Regularized Sparse Kernel Slow Feature Analysis

Wendelin Böhmer[1], Steffen Grünewälder[2],
Hannes Nickisch[3], and Klaus Obermayer[1]

[1] Neural Processing Group, Technische Universität Berlin, Germany
{wendelin,oby}@cs.tu-berlin.de
[2] Centre for Computational Statistics and Machine Learning,
University College London, United Kingdom
steffen@cs.ucl.ac.uk
[3] Philips Research Laboratories, Hamburg, Germany
hannes.nickisch@philips.com

Abstract. This paper develops a kernelized *slow feature analysis* (SFA) algorithm. SFA is an unsupervised learning method to extract features which encode latent variables from time series. Generative relationships are usually complex, and current algorithms are either not powerful enough or tend to over-fit. We make use of the *kernel trick* in combination with *sparsification* to provide 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 but complex 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. Versatility and performance of our method are demonstrated on audio and video data sets.

1 Introduction

Slow feature analysis (SFA [23]) is an unsupervised method to extract features which encode latent variables of time series. SFA aims for temporally coherent features out of high dimensional and/or delayed sensor measurements. Given enough training samples, the learned features will become sensitive to slowly changing latent variables [9, 22]. Although there have been numerous studies highlighting its resemblance to biological sensor processing [3, 9, 23], 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.

This paper provides such an extension in the form of a kernel SFA algorithm. Such an approach has previously been made by Bray and Martinez [4] and is reported to work well with a large image data set. Small and complex sets, however, lead to numerical instabilities in any kernel SFA algorithm. Our goal is to provide an algorithm that can be applied to both of the above cases.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 235–248, 2011.

Although formulated as a linear algorithm, SFA was originally intended to be applied on the space of polynomials (e.g. quadratic [23] or cubic [3]). 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 [23]. Although proven to work in complex tasks [9], this approach involves a multitude of hyper-parameters and no easy way to counteract inevitable over-fitting. 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 $y : \mathcal{X} \to \mathrm{I\!R}$ are elements of a *reproducing kernel Hilbert space* $\mathcal{H}$. The *representer theorem* [21] ensures the optimal solution for a given training set exists within the span of *kernel functions*, parametrized by training samples. Depending on the kernel, the considered Hilbert space can be equivalent to the space of continuous functions [17] and a mapping $y \in \mathcal{H}$ is thus very powerful.

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 Sections 2 and 4). This tendency has been analytically shown for the related *kernel canonical correlation analysis* [10]. We introduce a regularization term to 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 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* [5] that must recompute the whole solution to change the subset's size. (2) A costly *matching pursuit approach* to sparse kernel PCA [18] 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 [4] have previously introduced a *kernel SFA* algorithm that incorporates a simplistic sparsity scheme. Instead of the well-established framework of Wiskott and Sejnowski [23], they utilize the cost function of Stone [19] 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 will 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.

In the following section, we first introduce the general SFA optimization problem and derive a *regularized sparse kernel SFA algorithm*. In Section 3 the sparse subset selection is introduced and a novel matching pursuit algorithm derived. Section 4 evaluates the algorithms on multiple real-world data sets, followed by a discussion of the results in Section 5.

2 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 $y_i : \mathcal{X} \to \mathrm{I\!R}$, $i \in \{1, \ldots, p\}$, such that the values $y_i(\boldsymbol{x}_t)$ change slowly over time [23]. A mapping y_i's change over time is measured by the *discrete temporal derivative* $\dot{y}_i(\boldsymbol{x}_t) := y_i(\boldsymbol{x}_t) - y_i(\boldsymbol{x}_{t-1})$. The SFA objective (called *slowness*, Equation 1) is to minimize the squared mean of this derivative, where $\mathbb{E}_t[\cdot]$ is the sample mean[1] over all available indices t:

$$\min \quad s(y_i) := \mathbb{E}_t[\dot{y}_i^2(\boldsymbol{x}_t)] \quad \text{(Slowness)} \tag{1}$$

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* (Equations 2 and 3). *Decorrelation* ensures every mapping to extract unique information (Equation 4). The last degree of freedom is eliminated by demanding *order* (Equation 5), leading to the following constraints:

$$\begin{aligned}
\mathbb{E}_t[y_i(\boldsymbol{x}_t)] &= 0 && \text{(Zero Mean)} && (2)\\
\mathbb{E}_t[y_i^2(\boldsymbol{x}_t)] &= 1 && \text{(Unit Variance)} && (3)\\
\mathbb{E}_t[y_i(\boldsymbol{x}_t) y_j(\boldsymbol{x}_t)] &= 0, \forall j \neq i && \text{(Decorrelation)} && (4)\\
\forall j > i : s(y_i) &\leq s(y_j) && \text{(Order)} && (5)
\end{aligned}$$

