Journal of Machine Learning Research 6 (2005) 995–1018 Submitted 11/04; Revised 4/05; Published 6/05

Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection

Koji Tsuda KOJI.TSUDA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany, and
Computational Biology Research Center
National Institute of Advanced Science and Technology (AIST)
2-42 Aomi, Koto-ku, Tokyo
135-0064, Japan

Gunnar Rätsch GUNNAR.RAETSCH@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory of the Max Planck Society
Spemannstrasse 35
72076 Tübingen, Germany

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

Editor: Yoram Singer

Abstract

We address the problem of learning a symmetric positive definite matrix. The central issue is to design parameter updates that preserve positive definiteness. Our updates are motivated with the *von Neumann* divergence. Rather than treating the most general case, we focus on two key applications that exemplify our methods: on-line learning with a simple square loss, and finding a symmetric positive definite matrix subject to linear constraints. The updates generalize the exponentiated gradient (EG) update and AdaBoost, respectively: the parameter is now a symmetric positive definite matrix of trace one instead of a probability vector (which in this context is a diagonal positive definite matrix with trace one). The generalized updates use matrix logarithms and exponentials to preserve positive definiteness. Most importantly, we show how the derivation and the analyses of the original EG update and AdaBoost generalize to the non-diagonal case. We apply the resulting *matrix exponentiated gradient* (MEG) update and *DefiniteBoost* to the problem of learning a kernel matrix from distance measurements.

1. Introduction

Most learning algorithms have been developed to learn a *vector* of parameters from data. However, an increasing number of papers are now dealing with more structured parameters. More specifically, when learning a similarity or a distance function among objects, the parameters are defined as a *symmetric positive definite matrix* that serves as a kernel (e.g., Xing et al., 2003; Shai-Shwartz et al., 2004; Tsang and Kwok, 2003; Tsuda and Noble, 2004). Learning is typically formulated as a parameter updating procedure to optimize a *loss function*. The gradient descent update is one of the most commonly used algorithms, but it is not appropriate when the parameters form a positive definite matrix, because the updated parameter matrix does not necessarily stay positive definite. Xing et al. (2003) solved this problem by always correcting the updated matrix to be positive definite. However no bound has been proven for this update-and-correct approach. Also, Shai-Shwartz et al. (2004) proposed an on-line algorithm for learning a kernel matrix when only some of the class labels of the examples are provided. This algorithm is also based on the update-and-correction approach, but since the update step performs rank-one modification, the correction step can be efficiently implemented. They have shown a generalization bound inspired by similar previously known bounds for the perceptron.

In this paper, we introduce the *matrix exponentiated gradient update* which works as follows: First, the matrix logarithm of the current parameter matrix is computed. Then a step is taken in the direction of the steepest descent of the loss function. Finally, the parameter matrix is updated to the exponential of the modified log-matrix. Our update preserves symmetry and positive definiteness because the matrix exponential maps any symmetric matrix to a symmetric positive definite matrix.

Bregman divergences play a central role in the motivation and the analysis of *on-line learning algorithms* (Kivinen and Warmuth, 1997). A learning problem is essentially defined by a loss function and a divergence that measures the discrepancy between parameters. More precisely, the updates are motivated by minimizing the sum of the loss function and the Bregman divergence, where the loss function is multiplied by a positive learning rate. Different divergences lead to radically different updates (Kivinen and Warmuth, 1997, 2001). For example, the gradient descent update is derived from the squared Euclidean distance, and the exponentiated gradient update from the Kullback-Leibler divergence (relative entropy). In this work we use the *von Neumann* divergence (also called quantum relative entropy) for measuring the discrepancy between two positive definite matrices (Nielsen and Chuang, 2000). We derive a new *matrix exponentiated gradient update* from this divergence (which is a Bregman divergence for symmetric positive definite matrices). Finally we prove *relative loss bounds* using the *von Neumann* divergence as a measure of progress.

We apply our techniques to solve the following related key problem that has received a lot of attention recently (Xing et al., 2003; Shai-Shwartz et al., 2004; Tsang and Kwok, 2003; Tsuda and Noble, 2004). Find a symmetric positive definite matrix that satisfies a number of linear inequality constraints. The new *DefiniteBoost* algorithm greedily chooses a violated linear constraint and performs an approximated Bregman projection. In the diagonal case, we recover AdaBoost (Schapire and Singer, 1999). We also show how the convergence proof of AdaBoost generalizes to the non-diagonal case.

2. Preliminaries

In this section, we first present mathematical definitions and basic lemmas.

2.1 Matrix Basics

We denote matrices by capital bold letters and restrict ourselves to square matrices with real entries in this paper. For any such matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, $\mathbf{exp}\,\boldsymbol{A}$ and $\mathbf{log}\,\boldsymbol{A}$ denote the matrix exponential and logarithm, respectively. The matrix exponential is defined as the following power series,

$$\mathbf{exp}(\boldsymbol{A}) := \boldsymbol{I} + \boldsymbol{A} + \frac{1}{2!}\boldsymbol{A}^2 + \frac{1}{3!}\boldsymbol{A}^3 + \cdots . \tag{2.1}$$

In the case of symmetric matrices, the matrix exponential operation can be computed using the eigenvalue decomposition $\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$, where $\boldsymbol{V}$ is an orthonormal matrix with the eigenvectors of $\boldsymbol{A}$ as columns and Λ the diagonal matrix of eigenvalues. Thus, $\mathbf{exp}\,\boldsymbol{A} = \boldsymbol{V}(\mathbf{exp}\,\Lambda)\boldsymbol{V}^\top$, where $(\mathbf{exp}\,\Lambda)_{i,i} = \exp(\Lambda_{i,i})$. The matrix logarithm $\mathbf{log}\,\boldsymbol{A}$ is defined as the inverse function of $\mathbf{exp}\,\boldsymbol{A}$, which does not always exist for arbitrary $\boldsymbol{A}$. However, when $\boldsymbol{A}$ is symmetric and strictly positive definite, $\mathbf{log}\,\boldsymbol{A}$ is computed as $\mathbf{log}\,\boldsymbol{A} := \boldsymbol{V}(\mathbf{log}\,\Lambda)\boldsymbol{V}^\top$, where $(\mathbf{log}\,\Lambda)_{i,i} = \log\Lambda_{i,i}$. Throughout the paper $\log a$ and $\exp a$ denote the natural logarithm and exponential of scalar "a".

A square matrix is positive definite if all its eigenvalues are strictly positive. Positive semi-definiteness only requires the non-negativity of the eigenvalues. For two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{A} \preceq \boldsymbol{B}$ iff $\boldsymbol{B} - \boldsymbol{A}$ is positive semi-definite. Similarly, $\boldsymbol{A} \prec \boldsymbol{B}$ iff $\boldsymbol{B} - \boldsymbol{A}$ is (strictly) positive definite.

The trace of a matrix is the sum of its diagonal elements, i.e. $\mathrm{tr}(\boldsymbol{A}) = \sum_i A_{i,i}$ and thus $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \sum_{i,j} A_{i,j} B_{j,i} = \mathrm{tr}(\boldsymbol{B}\boldsymbol{A})$. In matrix algebra, $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B})$ plays a similar role as the dot product for vectors. Furthermore, $\mathrm{tr}(\boldsymbol{A}) = \sum_i \lambda_i$, where λ_i are the eigenvalues of $\boldsymbol{A}$ and the determinant $\det(\boldsymbol{A}) = \prod_i \lambda_i$.

If $\mathrm{F}(\boldsymbol{W}) : \mathbb{R}^{d\times d} \to \mathbb{R}$ is a real-valued function on matrices, then $\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W})$ denotes the *gradient* with respect to matrix $\boldsymbol{W}$:

$$\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W}) = \begin{pmatrix} \frac{\partial \mathrm{F}}{\partial W_{11}} & \cdots & \frac{\partial \mathrm{F}}{\partial W_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathrm{F}}{\partial W_{d1}} & \cdots & \frac{\partial \mathrm{F}}{\partial W_{dd}} \end{pmatrix}.$$

For example, it is easy to see that $\nabla_{\boldsymbol{A}}\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \boldsymbol{B}^\top$. More examples of computing gradients are given in Appendix A.

For a square matrix $\boldsymbol{X}$, $\mathbf{sym}(\boldsymbol{X}) = (\boldsymbol{X} + \boldsymbol{X}^\top)/2$ denotes the symmetric part of $\boldsymbol{X}$. If $\boldsymbol{W}$ is symmetric and $\boldsymbol{X}$ an arbitrary matrix, then

$$\mathrm{tr}(\boldsymbol{W}\boldsymbol{X}) = \mathrm{tr}\left(\boldsymbol{W}\frac{\boldsymbol{X}+\boldsymbol{X}^\top}{2}\right) + \mathrm{tr}\left(\boldsymbol{W}\frac{\boldsymbol{X}-\boldsymbol{X}^\top}{2}\right) = \mathrm{tr}(\boldsymbol{W}\,\mathbf{sym}(\boldsymbol{X})). \tag{2.2}$$

Our analysis requires the use of the Golden-Thompson inequality (Golden, 1965):

$$\mathrm{tr}(\mathbf{exp}(\boldsymbol{A}+\boldsymbol{B})) \le \mathrm{tr}(\mathbf{exp}(\boldsymbol{A})\,\mathbf{exp}(\boldsymbol{B})), \tag{2.3}$$

which holds for arbitrary *symmetric* matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

We also need the following two basic inequalities for symmetric matrices. The first one generalizes the following simple inequality, which is a realization of Jensen's inequality for the convex function $\exp(x)$: For any $0 \le a \le 1$ and $\rho_1, \rho_2 \in \mathbb{R}$,

$$\exp(a\rho_1 + (1-a)\rho_2) \le a\exp(\rho_1) + (1-a)\exp(\rho_2).$$

In the below generalization, the distribution $(a, 1-a)$ is replaced by $(\boldsymbol{A}, \boldsymbol{I} - \boldsymbol{A})$, where $\boldsymbol{A}$ is any symmetric matrix for which $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$.

