Efficient Multiplicative updates for Support Vector Machines

Vamsi K. Potluru* Sergey M. Plis* Morten Mørup† Vincent D. Calhoun*
Terran Lane*

January 26, 2009

Abstract

The dual formulation of the support vector machine (SVM) objective function is an instance of a nonnegative quadratic programming problem. We reformulate the SVM objective function as a matrix factorization problem which establishes a connection with the regularized nonnegative matrix factorization (NMF) problem. This allows us to derive a novel multiplicative algorithm for solving hard and soft margin SVM. The algorithm follows as a natural extension of the updates for NMF and semi-NMF. No additional parameter setting, such as choosing learning rate, is required. Exploiting the connection between SVM and NMF formulation, we show how NMF algorithms can be applied to the SVM problem. Multiplicative updates that we derive for SVM problem also represent novel updates for semi-NMF. Further this unified view yields algorithmic insights in both directions: we demonstrate that the Kernel Adatron algorithm for solving SVMs can be adapted to NMF problems. Experiments demonstrate rapid convergence to good classifiers. We analyze the rates of asymptotic convergence of the updates and establish tight bounds. We test them on several datasets using various kernels and report equivalent classification performance to that of a standard SVM.

1 Introduction

Support vector machines (SVM) are now routinely used for many classification problems in machine learning [15] due to their ease of use and ability to generalize. In the basic case, the input data, corresponding to two groups, is mapped into a higher dimensional space, where a maximum-margin hyperplane is computed to separate them. The "kernel trick" is used to ensure that the mapping into higher dimensional space is never explicitly calculated. This can be formulated as a non-negative quadratic programming (NQP) problem and there are efficient algorithms to solve it [13].

An SVM can be trained using variants of the gradient descent method applied to the NQP. Although these methods can be quite efficient [5], their drawback is that they require a manually-tuned and problem specific learning rate. Subset selection methods are an alternative approach to solving the SVM NQP problem [13]. At a high level, they work by splitting the arguments of the quadratic function at each iteration into two sets: a fixed set, where the arguments are held constant, and a working set of the variables being optimized in the current iteration. These methods, though efficient in space and time, still require a heuristic to exchange arguments between the working and the fixed sets.

An alternative algorithm for solving the general NQP problem has been applied to SVMs in [17]. The algorithm, called M^3, uses multiplicative updates to iteratively converge to the solution. It does not require any heuristics, such as setting the learning rate or choosing how to split the argument set. Multiplicative updates in the M^3 algorithm are formulated for the general NQP problem and then applied to SVM as a special case. It was also demonstrated in [16] that M^3 can solve soft-margin SVMs and the sum constraint can be accounted for. However, accounting for the sum constraint requires choosing a parameter, which violates the original intention of creating a parameter free SVM algorithm.

In this paper, we reformulate the dual SVM problem as a matrix factorization problem and demonstrate a connection to the non-negative matrix factorization (NMF) algorithm [8]. NMF employs multiplicative updates and is very successful in practice due to its independence from the learning rate parameter, low computational complexity, and ease of implementation.

The new formulation allows us to devise multiplicative updates for solving SVM, resulting in a novel multiplicative algorithm. The new updates inherit all of the good properties of the NMF algorithm, such as independence from hyper-parameters, low computational complexity, and ease of implementation. Furthermore, in the case when a positive kernel is used, the new algorithm converges faster than the previous multiplicative solution of the SVM problem from [17] both asymptotically (a proof is provided) and in practice (as we demonstrate empirically). Derived for the SVM problem, the

*University of New Mexico
†Technical University of Denmark

new multiplicative updates are equally applicable to the semiNMF problem [2]. As novel updates for semi-NMF they provide an algorithm with a simpler convergence proof than in [2].

Although our updates are derived for the case when the separating hyperplane passes through the origin, we demonstrate how to include the bias in the form of the sum constraint. In contrast to the M^3 algorithm, the bias is optimized via parameter free multiplicative updates. This makes the novel algorithm the first parameter free multiplicative algorithm for solving the general hard margin SVM problem. Finally we show how to solve the soft margin SVM problem using the new algorithm.

Establishing the connection between the dual SVM and the primal NMF problems opens up possibilities for application of algorithms developed for one of the problems to the other. As an example, we demonstrate how projected gradient algorithms developed for NMF can be applied to SVM. Further we show a possibility of adopting SVM algorithms to NMF problems by demonstrating how the Kernel Adatron algorithm [5] is applied to NMF.

To support theoretical findings about the improved asymptotic bounds of our novel multiplicative SVM algorithm, we demonstrate its convergence empirically on standard datasets.

2 NMF

We present a brief introduction to NMF mechanics with the notation that is standard in NMF literature. NMF is a tool to split a given non-negative data matrix into a product of two non-negative matrix factors [8]. The constraint of non-negativity (all elements are ≥ 0) usually results in a parts-based representation and is different from other factorization techniques which result in more holistic representations (e.g. PCA and VQ).

Given a non-negative $m \times n$ matrix $\boldsymbol{X}$, we want to represent it with a product of two non-negative matrices $\boldsymbol{W}, \boldsymbol{H}$ of sizes $m \times r$ and $r \times n$ respectively:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}. \tag{2.1}$$

Lee and Seung [8] describe two simple multiplicative updates for $\boldsymbol{W}$ and $\boldsymbol{H}$ which work well in practice. These correspond to two different cost functions representing the quality of approximation. Here, we use the Frobenius norm for the cost function. The cost function and the corresponding multiplicative updates are:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \tag{2.2}$$

$$\boldsymbol{W} = \boldsymbol{W} \odot \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \tag{2.3}$$

$$\boldsymbol{H} = \boldsymbol{H} \odot \frac{\boldsymbol{W}^T\boldsymbol{X}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}, \tag{2.4}$$

where $\|.\|_F$ denotes the Frobenius norm and the operator $\odot$ represents element-wise multiplication. Division is also element-wise. It should be noted that the cost function to be minimized is convex in either $\boldsymbol{W}$ or $\boldsymbol{H}$ but not in both [8]. In [8] it is proved that when the algorithm iterates using the updates (2.3) and (2.4), $\boldsymbol{W}$ and $\boldsymbol{H}$ monotonically decrease the cost function.

The slightly mysterious form for the above updates can be understood as described in [8]. A simple additive update for $\boldsymbol{H}$ is given by:

$$\boldsymbol{H} = \boldsymbol{H} + \boldsymbol{\eta} \odot (\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}) \tag{2.5}$$

If the learning rate given by the matrix elements of $\boldsymbol{\eta}$ be all set to some small positive number then this is the conventional gradient descent. However, setting the learning rate matrix as follows:

$$\boldsymbol{\eta} = \frac{\boldsymbol{H}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}} \tag{2.6}$$

gives us the NMF updates. We note the multiplicative factors for the updates correspond to the negative component of the derivative divided element-wise by the positive component of the derivative respectively.