The principle of slowness, although not the above definition, has been used very early in the context of neural networks [2, 8]. Recent variations of SFA differ either in the objective [4] or the constraints [7, 24]. 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 [9, 22]. In reality those conditions are never met and one requires a function class that can be adjusted to the data set at hand.

2.1 Kernel SFA

Let the considered mappings $y_i : \mathcal{X} \to \mathrm{I\!R}$ be elements of a *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}$, with corresponding positive semi-definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathrm{I\!R}$. The *reproducing property* of those kernels allows $\forall y \in \mathcal{H} : y(\boldsymbol{x}) = \langle y, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$, in particular $\langle \kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{x}')\rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$. The *representer theorem* ensures the solution to be in the span of *support functions*, parametrized by the training data [21], i.e. $y = \sum_{t=1}^n a_t \kappa(\cdot, \boldsymbol{x}_t)$. Together those two relationships set the basis for the *kernel trick* (for an introduction see e.g. Shawe-Taylor and Cristianini [17]).

The zero mean constraint can be achieved by centring all involved *support functions* $\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^n$ in $\mathcal{H}$ (for details see Section 2.2). Afterwards, the combined kernel SFA (K-SFA) optimization problem for all p mappings $\boldsymbol{y}(\cdot) \in \mathcal{H}^p$ is

[1] 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 [15].

$$\min_{\boldsymbol{y}\in\mathcal{H}^p} \sum_{i=1}^{p} \mathbb{E}_t\left[\dot{y}_i^2(\boldsymbol{x}_t)\right], \text{ s.t. } \mathbb{E}_t\Big[\boldsymbol{y}(\boldsymbol{x}_t)\boldsymbol{y}(\boldsymbol{x}_t)^\top\Big] = \mathbf{I}. \tag{6}$$

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

$$\begin{aligned} \min_{\mathbf{A}\in\mathbb{R}^{n\times p}} \tfrac{1}{n-1}\mathbf{tr}\left(\mathbf{A}^\top\mathbf{K}\mathbf{D}\mathbf{D}^\top\mathbf{K}^\top\mathbf{A}\right) \\ \text{s.t. } \tfrac{1}{n}\mathbf{A}^\top\mathbf{K}\mathbf{K}^\top\mathbf{A} = \mathbf{I}, \end{aligned} \tag{7}$$

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* with all zero entries except $\forall t \in \{1,\dots,n-1\} : D_{t,t} = -1$ 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 $\forall t \in \{1,\dots,n\} : \kappa(\cdot, \boldsymbol{x}_t) \in \mathbf{span}(\{\kappa(\cdot, \boldsymbol{z}_i)\}_{i=1}^m)$, 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 the infinite dimensional space of continuous functions [17]. 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. [10] 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 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 $\|y_i\|_{\mathcal{H}}^2$ by a *regularization parameter* λ. Analogous to K-SFA the kernel trick can be utilized to obtain the new objective:

$$\begin{aligned} \min_{\mathbf{A}\in\mathbb{R}^{m\times p}} \tfrac{1}{n-1}\mathbf{tr}\left(\mathbf{A}^\top\mathbf{K}\mathbf{D}\mathbf{D}^\top\mathbf{K}^\top\mathbf{A}\right) + \lambda\,\mathbf{tr}(\mathbf{A}^\top\bar{\mathbf{K}}\mathbf{A}) \\ \text{s.t. } \tfrac{1}{n}\mathbf{A}^\top\mathbf{K}\mathbf{K}^\top\mathbf{A} = \mathbf{I}, \end{aligned} \tag{8}$$

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

2.2 The RSK-SFA Algorithm

The RSK-SFA algorithm (Algorithm 1) is closely related to the linear SFA algorithm of Wiskott and Sejnowski [23]. It consists of three phases: (1) fulfilling zero mean by centring, (2) fulfilling unit variance and decorrelation by sphering and (3) minimizing the objective by rotation.

