Online Kernel PCA with Entropic Matrix Updates

Dima Kuzmin DIMA@CSE.UCSC.EDU
Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Computer Science Department, University of California - Santa Cruz

Abstract

A number of updates for density matrices have been developed recently that are motivated by relative entropy minimization problems. The updates involve a softmin calculation based on matrix logs and matrix exponentials. We show that these updates can be kernelized. This is important because the bounds provable for these algorithms are logarithmic in the feature dimension (provided that the 2-norm of feature vectors is bounded by a constant). The main problem we focus on is the kernelization of an online PCA algorithm which belongs to this family of updates.

1. Introduction

The are two main families of updates in machine learning when the parameter of the algorithm and the instances are vectors. The first family maintains a parameter vector that is a linear combination of the instances and for the second family the component-wise logarithm of the parameter vector is a linear combination of the instances. Family one is based on regularizing with the squared Euclidean distance and family two uses relative entropy as the regularization (Kivinen & Warmuth, 1997). Both families have their advantages. Family one can be kernelized, i.e. the instances $\boldsymbol{x}$ can be expanded into a feature vector $\phi(\boldsymbol{x})$ and the algorithm can be computed efficiently provided the kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}')$ can be computed efficiently. The advantage of family two is that it allows for bounds that are logarithmic in the dimension of the instances (see e.g. (Warmuth & Vishwanathan, 2005) for an extended discussion).

More recently the entropic family of updates has been generalized to the case when the instances are symmetric matrices $\boldsymbol{X}$ and the parameter is a density matrix (Tsuda et al., 2005; Warmuth & Kuzmin, 2006a; Warmuth & Kuzmin, 2006b; Arora & Kale, 2007). The regularization is now the quantum relative entropy for density matrices instead of the regular relative entropy for probability vectors, and the matrix logarithm of the density matrix parameter is essentially a linear combination of the instance matrices. We show in this paper that if the instances $\boldsymbol{X}$ are outer products of feature vectors,[1] i.e. $\underset{n\times n}{\boldsymbol{X}} = \underset{n\times 1}{\phi(\boldsymbol{x})}\underset{1\times n}{\phi(\boldsymbol{x})^\top}$, then the matrix generalization of the entropic updates can be kernelized. If there is an efficient kernel function available, then the most expensive operation in computing the generalized updates is the computation of the eigendecomposition of the kernel matrix.

2. Offline Kernel PCA

<u>P</u>rincipal <u>C</u>omponent <u>A</u>nalysis (PCA) is a well-known algorithm that chooses a low-dimensional linear subspace that best approximates the data (See Offline PCA Algorithm 1).

The Offline PCA Algorithm can be kernelized (see Offline Kernel PCA Algorithm 2): That is, the instances $\boldsymbol{x}$ can be transformed into feature vectors $\phi(\boldsymbol{x})$, the individual features of the instances never need to be accessed, and the entire computation can be done via calls to a kernel function (Schölkopf et al., 1998). For the sake of simplicity, we assume in this paper that the data is centered (in feature space) and deal with uncentered case in the full paper.

In the kernel setting, the m columns of the data matrix $\boldsymbol{X}$ are the expanded instances $\phi(\boldsymbol{x}_i) \in \mathbb{R}^N$, where the number of features N is typically much larger than the dimension n of the original instances $\boldsymbol{x}_i$. If we ran PCA in feature space, then we would need the eigendecomposition of $\boldsymbol{C} = \underset{N\times N}{\boldsymbol{X}\boldsymbol{X}^\top}$ and this is too expensive when N is large. However there is a related matrix

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

[1] Such outer products are called *dyads*

Algorithm 1 Offline PCA Algorithm

```
Input: Data matrix $\underset{n\times m}{\boldsymbol{X}}$ having the data points $\boldsymbol{x}_i \in \mathbb{R}^n$ as columns, dimension $r$
Compute covariance matrix $\underset{n\times n}{\boldsymbol{C}} = \boldsymbol{X}\boldsymbol{X}^\top$
Compute eigendecomposition of the covariance matrix
    $\boldsymbol{C} = \sum_{i=1}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$
Let $\underset{n\times r}{\boldsymbol{U}_r}$ be the column matrix of eigenvectors for the $r$ largest eigenvalues. Also, $\underset{n\times n}{\boldsymbol{P}} = \boldsymbol{U}_r\boldsymbol{U}_r^\top$
Projection: $\boldsymbol{U}_r^\top \boldsymbol{x}$ is the $r$-dimensional compression vector for $\boldsymbol{x}$ and $\boldsymbol{P}\boldsymbol{x}$ is the projection into the original space
Approximation error:
    $\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2 = \mathrm{tr}((\boldsymbol{I} - \boldsymbol{P})\boldsymbol{x}\boldsymbol{x}^\top)$
```

$\boldsymbol{K} = \underset{m\times m}{\boldsymbol{X}^\top \boldsymbol{X}}$ of the same rank called the *kernel matrix* because $\boldsymbol{K}_{ij} = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Kernel PCA gets away with computing only the eigendecomposition of this smaller matrix and the projection onto the principal components can also be obtained via kernel computations. The following lemma relates the eigensystems of the two matrices.

