Winnowing Subspaces

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

Abstract

We generalize the Winnow algorithm for learning disjunctions to learning subspaces of low rank. Subspaces are represented by symmetric projection matrices. The online algorithm maintains its uncertainty about the hidden low rank projection matrix as a symmetric positive definite matrix. This matrix is updated using a version of the Matrix Exponentiated Gradient algorithm that is based on matrix exponentials and matrix logarithms. As in the case of the Winnow algorithm, the bounds are logarithmic in the dimension n of the problem, but linear in the rank r of the hidden subspace. We show that the algorithm can be adapted to handle arbitrary matrices of any dimension via a reduction.

1. Introduction

Assume we want to label unit vectors as to whether they are "close to" a r dimensional subspace that is hidden from the learner. We will give a polynomial online algorithm that does this with a small number of prediction mistakes without ever producing an r dimensional subspace that determines the labels. Our algorithm is a matrix variant of the Winnow algorithm for learning disjunctions. In the noise-free case the number of mistakes of our algorithm is $O(r \log \frac{n}{r})$, where n is the dimension of the instances and r the rank of the hidden subspace. Whereas the original algorithm maintains a non-negative weight vector and is designed to learn as well as the best disjunction of size r, the new matrix version maintains a symmetric positive definite matrix as its parameter. This matrix summarizes the uncertainty about the hidden r dimensional subspace.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

The algorithm and its mistake bounds are presented in an online setting. Standard conversion algorithms lead to bounds for batch learning models when the examples are drawn independently at random according to a fixed distribution (See e.g. (Littlestone, 1989; Cesa-Bianchi & Gentile, 2006)).

The original Winnow algorithm uses a linear threshold function as its hypothesis. In the matrix version the dot product between vectors is replaced by the trace between matrices. In this conference paper we only describe the algorithm in its simplest deterministic form and only prove bounds for the noise-free case. The algorithm and bound retain the case of disjunctions as a special case.

We leave it to the reader to adapt the fancier algorithms and bounds developed for the disjunction case to the matrix case: versions that can handle shifted and scaled versions of the problem as well as noisy data, a randomized version that has an expected mistake bound that is half of the mistake bound of the deterministic algorithms and modifications of the algorithms that learn well when the subspace shifts over time (Auer & Warmuth, 1998). Finally, the algorithms can also be kernelized (as was done in (Kuzmin & Warmuth, 2007) for principal component analysis).

Our first algorithm, Symmetric Matrix Winnow, learns well when the labels indicate closeness to a low rank subspace in $\mathbb{R}^n$. A subspace is represented as a (symmetric) projection matrix $\boldsymbol{P} = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{u}_i^T$, where $\boldsymbol{u}_i$ are the r orthogonal directions defining the subspace. The algorithm maintains its uncertainty about the hidden projection matrix as an $n \times n$ dimensional symmetric positive definite matrix. It assumes that the instance matrices are symmetric $n \times n$ matrices with eigenvalues in $[0, 1]$. In Figure 3 we visualize the regions produced by thresholding a trace between a projection matrix and an instance matrix of the form $\boldsymbol{x}\boldsymbol{x}^\top$ where $\boldsymbol{x}$ is a unit vector (or direction). At the end of the paper we give a second algorithm that can handle the case of arbitrary $m \times n$ dimensional instance matrices (with bounded maximum singular value) that

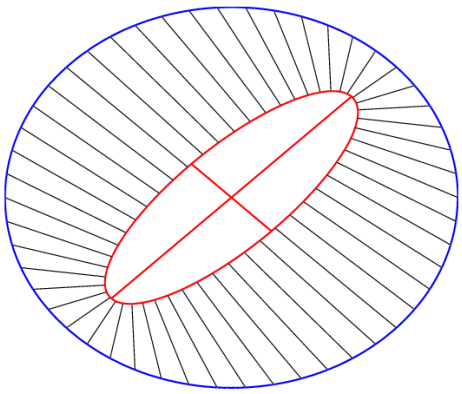

Figure 1: The symmetric matrix $\boldsymbol{A}$ is depicted as the inner ellipse: The curve of the ellipse is plot of vector $\boldsymbol{Au}$, where $\boldsymbol{u}$ is unit vector. The lines connect the corresponding points on the outer unit circle and the ellipse. The eigenvectors correspond to the axes of the ellipse.

are labeled whether they are close to rank r matrix of dimension $m \times n$ that has r non-zero singular values of value one. The generalization is achieved via a reduction from the symmetric case.

1.1. Related Work

There has been a number of papers where the exponentiated gradient algorithms (Kivinen & Warmuth, 1997) for vector parameters have been generalized to matrix parameters. Instead of maintaining a vector of non-negative numbers, the algorithm now maintains a symmetric positive definite matrix.[1] The linear regression case and a generalization to Boosting appears in (Tsuda et al., 2005) and the expert setting of on-learning has been "lifted" to the matrix case in (Warmuth & Kuzmin, 2006a; Arora & Kale, 2007). Also in (Warmuth & Kuzmin, 2006b; Kuzmin & Warmuth, 2007) the methodology was used to derive on-line algorithms for principal component analysis.

Note that in all of the above cases where symmetric positive definite matrices are used as a parameter, there is always an original problem where the parameter of the algorithm and the instances are vectors. The original problem is retained as a special case when the parameter matrix and instance matrices are restricted to be diagonal matrices. Curiously enough the bounds are always the same for the general symmetric matrix case as for the vector/diagonal matrix case. This phenomenon has been dubbed the "free matrix lunch" (Warmuth, 2007). In particular, our bound for learning the Close to Subspace problem is the same as the bounds for the original disjunction problem.

