Journal of Machine Learning Research 9 (2008) 2287-2320 Submitted 9/07; Published 10/08

Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension*

Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Dima Kuzmin DIMA@CSE.UCSC.EDU
Computer Science Department
University of California - Santa Cruz
Santa Cruz, CA, 95064

Editor: John Shawe-Taylor

Abstract

We design an online algorithm for Principal Component Analysis. In each trial the current instance is centered and projected into a probabilistically chosen low dimensional subspace. The regret of our online algorithm, that is, the total expected quadratic compression loss of the online algorithm minus the total quadratic compression loss of the batch algorithm, is bounded by a term whose dependence on the dimension of the instances is only logarithmic.

We first develop our methodology in the expert setting of online learning by giving an algorithm for learning as well as the best subset of experts of a certain size. This algorithm is then lifted to the matrix setting where the subsets of experts correspond to subspaces. The algorithm represents the uncertainty over the best subspace as a density matrix whose eigenvalues are bounded. The running time is $O(n^2)$ per trial, where n is the dimension of the instances.

Keywords: principal component analysis, online learning, density matrix, expert setting, quantum Bayes rule

1. Introduction

In Principal Component Analysis (PCA) the n-dimensional data instances are projected into a k-dimensional subspace ($k < n$) so that the total quadratic compression loss is minimized. After centering the data, the problem is equivalent to finding the eigenvectors of the k largest eigenvalues of the data covariance matrix. The variance along an eigendirection is always equal to the corresponding eigenvalue and the subspace defined by the eigenvectors corresponding to the k largest eigenvalues is the subspace that captures the largest total variance and this is equivalent to minimizing the total quadratic compression loss.

We develop a probabilistic online version of PCA: in each trial the algorithm chooses a center m^{t-1} and a k-dimensional projection matrix P^{t-1} based on some internal parameter (which summarizes the information obtained from the previous $t-1$ trials); then an instance x^t is received and the algorithm incurs compression loss $\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2$; finally, the internal parameters are updated. The goal is to obtain online algorithms whose total compression loss in all trials is close to the total compression loss $\min_{m,P} \sum_{t=1}^{T} \|(x^t - m) - P(x^t - m)\|_2^2$ of the batch algorithm which can choose its center and k-dimensional subspace in hindsight based on all T instances. Specifically,

*. Supported by NSF grant IIS 0325363. A preliminary version of this paper appeared in Warmuth and Kuzmin (2006b).

in this paper we obtain randomized online algorithms with bounded *regret*. Here we define regret as the difference between the total expected compression loss of the randomized online algorithm and the compression loss of the best mean and subspace of rank k chosen offline. In other words the regret is essentially the expected additional compression loss incurred by the online algorithm compared to normal batch PCA. The expectation is over the internal randomization of the algorithm.

We begin by developing our online PCA algorithm for the uncentered case, that is, all $m^t = 0$ and the compression loss of the offline comparator is simplified to $\min_P \sum_{t=1}^{T} \|x^t - Px^t\|_2^2$. In this simpler case our algorithm is motivated by a related problem in the expert setting of online learning, where our goal is to perform as well as the best size k subset of experts. The algorithm maintains a mixture vector over the n experts. At the beginning of trial t the algorithm chooses a subset P^{t-1} of k experts based on the current mixture vector w^{t-1} that summarizes the previous $t-1$ trials. It then receives a loss vector $\ell^t \in [0,1]^n$. Now the subset P^{t-1} corresponds to the subspace onto which we "compress" or "project" the data. The algorithm incurs no loss on the k components of P^{t-1} and its compression loss equals the sum of the remaining $n-k$ components of the loss vector, that is, $\sum_{i \in \{1,\ldots,n\} - P^{t-1}} \ell_i^t$. Finally it updates its mixture vector to w^t.

The key insight is to maintain a mixture vector w^{t-1} as a parameter with the additional constraint that $w_i^{t-1} \leq \frac{1}{n-k}$. We will show that this "capped" mixture vector represents an implicit mixture over all subsets of experts of size $n-k$, and given w^{t-1} we can efficiently sample a subset of size $n-k$ from the implicit mixture and choose P^{t-1} as the complementary subset of size k. This gives an online algorithm whose total loss over all trials is close to the smallest $n-k$ components of the total loss vector $\sum_{t=1}^{T} \ell^t$. We will show how this algorithm generalizes to an online PCA algorithm when the mixture vector w^{t-1} is replaced by a density matrix W^{t-1} whose eigenvalues are capped by $\frac{1}{n-k}$. Now the constrained density matrix W^{t-1} represents an implicit mixture of $(n-k)$-dimensional subspaces. Again, we can efficiently sample from this mixture, and the complementary k-dimensional subspace P^{t-1} is used for projecting the current instance x_t at trial t.

A simple way to construct an online algorithm is to run the offline or batch algorithm on all data received so far and use the resulting hypothesis on the next data instance. This is called the "Incremental Offline Algorithm" (Azoury and Warmuth, 2001). When the offline algorithm just minimizes the loss on the past instances, then this algorithm is also called the "Follow the Leader (FL) Algorithm" (Kalai and Vempala, 2005). For uncentered PCA we can easily construct a sequence of instances for which the total online compression loss of FL is $\frac{n}{n-k}$ times larger than the total compression loss of batch PCA. However, in this paper we have a more stringent goal. We design randomized online algorithms whose total expected compression loss is at most one times the compression loss of batch PCA plus an additional lower order term which we optimize. In other words, we are seeking online algorithms with bounded *regret*. Our regret bounds are worst-case in that they hold for arbitrary sequences of instances.

Simple online algorithms such as the Generalized Hebbian Algorithm (Sanger, 1989) have been investigated previously that provably converge to the best offline solution. No worst-case regret bounds have been proven for these algorithms. More recently, the online PCA problem was also addressed in Crammer (2006). However, that paper does not fully capture the PCA problem because the presented algorithm uses a full-rank matrix as its hypothesis in each trial, whereas we use a probabilistically chosen projection matrix of the desired rank k. Furthermore, that paper proves bounds on the filtering loss, which are typically easier to obtain, and it is not clear how the filtering loss relates to the more standard regret bounds for the compression loss proven in this paper.

Our algorithm is unique in that we can prove a regret bound for it that is linear in the target dimension k of the subspace but logarithmic in the dimension of the instance space. The key methodology is to use a density matrix as the parameter and employ the quantum relative entropy as a regularizer and measure of progress. This was first done in Tsuda et al. (2005) for a generalization of linear regression to the case when the parameter matrix is a density matrix. Our update of the density matrix can be seen as a "soft" version of computing the top k eigenvectors and eigenvalues of the covariance matrix. It involves matrix logarithms and exponentials which are seemingly more complicated than the FL Algorithm which simply picks the top k directions. Actually, the most expensive step in both algorithms is to update the eigendecomposition of the covariance matrix after each new instance, and this costs $O(n^2)$ time (see, e.g., Gu and Eisenstat, 1994).

The paper is organized as follows. We begin by introducing some basics about batch and online PCA (Section 2) as well as the Hedge Algorithm from the expert setting of online learning (Section 3). We then develop a version of this algorithm that learns as well as the best subset of experts of fixed size (Section 4). When lifted to the matrix setting, this algorithm does uncentered PCA online (Section 5). Surprisingly, the regret bound for the matrix setting stays the same and this is an example of a phenomenon that has been dubbed the "free matrix lunch" (Warmuth, 2007b). We briefly discuss the merits of various alternate algorithms in sections 4.1 and 5.1.

Our online algorithm for centered online PCA is more involved since it has to learn the center as well (Section 6). After motivating the updates to the parameters (Section 6.1) we generalize our regret bound to the centered case (Section 6.2). We then briefly describe how to construct batch PCA algorithms from our online algorithms via standard conversion techniques (Section 6.3). Surprisingly, the bounds obtained this way are competitive with the best known batch PCA bounds. Lower bounds are discussed in Section 7. A brief experimental evaluation is given in Section 8 and we conclude with an overview of online algorithms for matrix parameters and discuss a number of open problems (Section 9).

2. Setup of Batch PCA and Online PCA

Given a set (or batch) of instance vectors $\{x^1, \ldots, x^T\}$, the goal of *batch PCA* is to find a low-dimensional approximation of this data that minimizes the quadratic compression loss. Specifically, we want to find a center vector $m \in \mathbb{R}^n$ and a rank k projection matrix[1] P such that the following loss function is minimized:

$$\text{comp}(P, m) = \sum_{t=1}^{T} \|(x^t - m) - P(x^t - m)\|_2^2. \tag{1}$$

Differentiating and solving for m gives us $m^* = \bar{x}$, where $\bar{x}$ is the data mean. Substituting this optimal center m^* into loss (1) we obtain

$$\begin{aligned} \text{comp}(P) &= \sum_{t=1}^{T} \|(I-P)(x^t - \bar{x})\|_2^2 = \sum_{t=1}^{T} (x^t - \bar{x})^\top (I-P)^2 (x^t - \bar{x}) \\ &= \text{tr}\Big((I-P)^2 \underbrace{\sum_{t=1}^{T} (x^t - \bar{x})(x^t - \bar{x})^\top}_{C} \Big). \end{aligned}$$

1. Projection matrices are symmetric matrices P with eigenvalues in $\{0, 1\}$. Note that $P^2 = P$.

The sum of outer products in the above trace is called the data covariance matrix C. Since $I-P$ is a projection matrix, $(I-P)^2 = I-P$, and

$$\mathrm{comp}(P) = \mathrm{tr}((\underbrace{I-P}_{\text{rank } n-k})C) = \mathrm{tr}(C) - \mathrm{tr}(\underbrace{P}_{\text{rank } k}C).$$

We call the above loss the *compression loss of P* or the *loss of subspace $I-P$*. We now give a justification for this choice of terminology. Observe that $\mathrm{tr}(C)$ equals $\mathrm{tr}(CP)+\mathrm{tr}(C(I-P))$, the sum of the losses of the complementary subspaces. However, we project the data into subspace P and the projected parts of the data are perfectly reconstructed. We charge the subspace P with the parts that are missed, that is, $\mathrm{tr}((I-P)C)$, and therefore call this the compression loss of P.

We now show that $\mathrm{tr}(PC)$ is maximized (or $\mathrm{tr}((I-P)C)$ minimized) if P consists of the k eigendirections of C with the largest eigenvalues. This proof might seem a digression, but elements of it will appear throughout the paper. By rewriting C in terms of its eigendecomposition, that is, $C = \sum_{i=1}^n \gamma_i\, c_i c_i^\top$, we can upper bound $\mathrm{tr}(PC)$ as follows:

$$\mathrm{tr}(PC) = \sum_{i=1}^n \gamma_i\, \mathrm{tr}(P\, c_i c_i^\top) = \sum_{i=1}^n \gamma_i\, c_i^\top P c_i \leq \max_{0\leq\delta_i\leq 1, \sum_i \delta_i = k} \sum_{i=1}^n \gamma_i\, \delta_i.$$

We can replace the scalars $c_i^\top P c_i$ in the ending inequality by the constrained δ_i's because of the following facts:

$$c_i^\top P c_i \leq 1, \text{ for } 1\leq i\leq n, \text{ and } \sum_{i=1}^n c_i^\top P c_i = \mathrm{tr}(P\underbrace{\sum_{i=1}^n c_i c_i^\top}_{I}) = \mathrm{tr}(P) = k,$$

since the eigenvectors c_i of C are an orthogonal set of n directions. A linear function is maximized at one of the vertices of its polytope of feasible solutions. The vertices of this polytope defined by the constraints $0\leq\delta_i\leq 1$ and $\sum_i \delta_i = k$ are those δ vectors with exactly k ones and $n-k$ zeros. Thus the vertices of the polytope correspond to sets of size k and

$$\mathrm{tr}(PC) \;\leq\; \max_{1\leq i_1<i_2<\ldots<i_k\leq n} \sum_{j=1}^k \gamma_{i_j}.$$

Clearly the set that gives the maximum upper bound corresponds to the largest k eigenvalues of C and $\mathrm{tr}(P^*C)$ equals the above upper bound when P^* consists of the eigenvectors corresponding to the set of k largest eigenvalues.

In the online setting, learning proceeds in trials. At trial t the algorithm chooses a center m^{t-1} and a rank k projection matrix P^{t-1}. It then receives an instance x^t and incurs loss

$$\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2 = \mathrm{tr}((I-P^{t-1})\,(x^t-m^{t-1})(x^t-m^{t-1})^\top).$$

Note that this is the compression loss of the center m^{t-1} and subspace P^{t-1} on the instance x^t. Our goal is to obtain an algorithm whose total online compression loss over the entire sequence of T trials $\sum_{t=1}^T \mathrm{tr}((I-P^{t-1})\,(x^t-m^{t-1})(x^t-m^{t-1})^\top)$ is close to the total compression loss (1) of the best center m^* and best rank k projection matrix P^* chosen in hindsight by the batch algorithm.