Zero Mean. To fulfil the zero mean constraint, one centres 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 in Hilbert space $\mathcal{H}$. Although $\forall y \in \mathbf{span}(\{g_i\}_{i=1}^m) : \mathbb{E}_t[y(\boldsymbol{x}_t)] = 0$ already holds, it is of advantage to centre the support functions as well w.r.t. each other [16], 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 $\bar{\mathbf{K}}$:

$$\hat{\mathbf{K}} := (\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\, \mathbf{K}\, (\mathbf{I} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \tag{9}$$

$$\hat{\bar{\mathbf{K}}} := (\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\, \bar{\mathbf{K}}\, (\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\,, \tag{10}$$

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 fulfils the constraint in Equation 8, invariant to further rotations $\mathbf{R} \in \mathbb{R}^{m \times p} : \mathbf{R}^\top\mathbf{R} = \mathbf{I}$.

$$\mathbf{A} := \mathbf{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{R} \quad \Rightarrow \quad \tfrac{1}{n}\mathbf{A}^\top\hat{\mathbf{K}}\hat{\mathbf{K}}^\top\mathbf{A} = \mathbf{R}^\top\mathbf{R} = \mathbf{I} \tag{11}$$

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

Minimization of the Objective. Application of Equation 11 allows us to solve Equation 8 with a second eigenvalue decomposition:

$$\min_{\mathbf{R}} \mathbf{tr}\left(\mathbf{R}^\top \left\{\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{B}\mathbf{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\right\}\mathbf{R}\right) \tag{12}$$

$$\text{s.t.}\, \mathbf{R}^\top\mathbf{R} = \mathbf{I} \quad \text{with} \quad \mathbf{B} := \tfrac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}\,.$$

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

$$y_i(\boldsymbol{x}) = \sum_{j=1}^m A_{ji}\langle \hat{g}_j, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}} \tag{13}$$

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 $\boldsymbol{y}(\cdot) \in \mathcal{H}^p$ can be expressed more compactly:

$$\boldsymbol{y}(\boldsymbol{x}) = \hat{\mathbf{A}}^\top\boldsymbol{k}(x) - \hat{\boldsymbol{c}} \tag{14}$$

$$\text{with } \hat{\mathbf{A}} := \left(\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{A} \text{ and } \hat{\boldsymbol{c}} := \tfrac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n\;.$$

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 $\mathcal{O}(m^2)$.

Algorithm 1. Regularized Sparse Kernel Slow Feature Analysis (RSK-SFA)

```
Input: $\mathbf{K} \in \mathbb{R}^{m \times n}$, $\bar{\mathbf{K}} \in \mathbb{R}^{m \times m}$, $p \in \mathbb{N}$,
       $\lambda \in \mathbb{R}^+$, $\mathbf{D} \in \mathbb{R}^{n \times n-1}$
$\hat{\mathbf{K}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)$      (Eq. 9)
$\hat{\bar{\mathbf{K}}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\bar{\mathbf{K}}(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)$      (Eq. 10)
$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \mathrm{eig}\big(\frac{1}{n}\hat{\mathbf{K}}\hat{\mathbf{K}}^\top\big)$
$(\mathbf{U}_r, \mathbf{\Lambda}_r) = \mathrm{remove\_zero\_eigenvalues}(\mathbf{U}, \mathbf{\Lambda})$
$\mathbf{B} = \frac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}$      (Eq. 12)
$\mathbf{R}\mathbf{\Sigma}\mathbf{R}^\top = \mathrm{eig}\left(\mathbf{\Lambda}_r^{-1/2}\mathbf{U}_r^\top\mathbf{B}\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\right)$      (Eq. 12)
$(\mathbf{R}_p, \mathbf{\Sigma}_p) = \mathrm{keep\_lowest\_p\_eigenvalues}(\mathbf{R}, \mathbf{\Sigma}, p)$
$\hat{\mathbf{A}} = \big(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\big)\,\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\mathbf{R}_p$      (Eq. 11 + 14)
$\hat{\boldsymbol{c}} = \frac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n$      (Eq. 14)
Output: $\hat{\mathbf{A}}$, $\hat{\boldsymbol{c}}$
```

3 Sparse Subset Selection

The representer theorem guarantees the optimal feature maps $y_i^* \in \mathcal{H}$ for training set $\{\boldsymbol{x}_t\}_{t=1}^n$ can be found within $\mathbf{span}(\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^n)$. 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 y^*, it is straight forward to select a subset of the training data, indicated by an *index vector*[2] $\boldsymbol{i} \in \mathbb{N}^m$ with $\{\boldsymbol{z}_j\}_{j=1}^m := \{\boldsymbol{x}_{\boldsymbol{i}_j}\}_{j=1}^m$, that minimizes the *approximation error*

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

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 a NP hard combinatorial problem, but there exist several greedy approximations to it.