Lemma 2.1 *For any symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ such that $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$, and any $\rho_1, \rho_2 \in \mathbb{R}$,*

$$\mathbf{exp}(\boldsymbol{A}\rho_1 + (\boldsymbol{I}-\boldsymbol{A})\rho_2) \preceq \boldsymbol{A}\exp(\rho_1) + (\boldsymbol{I}-\boldsymbol{A})\exp(\rho_2).$$

Proof Assume $\boldsymbol{A}$ is eigen-decomposed as $\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$, where Λ is the diagonal matrix of eigenvalues and $\boldsymbol{V}$ is an orthogonal matrix with the eigenvectors of $\boldsymbol{A}$ as columns. By assumption, $0 \le \lambda_k \le 1$. Let θ_k be the k-th eigenvalue of the left hand side of the inequality that we are to prove. Clearly $\theta_k = \exp(\lambda_k\rho_1 + (1-\lambda_k)\rho_2)$ and by Jensen's inequality, $\theta_k \le \lambda_k \exp(\rho_1) + (1-\lambda_k)\exp(\rho_2)$. Let Θ be the diagonal matrix with entries θ_k. Then $\Theta \preceq \Lambda\exp(\rho_1) + (\boldsymbol{I} - \Lambda)\exp(\rho_2)$, and by multiplying both sides by $\boldsymbol{V}$ from left and by $\boldsymbol{V}^\top$ from right, we obtain the desired inequality. ■

Lemma 2.2 *For any positive semi-definite symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and any two symmetric matrices $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{d\times d}$, $\boldsymbol{B} \preceq \boldsymbol{C}$ implies* $\mathrm{tr}(\boldsymbol{AB}) \le \mathrm{tr}(\boldsymbol{AC})$.

Proof Let $\boldsymbol{D} = \boldsymbol{C} - \boldsymbol{B}$, then $\boldsymbol{D} \succeq \boldsymbol{0}$ by assumption. Suffices to show that $\mathrm{tr}(\boldsymbol{AD}) \ge 0$. Let us eigen-decompose $\boldsymbol{A}$ as $\boldsymbol{V}\Lambda\boldsymbol{V}^\top$. Since $\boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}$, $\boldsymbol{D} = \boldsymbol{V}P\boldsymbol{V}^\top$ where $P = \boldsymbol{V}^\top\boldsymbol{D}\boldsymbol{V} \succeq \boldsymbol{0}$. Then $\mathrm{tr}(\boldsymbol{AD}) = \mathrm{tr}(\boldsymbol{V}\Lambda\boldsymbol{V}^\top\boldsymbol{V}P\boldsymbol{V}^\top) = \mathrm{tr}(\Lambda P) = \sum_{i=1}^n \lambda_i P_{ii}$. Since P is positive semi-definite, the diagonal elements P_{ii} are nonnegative. Also by assumption the eigenvalues λ_i of $\boldsymbol{A}$ are nonnegative. Thus we conclude that $\mathrm{tr}(\boldsymbol{AD}) \ge 0$. ■

2.2 Von Neumann Divergence or Quantum Relative Entropy

If F is a real-valued strictly convex differentiable function on the parameter domain (a subset of matrices in $\mathbb{R}^{d\times d}$) and $\boldsymbol{f}(\boldsymbol{W}) := \nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W})$, then the Bregman divergence between two parameters $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{W}$ is defined as

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) := \mathrm{F}(\widetilde{\boldsymbol{W}}) - \mathrm{F}(\boldsymbol{W}) - \mathrm{tr}((\widetilde{\boldsymbol{W}} - \boldsymbol{W})\boldsymbol{f}(\boldsymbol{W})^\top).$$

Since F is strictly convex, $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W})$ is also strictly convex in its first argument. Furthermore, the gradient in the first argument has the following simple form:

$$\nabla_{\widetilde{\boldsymbol{W}}}\, \Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \boldsymbol{f}(\widetilde{\boldsymbol{W}}) - \boldsymbol{f}(\boldsymbol{W}),$$

since $\nabla_{\boldsymbol{A}}\mathrm{tr}(\boldsymbol{AB}) = \boldsymbol{B}^\top$ (cf. Section 2.1).

For the divergences used in this paper, we restrict ourselves to the domain of symmetric positive definite matrices. Our main choice of F is $\mathrm{F}(\boldsymbol{W}) = \mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W})$, which is called *von Neumann entropy* or *quantum entropy*. The strict convexity of this function is well known (Nielsen and Chuang, 2000). Furthermore we show in Appendix A that $\nabla_{\boldsymbol{W}}\mathrm{F}(\boldsymbol{W}) = \boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\,\boldsymbol{W}$.

The Bregman divergence corresponding to this choice of F is the *von Neumann divergence* or *quantum relative entropy* (e.g., Nielsen and Chuang, 2000):

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}\,\mathbf{log}\,\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\,\mathbf{log}\,\boldsymbol{W} - \widetilde{\boldsymbol{W}} + \boldsymbol{W}).$$

In this paper, we are primarily interested in the case when the parameters are normalized in the sense that $\mathrm{tr}(\boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}) = 1$. Symmetric positive definite matrices of trace one are related to density matrices commonly used in Statistical Physics. For normalized parameters the divergence simplifies to

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}\,\mathbf{log}\,\widetilde{\boldsymbol{W}} - \widetilde{\boldsymbol{W}}\,\mathbf{log}\,\boldsymbol{W}).$$

If $\boldsymbol{W} = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$ is our notation for the eigenvalue decomposition, then the von Neumann entropy[1] becomes $\mathrm{F}(\boldsymbol{W}) = \sum_i \lambda_i \log \lambda_i$. We can rewrite the normalized divergence[2] as

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \sum_i \tilde{\lambda}_i \log \tilde{\lambda}_i - \sum_{i,j} \tilde{\lambda}_i \log \lambda_j (\tilde{\boldsymbol{v}}_i^\top \boldsymbol{v}_j)^2. \tag{2.4}$$

This divergence quantifies the difference in the eigenvalues as well as the eigenvectors. When both eigen systems are the same (i.e., $\tilde{\boldsymbol{v}}_i = \boldsymbol{v}_i$), then the divergence becomes the usual relative entropy between the eigenvalues $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \sum_i \tilde{\lambda}_i \log \frac{\tilde{\lambda}_i}{\lambda_i}$.

2.3 Rotation Invariance

One can visualize a symmetric positive definite matrix $\boldsymbol{W} = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$ as an ellipse, where the eigenvectors $\boldsymbol{v}_i$ are the axes of the ellipse and the square-roots of the eigenvalues (i.e. $\sqrt{\lambda_i}$) are the lengths of the corresponding axes. Thus the von Neumann divergence quantifies the "discrepancy" between two ellipses and is invariant under a simultaneous rotation of both eigen systems. That is, for any orthonormal matrix $\boldsymbol{U}$, the von Neumann divergence has the property that

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}\boldsymbol{U}^\top, \boldsymbol{U}\boldsymbol{W}\boldsymbol{U}^\top). \tag{2.5}$$

This follows from (2.4) and

$$\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{V}}\widetilde{\Lambda}\widetilde{\boldsymbol{V}}^\top, \boldsymbol{V}\Lambda\boldsymbol{V}^\top) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{V}}\widetilde{\Lambda}(\boldsymbol{U}\widetilde{\boldsymbol{V}})^\top, \boldsymbol{U}\boldsymbol{V}\Lambda(\boldsymbol{U}\boldsymbol{V})^\top).$$

However, the divergence is decidedly not invariant under the unitary rotation of both parameters, i.e. typically $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) \neq \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}, \boldsymbol{U}\boldsymbol{W})$ for an orthonormal matrix $\boldsymbol{U}$. This is because such rotations can change the sign of the eigenvalues. Also rotating symmetric matrices typically produces non-symmetric matrices.

There is a second important divergence between symmetric positive definite matrices that is invariant under the simultaneous rotation of both eigen systems (2.5). It is a Bregman divergence based on the strictly convex function $\mathrm{F}(\boldsymbol{W}) = -\log\det(\boldsymbol{W})$ (e.g., Boyd and Vandenberghe (2004)) over the cone of positive definite matrices. Note that $\mathrm{F}(\boldsymbol{W}) = -\sum_i \log \lambda_i$, where the λ_i denote the eigenvalues of $\boldsymbol{W}$. Also since $\boldsymbol{f}(\boldsymbol{W}) = \nabla_{\boldsymbol{W}} F(\boldsymbol{W}) = (\boldsymbol{W}^{-1})^\top = \boldsymbol{W}^{-1}$, the Bregman divergence becomes:

$$\begin{aligned}\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) &= \log \frac{\det(\boldsymbol{W})}{\det(\widetilde{\boldsymbol{W}})} + \mathrm{tr}(\boldsymbol{W}^{-1}\widetilde{\boldsymbol{W}}) - d \\ &= \sum_i \log \frac{\lambda_i}{\tilde{\lambda}_i} + \mathrm{tr}(\boldsymbol{W}^{-1}\widetilde{\boldsymbol{W}}) - d,\end{aligned}$$

where d is the dimension of the parameter matrices. We call this the *LogDet* divergence. Notice that in this case, $\mathrm{F}(\boldsymbol{W})$ is essentially minus the log of the volume of the ellipse $\boldsymbol{W}$, and the LogDet divergence is the relative entropy between two multidimensional Gaussians with fixed mean and covariance matrices $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{W}$, respectively (see Singer and Warmuth, 1999). At the end of Section 3.1 we will also briefly discuss the updates derived from the LogDet divergence. Note that for this divergence $\Delta_{\mathrm{F}}(\widetilde{\boldsymbol{W}}, \boldsymbol{W}) = \Delta_{\mathrm{F}}(\boldsymbol{U}\widetilde{\boldsymbol{W}}, \boldsymbol{U}\boldsymbol{W})$ for any orthonormal matrix $\boldsymbol{U}$ and parameter matrices in the domain of F.

1. $\mathrm{F}(\boldsymbol{W})$ can be extended to symmetric positive semi-definite matrices by using the convention $0 \log 0 = 0$.
2. The domain of the first argument can be extended to symmetric positive semi-definite matrices.

3. On-line Learning

In this section we present a natural extension of the *exponentiated gradient* (EG) update (Kivinen and Warmuth, 1997) to an update for symmetric positive definite matrices.

3.1 Motivation of the Updates

On-line learning proceeds in trials. In the most basic form, the on-line algorithm produces a parameter $\boldsymbol{W}_t$ at trial t and then incurs a loss $L_t(\boldsymbol{W}_t)$. In this paper, the parameters are square matrices in $\mathbb{R}^{d\times d}$.

In a refined form, the algorithm aims to predict a label and several actions occur in each trial: The algorithm first receives an *instance* $\boldsymbol{X}_t$ in some instance domain $\mathcal{X}$. It then produces a prediction $\hat{y}_t$ for the instance $\boldsymbol{X}_t$ based on the algorithm's current parameter matrix $\boldsymbol{W}_t$ and receives a label y_t. (The prediction $\hat{y}_t$ and the label y_t lie some labeling domain $\mathcal{Y}$.) Finally the algorithm incurs a real valued loss $L(\hat{y}_t, y_t)$ and updates its parameter matrix to $\boldsymbol{W}_{t+1}$.

For example in Section 3.3 we consider a case where the labeling domain $\mathcal{Y}$ is the real line. The on-line algorithm we analyze for this case predicts with $\hat{y}_t = \mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t)$ and is based on the loss $L_t(\boldsymbol{W}_t) = L(\hat{y}_t, y_t) = (\hat{y}_t - y_t)^2$.