3. Learning as Well as the Best Expert with the Hedge Algorithm

The following setup and algorithm will be the basis of this paper. The algorithm maintains a probability distribution w^{t-1} over n experts. At the beginning of trial t it chooses an expert probabilistically according to the probability vector w^{t-1}, that is, expert i is chosen with probability w_i^{t-1}. Then a loss vector $\ell^t \in [0,1]^n$ is received, where ℓ_i^t specifies the loss of expert i incurred in trial t. The expected loss of the algorithm will be $w^{t-1} \cdot \ell^t$, since the expert was chosen probabilistically. At the end of the trial, the probability distribution is updated to w^t using exponential update factors (See Algorithm 1). This is essentially the Hedge Algorithm of Freund and Schapire (1997). In the

Algorithm 1 Hedge Algorithm

input: Initial n-dimensional probability vector w^0
for $t = 1$ to T **do**
  Draw an expert i with probability w_i^{t-1}
  Receive loss vector ℓ^t
  Incur loss ℓ_i^t
    and expected loss $w^{t-1} \cdot \ell^t$
  $w_i^t = \frac{w_i^{t-1} \exp(-\eta \ell_i^t)}{\sum_{j=1}^n w_j^{t-1} \exp(-\eta \ell_j^t)}$
end for

original version the algorithm proposes a distribution w^{t-1} at trial t and incurs loss $w^{t-1} \cdot \ell^t$ (instead of drawing an expert from w^{t-1} and incurring expected loss $w^{t-1} \cdot \ell^t$).

It is easy to prove the following bound on the total expected loss. Here $d(u,w)$ denotes the relative entropy between two probability vectors $d(u,w) = \sum_{i=1}^n u_i \log \frac{u_i}{w_i}$ and log is the natural logarithm.

Theorem 1 *For an arbitrary sequence of loss vectors $\ell^1, \ldots, \ell^T \in [0,1]^n$, the total expected loss of Algorithm 1 is bounded as follows:*

$$\sum_{t=1}^T w^{t-1} \cdot \ell^t \leq \frac{\eta \sum_{t=1}^T u \cdot \ell^t + \big(d(u,w^0) - d(u,w^T)\big)}{1 - \exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparison vector u in the n dimensional probability simplex.

Proof The update for w^t in Algorithm 1 is essentially the update of the Continuous Weighted Majority Algorithm where the absolute loss of expert i is replaced by ℓ_i^t. Since $\ell_i^t \in [0,1]$, we have $\exp(-\eta \ell_i^t) \leq 1 - (1 - \exp(-\eta))\ell_i^t$ and this implies (essentially Littlestone and Warmuth 1994, Lemma 5.2, or Freund and Schapire 1997):

$$-\log \sum_{i=1}^n w_i^{t-1} \exp(-\eta \ell_i^t) \geq -\log(1 - (1 - \exp(-\eta)) w^{t-1} \cdot \ell^t) \geq w^{t-1} \cdot \ell^t (1 - \exp(-\eta)).$$

The above can be reexpressed with relative entropies as follows (Kivinen and Warmuth, 1999):

$$\begin{aligned} d(u,w^{t-1}) - d(u,w^t) &= -\eta\, u \cdot \ell^t - \log \sum_{i=1}^n w_i^{t-1} \exp(-\eta \ell_i^t) \\ &\geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1 - \exp(-\eta)). \end{aligned} \tag{2}$$

The bound of theorem can now be obtained by summing over trials. ■

The original Weighted Majority algorithms were described for the absolute loss (Littlestone and Warmuth, 1994). The idea of using loss vectors instead was introduced in Freund and Schapire (1997). The latter paper also shows that when $\sum_t u \cdot \ell^t \leq L$ and $d(u,w^0) - d(u,w^T) \leq D \leq \log n$, then with $\eta = \log(1+\sqrt{2D/L})$, we get the bound

$$\sum_t w^{t-1} \cdot \ell^t \leq \sum_t u \cdot \ell^t + \sqrt{2LD} + d(u,w_0) - d(u,w^T). \tag{3}$$

By setting u to be the vector with a single one identifying the best expert, we get the following bound on the regret of the algorithm (Again log denotes the natural logarithm.):

$$\text{total loss of alg.} - \text{total loss of best expert} \leq \sqrt{2\,(\text{total loss of best expert})\,\log n} + \log n.$$

4. Learning as Well as the Best Subset of Experts

Recall that projection matrices are symmetric positive definite matrices with eigenvalues in $\{0,1\}$. Thus a rank k projection matrix can be written as $P = \sum_{i=1}^{k} p_i p_i^\top$, where the p_i are the k orthonormal vectors forming the basis of the subspace. Assume for the moment that the eigenvectors are restricted to be standard basis vectors. Now a projection matrix becomes a diagonal matrix with k ones in the diagonal and $n-k$ zeros. Also, the trace of a product of such a diagonal projection matrix and any symmetric matrix specifying the loss becomes a dot product between the diagonals of both matrices. The diagonal of the symmetric matrix may be seen as a loss vector ℓ^t. Thus, in this simplified diagonal setting, our goal is to develop online algorithms whose total loss is close to the sum of the lowest $n-k$ components of total loss vector $\sum_{t=1}^{T} \ell^t$. Equivalently, we want to find the highest k components of the total loss vector and per our nomenclature the loss of the lowest $n-k$ components is the compression loss of the complementary highest k components.

For this problem, we will encode the subsets of size $n-k$ as probability vectors: we call $r \in [0,1]^n$ an $(n-k)$-*corner* if it has $n-k$ components fixed to $\frac{1}{n-k}$ and the remaining k components fixed to zero. The algorithm maintains a probability vector w^t as its parameter. At trial t it probabilistically chooses an $(n-k)$-corner r based on the current probability vector w^{t-1} (Details of how this is done will be given shortly). The set of k components missed by r is the set P^{t-1} that we compress with at trial t. The algorithm then receives a loss vector ℓ^t and incurs compression loss $(n-k)\,r \cdot \ell^t = \sum_{i \in \{1,\ldots,n\} - P^{t-1}} \ell_i^t$. Finally the weight vector w^{t-1} is updated to w^t.

We now describe how the corner is chosen: The current probability vector is decomposed into a mixture of n corners and then one of the n corners is chosen probabilistically based on the mixture coefficients. In the description of the decomposition algorithm we use $d = n-k$ for convenience. Let A_d^n denote the convex hull of the $\binom{n}{d}$ corners of size d (where $1 \leq d < n$). Clearly, any component w_i of a vector w in the convex hull is at most $\frac{1}{d}$ because it is a convex combination of numbers in $\{0, \frac{1}{d}\}$. Therefore $A_d^n \subseteq B_d^n$, where B_d^n is the capped probability simplex, that is, the set of n-dimensional vectors w for which $|w| = \sum_i w_i = 1$ and $0 \leq w_i \leq \frac{1}{d}$, for all i. Figure 1 depicts the capped probability simplex for case $d = 2$ and $n = 3, 4$. The following theorem shows that the convex hull of the corners is exactly the capped probability simplex, that is, $A_d^n = B_d^n$. It shows this by expressing any probability vector in the capped simplex B_d^n as a convex combination of at most n $d-$corners. For example, when $d = 2$ and $n = 4$, B_d^n is an octahedron (which has 6 vertices). However, each point in this octahedron is contained in a tetrahedron which is the hull of only 4 of the 6 total vertices.

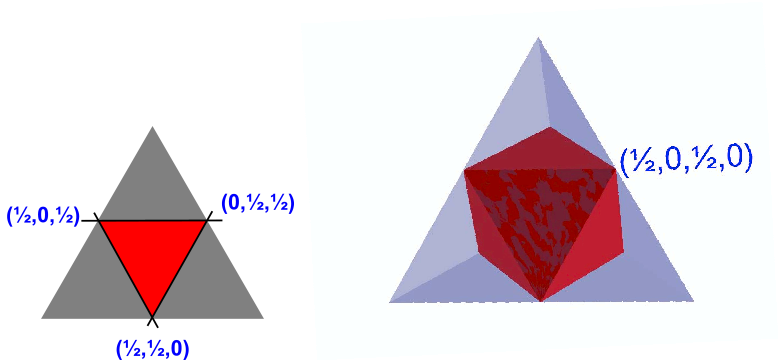

Figure 1: The capped probability simplex B_d^n, for $d = 2$ and $n = 3, 4$. This simplex is the intersection of n halfspaces (one per capped dimension) and its vertices are the $\binom{n}{d}$ d-corners.

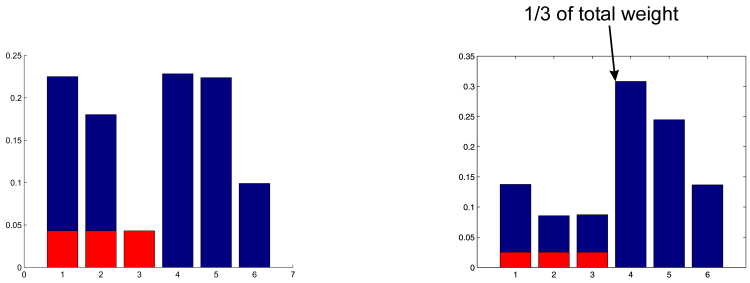

Figure 2: A step of the Mixture Decomposition Algorithm 2, $n = 6$ and $k = 3$. When a corner is removed, then at least one more component is set to zero or raised to a d-th fraction of the total weight. The left picture shows the case where a component inside the corner gets set to zero and the right one depicts the case where a component outside the picked corner gets d-th fraction of the total weight.

Theorem 2 *Algorithm 2 decomposes any probability vector w in the capped probability simplex B_d^n into a convex combination*[2] *of at most n d-corners.*

Proof Let $b(w)$ be the number of *boundary* components in w, that is, $b(w) = |\{i : w_i \text{ is } 0 \text{ or } \frac{|w|}{d}\}|$. Let $\widetilde{B}_d^n$ be all vectors w such that $0 \le w_i \le \frac{|w|}{d}$, for all i. If $b(w) = n$, then w is either a corner or 0. The loop stops when $w = 0$. If w is a corner then it takes one more iteration to arrive at 0. We show that if $w \in \widetilde{B}_d^n$ and w is neither a corner nor 0, then the successor $\widehat{w}$ lies in $\widetilde{B}_d^n$ and $b(\widehat{w}) > b(w)$. Clearly, $\widehat{w} \ge 0$, because the amount that is subtracted in the d components of the corner is at most as large as the corresponding components of w. We next show that $\widehat{w}_i \le \frac{|\widehat{w}|}{d}$. If i belongs to the corner that was chosen then $\widehat{w}_i = w_i - \frac{p}{d} \le \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. Otherwise $\widehat{w}_i = w_i \le l$, and $l \le \frac{|\widehat{w}|}{d}$ follows from the fact that $p \le |w| - d\,l$. This proves that $\widehat{w} \in \widetilde{B}_d^n$.

2. The existence of a convex combination of at most n corners is implied by Carathéodory's theorem (Rockafellar, 1970), but Algorithm 2 gives an effective construction.

Algorithm 2 Mixture Decomposition

input $1 \leq d < n$ and $w \in B_d^n$
repeat
Let r be a corner for a subset of d non-zero components of w
that includes all components of w equal to $\frac{|w|}{d}$
Let s be the smallest of the d chosen components of r
and l be the largest value of the remaining $n-d$ components
$w := w - \underbrace{\min(d\,s, |w| - d\,l)}_{p}\, r$ and **output** $p\,r$
until $w = 0$

For showing that $b(\widehat{w}) > b(w)$ first observe that all boundary components in w remain boundary components in $\widehat{w}$: zeros stay zeros and if $w_i = \frac{|w|}{d}$ then i is included in the corner and $\widehat{w}_i = \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. However, the number of boundary components is increased at least by one because the components corresponding to s and l are both non-boundary components in w and at least one of them becomes a boundary point in $\widehat{w}$: if $p = d\,s$ then the component corresponding to s in w is $s - \frac{p}{d} = 0$ in $\widehat{w}$, and if $p = |w| - d\,l$ then the component corresponding to l in w is $l = \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. It follows that it may take up to n iterations to arrive at a corner which has n boundary components and one more iteration to arrive at 0. Finally note that there is no weight vector $w \in \widetilde{B}_d^n$ s.t. $b(w) = n-1$ and therefore the size of the produced linear combination is at most n. More precisely, the size is at most $n - b(w)$ if $n - b(w) \leq n-2$ and one if w is a corner.

The algorithm produces a linear combination of $(n-k)$-corners, that is, $w = \sum_j p_j r_j$. Since $p_j \geq 0$ and all $|r_j| = 1$, $\sum_j p_j = 1$ and we actually have a convex combination. ■

It is easy to implement the Mixture Decomposition Algorithm in $O(n^2)$ time: simply sort w and spend $O(n)$ per loop.

The batch algorithm for the set problem simply picks the best set in a greedy fashion.