Online Maximization of the Affine Hull. A widely used algorithm [5], 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. Downsizing of the subset therefore requires a complete re-computation with larger η. The resulting subset size is not predictable, although monotonically dependent on η.

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

Algorithm 2. Matching Pursuit Maximization of the Affine Hull (MP MAH)

```
Input: $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X},\ \kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R},\ m \in \mathbb{N}$
$\mathbf{K} = \emptyset; \quad \mathbf{K}_1^{-1} = \emptyset;$
$\forall t \in \{1, \dots, n\} : \epsilon_t^1 = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t)$        (Eq. 16)
$\boldsymbol{i}_1 = \underset{t}{\mathrm{argmax}} \{\epsilon_t^1\}_{t=1}^n$        (Eq. 15)
for $j \in \{1, \dots, m-1\}$ do
  $\boldsymbol{\alpha}^j = \left[\mathbf{K}_{(j,:)}\mathbf{K}_j^{-1}, -1\right]^\top$        (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 $t \in \{1, \dots, n\}$ do
    $\mathbf{K}_{(t,j)} = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_{\boldsymbol{i}_j})$
    $\epsilon_t^{j+1} = \epsilon_t^j - \frac{1}{\epsilon_{\boldsymbol{i}_j}^j} \left(\mathbf{K}_{(t,:)}\boldsymbol{\alpha}^j\right)^2$        (Eq. 16)
  end for
  $\boldsymbol{i}_{j+1} = \underset{t}{\mathrm{argmax}} \{\epsilon_t^{j+1}\}_{t=1}^n$        (Eq. 15)
end for
Output: $\{\boldsymbol{i}_1, \dots, \boldsymbol{i}_m\}$
```

Matching Pursuit for Sparse Kernel PCA. This handicap is addressed by *matching pursuit* methods [14]. 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 [18] chooses the sample $\boldsymbol{x}_j$ that minimizes[3] $\mathbb{E}_t\left[\epsilon_t^{\boldsymbol{i} \cup j}\right]$. It was shown later that this algorithm performs sparse PCA in $\mathcal{H}$ [12]. 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.

3.1 Matching Pursuit for Online MAH

The ability to shrink 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 approximation error[4]. At iteration j, given the current subset $\boldsymbol{i} \in \mathbb{R}^j$,

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

[4] 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_∞.

$$\boldsymbol{i}_{j+1} := \operatorname*{argmin}_{t} \| [\epsilon_1^{\boldsymbol{i} \cup t}, \ldots, \epsilon_n^{\boldsymbol{i} \cup t}] \|_\infty \approx \operatorname*{argmax}_{t} \epsilon_t^{\boldsymbol{i}} \tag{15}$$

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:

$$\epsilon_t^{\boldsymbol{i} \cup j} = \epsilon_t^{\boldsymbol{i}} - \frac{1}{\epsilon_j^{\boldsymbol{i}}} \Big(K_{tj} - \boldsymbol{K}_{t\boldsymbol{i}} (\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1} \boldsymbol{K}_{\boldsymbol{i}j} \Big)^2 . \tag{16}$$

Algorithm 2 iterates between sample selection (Equation 15) and error update (Equation 16). The complexity is $\mathcal{O}(m^2 n)$ in time and $\mathcal{O}(mn)$ in memory (to avoid re-computations of $\mathbf{K}_{(:,\boldsymbol{i})}$).

4 Empirical Validation

Slow feature analysis is not restricted to any specific type of time series data. To give a proper evaluation of our kernel SFA algorithm, we therefore chose benchmark data from very different domains. The common element, however, is the existence of a low dimensional underlying cause.

We evaluated all algorithms on two data sets: Audio recordings from a vowel classification task and a video depicting a random sequence of two hand-signs. The second task covers high dimensional image data (40 × 30 pixels), which is a very common setting for SFA. In contrast, mono audio data is one dimensional. Multiple time steps have to be grouped into a sample to create a high dimensional space in which the state space is embedded as a manifold (see Takens Theorem [11, 20]). All experiments employ a Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\big(-\frac{1}{2\sigma^2} \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\big)$.