3 semi-NMF

NMF problem was extended by Ding et al. [2] to semi-NMF, where data and one of the factors were allowed to have elements of either sign. Ding et al. [2] derive and prove the convergence of multiplicative updates for this case:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \tag{3.7}$$

$$\boldsymbol{W} = \boldsymbol{X}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1} \tag{3.8}$$

$$\boldsymbol{H} = \boldsymbol{H} \odot \sqrt{\frac{[\boldsymbol{W}^T\boldsymbol{X}]^+ + [\boldsymbol{W}^T\boldsymbol{W}]^-\boldsymbol{H}}{[\boldsymbol{W}^T\boldsymbol{W}]^+\boldsymbol{H} + [\boldsymbol{W}^T\boldsymbol{X}]^-}} \tag{3.9}$$

4 SVM as matrix factorization

Let the set of labeled examples be $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, with binary class labels $y_i = \pm 1$ corresponding to two classes, denoted by A and B respectively. Let the mapping $\Phi(\boldsymbol{x}_i)$ be the representation of the input datapoint $\boldsymbol{x}_i$ in space Φ, where we denote the space by the name of the

mapping function performing the transformation. We now consider the problem of computing the maximum margin hyperplane for SVM in the case where the classes are linearly separable and the hyperplane passes through origin (We will relax this constraint presently.).

The dual quadratic optimization problem for SVM [15] is given by minimizing the following loss function:

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \quad \text{subject to } \alpha_i \geq 0, i \in \{1..n\}, \tag{4.10}$$

where $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is a kernel that computes the inner product $\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$ in the space Φ by performing all operations only in the original data space on x_i and x_j, thus defining a Hilbert space Φ.

The first sum can be split into three terms: two terms contain kernels of elements that belong to the same respective class (one term per class), and the third contains only the kernel between elements of the two classes. This rearrangement of terms allows us to drop class labels y_i, y_j from the objective function. Denoting $k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ with k_{ij} and defining $\rho_{ij} = \alpha_i\alpha_j k_{ij}$ for conciseness, we have:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\left(\sum_{ij\in A}\rho_{ij} - 2\sum_{\substack{i\in B\\ j\in A}}\rho_{ij} + \sum_{ij\in B}\rho_{ij}\right) - \sum_{i=1}^{n}\alpha_i \quad \text{subject to } \alpha_i \geq 0, i \in \{1..n\}. \tag{4.11}$$

Noticing the square and the fact that $k_{ij} = \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)$ we rewrite the problem as:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B\|_2^2 - \sum_{i\in\{A,B\}}\alpha_i \quad \text{subject to } \alpha_i \geq 0, \tag{4.12}$$

where the matrices $\boldsymbol{X}_A, \boldsymbol{X}_B$ contain the datapoints corresponding to groups A and B respectively with the stacking being column-wise. The map Φ applied to a matrix corresponds to mapping each individual column vector of the matrix using Φ and stacking them to generate the new matrix. The vectors $\boldsymbol{\alpha}_A$ and $\boldsymbol{\alpha}_B$ contain coefficients of the support vectors of the two groups A and B respectively. We will use the vector $\boldsymbol{\alpha}$ to denote the concatenation of vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$. Expression (4.12) is a form of matrix factorization problem and resembles NMF with an additional term in the objective [8]. The above formulation enables other metrics $D(\Phi(\boldsymbol{X_A})\boldsymbol{\alpha}_A||\Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B)$ than least squares for SVM such as more general Bregman divergence [1]. However, to be computationally efficient the metric used has to admit the use of the kernel trick.

5 Sign-insensitive Kernel SVMs

In this section, we derive two new updates for solving SVM's with sign-insensitive kernels based on NMF. A sign-insensitive kernel is one whose output can be either positive, negative or zero. One of them follows immediately by appealing to the semi-NMF formulation. The other is derived using the idea from NMF updates of Lee and Seung [8].

5.1 semi-NMF SVM We differentiate the objective (4.12) with respect to $\boldsymbol{\alpha}_A$:

$$\begin{aligned}\frac{\partial S}{\partial \boldsymbol{\alpha}_A} &= \Phi(\boldsymbol{X}_A)^T\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \\ &\quad -(\Phi(\boldsymbol{X}_A)^T\Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1}) \\ &= K(\boldsymbol{X}_A,\boldsymbol{X}_A)\boldsymbol{\alpha}_A \\ &\quad -(K(\boldsymbol{X}_A,\boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1})\end{aligned} \tag{5.13}$$

We slightly abuse notation to define kernel for matrices as follows: $K(\boldsymbol{C},\boldsymbol{D})$ is given by the matrix whose $(i,j)^{th}$ element is given by the inner product of i^{th} and j^{th} datapoints of matrices $\boldsymbol{C}$,$\boldsymbol{D}$ respectively in the feature space Φ for all values of (i,j) in range. We note that the derivative has a positive and a negative component. We use the following notation to represent kernel matrices:

$$K(\boldsymbol{X}_A,\boldsymbol{X}_B) = \boldsymbol{K}_{AB}$$
$$K(\boldsymbol{X}_A,\boldsymbol{X}_A) = \boldsymbol{K}_A$$

and their decomposition into $\boldsymbol{K}^+$ and $\boldsymbol{K}^-$:

$$K^+_{ij} = \begin{cases} K_{ij} & K_{ij} > 0, \\ 0 & \text{otherwise,}\end{cases} \qquad K^-_{ij} = \begin{cases} |K_{ij}| & K_{ij} < 0, \\ 0 & \text{otherwise.}\end{cases}$$

Similarly, we take the derivative with respect to $\boldsymbol{\alpha}_B$. Recalling the updates for semi-NMF from Section 3, we write down the multiplicative updates for problem (4.12):

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \sqrt{\frac{\boldsymbol{K}^+_{AB}\boldsymbol{\alpha}_B + \boldsymbol{K}^-_A\boldsymbol{\alpha}_A + \mathbf{1}}{\boldsymbol{K}^+_A\boldsymbol{\alpha}_A + \boldsymbol{K}^-_{AB}\boldsymbol{\alpha}_B}} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \sqrt{\frac{\boldsymbol{K}^+_{BA}\boldsymbol{\alpha}_A + \boldsymbol{K}^-_B\boldsymbol{\alpha}_B + \mathbf{1}}{\boldsymbol{K}^+_B\boldsymbol{\alpha}_B + \boldsymbol{K}^-_{BA}\boldsymbol{\alpha}_A}}\end{aligned} \tag{5.14}$$

where $\mathbf{1}$ is an appropriately sized vector of ones and $\odot$ denotes the Hadamard product as before. The proof of

the updates directly follows from the proof of semiNMF updates [2, 10].

5.2 MUSIK If instead of using semi-NMF formulation, we use NMF to derive the updates, i.e. updating by the ratio of the negative to the positive part of the gradient, we get the following:

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{\boldsymbol{K}^+_{AB}\boldsymbol{\alpha}_B + \boldsymbol{K}^-_A\boldsymbol{\alpha}_A + \mathbf{1}}{\boldsymbol{K}_A\boldsymbol{\alpha}_A + \boldsymbol{K}^-_{AB}\boldsymbol{\alpha}_B} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{\boldsymbol{K}^+_{BA}\boldsymbol{\alpha}_A + \boldsymbol{K}^-_B\boldsymbol{\alpha}_B + \mathbf{1}}{\boldsymbol{K}^+_B\boldsymbol{\alpha}_B + \boldsymbol{K}^-_{BA}\boldsymbol{\alpha}_A}\end{aligned} \tag{5.15}$$

In these updates, we note that the split is not done as in the previous section. Instead the kernel matrix is split as follows:

$$K^+_{ij} = \begin{cases} K_{ij} & K_{ij} > 0, \\ K_{ij} + D_{ii} & i = j, \\ 0 & \text{otherwise}, \end{cases}$$

$$K^-_{ij} = \begin{cases} |K_{ij}| & K_{ij} < 0, \\ |K_{ij}| + D_{ii} & i = j, \\ 0 & \text{otherwise}. \end{cases}$$