In this section we only discuss updates at a high level and only consider the basic form of the on-line algorithm. We assume that $L_t(\boldsymbol{W})$ is convex in the parameter $\boldsymbol{W}$ (for all t) and that the gradient $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W})$ is a well defined matrix in $\mathbb{R}^{d\times d}$. In the update, we aim to solve the following problem (see Kivinen and Warmuth, 1997, 2001):

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}}{\mathrm{argmin}} \quad \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}), \tag{3.1}$$

where the convex function F defines the Bregman divergence and η is a non-negative learning rate. The update balances two conflicting goals: staying close to the old parameter $\boldsymbol{W}_t$ (as quantified by the divergence) and achieving small loss on the current labeled instance. The learning rate becomes a trade-off parameter.

We can eliminate the argmin by setting the gradient (with respect to $\boldsymbol{W}$) of its objective to zero:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})\right). \tag{3.2}$$

If we assume that $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ preserve symmetry, then constraining $\boldsymbol{W}$ in (3.1) to be symmetric changes the update to (cf. Appendix B for details):

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))\right). \tag{3.3}$$

The above *implicit* update is usually not solvable in closed form. A common way to avoid this problem (Kivinen and Warmuth, 1997) is to approximate $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})$ by $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t)$, leading to the following *explicit* update for the constraint case:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right).$$

In the case of the von Neumann divergence, the functions $\boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\,\boldsymbol{W}$ and $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = \mathbf{exp}\,\boldsymbol{Q}$ clearly preserve symmetry. When using this divergence we arrive at the following (explicit) update:

$$\boldsymbol{W}_{t+1} = \underbrace{\mathbf{exp}\left(\underbrace{\mathbf{log}\overbrace{\boldsymbol{W}_t}^{\text{sym.pos.def.}} - \eta\, \mathbf{sym}(\overbrace{\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t)}^{\text{pos. semi. def.}})}_{\text{symmetric}}\right)}_{\text{symmetric positive definite}}. \tag{3.4}$$

We call this update the *unnormalized matrix exponentiated gradient update*. Note that $\boldsymbol{f}(\boldsymbol{W}) = \mathbf{log}\,\boldsymbol{W}$ maps symmetric positive definite matrices to arbitrary symmetric matrices, and after adding a scaled symmetrized gradient, the function $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = \mathbf{exp}\,\boldsymbol{Q}$ maps the symmetric exponent back to a symmetric positive definite matrix.

When the parameters are constrained to trace one, then we arrive at the *M̲atrix E̲xponentiated G̲radient (MEG) update*, which generalizes the exponentiated gradient (EG) update of Kivinen and Warmuth (1997) to non-diagonal matrices:

$$\boldsymbol{W}_{t+1} = \frac{1}{Z_t}\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\,\mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right), \tag{3.5}$$

where $Z_t = \operatorname{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_t - \eta\,\mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_t))\right)\right)$ is the normalizing constant (See Appendix B for details.)

Finally, observe that for the LogDet divergence $\boldsymbol{f}(\boldsymbol{W}) = \nabla_{\boldsymbol{W}} F = -\boldsymbol{W}^{-1}$ and $\boldsymbol{f}^{-1}(\boldsymbol{Q}) = -\boldsymbol{Q}^{-1}$. Thus both $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ negate and invert all eigenvalues. Both functions also preserve symmetry. However, $\boldsymbol{f}^{-1}$ does not map an arbitrary symmetric matrix back to a symmetric positive definite matrix. Note that for this divergence update (3.3) becomes

$$\boldsymbol{W}_{t+1} = \underbrace{-\left(\underbrace{-(\overbrace{\boldsymbol{W}_t}^{\text{sym.pos.def.}})^{-1} - \eta\,\mathbf{sym}(\overbrace{\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})}^{\text{pos.semi.def.}})}_{\text{symmetric negative definite}}\right)^{-1}}_{\text{symmetric positive definite}}.$$

This update also preserves symmetric positive definiteness of the parameter matrix under the assumption that the gradient $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})$ is positive semi-definite: If $\boldsymbol{W}_t$ is symmetric positive definite, then $\boldsymbol{f}(\boldsymbol{W}_t)$ is symmetric negative definite. Using this assumption, we have that the argument of $\boldsymbol{f}^{-1}$ is symmetric negative definite and therefore $\boldsymbol{W}_{t+1}$ is again symmetric positive definite.

In this paper we prove a certain type of relative loss bound for the MEG update which generalize the analogously known bounds for the EG algorithm to the non-diagonal case. To our knowledge, no relative loss bounds have been proven for the above update that is derived from the LogDet divergence. For this update, such bounds are not even known for the diagonal case. Also, if the gradients of the loss are only known to be symmetric then η must be small in order to guarantee that $\boldsymbol{W}_{t+1}$ stays in the positive definite cone.

3.2 Numerically Stable MEG Update

The MEG update (3.5) is numerically unstable when the eigenvalues of $\boldsymbol{W}_t$ are around zero. However we can "unwrap" this update to the following:

$$\boldsymbol{W}_{t+1} = \frac{1}{\tilde{Z}_t}\mathbf{exp}\left(c_t \boldsymbol{I} + \mathbf{log}\,\boldsymbol{W}_1 - \eta \sum_{s=1}^{t} \mathbf{sym}(\nabla_{\boldsymbol{W}} L_s(\boldsymbol{W}_s))\right),$$

where the constant $\tilde{Z}_t$ normalizes the trace of $\boldsymbol{W}_{t+1}$ to one. As long as the eigenvalues of $\boldsymbol{W}_1$ are not too small, the computation of $\mathbf{log}\,\boldsymbol{W}_1$ is stable. Note that the update is independent of the choice of $c_t \in \mathbb{R}$. We incrementally maintain an eigenvalue decomposition of the matrix in the exponent ($O(n^3)$ per iteration):

$$\boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t^\top = c_t \boldsymbol{I} + \mathbf{log}\,\boldsymbol{W}_1 - \eta \sum_{s=1}^{t} \mathbf{sym}(\nabla_{\boldsymbol{W}} L_s(\boldsymbol{W}_s))$$

where the constant c_t is chosen so that the maximum eigenvalue of the above is zero. Now $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t)\boldsymbol{V}_t^\top/\mathrm{tr}(\mathbf{exp}(\Lambda_t))$. The pseudo-code is given in Algorithm 1.

Algorithm 1 Pseudo-code of the matrix exponentiated gradient (MEG) algorithm for quadratic Loss

```
Choose $\boldsymbol{W}_1$ and $\eta$
Initialize $\boldsymbol{G}_0 = \mathbf{log}\,\boldsymbol{W}_1$
for $t = 1, 2, \ldots$ do
    Obtain instance matrix $\boldsymbol{X}_t$
    Predict $\hat{y}_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t)$
    Obtain label $y_t$ and determine the loss $L_t = (y_t - \hat{y}_t)^2$
    Update $\boldsymbol{G}_t = \boldsymbol{G}_{t-1} - 2\eta(\hat{y}_t - y_t)\,\mathbf{sym}(\boldsymbol{X}_t)$
    Compute spectral decomposition: $\boldsymbol{G}_t = \boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t^\top$
    Update $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t - c_t\boldsymbol{I})\boldsymbol{V}_t^\top/\mathrm{tr}(\mathbf{exp}(\Lambda_t - c_t\boldsymbol{I}))$, where $c_t = \max_s(\Lambda_t)_{s,s}$
end for
```

3.3 Relative Loss Bounds

For the sake of simplicity we now restrict ourselves to the case when the algorithm predicts with $\hat{y}_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t)$ and the loss function is quadratic: $L_t(\boldsymbol{W}_t) = L(\hat{y}_t, y_t) := (\hat{y}_t - y_t)^2$.

We begin with the definitions needed for the relative loss bounds. Let $S = (\boldsymbol{X}_1, y_1), \ldots, (\boldsymbol{X}_T, y_T)$ denote a sequence of examples, where the instance matrices $\boldsymbol{X}_t \in \mathbb{R}^{d\times d}$ and the labels $y_t \in \mathbb{R}$. The total loss of the on-line algorithm on the entire sequence S is $L_{MEG}(S) = \sum_{t=1}^{t}(\mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) - y_t)^2$. We prove a bound on the *relative loss* $L_{MEG}(S) - L_{\boldsymbol{U}}(S)$ that holds for any comparator parameter $\boldsymbol{U}$. Such a comparator parameter is any symmetric positive semi-definite matrix $\boldsymbol{U}$ with trace one, and its total loss is defined as $L_{\boldsymbol{U}}(S) = \sum_{t=1}^{T}(\mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t) - y_t)^2$. The relative loss bound is derived in two steps: Lemma 3.1 upper bounds the relative loss for an individual trial in terms of the progress towards the comparator parameter $\boldsymbol{U}$ (as measured by the divergence). In the second Lemma 3.2, the bound for individual trials is summed to obtain a bound for a whole sequence. These two lemmas generalize similar lemmas previously proven for the exponentiated gradient update (Lemmas 5.8 and 5.9 of Kivinen and Warmuth, 1997).

Lemma 3.1 *Let $\boldsymbol{W}_t$ be any symmetric positive definite matrix. Let $\boldsymbol{X}_t$ be any square matrix for which the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$ have range at most r, i.e.*

$$\lambda^{\max}(\mathbf{sym}(\boldsymbol{X}_t)) - \lambda^{\min}(\mathbf{sym}(\boldsymbol{X}_t)) \leq r.$$

Assume $\boldsymbol{W}_{t+1}$ is produced from $\boldsymbol{W}_t$ by the MEG update with learning rate η, and let $\boldsymbol{U}$ be any symmetric positive semi-definite matrix. Then for any $b > 0$ and $a = \eta = 2b/(2 + r^2 b)$:

$$a\,\underbrace{(y_t - \mathrm{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2}_{\textit{MEG-loss}} - b\,\underbrace{(y_t - \mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2}_{\boldsymbol{U}\textit{-loss}} \leq \underbrace{\Delta_F(\boldsymbol{U}, \boldsymbol{W}_t) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1})}_{\textit{progress towards } \boldsymbol{U}}. \qquad (3.6)$$

The above type of inequality is central to all relative loss bounds (Kivinen and Warmuth, 1997). If the loss of the algorithm is small, then the inequality becomes vacuous. However, if the algorithm incurs a large loss, then its parameter $\boldsymbol{W}_t$ must make progress towards any parameter vector $\boldsymbol{U}$ that has small loss on the current example (if such parameters exist).

The proof of this inequality is given in Appendix C. It has the same structure as the corresponding previous lemma proven for the exponentiated gradient algorithm, but now we apply the various matrix inequalities given at the end of Section 2.1 (in particular the Golden-Thompson inequality (2.3) and the approximation of the matrix exponential (Lemma 2.1)). These inequalities will also be essential for the analysis of *DefiniteBoost* in the next section.