Fact 1 *For any loss vector ℓ, the following corner has the smallest loss of any convex combination of corners in $A_d^n = B_d^n$: Greedily pick the component of minimum loss (d times).*

How can we use the above mixture decomposition and fact to construct an online algorithm? It seems too hard to maintain information about all $\binom{n}{n-k}$ corners of size $n-k$. However, the best corner is also the best convex combination of corners, that is, the best from the set A_{n-k}^n where each member of this set is given by $\binom{n}{n-k}$ coefficients. Luckily, this set of convex combinations equals the capped probability simplex B_{n-k}^n and it takes only n coefficients to specify a member in B_{n-k}^n. Therefore we can maintain a parameter vector in B_{n-k}^n and for any such capped vector w, Algorithm 2 decomposes it into a convex combination of at most n many $(n-k)$-corners. This means that any algorithm producing a hypothesis vector in B_{n-k}^n can be converted to an efficient algorithm that probabilistically chooses an $(n-k)$-corner.

Algorithm 3 spells out the details for this approach. The algorithm chooses a corner probabilistically and $(n-k)\,w^{t-1} \cdot \ell^t$ is the expected loss at trial t. After updating the weight vector w^{t-1} by multiplying with the factors $\exp(-\eta \ell_i^t)$ and renormalizing, the resulting weight vector $\widehat{w}^t$ might lie outside of the capped probability simplex B_{n-k}^n. We then use a Bregman projection with the relative

Algorithm 3 Capped Hedge Algorithm

input: $1 \leq k < n$ and an initial probability vector $w^0 \in B^n_{n-k}$
for $t = 1$ to T **do**
  Decompose w^{t-1} into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
    by applying Algorithm 2 with $d = n-k$
  Draw a corner $r = r_j$ with probability p_j
  Let P^{t-1} be the k components outside of the drawn corner r
  Receive loss vector ℓ^t
  Incur compression loss $(n-k)\, r \cdot \ell^t = \sum_{i \in \{1,\ldots,n\} \setminus P^{t-1}} \ell^t_i$
    and expected compression loss $(n-k)\, w^{t-1} \cdot \ell^t$
  Update: $\widehat{w}^t_i = \dfrac{w^{t-1}_i \exp(-\eta \ell^t_i)}{\sum_{j=1}^n \exp(-\eta \ell^t_j)}$
    $w^t = \mathrm{cap}_{n-k}(\widehat{w}^t)$ where $\mathrm{cap}_{n-k}(.)$ invokes Algorithm 4
end for

Algorithm 4 Capping Algorithm

input probability vector w, set size d
Let $w^{\downarrow}$ index the vector in decreasing order, that is, $w^{\downarrow}_1 = \max(w)$
if $\max(w) \leq \frac{1}{d}$ **then**
  return w
end if
$i = 1$
repeat
  (* Set first i largest components to $\frac{1}{d}$ and normalize the rest to $\frac{d-i}{d}$ *)
  $\widetilde{w} = w$
  $\widetilde{w}^{\downarrow}_j = \frac{1}{d}$, for $j = 1 \ldots i$
  $\widetilde{w}^{\downarrow}_j := \frac{d-i}{d} \frac{\widetilde{w}^{\downarrow}_j}{\sum_{l=i+1}^n \widetilde{w}^{\downarrow}_l}$, for $j = i+1 \ldots n$
  $i := i+1$
until $\max(\widetilde{w}) \leq \frac{1}{d}$
return $\widetilde{w}$

entropy as the divergence to project the intermediate vector $\widehat{w}^t$ back into B^n_{n-k}:

$$w^t = \operatorname*{argmin}_{w \in B^n_{n-k}} d(w, \widehat{w}^t).$$

This projection can be achieved as follows (Herbster and Warmuth, 2001): find the smallest i s.t. capping the largest i components to $\frac{1}{n-k}$ and rescaling the remaining $n-i$ weights to total weight $1 - \frac{i}{n-k}$ makes none of the rescaled weights go above $\frac{1}{n-k}$. The simplest algorithm starts with sorting the weights and then searches for i (see Algorithm 4). However, a linear time algorithm is given in Herbster and Warmuth (2001)[3] that recursively uses the median.

3. The linear time algorithm of Figure 3 of that paper bounds the weights from below. It is easy to adapt this algorithm to the case of bounding the weights from above (as needed here).

When $k = n-1$ and $d = n-k = 1$, B_1^n is the entire probability simplex. In this case the call to Algorithm 2 and the projection onto B_1^n are vacuous and we get the standard Hedge Algorithm (Algorithm 1) as a degenerate case. Note that $(n-k)\sum_{t=1}^T u \cdot \ell^t$ is the total compression loss of comparator vector u. When u is an $(n-k)$-corner, that is, the uniform distribution on a set of size $n-k$, then $(n-k)\sum_{t=1}^T u \cdot \ell^t$ is the total loss of this set.

Theorem 3 *For an arbitrary sequence of loss vectors $\ell^1, \ldots, \ell^T \in [0,1]^n$, the total expected compression loss of Algorithm 3 is bounded as follows:*

$$(n-k)\sum_{t=1}^T w^{t-1} \cdot \ell^t \leq \frac{\eta(n-k)\sum_{t=1}^T u \cdot \ell^t + (n-k)(d(u,w^0) - d(u,w^T))}{1-\exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparison vector $u \in B_{n-k}^n$.

Proof The update for $\widehat{w}^t$ in Algorithm 3 is the same as update for w^t in Algorithm 1. Therefore we can use inequality (2):

$$d(u,w^{t-1}) - d(u,\widehat{w}^t) \geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1-\exp(-\eta)).$$

Since the relative entropy is a Bregman divergence (Bregman, 1967; Censor and Lent, 1981), the weight vector w^t is a Bregman projection of vector $\widehat{w}^t$ onto the convex set B_{n-k}^n. For such projections the Generalized Pythagorean Theorem holds (see, e.g., Herbster and Warmuth, 2001, for details):

$$d(u,\widehat{w}^t) \geq d(u,w^t) + d(w^t,\widehat{w}^t).$$

Since Bregman divergences are non-negative, we can drop the $d(w^t,\widehat{w}^t)$ term and get the following inequality:

$$d(u,\widehat{w}^t) - d(u,w^t) \geq 0, \text{ for } u \in B_{n-k}^n.$$

Adding this to the previous inequality we get:

$$d(u,w^{t-1}) - d(u,w^t) \geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1-\exp(-\eta)).$$

By summing over t, multiplying by $n-k$, and dividing by $1-\exp(-\eta)$, the bound follows. ■

It is easy to see that $(n-k)(d(u,w^0) - d(u,w^T)) \leq (n-k)\log\frac{n}{n-k}$ and this is bounded by $k\log\frac{n}{k}$ when $k \leq n/2$. By tuning η as in (3), we get the following regret bound:

$$\begin{aligned}&\text{(expected total compression loss of alg.) - (total compression loss of best } k\text{-subset)}\\ &\overset{k\leq n/2}{\leq} \sqrt{2(\text{total compression loss of best } k\text{-subset})\,k\log\frac{n}{k}} + k\log\frac{n}{k}. \qquad (4)\end{aligned}$$

The last inequality follows from the fact that $(n-k)\log\frac{n}{n-k} \leq k\log\frac{n}{k}$ when $k \leq n/2$. Note that the dependence on k in the last regret bound is essentially linear and dependence on n is logarithmic.

4.1 Alternate Algorithms for Learning as Well as the Best Subset

The question is whether projections onto the capped probability simplex are really needed. We could simply have one expert for each set of $n-k$ components and run Hedge on the $\binom{n}{n-k}$ set experts, where the loss of a set expert is always the sum of the $n-k$ component losses. The set expert $\{i_1,\ldots,i_{n-k}\}$ receives weight proportional to $\exp(-\sum_{j=1}^{n-k}\ell_{i_j}^{<t}) = \prod_{j=1}^{n-k}\exp(-\ell_{i_j}^{<t})$, where $\ell_q^{<t} = \sum_{p=1}^{t}\ell_q^p$. These product weights can be maintained implicitly: keep one weight per component where the ith component receives weight $\exp(-\ell_i^{<t})$, and use dynamic programming for summing the produced weights over the $\binom{n}{n-k}$ sets and for choosing a random set expert based on the product weights. See, for example, Takimoto and Warmuth (2003) for this type of method. While this dynamic programming algorithm can be made reasonably efficient ($O(n^2(n-k))$ per trial), the range of the losses of the set experts is now $[0,\mathbf{n}-\mathbf{k}]$ and this introduces factors of $\mathbf{n}-\mathbf{k}$ into the tuned regret bound:

$$\sqrt{2(\text{total compression loss of best } k\text{-subset})\,(\mathbf{n}-\mathbf{k})\,k\log\frac{n}{k}} + (\mathbf{n}-\mathbf{k})\,k\log\frac{n}{k}. \tag{5}$$

Curiously enough our new capping trick avoids these additional factors in the regret bound by using only the original n experts whose loss is in [0,1]. We do not know whether the improved regret bound (4) (i.e., no additional $n-k$ factors) also holds for the sketched dynamic programming algorithm. However, the following example shows that the two algorithms produce qualitatively different distributions on the sets.

Assume $n=3$ and $k=1$ and the update factors $\exp(-\eta\ell_i^{<t})$ for experts 1, 2 and 3 are proportional to 1, 2, and 4, respectively, which results in the normalized weight vector $(\frac{1}{7},\frac{2}{7},\frac{4}{7})$. Capping the weights at $\frac{1}{n-k}=\frac{1}{2}$ with Algorithm 4 produces the following vector which is then decomposed via Algorithm 2:

$$(\frac{1}{6},\frac{1}{3},\frac{1}{2}) \;= \frac{1}{3}\underbrace{(\frac{1}{2},0,\frac{1}{2})}_{\text{set }\{1,3\}} + \frac{2}{3}\underbrace{(0,\frac{1}{2},\frac{1}{2})}_{\text{set }\{2,3\}}. \tag{6}$$

On the other hand the product weights $\exp(-\eta\ell_i^{<t})*\exp(-\eta\ell_j^{<t})$ of the dynamic programming algorithm for the three sets $\{1,2\}$, $\{1,3\}$ and $\{2,3\}$ of size 2 are $1*2$, $1*4$, and $2*4$, respectively. That is, the dynamic programming algorithm gives (normalized) probability $\frac{1}{7}$, $\frac{2}{7}$ and $\frac{4}{7}$ to the three sets. Notice that Capped Hedge gives expert 3 probability 1 (since it is included in all corners of the decomposition (6)) and the dynamic programming algorithm gives expert 3 probability $\frac{6}{7}$, the total probability it has assigned to the two sets $\{1,3\}$ and $\{2,3\}$ that contain expert 3.

A second alternate is the Follow the Perturbed Leader (FPL) Algorithm (Kalai and Vempala, 2005). This algorithm adds random perturbations to the losses of the individual experts and then selects the set of minimum perturbed loss as its hypothesis. The algorithm is very efficient since it only has to find the set with minimum perturbed loss. However its regret bound has additional factors in addition to the $\mathbf{n}-\mathbf{k}$ factors appearing in the above bound (5) for the dynamic programming algorithm. For the original Randomized Hedge setting with just n experts (Section 3), a distribution of perturbations was found for which FPL simulates the Hedge exactly (Kalai, 2005; Kuzmin and Warmuth, 2005) and therefore the additional factors can be avoided. However we don't know whether there is a distribution of additive perturbations for which FPL simulates Hedge with set experts.

5. Uncentered Online PCA

We create an online PCA algorithm by lifting our new algorithm for sets of experts based on capped weight vector to the matrix case. Now *matrix corners* are density matrices[4] with d eigenvalues equal to $\frac{1}{d}$ and the rest are 0. Such matrix corners are just rank d projection matrices scaled by $\frac{1}{d}$. (Notice that the number of matrix corners is uncountably infinite.) We define the set $\mathcal{A}_d^n$ as the convex hull of all matrix corners. The maximum eigenvalue of a convex combination of symmetric matrices is at most as large as the maximum eigenvalue of any of the individual matrices (see, e.g., Bhatia, 1997, Corollary III.2.2). Therefore each convex combination of corners is a density matrix whose eigenvalues are bounded by $\frac{1}{d}$ and $\mathcal{A}_d^n \subseteq \mathcal{B}_d^n$, where $\mathcal{B}_d^n$ consists of all density matrices whose maximum eigenvalue is at most $\frac{1}{d}$. Assume we have some density matrix $W \in \mathcal{B}_d^n$ with eigendecomposition $\mathcal{W}\,\mathrm{diag}(\omega)\mathcal{W}^\top$. Algorithm 2 can be applied to the vector of eigenvalues ω of this density matrix. The algorithm decomposes ω into at most n diagonal corners r_j: $\omega = \sum_j p_j r_j$. This convex combination can be turned into a convex combination of matrix corners that decomposes the density matrix: $W = \sum_j p_j \mathcal{W}\,\mathrm{diag}(r_j)\mathcal{W}^\top$. It follows that $\mathcal{A}_d^n = \mathcal{B}_d^n$, as in the diagonal case.

As discussed before, losses can always be viewed in two different ways: the loss of the algorithm at trial t is the compression loss of the chosen projection matrix P^{t-1} or the loss of the complementary subspace $I - P^{t-1}$, that is,

$$\|\underbrace{P^{t-1}}_{\text{rank } k} x^t - x^t\|_2^2 = \mathrm{tr}(\underbrace{(I - P^{t-1})}_{\text{rank } n-k} x^t (x^t)^\top).$$

Our online PCA Algorithm 5 has uncertainty about which subspace of rank $n-k$ is best and it represents this uncertainty by a density matrix $W^{t-1} \in \mathcal{A}_{n-k}^n$, that is, a mixture of $(n-k)$-dimensional matrix corners. The algorithm efficiently samples a subspace of rank $n-k$ from this mixture and uses the complementary subspace P^{t-1} of rank k for compression. The expected compression loss of algorithm will be $(n-k)\mathrm{tr}(W^{t-1}xx^\top)$.

