Matrix Exponential Updates for On-line Learning and Bregman Projection

Koji Tsuda[*†] **Gunnar Rätsch**[*‡] **and Manfred K. Warmuth**[§]
[*]Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
[†]AIST CBRC, 2-43 Aomi, Koto-ku, Tokyo, 135-0064, Japan
[‡]Fraunhofer FIRST, Kekuléstr. 7, 12489 Berlin, Germany
[§]University of Calif. at Santa Cruz
{koji.tsuda,gunnar.raetsch}@tuebingen.mpg.de, manfred@cse.ucsc.edu

Abstract

We address the problem of learning a positive definite matrix from examples. The central issue is to design parameter updates that preserve positive definiteness. We introduce an update based on matrix exponentials which can be used as an on-line algorithm or for the purpose of finding a positive definite matrix that satisfies linear constraints. We derive this update using the von Neumann divergence and then use this divergence as a measure of progress for proving relative loss bounds. In experiments, we apply our algorithms to learn 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 *positive definite matrix* that serves as a kernel (e.g. [12, 9, 11]). Learning is typically formulated as a parameter updating procedure to optimize a *loss function*. The gradient descent update [5] 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 is not necessarily positive definite. Xing et al. [12] solved this problem by always correcting the updated matrix to be positive. However no bound has been proven for this update-and-correction approach. In this paper, we introduce the *matrix exponential 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. 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 positive definite matrix.

Bregman divergences play a central role in the motivation and the analysis of *on-line learning algorithms* [4]. 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 [5]. For example, the gradient descent is derived from the squared Euclidean distance, and the exponentiated gradient from the Kullback-Leibler divergence. We use the von Neumann divergence (from quantum physics) for measuring the discrepancy between two positive definite matrices [6]. We derive a new *matrix exponential update* from this divergence (which is a Bregman divergence for positive definite matrices). Finally we prove *relative loss bounds* using the von Neumann divergence as a measure of progress.

Also the following related problem has received a lot of attention recently [12, 9, 11]: Find a positive definite matrix that satisfies a number of linear inequality constraints. We greedily choose the most violated constraint and perform an approximated Bregman projection. In the degenerate case when the parameter matrix is diagonal, we recover AdaBoost [7]. We prove the convergence of our algorithm which is a generalization of the standard analyses used for Boosting.

Notation: For a symmetric matrix A, $\exp A$ and $\log A$ denote the matrix exponential and logarithm operations, respectively. For two symmetric matrices A and B, $A \leq B$ iff $B - A$ is positive semi-definite. Similarly, $A < B$ iff $B - A$ is strictly positive definite.

2 Von Neumann Divergence

If $\mathcal{F}$ is a real convex differentiable function on the parameter domain (symmetric $d \times d$ positive definite matrices), then the Bregman divergence between two parameters $\tilde{W}$ and W is defined as

$$\Delta_{\mathcal{F}}(\tilde{W}, W) = \mathcal{F}(\tilde{W}) - \mathcal{F}(W) - \mathrm{tr}[(\tilde{W} - W)\nabla\mathcal{F}(W)].$$

When choosing $\mathcal{F}(W) = \mathrm{tr}(W \log W - W)$, then $\nabla\mathcal{F}(W) = \log W$ and the corresponding Bregman divergence becomes the von Neumann divergence [6]:

$$\Delta_{\mathcal{F}}(\tilde{W}, W) = \mathrm{tr}(\tilde{W} \log \tilde{W} - \tilde{W} \log W - \tilde{W} + W). \tag{1}$$

If $W = \sum_i \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$ is our notation for the eigenvalue decomposition, then we can rewrite the divergence as

$$\Delta_{\mathcal{F}}(\tilde{W}, W) = \sum_i \tilde{\lambda}_i \ln \tilde{\lambda}_i + \lambda_i - \tilde{\lambda}_i + \sum_{i,j} \tilde{\lambda}_i \ln \lambda_j (\tilde{\boldsymbol{v}}_i^\top \boldsymbol{v}_j)^2.$$

So this divergence quantifies the difference in the eigenvalues as well as the eigenvectors. In this paper, we are primarily interested in the normalized case ($\mathrm{tr}(W) = 1$), because kernel learning algorithms are insensitive to the scale of the kernel matrix. In this case the divergence simplifies to $\Delta_{\mathcal{F}}(\tilde{W}, W) = \mathrm{tr}(\tilde{W} \log \tilde{W} - \tilde{W} \log W)$.