Lemma 1. *If the kernel matrix $\boldsymbol{K}$ has eigenvalues λ_i and eigenvectors $\boldsymbol{u}_i$, then the covariance matrix $\boldsymbol{C}$ has the same eigenvalues λ_i and orthonormal eigenvectors $\frac{\boldsymbol{X}\boldsymbol{u}_i}{\sqrt{\lambda_i}}$.*

Proof. Since

$$\boldsymbol{C}\,\boldsymbol{X}\boldsymbol{u}_i = \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{u}_i = \boldsymbol{X}\boldsymbol{K}\boldsymbol{u}_i = \lambda_i \boldsymbol{X}\boldsymbol{u}_i,$$

the eigenvalues λ_i for $\boldsymbol{K}$ are also eigenvalues for $\boldsymbol{C}$. The orthogonality of the eigenvectors $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ of $\boldsymbol{K}$ implies that $\boldsymbol{X}\boldsymbol{u}_i$ and $\boldsymbol{X}\boldsymbol{u}_j$ are also orthogonal:

$$(\boldsymbol{X}\boldsymbol{u}_i)^\top (\boldsymbol{X}\boldsymbol{u}_j) = \boldsymbol{u}_i^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{u}_j = \boldsymbol{u}_i^\top \boldsymbol{K}\boldsymbol{u}_j = \lambda_j \boldsymbol{u}_i^\top \boldsymbol{u}_j = 0.$$

Finally, the $\frac{1}{\sqrt{\lambda_i}}$ factor makes our eigenvectors have norm 1:

$$\|\frac{\boldsymbol{X}\boldsymbol{u}_i}{\sqrt{\lambda_i}}\|_2^2 = \frac{1}{\lambda_i}(\boldsymbol{X}\boldsymbol{u}_i)^\top (\boldsymbol{X}\boldsymbol{u}_i) = \frac{1}{\lambda_i}\boldsymbol{u}_i^\top \boldsymbol{K}\boldsymbol{u}_i = \frac{1}{\lambda_i}\lambda_i \boldsymbol{u}_i^\top \boldsymbol{u}_i.$$

We conclude that the vectors $\frac{\boldsymbol{X}\boldsymbol{u}_i}{\sqrt{\lambda_i}}$ form an orthonormal eigensystem for $\boldsymbol{C}$. □

Algorithm 2 Offline Kernel PCA Algorithm

```
Input: Data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$, dimension $r$, kernel function $k$
Compute kernel matrix $\underset{m\times m}{K_{ij}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
Compute eigendecomposition of the kernel matrix
    $\boldsymbol{K} = \sum_{i=1}^{m} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$
Let $\underset{m\times r}{\boldsymbol{U}_r}$ be the matrix with the top $r$ eigenvectors as columns and let $\underset{r\times r}{\boldsymbol{\Lambda}_r}$ be the diagonal matrix of the top $r$ eigenvalues
Compute $\underset{m\times m}{\widetilde{\boldsymbol{P}}} = \boldsymbol{U}_r \boldsymbol{\Lambda}_r^{-1} \boldsymbol{U}_r^\top$
For a new data point $\boldsymbol{x}$ compute the kernel vector
    $\underset{m\times 1}{\boldsymbol{y}_i} = k(\boldsymbol{x}_i, \boldsymbol{x})$
Projection: $\boldsymbol{\Lambda}_r^{-1} \boldsymbol{U}_r^\top \boldsymbol{y}$ is the $r$-dimensional compression vector for $\boldsymbol{x}$
Approximation error:
    $k(\boldsymbol{x}, \boldsymbol{x}) - \mathrm{tr}(\widetilde{\boldsymbol{P}} \boldsymbol{y}\boldsymbol{y}^\top)$
```

3. Online Kernel PCA Algorithm

Online algorithms often maintain a parameter which expresses the uncertainty of the algorithm over which choice is good. For example, in the expert setting, Weighted Majority Algorithm maintains a probability vector over the experts. The algorithm hedges its bets by using a softmin function on the experts' losses so far, since committing to the best expert can be easily exploited by an adversary.

In the case of PCA the uncertainty over the r-dimensional subspace can be represented as a density matrix. The Online PCA Algorithm in (Warmuth & Kuzmin, 2006b) uses this idea and has good worst-case loss bounds. It uses the density matrix update based on matrix logs and exponentials like the one used in (Tsuda et al., 2005; Warmuth & Kuzmin, 2006a; Arora & Kale, 2007) but with the additional constraint that the eigenvalues are upper bounded (capped). Density matrices are symmetric, positive definite matrices of trace one. Thus their eigenvalues form a probability vector and the whole matrix can be seen as a mixture of the dyads formed by its eigenvectors.

A subspace of dimension r is characterized by a projection matrix $\boldsymbol{P}$ of rank r. We need to work with the complementary subspace which is characterized by the projection matrix $\boldsymbol{I}-\boldsymbol{P}$ of rank $n-r$. The matrix $\boldsymbol{I}-\boldsymbol{P}$ is the sum of $n-r$ orthogonal dyads and the scaled projection matrix $\frac{1}{n-r}\boldsymbol{P}$ is a density matrix with $n-r$ non-zero eigenvalues of value $\frac{1}{n-r}$. The Online PCA Algorithm maintains a mixture of scaled rank $n-r$ projection matrices and such mixtures are density matrices with the constraint that each eigenvalue is at most $\frac{1}{n-r}$. Crucially, every convex combination of rank $n-r$ projection matrices is a constrained density matrix and every such density matrix can be written as convex combination of at most n rank $n-r$ projection matrices, thus making capped density matrices the natural objects for representing uncertainty over subspaces.