Lemma 3.2 *Let S be any sequence of examples with square real matrices as instances and real labels, and let r be an upper bound on the range of eigenvalues of the symmetric part of each instance matrix of S. Let the initial parameter $\boldsymbol{W}_1$ and comparison parameter $\boldsymbol{U}$ be arbitrary symmetric positive definite matrices of trace one. Then for any c such that* $\eta = 2c/(r^2(2+c))$,

$$L_{MEG}(S) \le \left(1+\frac{c}{2}\right) L_{\boldsymbol{U}}(S) + \left(\frac{1}{2}+\frac{1}{c}\right) r^2 \Delta_F(\boldsymbol{U},\boldsymbol{W}_1). \tag{3.7}$$

Proof For the maximum tightness of (3.6), a should be chosen as $a = \eta = 2b/(2+r^2 b)$. Let $b = c/r^2$, and thus $a = 2c/(r^2(2+c))$. Then (3.6) is rewritten as

$$\frac{2c}{2+c}(y_t - \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2 - c(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2 \le r^2(\Delta_F(\boldsymbol{U},\boldsymbol{W}_t) - \Delta_F(\boldsymbol{U},\boldsymbol{W}_{t+1}))$$

Adding the bounds for $t = 1,\cdots,T$, we get

$$\frac{2c}{2+c}L_{MEG}(S) - cL_{\boldsymbol{U}}(S) \le r^2(\Delta_F(\boldsymbol{U},\boldsymbol{W}_1) - \Delta_F(\boldsymbol{U},\boldsymbol{W}_{t+1})) \le r^2\Delta_F(\boldsymbol{U},\boldsymbol{W}_1),$$

which is equivalent to (3.7). ■

Assuming $L_{\boldsymbol{U}}(S) \le L_{\max}$ and $\Delta_F(\boldsymbol{U},\boldsymbol{W}_1) \le d_{\max}$, then the bound (3.7) is tightest when $c = r\sqrt{2d_{\max}/L_{\max}}$. With this choice of c, we have

$$L_{MEG}(S) - L_{\boldsymbol{U}}(S) \le r\sqrt{2L_{\max}d_{\max}} + \frac{r^2}{2}\Delta_F(\boldsymbol{U},\boldsymbol{W}_1).$$

In particular, if $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, then $\Delta_F(\boldsymbol{U},\boldsymbol{W}_1) = \log d - \sum_i \lambda_i \log\frac{1}{\lambda_i} \le \log d$. Additionally, when $L_{\max} = 0$, then the total loss of the algorithm is bounded by $\frac{r^2\log d}{2}$.

Note that the MEG algorithm generalizes the EG algorithm of Kivinen and Warmuth (1997). In the case of linear regression, a square of a product of dual norms appears in the bounds for the EG algorithm: $||\boldsymbol{u}||_1^2 X_\infty^2$. Here $\boldsymbol{u}$ is a parameter *vector* and X_∞ is an upper bound on the infinity norm of the instance vectors $\boldsymbol{x}_t$. Note the correspondence with the above bound (which generalizes the bounds for EG to the non-diagonal case): the one norm of the parameter vector is replaced by the trace and the infinity norm by the maximum range of the eigenvalues.

4. Bregman Projection and *DefiniteBoost*

Using the von Neumann divergence, we will generalize the boosting algorithms for matrix parameters.

4.1 Preliminaries

In this section, we address the following Bregman projection problem of finding a positive semi-definite symmetric matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ of trace one satisfying a set of linear constraints:[3]

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) \\ \text{s.t.} \quad & \boldsymbol{W} = \boldsymbol{W}^\top, \operatorname{tr}(\boldsymbol{W}) = 1 \\ & \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}_j) \le 0, \text{ for } j = 1, \dots, n, \end{aligned} \tag{4.1}$$

where the symmetric positive definite matrix $\boldsymbol{W}_1$ of trace one is the initial parameter matrix and $\boldsymbol{C}_1, \dots, \boldsymbol{C}_n$ are arbitrary matrices. Note that we do not explicitly constrain $\boldsymbol{W}$ to be positive semi-definite because when the von Neumann divergence is used, then the solution $\boldsymbol{W}^*$ will always be positive semi-definite. Prior knowledge about $\boldsymbol{W}$ is encoded in the constraints, and the matrix closest to $\boldsymbol{W}_1$ is chosen among the matrices satisfying all constraints. Tsuda and Noble (2004) employed this approach for learning a kernel matrix among graph nodes, and this method can be potentially applied to learn a kernel matrix in other settings (e.g., Xing et al., 2003; Tsang and Kwok, 2003). In the previous work by (Tsuda and Noble, 2004), an algorithm was developed that processes a batch of constraints. The problem was converted to a dual unconstraint problem (as done below) and an iterative gradient descent algorithm was given. However, no convergence proofs were provided previously. In this paper we give on-line algorithms with strong convergence proofs.[4]

The problem (4.1) is a projection of $\boldsymbol{W}_1$ to the intersection of convex regions defined by the constraints. It is well known that the Bregman projection into the intersection of convex regions can be solved by sequential projections to each region (Bregman, 1967; Censor and Lent, 1981). In the original papers only asymptotic convergence was shown. More recently a connection (Kivinen and Warmuth, 1999; Lafferty, 1999) was made to the AdaBoost algorithm which has an improved convergence analysis (Freund and Schapire, 1997; Schapire and Singer, 1999). We generalize the latter algorithm and its analysis to symmetric positive definite matrices and call the new algorithm *DefiniteBoost*. As in the original setting, only *approximate* projections (Figure 1) are required to show fast convergence.

Before presenting the algorithm, let us describe the dual problem of minimizing the von Neumann divergence subject to linear constraints (4.1). The dual variables are the Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^n$ ($\boldsymbol{\alpha} \ge \mathbf{0}$) associated with this optimization problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha} \ge \mathbf{0}} \quad -\log\left\{ \operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j\, \mathbf{sym}(\boldsymbol{C}_j)) \right) \right\}. \tag{4.2}$$

See Appendix D for a detailed derivation of the dual problem that handles the case when the constraint matrix $\boldsymbol{C}_j$ is allowed to be an arbitrary square matrix. Previous derivations required symmetric $\boldsymbol{C}_j$ (Tsuda and Noble, 2004). When (4.1) is feasible, the optimal solution is described as

$$\boldsymbol{W}^* = \frac{1}{Z(\boldsymbol{\alpha}^*)} \mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j^*\, \mathbf{sym}(\boldsymbol{C}_j)),$$

3. Note that if η is large then the on-line update (3.1) becomes a Bregman projection subject to a single equality constraint $\operatorname{tr}(\boldsymbol{W}\boldsymbol{X}_t) = y_t$.
4. The methodology employed in this paper is not limited to on-line learning. For example in Littlestone et al. (1992), cf. Corollary 15, the EG algorithm was used for solving a system of linear equations and fast convergence was shown.

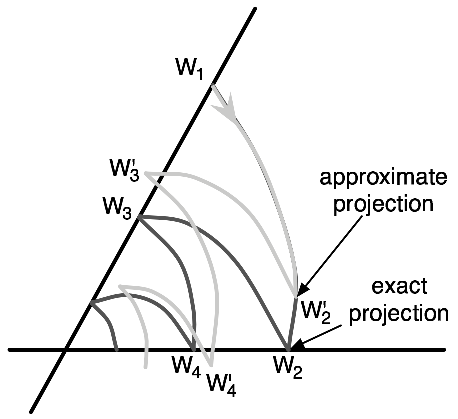

Figure 1: The intersection of two convex sets (here two straight lines) can be found by projecting back and forth between the two sets with exact Bregman projections ($\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots$). In this paper we use certain approximate projections ($\boldsymbol{W}_1, \boldsymbol{W}_2', \ldots$). Now each projection may over or undershoot the alternating target set. Nevertheless, global convergence to the optimal solution is still guaranteed via our proofs.

where $\mathrm{Z}(\boldsymbol{\alpha}^*) = \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^n \alpha_j^* \,\mathbf{sym}(\boldsymbol{C}_j))\right)$ and $\boldsymbol{\alpha}^*$ is the optimal dual solution.

4.2 Exact Bregman Projections

Problem (4.1) can be solved with the following algorithm: Start from some initial parameter $\boldsymbol{W}_1$ (for instance $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$). At the t-th step, choose an unsatisfied constraint j_t, i.e. $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) > 0$.[5] Then solve the following Bregman projection with respect to the chosen constraint:

$$\begin{aligned} \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_F(\boldsymbol{W}, \boldsymbol{W}_t) \\ \text{s.t.} \quad & \boldsymbol{W} = \boldsymbol{W}^\top, \mathrm{tr}(\boldsymbol{W}) = 1, \\ & \mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_{j_t}) \le 0. \end{aligned} \tag{4.3}$$

By means of a Lagrange multiplier α, the dual problem is described as (cf. Appendix D)

$$\alpha_t^* = \operatorname*{argmin}_{\alpha \ge 0} \quad \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right). \tag{4.4}$$

Using the solution of the dual problem, $\boldsymbol{W}_t$ is updated as

$$\boldsymbol{W}_{t+1} = \frac{1}{\mathrm{Z}_t(\alpha_t^*)} \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \alpha_t^* \,\mathbf{sym}(\boldsymbol{C}_{j_t})) \tag{4.5}$$

where the normalization factor is $\mathrm{Z}_t(\alpha_t^*) = \mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \alpha_t^* \,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right)$. If $\boldsymbol{W}_t$ is symmetric positive definite, then $\boldsymbol{W}_{t+1}$ is as well. Note that we can use the same numerically stable reformulation of the update as discussed in Section 3.2.

5. For instance, the most unsatisfied constraint, i.e. $j_t = \mathrm{argmax}_{j=1,\cdots,n}\,\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_j)$, can be chosen.

4.3 Approximate Bregman Projections

The solution of (4.4) cannot be obtained in closed form. However, one can use the following approximate choice of α_t:

$$\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right), \tag{4.6}$$

when the eigenvalues of $\mathbf{sym}(\boldsymbol{C}_{j_t})$ lie in the interval $[\lambda_t^{\min}, \lambda_t^{\max}]$ and $r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t})$. Since the most unsatisfied constraint is chosen, $r_t \geq 0$ and thus $\widehat{\alpha}_t \geq 0$. We call this approximate Bregman projection algorithm *DefiniteBoost*. It may be seen as a natural extension of AdaBoost (cf. Section 4.5), where probability distributions are replaced by symmetric positive definite matrices of trace one. The pseudo-code of DefiniteBoost is given in Algorithm 2.