In other words the new split when defined in terms of the old split looks like:

$$\begin{aligned}\boldsymbol{K}^+_{\text{new}} &= \boldsymbol{K}^+ + \boldsymbol{D} \\ \boldsymbol{K}^-_{\text{new}} &= \boldsymbol{K}^- + \boldsymbol{D} \\ \boldsymbol{K} &= \boldsymbol{K}^+_{\text{new}} - \boldsymbol{K}^-_{\text{new}} \\ &= \boldsymbol{K}^+ - \boldsymbol{K}^-\end{aligned}$$

where matrix $\boldsymbol{D}$ is a non-negative diagonal matrix. We note that in practice, we explicitly work with the matrices in the old split even though we are using the new split. This is compensated for in the new updates. The construction of matrix $\boldsymbol{D}$ is as follows:

$$[\boldsymbol{D}_A]_{ii} = \max(0, \sum_{j \neq i} \left[\boldsymbol{K}^-_A\right]_{ij} - \left[\frac{\boldsymbol{K}^-_{AB}\boldsymbol{\alpha}_B}{\boldsymbol{\alpha}_A}\right]_i). \tag{5.16}$$

$$[\boldsymbol{D}_B]_{ii} = \max(0, \sum_{j \neq i} \left[\boldsymbol{K}^-_B\right]_{ij} - \left[\frac{\boldsymbol{K}^-_{BA}\boldsymbol{\alpha}_A}{\boldsymbol{\alpha}_B}\right]_i). \tag{5.17}$$

This ensures that $\boldsymbol{K}^-_{\text{new}}$ becomes positive semi-definite. At each new iteration of the updates, we choose $\boldsymbol{D}$ adaptively using eq 5.16 and the new updates are given by:

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{\boldsymbol{K}^+_{AB}\boldsymbol{\alpha}_B + \boldsymbol{K}^-_A\boldsymbol{\alpha}_A + \mathbf{1} + \boldsymbol{D}_A\boldsymbol{\alpha}_A}{\boldsymbol{K}_A\boldsymbol{\alpha}_A + \boldsymbol{K}^-_{AB}\boldsymbol{\alpha}_B + \boldsymbol{D}_A\boldsymbol{\alpha}_A} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{\boldsymbol{K}^+_{BA}\boldsymbol{\alpha}_A + \boldsymbol{K}^-_B\boldsymbol{\alpha}_B + \mathbf{1} + \boldsymbol{D}_B\boldsymbol{\alpha}_B}{\boldsymbol{K}^+_B\boldsymbol{\alpha}_B + \boldsymbol{K}^-_{BA}\boldsymbol{\alpha}_A + \boldsymbol{D}_B\boldsymbol{\alpha}_B}\end{aligned} \tag{5.18}$$

This condition is required for convergence properties of the updates. We defer the proof to the appendix. We note that in the case of non-negative kernels i.e. kernels which output a nonnegative value for all valid inputs, the split can be done trivially by having $\boldsymbol{K}^-$ set to zero and $\boldsymbol{K}^+$ set to the original kernel matrix.

We call this new algorithm **Multiplicative Updates for sign-insensitive Kernel** SVM (MUSIK). We note that besides solving SVM problem, this formulation presents multiplicative updates for semi-NMF alternative to Ding et al. [2]. Further, it positions us to extend to the general, soft-margin, biased SVM (Sections 8 and 9).

6 Non-negative Quadratic Programming

It is well known that the dual formulation (4.11) can be represented as a quadratic programming problem with a non-negativity constraint on alphas [15]:

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{A}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha}, \tag{6.19}$$

where $\boldsymbol{A}$ is the Gram matrix of data points whose values are scaled by corresponding label products ($A_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$) and $\mathbf{1}$ denotes an appropriately sized vector of ones. A more general form of quadratic programming can be written as:

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{A}\boldsymbol{\alpha} + \boldsymbol{b}^T\boldsymbol{\alpha}. \tag{6.20}$$

This problem is called Non-negative Quadratic Programming (NQP) when the non-negativity constraint is enforced on $\boldsymbol{\alpha}$. SVM is a special case of NQP.

Parameter free multiplicative updates for NQP have been previously introduced in [17]. For the special case of SVM the updates from [17] have the following form:

$$\boldsymbol{\alpha} = \boldsymbol{\alpha} \odot \frac{\mathbf{1} + \sqrt{\mathbf{1} + 4(\boldsymbol{A}^+\boldsymbol{\alpha}) \odot (\boldsymbol{A}^-\boldsymbol{\alpha})}}{2(\boldsymbol{A}^+\boldsymbol{\alpha})}, \tag{6.21}$$

where $\boldsymbol{A}^+$ and $\boldsymbol{A}^-$ are defined as:

$$A^+_{ij} = \begin{cases} A_{ij} & A_{ij} > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{6.22}$$

$$A^-_{ij} = \begin{cases} |A_{ij}| & A_{ij} < 0, \\ 0 & \text{otherwise}. \end{cases} \tag{6.23}$$

Reformulated SVMs for which we have derived multiplicative updates in Section 5, can be represented as NQP with a special form of $\boldsymbol{A}$ and $\boldsymbol{\alpha}$:

$$\tilde{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{\alpha}_A & \boldsymbol{\alpha}_B \end{bmatrix}^T \tag{6.24}$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} K(\boldsymbol{X}_A, \boldsymbol{X}_A) & -K(\boldsymbol{X}_A, \boldsymbol{X}_B) \\ -K(\boldsymbol{X}_B, \boldsymbol{X}_A) & K(\boldsymbol{X}_B, \boldsymbol{X}_B) \end{bmatrix} \tag{6.25}$$

The block structure of $\tilde{\boldsymbol{A}}$ allows for a clear and easy split of this matrix into $\tilde{\boldsymbol{A}}^+$ and $\tilde{\boldsymbol{A}}^-$ after which it is clear that the multiplicative update of NQP (6.21) is different from the updates in (5.15).

In order to highlight that difference we have generated a random matrix $\boldsymbol{A}$ of form (6.25) for dimension 2 and solved the problem using the method introduced in Section 5 and the update (6.21), introduced in [17]. Convergence paths for both algorithms are shown in Figure 1. The figure shows a paraboloid of the two dimensional objective function generated by a random construction of the Gram matrix satisfying the structure in (6.25). MUSIK and M^3 algorithms [17] were applied to this problem starting at $\boldsymbol{\alpha} = [1, 1]^T$. As expected, both algorithms arrive at the unique solution of the convex problem, however they follow different paths and MUSIK takes fewer steps.

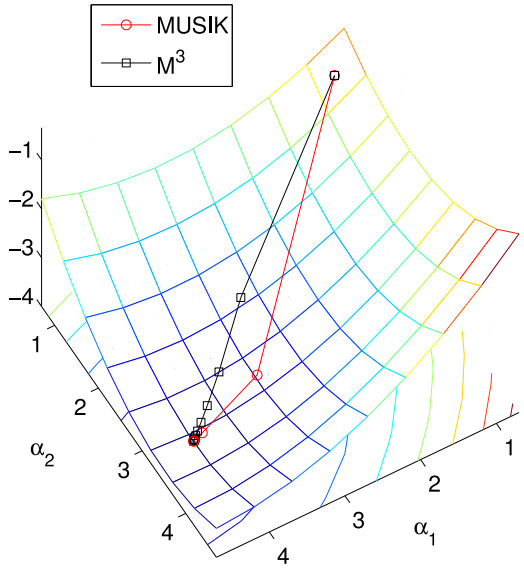

Figure 1: The figure shows a paraboloid of the two dimensional objective function generated by a random construction of the Gram matrix satisfying the structure in (6.25). MUSIK and M^3 algorithms [17] were applied to this problem starting at $\boldsymbol{\alpha} = [1, 1]^T$. As expected, both algorithms arrive at the unique solution of the convex problem, however they follow different routes and the non-negative kernel SVM takes fewer steps.