In this paper we "lift" the Winnow algorithm for learning disjunctions to the symmetric matrix case. This is particularly interesting because it lets us learn well when the data is close to a low rank subspace. Many algorithms have been proposed for learning low-rank matrices. However, with the notable exception of principal component analysis, generalization bounds are often hard to obtain for these algorithms. In this paper we address a very simple problem where the target is a subspace of low rank (i.e. all eigenvalues defining the low-rank object are the same). In this simple case we are able to prove bounds and show how they carry over to general matrices of arbitrary shape. As for principal component analysis, the bounds are linear in the rank of the subspace but logarithmic in the feature dimension.

2. Notation and Some Linear Algebra

A *dyad* is an outer product $\boldsymbol{uu}^\top$, where $\boldsymbol{u}$ is a unit vector (i.e. a direction). Any linear combination of dyads (with real coefficients) is a symmetric matrix and any symmetric matrix $\boldsymbol{A}$ of dimension n always has an eigendecomposition, i.e. it can be expressed as a linear combination of n orthogonal dyads: $\boldsymbol{A} = \sum_{i=1}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\top$, where the linear coefficients λ_i are the *real* eigenvalues and the $\boldsymbol{u}_i$ are the orthonormal eigenvectors. We depict a symmetric matrix as an ellipse which is an affine transformation of the unit ball (See Figure 1). A dyad is a degenerate ellipse that has exactly one eigenvector with eigenvalue one and all other eigenvalues are zero. In other words a dyad can be viewed as an axis of length two that goes through the origin at the half point. The parameter of our algorithm will be a symmetric positive definite matrix, i.e. now the λ_i must be non-negative.

Alternatively, we can view the symmetric positive definite matrix $\boldsymbol{A}$ as a covariance matrix of some random cost vector $\boldsymbol{c} \in \mathbb{R}^n$, i.e.

$$\boldsymbol{A} = \mathbb{E}\left((\boldsymbol{c} - \mathbb{E}(\boldsymbol{c})(\boldsymbol{c} - \mathbb{E}(\boldsymbol{c}))^\top\right)$$

The variance along any vector $\boldsymbol{u}$ is then

$$\begin{aligned}\mathbf{V}(\boldsymbol{c}^\top \boldsymbol{u}) &= \mathbb{E}(\ \left(\boldsymbol{c}^\top \boldsymbol{u} - \mathbb{E}(\boldsymbol{c}^\top \boldsymbol{u})\right)^2\ ) \\ &= \boldsymbol{u}^\top \underbrace{\mathbb{E}\left((\boldsymbol{c} - \mathbb{E}(\boldsymbol{c})(\boldsymbol{c} - \mathbb{E}(\boldsymbol{c}))^\top\right)}_{\boldsymbol{A}} \boldsymbol{u}.\end{aligned}$$

[1] Closely related is the case when the parameter vector is a *density matrix* (i.e. a symmetric positive definite matrix whose trace is normalized to one).

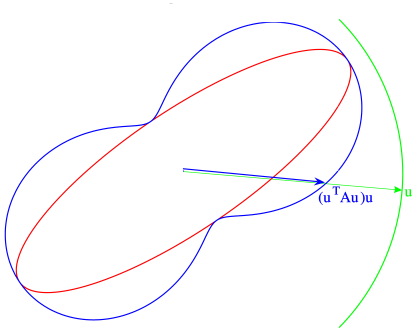

Figure 2: Plot of the variance: The outer figure eight is direction $\boldsymbol{u}$ times the variance $\boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}$. The inner red curve is matrix $\boldsymbol{A}$ represented as an ellipse. The partial green arc is part of the unit circle.

We often express the variance as a trace[2]:

$$\boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u} = \mathrm{tr}(\boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u}) = \mathrm{tr}(\boldsymbol{A}\,\boldsymbol{u}\boldsymbol{u}^\top) \;\geq\; 0.$$

For an eigenvector, this variance equals the eigenvalue and touches the ellipse (See Figure 2).

3. Staying Close to a Subspace

Consider for example the 2 out of 5 literal monotone disjunction $v_1 \vee v_3$, which is represented by the bit vector $\boldsymbol{w} = (1,0,1,0,0)^\top$. The disjunction evaluates to true on the bit vector $\boldsymbol{x} = (0,1,1,0,0)^\top$ because $\boldsymbol{w} \cdot \boldsymbol{x} \geq \frac{1}{2}$.[3] In general, an r literal disjunction is modeled as a bit vector with r ones and the disjunction is true if $\boldsymbol{d} \cdot \boldsymbol{x} \geq 1$. Note that if the vectors $\boldsymbol{d}$ and $\boldsymbol{x}$ are viewed as diagonal matrices, then the dot product becomes a trace: $\boldsymbol{d} \cdot \boldsymbol{x} = \mathrm{tr}(\mathrm{diag}(\boldsymbol{d})\,\mathrm{diag}(\boldsymbol{x}))$.

When we generalize our setup to matrices then disjunctions naturally correspond to subspaces. A subspace $\boldsymbol{P}$ is defined by a rank r projection matrix, $\boldsymbol{P} = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{u}_i^\top$, where the $\boldsymbol{u}_i$ are r orthogonal directions defining the subspace. An instance is a positive definite symmetric matrix $\boldsymbol{X}$ with eigenvalues in the range [0..1] and $\boldsymbol{P}$ labels $\boldsymbol{X}$ as 1 if it is "close to" $\boldsymbol{P}$ and 0 otherwise. Here closeness is measured with the trace $\mathrm{tr}(\boldsymbol{P}\boldsymbol{X})$ which can be rewritten as a sum of variances over the r directions defining $\boldsymbol{P}$:

$$\mathrm{tr}(\boldsymbol{P}\boldsymbol{X}) = \mathrm{tr}(\sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{u}_i^\top \boldsymbol{X}) = \sum_{i=1}^{r} \underbrace{\boldsymbol{u}_i^\top \boldsymbol{X} \boldsymbol{u}_i}_{\text{variance in dir.}\boldsymbol{u}_i}.$$

[2]The trace of a square matrix is the sum of the diagonal. For two such matrices $\boldsymbol{A}, \boldsymbol{B}$, $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \sum_{i,j} A_{i,j} B_{j,i} = \mathrm{tr}(\boldsymbol{B}\boldsymbol{A})$. This inner product can also be denoted as $\boldsymbol{A}^\top \bullet \boldsymbol{B}$.

