chose the presented methodology to compare SFA and PCA feature spaces because

1. a training set of 245 ± 10 words must over-fit in high dimensional feature spaces.
2. classification of individual samples $\boldsymbol{x}_t$ must always result in very poor accuracy as most of them do not encode the vowel and will thus yield only noise.
3. $\bar{\boldsymbol{\phi}}$ lives in the *same feature space* as the individual samples and should be affected by the *same insufficiencies* of that feature space Φ.

[18] Note that this is not the same as the discriminant output of the temporal mean over the whole word, i.e. $\mathbb{E}_t[f(\boldsymbol{x}_t)] = \sum_{i=1}^{p} w_i \mathbb{E}_t[\phi_i(\boldsymbol{x}_t)] \neq f(\mathbb{E}_t[\boldsymbol{x}_t])$, as $\phi_i(\cdot)$ are non-linear.

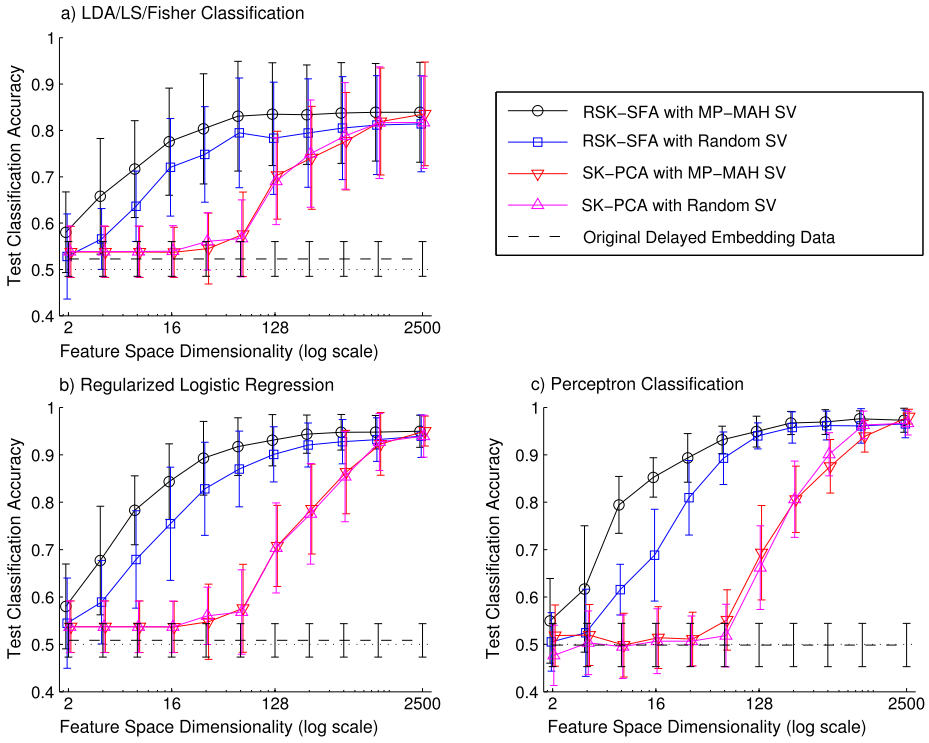

Fig. 3 Accuracy of vowel detection on unseen test set against feature space size for all discussed linear classification algorithms (**a**–**c**). Each algorithm was tested on feature spaces generated by RSK-SFA ($\lambda = 0, \sigma = 5$) and SK-PCA with 2500 support vectors selected at random or with MP MAH. As a comparison the accuracy on the original 500 dimensional space of delayed audio amplitudes is given as well (*dashed line*). All accuracies are plotted with mean and standard deviation for 20 random splits into training and test sets

Figure 3 shows mean and standard deviation of the *test accuracy*[19] over 20 random splits of the large data set into training and test set (see Sect. 5.1) against the number of RSK-SFA or SK-PCA features. Both feature spaces are generated using the same support vectors selected at random or by the novel MP MAH algorithm. To demonstrate that this is not a trivial problem, the algorithms' performance on the original 500 dimensional delayed amplitude space is given as a dashed line.