Figure 1 demonstrates the differences between the methods on a single problem case. In order to have an aggregate measure of the difference we have implemented the following simulation. We have randomly constructed 100 positive definite matrices $\tilde{\boldsymbol{A}}$ with the structure required by our algorithm (6.25) (recall that the structure comes from the requirement of the kernel to be non-negative) for each dimension from the following list: (16, 32, 64, 128, 256, 512, 1024, 2048). Equal number of data points of each class was assumed. For each of these matrices we have solved the QP problem (6.19) using `quadprog` function of Matlab. All 800 problems were constructed to be well conditioned and solvable by this function. Knowing the exact solution to a given problem we ran both MUSIK and M^3 until they were within the given percent of the solution (convergence tolerances of $1\%, 0.1\%$ and 0.01% were used). Although the absolute value of this percent depends on the distance of the optimum from the base hyperplane it is not an issue in our case due to the shift $\boldsymbol{b}$ being equal to $\mathbf{1}$ for all the problems. For each problem we have computed the ratio of the number of iterations it took the M^3 algorithm to reach within the given percent of the solution to the number of iterations it took MUSIK to finish. Results of this simulation are displayed in Figure 2.

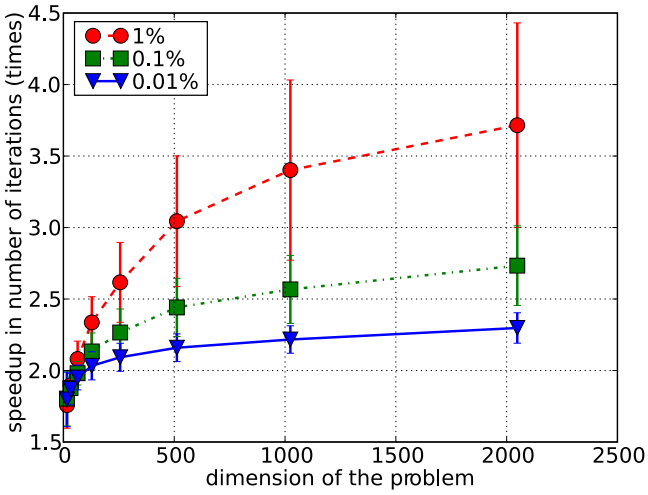

Figure 2: The figure shows the average ratio of the number of iterations for M^3 to the number of iterations for MUSIK taken to achieve given tolerance on the same problem (up is good). Computation is done at error bar points, the lines connecting them are for the visual guide only. The larger the problem size the smaller the number of iterations the algorithm needs compared to M^3, which can be up to 4 times less. Since the running time per iteration is comparable for both algorithms 4 times improvement in iterations means 4 times faster. Even for 0.01% distance from the solution our algorithm is more than two times faster on reasonable sized problems.

7 Decomposition

As we show in the previous section MUSIK updates converge faster than M^3. In part this is due to the better asymptotic bound on the convergence rate which we discuss in Section 11. However, the next feature that improves the convergence rate is splitting $\boldsymbol{\alpha}$ into parts. Separately updating two groups of alphas is similar to decomposition techniques [12], only the way we set the problem does not require any additional heuristics.

In order to demonstrate that decomposition affects the performance in our multiplicative updates, we compare it with MUSIK algorithm in which elements of $\boldsymbol{\alpha}$ are updated simultaneously:

$$\begin{bmatrix}\boldsymbol{\alpha}_A\\ \boldsymbol{\alpha}_B\end{bmatrix} = \begin{bmatrix}\boldsymbol{\alpha}_A\\ \boldsymbol{\alpha}_B\end{bmatrix} \odot \frac{\begin{bmatrix}\boldsymbol{K}_A^- & \boldsymbol{K}_{AB}^+\\ \boldsymbol{K}_{BA}^+ & \boldsymbol{K}_B^-\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_A\\ \boldsymbol{\alpha}_B\end{bmatrix} + \mathbf{1}}{\begin{bmatrix}\boldsymbol{K}_A^+ & \boldsymbol{K}_{AB}^-\\ \boldsymbol{K}_{BA}^- & \boldsymbol{K}_B^+\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_A\\ \boldsymbol{\alpha}_B\end{bmatrix}} \tag{7.26}$$

We call this modification of the algorithm integrative MUSIK (iMUSIK).

Figure 3 shows objective function, training error and testing error as a function of iteration number for MUSIK, iMUSIK and M^3 algorithms. Asymptotically the fastest convergence is exhibited by the MUSIK algorithm and the iMUSIK algorithm fall between MUSIK and M^3. The difference between iMUSIK and MUSIK is only due to the decomposition. Decomposition improves the convergence rate as improved updated parameters are used when updating the remaining parameters.

If we start making the size of the subsets updated at once smaller, we arrive at chunking algorithms of which SMO [13] represents the extreme case. In the extreme case we can update only a single element of $\boldsymbol{\alpha}$ per iteration. In this case we end up with multiplicative variant of the Kernel Adatron (KA) algorithm [5].

KA is a simple gradient ascent procedure for learning support vectors with adaptive learning rate. It has a learning rate parameter which needs to be set. The updates for kernel adatron are as follow:

$$\alpha_i = \alpha_i + \eta_i(1 - y_i(\sum_j K(x_i, x_j) y_j \alpha_j)) \tag{7.27}$$

where η_i is the learning rate parameter. In the case of support vector machines it is set as $\eta_i = \frac{1}{K(x_i,x_i)}$. If we instead set the learning rate to be

$$\eta_i = \frac{\alpha_i}{\boldsymbol{C}_i \alpha} \tag{7.28}$$

we obtain a multiplicative algorithm for KA through MUSIK updates. Note that the matrix $\boldsymbol{C}$ corresponds to the matrix in the denominator of the updates in equation (7.26) and we subscript it to denote the corresponding row vector. We get the multiplicative updates of MUSIK done sequentially. Kernel adatron (KA) belongs to the class of subset methods and can be shown equivalent to the popular SMO algorithm [6].

When heuristics are used to choose which α_i to update KA demonstrates very fast convergence. Thus it is expected that multiplicative KA with heuristics is considerably faster than MUSIK. However, the attractive feature of M^3 and MUSIK is the absence of hyperparameters, a feature that is removed by the need to use heuristics in multiplicative KA algorithm.

Also, the KA algorithm can be adapted to solve the NMF problem. This was indeed done by applying sequential updates to solve nonnegative least squares problem (NNLS) [4]. This was subsequently adapted for solving NMF by Zdunek and Cichoki [18].

8 Soft Margin SVM

We can extend the multiplicative updates to incorporate upper bound constraints of the form $\alpha_i \leq l$ where l is a constant as follows:

$$\alpha_i = \min\{\alpha_i, l\} \tag{8.29}$$

These are referred to as box constraints, since they bound α_i from both above and below.