The parameter matrix $\boldsymbol{W}^t$ of the Online PCA Algorithm at trial t is a capped density matrix. The parameter update involves matrix logs and exponentials as well as a step that caps the eigenvalues of the updated matrix. At each trial the online algorithm produces a rank r projection matrix $\boldsymbol{P}^t$ for compressing the current data point $\boldsymbol{x}^t$ by decomposing the capped density matrix $\boldsymbol{W}^t$ into a mixture of scaled projection matrices of rank $n-r$. From this mixture it is easy to probabilistically choose a single projection matrix of rank $n-r$ and then choose its complement as the projection matrix $\boldsymbol{P}^t$ used for compressing at trial t. The decomposition and the capping can be done efficiently via algorithms 4 and 5, respectively. Note that capping in this context doesn't mean that we just set any components above the threshold to the threshold value, since this would result in a vector that does not sum to one anymore. More details on the Decomposition Algorithm 4 are provided in (Warmuth & Kuzmin, 2006b) and a linear time implementation of the Capping Algorithm 5 is given in (Herbster & Warmuth, 2001). The Online PCA Algorithm 3 gives a summary of all the steps. Here a *corner* is a diagonal matrix with $n-r$ of the diagonal elements set to $\frac{1}{n-r}$ and the rest set to zero.

An important property of the standard PCA problem is that the 2-norm approximation error is linear in the matrix parameter space:

$$\|\boldsymbol{x}-\boldsymbol{P}\boldsymbol{x}\|_2^2 = \mathrm{tr}((\boldsymbol{I}-\boldsymbol{P})\boldsymbol{x}\boldsymbol{x}^\top)$$

This means that the expected loss of the online algorithm that chooses a subspace probabilistically based on the decomposed density matrix $\boldsymbol{W} = \frac{1}{n-r}\sum_i p_i(\boldsymbol{I}-\boldsymbol{P}_i)$ will be equal to the linear loss of the matrix parameter $\boldsymbol{W}$:

$$\sum_i p_i\|\boldsymbol{x}-\boldsymbol{P}_i\boldsymbol{x}\|_2^2 = (n-r)\mathrm{tr}(\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^\top).$$

(We need to multiply by $n-r$ because the density

Algorithm 3 Online PCA Algorithm

```
Input: Dimension $r$ and initial density matrix $\boldsymbol{W}^1$
with eigenvalues at most $\frac{1}{n-r}$, learning rate $\eta > 0$
for $t = 1$ to $T$ do
    Perform eigendecomposition $\boldsymbol{W}^t = \boldsymbol{U}\boldsymbol{\lambda}\boldsymbol{U}^\top$
    Decompose $\boldsymbol{\lambda}$ into a mixture of $n$ corners using
    the Decomposition Algorithm 4: $\boldsymbol{\lambda} = \sum_{j=1}^n p_j\boldsymbol{c}_j$,
    where the $p_j$ are the mixture coefficients and $\boldsymbol{c}_j$
    the corners.
    Draw a corner $\boldsymbol{c} = \boldsymbol{c}_j$ with probability $p_j$
    Form a matrix corner $\boldsymbol{R} = \boldsymbol{U}\,\mathrm{diag}(\boldsymbol{c})\boldsymbol{U}^\top$
    Form a rank $r$ projection matrix
        $$\boldsymbol{P}^t = \boldsymbol{I} - (n-r)\boldsymbol{R}$$
    Let $\boldsymbol{U}_r^t$ be a column matrix of those $r$ eigenvectors
    that correspond to zero entries in $\boldsymbol{c}$
    Receive data instance vector $\boldsymbol{x}^t$
    Projection: $(\boldsymbol{U}_r^t)^\top\boldsymbol{x}^t$ is the $r$-dimensional com-
    pression vector for $\boldsymbol{x}^t$
    Approximation error:
        $$\|\boldsymbol{x}^t - \boldsymbol{P}^t\boldsymbol{x}^t\|_2^2 = \mathrm{tr}((\boldsymbol{I}-\boldsymbol{P}^t)\,\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)$$
    Expected approximation error:
        $$(n-r)\mathrm{tr}(\boldsymbol{W}^t\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)$$
    Update:
        $$\widehat{\boldsymbol{W}}^t = \frac{\exp(\log\boldsymbol{W}^t - \eta\,\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)}{Z^t},$$
    where $Z^t$ normalizes the trace to 1
    $\boldsymbol{W}^{t+1}$ is obtained from $\widehat{\boldsymbol{W}}^t$ by capping the eigen-
    values to at most $\frac{1}{n-r}$ using the Capping Algo-
    rithm 5.
end for
```

Algorithm 4 Decomposition Algorithm

```
Input probability vector $\boldsymbol{w}$, $\max(\boldsymbol{w}) \le 1/k$
repeat
    Pick a $k$-corner $\boldsymbol{r}$ whose components correspond
    to non-zero components of $\boldsymbol{w}$ and contain all the
    components of $\boldsymbol{w}$ that are equal to $\frac{|\boldsymbol{w}|}{k}$
    Let $s$ be the smallest of the $k$ components in $\boldsymbol{w}$
    Let $l$ be the largest of the remaining $n-k$ com-
    ponents
        $$\boldsymbol{w} := \boldsymbol{w} - \overbrace{\min(m\,s, |w| - m\,l)}^{p}\,\boldsymbol{r}$$
    output $p\,\boldsymbol{r}$
until $\boldsymbol{w} = \boldsymbol{0}$
```