It is apparent that in all cases the RSK-SFA feature space is superior to SK-PCA for all but 2500 features. At this point both feature spaces are identical up to rotation and any linear algorithm must therefore perform identical. Moreover, RSK-SFA features based on support vectors selected by MP MAH clearly outperform those based on randomly selected SV in all cases. In a comparison between algorithms, perceptron classification (Fig. 3c) yields the highest accuracy with the least features, i.e. more that 97 % with 256 RSK-SFA features. More features do not significantly raise the performance. This is particularly interesting because the training does not require a matrix inverse and is therefore the fastest of the evaluated algorithms in large feature spaces. It is also worth mentioning that 8 (32) RSK-SFA features reach an accuracy of 80 % (90 %), whereas the first 64 SK-PCA features apparently

[19] *Test accuracy* is the fraction of correct classifications for an previously unseen test set.

do not encode any information about Θ. Regularized logistic regression[20] (Fig. 3b) reaches comparable accuracy (around 95 % with 256 or more features) but exhibits higher variance between folds. The poor performance of LS/LDA/FDA classification (Fig. 3a) must stem from the violated assumption of equal class covariances (see Sect. 2). Note, however, that all above comparisons between feature spaces and SV selection schemes still hold.

These results demonstrate that, in difference to SK-PCA, RSK-SFA feature spaces Φ aim to encode Θ. The high accuracy of over 97 % establishes that either (i) the problem is almost linearly separable in Θ or (ii) Φ resembles a functional basis in Θ, presumably[21] a Fourier basis. At this point in time it is not possible to distinguish between (i) and (ii). However, the large number of required features could be interpreted as evidence for (ii).

We therefore conclude that RSK-SFA features encode the space of latent variables Θ of a given time series, presumably by resembling a Fourier basis of Θ.

6 Discussion

This article investigates the hypothesis that sufficiently powerful *non-linear slow feature analysis features resemble a Fourier basis in the space of latent variables Θ*, which underlies complex real-world time series. To perform a powerful but easily operated non-linear SFA, we derived a kernelized SFA algorithm (RSK-SFA). The novel algorithm is capable of handling small data sets by *regularization* and large data sets through *sparsity*. To select a sparse subset for the latter, we developed a matching pursuit approach to a widely used algorithm (MP MAH). In combination with linear classification algorithms, particularly the perceptron algorithm, our results support the hypothesis' validity and demonstrate excellent performance.

6.1 Comparison to previous works

The major advancement of our approach over the kernel SFA algorithm of Bray and Martinez (2002) is the ability to obtain features that generalize well for *small data sets*. If one is forced to use a large proportion of the training set as support vectors, e.g. for small training sets of complex data, the solution can violate the unit variance constraint.

As suggested by Bray and Martinez (2002), our experiments show that for large data sets no explicit regularization is needed. The implicit regularization introduced by sparsity is sufficient to generate features that generalize well over a wide range of Gaussian kernels. However, our work documents the dependence of the algorithms' performance on the sparse subset for the first time. In this setting, the subset size m takes the role of regularization parameter λ. It is therefore imperative to control m with minimal computational overhead.

We compared two state-of-the-art algorithms that select sparse subsets in polynomial time. *Online MAH* is well suited to process large data sets, but selects unpredictably large subsets. A change of m therefore requires a full re-computation without the ability to target a specific size. *Matching pursuit for sparse kernel PCA* (MP KPCA), on the other hand, returns an ordered list of selected samples. Increasing the subsets size requires simply another

[20]The unregularized algorithm showed clear signs of over-fitting in large feature spaces, i.e. the accuracy dropped for large p. We repeated the training and evaluation procedure with slowly increasing regularization parameter η. The (presented) results stabilized at $\eta = 500$.