The true latent variables of the examined data are not known. To ensure that any 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. The variance of a slow feature on unseen data is not strictly specified. This changes the feature's slowness and for comparison we normalized all test outputs to unit variance before measuring the test slowness.

4.1 Benchmark Data Sets

Audio Data. The "north Texas vowel database"[5] contains uncompressed audio files with English words of the form H...D, spoken multiple times by multiple persons [1]. The natural task is to predict the central vowels of unseen instances. We selected two data sets: (1) A small set with four training and four test instances for each of the words "heed" and "head", spoken by the same person. (2) A large training set of four speakers with eight instances per person and each of the

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

words “heed” and “head”. The corresponding test set consists of eight instances of each word spoken by a fifth person. The spoken words are provided as mono audio streams of varying length at 48kHz, i.e. as a series of amplitude readings $\{a_1, a_2, \ldots\}$. To obtain an embedding of the latent variables, one groups a number of amplitude readings into a sample $\boldsymbol{x}_t = [a_{\delta t}, a_{\delta t+\epsilon}, a_{\delta t+2\epsilon}, \ldots, a_{\delta t+(l-1)\epsilon}]^\top$. We evaluated the parameters δ, ϵ and l empirically and chose $\delta = 50$, $\epsilon = 5$ and $l = 500$. This provided us with 3719 samples $\boldsymbol{x}_t \in [-1, 1]^{500}$ for the small and 25448 samples for the large training set. 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. The output of two RSK-SFA features is plotted in Figure 2c.

Video Data. To obtain a video with a simple underlying cause, we recorded a hand showing random sequences of the hand-signs “two-finger salute” and “open palm” with an intermediate “fist” between each sign (Figure 2b). The hand was well lit and had a good contrast to the mostly dark background. The frames were scaled down to 40 × 30 gray-scale pixels, i.e $\boldsymbol{x}_t \in [0, 1]^{1200} \subset \mathbb{R}^{1200}$. Training and test set consist of 3600 frames each, recorded at 24Hz and showing roughly one sign per second.

4.2 Algorithm Performance

Figure 1 shows the test slowness of RSK-SFA features[6] on all three data sets for multiple kernel parameter σ and regularization parameter λ. The small audio data set (Column a) and video data set (Column c) use the complete training set as support vectors, whereas for the large audio data set (Column b) a full kernel approach is not feasible. Instead we selected a subset of size 2500 (based on kernel parameter $\sigma = 2$) with the MP MAH algorithm before training.

In the absence of significant sparseness (Figure 1a and 1c), unregularized kernel SFA ($\lambda = 0$, equivalent to K-SFA, Equations 6 and 7) shows both over-fitting and numerical instability. Over-fitting can be seen at small σ, where the features fulfil the unit variance constraint (lower right plot), but do not reach the minimal test slowness (lower left 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 Figure 1a, a fitting λ can be found between $\sigma = 0.5$ and $\sigma = 20$.

A more common case, depicted in Figure 1b, is a large training set from which a small subset is selected by MP MAH. Here no regularization is necessary and