The following lemma shows how to pick the best matrix corner. When $S = \sum_{t=1}^T x^t (x^t)^\top$, then this lemma justifies the choice of the batch PCA algorithm.

Theorem 4 *For any symmetric matrix S, $\min_{W \in \mathcal{B}_d^n} \mathrm{tr}(WS)$ attains its minimum at the matrix corner formed by choosing d orthogonal eigenvectors of S of minimum eigenvalue.*

Proof Let $\lambda^\downarrow(W)$ denote the vector of eigenvalues of W in descending order and let $\lambda^\uparrow(S)$ be the same vector of S but in ascending order. Since both matrices are symmetric, $\mathrm{tr}(WS) \geq \lambda^\downarrow(W) \cdot \lambda^\uparrow(S)$ (Marshall and Olkin 1979, Fact H.1.h of Chapter 9, we will sketch a proof below). Since $\lambda^\downarrow(W) \in B_d^n$, the dot product is minimized and the inequality is tight when W is a d-corner (on the n-dimensional probability simplex) corresponding to the d smallest eigenvalues of S. Also the greedy algorithm finds the solution (see Fact 1 of this paper).

For the sake of completeness, we will sketch a proof of the inequality $\mathrm{tr}(WS) \geq \lambda^\downarrow(W) \cdot \lambda^\uparrow(S)$. We begin by rewriting the trace using an eigendecomposition of both matrices:

$$\mathrm{tr}(WS) = \mathrm{tr}(\sum_i \omega_i w_i w_i^\top \sum_j \sigma_j s_j s_j^\top) = \sum_{i,j} \omega_i \sigma_j \underbrace{(w_i \cdot s_j)^2}_{:=M_{i,j}}.$$

4. Density matrix is a symmetric positive definite matrix of trace 1, that is, they are symmetric matrices whose eigenvalues form a probability vector

The matrix M is *doubly stochastic*, that is, its entries are nonnegative and its rows and columns sum to 1. By Birkhoff's Theorem (see, e.g., Bhatia, 1997), such matrices are the convex combinations of permutations matrices (matrices with a single one in each row and column). Therefore the minimum of this linear function occurs at a permutation, and by a swapping argument one can show that the permutation which minimizes the linear function is the one that matches the ith smallest eigenvalue of W with the $(n-i)$th largest eigenvalue of S. ■

We obtain our algorithm for online PCA (Algorithm 5) by lifting Algorithm 3 for set experts to the matrix setting. The exponential factors used in the updates of the expert setting are replaced by the corresponding matrix version which employs the matrix exponential and matrix logarithm (Warmuth and Kuzmin, 2006a).[5] For any symmetric matrix A with eigendecomposition $\sum_{i=1}^{n} \alpha_i a_i a_i^\top$, the matrix exponential $\mathbf{exp}(A)$ is defined as the symmetric matrix $\sum_{i=1}^{n} \exp(\alpha_i) a_i a_i^\top$. Observe that the matrix exponential $\exp(A)$ (and analogously the matrix logarithm $\mathbf{log}(A)$ for symmetric positive definite A) affects only the eigenvalues and not the eigenvectors of A.

The following theorem shows that for the Bregman projection we can keep the eigensystem fixed. Here the quantum relative entropy $\Delta(U,W) = \mathrm{tr}(U(\mathbf{log}\,U - \mathbf{log}\,W))$ is used as the Bregman divergence.

Theorem 5 *Projecting a density matrix onto $\mathcal{B}_d^n$ w.r.t. the quantum relative entropy is equivalent to projecting the vector of eigenvalues w.r.t. the "normal" relative entropy: If W has the eigendecomposition $\mathcal{W}\,\mathrm{diag}(\omega)\,\mathcal{W}^\top$, then*

$$\operatorname*{argmin}_{U \in \mathcal{B}_d^n} \Delta(U,W) \;=\; \mathcal{W} u^* \mathcal{W}^\top, \text{ where } u^* = \operatorname*{argmin}_{u \in B_d^n} d(u,\omega).$$

Proof The quantum relative entropy can be rewritten as follows:

$$\Delta(U,W) = \mathrm{tr}(U\,\mathbf{log}\,U) - \mathrm{tr}(U\,\mathbf{log}\,W) = \lambda(U)\cdot\log(\lambda(U)) - \mathrm{tr}(U\,\mathbf{log}\,W),$$

where $\lambda(U)$ denotes the vector of eigenvalues of U and log is the componentwise logarithm of a vector. For any symmetric matrices S and T, $\mathrm{tr}(ST) \le \lambda^{\downarrow}(S)\cdot\lambda^{\downarrow}(T)$ (Marshall and Olkin 1979, Fact H.1.g of Chapter 9; also see proof sketch of a similar fact given in previous theorem). This implies that

$$\Delta(U,W) \ge \lambda(U)\cdot\log(\lambda(U)) - \lambda^{\downarrow}(U)\cdot\lambda^{\downarrow}(\mathbf{log}(W)) = \lambda(U)\cdot\log(\lambda(U)) - \lambda^{\downarrow}(U)\cdot\log\lambda^{\downarrow}(W).$$

Therefore $\min_{U \in \mathcal{B}_d^n} \Delta(U,W) \ge \min_{u \in B_d^n} d(u,\omega)$, and if u^* minimizes the r.h.s. then $\mathcal{W}\,\mathrm{diag}(u^*)\,\mathcal{W}^\top$ minimizes the l.h.s. because $\Delta(\mathcal{W}\,\mathrm{diag}(u^*)\,\mathcal{W}, W) = d(u^*,\omega)$. ■

The lemma means that the projection of a density matrix onto $\mathcal{B}_{n-k}^n$ is achieved by applying Algorithm 4 to the vector of eigenvalues of the density matrix.

We are now ready to prove a worst-case loss bound for Algorithm 5 for the uncentered case of online PCA. Note that the expected loss in trial t of this algorithm is $(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top)$. When U is a matrix corner then $(n-k)\sum_{t=1}^{T}\mathrm{tr}(Ux^t(x^t)^\top)$ is the total loss of the corresponding subspace.

5. This update step is a special case of the Matrix Exponentiated Gradient update for the the linear loss $\mathrm{tr}(Wx^t(x^t)^\top)$ (Tsuda et al., 2005).

Algorithm 5 Uncentered online PCA algorithm

input: $1 \leq k < n$ and an initial density matrix $W^0 \in \mathcal{B}^n_{n-k}$
for $t = 1$ to T **do**
  Perform eigendecomposition $W^{t-1} = \mathcal{W}\omega\mathcal{W}^\top$
  Decompose ω into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
    by applying Algorithm 2 with $d = n-k$
  Draw a corner $r = r_j$ with probability p_j
  Form a matrix corner $R = \mathcal{W}\,\mathrm{diag}(r)\,\mathcal{W}^\top$
  Form a rank k projection matrix $P^{t-1} = I - (n-k)R$
  Receive data instance vector x^t
  Incur compression loss $\|x^t - P^{t-1}x^t\|_2^2 = \mathrm{tr}((I - P^{t-1})\,x^t(x^t)^\top)$
    and expected compression loss $(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top)$
  Update: $\widehat{W}^t = \dfrac{\mathbf{exp}(\mathbf{log}\,W^{t-1} - \eta\,x^t(x^t)^\top)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^{t-1} - \eta\,x^t(x^t)^\top))}$
    $W^t = \mathrm{cap}_{n-k}(\widehat{W}^t)$,
      where $\mathrm{cap}_{n-k}(A)$ applies Algorithm 4 to the vector of eigenvalues of A
end for

Theorem 6 *For an arbitrary sequence of data instances $x^1, \ldots, x^T$ of 2-norm at most one, the total expected compression loss of the algorithm is bounded as follows:*

$$\sum_{t=1}^{T}(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top) \leq \frac{\eta(n-k)\sum_{t=1}^{T}\mathrm{tr}(Ux^t(x^t)^\top) + (n-k)(\Delta(U,W^0) - \Delta(U,W^T))}{1-\exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparator density matrix $U \in \mathcal{B}^n_{n-k}$.

Proof The update for $\widehat{W}^t$ is a density matrix version of the Hedge update which was used for variance minimization along a single direction (i.e., $k = n-1$) in Warmuth and Kuzmin (2006a). The basic inequality (2) for that update becomes:

$$\Delta(U,W^{t-1}) - \Delta(U,\widehat{W}^t) \geq -\eta\,\mathrm{tr}(Ux^t(x^t)^\top) + \mathrm{tr}(W^{t-1}x^t(x^t)^\top)(1-\exp(-\eta)).$$

As in the proof of Theorem 3 of this paper, the Generalized Pythagorean Theorem applies and dropping one term we get the following inequality:

$$\Delta(U,\widehat{W}^t) - \Delta(U,W^t) \geq 0, \text{ for } U \in \mathcal{B}^n_{n-k}.$$

Adding this to the previous inequality we get:

$$\Delta(U,W^{t-1}) - \Delta(U,W^t) \geq -\eta\,\mathrm{tr}(Ux^t(x^t)^\top) + \mathrm{tr}(W^{t-1}x^t(x^t)^\top)(1-\exp(-\eta)).$$

By summing over t, multiplying by $n-k$, and dividing by $1-\exp(-\eta)$, the bound follows. ■

It is easy to see that $(n-k)(\Delta(U,W^0)-\Delta(U,W^T)) \leq (n-k)\log\frac{n}{n-k}$ and this is bounded by $k\log\frac{n}{k}$ when $k \leq n/2$. By tuning η as in (3), we can get regret bounds of the form:

$$\begin{aligned}&\text{(expected total compression loss of alg.) - (total compression loss of best } k\text{-subspace)}\\ &\overset{k\leq n/2}{\leq} \sqrt{2(\text{total compression loss of best } k\text{-subspace})\,k\log\frac{n}{k}} + k\log\frac{n}{k}. \end{aligned} \tag{7}$$

Let us complete this section by discussing the minimal assumptions on the loss functions needed for proving the regret bounds obtained so far. Recall that in the regret bounds for experts as well as set experts we always assumed that the loss vector ℓ^t received at trial t lies in $[0,1]^n$. In the case of uncentered PCA, the loss at trial t is specified by an instance vector x^t that has 2-norm at most one. In other words, the single eigenvalue of the instance matrix $x^t(x^t)^\top$ must be bounded by 1. However, it is easy to see that the regret bound of the previous theorem still holds if at trial t the instance matrix $x^t(x^t)^\top$ is replaced by any symmetric instance matrix S^t whose vector of eigenvalues lies in $[0,1]^n$.

5.1 Alternate Algorithms for Uncentered Online PCA

We conjecture that the following algorithm has the regret bound (7) as well: run the dynamic programming algorithm for the set experts sketched in Section 4 on the vector of eigenvalues of the current covariance matrix. The produced set for size k is converted to a projection matrix of rank k by replacing it with the k outer products of the corresponding eigenvectors. We are not elaborating on this approach since the algorithm inherits the additional $n-k$ factors contained in the regret bound (5) for set experts. If these factors in the regret bound for set experts can be eliminated then this approach might lead to a competitive algorithm.

Versions of FPL might also be used to design an online PCA algorithm for compressing with a k dimensional subspace. Such an algorithm would be particularly useful if the same regret bound (7) could be proven for it as for our online PCA algorithm. The question is whether there exists a distribution of additive perturbations of the covariance matrix for which the loss of the subspace formed by the eigenvectors of the $n-k$ smallest eigenvalues simulates a matrix version of Hedge on subspaces of rank $n-k$ and whether this algorithm does not have the $n-k$ factors in its bound. Note that extracting the subspace formed by the eigenvectors of the $n-k$ smallest (or k largest) eigenvalues might be more efficient than performing a full eigendecomposition.

6. Centered Online PCA

In this section we extend our online PCA algorithm to also estimate the data center online. Under the extended protocol, the algorithm needs to produce both a rank k projection matrix P^{t-1} and a data center m^{t-1} at trial t. It then receives a data point x^t and incurs compression loss $\|(x^t-m^{t-1})-P^{t-1}(x^t-m^{t-1})\|_2^2$. As for uncentered online PCA, we will use a capped density matrix W^{t-1} to represent the algorithm's uncertainty about the hidden subspace.

6.1 Motivation of the Updates

We begin by motivating the updates of all the algorithms analyzed so far. We follow Kivinen and Warmuth (1997) and motivate the updates by minimizing a tradeoff between a parameter divergence and a loss function. Here we also have the linear capping constraints. Since our loss is linear, the

tradeoff minimization problem can be solved exactly instead of using approximations as is done in Kivinen and Warmuth (1997) for non-linear losses. Updates motivated by exact solution of tradeoff minimization problems involving non-linear loss functions are sometimes called *implicit* updates since they typically do not have a closed form (Kivinen et al., 2005). Even though the loss function used here is linear, the additional capping constraints are responsible for the fact that there is again no closed form for the updates. Nevertheless our algorithms are always able to compute the optimal solutions of the tradeoff minimization problems defining the updates.