[3]The dot product with a positively labeled bit vector is ≥ 1 and with a negative bit vector is 0. The threshold of $\frac{1}{2}$ is right in the middle of the gap between the positive and negative bit vectors.

Since the eigenvalues of $\boldsymbol{X}$ lie in [0,1], the variance along each direction is at most one. The "generalized disjunction" evaluates to one if the total variance along all directions is above a threshold. In the simplest case the instance matrices $\boldsymbol{X}_t$ are dyads $\boldsymbol{x}_t \boldsymbol{x}_t^\top$. See Figure 3 for a visualization of the regions produced by the threshold $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top) \geq \frac{1}{2}$ when $\boldsymbol{P}$ is a projection matrix.

Close to Subspace Problem
Input: A sequence of examples $(\boldsymbol{X}_t, y_t)$, where the instances $\boldsymbol{X}_t$ are symmetric matrices with eigenvalues in [0,1], the labels y_t are ± 1, and there is a projection matrix $\boldsymbol{P}$ of rank r s.t. $\forall\, t$

$$\mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) = \begin{cases} \geq \frac{1}{2} & \text{if } y_t = 1 \\ 0 & \text{if } y_t = -1 \end{cases}$$

Output: A symmetric positive definite matrix $\boldsymbol{W}$ of rank r s.t. $\forall t$

$$\mathrm{tr}(\boldsymbol{W}\boldsymbol{X}_t) = \begin{cases} \geq \frac{1}{2} & \text{if } y_t = 1 \\ 0 & \text{if } y_t = -1 \end{cases}$$

We don't know whether the Close to Subspace problem is NP-hard. However, this is easy to show if we augment the problem with a fixed set of additional linear constraints that must hold for both the consistent projection matrix $\boldsymbol{P}$ and the output matrix $\boldsymbol{W}$.[4]

Theorem 1. *The Close to Subspace problem with linear side constraints is NP-hard.*

Proof. Given an instance of the Set Cover problem: a family of subsets S^i of a fixed set of n elements and a cover size r. The size r becomes the rank constraint of the corresponding CSP instance. Represent each set S^i as a positive example $(\boldsymbol{X}_i, +1)$ where $\boldsymbol{X}^i$ is a binary diagonal instance matrix s.t. whose j diagonal element is 1 if the j-th element lies in S^i (and 0 otherwise). Add $O(n^2)$ additional constraints to the CSP instance that explicitly force all off-diagonal elements to be zero: $\mathrm{tr}(\boldsymbol{W}\boldsymbol{X}^{i,j}) = 0$, where $\boldsymbol{X}^{i,j}$ is the $n \times n$ dimensional 0 matrix with positions (i,j) and (j,i) replaced by 1.

Clearly, any cover of size r can be encoded as a diagonal matrix $\boldsymbol{W}$ with r ones and $n-r$ zeros along the diagonal. Also if a symmetric positive definite matrix $\boldsymbol{W}$ has rank r and satisfies the side constraints then it must be a diagonal matrix with r non-zero entries in the diagonal. Wlog the non-zero elements are all 1's and in this case $\boldsymbol{W}$ corresponds to a cover. □

[4]NP-hardness reductions from Maximum Cut are also possible (already for $r = 1$). In this case no side constraints are needed but the Close to Subspace problem needs to be scaled in various ways. Here we only give a simple reduction that uses side constraints.

Algorithm 1 Symmetric Matrix Winnow(η, θ, w_0, n)
$\eta > 0, \theta, w_0 \in \mathbb{R}, n \in \mathbb{N}$

Initialize $\boldsymbol{W}_1 = w_0 \underset{n\times n}{\boldsymbol{I}}$
for $t = 1$ to T **do**
Receive instance $\underset{n\times n}{\boldsymbol{X}_t}$ with eigenvalues in $[0, 1]$
Predict with

$$\hat{y}_t = \begin{cases} +1 & \text{if } \mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

Receive label y_t
Update

$$\boldsymbol{W}_{t+1} = \begin{cases} \boldsymbol{W}_t & \text{if } \hat{y}_t = y_t \\ \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t + \eta\, y_t \boldsymbol{X}_t) & \text{otherwise} \end{cases} \quad (1)$$

end for

4. The Algorithm

Consider the following online protocol: In each trial the algorithm is given an instance matrix $\boldsymbol{X}_t$. The algorithm then produces a prediction $\hat{y}_t \in \{0, 1\}$ based on its internal parameters and receives a label $y_t \in \{+1, -1\}$. Finally, the algorithm incurs a mistake if $\hat{y}_t \neq y_t$ and updates its parameters.

The Symmetric Matrix Winnow algorithm never outputs a projection matrix of low rank that predicts well. Instead it maintains a symmetric positive definite matrix $\boldsymbol{W}_t$ (of full rank) as its parameter and produces a binary prediction $\hat{y}_t$ based on thresholding $\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$. The parameter matrix is updated using the matrix log and exponential.[5] As the original Winnow algorithm, the parameter is only updated when a prediction mistake occurs.[6] Note that the matrix log and exponential of a symmetric matrix are computed by first computing the eigendecomposition of the argument matrix and then applying the function to all eigenvalues.