The dual problem for soft margin SVM is given by:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \quad \text{subject to } 0 \leq \alpha_i \leq l, i \in \{1..n\}. \tag{8.30}$$

The parameter l is a regularization term, which provides a way to avoid overfitting. We note that this objective differs from hard margin SVM (4.11) only in box constraints. Soft margin SVM involves box constraints and that can be handled by the above formulation. At each update of $\boldsymbol{\alpha}$, we implement a step given by (8.29) to ensure the box constraint is satisfied.

9 Bias

SVM with a bias term is given by the following formulation:

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \quad \text{subject to } \alpha_i \geq 0, \sum_i y_i \alpha_i = 0, i \in \{1 \dots n\}. \tag{9.31}$$

We can incorporate bias into MUSIK by considering the following modifications as shown in Keerthi et al. [7]. We introduce a weight variable λ and rewrite the

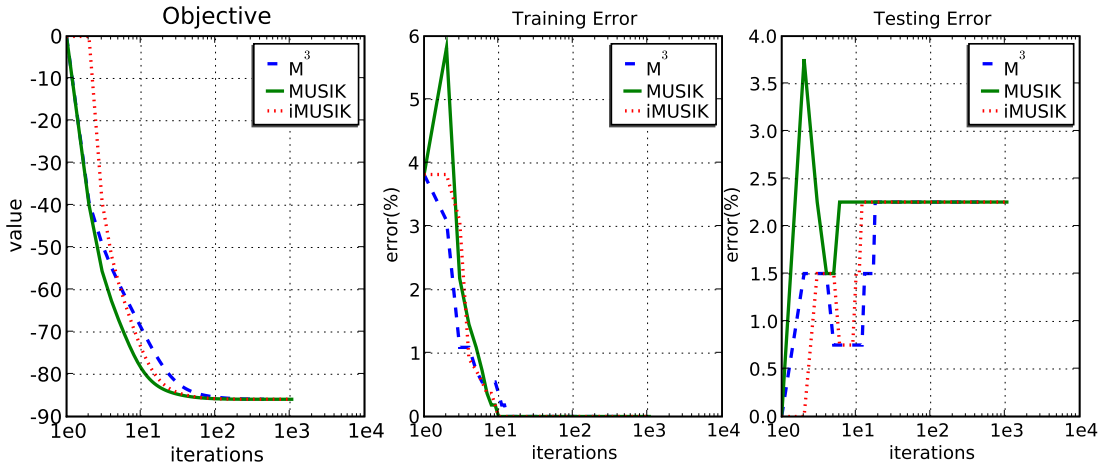

Figure 3: Differences in convergence on the UCI breast cancer dataset for MUSIK, M^3 and integrated MUSIK (iMUSIK) algorithms.

equality constraint $\sum_i y_i \alpha_i = 0$ as the following two equality constraints :

$$\sum_{i \in A} \alpha_i = \lambda, \sum_{i \in B} \alpha_j = \lambda$$

Let us introduce new variables $\beta_k = \alpha_k / \lambda$ for all k and we obtain the following new objective :

$$S_1(\boldsymbol{\beta}, \lambda) = \frac{\lambda^2}{2} \sum_{i,j=1}^{n} \beta_i \beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - 2\lambda \tag{9.32}$$
$$\text{s.t. } \beta_i \geq 0, \sum_{i \in A} \beta_i = 1, \sum_{i \in B} \beta_i = 1.$$

First, we optimize λ keeping the vector β fixed and then alternate by optimizing β keeping λ fixed. Optimizing with respect to λ gives :

$$\lambda = \frac{2}{\sum_i \sum_j \beta_i \beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \tag{9.33}$$

We substitue this value of λ in the above formulation to get the new objective :

$$S_2(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{9.34}$$
$$\text{s.t. } \beta_i \geq 0, \sum_{i \in A} \beta_i = 1, \sum_{i \in B} \beta_i = 1.$$

We update the β vector corresponding to each group alternatingly and derive the following updates similar to Eggert and Körner [3]:

$$\boldsymbol{\beta}_A = \boldsymbol{\beta}_A \odot \frac{\boldsymbol{K}_{AB}\boldsymbol{\beta}_B + \mathbf{1}\boldsymbol{\beta}_A^T \boldsymbol{K}_A \boldsymbol{\beta}_A}{\boldsymbol{K}_A \boldsymbol{\beta}_A + \mathbf{1}\boldsymbol{\beta}_A^T \boldsymbol{K}_{AB} \boldsymbol{\beta}_B} \tag{9.35}$$
$$\boldsymbol{\beta}_B = \boldsymbol{\beta}_B \odot \frac{\boldsymbol{K}_{BA}\boldsymbol{\beta}_A + \mathbf{1}\boldsymbol{\beta}_B^T \boldsymbol{K}_B \boldsymbol{\beta}_B}{\boldsymbol{K}_B \boldsymbol{\beta}_B + \mathbf{1}\boldsymbol{\beta}_B^T \boldsymbol{K}_{BA} \boldsymbol{\beta}_A}$$

when there is normalization involved. The updates are not guaranteed to be non-increasing but in practice converge to global optimum – an observation similar to [3]. The updates assume that the kernels are nonnegative. A nonnegative kernel is one whose output is always nonnegative irrespective of its input. Similar to MUSIK algorithm, the updates can be extended to general kernels.

10 Fixed points

We show that the updates have fixed points wherever the objective function $S(\boldsymbol{\alpha})$ achieves its minimum value. Let $\boldsymbol{\alpha}^*$ be the global minimum. Let us consider the coefficients corresponding to group A. At such a point, we either have that each α_A^i is greater than zero and derivative of objective with respect to α_A^i vanishes or it is zero and derivative is greater than or equal to zero. The first condition applies to the positive elements of $\boldsymbol{\alpha}_A^*$ with the requirement that their corresponding terms in the gradient be zero. The derivatives of these terms

are given by:

$$\begin{aligned}\left.\frac{\partial S}{\partial \alpha_A^i}\right|_{\boldsymbol{\alpha}_A^*} &= (K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A^*)_i \\ &\quad -(K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B^*)_i - 1 \\ &= -(\boldsymbol{K}_{AB}^+\boldsymbol{\alpha}_B^*)_i - (\boldsymbol{K}_A^-\boldsymbol{\alpha}_A^*)_i - 1 \\ &\quad +(\boldsymbol{K}_A\boldsymbol{\alpha}_A^*)_i + (\boldsymbol{K}_{AB}^-\boldsymbol{\alpha}_B^*)_i\end{aligned} \tag{10.36}$$

This condition applies to the support vectors. For non-support vectors corresponding to them being zero we have the second condition. Fixed points occur when one of the following two conditions hold. Either the element to be updated is greater than zero and multiplicative factor is unity or the element is zero. We can see that in the case of the element being non-zero the multiplicative factor is indeed one. Similar analysis can be done for coefficients corresponding to group B. Thus the updates have fixed points wherever the objective reaches its minimum value. We note that at the fixed point M^3 and MUSIK are the same.

11 Asymptotic convergence

The M^3 algorithm [17] observed a rapid decay of non-support vector coefficients and did an analysis of rates of asymptotic convergence. They perturb one of the non-support vector coefficients, say $\boldsymbol{\alpha}_i$ away from the fixed point to some nonzero value $\delta\boldsymbol{\alpha}_i$ and fix all the remaining values. Applying their multiplicative update from (6.21) gives a bound on the asymptotic rate of convergence.