Algorithm 5 Capping Algorithm

```
Input $\boldsymbol{\omega}$ probability vector, $1/k$ capping constant
Let $\boldsymbol{\omega}^{\downarrow}$ index the vector in decreasing order, i.e. $\omega_1^{\downarrow} = \max(\boldsymbol{\omega})$
if $\max(\boldsymbol{\omega}) \leq 1/k$ then
    Return $\boldsymbol{\omega}$
end if
$i := 1$
repeat
    Set first $i$ largest components to $1/k$ and normalize the rest:
    $\bar{\boldsymbol{\omega}} := \boldsymbol{\omega}$
    $\bar{\omega}_j^{\downarrow} := 1/k,\ j = 1 \dots i$
    $\bar{\omega}_j^{\downarrow} := \frac{k-i}{k} \frac{\bar{\omega}_j^{\downarrow}}{\sum_{l=i+1}^{n} \bar{\omega}_l^{\downarrow}},\ j = i+1 \dots n$
    $i := i + 1$
until $\max(\bar{\boldsymbol{\omega}}) \leq 1/k$
Return $\bar{\boldsymbol{\omega}}$
```

matrix is a mixture of scaled projection matrices.) One might want to solve a PCA-like problem, where the approximation error is measured by something other than a 2-norm. Online learning techniques exist for dealing with very many different loss functions (see e.g. (Kivinen & Warmuth, 1999)), but only for the case of linear loss will the expectation of the losses equal the loss of the expected parameter. Thus, for other loss functions the algorithm might have good bounds, but it won't produce an actual low-dimensional subspace as a prediction.

Our new Online Kernel PCA Algorithm uses techniques from the Offline Kernel PCA Algorithm and makes use of the relationship between the offline and online PCA algorithms. Note that offline PCA picks the eigenvectors corresponding to the top k eigenvalues of the covariance matrix. Online PCA can be seen as a different way of choosing k eigenvectors from the eigensystem of the data covariance matrix: Probabilistically choose a rank $n-k$ subspace based on a softmin function of the eigenvalues and then use the complementary subspace of rank k as the subspace for compression. Online kernel PCA uses a softmin function on the kernel matrix instead of the covariance matrix.

First note that if we drop the capping constraint and initialize with $\boldsymbol{W}^1 = \frac{1}{n}\boldsymbol{I}$, then the density matrix at step t of the Online PCA Algorithm 3 has the form

$$\boldsymbol{W}^t = \frac{\mathbf{exp}(-\eta \sum_{i=1}^{t-1} \boldsymbol{x}^i (\boldsymbol{x}^i)^\top)}{Z^t}.$$

Thus in this case, the parameter matrix $\boldsymbol{W}^t$ has the same eigensystem as the covariance matrix and its eigenvalues are obtained by applying a softmin function to the eigenvalues of the covariance matrix. By Lemma 1 which relates the eigensystems of the covariance and kernel matrices, all the information needed to compute $\boldsymbol{W}^t$ is given by the eigendecomposition of the smaller kernel matrix. The capped version[2] of the above $\boldsymbol{W}^t$ will be the crucial parameter matrix implicitly maintained by our Online Kernel PCA Algorithm 6. Our algorithm differs from the Offline Kernel PCA Algorithm only in the way it chooses k eigenvectors from the kernel matrix for forming the projection matrix: instead of taking the eigenvectors corresponding to the maximum k eigenvalues, it picks $n-k$ eigenvectors probabilistically based on a softmin function of the eigenvalues and then chooses the complementary k eigenvectors as the projection matrix. This more complicated procedure assures the expected compression error for all sequences of examples is not too much larger than the compression error of the offline algorithm.

The next section discusses how the proof of the original loss bound for online PCA needs to be altered to work with this kernelized version.

4. Relative Loss Bounds for Online Kernel PCA

We begin by recalling the motivation for the online PCA update as a solution of some optimization problem. The regularization used in this optimization is the quantum relative entropy for density matrices, which is defined as:

$$\Delta(\boldsymbol{V}, \boldsymbol{W}) = \operatorname{tr}(\boldsymbol{V}(\mathbf{log}\, \boldsymbol{V} - \mathbf{log}\, \boldsymbol{W})) - \operatorname{tr}(\boldsymbol{V}) + \operatorname{tr}(\boldsymbol{W}).$$

The update of the Online PCA Algorithm 3 is determined as

$$\boldsymbol{W}^{t+1} = \underset{\substack{\operatorname{tr}(\boldsymbol{W})=1 \\ \boldsymbol{W} \preceq \frac{1}{N-r}\boldsymbol{I}}}{\operatorname{argmin}} \Big(\Delta(\boldsymbol{W}, \boldsymbol{W}^t) + \eta \operatorname{tr}(\boldsymbol{W} \boldsymbol{x}^t (\boldsymbol{x}^t)^\top) \Big). \tag{1}$$

We can't kernelize the above update because the iterative capping complicates things.[3] Instead we always go back to the beginning, which allows us to define

[2]Note that in the Online Kernel PCA Algorithm 6 the kernel matrix is capped with $\frac{1}{t-r}$ whereas in the original Online PCA Algorithm the covariance matrix is capped with $\frac{1}{N-r}$. It can be shown that the two types of capping have the same effect.