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

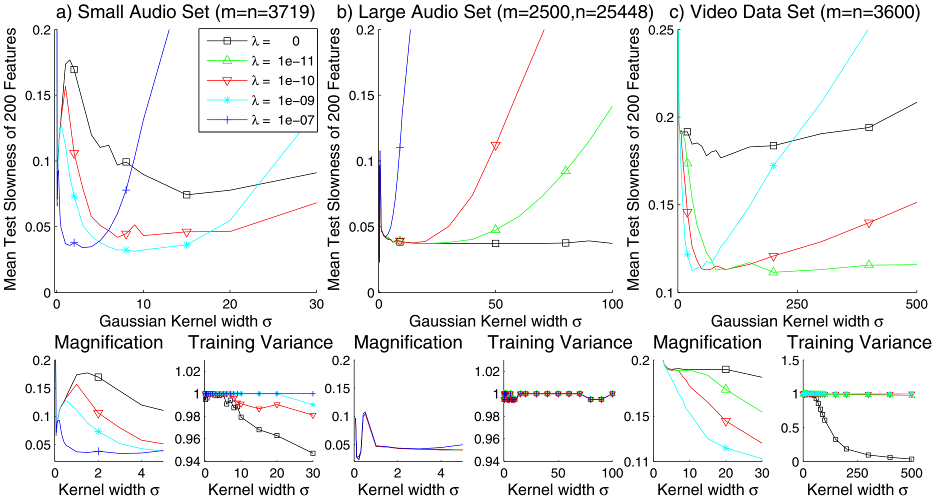

Fig. 1. *(Regularization)* Mean test slowness of 200 RSK-SFA features over varying kernel parameter σ for different regularization parameters λ: **(a)** *Small audio data set* with all $m = n = 3719$ training samples as support vectors, **(b)** *large audio data set* with $m = 2500$ support vectors selected out of $n = 25448$ training samples by MP MAH and **(c)** *video data set* with all $m = n = 3600$ available support vectors. The lower left plots magnify the left side of the above plot. Notice the difference in scale. The lower right plots show the variance of the training output. Significant deviation from one violates the unit variance constraint and thus demonstrates numerical instability. The legend applies to all plots.

unregularized sparse kernel SFA ($\lambda = 0$) learns mappings of minimal slowness in the range from $\sigma = 1$ to far beyond $\sigma = 100$.

4.3 Sparsity

To evaluate the behaviour of RSK-SFA for sparse subsets of different size, Figure 2a plots the test slowness of audio data for all discussed sparse subset selection algorithms. As a baseline, we plotted mean and standard deviation of a random selection scheme. One 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 Discussion

To provide a powerful but easily operated algorithm that performs non-linear *slow feature analysis*, we derived a kernelized SFA algorithm (RSK-SFA). The

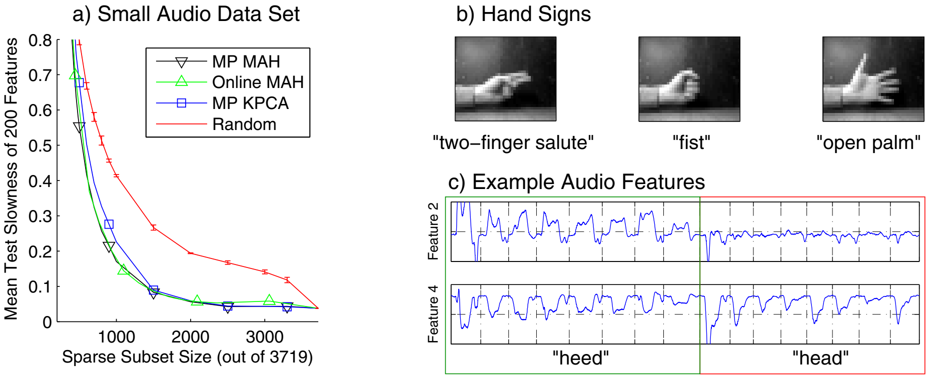

Fig. 2. *(Sparseness)* **(a)** Mean test slowness of 200 RSK-SFA features of the *small audio data set* ($\lambda = 10^{-7}, \sigma = 2$) over sparse subset size, selected by different algorithms. For *random selection* the mean of 10 trials is plotted with standard deviation error bars. **(b)** Examples from the video data set depicting the performed hand-signs. **(c)** Two RSK-SFA features applied on the *large audio test set* ($\lambda = 0, \sigma = 20$). Vertical lines separate eight instances of "heed" and eight instances of "head".

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.

As suggested by previous works, our experiments show that for large data sets no explicit regularization is needed. The implicit regularization introduced by sparseness is sufficient to generate features that generalize well over a wide range of Gaussian kernels. The performance of sparse kernel SFA depends on the sparse subset, selected in a pre-processing step. In this setting, the subset size m takes the place 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 recomputation 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. After selection of a sufficiently large subset, lowering m yields no additional cost. The downside is a quadratic dependency on the training set's size, both in time and memory. Both algorithms showed 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 set's size. However, reducing 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.

The major advancement of our approach over the kernel SFA algorithm of Bray and Martinez [4] 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. Fukumizu et al. [10] have shown this analytically for the related *kernel canonical correlation analysis* and demonstrated the use of *regularization*. In difference to their approach we penalize the *Hilbert* norm of the selected function, rather than regularizing the unit variance constraint. This leads to a faster learning procedure: First one fulfils the constraints as layed out in Section 2.1, followed by repeated optimizations of the objective with slowly increasing λ (starting at 0), until the constraints are no longer violated. The resulting features are numerically stable, but may still exhibit over-fitting. The latter can be reduced by raising λ even further or by additional sparsification. Note that our empirical results in Section 4.2 suggest that for a fitting λ the RSK-SFA solution remains optimal over large regimes of kernel parameter σ, rendering an expensive parameter search unnecessary.