5. The Bound

We bound the number of mistakes made by our algorithm with similar methods as were used for the Winnow algorithm (see e.g. (Auer & Warmuth, 1998)). The original Winnow handles the special case when all matrices are diagonal. As customary for this type of analysis[7] with symmetric positive definite matrix parameters, we use the quantum relative entropy (see e.g. (Nielsen & Chuang, 2000)) between such matrices $\mathbf{V}$ and $\boldsymbol{W}$ as our measure of progress:

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

As before, $\mathbf{log}\,\boldsymbol{A}$ of the symmetric positive definite matrix $\boldsymbol{A}$ is defined as a spectral function which applies the logarithm to the eigenvalues of $\boldsymbol{A}$ but leaves eigensystem unaltered.

The proof follows the methodology developed in (Tsuda et al., 2005; Warmuth & Kuzmin, 2006a) except that here we work with arbitrary symmetric positive definite matrices instead of density matrices which have the additional requirement that their trace is 1. We make use of the Golden Thompson inequality (see e.g. (Bhatia, 1997)), which holds for arbitrary symmetric matrices

$$\mathrm{tr}(\mathbf{exp}(\boldsymbol{A} + \boldsymbol{B})) \leq \mathrm{tr}(\mathbf{exp}(\boldsymbol{A})\,\mathbf{exp}(\boldsymbol{B}))$$

We also need two lemmas from (Tsuda et al., 2005):

Lemma 1. *For any symmetric matrix $\boldsymbol{X}$, such that $\mathbf{0} \preceq \boldsymbol{X} \preceq \boldsymbol{I}$ and any constant $a \in \mathbb{R}$, the following holds:*

$$\mathbf{exp}(a\boldsymbol{X}) \preceq \boldsymbol{I} - (1 - \exp(a))\boldsymbol{X}.$$

Lemma 2. *For any positive semidefinite matrix $\boldsymbol{A}$ and any symmetric matrices $\boldsymbol{B}$ and $\boldsymbol{C}$, $\boldsymbol{B} \preceq \boldsymbol{C}$ implies $\mathrm{tr}(\boldsymbol{AB}) \leq \mathrm{tr}(\boldsymbol{AC})$.*

We are now ready to prove our main result. For the sake of simplicity this theorem only handles the noise-free case.

Theorem 2. *Given any sequence of examples $(\boldsymbol{X}_t, y_t)$ such that the instances $\boldsymbol{X}_t$ are symmetric positive definite matrices with eigenvalues in [0,1], the labels y_t are ± 1, and there is a projection matrix s.t.*

$$\mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) = \begin{cases} \geq \frac{1}{2} & \text{if } y_t = +1 \\ 0 & \text{if } y_t = -1 \end{cases} \quad (2)$$

Then the online algorithm Symmetric Matrix Winnow makes at most

$$7.18\, r \ln \frac{n}{r}$$

mistakes on this sequence, when $\eta \approx 1.28$, $\theta \approx .19$ and $w_0 = r$.

Proof. We lower bound the per trial progress

$$\Delta(\boldsymbol{P}, \boldsymbol{W}_t) - \Delta(\boldsymbol{P}, \boldsymbol{W}_{t+1})$$

[5] The normalized version of Symmetric Matrix Winnow scales the trace of the resulting matrix to the desired value.

[6] Such updates are called *conservative* in (Littlestone, 1988). The algorithm can be interpreted as a Matrix Exponentiated Gradient algorithm (Tsuda et al., 2005) wrt a hinge loss. In trials when no mistake occurs, the gradient of this hinge loss is zero, which results in no update (Gentile & Warmuth, 1998).

[7] Alternatively, $\mathrm{tr}(\boldsymbol{W}_t)$ can be used as a potential.

towards a comparator matrix $\boldsymbol{P}$, which is any projection matrix $\boldsymbol{P}$ satisfying the constraints (2). If no mistake occurs in trial t, then $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t$ and the progress is zero. Otherwise, $\boldsymbol{W}_{t+1}$ is updated to $\mathbf{exp}(\log \boldsymbol{W}_t + \eta y_t \boldsymbol{X}_t)$ and

$$\begin{aligned}
&\Delta(\boldsymbol{P}, \boldsymbol{W}_t) - \Delta(\boldsymbol{P}, \boldsymbol{W}_{t+1}) \\
&= \mathrm{tr}(\boldsymbol{P}(\log \boldsymbol{W}_{t+1} - \log \boldsymbol{W}_t) + \boldsymbol{W}_t - \boldsymbol{W}_{t+1}) \\
&\overset{(1)}{=} \eta y_t \mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) + \mathrm{tr}(\boldsymbol{W}_t - \mathbf{exp}(\log \boldsymbol{W}_t + \eta y_t \boldsymbol{X}_t)) \\
&\overset{\text{G.Th.}}{\geq} \eta y_t \mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) + \mathrm{tr}(\boldsymbol{W}_t - \mathbf{exp}(\log \boldsymbol{W}_t)\,\mathbf{exp}(\eta y_t \boldsymbol{X}_t)) \\
&= \eta y_t \mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) + \mathrm{tr}(\boldsymbol{W}_t(\boldsymbol{I} - \mathbf{exp}(\eta y_t \boldsymbol{X}_t)) \\
&\overset{Le.1\&2}{\geq} \eta y_t \mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) + (1 - \exp(\eta y_t))\ \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t)
\end{aligned}$$