[21]The functional form in Θ can not be derived from the presented results. However, theoretical works by Wiskott (2003) and Franzius et al. (2007) suggest that Φ approximates a *Fourier basis* in Θ, as discussed in Sects. 1 and 3.

loop of the algorithm. The downside is a quadratic dependency on the training set size, both in time and memory. Both algorithms exhibited similar performance and significantly outperformed a random selection scheme. The subsets selected by the novel *matching pursuit to online MAH* (MP MAH) algorithm yielded virtually the same performance as those selected by Online MAH. There is no difference in computation time, but the memory complexity of MP MAH is linearly dependent on the training sets size. However, increasing m works just as with MP KPCA, which makes this algorithm the better choice if one can afford the memory. If not, Online MAH can be applied several times with slowly decreasing hyper-parameter η. Although a subset of suitable size will eventually be found, this approach will take much more time than MP MAH.

6.2 Hyper-parameter selection

For a first assessment of the feature space generated by RSK-SFA, selecting the support vectors randomly is sufficient. For optimal performance, however, one should select the sparse subset with MP MAH or Online MAH, depending on available resources and time (see last section). As shown in Fig. 1b, selecting the Gaussian kernel parameter σ is not an issue in face of sufficient sparsity. Empirically, setting σ such that more than half of all kernel outputs are above 0.5 has yielded sufficient performance in most cases.

Before raising the regularization parameter λ above 0, it suggests itself to check the unit variance constraint on the training data. Particularly if the kernel can not be adjusted, a violation can be compensated by slowly rising λ until the variance is sufficiently close to one. A numerically stable solution is no guarantee for optimal slowness, though. It is therefore always recommendable to track the test slowness on an independent test set. When in doubt of over-fitting, try to shrink the sparse subset size m, raise λ or increase kernel width σ, in this order.

6.3 Limitations

The methodology developed in this article will work in most standard machine learning scenarios involving time series. Although not unique to our method, there are certain limitations that are worth mentioning.

1. Our method of generating feature spaces Φ is based on the assumption of an underlying space of latent variables Θ. This implies that the *generative process* $\Theta \to \mathcal{X}$ is *stationary* over time. Preliminary experiments on EEG data (not shown), which are known to be non-stationary, yielded no generalizing features.
2. The generative mapping must be unique. Two elements of Θ which are reliably mapped onto the same element of $\mathcal{X}$ are not distinguishable in Φ.
3. As the exact nature of Θ is ambiguous, there can be arbitrary many "unwanted" latent variables, which raises three problems: (i) If the slowest variables are not relevant for the task, they will appear as noise. (ii) The size of a Fourier basis Φ of Θ grows exponential in the dimensionality of Θ. (iii) Generating a reliable feature space Φ requires training samples from all regions of Θ, which could require infeasible amounts of training samples if Θ is high dimensional.

6.4 Including label or importance information

Applied on real-world time series, RSK-SFA features resemble a Fourier basis in Θ. We want to discuss two possible modifications to generate feature spaces.

1. SFA exploits the temporal succession of samples and thus can only be applied on time series. It is imaginable, however, to use other available information to modify the approach without much change to the methodology. For example, if one is faced with labeled *iid* data, Algorithm 1 can be modified with a different *objective* to minimize the distance between *all* samples of *each class*:

$$\min_{\boldsymbol{\phi}\in\mathcal{H}^p} \sum_{k=1}^{p} \mathbb{E}_i\left[\mathbb{E}_j\left[\delta_{c_i c_j}\left(\phi_k(\boldsymbol{x}_i)-\phi_k(\boldsymbol{x}_j)\right)^2\right]\right], \tag{22}$$

   where $\delta_{c_i c_j}$ is the Kronecker delta, which is 1 if $c_i = c_j$ and 0 otherwise. The two nested expectations induce a quadratic dependency on the training set size n, but can be approximated by randomly drawing sample pairs from the same class. The resulting feature space Φ will not resemble a functional basis in Θ, but reflect a between-classes metric by mapping same class samples close to each other. Preliminary experiments (not shown) have generated promising feature spaces for linear classification. This approach has also been taken by Berkes (2005), who used SFA on 2 step "time series" of same class samples of hand written digits.
2. The individual *components* (or dimensions) of Θ are encoded according to their slowness. This can result in an inefficient encoding if the slowest changing components of Θ are not of any use to the task. It is therefore of interest if one can use additional information about the samples to restrict Φ to encode only a relevant *subspace* of Θ, at least in the first p features. If one can define an *importance measure* $p(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) > 0$ with high values for *transitions within the desired subspace*, the slowness objective in (6) can be modified to

$$\min s'(\phi_i) := \frac{1}{n-1}\sum_{t=1}^{n-1} \frac{(\phi_i(\boldsymbol{x}_{t+1})-\phi_i(\boldsymbol{x}_t))^2}{p(\boldsymbol{x}_{t+1},\boldsymbol{x}_t)}. \tag{23}$$