3 On-line Learning

In this section, we present a natural extension of the *exponentiated gradient* (EG) algorithm [5] to an update for symmetric positive definite matrices. At the t-th trial, the algorithm receives a symmetric instance matrix $X_t \in \mathcal{R}^{d \times d}$. It then produces a prediction $\hat{y}_t = \mathrm{tr}(W_t X_t)$ based on the algorithm's current symmetric positive definite parameter matrix W_t. Finally it incurs a loss $(\hat{y}_t - y_t)^2$ and updates its parameter matrix W_t. In the update we aim to solve the following problem [1]:

$$W_{t+1} = \mathrm{argmin}_W \, \Delta_{\mathcal{F}}(W, W_t) + \eta(\mathrm{tr}(W X_t) - y_t)^2, \tag{2}$$

[1] For the sake of simplicity, we use the simple quadratic loss: $L_t(W) = (\mathrm{tr}(X_t W) - y_t)^2$. In general, the gradient $\nabla L_t(W_t)$ is exponentiated in the update (4). More general loss functions (based on Bregman divergences) are amenable to our techniques (see e.g. [4]).

where the convex function $\mathcal{F}$ defines the Bregman divergence. Setting the derivative with respect to W to zero, we have

$$\nabla\mathcal{F}(W_{t+1}) - \nabla\mathcal{F}(W_t) + \eta\nabla[\mathrm{tr}(W_{t+1}X_t) - y_t)^2] = 0. \tag{3}$$

The update rule is derived by solving (3) with respect to W_{t+1}, but it is not solvable in closed form. A common way to avoid this problem is to approximate $\mathrm{tr}(W_{t+1}X_t)$ by $\mathrm{tr}(W_tX_t)$ [4]. Then, we have the following update:

$$W_{t+1} = (\nabla\mathcal{F})^{-1}(\nabla\mathcal{F}(W_t) - 2\eta(\hat{y}_t - y_t)X_t).$$

In our case, $\mathcal{F}(W) = \mathrm{tr}(W\log W - W)$ and thus $\nabla\mathcal{F}(W) = \log W$ and $(\nabla\mathcal{F})^{-1}(W) = \exp W$. We also augment (2) with the constraint $\mathrm{tr}(W) = 1$, leading to the following *matrix exponential update*:

$$W_{t+1} = \frac{1}{\mathcal{Z}_t}\exp(\log W_t - 2\eta(\hat{y}_t - y_t)X_t), \tag{4}$$

where the normalization factor is $\mathcal{Z}_t = \mathrm{tr}[\exp(\log W_t - 2\eta(\hat{y}_t - y_t)X_t)]$. Note that in the above update, the exponent $\log W_t - 2\eta(\hat{y}_t - y_t)X_t$ is an arbitrary symmetric matrix and the matrix exponential converts this matrix back into a symmetric positive definite matrix.

Relative Loss Bound Let $S = \{(X_1, y_1), \ldots, (X_T, y_T)\}$ denote a sequence of examples, where the instance matrices $X_t \in \Re^{d\times d}$ are symmetric and the labels $y_t \in \Re$. For any symmetric positive semi-definite matrix U with $\mathrm{tr}(U) = 1$, define its total loss as $Loss_U(S) = \sum_{t=1}^{T}(\mathrm{tr}(UX_t) - y_t)^2$. The total loss of the on-line algorithm is $L_{MEG}(S) = \sum_{t=1}^{T}(\mathrm{tr}(W_tX_t) - y_t)^2$. We prove a bound on the *relative loss* $L_{MEG}(S) - Loss_U(S)$ that holds for any U. The proof generalizes a similar bound for the exponentiated gradient algorithm (Lemmas 5.8 and 5.9 of [5]). The relative loss bound is derived in two steps: Lemma 3.1 bounds the relative loss for an individual trial and Lemma 3.2 for a whole sequence (Proofs in Appendix).