If $y_t = -1$ and $\hat{y}_t = +1$, then $\mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) = 0$, $1 - \exp(\eta y_t) > 0$ and $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t) \geq \theta$. Thus the progress is lower bounded by

$$(1 - \exp(-\eta))\theta.$$

Similarly, if $y_t = +1$ and $\hat{y}_t = -1$, then $\mathrm{tr}(\boldsymbol{P}\boldsymbol{X}_t) \geq \frac{1}{2}$, $1 - \exp(\eta y_t) < 0$ and $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t) \leq \theta$. Now the progress is lower bounded by

$$\frac{\eta}{2} + (1 - \exp(\eta))\theta.$$

By setting both cases equal we obtain

$$\theta = \frac{\eta}{2(\exp(\eta) - \exp(-\eta))}.$$

With this choice of θ, the progress per mistake is $\frac{\eta}{2+2\exp(\eta)}$ and this is optimized at $\eta \approx 1.28$. This choice of η makes $\theta \approx .19$ and the progress $\approx .14$. By summing over trials we get

$$\underbrace{\Delta(\boldsymbol{P}, \overbrace{\boldsymbol{W}^1}^{r\boldsymbol{I}})}_{r \ln \frac{n}{r}} - \underbrace{\Delta(\boldsymbol{P}, \boldsymbol{W}^{T+1})}_{\geq 0} \geq .14M,$$

where M is the number of mistakes. This implies that $M \leq 7.18\, r \log_2 \frac{n}{r}$. □

Tightness of the Bound: The Vapnik-Chervonenkis dimension of r out of n monotone disjunctions is at least $r\lfloor \log_2 \frac{n}{r} \rfloor$ (Littlestone, 1988). In other words, there are this many bit vectors $\boldsymbol{x}_t$ of dimension n with the property that for each possible labeling there is a consistent monotone disjunction of r of the n variables. By feeding the instances $\mathrm{diag}(\boldsymbol{x}_i)$ to any algorithm that solves the Close to Subspace problem and contradicting the prediction of the algorithm each time, we can force any such algorithm to make $r\lfloor \log_2 \frac{n}{r} \rfloor$ mistakes. The sequence of examples will have a consistent disjunction of size r which is represented as a bit vector $\boldsymbol{d}$ with r ones. This means that the r dimensional projection matrix $\mathrm{diag}(\boldsymbol{d})$ with the identity eigensystem will label the matrix instances consistently. Thus the upper bound proven above for Symmetric Matrix Winnow cannot be improved by more than a constant.

Time Complexity: Given that you have the eigendecomposition of $\boldsymbol{W}_t$, then the expensive part in computing $\boldsymbol{W}_{t+1}$ is producing the eigendecomposition of the exponent $\log \boldsymbol{W}_t + \eta y_t \boldsymbol{X}_t$ of update (1). This costs $O(n^3)$ time per update. The algorithm of the next section is based on the singular value decomposition, which has similar complexity. In some applications one might get away with not updating at every trial. However, at this point the methods presented here are only applicable to small matrices. Nevertheless, they show what kind of bounds are possible if computation time is cheap.

Noisy Case: In this paper we kept things simple and only focused on the noise-free case. The bounds for learning noisy linear threshold functions with Winnow when the loss function is the hinge loss (see e.g. (Gentile & Warmuth, 1998)) immediately carry over to the matrix case. For example this leads to bounds in terms of the hinge loss of the best subspace.

Linear Side Constraints: Now the parameter matrix is projected onto the linear side constraints after the main update (1). Using the Bregman projection methods of (Herbster & Warmuth, 2001) one can show that the bounds stay the same provided that the target satisfies the constraints.[8]

Generalization Bounds: Any mistake bounded on-line algorithm can easily be converted to a generalization bound for the case when the instances are generated independently at random by a fixed but unknown distribution (Littlestone, 1989; Cesa-Bianchi & Gentile, 2006). One of the simplest conversion algorithm for a conservative on-line algorithm might be the following (Floyd & Warmuth, 1995). Draw a batch of m examples and in some default order on the instances do one pass over the examples with the mistake bounded on-line algorithm. If the on-line algorithm is guaranteed to make at most M mistakes then the subset of at most M examples where mistakes were made represents a consistent hypothesis. In other words a mistake bound of size M leads to a "compression scheme" of size M. This immediately implies that for any $\epsilon, \delta \in [0, 1]$, $O(\frac{1}{\epsilon}(M \log \frac{1}{\epsilon} + \log \frac{1}{\delta}))$ examples suffice

[8] Actually projections onto an arbitrary convex region can be used.

to guarantee the following about the produced consistent hypothesis: its probability of predicting wrongly is at most ϵ with probability at least $1-\delta$.

6. Non-square Instance Matrices

Now the instances $\boldsymbol{X}$ are non-square matrices of dimension $m \times n$, where $m \geq n$ for the purpose of discussion. We use the standard trick (see e.g. (Golub & Loan, 1996)) to embed[9] such matrices into a symmetric matrix of dimension $(m+n) \times (m+n)$:

$$\begin{pmatrix} \underset{n\times n}{\mathbf{0}} & \underset{n\times m}{\boldsymbol{X}^T} \\ \underset{m\times n}{\boldsymbol{X}} & \underset{m\times m}{\mathbf{0}} \end{pmatrix}.$$

Note that $\mathbf{0}$ always denotes a square zero matrix of the appropriate dimension. From now on we also use bold greek letters to denote square diagonal matrices.