   As a result, the relative slowness of important transitions is reduced and the respective subspace of Θ will be encoded earlier. Increasing the importance will eventually encode the *subspace spanned by the important transitions only* within the first p features extracted by the modified algorithm.
3. If only arbitrary subsets of Θ are of consequence to the task, another modification can increase encoding by including additional information. If the *samples* can be labeled according to another *importance measure* $q(\boldsymbol{x}) \geq 0$, which marks the important ($q(\boldsymbol{x}) \gg 0$) and unimportant samples ($q(\boldsymbol{x}) \approx 0$), the *unit variance* and *decorrelation constraints* of (11) can be modified, i.e.

$$\mathbb{E}_t\left[\boldsymbol{\phi}(\boldsymbol{x}_t) q(\boldsymbol{x}_t) \boldsymbol{\phi}(\boldsymbol{x}_t)^\top\right] = \mathbf{I}, \tag{24}$$

   which would enforce unit variance *on the important samples only*. Slowness still applies to all samples equally and the resulting feature space Φ would map samples of low importance onto[22] each other. Given a powerful enough function class, the modified algorithm should construct a Fourier basis in Θ on the subset of important samples only.[23]

[22] Samples of low (or no) importance would be ideally mapped onto the feature output of the *last* important sample seen in the trajectory. However, due to the structure of Hilbert space $\mathcal{H}$ the modified algorithm can produce mappings that generalize well in Θ.

[23] The feature output of the unimportant part of Θ should be either constant or change as smooth as possible between *adjacent* important samples.

For example, if we would know which samples $\boldsymbol{x}_t$ contain a vowel (but not which vowel it is), the resulting feature space Φ should exclusively encode the subset of Θ that represents the vowels. When this subset is relatively small, the resulting feature space Φ will approximate the same task with much less basis functions.

6.5 Relationship to deep belief networks

The sound performance of the perceptron algorithm in combination with non-linear SFA suggests an interesting connection to *deep belief networks* (Hinton and Salakhutdinov 2006). It is long known that *multilayer perceptrons* (MLP) are so prone to initialization, that it is virtually impossible to learn a *deep architecture* (i.e. many layers) with any random initialization thereof (Haykin 1999). Hinton and Osindero (2006) came up with the idea[24] to train the first layer as an *auto-encoder*, i.e. to predict its own inputs via a hidden layer of perceptrons. The trained auto-encoder is then fixed and the hidden layer used as the input of the next layer, which is trained in the same way. The resulting deep architecture is a generative model of the data and has proven to be a suitable initialization for many problems.

In line with this article's hypothesis, however, we assume that the target function is defined over a space of latent variables Θ. Layer-wise training of an MLP version of SFA might therefore yield even better initialization, as it already spans a suitable *function space* rather than simply encoding the data (see Franzius et al. 2007 for an example of a hierarchical SFA). However, repeated application of non-linear SFA, even with the limited function class of one perceptron per feature, bears sooner or later the problem of over-fitting. Preliminary experiments on repeated application of quadratic SFA lost any generalization ability within a few layers. Without an automatic way to regularize the solution in each layer, this approach will thus not yield a reliable initialization for classical MLP backpropagation (Haykin 1999). Future research must show whether a properly regularized MLP-SFA algorithm can be used to initialize an MLP and whether or not this approach can compete with deep belief networks based on auto-encoders.