Algorithm 2 Pseudo-code of the DefiniteBoost algorithm; $\lambda_t^{\min}$ and $\lambda_t^{\max}$ are lower and upper bounds on the eigenvalues of $\mathbf{sym}(\boldsymbol{C}_t)$.

```
Choose $\boldsymbol{W}_1$
Initialize $\boldsymbol{G}_0 = \mathbf{log}\,\boldsymbol{W}_1$
for $t = 1, 2, \ldots$ do
    Choose an unsatisfied constraint $j_t$ (i.e. $\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) > 0$) or stop when all constraints satisfied
    Compute constraint violation $r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t})$
    Compute approximate step size $\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right)$
    Update $\boldsymbol{G}_t = \boldsymbol{G}_{t-1} - \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t})$
    Compute spectral decomposition: $\boldsymbol{G}_t = \boldsymbol{V}_t \Lambda_t \boldsymbol{V}_t$
    Update $\boldsymbol{W}_{t+1} = \boldsymbol{V}_t\,\mathbf{exp}(\Lambda_t - c_t \boldsymbol{I}) \boldsymbol{V}_t^\top / \mathrm{tr}(\mathbf{exp}(\Lambda_t - c_t \boldsymbol{I}))$, where $c_t = \max_s (\Lambda_t)_{s,s}$
end for
```

Although the projection is done only approximately,[6] the convergence of the dual objective (4.2) can be shown using the following upper bound of the negative dual objective , i.e.

$$\mathrm{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j))\right).$$

Theorem 4.1 *The negative exponentiated dual objective is bounded from above by*

$$\mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^{T} \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t})\right)\right) \leq \prod_{t=1}^{T} \rho(r_t), \tag{4.7}$$

where

$$\widehat{\alpha}_t = \frac{1}{\lambda_t^{\max} - \lambda_t^{\min}} \log\left(\frac{1 + r_t/\lambda_t^{\max}}{1 + r_t/\lambda_t^{\min}}\right),\ r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}),$$

and

$$\rho(r_t) = \left(1 - \frac{r_t}{\lambda_t^{\max}}\right)^{\frac{\lambda_t^{\max}}{\lambda_t^{\max} - \lambda_t^{\min}}} \left(1 - \frac{r_t}{\lambda_t^{\min}}\right)^{\frac{-\lambda_t^{\min}}{\lambda_t^{\max} - \lambda_t^{\min}}}.$$

6. The approximate Bregman projection (with α_t as in (4.6)) can also be motivated as an on-line algorithm based on an entropic loss and learning rate one (following Section 3 and Kivinen and Warmuth (1999)).

The proof of this inequality for our setting is given in Appendix E. The bound (4.7) is monotonically decreasing, because $\rho(r_t) \leq 1$. Also, since we always chose a violated constraint (if there is one), we have $r_t > 0$ and therefore $\rho(r_t) < 1$ (or we stop). Thus the dual objective (4.2) continues to increase until all constraints are satisfied.

4.4 Convergence Speed

Next we determine the maximal number of iterations needed to find a matrix $\boldsymbol{W}$ which satisfies all constraints up to the predetermined accuracy ε, i.e. $\mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \varepsilon$, for $1 \leq j \leq n$. The algorithm selects in each iteration an constraint j_t that is violated by at least ε (i.e. $r_t = \mathrm{tr}(\boldsymbol{W}_t\boldsymbol{C}_{j_t}) \geq \varepsilon$), or stops if no such constraint exists. Assuming the algorithm stops at $(T+1)$-th step, we derive an upper bound on T as a function of ε.

For simplicity, let us assume $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, $\lambda_j^{\min} = -\lambda$, and $\lambda_j^{\max} = \lambda$ (for all j). Denote by $h_{primal}(\boldsymbol{W})$ and $h_{dual}(\boldsymbol{\alpha})$ the primal and dual objective functions in (4.1) and (4.2), respectively.

$$h_{primal}(\boldsymbol{W}) = \Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) \tag{4.8}$$

$$h_{dual}(\boldsymbol{\alpha}) = -\log \mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\, \mathbf{sym}(\boldsymbol{C}_j)\right)\right) \tag{4.9}$$

The primal objective is upper-bounded by $\log d$, since $\Delta_F(\boldsymbol{W}, \boldsymbol{W}_1) = \sum_i \lambda_i \log \lambda_i + \log d \leq \log d$. Since the algorithm stops at the $(T+1)$-th iteration (with $r_t \geq \varepsilon$ for $t = 1, \ldots, T$), we get from Theorem 4.1:

$$\exp(-h_{dual}(\tilde{\boldsymbol{\alpha}})) = \mathrm{tr}\left(\mathbf{exp}\left(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^{T} \widehat{\alpha}_t\, \mathbf{sym}(\boldsymbol{C}_{j_t})\right)\right) \leq \left(\frac{\lambda^2 - \varepsilon^2}{\lambda^2}\right)^{T/2},$$

where $\tilde{\boldsymbol{\alpha}}$ is the cumulative coefficient vector for the constraints, i.e. $\widetilde{\alpha}_j = \sum_{t=1}^{T} \widehat{\alpha}_t \delta(j_t = j)$, for $1 \leq j \leq n$.

Thus the objective in (4.2) is lower bounded by $\frac{1}{2}T\frac{\varepsilon^2}{\lambda^2}$, since

$$\begin{aligned} h_{dual}(\widetilde{\boldsymbol{\alpha}}) &\geq -\log\left(\frac{\lambda^2 - \varepsilon^2}{\lambda^2}\right)^{T/2} \\ &\geq \frac{T\varepsilon^2}{2\lambda^2}, \end{aligned} \tag{4.10}$$

where the last inequality follows by convexity of $-\log\left(\frac{\lambda^2-\varepsilon^2}{\lambda^2}\right)$ with respect to ε. At the optimal solution $\boldsymbol{W}^*$ and $\boldsymbol{\alpha}^*$, the values of the objective functions coincide, i.e. $h_{dual}(\boldsymbol{\alpha}^*) = h_{primal}(\boldsymbol{W}^*)$. Finally, we obtain

$$\frac{T\varepsilon^2}{2\lambda^2} \leq h_{dual}(\widetilde{\boldsymbol{\alpha}}) \leq h_{dual}(\boldsymbol{\alpha}^*) = h_{primal}(\boldsymbol{W}^*) \leq \log d,$$

and the upper bound $T \leq \frac{2\lambda^2 \log d}{\varepsilon^2}$. In summary, we have proven the following:

Corollary 4.2 *Suppose we are solving problem* (4.1) *with DefiniteBoost, where* $\boldsymbol{C}_j$ *(*$j = 1, \ldots, n$*) are arbitrary matrices with* $\lambda_{\min}(\boldsymbol{C}_j) \geq -\lambda$ *and* $\lambda_{\max}(\boldsymbol{C}_j) \leq \lambda$ *and* $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$. *Assume an optimal solution* $\boldsymbol{W}^*$ *to* (4.1) *exists and the algorithm selects in each iteration an* ε*-violated constraint, i.e.*

$r_t = \mathrm{tr}(\boldsymbol{W}_t \boldsymbol{C}_{j_t}) \geq \varepsilon$, or stops if no such constraint exists. Then after at most $T = \frac{2\lambda^2 \log d}{\varepsilon^2}$ iterations, DefiniteBoost stops and the resulting $\boldsymbol{W}$ satisfies all linear constraints up to accuracy ε, i.e.

$$\mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \varepsilon \quad \textit{for all } j = 1, \ldots, n.$$

This result implies that we can solve (4.1) with accuracy ε in $O(d^3 \log d/\varepsilon^2)$ operations (excluding the cost of identifying violated constraints). Similar bounds on the number of iterations for solving a system of linear equations with the EG algorithm were first proven in (Littlestone et al., 1992, Corollary 15). Observe that if (4.1) is not feasible, then one may continue finding ε-violated constraints and the primal objective can become unbounded, i.e. $\sum_t \widehat{\alpha}_t$ may become unbounded.

4.5 Relation to Boosting

When all matrices are diagonal, then DefiniteBoost specializes to the AdaBoost algorithm (Schapire and Singer, 1999). Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^d$ be the training samples, where $\boldsymbol{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$. Let $h_1(x), \ldots, h_n(x) \in [-1, 1]$ be the weak hypotheses. For the j-th hypothesis $h_j(x)$, let us define $\boldsymbol{C}_j = \mathrm{diag}(y_1 h_j(x_1), \ldots, y_d h_j(x_d))$. Since $|y h_j(x)| \leq 1$, we may choose $\lambda_t^{\max} = 1$ and $\lambda_t^{\min} = -1$ for any t. Setting $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$, the dual objective (4.7) is rewritten as

$$-\log\left(\frac{1}{d}\sum_{i=1}^{d} \exp\left(-y_i \sum_{j=1}^{n} \alpha_j h_j(x_i)\right)\right),$$

which is equivalent to the exponential loss function used in AdaBoost. Since $\boldsymbol{C}_j$ and $\boldsymbol{W}_1$ are diagonal, the matrix $\boldsymbol{W}_t$ stays diagonal after the update. If $w_{t,i} = (\boldsymbol{W}_t)_{i,i}$, the updating formula (4.5) becomes the AdaBoost update: $w_{t+1,i} = w_{t,i} \exp(-\alpha_t y_i h_t(x_i))/Z_t(\alpha_t)$. The approximate solution of α_t (4.6) is described as $\alpha_t = \frac{1}{2}\log\frac{1+r_t}{1-r_t}$, where r_t is the weighted training error of the t-th hypothesis, i.e. $r_t = \sum_{i=1}^d w_{t,i} y_i h_t(x_i)$.

4.6 Solving Semi-definite Programs

Suppose we aim to solve the following semi-definite programming problem:

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W},\theta} \quad & \theta \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{W}) = 1, \boldsymbol{W} \succeq \boldsymbol{0}, \boldsymbol{W} = \boldsymbol{W}^\top \\ & \mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq \theta, \text{ for } j = 1, \ldots, n. \end{aligned} \tag{4.11}$$

If one would know the optimal θ^* beforehand, then following problem would lead to an optimal solution of (4.11):

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_F(\boldsymbol{W}, \frac{1}{d}\boldsymbol{I}) \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{W}) = 1, \boldsymbol{W} = \boldsymbol{W}^\top \\ & \mathrm{tr}(\boldsymbol{W}(\boldsymbol{C}_j - \theta^*\boldsymbol{I})) \leq 0, \text{ for } j = 1, \ldots, n. \end{aligned} \tag{4.12}$$

Running DefiniteBoost on the above problem with matrices $\widetilde{\boldsymbol{C}}_j = (\boldsymbol{C}_j - \theta^*\boldsymbol{I})$ can approximate the solution of (4.12) rather efficiently and, hence, it is only left to determine the optimal value

θ^*. If it is chosen too small, then no feasible solution to (4.12) exists and DefiniteBoost will not terminate after $2\lambda^2 \log d/\varepsilon^2$ iterations with accuracy ε,[7] where $\lambda^{\min}(\widetilde{C}_j) \geq -\lambda$ and $\lambda^{\max}(\widetilde{C}_j) \leq \lambda$. If it is chosen too large, then a feasible solution exists and DefiniteBoost terminates in a bounded number of iterations. Hence one has a way of identifying when $\theta < \theta^*$ and also $\theta > \theta^*$. This allows the design of a binary search procedure to approximate θ^* in a few steps. Based on this idea we previously proposed a margin maximizing version of AdaBoost (Rätsch and Warmuth, 2002). For this algorithm we could show that after $O(\log d \log(1/\varepsilon)/\varepsilon^2)$ iterations the algorithm achieved an optimal solution within accuracy ε. We claim that the outlined binary search procedure can also be applied in combination with DefiniteBoost for solving the semi-definite problem (4.11) in time $O(d^3 \log d \log(1/\varepsilon)/\varepsilon^2)$ (excluding the cost of identifying violated constraints). Additionally we assert that a slightly more advanced adaptation of θ during the optimization (as was done by Rätsch, 2001; Rätsch and Warmuth, 2005, for the diagonal case) will yield the reduced time complexity of $O(d^3 \log d/\varepsilon^2)$. Rigorous proofs of these conjectures go beyond the scope of this paper.