Let $d_i = K(\boldsymbol{x}_i, \boldsymbol{w})/\sqrt{K(\boldsymbol{w}, \boldsymbol{w})}$ denote the perpendicular distance in the feature space from $\boldsymbol{x}_i$ to the maximum margin hyperplane and $d = \min_i d_i = 1/\sqrt{K(\boldsymbol{w}, \boldsymbol{w})}$ denote the one-sided margin to the maximum-margin hyperplane. Also, $l_i = \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)}$ denotes the distance of $\boldsymbol{x}_i$ to the origin in the feature space and $l = \max_i l_i$ denote the largest such distance. The following bound on the asymptotic rate of convergence $\gamma_i^{M^3}$ was established:

$$\gamma_i^{M^3} \leq [1 + \frac{1}{2}\frac{(d_i - d)d}{l_i l}]^{-1} \tag{11.37}$$

We do a similar analysis for rate of asymptotic convergence of the multiplicative updates of the MUSIK algorithm in the case of nonnegative kernels. We perturb one of the non-support vector coefficients fixing all the other coefficients and apply the multiplicative update. This enables us to calculate a bound on rate of convergence. A bound on the asymptotic rate of convergence in terms of geometric quantities is given as follows:

$$\gamma_i^{MUSIK} \leq [1 + \frac{(d_i - d)d}{l_i l}]^{-1} \tag{11.38}$$

The proof sketch can be found in appendix. It is for non-negative kernels, but we note that they constitute the majority of the popular and widely used kernels. We note that our bound is tighter compared to the M^3 algorithm as $\gamma_i^{MUSIK} \leq \gamma_i^{M^3}$.

12 Adapting NMF algorithms for SVM

Multiplicative updates are not the only way to solve NMF-type problems. For example Lin [9] shows a fast projected-gradient algorithm for solving NMF. Zdunek and Cichoki [18] , Dhillon and Sra [1] etc give more algorithms for solving NMF. Projected gradient algorithm can be used for solving SVM with a slight modification to the algorithm. The derivative has to be modified and the rest of the algorithm of updating dual vectors $\boldsymbol{\alpha}$ corresponding to group A and group B alternatively remains.

We will show how to adapt the Landweber method for solving NMF [18] to solve the SVM problem.

Taking the gradient as given in equation 5.13, we can update the dual variables as follows:

$$\boldsymbol{\alpha}_A = \boldsymbol{\alpha}_A - \boldsymbol{\eta} \odot \boldsymbol{d} \tag{12.39}$$

$$\boldsymbol{\eta} = \frac{2}{\boldsymbol{K}_A\mathbf{1}} \tag{12.40}$$

$$\boldsymbol{\alpha}_A = \max(\mathbf{0}, \boldsymbol{\alpha}_A), \tag{12.41}$$

where $\boldsymbol{d}$ corresponds to derivative in 5.13 and max is applied to two vectors element-wise. Similarly, we update the dual variables corresponding to group B given by the vector $\boldsymbol{\alpha}_B$.

13 Power methods

Following the work in [14], we can increase the convergence speed of the algorithms by raising the multiplicative factor in the updates by a power greater than one. In the original work, it was applied to NMF as the Adaptive Overrelaxed NMF(ANMF) algorithm. Given a cost function $C(\alpha)$ over nonnegative α, we can define its positive and negative component of the derivative by $pd = \frac{\partial C(\boldsymbol{\alpha})^+}{\partial \boldsymbol{\alpha}_i}$, $nd = \frac{\partial C(\boldsymbol{\alpha})^-}{\partial \boldsymbol{\alpha}_i}$ respectively. The multiplicative updates can now be written as follows:

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i(\frac{nd}{pd})^\gamma \tag{13.42}$$

where γ is a real number greater than one. This is applied to MUSIK and we get faster convergence as expected.

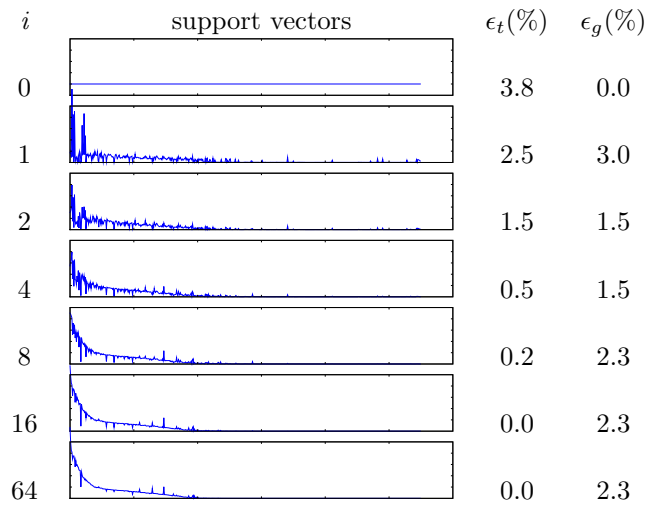

Figure 4: Rate of convergence of multiplicative updates for breast cancer dataset using RBF kernel with $\sigma = 3$. i is the iteration number, ϵ_t is the training error, ϵ_g is the test error. The support vectors have been rearranged for visualization into active and inactive.

Kernel		Breast			Sonar		
		M^3	M	KA	M^3	M	KA
Poly	4	2.26	2.26	2.26	9.62	9.62	9.62
	6	3.76	3.76	3.76	10.58	10.58	10.58
Gaussian	3	2.26	2.26	2.26	11.53	11.53	11.53
	1	0.75	0.75	0.75	7.69	7.69	7.69

Table 1: Misclassification rates (%) on the breast cancer and sonar datasets after convergence of the M^3, MUSIK (M) and Kernel Adatron (KA) algorithms. Polynomial kernels of degree 4 and 6 and Gaussian kernels of σ 1 and 3 were used.

14 Experiments

In order to demonstrate practical applicability of theoretical properties proved in previous section, we test the above updates on two real world problems consisting of breast cancer dataset and aspect-angle dependent sonar signals from the UCI Repository [11]. They contain 683 and 208 labelled examples respectively. The breast cancer dataset was split into 80% and 20% for training and test sets respectively. The sonar dataset was equally divided into test and training sets. The support vectors were all initialized to one. Different kernels involving polynomial and radial basis functions were applied to the dataset. Misclassification rates on the test datasets after 750 iterations are shown in Table 1. They match previously reported error rates on this dataset [17]. The rate of convergence of support vectors is shown in Figure 4.

These results support our derivations and demonstrate that the algorithm can be used for training SVM with non-negative kernels. However, since the problem is convex and there exists a unique solution all correct algorithms will converge to the same solution and arrive at similar classification error rates.

In the following we test the MUSIK algorithm on a medium sized problem of USPS handwritten digits data set. It contains 7291 training examples. We consider the binary class problem with all the samples having digit '2' as labels belong to one and all the rest to another. This was compared with the state of the art multiplicative updates for NQP from [17].

For the experiments we have normalized the USPS dataset to lie in the range $[-1, 1]$ and smoothed it with a 2×2 Gaussian kernel. The non-negative kernel used for the experiment was the Gaussian radial basis function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2}$, with $\sigma = 6.0$. The slack penalty was set to 10.

Our algorithm is slightly faster per iteration due to an extra square root and multiplication per training pattern in the M^3 algorithm. We ignore that slight difference and plot the objective function per iteration of both algorithms on the USPS data set in Figure 5. The result agrees with the theoretically shown upper bound and the simulations from Figure 2.

Figure 6 shows misclassification rate on the training samples using MUSIK and M^3 algorithm.

To test the MUSIK algorithm with sign-insensitive kernel we generate an artificial dataset with 50 samples of each class. We compare convergence speed of M^3, MUSIK, and MUSIK with semiNMF updates. Results are shown in Figure 7.