Lemma 3.1 *Let X_t be any symmetric matrix whose smallest and largest eigenvalues satisfy $\lambda^{min} > r_0$ and $\lambda^{max} \leq r_0 + r$, respectively. Let U be any symmetric positive semi-definite matrix. Then for any constants a and b such that $0 < a \leq 2b/(2 + r^2b)$ and any learning rate $\eta = 2b/(2 + r^2b)$, we have*

$$a(y_t - \mathrm{tr}(W_tX_t))^2 - b(y_t - \mathrm{tr}(UX_t))^2 \leq \Delta(U, W_t) - \Delta(U, W_{t+1}) \tag{5}$$

Lemma 3.2 *Let W_1 and U be arbitrary symmetric positive definite start and comparison matrices. Then for any c such that $\eta = 2c/(r^2(2 + c))$,*

$$L_{MEG}(S) \leq (1 + \frac{c}{2})Loss_U(S) + (\frac{1}{2} + \frac{1}{c})r^2\Delta(U, W_1). \tag{6}$$

Assuming $Loss_U(S) \leq \ell_{max}$ and $\Delta(U, W_1) \leq d_{max}$, the bound (6) is tightest when $c = r\sqrt{2d_{max}/\ell_{max}}$. Then we have $L_{MEG}(S) - Loss_U(S) \leq \frac{r^2}{2}\Delta(U, W_1) + r\sqrt{2\ell_{max}d_{max}}$.

4 Bregman Projection

In this section, we address the following Bregman projection problem[2]

$$W^* = \mathrm{argmin}_W\, \Delta_{\mathcal{F}}(W, W_1),\ \mathrm{tr}(W) = 1, \mathrm{tr}(WC_j) \leq 0, j = 1, \ldots, n, \tag{7}$$

[2]Note that if η is large then the on-line update (2) becomes a Bregman projection subject to a single equality constraint $\mathrm{tr}(WX_t) = y_t$.

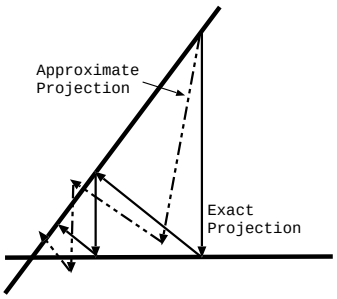

Figure 1: In (exact) Bregman projections, the intersection of convex sets (i.e., two lines here) is found by iterating projections to each set. We project only approximately, so the projected point does not satisfy the current constraint. Nevertheless, global convergence to the optimal solution is guaranteed via our proofs.

where $W_1 > 0, \mathrm{tr}(W_1) = 1$ is the initial parameter matrix, and $C_1, \ldots, C_n$ are arbitrary symmetric matrices. Prior knowledge about W is encoded in the constraints, and the matrix closest to W_1 is chosen among the matrices satisfying all constraints. Tsuda and Noble [11] employed this approach for learning a kernel matrix among graph nodes, and it can be potentially applied to learn a kernel matrix in other settings (e.g. [12, 9]).

The problem (7) is perceived as a projection of 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 [1]. An interesting point of our algorithm is that the parameter is only *approximately* projected in each iteration (Figure 1), but nevertheless one can prove the convergence as follows: Interpret Boosting as a Bregman projection [3] and use a generalization of the convergence proof for Boosting [7].

Before presenting the algorithm, let us derive the dual problem of (7) by means of Lagrange multipliers $\boldsymbol{\gamma}$,

$$\boldsymbol{\gamma}^* = \mathrm{argmin}_{\boldsymbol{\gamma}} \, \mathrm{tr}[\exp(\log W_1 - \sum_{j=1}^{n} \gamma_j C_j)], \quad \gamma_j \geq 0. \tag{8}$$

See [11] for detailed derivation of the dual problem. When (7) is feasible, the optimal solution is described as $W^* = \frac{1}{\mathcal{Z}(\boldsymbol{\gamma}^*)} \exp(\log W_1 - \sum_{j=1}^{n} \gamma_j^* C_j)$, where $\mathcal{Z}(\boldsymbol{\gamma}^*) = \mathrm{tr}[\exp(\log W_1 - \sum_{j=1}^{n} \gamma_j^* C_j)]$.