Our experimental results on audio data suggest that RSK-SFA is a promising pre-processing method for audio detection, description, clustering and many other applications. Applied on large unlabelled natural language data bases, e.g. telephone records or audio books, RSK-SFA in combination with MP MAH or Online MAH will construct features that are sensitive to speech patterns of the presented language. If the function class is powerful enough, i.e. provided enough support vectors, those features will encode vowels, syllables or words, depending on the embedding parameters. Based on those features, a small labelled data base might be sufficient to learn the intended task. The amount of support vectors necessary for a practical task, however, is yet unknown and calls for further investigation.

Although few practical video applications resemble our benchmark data, previous studies of SFA on sub-images show it's usefulness in principle [3]. Landmark recognition in camera-based *simultaneous localization and mapping* (Visual SLAM [6]) algorithms is one possible field of application. To evaluate the potential of RSK-SFA in this area, future works must include comparisons to state-of-the-art feature extractions, e.g. the *scale invariant feature transform* (SIFT [13]).

The presented results show RSK-SFA to be a powerful and reliable SFA algorithm. Together with Online MAH or MP MAH, it combines the advantages of regularization, fast feature extraction and large training sets with the performance of kernel methods.

Acknowledgements. This work has been supported by the Integrated Graduate Program on Human-Centric Communication at Technische Universität Berlin and the German Federal Ministry of Education and Research (grant 01GQ0850).

References

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

[2] Becker, S., Hinton, G.E.: A self-organizing neural network that discovers surfaces in randomdot stereograms. Nature 355(6356), 161–163 (1992)

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

[4] Bray, A., Martinez, D.: Kernel-based extraction of Slow features: Complex cells learn disparity and translation invariance from natural images. In: Neural Information Processing Systems, vol. 15, pp. 253–260 (2002)

[5] Csató, L., Opper, M.: Sparse on-line gaussian processes. Neural Computation 14(3), 641–668 (2002)

[6] Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: IEEE International Conference on Computer Vision, pp. 1403–1410 (2003)

[7] Einhäuser, W., Hipp, J., Eggert, J., Körner, E., König, P.: Learning viewpoint invariant object representations using temporal coherence principle. Biological Cybernetics 93(1), 79–90 (2005)

[8] Földiák, P.: Learning invariance from transformation sequences. Neural Computation 3(2), 194–200 (1991)

[9] Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness leads to place, head-direction, and spatial-view cells. PLoS Computational Biology 3(8), e166 (2007)

[10] Fukumizu, K., Bach, F.R., Gretton, A.: Statistical consistency of kernel canonical correlation analysis. Journal of Machine Learning Research 8, 361–383 (2007)

[11] Huke, J.P.: Embedding nonlinear dynamical systems: A guide to takens' theorem. Technical report, University of Manchester (2006)

[12] Hussain, Z., Shawe-Taylor, J.: Theory of matching pursuit. In: Advances in Neural Information Processing Systems, vol. 21, pp. 721–728 (2008)

[13] Lowe, D.G.: Object recognition from local scale-invariant features. In: International Conference on Computer Vision, pp. 1150–1157 (1999)

[14] Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Transactions On Signal Processing 41, 3397–3415 (1993)

[15] Meyn, S.P., Tweedie, R.L.: Markov chains and stochastic stability. Springer, London (1993)

[16] Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)

[17] Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)

[18] Smola, A.J., Schölkopf, B.: Sparse greedy matrix approximation for machine learning. In: Proceedings to the 17th International Conference Machine Learning, pp. 911–918 (2000)

[19] Stone, J.V.: Blind source separation using temporal predictability. Neural Computation 13(7), 1559–1574 (2001)

[20] Takens, F.: Detecting strange attractors in turbulence. Dynamical Systems and Turbulence, 366–381 (1981)

[21] Wahba, G.: Spline Models for Observational Data. Society for Industrial and Applied Mathematics, Philadelphia (1990)

[22] Wiskott, L.: Slow feature analysis: A theoretical analysis of optimal free responses. Neural Computation 15(9), 2147–2177 (2003)

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

[24] Wyss, R., König, P., Verschure, P.F.M.J.: A model of the ventral visual system based on temporal stability and local memory. PLoS Biology 4(5), e120 (2006)