15 Conclusions

We have derived simple multiplicative update rules for solving the maximum-margin classifier problem in SVMs. No additional parameter tuning is required and the convergence is guaranteed. The updates are straight-forward to implement. In practice the method converges within a few iterations. Extensions to multiple kernel learning are left as future work. The updates could also be used as part of a subset method which could potentially speed up MUSIK algorithm. MUSIK shares the utility of M^3 algorithm in that it is easy to implement in higher-level languages like MATLAB with application to small datasets. It also shares the

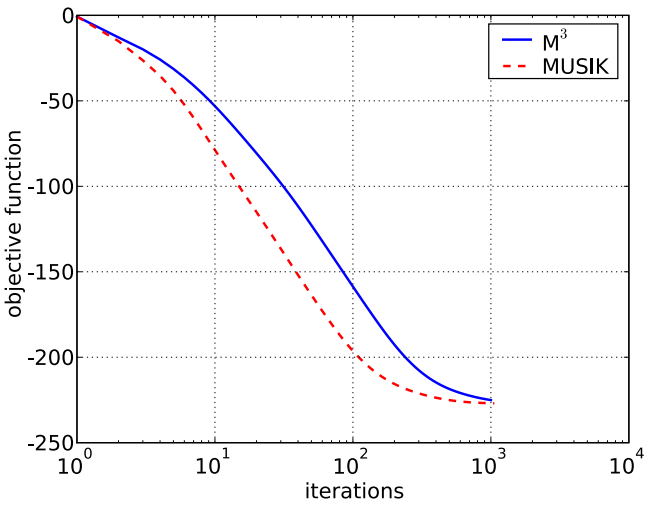

Figure 5: The objective function (4.12) versus training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better).

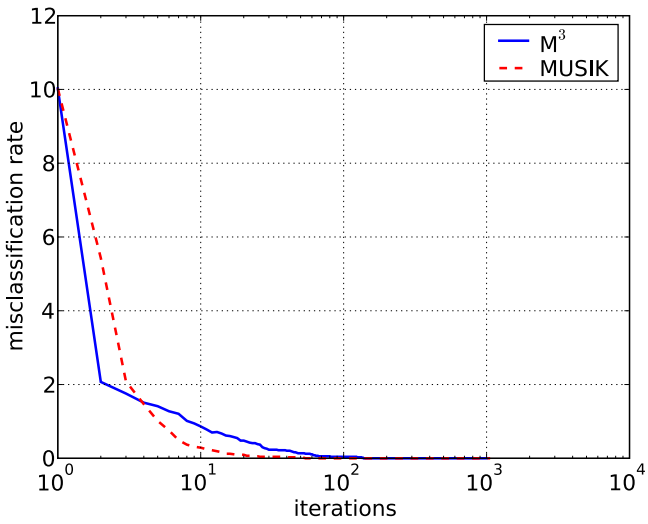

Figure 6: Percentage of the misclassification versus the training iteration number (log scale) on the USPS handwritten digits dataset for the M^3 and the MUSIK algorithms (down is better).

drawback of M^3 in its inability to directly set a variable to zero. However, we have shown MUSIK to have an asymptotically faster rate of convergence compared to M^3 algorithm and we believe this provides a motivation for further research in multiplicative updates for support vector machines. Also the derivation was constructed in such a way that it highlights the connection between SVM and NMF. We also show a connection to the Kernel Adatron algorithm. Sequential updates similar to ones in KA have been used to solve the NMF problem and it would be interesting if heuristics used in KA can be imported to solve NMF-type problems. Since multiplicative updates emerge in different settings and algorithms it might be interesting to find the pattern of when such updates are possible and how to automatically derive them. Our presentation of NMF and SVM correspondence can be considered a step towards this direction.

Acknowledgement

Thanks to Barak A. Pearlmutter for helpful comments. The first author would like to acknowledge the support from NIBIB grants 1 R01 EB 000840 and 1 R01 EB 005846. The second author was supported by NIMH grant 1 R01 MH076282-01. The latter two grants were funded as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program.

References

[1] Inderjit Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with Bregman divergences. In Y. Weiss, B. Schlkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 283–290. MIT Press, Cambridge, MA, 2006.

[2] Chris Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *LBNL Tech Report 60428*, 2006.

[3] J. Eggert and E. Körner. Sparse coding and NMF. *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, 4:2529–2533, 25-29 July 2004.

[4] V. Franc, V. Hlavac, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *Computer Analysis of Images and Patterns*, page 407, 2005.

[5] Thilo-Thomas Frieß, Nello Cristianini, and Colin Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 188–196. Morgan Kaufmann, San Francisco, CA, 1998.

[6] Vojislav Kecman, Michael Vogt, and Te Ming Huang. On the equality of kernel adatron and sequential minimal optimization in classification and regression tasks and alike algorithms for kernel machines. In *ESANN*, pages 215–222, 2003.

[7] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *Neural Networks, IEEE Transactions on*, 11(1):124–136, Jan 2000.

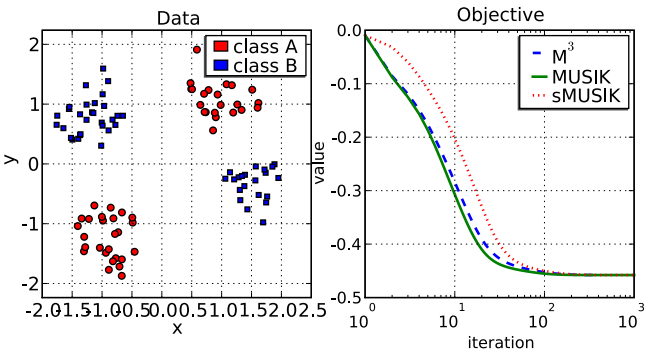

Figure 7: Convergence performance on a dataset containing negative and positive values (shown on the left side) with polynomial kernel of degree 3. MUSIK algorithm with semiNMF updates from Section 5.1 is called sMUSIK in the legend.

[8] Daniel D. Lee and Sebastian H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[9] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Comp.*, 19(10):2756–2779, October 2007.

[10] M. Mørup and L. H. Clemmensen. Multiplicative updates for the LASSO,. *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 33–38, 2007.

[11] C. L. Blake D. J. Newman and C. J. Merz. UCI repository of machine learning databases, 1998.

[12] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285, Sep 1997.

[13] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[14] Ruslan Salakhutdinov and Sam Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of the International Conference on Machine Learning*, volume 20, pages 664–671, 2003.

[15] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.

[16] Fei Sha, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for large margin classifiers. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory (COLT)*, Washington D.C., USA, 2003.

[17] Fei Sha, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for nonnegative quadratic programming in support vector machines. In Sebastian Thrun Suzanna Becker and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[18] Rafal Zdunek and Andrzej Cichocki. Fast nonnegative matrix factorization algorithms using projected gradient approaches for large-scale problems. *Computational Intelligence and Neuroscience*, page 13, 2008.

Appendix

In the following subsections we show how to prove the updates are non-increasing and bound the rate of convergence.

Multiplicative updates and convergence We now derive the update rules for the dual variables $\boldsymbol{\alpha}$. Let us denote the matrices $\Phi(\boldsymbol{X}_A), \Phi(\boldsymbol{X}_B)$ by the matrices $\boldsymbol{M}, \boldsymbol{N}$ and the vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$ by $\boldsymbol{u}, \boldsymbol{v}$ respectively.