[3]Specifically, the accumulated capping Lagrangians can make the eigenvectors of the resulting density matrix different from the eigenvectors of the data covariance matrix. If you look at equation (4) in the proof of Theorem 1 later in this section, you can deduce the form of $\boldsymbol{W}^t$ for the above update: $\boldsymbol{W}^t \sim \mathbf{exp}(-\eta \sum \boldsymbol{x}^i (\boldsymbol{x}^i)^\top - \sum \boldsymbol{\Gamma}^i)$. In general, this is no longer a spectral function of the data covariance

Algorithm 6 Online Kernel PCA Algorithm

```
Input: dimension $r$, kernel function $k$, initial $r$ data points $\boldsymbol{x}^1, \dots, \boldsymbol{x}^r \in \mathbb{R}^n$, learning rate $\eta$
for $t = r+1$ to $T$ do
    Compute kernel matrix on data so far:
        $\underset{(t-1)\times(t-1)}{\boldsymbol{K}^t_{ij}} = k(\boldsymbol{x}^i, \boldsymbol{x}^j)$
    Eigendecompose the kernel matrix
        $\boldsymbol{K}^t = \sum_{i=1}^{t} \lambda_i \, \boldsymbol{u}_i \boldsymbol{u}_i^\top = \boldsymbol{U}^t \boldsymbol{\Lambda}^t (\boldsymbol{U}^t)^\top$
    Transform the eigenvalues:
        $\bar{\lambda}_i \leftarrow \frac{e^{-\eta\lambda_i}}{Z}, \;\; Z = \sum_{j=1}^{t-1} e^{-\eta\lambda_j}$
    Cap values in vector $\bar{\boldsymbol{\lambda}}$ to at most $\frac{1}{t-1-r}$ using the Capping Algorithm 5
    Decompose $\bar{\boldsymbol{\lambda}}$ as $\sum_j p_j \boldsymbol{c}_j$, where $\boldsymbol{c}_j$ are corners of size $t-1-r$ using the Decomposition Algorithm 4
    Draw the corner $\boldsymbol{c} = \boldsymbol{c}_j$ with probability $p_j$
    Form matrix $\underset{(t-1)\times(r)}{\boldsymbol{U}^t_r}$ with the columns being those eigenvectors of $\boldsymbol{K}^t$ that correspond to zero values in the selected corner $\boldsymbol{c}$.
    Let $\boldsymbol{\Lambda}^t_r$ be the diagonal matrix of the eigenvalues of $\boldsymbol{K}^t$ associated with eigenvectors in $\boldsymbol{U}^t_r$.
        $\widetilde{\boldsymbol{P}}^t = \boldsymbol{U}^t_r (\boldsymbol{\Lambda}^t_r)^{-1} (\boldsymbol{U}^t_r)^\top$
        $\widetilde{\boldsymbol{W}}^t = \boldsymbol{U}^t (\boldsymbol{I} - (t-1-r)\,\mathrm{diag}(\bar{\boldsymbol{\lambda}})) (\boldsymbol{\Lambda}^t)^{-1} (\boldsymbol{U}^t)^\top$
    Receive data instance vector $\boldsymbol{x}^t$
    Compute kernel vector $\boldsymbol{y}^t_i = k(\boldsymbol{x}_i, \boldsymbol{x}^t)$
    Projection: $(\boldsymbol{\Lambda}^t_r)^{-1} (\boldsymbol{U}^t_r)^\top \boldsymbol{y}^t$ is the $r$-dimensional compression vector for $\boldsymbol{x}^t$
    Approximation error:
        $k(\boldsymbol{x}^t, \boldsymbol{x}^t) - \mathrm{tr}(\widetilde{\boldsymbol{P}}^t \boldsymbol{y}^t (\boldsymbol{y}^t)^\top)$
    Expected approximation error:
        $k(\boldsymbol{x}^t, \boldsymbol{x}^t) - \mathrm{tr}(\widetilde{\boldsymbol{W}}^t \boldsymbol{y}^t (\boldsymbol{y}^t)^\top)$
end for
```

$\boldsymbol{W}^{t+1}$ by a single capping:

$$\boldsymbol{W}^{t+1} = \underset{\substack{\mathrm{tr}(\boldsymbol{W})=1 \\ \boldsymbol{W} \preceq \frac{1}{N-r}\boldsymbol{I}}}{\mathrm{argmin}} \Big(\Delta(\boldsymbol{W}, \frac{1}{N}\boldsymbol{I}) + \eta \sum_{i=1}^{t} \mathrm{tr}(\boldsymbol{W}\boldsymbol{x}^i(\boldsymbol{x}^i)^\top)\Big). \tag{2}$$

We will use $\boldsymbol{C}^{t+1}$ to denote the covariance matrix used in obtaining $\boldsymbol{W}^{t+1}$, i.e $\boldsymbol{C}^{t+1} = \sum_{i=1}^{t} \boldsymbol{x}^i (\boldsymbol{x}^i)^\top$. The resulting density matrix used by the algorithm looks like this:

$$\boldsymbol{W}^{t+1} = \mathrm{cap}\left(\frac{\mathbf{exp}(-\eta \boldsymbol{C}^{t+1})}{Z}\right)$$

This is a spectral function of $\boldsymbol{C}^{t+1}$ and thus allows for easy kernelization. With some additional effort, we were able to prove the same loss bound for the update (2) as was proven in (Warmuth & Kuzmin, 2006b) for the update (1). This is done in the following theorem, which says that the loss incurred by the algorithm over a sequence of trials is not much larger than the loss incurred by the offline algorithm that is given all of the data in advance. In our case this offline algorithm is the standard PCA for the entire data. Note that with our representation of mixtures of subspaces as density matrices, the expected loss of algorithm (i.e. the expected approximation error) at trial t has the form $(N-r)\mathrm{tr}(\boldsymbol{W}^t \boldsymbol{x}^t (\boldsymbol{x}^t)^\top)$. The expectation is wrt the internal randomization of the algorithm, the bound itself makes no probabilistic assumptions on the data.