Exact Bregman Projections First, let us present the exact Bregman projection algorithm to solve (7). We start from the initial parameter W_1. At the t-th step, the most unsatisfied constraint is chosen, $j_t = \mathrm{argmax}_{j=1,\cdots,n} \mathrm{tr}(W_t C_j)$. Let us use C_t as the short notation for C_{j_t}. Then, the following Bregman projection with respect to the chosen constraint is solved.

$$W_{t+1} = \mathrm{argmin}_W \, D(W, W_t), \;\; \mathrm{tr}(W) = 1, \mathrm{tr}(W C_t) \leq 0. \tag{9}$$

By means of a Lagrange multiplier α, the dual problem is described as

$$\alpha_t = \mathrm{argmin}_{\alpha} \, \mathrm{tr}[\exp(\log W_t - \alpha C_t)], \quad \alpha \geq 0. \tag{10}$$

Using the solution of the dual problem, W_t is updated as

$$W_{t+1} = \frac{1}{\mathcal{Z}_t(\alpha_t)} \exp(\log W_t - \alpha_t C_t) \tag{11}$$

where the normalization factor is $\mathcal{Z}_t(\alpha_t) = \mathrm{tr}[\exp(\log W_t - \alpha_t C_t)]$.

Approximate Bregman Projections The solution of (10) cannot be obtained in closed form. However, one can use the following approximate solution:

$$\alpha_t = \frac{1}{\lambda_t^{max} - \lambda_t^{min}} \log \left(\frac{1 + r_t/\lambda_t^{max}}{1 + r_t/\lambda_t^{min}} \right), \tag{12}$$

when the eigen values of C_t lie in the interval $[\lambda_t^{min}, \lambda_t^{max}]$ and $r_t = \mathrm{tr}(W_t C_t)$. Since the most unsatisfied constraint is chosen, $r_t \geq 0$ and thus $\alpha_t \geq 0$. Although the projection is done only approximately, the convergence of the dual objective (8) can be shown using the following upper bound.

Theorem 4.1 *The dual objective (8) is bounded as*

$$\mathrm{tr}[\exp(\log W_1 - \sum_{j=1}^{n} \gamma_j C_j)]) \leq \prod_{t=1}^{T} \rho(r_t) \tag{13}$$

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

The dual objective is monotonically decreasing, because $\rho(r_t) \leq 1$. Also, since r_t corresponds to the maximum value among all constraint violations $\{r_j\}_{j=1}^n$, and $\rho(r_t) = 1$ only if $r_t = 0$. Thus the dual objective continues to decrease until all constraints are satisfied.

Relation to Boosting When all matrices are diagonal, our algorithm degenerates to AdaBoost [7]: Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^d$ be the training samples, where $\boldsymbol{x}_i \in \Re^m$ 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 $C_j = \mathrm{diag}(y_1 h_j(x_1), \ldots, y_d h_j(x_d))$. Since $|y h_j(x)| \leq 1$, $\lambda_t^{max/min} = \pm 1$ for any t. Setting $W_1 = I/d$, the dual objective (13) is rewritten as

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

which is equivalent to the exponential loss function used in AdaBoost. Since C_j and W_1 are diagonal, the matrix W_t stays diagonal after the update. If $w_{ti} = [W_t]_{ii}$, the updating formula (11) becomes the AdaBoost update: $w_{t+1,i} = w_{ti} \exp(-\alpha_t y_i h_t(x_i))/\mathcal{Z}_t(\alpha_t)$. The approximate solution of α_t (12) 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. $r_t = \sum_{i=1}^{d} w_{ti} y_i h_t(x_i)$.

5 Experiments on Learning Kernels

In this section, our algorithms are applied to learning a kernel matrix from a set of distance measurements. When 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 [8]. 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) [12, 11]. Our task is to obtain a positive definite kernel matrix which fits well to the given distance data.

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 $\{W_t\}_{t=1}^T$, and the corresponding points in the feature space as $\{\boldsymbol{x}_{ti}\}_{i=1}^d$ (i.e. $[W_t]_{ab} = \boldsymbol{x}_{ta}^\top \boldsymbol{x}_{tb}$). Then, the total loss incurred by this sequence is

$$\sum_{t=1}^{T} (\|\boldsymbol{x}_{ta_t} - \boldsymbol{x}_{tb_t}\|^2 - y_t)^2 = \sum_{t=1}^{T} (\mathrm{tr}(W_t X_t) - y_t)^2,$$