5. Experiments on Learning Kernels

In this section, our technique is applied to learning a kernel matrix from a set of distance measurements. This application is not on-line *per se*, but it shows nevertheless that the theoretical bounds can be reasonably tight on natural data.

When $\boldsymbol{K}$ is a $d \times d$ kernel matrix among d objects, then the K_{ij} characterizes the similarity between objects i and j. In the feature space, K_{ij} corresponds to the inner product between object i and j, and thus the Euclidean distance can be computed from the entries of the kernel matrix (Schölkopf and Smola, 2002). In some cases, the kernel matrix is not given explicitly, but only a set of distance measurements is available. The data are represented either as (i) quantitative distance values (e.g., the distance between i and j is 0.75), or (ii) qualitative evaluations (e.g., the distance between i and j is small) (Xing et al., 2003; Tsuda and Noble, 2004). Our task is to obtain a positive definite kernel matrix which fits well to the given distance data.

5.1 On-line Kernel Learning

In the first experiment, we consider the on-line learning scenario in which only one distance example is shown to the learner at each time step. The distance example at time t is described as $\{a_t, b_t, y_t\}$, which indicates that the squared Euclidean distance between objects a_t and b_t is y_t. Let us define a time-developing sequence of kernel matrices as $\{\boldsymbol{W}_t\}_{t=1}^T$, and the corresponding points in the feature space as $\{\boldsymbol{x}_{ti}\}_{i=1}^d$ (i.e. $(\boldsymbol{W}_t)_{ab} = \boldsymbol{x}_{ta}^\top \boldsymbol{x}_{tb}$). Then, the total loss incurred by this sequence is

$$\sum_{t=1}^T \left(\|\boldsymbol{x}_{ta_t} - \boldsymbol{x}_{tb_t}\|^2 - y_t\right)^2 = \sum_{t=1}^T (\mathrm{tr}(\boldsymbol{W}_t \boldsymbol{X}_t) - y_t)^2,$$

where $\boldsymbol{X}_t$ is a symmetric matrix whose (a_t, a_t) and (b_t, b_t) elements are 0.5, (a_t, b_t) and (b_t, a_t) elements are -0.5, and all the other elements are zero. We consider a controlled experiment in which the distance examples are created from a known *target kernel matrix*. We used a 52×52 kernel matrix among `gyrB` proteins of bacteria ($d = 52$). This data contains three bacteria species (see Tsuda et al., 2003, for details). Each distance example is created by randomly choosing one element of the target kernel. The initial parameter was set as $\boldsymbol{W}_1 = \frac{1}{d}\boldsymbol{I}$. When the comparison matrix U is set

7. This statement is slightly simplified. Please check Rätsch and Warmuth (2002) for details.

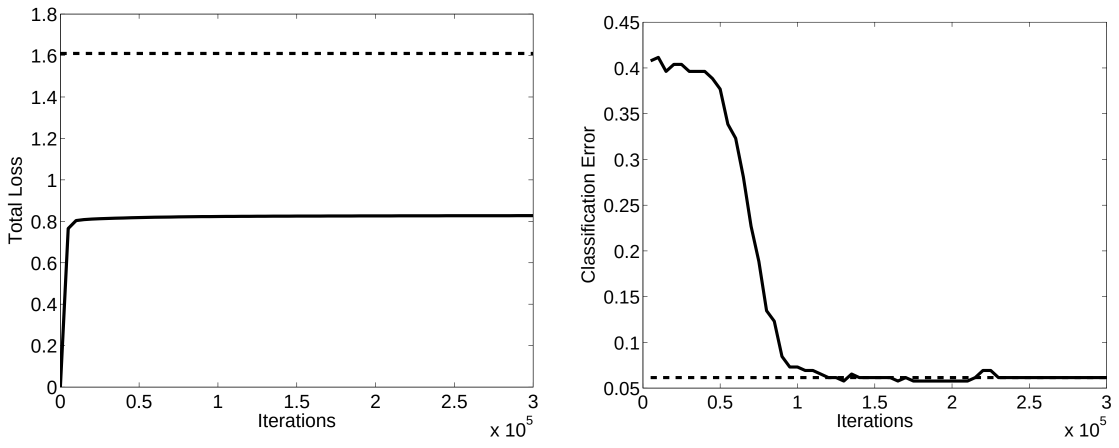

Figure 2: Numerical results of on-line learning. (Left) total loss against the number of iterations. The dashed line shows the loss bound. (Right) classification error of the nearest neighbor classifier using the learned kernel. The dashed line shows the error by the target kernel.

to the target matrix, then because all the distance examples are derived from this matrix, $L_U(S) = 0$ and $L_{\max} = 0$. Therefore we choose learning rate $\eta = 2$, which minimizes the relative loss bound of Lemma 3.2. The total loss of the kernel matrix sequence obtained by the matrix exponential update is shown in Figure 2 (left). In the plot, we have also shown the relative loss bound. The bound seems to give a reasonably tight performance guarantee—it is about twice the actual total loss. To evaluate the learned kernel matrix, the prediction accuracy of bacteria species by the nearest neighbor classifier is calculated (Figure 2, right), where the 52 proteins are randomly divided into 50% training and 50% testing data. The value shown in the plot is the test error averaged over 10 different divisions. It took a large number of iterations ($\sim 2 \times 10^5$) for the error rate to converge to the level of the target kernel. In practice one can often increase the learning rate for faster convergence, but here we chose the small rate suggested by our analysis to check the tightness of the bound.

5.2 Kernel Learning by Bregman Projection

Next, let us consider a batch learning scenario where we have a set of qualitative distance evaluations (i.e. inequality constraints). Given n pairs of similar objects $\{a_j, b_j\}_{j=1}^n$, the inequality constraints are constructed as $\|\boldsymbol{x}_{a_j} - \boldsymbol{x}_{b_j}\| \leq \gamma, j = 1, \ldots, n$, where γ is a predetermined constant. If $\boldsymbol{X}_j$ is defined as in the previous section and $\boldsymbol{C}_j = \boldsymbol{X}_j - \gamma \boldsymbol{I}$, the inequalities are then rewritten as $\mathrm{tr}(\boldsymbol{W}\boldsymbol{C}_j) \leq 0, j = 1, \ldots, n$. The largest and smallest eigenvalues of any $\boldsymbol{C}_j$ are $1-\gamma$ and $-\gamma$, respectively. As in the previous section, distance examples are randomly generated from the target kernel matrix between gyrB proteins. Setting $\gamma = 0.2/d$, we collected all object pairs whose distance in the feature space is less than γ to yield 980 inequalities ($n = 980$). Figure 3 (left) shows the convergence of the dual objective function as proven in Theorem 4.1. The convergence was much faster than the previous experiment, because in the batch setting, one can choose the most unsatisfied constraint and optimize the step size as well. Figure 3 (right) shows the classification error of the nearest

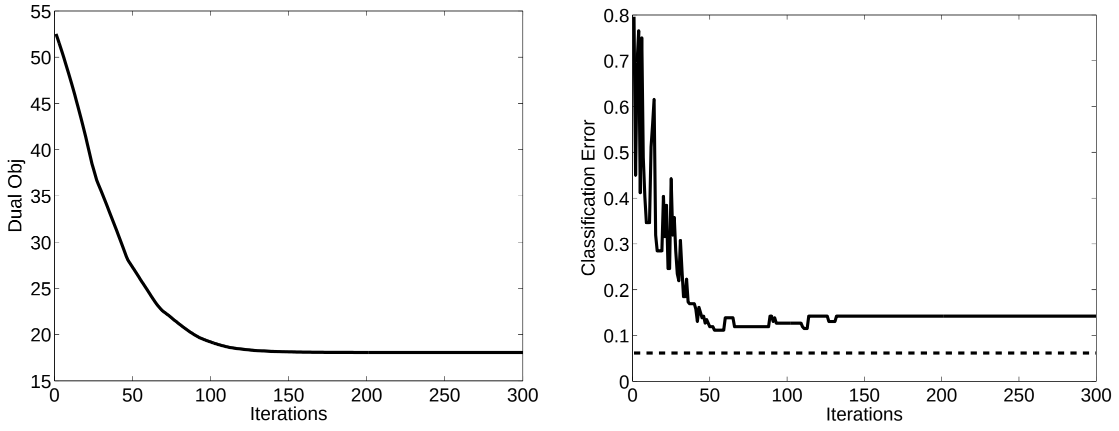

Figure 3: Numerical results of Bregman projection. (Left) convergence of the dual objective function. (Right) classification error of the nearest neighbor classifier using the learned kernel.

neighbor classifier. As opposed to the previous experiment, the error rate is higher than that of the target kernel matrix, because a substantial amount of information is lost by the conversion to inequality constraints.

6. Summary and Discussion

We motivated and analyzed a new update for symmetric positive matrices using the *von Neumann* divergence. We showed that the standard bounds for on-line learning and boosting generalize to the case when the parameters are symmetric positive definite matrices of trace one instead of a probability vector. As in quantum physics, the eigenvalues act as probabilities. In addition to the applications suggested by the experiments, our algorithm can be straightforwardly applied to learning a covariance matrix. It would also be interesting to use a robust loss $L_t(\boldsymbol{W})$ for the purpose of ignoring outliers (Huber, 1981) and investigate possible applications of our learning algorithms to quantum statistical inference problems (Barndorff-Nielsen et al., 2003).