If $\boldsymbol{X}$ has the singular value decomposition $\underset{m\times m}{\boldsymbol{U}} \underset{m\times n}{\begin{pmatrix} \boldsymbol{\sigma} \\ \mathbf{0} \end{pmatrix}} \underset{n\times n}{\mathbf{V}^\top} = \underset{m\times n}{\boldsymbol{U}_1} \underset{n\times n}{\boldsymbol{\sigma}} \mathbf{V}^\top$, where $\boldsymbol{U}_1$ consists of the first n columns of $\boldsymbol{U}$, then the corresponding symmetric matrix has the following eigendecomposition (Golub & Loan, 1996):

$$\underset{(m+n)\times(m+n)}{\begin{pmatrix} \mathbf{0} & \boldsymbol{X}^T \\ \boldsymbol{X} & \mathbf{0} \end{pmatrix}} = \mathbf{Q} \begin{pmatrix} \boldsymbol{\sigma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{Q}^\top = \widehat{\mathbf{Q}} \underset{(2n)\times(2n)}{\begin{pmatrix} \boldsymbol{\sigma} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\sigma} \end{pmatrix}} \widehat{\mathbf{Q}}^\top$$

$$\text{where } \underset{(m+n)\times(m+n)}{\mathbf{Q}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{V} & \mathbf{V} & \mathbf{0} \\ \boldsymbol{U}_1 & -\boldsymbol{U}_1 & \sqrt{2}\boldsymbol{U}_1 \end{pmatrix}$$

$$\text{and } \underset{(m+n)\times(2n)}{\widehat{\mathbf{Q}}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{V} & \mathbf{V} \\ \boldsymbol{U}_1 & -\boldsymbol{U}_1 \end{pmatrix}.$$

We can now run Symmetric Matrix Winnow with the embedded instances $\begin{pmatrix} \mathbf{0} & \boldsymbol{X}_t^\top \\ \boldsymbol{X}_t & \mathbf{0} \end{pmatrix}$ and the initial parameter matrix $\boldsymbol{W}_1 = w_0 \begin{pmatrix} \mathbf{0} & \boldsymbol{I}^\top \\ \boldsymbol{I} & \mathbf{0} \end{pmatrix}$, where $w_0 = r$ and $\boldsymbol{I}$ denotes the $m \times n$ dimensional identity matrix. Our update produces a symmetric positive definite matrix $\boldsymbol{W}_t$ of the form

$$\frac{1}{2} \exp \begin{pmatrix} \mathbf{0} & \mathbf{R}_t^\top \\ \mathbf{R}_t & \mathbf{0} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \widetilde{\mathbf{V}} & \widetilde{\mathbf{V}} \\ \widetilde{\boldsymbol{U}}_1 & -\widetilde{\boldsymbol{U}}_1 \end{pmatrix} \begin{pmatrix} \exp(\widetilde{\boldsymbol{\sigma}}) & \mathbf{0} \\ \mathbf{0} & \exp(-\widetilde{\boldsymbol{\sigma}}) \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{V}} & \widetilde{\mathbf{V}} \\ \widetilde{\boldsymbol{U}}_1 & -\widetilde{\boldsymbol{U}}_1 \end{pmatrix}^\top$$

where $\mathbf{R}_t$ is a linear combination of the instances $\boldsymbol{X}_1 \ldots \boldsymbol{X}_{t-1}$ that has the singular value decomposition $\underset{m\times n}{\widetilde{\boldsymbol{U}}_1} \underset{n\times n}{\widetilde{\boldsymbol{\sigma}}} \widetilde{\mathbf{V}}^T$. We can rewrite $\boldsymbol{W}_t$ as

$$\frac{1}{2} \begin{pmatrix} \widetilde{\mathbf{V}} \cosh(\widetilde{\boldsymbol{\sigma}}) \widetilde{\mathbf{V}}^\top & \widetilde{\mathbf{V}} \sinh(\widetilde{\boldsymbol{\sigma}}) \widetilde{\boldsymbol{U}}_1^\top \\ \widetilde{\boldsymbol{U}}_1 \sinh(\widetilde{\boldsymbol{\sigma}}) \widetilde{\mathbf{V}}^\top & \widetilde{\boldsymbol{U}}_1 \cosh(\widetilde{\boldsymbol{\sigma}}) \widetilde{\boldsymbol{U}}_1^\top \end{pmatrix}^\top$$

[9] The reduction is different from the $EG^{\pm}$ reduction of (Kivinen & Warmuth, 1997) which would embed $\boldsymbol{X}$ as $\begin{pmatrix} \boldsymbol{X} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{X} \end{pmatrix}$, which is unfortunately not symmetric.

Algorithm 2 General Matrix Winnow$(\eta, \theta, w_0, m, n)$
$\eta > 0, \theta, w_0 \in \mathbb{R}, m, n \in \mathbb{N}$

Initialize $\mathbf{R}_1 = w_0 \underset{m\times n}{\boldsymbol{I}}$
for $t = 1$ to T do **do**
Receive instance $\underset{m\times n}{\boldsymbol{X}_t}$ with singular values in $[0,1]$
Predict with

$$\hat{y}_t = \begin{cases} +1 & \text{if } \operatorname{tr}(\mathbf{R}_t \boldsymbol{X}_t) \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

Receive label y_t
Update

$$\mathbf{R}_{t+1} = \begin{cases} \mathbf{R}_t & \text{if } \hat{y}_t = y_t \\ \mathbf{sinh}(\mathbf{arcsinh}(\mathbf{R}_t) + \eta \boldsymbol{X}_t) & \text{otherwise} \end{cases}$$

end for

where **sinh** and **cosh** are the diagonalized versions of the scalar sinh and cosh functions.[10] This lets us write the trace which is used for forming the prediction as follows:

$$\operatorname{tr}\left(\boldsymbol{W}_t \begin{pmatrix} \mathbf{0} & \boldsymbol{X}_t^\top \\ \boldsymbol{X} & \mathbf{0} \end{pmatrix}\right) = \operatorname{tr}(\widetilde{\boldsymbol{U}}_1 \mathbf{sinh}(\widetilde{\boldsymbol{\sigma}}) \widetilde{\mathbf{V}}^\top \ \boldsymbol{X}_t^\top).$$