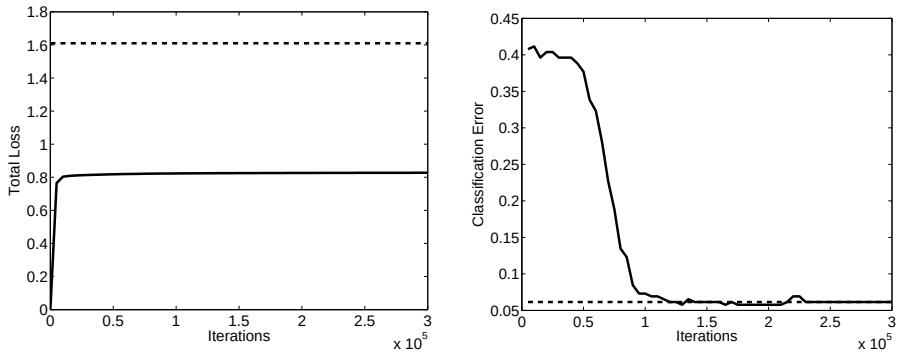

Figure 2: Numerical results of on-line learning. (Left) total loss against the number of iterations. The broken flat line shows the loss bound. (Right) classification error of the nearest neighbor classifier using the learned kernel. The broken line shows the error by the target kernel.

where 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 [10] for details). Each distance example is created by randomly choosing one element of the target kernel. The initial parameter was set as $W_1 = I/d$. When the comparison matrix U is set to the target matrix, $Loss_U(S) = 0$ and $\ell_{max} = 0$, because all the distance examples are derived from the target matrix. 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.

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 X_j is defined as in the previous section and $C_j = X_j - \gamma I$, the inequalities are then rewritten as $\mathrm{tr}(WC_j) \leq 0, j = 1, \ldots, n$. The largest and smallest eigenvalues of any C_j are $1 - \gamma$ and $-\gamma$, respectively. As in the previous section, distance examples are 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 neighbor classifier. As opposed to the previous experiment, the error rate is higher than that of the target kernel

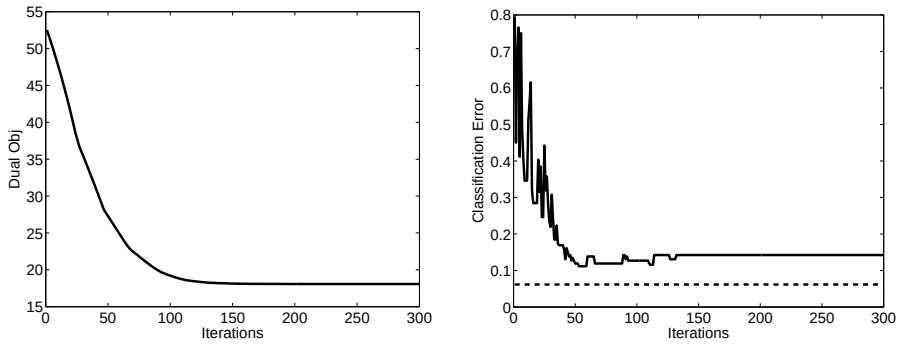

Figure 3: Numerical results of Bregman projection. (Left) convergence of the dual objective function. (Right) classification error of the nearest neighbor classifier using the learned kernel.

matrix, because substantial amount of information is lost by the conversion to inequality constraints.

6 Conclusion

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 a symmetric positive definite matrix (of trace one) instead of a probability vector. As in quantum physics, the eigenvalues act as probabilities.

Acknowledgment We would like to thank B. Schölkopf, M. Kawanabe, J. Liao and W.S. Noble for fruitful discussions. M.W. was supported by NSF grant CCR 9821087 and UC Discovery prant LSIT02-10110.

A Proof of Lemma 3.1

Let $\beta_t = 2\eta(y_t - \mathrm{tr}(XW_t))$, then the right hand side of (5) is rewritten as $\Delta(U, W_t) - \Delta(U, W_{t+1}) = \beta_t \mathrm{tr}(UX_t) - \log \mathrm{tr}(\exp(\log W_t + \beta_t X_t))$. Therefore, (5) is equivalent to $f \leq 0$, where $f = \log \mathrm{tr}(\exp(\log W_t + \beta_t X_t)) - \beta_t \mathrm{tr}(UX_t) + a(y_t - \mathrm{tr}(W_t X_t))^2 - b(y_t - \mathrm{tr}(UX_t))^2$. Let us bound the first term. Due to Golden-Thompson inequality [2], we have

$$\mathrm{tr}[\exp(\log W_t + \beta_t X_t)] \leq \mathrm{tr}[W_t \exp(\beta_t X_t)]. \tag{14}$$

The right hand side is rewritten as $\exp(\beta_t X_t) = \exp(r_0 \beta_t) \exp(\beta_t (X_t - r_0 I))$. Then we use the following lemma.