Theorem 1. *For any sequence of data $\boldsymbol{x}^1, \dots, \boldsymbol{x}^T \in \mathbb{R}^n$, such that the squared 2-norm of each instance is bounded by 1*[4] *, positive learning rate η and arbitrary capped comparator density matrix $\boldsymbol{V}$, the following relative loss bound holds:*

$$\sum_{t=1}^{T} (N-r)\mathrm{tr}(\boldsymbol{W}^t \boldsymbol{x}^t (\boldsymbol{x}^t)^\top) \le$$

$$(N-r)\frac{\eta \sum_{t=1}^{T} \mathrm{tr}(\boldsymbol{V}\boldsymbol{x}^t(\boldsymbol{x}^t)^\top) + \Delta(\boldsymbol{V}, \frac{1}{N}\boldsymbol{I})}{1 - \exp(-\eta)}.$$

Proof. As was mentioned before, the update we use is different from online PCA described previously, and thus the proof of (Warmuth & Kuzmin, 2006b) does not apply. The old proof relied on bounding the progress of the algorithm in terms of the quantum relative entropy and then using generalized Pythagorean

matrix. On the other hand, capping once only affects the eigenvalues of the capped matrix as shown in (Warmuth & Kuzmin, 2006b).

[4] The bound can be generalized based on the assumption that the squared norm of the expanded instances $\phi(\boldsymbol{x}_t)$ is upper bounded by some constant Q.

theorem for Bregman projections to deal with the capping constraints. Here we adopt a somewhat different approach which makes use of the dual optimization problem of the update (2). Let $F^t(\boldsymbol{W})$ be the objective function of this update, i.e.

$$F^t(\boldsymbol{W}) = \Delta(\boldsymbol{W}, \frac{1}{N}\boldsymbol{I}) + \eta \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}^t)$$

Following (Kivinen & Warmuth, 1999), we use the optimum value of $F^t(\boldsymbol{W})$ as a potential, i.e.

$$P^t := \min_{\substack{\operatorname{tr}(\boldsymbol{W})=1 \\ \boldsymbol{W} \preceq \frac{1}{N-r}\boldsymbol{I}}} F^t(\boldsymbol{W})$$

We would like to lower bound the per trial drop of the potential in terms of the loss of the algorithm.

$$P^{t+1} - P^t \tag{3}$$

We use strong duality (Boyd & Vandenberghe, 2004) to compute the potential as a maximum of the dual problem. The purpose of going to the dual problem is only to help us analyze the algorithm, the algorithm itself is fully determined by the optimization problem in (2) and kernelization of the resulting density matrix computations. We begin by formulating a Lagrangian with δ as the dual variable for the trace constraint and the symmetric positive definite matrix $\boldsymbol{\Gamma}$ as the dual variable for the capping constraint. For the relevant derivatives see (Tsuda et al., 2005).

$$L^t(\boldsymbol{W}, \boldsymbol{\Gamma}, \delta) = \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} + \ln N\,\boldsymbol{I})) - \operatorname{tr}(\boldsymbol{W})$$
$$- \ln N + \eta \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}^t) + \delta(\operatorname{tr}(\boldsymbol{W}) - 1) + \operatorname{tr}((\boldsymbol{W} - \frac{1}{r}\boldsymbol{I})\boldsymbol{\Gamma})$$

$$\frac{\partial L^t}{\partial \boldsymbol{W}} = \mathbf{log}\,\boldsymbol{W} + \ln N\,\boldsymbol{I} + \eta\boldsymbol{C}^t + \delta\boldsymbol{I} + \boldsymbol{\Gamma}.$$

The extra constant $\ln N$ can be rolled into δ, i.e. $\delta' = \delta + \ln N$. Setting the above to zero we get:

$$\boldsymbol{W} = \mathbf{exp}(-\eta\boldsymbol{C}^t - \delta'\boldsymbol{I} - \boldsymbol{\Gamma}). \tag{4}$$

We substitute this form of $\boldsymbol{W}$ into L^t and enforce the $\operatorname{tr}(\boldsymbol{W}) = 1$ constraint by choosing $\delta' = \ln \operatorname{tr}(\mathbf{exp}(-\eta\boldsymbol{C}^t - \boldsymbol{\Gamma}))$. After numerous simplifications we arrive at the dual optimization problem

$$\max_{\boldsymbol{\Gamma} \succeq \mathbf{0}} L^t(\boldsymbol{\Gamma}) = -\ln \operatorname{tr}\left(\mathbf{exp}(-\eta\boldsymbol{C}^t - \boldsymbol{\Gamma}) - 1 - \frac{1}{r}\operatorname{tr}(\boldsymbol{\Gamma})\right)$$

Let $\boldsymbol{\Gamma}^t$ be the maximizer of the dual optimization problem for L^t and define $\boldsymbol{\Gamma}^{t+1}$ similarly. Strong duality holds for our problem. We now want to lower bound the rewritten drop of the potential (3) by the same expression that was used in (Warmuth & Kuzmin, 2006a) for their proof:

$$L^{t+1}(\boldsymbol{\Gamma}^{t+1}) - L^t(\boldsymbol{\Gamma}^t) \geq$$
$$-\ln \operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}^t - \eta\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)). \tag{5}$$