Acknowledgements This work has been supported by the Integrated Graduate Program on Human-Centric Communication at Technische Universität Berlin, the German Research Foundation (DFG SPP 1527 *autonomous learning*), the German Federal Ministry of Education and Research (grant 01GQ0850) and EPSRC grant #EP/H017402/1 (CARDyAL). We want to thank Matthias Franzius, who gave us a sound introduction into non-linear SFA, and Roland Vollgraf for his contribution to an earlier version of RSK-SFA.

References

Assmann, P. F., Nearey, T. M., & Bharadwaj, S. (2008). Analysis and classification of a vowel database. *Canadian Acoustics*, *36*(3), 148–149.

Becker, S., & Hinton, G. E. (1992). A self-organizing neural network that discovers surfaces in random dot stereograms. *Nature*, *355*(6356), 161–163.

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

Berkes, P. (2005). Pattern recognition with slow feature analysis. *Cognitive Sciences EPrint Archive (CogPrint)* (4104).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin: Springer. ISBN 978-0-387-31073-2.

Böhmer, W., Grünewälder, S., Nickisch, H., & Obermayer, K. (2011). Regularized sparse kernel slow feature analysis. In *ECML/PKDD 2011* (vol. I, pp. 235–248).

[24] Hinton and Osindero (2006) showed this for *restricted Boltzmann machines*, but the principle holds for MLP as well.

Bray, A., & Martinez, D. (2002). Kernel-based extraction of slow features: complex cells learn disparity and translation invariance from natural images. *Neural Information Processing Systems*, *15*, 253–260.
Csató, L., & Opper, M. (2002). Sparse on-line gaussian processes. *Neural Computation*, *14*(3), 641–668.
Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.
Einhäuser, W., Hipp, J., Eggert, J., Körner, E., & König, P. (2005). Learning viewpoint invariant object representations using temporal coherence principle. *Biological Cybernetics*, *93*(1), 79–90.
Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*, 179–188.
Fletcher, R. (1987). *Practical methods of optimization* (2nd ed.). New York: Wiley.
Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, *3*(2), 194–200.
Franzius, M., Sprekeler, H., & Wiskott, L. (2007). Slowness and sparseness leads to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, *3*(8), e166.
Fukumizu, K., Bach, F. R., & Gretton, A. (2007). Statistical consistency of kernel canonical correlation analysis. *Journal of Machine Learning Research*, *8*, 361–383.
Haykin, S. (1999). *Neural networks: a comprehensive foundation* (2nd ed.). New York: Prentice Hall.
Hinton, G. E., & Osindero, S. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.
Huke, J. P. (2006). *Embedding nonlinear dynamical systems: a guide to Takens' theorem*. Technical report, University of Manchester.
Hussain, Z., & Shawe-Taylor, J. (2008). Theory of matching pursuit. In *Advances in neural information processing systems* (vol. 21, pp. 721–728).
Mallat, S., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, *41*, 3397–3415.
Meyn, S. P., & Tweedie, R. L. (1993). *Markov chains and stochastic stability*. London: Springer.
Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan.
Rubin, D. B. (1983). Iteratively reweighted least squares. *Encyclopedia of Statistical Sciences*, *4*, 272–275.
Schölkopf, B., Smola, A., & Müller, K. R. (1997). Kernel principal component analysis. In *Artificial neural networks ICANN*.
Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.
Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.
Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings to the 17th international conference machine learning* (pp. 911–918).
Stone, J. V. (2001). Blind source separation using temporal predictability. *Neural Computation*, *13*(7), 1559–1574.
Takens, F. (1981). Detecting strange attractors in turbulence. In *Dynamical systems and turbulence* (pp. 366–381).
Wahba, G. (1990). *Spline models for observational data*. Philadelphia: Society for Industrial and Applied Mathematics.
Wiskott, L. (2003). Slow feature analysis: a theoretical analysis of optimal free responses. *Neural Computation*, *15*(9), 2147–2177.
Wiskott, L., & Sejnowski, T. (2002). Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, *14*(4), 715–770.
Wyss, R., König, P., & Verschure, P. F. M. J. (2006). A model of the ventral visual system based on temporal stability and local memory. *PLoS Biology*, *4*(5), e120.