We begin our discussion of motivations of updates with the set expert case. Consider the following two updates:

$$w^t = \text{arginf}_{w \in B^n_{n-k}} \left(\eta^{-1} d(w, w^{t-1}) + w \cdot \ell^t \right), \tag{8}$$

$$w^t = \text{arginf}_{w \in B^n_{n-k}} \left(\eta^{-1} d(w, w^0) + \sum_{q=1}^{t} w \cdot \ell^q \right). \tag{9}$$

In the motivations of all our updates, the divergences are always versions of relative entropies which are special cases of Bregman divergences. Here d denotes the standard relative entropy between probability vectors. The first update above trades off the divergence to the last parameter vector with the loss in the last trial. The second update trades off the divergence to the initial parameter with the total loss in all past trials. In both cases the minimization is over B^n_{n-k} which as we recall is the n-dimensional probability simplex with the components capped at $\frac{1}{n-k}$. One can show that the combined two update steps of the Capped Hedge Algorithm 3 coincide with the first update (8) above. The solution to (8) has the following exponential form:

$$w^t_i = \frac{w^{t-1}_i \exp(-\eta \ell^t_i + \gamma^t_i)}{\sum_{j=1}^n w^{t-1}_j \exp(-\eta \ell^t_j + \gamma^t_j)},$$

where γ^t_i is the Lagrangian coefficient that enforces the cap on the weight w^t_i. The non-negativity constraints don't have to be explicitly enforced because the relative entropy is undefined on vectors with negative elements and thus acts as a barrier function. Because of the capping constraints, the two updates (8) and (9) given above are typically not the same. However when $k = n-1$, then $B^n_{n-k} = B^n_1$ is the entire probability simplex and the γ^t_i coefficients disappear. In that case both updates agree and motivate the update of vanilla Hedge (Algorithm 1) (See Kivinen and Warmuth, 1999).

Furthermore, the above update (8) can be split into two steps as is done in Algorithm 3: the first update step uses exponential factors to update the probability vector and the second step performs a relative entropy projection of the intermediate vector onto the capped probability simplex. Here we give the sequence of two optimization problems that motivate the two update steps of Algorithm 3:

$$\widehat{w}^t = \text{arginf}_{w_i \geq 0,\, \sum w_i = 1} \left(\eta^{-1} d(w, w^{t-1}) + w \cdot \ell^t \right),$$

$$w_t = \text{arginf}_{w \in B^n_{n-k}} d(w, \widehat{w}^t).$$

For the motivation of the uncentered online PCA update (Algorithm 5), we replace the relative entropy $d(w, w^{t-1})$ between probability vectors in (8) by the Quantum Relative Entropy $\Delta(W, W^{t-1}) = \text{tr}(W(\mathbf{log}\, W - \mathbf{log}\, W^{t-1}))$ between density matrices. Furthermore, we change the loss function from a dot product to a trace:

$$W^t = \text{arginf}_{W \in \mathcal{D}^n_{n-k}} \left(\eta^{-1} \Delta(W, W^{t-1}) + \text{tr}(W x^t (x^t)^\top) \right).$$

Recall that $\mathcal{B}^n_{n-k}$ is the set of all $n \times n$ density matrices whose maximum eigenvalue is at most $\frac{1}{n-k}$. Note that in Algorithm 5, this update is again split into two steps.

The case of centered online PCA, which we will address now, is the most interesting because now we have two parameters. We use the following update which uses a divergence to the initial parameters (as in (9)):

$$(W^t, m^t) \;=\; \operatorname*{arginf}_{W \in \mathcal{B}^n_{n-k},\, m \in \mathbb{R}^n} \quad \Big(\eta^{-1}\Delta(W, W^0) + \widetilde{\eta}^{-1}(m-m^0)^\top W (m-m^0)\Big) + \sum_{q=1}^{t} \operatorname{tr}(W(x^q-m)(x^q-m)^\top). \tag{10}$$

Notice that we have two learning rates: η for the density matrix parameter and $\widetilde{\eta}$ for the center parameter. The above update may be viewed as a maximum a posteriori estimator since the divergences act as priors or initial examples and the inverse learning rates that multiply the divergences determine the importance of the priors (See, e.g., Azoury and Warmuth, 2001, for a discussion). When $\eta^{-1} = \widetilde{\eta}^{-1} = 0$, then there are no priors and the update become the Maximum Likelihood estimator or Follow the Leader (FL) Algorithm. If $\widetilde{\eta}^{-1} \to \infty$, then m^t is clamped to the fixed center m^0. If further $m^0 = 0$, then the above motivation becomes a motivation for an uncentered update with a divergence to the initial density matrix W^0 (analogous to (9)). Similarly, when $\eta^{-1} \to \infty$, then W^t is clamped to the fixed density matrix W^0 and the resulting optimization problem motivates the Incremental Off-line Algorithm for Gaussian density estimation with a fixed covariance matrix (Azoury and Warmuth, 2001).

As in Kuzmin and Warmuth (2007), we analyze this update for centered PCA by rewriting its optimization problem (10) as the dual maximization problem. The constraint $W \in \mathcal{B}^n_{n-k}$ in equation (10) is equivalent to having constraints $\operatorname{tr}(W) = 1$ and $W \preceq \frac{1}{n-k} I$. The constraint $W \succeq 0$ is automatically enforced since the quantum relative entropy acts as a barrier. With this in mind, we write down the Lagrangian function, where $U^t(W, m)$ is the objective function of our optimization problem (10) that includes data points from t trials, δ is the dual variable for the trace constraint and the symmetric positive definite matrix Γ is the dual variable for the capping constraint:

$$L^t(W, m, \Gamma, \delta) \;=\; U^t(W, m) + \delta(\operatorname{tr}(W) - 1) + \operatorname{tr}\Big((W - \frac{1}{n-k} I)\Gamma\Big).$$

The optimization over m is unconstrained, giving the solution for m^t:

$$m^t = \frac{\widetilde{\eta}^{-1} m^0 + \sum_{q=1}^t x^q}{\widetilde{\eta}^{-1} + t}. \tag{11}$$

This is essentially the normal mean of an extended sample, where we added $\widetilde{\eta}^{-1}$ copies of m^0 to $x^1, \ldots, x^t$. To write down the form of the solution for W^t compactly we will introduce the following matrix:

$$C^t = \widetilde{\eta}^{-1}(m^0 - m^t)(m^0 - m^t)^\top + \sum_{q=1}^{t} (x^q - m^t)(x^q - m^t)^\top. \tag{12}$$

This can be seen as the extended sample covariance matrix where we added $\widetilde{\eta}^{-1}$ copies of instance m^0.

Setting the derivatives to zero and solving (see Tsuda et al. 2005 for similar derivation), we obtain the following form of W^t in terms of the dual variables $\delta' = \eta\delta$ and Γ:

$$W^t(\delta', \Gamma) = \mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \delta' I - \eta\Gamma).$$

The constraint $\mathrm{tr}(W) = 1$ is enforced by choosing $\delta' = \log \mathrm{tr}(\mathbf{exp}(\mathbf{log} W_1 - \eta C^t - \Gamma))$. By substituting $W^t(\delta', \Gamma)$ and the formula for m^t into the Lagrangian L^t and simplifying, we obtain the following dual problem:

$$\max_{\Gamma \succeq 0} \widehat{L}^t(\Gamma), \text{ where } \widehat{L}^t(\Gamma) = -\eta^{-1} \log \mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma)) - \frac{\mathrm{tr}(\Gamma)}{n-k}. \tag{13}$$

Let Γ^t be the optimal solution of the dual problem above and let $\mathrm{cap}_d(W)$ be the density matrix obtained when the capping Algorithm 4 is applied to the vector of eigenvalues of W and capping parameter d. This lets us express W^t as:

$$W^t = \frac{\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma^t))} = \mathrm{cap}_{n-k}\Big(\frac{\mathbf{exp}(\mathbf{log} W^0 - \eta C^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t))}\Big). \tag{14}$$

For the analysis we express m^t and C^t as online updates:

Lemma 7 *The estimates of mean and covariance can be updated as follows:*

$$m^t = \frac{(\widetilde{\eta}^{-1} + t - 1)m^{t-1} + x^t}{\widetilde{\eta}^{-1} + t} = m^{t-1} - \frac{1}{\widetilde{\eta}^{-1} + t}(m^{t-1} - x^t),$$

$$C^t = C^{t-1} + \frac{\widetilde{\eta}^{-1} + t - 1}{\widetilde{\eta}^{-1} + t}(x^t - m^{t-1})(x^t - m^{t-1})^\top$$

Proof The update rule for m^t is easy to verify. For the update of C^t, we start by expanding the expression (12) for C^{t-1}:

$$\begin{aligned} C^{t-1} &= \widetilde{\eta}^{-1}(m^0(m^0)^\top - m^0(m^{t-1})^\top - m^{t-1}(m^0)^\top + m^{t-1}(m^{t-1})^\top) \\ &\quad + \sum_{q=1}^{t-1}(m^{t-1}(m^{t-1})^\top - x^q(m^{t-1})^\top - m^{t-1}(x^q)^\top + x^q(x^q)^\top) \\ &= \sum_{q=1}^{t-1} x^q(x^q)^\top + (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top \\ &\quad - (\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q)(m^{t-1})^\top - m^{t-1}(\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q)^\top + \widetilde{\eta}^{-1}m^0(m^0)^\top. \end{aligned}$$

By substituting

$$\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q = (\widetilde{\eta}^{-1} + t - 1)m^{t-1}$$

we get the following:

$$C^{t-1} = \sum_{q=1}^{t-1} x^q(x^q)^\top - (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top + \widetilde{\eta}^{-1}m^0(m^0)^\top.$$

Algorithm 6 Centered Online PCA Algorithm

input: $1 \leq k < n$ and an initial offset m^0, initial density matrix $W^0 \in \mathcal{B}^n_{n-k}$, $C^0 = 0$
for $t = 1$ to T **do**
Perform eigendecomposition $W^{t-1} = \mathcal{W}\omega\mathcal{W}^\top$
Decompose ω into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
by applying Algorithm 2 with $d = n-k$
Draw corner $r = r_j$ with probability p_j
Form a matrix corner $R = \mathcal{W}\,\mathrm{diag}(r)\,\mathcal{W}^\top$
Form a rank k projection matrix $P^{t-1} = I - (n-k)R$
Receive data instance vector x^t
Incur compression loss
$\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2 = \mathrm{tr}((I - P^{t-1})(x^t - m^{t-1})(x^t - m^{t-1})^\top)$
and expected compression loss $(n-k)\mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)$
Update:

$$m^t = m^{t-1} - \frac{1}{\widetilde{\eta}^{-1} + t}(m^{t-1} - x^t) \tag{15}$$

$$C^t = C^{t-1} + \frac{\widetilde{\eta}^{-1} + t - 1}{\widetilde{\eta}^{-1} + t}(x^t - m^{t-1})(x^t - m^{t-1})^\top \tag{16}$$

$$\widehat{W}^t = \frac{\mathbf{exp}(\mathbf{log}\, W^0 - \eta C^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log}\, W^0 - \eta C^t))}$$

$$W^t = \mathrm{cap}_{n-k}(\widehat{W}^t),$$

where $\mathrm{cap}_{n-k}(A)$ applies Algorithm 4 to the vector of eigenvalues of A
end for

Now the update for C can be written as:

$$C^t = C^{t-1} + (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top + x^t(x^t)^\top - (\widetilde{\eta}^{-1} + t)m^t(m^t)^\top.$$

Substituting the left update for m^t from the statement of the lemma and simplifying gives the desired online update for C^t. ■

: All the steps for the Centered Online PCA Algorithm are summarized as Algorithm 6. We already reasoned that the capping and decomposition steps are $O(n^2)$. The remaining expensive step is maintaining the eigendecomposition of the covariance matrix for computing the matrix exponential. Using standard rank one update techniques for the eigendecomposition of a symmetric matrix, this costs $O(n^2)$ per trial (see, e.g., Gu and Eisenstat, 1994).

6.2 Regret Bound for Centered PCA

The following theorem proves a regret bound for our Centered Online PCA Algorithm.

Theorem 8 *For any data sequence $x^1,\ldots,x^T$, initial center value m^0 such that $\|x^t - m^0\|_2 \le \frac{1}{2}$, any density matrix $U \in \mathcal{B}^n_{n-k}$ and any center vector m, the following bound holds:*

$$comp_{alg} \le \frac{\eta\, comp_{U,m} + \Delta(U,W^0) + \eta\widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0)}{1-\exp(-\eta)} + 1 + \log\big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\big)$$

where

$$comp_{alg} = \sum_{i=1}^{T} \mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)$$

is the overall expected compression loss of the centered online PCA Algorithm 6 and

$$comp_{U,m} = \sum_{i=1}^{T} \mathrm{tr}(U(x^t - m)(x^t - m)^\top)$$

is the total compression loss of comparison parameters (U,m).