The General Matrix Winnow[11] algorithm 2 reformulates the entire algorithm sketched above using the original $m \times n$ dimensional instances $\boldsymbol{X}_t$ and an $m \times n$ dimensional parameter matrix $\mathbb{R}_t$. For any $m \times n$ dimensional matrix $\boldsymbol{A}$ with singular value decomposition $\boldsymbol{U} \widetilde{\boldsymbol{\sigma}} \mathbf{V}^T$, the matrix function $\mathbf{sinh}(\boldsymbol{A})$ applies the scalar sinh function to the diagonal elements of $\widetilde{\boldsymbol{\sigma}}$ and $\mathbf{arcsinh}(\boldsymbol{A})$ is defined similarly based on the inverse of the scalar sinh function. The bounds for this algorithm follow from the symmetric case via the above embedding. The symmetric projection matrices of rank r are now *generalized projection matrices* of the form $\boldsymbol{P} = \sum_{i=1}^r s_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$, where $s_i \in \{+1, -1\}$ and the $\boldsymbol{u}_i$ are a set of r orthogonal m-dimensional unit vector and the $\boldsymbol{v}_i$ are a set of r orthogonal n-dimensional unit vectors.

7. Conclusion

It is unknown how to handle rank constraints. When linear side constraints are added, the Close to Subspace problem is hard even if we allow the output matrix $\boldsymbol{W}$ to be any positive symmetric matrix of low rank. We follow the route taken by the original Winnow algorithm and bypass the hardness by never pro-

[10] $\sinh(x) := \frac{\exp(x)-\exp(-x)}{2}$, $\cosh(x) := \frac{\exp(x)+\exp(-x)}{2}$.

[11] The normalized version scales with the trace norm.

ducing a hypotheses of a certain form (here low rank). The algorithm maintains a positive definite parameter matrix of *full rank* instead. Nevertheless it is guaranteed to make few prediction mistakes if the instances are labeled based on a low rank projection matrix. The update of the algorithm is a special case of the Matrix Exponentiated Gradient algorithm and is motivated by regularizing the hinge loss on the linear trace with a quantum relative entropy. Note that the original Winnow is motivated along the same lines (see e.g. (Gentile & Warmuth, 1998)).

More generally the following picture is emerging. In the vector case the algorithms related to the original Winnow algorithm learn with a thresholded dot product between the weight vector and instance vector. In the noise-free case there has to be a gap between the positive and negative examples, and a threshold can be placed in this gap for the purpose of classifying the examples. For the original Winnow algorithm, the weight vectors are measured by the one-norm and this is also the norm by which the margin of the examples is normalized. For the instance vectors the range of the components (essentially the infinity norm) is crucial.

In the matrix case the dot product becomes a trace and the parameter matrices are measured by their trace norm (sum of singular values). This norm also naturally normalizes the margin of the examples. The instance matrices are naturally measured by the matrix 2-norm (maximum singular value).

Acknowledgment

Thanks to Dima Kuzmin for many brainstorming sessions and for preparing all the plots of this paper. Also thanks to Amin Coja-Oghlan and Elad Hazan for valuable discussions.

References

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

Auer, P., & Warmuth, M. K. (1998). Tracking the best disjunction. *Machine Learning*, *32*, 127–150. Earlier version in 36th FOCS, 1995.

Bhatia, R. (1997). *Matrix analysis*. Berlin: Springer.

Cesa-Bianchi, N., & Gentile, C. (2006). Improved risk tail bounds for on-line learning. *Advances in Neural Information Processing Systems 18 (NIPS 05)*. MIT Press.

Floyd, S., & Warmuth, M. (1995). Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, *21*, 269–304.

Gentile, C., & Warmuth, M. K. (1998). Hinge loss and average margin. *In Advances in Neural Information Processing Systems 11 (NIPS*98)*.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations (third edition)*. The John Hopkins University Press.

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

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

Kuzmin, D., & Warmuth, M. K. (2007). Online Kernel PCA with entropic matrix updates. *ICML '07: Proceedings of the 24rd international conference on Machine learning*. Corvallis, Oregon: ACM Press.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N. (1989). From on-line to batch learning. *Proceedings of the Second Annual Workshop on Computational Learning Theory* (pp. 269–284). San Mateo, CA: Morgan Kaufmann.

Nielsen, M., & Chuang, I. (2000). *Quantum computation and quantum information*. Cambridge University Press.

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

Warmuth, M. K. (2007). When is there a free matrix lunch. *Proc. of the 20th Annual Conference on Learning Theory (COLT 07)*. San Diego, CA: Springer. Open problem, to appear.