We begin by substituting the solution

$$\boldsymbol{W}^t = \mathbf{exp}(-\eta\boldsymbol{C}^t - \ln \operatorname{tr}(\mathbf{exp}(-\eta\boldsymbol{C}^t - \boldsymbol{\Gamma}^t))\boldsymbol{I} - \boldsymbol{\Gamma}^t)$$

into the rhs and rewrite it as follows:

$$-\ln \operatorname{tr}(\mathbf{exp}(-\eta\boldsymbol{C}^{t+1} - \boldsymbol{\Gamma}^t)) + \ln \operatorname{tr}(\mathbf{exp}(-\eta\boldsymbol{C}^t - \boldsymbol{\Gamma}^t))$$
$$= L^{t+1}(\boldsymbol{\Gamma}^t) - L^t(\boldsymbol{\Gamma}^t).$$

By substituting this rhs into (5) and simplifying, we get that inequality (5) holds iff

$$L^{t+1}(\boldsymbol{\Gamma}^{t+1}) - L^{t+1}(\boldsymbol{\Gamma}^t) \geq 0.$$

However this inequality trivially hold, since $\boldsymbol{\Gamma}^{t+1}$ maximizes $L^{t+1}(\boldsymbol{\Gamma})$.

We now lower bound the rhs of (5) in term of the loss of the algorithm in the usual way (Warmuth & Kuzmin, 2006a):

$$\text{r.h.s.} \geq \operatorname{tr}(\boldsymbol{W}^t\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)(1 - e^{-\eta})$$

Finally we sum the drop of the potential (5) over t. Telescoping occurs and we get the following bound:

$$P^{T+1} - \overbrace{P^1}^{0} = L^{T+1}(\boldsymbol{\Gamma}^{T+1}) - L^1(\boldsymbol{\Gamma}^1)$$
$$\geq \sum_{t=1}^{T} \operatorname{tr}(\boldsymbol{W}^t\boldsymbol{x}^t(\boldsymbol{x}^t)^\top)(1 - e^{-\eta})$$

Since

$$P^{T+1} \leq \Delta(\boldsymbol{V}, \frac{1}{n}) + \eta \operatorname{tr}(\boldsymbol{V}\boldsymbol{C}^{T+1}),$$

the bound follows. □

We now set η as a function of N, r, the maximum 2-norm Q of the instances $\phi(\boldsymbol{x}_t)$ and an upper bound on the loss of the best r-projection $\hat{L}$ (Freund & Schapire, 1997):

$$\eta = \frac{\ln(1 + \sqrt{\frac{2r \ln \frac{N}{r}}{\hat{L}}})}{Q^2}. \tag{6}$$

This tuning of η results in the following bound for the Online Kernel PCA Algorithm, that holds for all sequences with 2-norm of expanded instances $\phi(\boldsymbol{x}^t)$ bounded by Q and that have the loss of the best r-projection bounded by $\hat{L}$:

$$\text{(expected loss of alg.) - (loss best } r\text{-projection)}$$
$$\leq Q\sqrt{2\hat{L}\,r \ln \frac{N}{r}} + rQ^2 \ln \frac{N}{r}. \tag{7}$$

Note that the dependence on the feature dimension N is logarithmic. Therefore we can accommodate a large number of features without degrading the bound substantially. For example for the polynomial kernel of degree d, $\ln N = d \ln n$, and if the feature vectors have bounded 2-norm then the bound is still reasonable. The current analysis is not directly applicable to the infinite-dimensional kernels. To handle that case we need a different way to bound the entropy of the comparator $\Delta(\boldsymbol{V}, \frac{1}{n}\boldsymbol{I})$. Intuitively, the comparator $\boldsymbol{V}$ will only use the the finite dimensional subspace spanned by the past expanded instances and thus the entropy remains bounded (probably by $O(\log T)$), but we haven't worked out the details yet.

5. Experiments

We implemented the Online Kernel PCA Algorithm and tested it on a simple synthetic dataset generated as follows: A point was picked at random on a cone defined by the equation $x^2 + y^2 - z^2 = 0$. These three dimensions were embedded into $\mathbb{R}^{20}$ by padding with zeros. Finally Gaussian noise was added to all dimensions. The value of z was constrained to the range $[-0.6 : 0.6]$, which had the effect of bounding the 2-norm in feature space to at most 1. The resulting dataset is visualized in Figure 1.

Since the cone is a quadratic surface, PCA without kernelization is not able to pick up the 2-dimensional structure of this dataset. A polynomial kernel of degree 2 (i.e., $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y})^2$) is more suitable (the dimension N of the feature space in this case is 400). Figure 2 shows the total loss of our Online Kernel PCA Algorithm 6 and compares it to the total loss of the Offline Kernel PCA Algorithm 2 whose projection matrix is formed based on the entire dataset. As we can see the online algorithm is not too much worse. We also plotted the upper bound of Theorem 1 which not very tight on this random data set. (Note that our bounds hold for worst-case sequences.) Figure 3 shows the average "regret" per time step of our Online Kernel PCA Algorithm, i.e. the difference in total losses between the online and offline kernel PCA algorithms divided by the number of iterations so far.

While implementing the algorithm we found some numerical stability issues that need to be addressed. First, the theoretical guarantee for the algorithm allows us to assume an arbitrary bound Q on the 2-norm of expanded instance vectors $\phi(\boldsymbol{x}_t)$. In practice, however, large norms of the data instances lead to overly large eigenvalues of the kernel matrix, which then cause problems during exponentiation for the softmin computation. This difficulty is specific to our Online Kernel PCA Algorithm 6, which at trail t uses the kernel matrix formed by the past $t-1$ instances and the eigenvalues of this matrix may diverge. Appropriate scaling of the instance avoids this problem.

Second, the Capping Algorithm 5 may encounter the case where the remaining portion of the probability vector is all zeros due to the exponentiation of large numbers. In that case, a division by zero can occur. The proper approach in this case is to set enough of the remaining zeros to the capping constant, so that the whole vector sums to one again.