Proof There are two main proof methods for the expert setting. The first is based on Bregman projections and was used so far in this paper. The second uses the value of the optimization problem defining the update as a potential and then shows that the drop of this value (Kivinen and Warmuth, 1999; Cesa-Bianchiand and Lugosi, 2006) is lower bounded by a constant times the per trial loss of the algorithm. Here we use a refinement of the second method that expresses the value of the optimization problem in terms of its dual. These variations of the second method were developed in the context of boosting (Warmuth et al., 2006; Liao, 2007) and in the conference paper (Kuzmin and Warmuth, 2007) where we enhanced the Uncentered Online PCA Algorithm of this paper with a kernel.

For our problem the value[6] of optimization problem (10) is $v^t = U^t(W^t, m^t)$ and this equals the value of the dual problem $\widehat{L}^t(\Gamma^t)$ where Γ^t maximizes the dual problem (13).

We want to establish the following key inequality:

$$v^t - v^{t-1} \;\ge\; \eta^{-1}(1-e^{-\eta})\Big(\mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top) - \frac{1}{\widetilde{\eta}^{-1}+t}\Big). \tag{17}$$

Since Γ^t optimizes the dual function $\widehat{L}^t$ and Γ^{t-1} is a non-optimal choice, $\widehat{L}^t(\Gamma^t) \ge \widehat{L}^t(\Gamma^{t-1})$ and therefore

$$v^t - v^{t-1} \;=\; \widehat{L}^t(\Gamma^t) - \widehat{L}^{t-1}(\Gamma^{t-1}) \;\ge\; \widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) \tag{18}$$

Substituting $\widehat{L}^t$ and $\widehat{L}^{t-1}$ from (13) into the right hand side of this inequality gives the following:

$$\begin{aligned}
&\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) \\
&\quad = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^t - \eta\Gamma^{t-1})) + \eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1})) \\
&\quad = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^t - \eta\Gamma^{t-1} - \log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1}))).
\end{aligned}$$

Now we expand C^t and use the covariance matrix update from Lemma 7:

$$\begin{aligned}
\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(&\log W^0 - \eta C^{t-1} - \eta\Gamma^{t-1} \\
&- \log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1})) - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t - m^{t-1})(x^t - m^{t-1})^\top)).
\end{aligned}$$

6. Optimization problem (10) minimizes a convex function subject to linear cone constraint. Since this problem has a strictly feasible solution, strong duality is implied by a generalized Slater condition (Boyd and Vandenberghe, 2004).

The first four terms under the matrix exponential form $\log W^{t-1}$, which can be seen from the first expression for W^{t-1} from (14):

$$\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) = -\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big).$$

Going back to (18) we get the inequality:

$$v^t - v^{t-1} \geq -\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big).$$

This expression for the drop of the value is essentially the same expression that is normally bounded in the proof of online variance minimization algorithm in Warmuth and Kuzmin (2006a). Using those techniques (assumption in the theorem implies that that $\|x^t - m^{t-1}\|_2^2 \leq 1$ and all the necessary inequalities hold) we get the following inequality:

$$-\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big) \geq \eta^{-1}\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(1-e^{-\eta})\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top).$$

W^{t-1} is a density matrix and its eigenvalues are at most 1. And by assumption, norm of $x^t - m^{t-1}$ is at most 1. Therefore, the loss $\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top)$ is also at most 1. We split the factor in front of the loss as $\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t} = 1 - \frac{1}{\widetilde{\eta}^{-1}+t}$, upper bounding the loss by 1 for the second part and leaving it as is for the first. With this (17) is obtained.

Note that the trace in the inequality (17) is the loss of the algorithm at trial t. Summation over t and telescoping gives us:

$$v^T - v^0 \geq \eta^{-1}(1-e^{-\eta})\Big(\mathrm{comp}_{\mathrm{alg}} - \sum_{t=1}^{T}\frac{1}{\widetilde{\eta}^{-1}+t}\Big).$$

We consider the left side first: v^0 is equal to zero, and v^T is a minimum of optimization problem (10), thus we can make it bigger by substituting arbitrary non-optimal values U and m. Index T means the optimization problem is defined with respect to the entire data sequence, therefore the loss term becomes the loss of the comparator. On the right side we use the following bound on the sum of generalized harmonic series: $\sum_{t=1}^{T}\frac{1}{\widetilde{\eta}^{-1}+t} \leq 1 + \log\Big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\Big)$. Overall, we get:

$$\eta^{-1}\Delta(U,W^0) + \widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0) + \mathrm{comp}_{U,m} \geq \eta^{-1}(1-e^{-\eta})\left(\mathrm{comp}_{\mathrm{alg}} - \Big(1+\log\big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\big)\Big)\right).$$

Moving things over and dividing results in the bound of the theorem. ■

As discussed before, when $\eta^{-1} = \widetilde{\eta}^{-1} = 0$, then the algorithm becomes the FL Algorithm. When $\widetilde{\eta}^{-1} \to \infty$, then m^t is clamped to m^0, that is, the update for the center (11,15) becomes $m^t = m^0$ and

is vacuous. Also in that case the term $\eta\widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0)$ in the upper bound of Theorem 8 is infinity unless the comparison center m is m^0 as well. If $m^0=0$ in addition to $\widetilde{\eta}^{-1}\to\infty$, then we call this the *uncentered version of Algorithm 6*: this version simply ignores step (15) and in (16) uses $m^{t-1}=0$. Our original Algorithm 5 for uncentered PCA as well as the uncentered version of Algorithm 6 have the same regret bound[7] of Theorem 6. Recall however that the two algorithms were motivated differently: Algorithm 5 trades off divergence to the last parameter with the loss in the last trial, whereas Algorithm 6 trades off a divergence to the initial parameter matrix with the total loss in all past trials. If all constraints are equality constraints, then the two algorithms are the same. However, capping introduces inequality constraints and therefore the two algorithms are decidedly not the same. Both algorithm can behave quite differently experimentally (Section 8). The difference between the two algorithms will become important in the followup paper (Kuzmin and Warmuth, 2007), where we were only able to use a kernel with the algorithm that trades off a divergence to the initial parameter matrix with the total loss in all past trials.

Similarly, when $\eta^{-1}\to\infty$, then W^t is clamped to W^0 and the algorithm degenerates to a previously analyzed algorithm, the Incremental Off-line Algorithm for Gaussian density estimation with fixed covariance matrix (Azoury and Warmuth, 2001). For this restricted density estimation problem, improved regret bounds were proven for the Forward Algorithm which further shrinks the estimate of the mean towards the initial mean. So far we were not able to improve our regret bound for uncentered PCA using additional shrinkage towards the initial mean.

The statement of the theorem requires strong initial knowledge about the center of the data sequence we are about to observe: the condition of the theorem says that our data sequence has to be contained in a ball of radius $\frac{1}{2}$ around m^0. This can be relaxed by using $m^0=0$ and $\widetilde{\eta}^{-1}=0$, which corresponds to using standard empirical mean for m^t. Now it suffices to assume that data is contained in some ball, but we are not required to know where exactly that ball is. The appropriate assumption and the change to the bound are detailed in the following corollary.

Corollary 9 *For any data sequence $x^1,\ldots,x^T$ that can be covered by a ball of radius $\frac{1}{2}$, that is, $\|x^{t_1}-x^{t_2}\|_2\le 1$ and that also has the bound on the norm of instances $\|x^t\|_2\le R$, any density matrix $U\in\mathcal{B}^n_{n-k}$ and any center vector m, the total expected loss of centered online PCA Algorithm 6 being used with parameters $\widetilde{\eta}^{-1}=0$ and $m^0=0$ is bounded as follows:*

$$comp_{alg}\le\frac{\eta\, comp_{U,m}+\Delta(U,W^0)}{1-e^{-\eta}}+\log T+R^2,$$

Proof The ball assumption means that the empirical mean m^{t-1} and any element of the data sequence are not too far from each other: $\|m^{t-1}-x^t\|_2\le 1$. Thus we can still use the Inequality (17), for all trials but the first one, where we haven't seen any data points yet. Summing the drops of the value starting from $t=1$ we get:

$$v^T-v^1\ge\eta^{-1}(1-e^{-\eta})\left(\sum_{t=2}^{T}\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top)-\sum_{t=2}^{T}\frac{1}{t}\right).$$

7. The remaining $+1$ is an artifact of our bound on the harmonic sum.

We now add the loss of the first trial into the sum and rearrange terms:

$$\overbrace{\sum_{t=1}^{T} \mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)}^{\mathrm{comp}_{\mathrm{alg}}}$$
$$\leq \frac{\eta(v^T - \overbrace{v^1}^{\geq 0})}{1-e^{-\eta}} + \overbrace{\sum_{t=2}^{T} \frac{1}{t}}^{\leq \log T} + \overbrace{\mathrm{tr}(W^0(x^1 - m^0)(x^1 - m^0)^\top)}^{\leq R^2}.$$

Finally, from the definition of v^T it follows that $v^T \leq \eta^{-1}\Delta(U, W^0) + \mathrm{comp}_{U,m}$, for any comparator U and m, and this gives the bound of the theorem. ■

Tuning η as in (3), Corollary 9 gives the following regret bound for our centered online PCA Algorithm 6 (when $k \leq \frac{n}{2}$):

(expected total compression loss of alg.) - (total comp. loss of best centered k-subspace)
$$\leq \sqrt{2(\text{total comp. loss of best centered } k\text{-subspace})\, k \log \frac{n}{k}} + k \log \frac{n}{k} + R^2 + \log T.$$

6.3 Converting the Online PCA Algorithms to Batch PCA Algorithms

In the online learning community a number of conversion techniques have been developed that allow one to construct a hypothesis with good generalization bounds in the batch setting from the hypotheses produced by a run of the online learning algorithm over the given batch of examples.

For example, using the standard conversion techniques developed for the expert setting based on the leave-one-out loss (Cesa-Bianchi et al., 1997), we obtain algorithms with good expected regret bounds in the following model: The algorithm is given $T-1$ instances drawn from a fixed but unknown distribution and produces a k-dimensional subspace based on those instances; it then receives a new instance from the same distribution. We can bound the expected loss on the new instance (under the usual norm less than one assumption on instances):

(expected compression loss of alg.) - (expected compression loss best k-space)
$$= O\Big(\sqrt{\frac{(\text{expected compression loss of best } k\text{-subspace})\, k \log \frac{n}{k}}{T}} + \frac{k \log \frac{n}{k}}{T}\Big).$$

The expected loss of the algorithm is taken as expectation over both the internal randomization of the algorithm and fixed distribution over the instances. The expected loss of the best subspace just averages over the distribution of the instances. The best subspace itself will be determined by the covariance matrix of this distribution.

Additionally, there also exist very general conversion methods that allow us to state bounds that say that the generalization error will be big with small probability (Cesa-Bianchi and Gentile, 2005). These bounds are more complicated and therefore we don't state them here. The conversion algorithms however, are pretty simple: for example, one can use the average density matrix of all density matrices produced by the online algorithm while doing one pass through the batch of instances. Perhaps surprisingly, the generalization bounds for batch PCA obtained via the online-to-batch conversions are competitive with the best bounds for batch PCA that we are aware of Shawe-Taylor et al. (2005).

7. Lower Bounds

We first prove some lower bounds for the simplest online algorithm that just predicts with the model that has incurred minimum loss so far (the Follow the Leader (FL) Algorithm). After that we give a lower bound for uncentered PCA that shows that the algorithm presented in this paper is optimal in a very strong sense.

Our first lower bound is in the standard expert setting. We assume that there is a deterministic tie-breaking rule, because by adding small perturbations, ties can always be avoided in this construction. It is easy to see that the following adversary strategy forces FL to have loss n times larger than the loss of the best expert chosen in hindsight: in each trial have the expert chosen by FL incur one unit of loss. Note that the algorithm incurs loss one in each trial, whereas the loss of the best expert is $\lfloor \frac{T}{n} \rfloor$ after T trials. We conclude that the loss of FL can be by a factor of n larger than the loss of the best expert.

We next show that for the set expert case, FL can be forced to have loss at least $\frac{n}{d}$ times the loss of the best set of size d. In this case FL chooses a set of size d of minimum loss and the adversary forces the lowest loss expert in the set chosen by FL to incur one unit of loss. The algorithm again incurs loss one in each trial, but the loss of the best set lies in the range $d\,[\lfloor \frac{T}{n} \rfloor, \lceil \frac{T}{n} \rceil]$. Thus in this case the loss of FL can be by a factor of $\frac{n}{d}$ larger than the loss of the best set of size d.

When rephrased i.t.o. compression losses, FL picks a set of size $n-d$ whose complementary set of size d has minimum compression loss. We just showed that the total compression loss of FL can be at least $\frac{n}{d}$ times the compression loss of the best subset of size $n-d$.