Lemma A.1 *If an $n \times n$ symmetric matrix A satisfies $0 < A \leq I$, $\exp(\delta_1 A + \delta_2 (I - A)) \leq \exp(\delta_1) A + \exp(\delta_2)(I - A)$ for finite $\delta_1, \delta_2 \in \Re$.*

(proof) A is eigen decomposed as $A = V \Lambda V^\top$, where Λ is a diagonal matrix of eigenvalues $0 \leq \lambda_k \leq 1$, and V is the matrix of eigenvectors. The k-th eigenvalue of the left hand side is bounded as $\theta_k = \exp(\delta_1 \lambda_k + \delta_2 (1 - \lambda_k)) \leq \exp(\delta_1) \lambda_k + \exp(\delta_2)(1 - \lambda_k)$ due to Jensen's inequality. Let Θ be the diagonal matrix of θ_k, then $\Theta \leq \exp(\delta_1)\Lambda + \exp(\delta_2)(I - \Lambda)$. By multiplying both sides by V from left and by $V^\top$ from right, we prove the lemma.

Using the lemma with $A = (X_t - r_0 I)/r$, $\delta_1 = r\beta_t$, $\delta_2 = 0$, we have $\exp(\beta_t (X_t - r_0 I)) \leq I - \frac{X_t - r_0 I}{r}(1 - \exp(r\beta_t))$. Here $0 < A \leq I$, because $r_0 I < X_t \leq (r_0 + r) I$ by assumption.

Since W_t is strictly positive definite, $\mathrm{tr}(W_t B) \leq \mathrm{tr}(W_t C)$ if $B \leq C$. So, the right hand side of (14) can be written as

$$\mathrm{tr}[W_t \exp(\beta_t X_t)] \leq \exp(r_0 \beta_t) \left[1 - \frac{\mathrm{tr}(W_t X_t) - r_0}{r}(1 - \exp(r\beta_t))\right],$$

where we used the assumption $\mathrm{tr}(W_t) = 1$. We now plug this upper bound of the first term of f back into f and obtain $f \leq g$, where

$$\begin{aligned} g = \quad & r_0\beta_t + \log[1 - \tfrac{\mathrm{tr}(W_t X_t) - r_0}{r}(1 - \exp(r\beta_t))] - \mathrm{tr}(UX_t)\beta_t \\ & + a(y_t - \mathrm{tr}(W_t X_t))^2 - b(y_t - \mathrm{tr}(UX_t))^2. \end{aligned} \tag{15}$$

Let us define $z = \mathrm{tr}(UX_t)$ and maximize the upper bound (15) with respect to z. Solving $\frac{\partial g}{\partial z} = 0$, we have $z = y_t - \beta_t/(2b) = y_t + \eta(\mathrm{tr}(X_t W_t) - y_t)/b$. Substituting this into (15), we have the upper bound $g \leq h$ where

$$\begin{aligned} h = \quad & 2\eta r_0 (y_t - \mathrm{tr}(X_t W_t)) + \log[1 - \tfrac{\mathrm{tr}(X_t W_t) - r_0}{r}(1 - \exp(2\eta r(y - \mathrm{tr}(X_t W_t))))] \\ & -2\eta y_t (y_t - \mathrm{tr}(X_t W_t)) + (a + \tfrac{\eta^2}{b}(y - \mathrm{tr}(X_t W_t))^2. \end{aligned}$$

Using the upper bound $\log(1 - q(1 - e^p)) \leq pq + p^2/8$ in the second term, we have

$$h \leq \frac{(y_t - \mathrm{tr}(X_t W_t))^2}{2b}((2 + r^2 b)\eta^2 - 4b\eta + 2ab).$$

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 $a \leq 2b/(2 + r^2 b)$.

References

[1] L.M. Bregman. Finding the common point of convex sets by the method of successive projections. *Dokl. Akad. Nauk SSSR*, 165:487–490, 1965.

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

[3] J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conference on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

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

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

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

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

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

[9] 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, 2003.

[10] 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.

[11] K. Tsuda and W.S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 2004. to appear.

[12] 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.