We also did some preliminary experiments using human face datasets and the Gaussian kernel. While we don't yet have a theoretical guarantee for infinite-dimensional feature spaces, the algorithm still seems to work in practice.

6. Conclusion

The question of whether kernels can be applied to updates that are motivated with relative entropy regularizations has generated great interest (See e.g. discussion in (Warmuth & Vishwanathan, 2005)). Only very few cases have been found for applying kernels with entropic updates when the instances are vectors (Takimoto & Warmuth, 2003). In this paper we show how the matrix generalizations of these updates can be kernelized when the instance matrices are of the form $\phi(\boldsymbol{x}_t)\phi(\boldsymbol{x}_t)^\top$. We chose the online PCA problem as our example problem for working out the details. The crucial property needed is that the exponent must be a linear combination of the outer products $\phi(\boldsymbol{x}_t)\phi(\boldsymbol{x}_t)^T$. The other crucial property is that the exponential of this linear combination only depends on the eigenvalues of the linear combination and keeps the eigensystem unchanged. All applications of the Matrix Exponentiated Gradient Algorithm (Tsuda et al., 2005; Warmuth & Kuzmin, 2006a; Warmuth & Kuzmin, 2006b) have these properties and generalizations are straightforward to families of algorithms defined by other Bregman divergences.

In this paper we only considered bounds compared to the best fixed off-line comparator. However these online algorithms can be adapted to the case when the comparator shifts with time (See e.g. (Bousquet & Warmuth, 2002)). A thorough experimental analysis of these extensions would be useful.

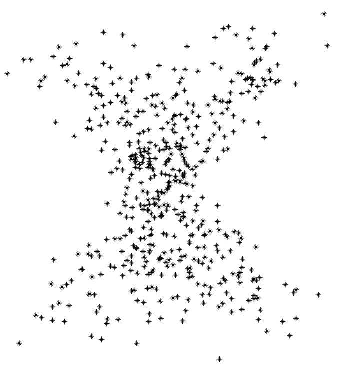

Figure 1. Our synthetic dataset is a sample of a 3-dimensional cone embedded in $\mathbb{R}^{20}$ with added Gaussian noise. We depict the key 3 of the 20 dimensions.

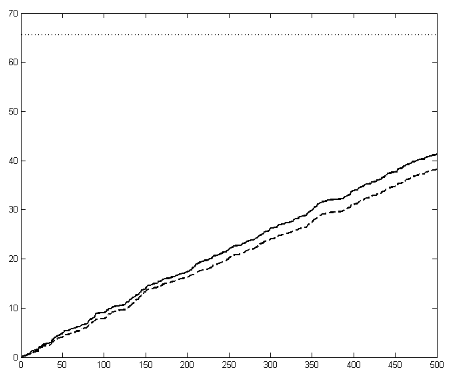

Figure 2. Plots of the total losses the algorithms and our bound versus the iteration number. Lowest (dashed) line: total loss of the Offline Kernel PCA Algorithm 2 (based on the entire dataset). Solid line right above: total loss of our Online Kernel PCA Algorithm with tuned learning rate (6) (for this dataset $\eta = 1.5$). Flat dotted line: tuned theoretical bound (7) on the total loss of the Online Kernel PCA Algorithm 6.

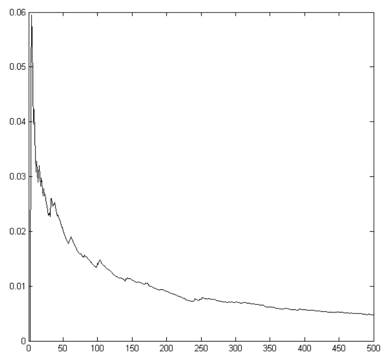

Figure 3. The average per-trial regret of the online algorithm converges to zero (Here η was set to 1).

References

Arora, S., & Kale, S. (2007). A combinatorial primal-dual approach to semidefinite programs. *Proc. 39th Annual ACM Symposium on Theory of Computing.* ACM. To appear.

Bousquet, O., & Warmuth, M. K. (2002). Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, *3*, 363–396.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization.* Cambridge University Press.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Herbster, M., & Warmuth, M. K. (2001). Tracking the best linear predictor. *Journal of Machine Learning Research*, *1*, 281–309.

Kivinen, J., & Warmuth, M. K. (1997). Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, *132*, 1–64.

Kivinen, J., & Warmuth, M. K. (1999). Averaging expert predictions. *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings* (pp. 153–167). Springer.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Takimoto, E., & Warmuth, M. K. (2003). Path kernels and multiplicative updates. *Journal of Machine Learning Research*, *4*, 773–818.

Tsuda, K., Rätsch, G., & Warmuth, M. K. (2005). Matrix exponentiated gradient updates for on-line learning and Bregman projections. *Journal of Machine Learning Research*, *6*, 995–1018.

Warmuth, M. K., & Kuzmin, D. (2006a). Online variance minimization. *Proceedings of the 19th Annual Conference on Learning Theory (COLT 06).* Pittsburg: Springer.

Warmuth, M. K., & Kuzmin, D. (2006b). Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. *Advances in Neural Information Processing Systems 19 (NIPS 06).* MIT Press.

Warmuth, M. K., & Vishwanathan, S. (2005). Leaving the span. *Proceedings of the 18th Annual Conference on Learning Theory (COLT 05).* Bertinoro, Italy: Springer. Journal version: http://www.cse.ucsc.edu/ manfred/pubs/span.pdf.