The objective is now given by:

$$F(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{M}\boldsymbol{u} - \boldsymbol{N}\boldsymbol{v}\|_2^2 - \sum_i \boldsymbol{v}_i \tag{15.43}$$

We define an auxillary function $G(\boldsymbol{v}, \boldsymbol{v}^t)$ with the properties that $G(\boldsymbol{v}, \boldsymbol{v}) = F(\boldsymbol{v})$ and $G(\boldsymbol{v}, \boldsymbol{v}^t) \geq F(\boldsymbol{v})$. The multiplicative update rule is found at each iteration by minimizing the auxiliary function :

$$\boldsymbol{v}^{t+1} = \arg\min_{\boldsymbol{v}} G(\boldsymbol{v}, \boldsymbol{v}^t) \tag{15.44}$$

We know that this does not increase the objective function F ,as we have

$$F(\boldsymbol{v}^{t+1}) \leq G(\boldsymbol{v}^{t+1}, \boldsymbol{v}^t) \leq G(\boldsymbol{v}^t, \boldsymbol{v}^t) = F(\boldsymbol{v}^t) \tag{15.45}$$

Define G as follows:

$$\begin{aligned} G(\boldsymbol{v}, \boldsymbol{v}^t) = F(\boldsymbol{v}^t) + (\boldsymbol{v} - \boldsymbol{v}^t)\nabla F(\boldsymbol{v}^t) \\ + \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}^t)L(\boldsymbol{v}^t)(\boldsymbol{v} - \boldsymbol{v}^t) \end{aligned} \tag{15.46}$$

where the diagonal matrix $L(\boldsymbol{v}^t)$ is defined as

$$L_{ab}(\boldsymbol{v}^t) = \delta_{ab}\frac{(K_{BA}^- \boldsymbol{u} + K_B^+ \boldsymbol{v}^t)_a}{\boldsymbol{v}_a^t} \tag{15.47}$$

We see that $G(\boldsymbol{v}, \boldsymbol{v}) = F(\boldsymbol{v})$ trivially. The second property that $G(\boldsymbol{v}, \boldsymbol{v}^t) \geq F(\boldsymbol{v})$ is satisfied if

$$0 \leq (\boldsymbol{v} - \boldsymbol{v}^t)^T[L(\boldsymbol{v}^t) - K_B](\boldsymbol{v} - \boldsymbol{v}^t) \tag{15.48}$$

This can be split into three parts as follows:

$$L - K_B = L_1 + L_2 + L_3 \tag{15.49}$$

$$L_1 = diag(\frac{K_{BA}^- \boldsymbol{u}}{\boldsymbol{v}}) \tag{15.50}$$

$$L_2 = diag(\frac{(K_B^+ \boldsymbol{v}^t)}{\boldsymbol{v}^t}) - K_B^+ \tag{15.51}$$

$$L_3 = K_B^- \tag{15.52}$$

We have $L_1 + L_3$ to be positive semidefinite by construction in Section 5.2. If L_2 can be shown to positive semidefinite then the sum is positive semidefinite. L_2 is shown to be true using the argument in [8] which is as follows:

(15.53)
$$\boldsymbol{Q}_{ab}(\boldsymbol{v}^t) = \boldsymbol{v}_a^t (L_2(\boldsymbol{v}^t))_{ab} \boldsymbol{v}_b^t$$

(15.54)
$$\boldsymbol{\nu}^T \boldsymbol{Q} \boldsymbol{\nu} = \sum_{ab} \boldsymbol{\nu}_a \boldsymbol{Q}_{ab} \boldsymbol{\nu}_b$$

(15.55)
$$= \sum_{ab} (K_B^+)_{ab} \boldsymbol{v}_a^t \boldsymbol{v}_b^t [\frac{1}{2}\boldsymbol{\nu}_a^2 + \frac{1}{2}\boldsymbol{\nu}_b^2 - \boldsymbol{\nu}_a \boldsymbol{\nu}_b]$$

(15.56)
$$= \frac{1}{2} \sum_{ab} (K_B^+)_{ab} \boldsymbol{v}_a^t \boldsymbol{v}_b^t (\boldsymbol{\nu}_a - \boldsymbol{\nu}_b)^2$$

(15.57)
$$\geq 0$$

We select the minimum of G. This is found by setting the gradient of G to zero.

$$\boldsymbol{v}^{t+1} = \boldsymbol{v}^t - \frac{\boldsymbol{v}^t}{K_{BA}^- \boldsymbol{u} + K_B^+ \boldsymbol{v}} \odot (K_B \boldsymbol{v}^t - K_{BA} \boldsymbol{u} - \mathbf{1})$$

(15.58)
$$= \boldsymbol{v}^t \odot \frac{K_{BA}^+ \boldsymbol{u} + K_B^- \boldsymbol{v} + \mathbf{1}}{K_{BA}^- \boldsymbol{u} + K_B^+ \boldsymbol{v}}$$

This is the update rule for $\boldsymbol{v}$ and similarly we can derive the update rule for $\boldsymbol{u}$.

Convergence rate Let the fixed point be $\boldsymbol{\alpha}^*$. Let us denote $K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A^*$ by $\boldsymbol{z}^+$ and $K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B^*$ by $\boldsymbol{z}^-$. If we choose an ith non-support vector coefficient from $\boldsymbol{\alpha}_A$, then we have $\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- \geq 1$.

Let the multiplicative factor be denoted by γ_i. We then have:

(15.59)
$$\frac{1}{\gamma_i} = \frac{\boldsymbol{z}_i^+}{\boldsymbol{z}_i^- + 1}$$

(15.60)
$$= 1 + \frac{\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- - 1}{\boldsymbol{z}_i^- + 1}$$

(15.61)
$$\geq 1 + \frac{K(\boldsymbol{x}_i, \boldsymbol{w}) - 1}{\boldsymbol{z}_i^+}$$

where we have $\boldsymbol{w} = \sum_i \alpha_i^* x_i y_i$ is the normal vector to the maximum margin hyperplane. We have used the following:

$$\boldsymbol{z}_i^+ - \boldsymbol{z}_i^- = \sum_{j \in A} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{\alpha}_j^* - \sum_{k \in B} K(\boldsymbol{x}_i, \boldsymbol{x}_k)\boldsymbol{\alpha}_k^*$$

(15.62)
$$= K(\boldsymbol{x}_i, \boldsymbol{w})$$

We now obtain a bound on the denominator:

(15.63)
$$\boldsymbol{z}_i^+ = \sum_{j \in A} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{\alpha}_j^*$$

(15.64)
$$\leq \max_{k \in A} K(\boldsymbol{x}_i, \boldsymbol{x}_k) \sum_{j \in A} \boldsymbol{\alpha}_j^*$$

(15.65)
$$\leq \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} \max_{k \in A} \sqrt{K(\boldsymbol{x}_k, \boldsymbol{x}_k)} K(\boldsymbol{w}, \boldsymbol{w})$$

We have used the Cauchy-Schwartz inequality for kernels and an upper bound for the sum of vector α_A^*.

We do a similar analysis by perturbing an ith non-support vector coefficient from group B. Combining the analysis, we have a lower bound as follows:

$$\frac{1}{\gamma_i} \geq 1 + \frac{K(\boldsymbol{x}_i, \boldsymbol{w}) - 1}{\sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} \max_k \sqrt{K(\boldsymbol{x}_k, \boldsymbol{x}_k)} K(\boldsymbol{w}, \boldsymbol{w})}$$