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

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

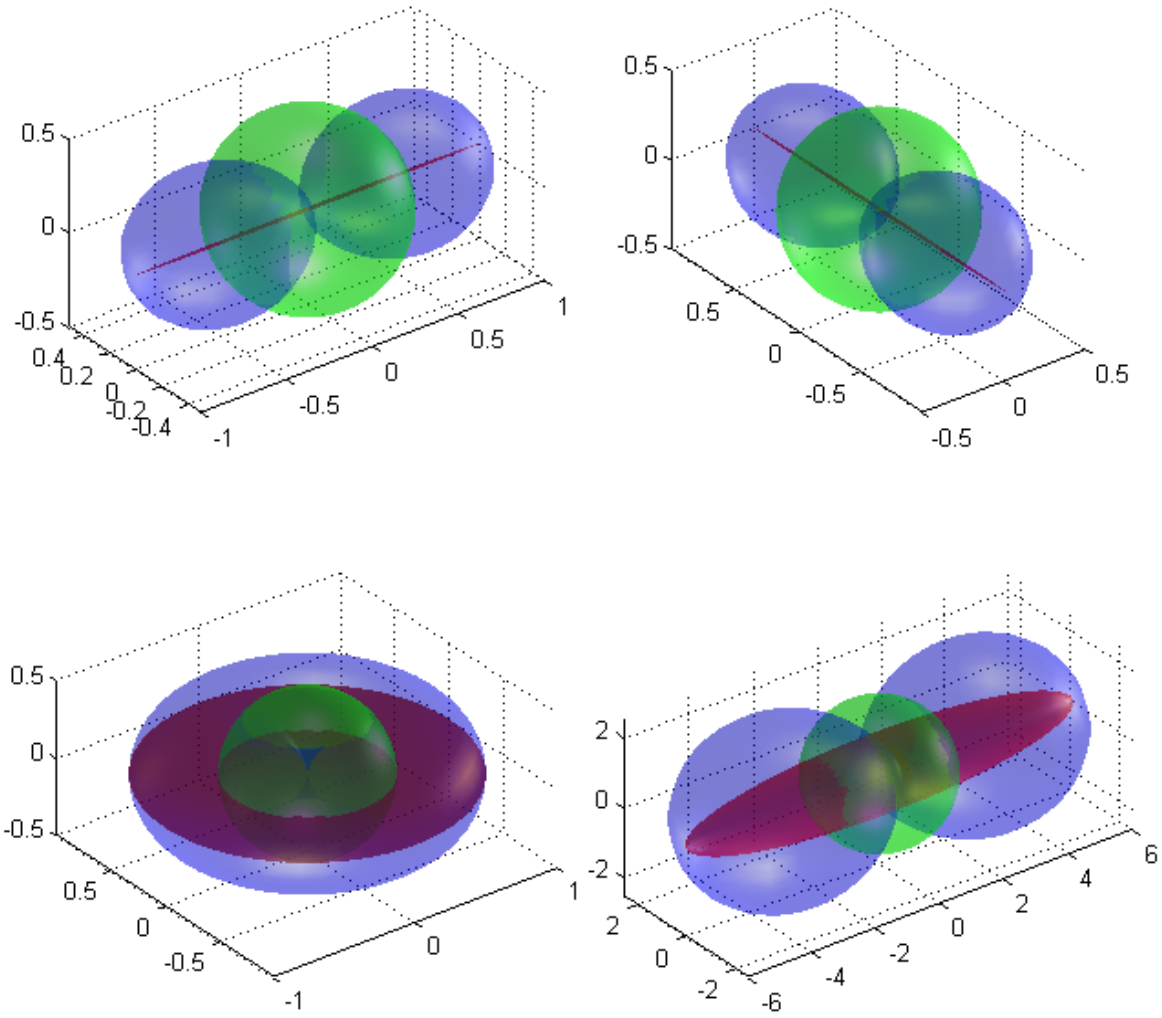

Figure 3: Depiction of a threshold of trace $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top)$: **Top left**: The projection matrix $\boldsymbol{P}$ has dimension one. It is a single (red) dyad along the x-axis, i.e. $\boldsymbol{P} = \boldsymbol{e}_1\boldsymbol{e}_1^\top$, where $\boldsymbol{e}_1 = (1,0,0)^\top$. The blue double cloud is a plot of the variance $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top)$ for all unit directions $\boldsymbol{x}$. Recall that in dimension 2 the variance has figure eight shape (see blue curve in Figure 2). The green ball has radius $\frac{1}{2}$ and the threshold $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top) \geq \frac{1}{2}$ is satisfied if direction $\boldsymbol{x}\boldsymbol{x}^\top$ is close to the red axis, i.e. in the blue minus the green region. It is instructive to see what happens if the threshold (radius of the green ball) is raised to 1. Then the region labeled one (blue - green) just consists of the two tips of the red dyad $\boldsymbol{e}_1\boldsymbol{e}_1^\top$. That is, $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top) \geq 1$ iff $\boldsymbol{x} = \pm\boldsymbol{e}_1$. **Top right**: The same picture but now $\boldsymbol{P}$ is the dyad $\boldsymbol{e}_2\boldsymbol{e}_2^\top$ corresponding to the y-axis. **Bottom left**: Now $\boldsymbol{P}$ is the sum of the previous two dyads, i.e. $\boldsymbol{P} = \boldsymbol{e}_1\boldsymbol{e}_1^\top + \boldsymbol{e}_2\boldsymbol{e}_2^\top$ depicted as the red degenerate ellipse (disk) in the xy plane. The variance is a blue donut around this ellipse. The green ball again has radius $\frac{1}{2}$ and the threshold $\mathrm{tr}(\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top) \geq \frac{1}{2}$ labels all directions $\boldsymbol{x}\boldsymbol{x}^\top$ as one that are close to the red ellipse, i.e. lie in the blue donut minus the green ball. If the radius of the green ball is raised to 1, then the blue minus the red region shrinks to the circle where the donut touches the disk. **Bottom right:** Now the inner red 3-dimensional ellipse is sum of three directions, i.s. $\boldsymbol{P} = 6\,\boldsymbol{e}_1\boldsymbol{e}_1^\top + \boldsymbol{e}_2\boldsymbol{e}_2^\top + \boldsymbol{e}_3\boldsymbol{e}_3^\top$. The variance is the blue handle and the threshold (green ball) is 2.