Our method is designed for learning a positive definite parameter matrix of fixed size. It is not straightforward to extend it to the case where the size of the parameter matrix grows on-line as more examples are seen. Our methods immediately generalize to the Hermitian matrices, i.e. square matrices in $\mathbb{C}^{d\times d}$ for which $\boldsymbol{A} = \bar{\boldsymbol{A}}^\top = \boldsymbol{A}^*$. The spectral decomposition of these matrices becomes $\boldsymbol{A} = \boldsymbol{U}\Lambda\boldsymbol{U}^*$, where $\boldsymbol{U}$ is a unitary matrix (i.e. $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{I}$) and Λ is a diagonal matrix of *real* eigenvalues. In the case when all entries of the matrix are real, then Hermitian is equivalent to symmetric. All algorithms of this paper (and their analyzes) immediately generalize to the case when symmetric is replaced by Hermitian and symmetric positive definite by positive Hermitian (i.e. Hermitian with positive eigenvalues). In particular, the Golden-Thompson inequality, Jensen's inequality for the matrix exponential (Lemma 2.1) and Lemma 2.2 all hold for Hermitian matrices. Note that density matrices (as used in Statistical Physics) are positive Hermitian matrices of trace one.

Acknowledgments

We dedicate this paper to Nick Littlestone who first proved relative loss bounds for an algorithm in the EG family—his well known *Winnow* algorithm for learning disjunctions (Littlestone, 1988, 1989).

K.T. and G.R. gratefully acknowledge partial support from the PASCAL Network of Excellence (EU #506778). M.W. was supported by NSF grant CCR 9821087 and UC Discovery grant LSIT02-10110. This work was partially done while G.R. was still at Fraunhofer FIRST in Berlin and Max Planck Institute for Biological Cybernetics in Tübingen. Part of this work was done while all three authors were visiting the National ICT Australia in Canberra.

Appendix A. Derivatives of Matrix Functions

The matrix functions considered in this paper are mostly trace functions e.g. $\mathrm{tr}(\mathbf{exp}(\boldsymbol{W}))$ and $\mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W})$, which we will expand into power series. Thus we begin with computing the gradient of $\mathrm{F}(\boldsymbol{W}) = \mathrm{tr}(\boldsymbol{W}^k)$. The partial derivative with respect to (i,j) element is described as

$$\frac{\partial \mathrm{tr}(\boldsymbol{W}^k)}{\partial W_{ij}} = \lim_{\lambda\to 0} \frac{\mathrm{tr}((\boldsymbol{W}+\lambda \boldsymbol{E}_{ij})^k) - \mathrm{tr}(\boldsymbol{W}^k)}{\lambda},$$

where $\boldsymbol{E}_{ij}$ is the sparse matrix whose (i,j) element is one and all the others are zero. For example, when $k=3$,

$$(\boldsymbol{W}+\lambda\boldsymbol{E}_{ij})^3 = (\boldsymbol{W}^3 + \lambda\boldsymbol{E}_{ij}\boldsymbol{W}\boldsymbol{W} + \lambda\boldsymbol{W}\boldsymbol{E}_{ij}\boldsymbol{W} + \lambda\boldsymbol{W}\boldsymbol{W}\boldsymbol{E}_{ij}) + O(\lambda^2).$$

The trace is simply described as

$$\begin{aligned} \mathrm{tr}((\boldsymbol{W}+\lambda\boldsymbol{E}_{ij})^3) &= \mathrm{tr}(\boldsymbol{W}^3) + 3\lambda\mathrm{tr}(\boldsymbol{E}_{ij}\boldsymbol{W}^2) + O(\lambda^2) \\ &= \mathrm{tr}(\boldsymbol{W}^3) + 3\lambda[\boldsymbol{W}^2]_{j,i} + O(\lambda^2). \end{aligned}$$

Therefore, $\nabla_{\boldsymbol{W}}\mathrm{tr}(\boldsymbol{W}^3) = 3(\boldsymbol{W}^2)^\top$. For general k, we get

$$\nabla_{\boldsymbol{W}}\mathrm{tr}(\boldsymbol{W}^k) = k(\boldsymbol{W}^{k-1})^\top. \tag{A.1}$$

The matrix exponential is defined as

$$\mathbf{exp}(\boldsymbol{W}) = \boldsymbol{I} + \boldsymbol{W} + \frac{1}{2!}\boldsymbol{W}^2 + \frac{1}{3!}\boldsymbol{W}^3 + \cdots.$$

Applying (A.1) to all terms, we get $\nabla_{\boldsymbol{W}}\mathrm{tr}(\mathbf{exp}(\boldsymbol{W})) = \mathbf{exp}(\boldsymbol{W})^\top$. Next, let us calculate the gradient of $\mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W})$. Using the expansion

$$\mathbf{log}\,\boldsymbol{W} = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i}(\boldsymbol{W}-\boldsymbol{I})^i,$$

we get

$$\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W} = \sum_{i=2}^{\infty} \frac{(-1)^i}{i(i-1)}(\boldsymbol{W}-\boldsymbol{I})^i - \boldsymbol{I}.$$

Applying the shifted version of (A.1), i.e. $\nabla_{\boldsymbol{W}}\mathrm{tr}((\boldsymbol{W}-\boldsymbol{I})^k) = k((\boldsymbol{W}-\boldsymbol{I})^{k-1})^\top$, to all terms, the gradient is obtained as $\nabla_{\boldsymbol{W}}\mathrm{tr}(\boldsymbol{W}\,\mathbf{log}\,\boldsymbol{W} - \boldsymbol{W}) = (\mathbf{log}\,\boldsymbol{W})^\top$. When $\boldsymbol{W}$ is symmetric, then one can drop the transposition. Thus in in this case $\nabla_{\boldsymbol{W}}\mathrm{tr}(\mathbf{exp}\,\boldsymbol{W}) = \mathbf{exp}\,\boldsymbol{W}$.

Appendix B. Derivation of the MEG Update

In this appendix we derive parameter updates when the parameter must meet some linear constraints. One method is to incorporate such constraints into the strictly convex function F defining the Bregman divergence. The modified function F is then only defined when the constraints are met. The updates always have the simple form (3.2). However this method often leads to difficult forms of F and $\boldsymbol{f} = \nabla \mathrm{F}$. Here we choose the alternate method of keeping the linear constraints on the side. We begin by discussing how to enforce symmetry. Consider the following optimization problem, where $\boldsymbol{X}_t$ is an arbitrary matrix in $\mathbb{R}^{d\times d}$, $\boldsymbol{W}_t$ an arbitrary symmetric matrix in $\mathbb{R}^{d\times d}$ and $y_t \in \mathbb{R}$:

$$\begin{aligned} \boldsymbol{W}_{t+1} = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}) \\ \text{s.t. } \boldsymbol{W} = \boldsymbol{W}^\top. & \end{aligned}$$

We assume that $\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W})$ is always a well defined matrix in $\mathbb{R}^{d\times d}$.

We introduce one Lagrange multiplier $\boldsymbol{\Gamma}_{i,j}$ for the each of the constraints $\boldsymbol{W}_{i,j} = \boldsymbol{W}_{j,i}$. This contributes the term $\boldsymbol{\Gamma}_{i,j}(\boldsymbol{W}_{i,j} - \boldsymbol{W}_{j,i})$ to the Lagrangian. In matrix form these constraints can be summarized as $\mathrm{tr}(\boldsymbol{\Gamma}(\boldsymbol{W}^\top - \boldsymbol{W})) = \mathrm{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W})$. This gives us the Lagrangian

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{\Gamma}) = \Delta_{\mathrm{F}}(\boldsymbol{W}, \boldsymbol{W}_t) + \eta L_t(\boldsymbol{W}) + \mathrm{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W}).$$

for $\boldsymbol{\Gamma} \in \mathbb{R}^{d\times d}$. Setting the gradient with respect to $\boldsymbol{W}$ to zero yields:

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}) - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top)\right).$$

Since the objective is convex, it suffices to exhibit a choice of $\boldsymbol{\Gamma}$ such that the symmetry constraint is satisfied. Under the assumption that $\boldsymbol{f}$ and $\boldsymbol{f}^{-1}$ preserve symmetry, $\boldsymbol{\Gamma} = -\eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})/2$ achieves this and the update becomes (3.3):

$$\boldsymbol{W}_{t+1} = \boldsymbol{f}^{-1}\left(\boldsymbol{f}(\boldsymbol{W}_t) - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))^\top)\right).$$

For the normalized case we still need to enforce the trace one constraint on $\boldsymbol{W}_{t+1}$. This adds a term $\delta(\mathrm{tr}(\boldsymbol{W}) - 1)$ to the Lagrangian and the update now has the form

$$\boldsymbol{W}_{t+1} = \mathbf{exp}\left(\mathbf{log}\, \boldsymbol{W}_t - \eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}) - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top) - \delta \boldsymbol{I}\right).$$

Choosing $\boldsymbol{\Gamma} = -\eta \nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1})/2$ and

$$\delta = -\log\left(\mathrm{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t - \eta\, \mathbf{sym}(\nabla_{\boldsymbol{W}} L_t(\boldsymbol{W}_{t+1}))))\right)$$

enforces the symmetry and trace constraints and after approximating the gradient we arrive at the explicit MEG update (3.5).

Appendix C. Proof of Lemma 3.1

Let $\delta_t = -2\eta(\mathrm{tr}(\boldsymbol{X}\boldsymbol{W}_t) - y_t)$, then the right hand side of (3.6) can be reformulated as

$$\Delta_F(\boldsymbol{U}, \boldsymbol{W}_t) - \Delta_F(\boldsymbol{U}, \boldsymbol{W}_{t+1}) = \delta_t \mathrm{tr}(\boldsymbol{U}\boldsymbol{X}_t) - \log \mathrm{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))).$$

Therefore, (3.6) is equivalent to $f \leq 0$, where

$$f = \log \operatorname{tr}(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))) - \delta_t \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t) + a(y_t - \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2 - b(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2.$$

Let us bound the first term. Due to Golden-Thompson inequality (2.3), we have

$$\operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\, \boldsymbol{W}_t + \delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right) \leq \operatorname{tr}\left(\boldsymbol{W}_t\, \mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right). \tag{C.1}$$

The right hand side can be rewritten as

$$\mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t)) = \exp(r_0\delta_t)\, \mathbf{exp}(\delta_t(\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})).$$

Let r_0 be a lower bound of the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$. By assumption, the range of the eigenvalues of $\mathbf{sym}(\boldsymbol{X}_t)$ is at most r, i.e.

$$r_0\boldsymbol{I} \preceq \mathbf{sym}(\boldsymbol{X}_t) \preceq (r_0 + r)\boldsymbol{I}.$$

Thus $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$, for $\boldsymbol{A} = (\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})/r$. Applying Lemma 2.1 with this choice of $\boldsymbol{A}$ and $\rho_1 = r\delta_t,\ \rho_2 = 0$, we obtain

$$\mathbf{exp}(\delta_t(\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I})) \preceq \boldsymbol{I} - \frac{\mathbf{sym}(\boldsymbol{X}_t) - r_0\boldsymbol{I}}{r}(1 - \exp(r\delta_t)).$$