We can lift the above lower bound for sets to the case of uncentered PCA. Now $d = n-k$ and k is the rank of the subspace we want to compress onto. To simplify the argument, we let the first n instances be small multiples of the standard basis vectors. More precisely, $x_t = t\varepsilon\, e^t$, for $1 \leq t \leq n$ and small real ε. These instances cause the uncentered data covariance matrix $\sum_{t=1}^{n} x_t x_t^\top$ to be a diagonal matrix. Also, if ε is small enough then the loss in the first n trials is negligible. From now on FL always chooses a unique set of $d = n-k$ standard basis vectors of minimum loss and the adversary chooses a standard basis vector with the lowest loss in the set as the next instance. So the lower bound argument essentially reduces to the set case, and FL can be forced to have compression loss $\frac{n}{n-k}$ times the loss of the compression loss of the best k dimensional subspace.

So far we have shown that our online algorithms are better than the simplistic FL Algorithm since their compression losses are at most one times the loss of the best plus essentially a square root term. We now show that the constant in front of the square root term is rather tight as well. For the expert setting ($d = 1$) this was already done:

Theorem 10 *(Theorem C.3. of the journal version Helmbold and Warmuth 2008 of the conference paper Helmbold and Warmuth 2007.) For all $\varepsilon > 0$ there exists n_ε such that for any number of experts $n \geq n_\varepsilon$, there exists a $T_{\varepsilon,n}$ where for any number of trials $T \geq T_{\varepsilon,n}$ the following holds for any algorithm in the expert setting: there is a sequence of T trials with n experts for which the loss of the best expert is at most $T/2$ and the regret of the algorithm is at least $(1-\varepsilon)\sqrt{(T/2)\log n}$.*

In the expert model used in this paper, we follow Freund and Schapire (1997) and assume that the losses of the experts in each trial are specified by a loss vector in $[0,1]^N$. There is a related model (studied earlier), where the experts produce predictions in each trial. After receiving those predictions the algorithm produces its own prediction and receives a label. The loss of the experts

and algorithm is the absolute value of the difference between the predictions and the label, respectively (Littlestone and Warmuth, 1994). The above theorem actually holds for this model of online learning with the absolute loss (when all predictions are in $[0,1]$ and the labels are in $\{0,1\}$), and the model used in this paper may be seen as the special case where the prediction of the algorithm is formed by simply averaging the predictions of the experts (Freund and Schapire, 1997). Therefore any lower bound for the described expert model with absolute loss immediately holds for the expert model where the loss is specified by a loss vector.

Note that the regret bounds for the Hedge Algorithm discussed at the end of Section 3 have an additional factor of 2 in the square root term. By choosing a prediction function other than the weighted average, the factor of 2 can be avoided in the expert model with the absolute loss, and the upper and lower bounds for the regret have the same constant in front of the square root term (provided that N and T are large enough) (Cesa-Bianchi et al., 1997; Cesa-Bianchiand and Lugosi, 2006).

The above lower bound theorem immediately generalizes to the case of set experts. Partition the experts into d blocks of size $\frac{n}{d}$ (assume d divides n). For any algorithm and block, construct a sequence of length T as before. During the sequence for one block, the experts for all the other blocks have loss zero. The loss of the best set of size d on the whole sequence of length Td is at most $Td/2$ and the regret, that is, the loss of the algorithm on the sequence minus the loss of the best set of size d, is lower bounded by

$$(1-\varepsilon)d\sqrt{\frac{T}{2}\log\frac{n}{d}} = (1-\varepsilon)\sqrt{\frac{dT}{2}d\log\frac{n}{d}}.$$

Rewritten in terms of compression losses for compression sets of size k (i.e., $d = n-k$), the lower bound on the compression loss regret becomes

$$(1-\varepsilon)\sqrt{\text{compression loss of best } k\text{-subset } (n-k)\log\frac{n}{n-k}}. \tag{19}$$

Note that the upper bound (4) obtained by our algorithm for learning as well as the best subset is essentially a factor of $\sqrt{2}$ larger than this lower bound.

Finally, we lift the above lower bound for subsets to a lower bound for uncentered PCA. In the setup for uncentered PCA, the instance matrix at trial t is $S^t = x^t(x^t)^\top$, where x^t has 2-norm at most 1. For the lower bound we need the instance matrix S^t to be an arbitrary symmetric matrix with eigenvalues in $[0,1]$. As discussed before Section 5.1, the upper bound for uncentered PCA still holds for these more general instance matrices.

To lift the lower bound for subsets to uncentered PCA, we simply replace the loss vector ℓ^t by the instance matrix $S^t = \operatorname{diag}(\ell^t)$. At trial t, the PCA algorithm uses the density matrix W^{t-1} and incurs expected loss $\operatorname{tr}(W^{t-1}\operatorname{diag}(\ell^t)) = \operatorname{diag}(W^{t-1})\cdot\ell^t$. Note that the diagonal vector $\operatorname{diag}(W^{t-1})$ is a probability vector. Thus the PCA algorithm doesn't have any advantage from using non-diagonal density matrices and the lower bound reduces to the set case. We conclude that the lower bound (19) also holds for the compression loss regret of uncentered PCA algorithms when the instance matrices are allowed to be symmetric matrices with eigenvalues in $[0,1]$. Again the corresponding upper bound (7) is essentially a factor of $\sqrt{2}$ larger.

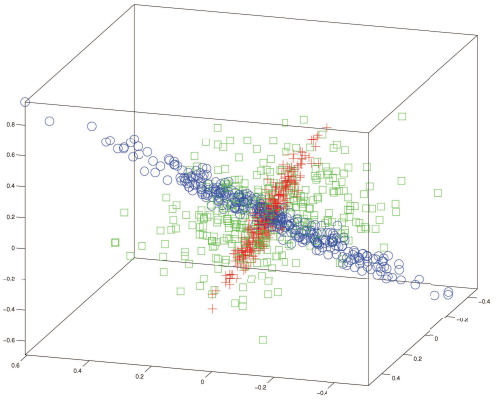

Figure 3: The data sequence used for the first experiment switches between three different subspaces. It is split into three segments. Within each segment, the data is drawn from a different 20-dimensional Gaussian with a rank 2 covariance matrix. We plot the first three coordinates of each data point. Different colors/symbols denote the data points that came from the three different subspaces.

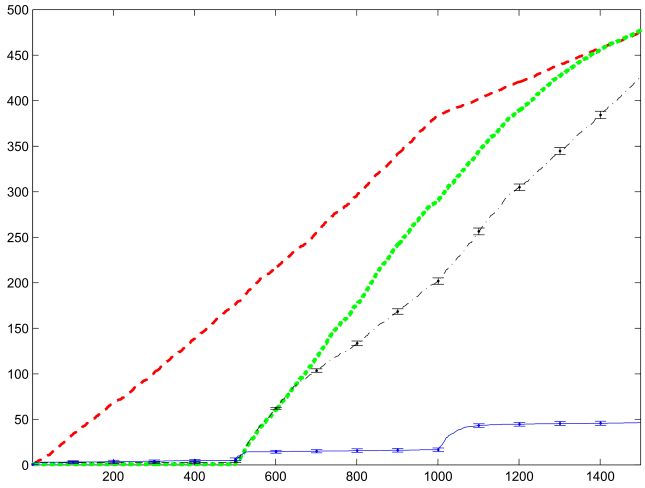

Figure 4: The blue/solid curve is the total loss of uncentered online PCA Algorithm 5 for the data sequence described in Figure 3 (with $n = 20, k = 2$ and $\eta = 1$). The algorithm uses internal randomization for choosing a subspace and therefore the curve is actually the average total loss over 50 runs for the same data sequence. The error bars (one standard deviation) indicate the variance of the algorithm. The black/dash-dotted curve plots the same for the uncentered version of Algorithm 6 (again $\eta = 1$). The visible bumps in the curves correspond to places in the data sequence where it shifts from one subspace to another. The red/dashed curve is the total loss of the best projection matrix determined in hindsight (i.e., loss of batch uncentered PCA). The green/dotted curve is the total loss of the Follow the Leader Algorithm.

8. Simple Experiments

The regret bounds we prove for our online PCA algorithms hold for arbitrary sequences of instances. In other words, they hold even if the instances are produced by an adversary which aims to make the algorithm have large regret. In many cases, natural data does not have a strong adversarial nature and even the simple Follow the Leader Algorithm might have small regret against the best subspace chosen in hindsight. However, natural data commonly shifts with time. It is on such time-changing data sets that online algorithms have an advantage. In this section we present some simple experiments that bring out the ability of our online algorithms to adapt to data that shifts with time.

For our first experiment we constructed a simple synthetic data set of time-changing nature. The data sequence is divided into three equal intervals, of 500 points each. Within each interval data points are picked at random from a multivariate Gaussian distribution on $\mathbb{R}^{20}$ with zero mean.

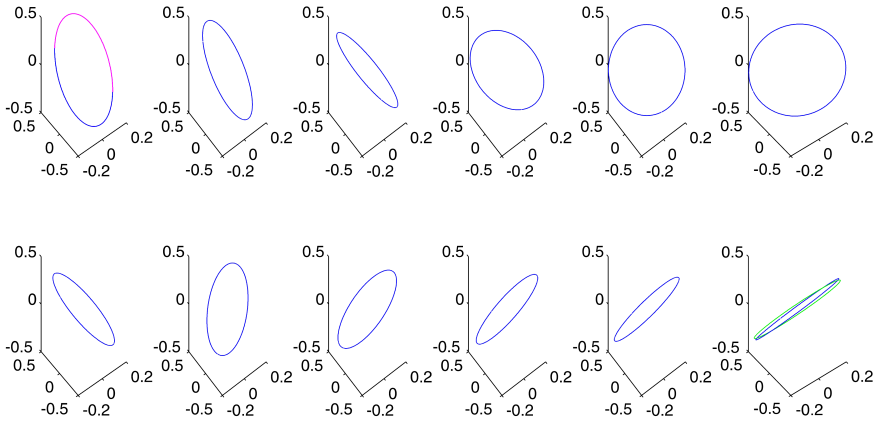

Figure 5: Behavior of the Uncentered Online PCA Algorithm 5 when data shifts from one subspace to another. First shift for one of the runs in Figure 4 is shown. We show the projection matrices that have the highest probability of being picked by the algorithm in a given trial. Since $k = 2$, each such matrix P^t can be seen as a 2-dimensional ellipse in $\mathbb{R}^{20}$: the ellipse is formed by points $P^t x$ for all $\|x\|_2 = 1$. We plot the first three coordinates of this ellipse. The transition sequence starts with the algorithm focused on the optimal projection matrix for the first subset of data and ends with essentially the optimal matrix for the second subset. The depicted transition takes about 60 trials and only every $5th$ trial is plotted.

The covariance matrices for the Gaussians were picked at random but constrained to rank 2, thus ensuring that the generated points lie in some 2-dimensional subspace. The generated points with norm bigger than one were normalized to 1. The data set is graphically represented in Figure 3, which plots the first three dimensions of each one of the data points. Different colors/symbols indicate data points that came from the three different subspaces.

In Figure 4 we plot the total compression loss for some of the algorithms introduced in this paper. For the sake of simplicity we restrict ourselves to uncentered PCA. Here the data dimension n is 20 and the subspace dimension k is 2. We plot the total loss of the following algorithms as a function of the trial number: the FL Algorithm, the original uncentered PCA Algorithm 5 and the uncentered version of Algorithm 6. For the latter two algorithms we need to select a learning rate. One possibility is to choose the learning rate that optimizes our upper bound on the regret of the algorithms (3). Since the bound is the same for both algorithms this choice of η is also the same: the choice depends on an upper bound on the compression loss of the batch algorithm. Plugging in the actual compression loss of the batch algorithm gives $\eta = 0.12$. In practice, heuristics can be used to tune η. For the experiment of Figure 4 we simply chose $\eta = 1$. Recall that our online PCA algorithms decompose their density matrix parameter W^t into a convex combination of projection matrices using the deterministic Algorithm 2. However, the PCA algorithms then randomly select one of the k dimensional projection matrices in the convex combination with probability equal to its coefficient. This introduces randomness into the execution of the algorithm even when run on a fixed data sequence. We run the algorithms 50 times and plot the average total loss as a function

of t for the fixed data sequence depicted in Figure 3. We also indicate the variance of this total loss with error bars of one standard deviation. Note again that the average and variance is w.r.t. internal randomization of the algorithm. The bumps in the loss curves of Figure 4 correspond to the places in the data sequence where it shifts from one subspace to another. When the loss curve of an algorithm flattens out then the algorithm has learned the correct subspace for the current segment. For example, Figure 5 depicts how the density matrix of the uncentered PCA Algorithm 5 ($\eta = 1$) transitions around a segment boundary.

The FL Algorithm (which coincides with the uncentered version of Algorithm 10 when $\eta \to \infty$) learns the first segment really quickly, but it does not recover during the later segments. The original uncentered PCA Algorithm 5 (with $\eta = 1$) recovers in each segment, whereas the uncentered version of Algorithm 6 (with $\eta = 1$) does not recover as quickly and has higher total loss on this data sequence. Recall that both online algorithms for uncentered PCA have the same regret bound against the best fixed offline comparator.

In Figure 4 we also plot the compression loss of the best subspace selected in hindsight by running batch uncentered PCA (dashed/red line). The data set we generated is not well approximated by a single 2-dimensional subspace, since it consists of three different 2-dimensional subspaces. As a result, the overall loss of the fixed subspace is higher than the loss of both of our uncentered online PCA algorithms that to a varying extent were able to switch between the different 2-dimensional subspaces.

There are many heuristics for detecting switches of the data. For example one could simply check for a performance loss of any algorithm in a suitably chosen final segment. However, such algorithms are often unwieldy and are hard to tune when the data in difference segments are not drastically different. Note that for the Uncentered Online PCA Algorithm presented in this paper we only have provable good regret bounds against the best fixed subspace chosen offline (based on the entire data sequence). (In Figure 4 both of our online algorithms beat the offline comparator and thus have negative regret against this simple comparator.) Ideally we would like measure regret against stronger offline comparators that can exploit the shifting nature of the data. In the expert setting there is a long line of research where the offline algorithm is allowed to partition the data sequence into s segments and pick the best subspace for each of the s segments. There are simple modifications of the online algorithm in the experts setting that have good regret bounds against the best partition of size s (see, e.g., Herbster and Warmuth, 1998). Naturally, the regret bounds grow with the number of segments s. The modifications that are needed to achieve these regret bounds are simple: insert an additional update step at the end of each trial that mixes a little bit of the uniform distribution into the weight vector. In the context of uncentered PCA this amounts to adding the following update at the end of each trial:

$$W^t = (1-\alpha)W^t + \alpha\frac{I}{n}.$$

We claim that it is easy to again lift the techniques developed in the expert setting to PCA and prove the analogous regret bounds against the best partition.

The following subtle modification of the mixing rule leads to algorithms with even more interesting properties. If the algorithm mixes in a little bit of the average of all past weight vectors instead of the uniform vector, then it is able to switch quickly to experts that have been good at some time in the past. This has been called the "long-term memory" property of such algorithms. In the context of uncentered PCA, this amounts the following additional update step for the density

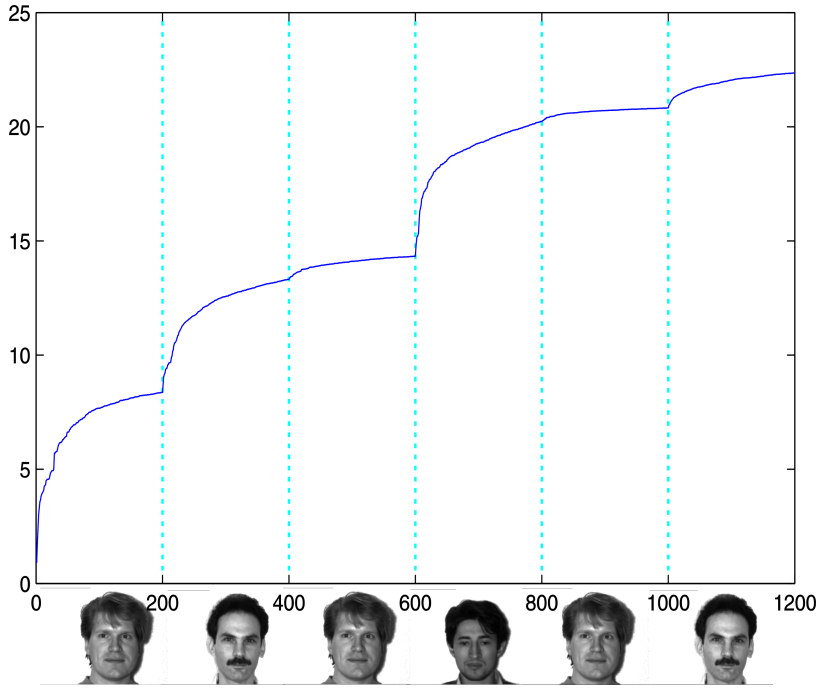

Figure 6: Long term memory effect when mixing in past average is added to the uncentered online PCA Algorithm 5. The data sequence is comprised of several segments, each one of two hundred randomly sampled images of the same person but with different facial expressions and lighting conditions. These segments are indicated with dotted lines and are labeled with the face of the person that was used to generate the segment. The plot depicts the regret of the uncentered online PCA Algorithm 5 ($\eta = 1$) with added mixing update (20) ($\alpha = .001$). The regret is w.r.t. the indicated partition of size six.

matrix:

$$W^t = (1-\alpha)W^t + \alpha\frac{\sum_{q=0}^{t-1} W^q}{t}. \tag{20}$$

We performed another experiment that demonstrates this "long-term memory" effect by adding update (20) to the uncentered online PCA Algorithm 5: For this experiment we used face image data from Yale-B data set. A segmented data sequence was constructed by sampling the face images, where each segment contained images of the same person, but with different facial expressions, lighting conditions, etc. Figure 6 plots the regret of our uncentered online PCA Algorithm 5 against the best partition of size 6 chosen offline ($\eta = 1, \alpha = .001$). In the picture the segment boundaries are indicated with dotted lines and the face below each segment shows the person who's pictures were sampled during that section. Note that our online algorithm doesn't have knowledge of segment boundaries. The first segment shows the typical behavior of online algorithms: the regret grows initially, but then flattens out once the algorithm converges to the best solution for that segment. Thus the "bump" in the regret curve is due to the fact that the algorithm has to learn a new segment of pictures from a different person. By comparing the bumps of different segments we see that they are significantly smaller in segments where the algorithm encounters pictures from a person that it

has previously seen: the algorithm "remembers" subspaces that were good in the past. These small bumps can be seen in segments 3,5 and 6 in our data set. For additional plots of long-term memory effects in the expert setting see Bousquet and Warmuth (2002). Our experiments are preliminary and we did not formally prove regret bounds for shifting comparators in PCA setting, but such bounds are straightforwardly obtained using the methodology from this paper and Bousquet and Warmuth (2002).

9. Conclusions

We developed a new set of techniques for learning as well as a low dimensional subspace. We first developed the algorithms in the expert case and then lifted these algorithms and bounds to the matrix case as essentially done in Tsuda et al. (2005); Warmuth and Kuzmin (2006a). The new insight is to represent our uncertainty over the subspace as a density matrix with capped eigenvalues. We show how to decompose such a matrix into a mixture of n subspaces of the desired rank. In the case of PCA the seemingly quadratic compression loss can be rewritten as a linear loss of our matrix parameter. Therefore a random subspace chosen from the mixture has the same expected loss as the loss of the parameter matrix.

Similar techniques (albeit not capping) have been used recently for learning permutations (Helmbold and Warmuth, 2008): Instead of maintaining a probability vector over the $n!$ permutations, the parameter used in that paper is a convex combination of permutation matrices which is an $n \times n$ doubly stochastic matrix. This matrix can be efficiently decomposed into a mixture of $O(n^2)$ permutation matrices. If the loss is linear, then a random permutation chosen from the mixture has the same expected loss as the loss of the doubly stochastic parameter matrix.

When the loss is convex and not linear (as for example the quadratic loss in the generalization of linear regression to matrix parameters Tsuda et al. 2005), then our new capping trick can still be applied to the density matrix parameter. In this case, the online loss of the capped density matrix can be shown to be close to the loss of the best low dimensional subspace. However the quadratic loss of the capped density matrix (which is a convex combination of subspaces) can be much smaller than the convex combination of the losses of the subspaces. The bounds of that paper only hold for the smaller loss of the capped density matrix and not for the expected loss of the subspace sampled from the convex combination represented by the capped density matrix.

Our new capping technique is interesting in its own right. Whereas the vanilla multiplicative update on n experts with exponential factors is prone to be unstable because it tends to go into a corner of the simplex (by putting all weight on the currently best expert), the technique of capping the weight at $\frac{1}{d}$ keeps a set of at least d experts alive. In some sense the capping has the same effect as a "super predator" has on a biological system: such a predator specializes on the most frequent species of prey, preventing the dominance of any particular species and thus preserving variety. See Warmuth (2007a) for a discussion of the relationships of our updates to biological systems.

Following Kivinen and Warmuth (1997); Helmbold et al. (1999), we motivate our updates by trading off a parameter divergence against a loss divergence. In our case, the parameter divergence is always a capped relative entropy and the loss divergence is simply linear. It would be interesting to apply the capping trick to the logistic loss used in logistic regression. The logistic loss can be seen as a relative entropy between the desired probabilities of the outcomes and the predicted probabilities of the outcomes obtained by applying a sigmoid function to the linear activations (Kivinen and Warmuth, 2001). For "capped logistic regression", linear constraints would be added to the op-

timization problem defining the updates that cap the predicted probabilities of the outcomes. This would lead to versions of logistic regression where the loss favors a small set of outcomes instead of a single outcome.

The algorithms of this paper belong to the family of multiplicative updates, that is, the parameters are updated by multiplicative factors of exponentials. In the matrix case the updates make use of the matrix log and matrix exponential. There is a second family of updates that does additive parameter updates. In particular, there are additive online updates for PCA (Crammer, 2006). The latter update family has the key advantage that they can be easily used with kernels. However, in a recent conference paper (Kuzmin and Warmuth, 2007) we also were able to give a special case where the multiplicative updates could be enhanced with a kernel. In this case the instance matrices are outer products $x^t(x^t)^\top$ and are replaced by $\phi(x)^t(\phi(x^t))^\top$. In particular, the online PCA algorithms of this paper can be "kernelized" this way.

In PCA the instance matrices are *symmetric* matrices $x^t(x^t)^\top$ and we seek a low rank *symmetric* subspace that approximates the instances well. In a recent conference paper we showed in a slightly different context how to generalize our methods to the case of "asymmetric" matrices of arbitrary shape. Using those techniques, the online PCA algorithms of this paper can be generalized to the asymmetric case: now the instance matrices are rank one $n \times m$ matrices of the form $x^t(\widetilde{x}^t)^\top$, where x and $\widetilde{x}$ are vectors of dimension n and m, respectively. In the asymmetric case, the underlying decomposition is the SVD decomposition instead of the eigendecomposition.

There are two technical open problems that arise in this paper. We gave a number of dynamic programming algorithms, such as the one given for the set expert problem. If the loss range for the individual experts is [0,1] then the loss range for the set experts is $[0,d]$ when d is the size of the set. The straightforward application of results for the Hedge Algorithm leads to extra factors of d in the regret bounds for set experts. We avoided these factors using our capping trick. However, the question is whether the same bounds (without the d factors) hold for the dynamic programming algorithm as well. There is a different fundamental algorithmic technique for dealing with structural domains: the Follow the Perturbed Leader Algorithm (Kalai and Vempala, 2005). The second open problem is how to adapt this algorithm to online PCA and prove bounds for it that don't contain the extra d factors.

Finally, independent of what theoretical progress might have been achieved in this paper, we still have to find a convincing experimental setting where our new online PCA algorithms are indispensable. The update time of our algorithms is $O(n^2)$ and further time improvements via approximations or lazy evaluation might be needed to make the algorithms widely applicable.

Acknowledgments

Thanks to Allen Van Gelder for valuable discussions re. Algorithm 2.

References

K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Journal of Machine Learning*, 43(3):211–246, June 2001. Special issue on *Theoretical Advances in On-line Learning, Game Theory and Boosting*, edited by Yoram Singer.

R. Bhatia. *Matrix Analysis*. Springer, Berlin, 1997.

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

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, July 1981.

N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. In *NIPS*, 2005.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

N. Cesa-Bianchiand and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

K. Crammer. Online tracking of linear subspaces. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 06)*, Pittsburg, June 2006. Springer.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1266–1276, October 1994.

D. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. In *Proceedings of the 20th Annual Conference on Learning Theory (COLT07)*. Springer, 2007.

D. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. Submitted journal version, August 2008.

D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, November 1999.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998. Earlier version in 12th ICML, 1995.

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

A. Kalai. Simulating weighted majority with FPL. Private communication, 2005.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005. Special issue Learning Theory 2003.

J. Kivinen and M. K. Warmuth. Averaging expert predictions. In *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 153–167. Springer, 1999.

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

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

J. Kivinen, M. K. Warmuth, and B. Hassibi. The *p*-norm generalization of the LMS algorithm for adaptive filtering. *Journal of IEEE Transactions on Signal Processing (to appear)*, 54(5): 1782–1793, May 2005.

D. Kuzmin and M. K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT 05)*, pages 684–686. Springer, June 2005. Open problem.

D. Kuzmin and M. K. Warmuth. Online Kernel PCA with entropic matrix updates. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*. ACM Press, June 2007.

J. Liao. *Totally Corrective Boosting Algorithms that Maximize the Margin*. PhD thesis, University of California at Santa Cruz, June 2007.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inform. Comput.*, 108(2): 212–261, 1994. Preliminary version in in FOCS 89.

A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 1979.

R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473, 1989.

J. Shawe-Taylor, C. K. I. Williams, N. Cristianini, and J. S. Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005. URL `http://dx.doi.org/10.1109/TIT.2005.850052`.

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

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

M. K. Warmuth. The blessing and the curse of the multiplicative updates. Work in progress, unpublished manuscript., 2007a.

M. K. Warmuth. When is there a free matrix lunch. In *Proc. of the 20th Annual Conference on Learning Theory (COLT 07)*. Springer, June 2007b. Open problem.

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

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

M. K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 1001–1008, New York, NY, USA, 2006. ACM Press.