Since $\boldsymbol{W}_t$ is symmetric positive definite and both sides of the above inequality are symmetric, we can apply Lemma 2.2 by pre-multiplying the inequality by $\boldsymbol{W}_t$ and taking a trace of both sides:

$$\operatorname{tr}\left(\boldsymbol{W}_t\, \mathbf{exp}(\delta_t\, \mathbf{sym}(\boldsymbol{X}_t))\right) \leq \exp(r_0\delta_t)\left(1 - \frac{\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) - r_0}{r}(1 - \exp(r\delta_t))\right).$$

Note that we used the assumption that $\operatorname{tr}(\boldsymbol{W}_t) = 1$. The above gives an upper bound on the right hand side of inequality (C.1) We now plug this upper bound into the first term of f and obtain $f \leq g$, where

$$\begin{aligned} g = \quad & r_0\delta_t + \log(1 - \tfrac{\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t) - r_0}{r}(1 - \exp(r\delta_t))) - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t)\delta_t \\ & + a(y_t - \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{X}_t))^2 - b(y_t - \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t))^2. \end{aligned} \tag{C.2}$$

Let us define $z = \operatorname{tr}(\boldsymbol{U}\boldsymbol{X}_t)$ and maximize the upper bound (C.2) with respect to z. Solving $\frac{\partial g}{\partial z} = 0$, we have $z = y_t - \delta_t/(2b) = y_t + \eta(\operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t) - y_t)/b$. Substituting this into (C.2), we have the upper bound $g \leq h$ where

$$\begin{aligned} h = \quad & 2\eta r_0(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t)) + \log\left(1 - \tfrac{\operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t) - r_0}{r}(1 - \mathbf{exp}(2\eta r(y - \operatorname{tr}(X_t\boldsymbol{W}_t))))\right) \\ & -2\eta y_t(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t)) + (a + \tfrac{\eta^2}{b}(y - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t))^2. \end{aligned}$$

We now upper bound the second term using the inequality $\log(1 - p(1 - \exp q)) \leq pq + q^2/8$, for $0 \leq q \leq 1$ and $q \in \mathbb{R}$ (Helmbold et al., 1997):

$$h \leq \frac{(y_t - \operatorname{tr}(\boldsymbol{X}_t\boldsymbol{W}_t))^2}{2b}((2 + r^2 b)\eta^2 - 4b\eta + 2ab).$$

It remains to show $q = (2 + r^2 b)\eta^2 - 4b\eta + 2ab \leq 0$. We easily see that q is minimized for $\eta = 2b/(2 + r^2 b)$ and that for this value of η we have $q \leq 0$ if and only if $a \leq 2b/(2 + r^2 b)$.

Appendix D. Derivation of the DefiniteBoost Dual Problem

For the sake of brevity we assume that the primal problem has one inequality constraint (note that (4.1) has multiple constraints):

$$\begin{aligned} \boldsymbol{W}^* = \operatorname*{argmin}_{\boldsymbol{W}} \quad & \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_1) + \boldsymbol{W}_1 - \boldsymbol{W} \\ \text{s.t.} \quad & \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}) \leq 0 \\ & \operatorname{tr}(\boldsymbol{W}) = 1 \\ & \boldsymbol{W} = \boldsymbol{W}^\top. \end{aligned}$$

Following Appendix B we arrive at the Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}, \alpha, \beta, \boldsymbol{\Gamma}) \quad := \quad & \operatorname{tr}(\boldsymbol{W}(\mathbf{log}\,\boldsymbol{W} - \mathbf{log}\,\boldsymbol{W}_1) + \boldsymbol{W}_1 - \boldsymbol{W} + \alpha \operatorname{tr}(\boldsymbol{W}\boldsymbol{C}) + \\ & + \beta\,(\operatorname{tr}(\boldsymbol{W}) - 1) + \operatorname{tr}((\boldsymbol{\Gamma}^\top - \boldsymbol{\Gamma})\boldsymbol{W}), \end{aligned} \tag{D.1}$$

which is minimized w.r.t. $\boldsymbol{W}$ and maximized w.r.t. $\alpha \geq 0$, $\beta \in \mathbb{R}$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{d \times d}$. Setting the gradient w.r.t. $\boldsymbol{W}$ to zero we obtain

$$\begin{aligned} \boldsymbol{W}^* \quad &= \quad \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\boldsymbol{C} - \beta\boldsymbol{I} - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top)) \\ &= \quad \exp(-\beta)\,\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\boldsymbol{C} - (\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^\top)). \end{aligned}$$

We now enforce the symmetry constraint, giving us $\boldsymbol{\Gamma} = -\alpha(\boldsymbol{C} - \boldsymbol{C}^\top)/2$, and plug this choice into the above

$$\boldsymbol{W}^* = \exp(-\beta)\,\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})).$$

Similarly, $\beta = \log\operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\ \mathbf{sym}(\boldsymbol{C}))\right)$ enforces the trace constraint. Now

$$\boldsymbol{W}^* = \mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})/Z(\alpha),$$

where $Z(\alpha) = -\log\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \alpha\,\mathbf{sym}(\boldsymbol{C})))$. Plugging $\boldsymbol{W}^*$ into in the Lagrangian, we obtain the dual optimization problem for one constraint:

$$\alpha^* = \operatorname*{argmax}_{\alpha \geq 0}\ -\log \mathrm{Z}_t(\alpha).$$

One can easily verify that the solution of the problem with n constraints is of the form:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha} \geq \mathbf{0}}\ -\log\operatorname{tr}(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{j=1}^{n} \alpha_j\,\mathbf{sym}(\boldsymbol{C}_j))).$$

Appendix E. Proof of Theorem 4.1

Recall the definition of the normalization factor $\mathrm{Z}_t(\alpha) = \operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_t - \alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right)$ of Definite-Boost. By the Golden-Thompson inequality,

$$\mathrm{Z}_t(\alpha) \leq \operatorname{tr}(\boldsymbol{W}_t\,\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))). \tag{E.1}$$

Similarly to the proof of Lemma 3.1, we now upper bound the right hand side of this inequality by applying lemmas 2.1 and 2.2. We choose $\boldsymbol{A}$ as $(\lambda_t^{\min}\boldsymbol{I} + \mathbf{sym}(\boldsymbol{C}_{j_t}))/(\lambda_t^{\max} + \lambda_t^{\min})$. Then $\mathbf{sym}(\boldsymbol{C}_{j_t})$ can be expressed as $\lambda_t^{\max}\boldsymbol{A} - \lambda_t^{\min}(\boldsymbol{I} - \boldsymbol{A})$ and $\mathbf{0} \preceq \boldsymbol{A} \preceq \boldsymbol{I}$. Thus by Lemma 2.1,

$$\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t})) \preceq \exp(-\alpha\lambda_t^{\max})\boldsymbol{A} + \exp(\alpha\lambda_t^{\min})(\boldsymbol{I} - \boldsymbol{A}).$$

Since $\boldsymbol{W}_t$ is positive definite and both sides of the above inequality are symmetric, we can apply Lemma 2.2 by multiplying this inequality by $\boldsymbol{W}_t$ and taking a trace of both sides:

$$\operatorname{tr}(\boldsymbol{W}_t\,\mathbf{exp}(-\alpha\,\mathbf{sym}(\boldsymbol{C}_{j_t}))) \le \exp(-\alpha\lambda_t^{\max})\operatorname{tr}(\boldsymbol{W}_t\boldsymbol{A}) + \exp(\alpha\lambda_t^{\min})\operatorname{tr}(\boldsymbol{W}_t(\boldsymbol{I} - \boldsymbol{A}))\,.$$

By expanding $\boldsymbol{A}$ and using the shorthand $r_t = \operatorname{tr}(\boldsymbol{W}_t\boldsymbol{C}_{j_t})$, we obtain

$$\mathrm{Z}_t(\alpha) \le \exp(-\alpha\lambda_t^{\max})\frac{\lambda_t^{\min} + r_t}{\lambda_t^{\max} + \lambda_t^{\min}} + \exp(\alpha\lambda^{\min})\frac{\lambda_t^{\max} - r_t}{\lambda_t^{\max} + \lambda_t^{\min}}.$$

We now choose the α that minimizes the right hand side of the above inequality (which is the $\widehat{\alpha}_t$ given in equation (4.6)). With this choice, the inequality becomes

$$\mathrm{Z}_t(\widehat{\alpha}_t) \le (1 - \frac{r_t}{\lambda_t^{\max}})^{\frac{\lambda_t^{\max}}{\lambda_t^{\max}+\lambda_t^{\min}}}(1 + \frac{r_t}{\lambda_t^{\min}})^{\frac{\lambda_t^{\min}}{\lambda_t^{\max}+\lambda_t^{\min}}}. \tag{E.2}$$

Applying the update rule (4.5) T times, we have

$$\boldsymbol{W}_{T+1} = \frac{\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^T \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t}))}{\prod_t \mathrm{Z}_t(\widehat{\alpha}_t)}.$$

Taking the trace of both sides and rearranging terms, we get

$$\operatorname{tr}\left(\mathbf{exp}(\mathbf{log}\,\boldsymbol{W}_1 - \sum_{t=1}^T \widehat{\alpha}_t\,\mathbf{sym}(\boldsymbol{C}_{j_t}))\right) = \prod_{t=1}^T \mathrm{Z}_t(\widehat{\alpha}_t).$$

By using the bound (E.2) for each $\mathrm{Z}_t(\alpha_t)$, the inequality of the theorem readily follows.

References

O. E. Barndorff-Nielsen, R. D. Gill, and P. E. Jupp. On quantum statistical inference. *J. R. Statist. Soc. B*, 65(4):775–816, 2003.

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.

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, 1997.

S. Golden. Lower bounds for the Helmholtz function. *Phys. Rev.*, 137:B1127–B1128, 1965.

D. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. A comparison of new and old algorithms for amixture estimation problem. *Machine Learning*, 27(1):97–119, 1997.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

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

J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 125–133. ACM Press, New York, NY, 1999.

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

N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, Technical Report UCSC-CRL-89-11, University of California, Santa Cruz, 1989.

N. Littlestone, P. M. Long, and M. K. Warmuth. On-line learning of linear functions. Technical Report UCSC-CRL-91-29, University of California, Santa Cruz, May 1992.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Potsdam, Germany, October 2001.

G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 319–333. Springer, Sydney, Australia, 2002.

G. Rätsch and M. K. Warmuth. Efficient margin maximization with boosting. submitted to Journal of Machine Learning Research, 2005.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

S. Shai-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004)*. ACM Press, New York, NY, 2004.

Y. Singer and M. K. Warmuth. Batch and on-line parameter estimation of Gaussian mixtures based on the joint entropy. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11 (NIPS'98)*, pages 578–584. MIT Press, 1999.

I. W. Tsang and J. T. Kwok. Distance metric learning with kernels. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'03)*, pages 126–129. Springer Verlag, New York, NY, 2003.

K. Tsuda, S. Akaho, and K. Asai. The em algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, May 2003.

K. Tsuda and W. S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(Suppl. 1):i326–i333, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA, 2003.