August 2003

Multiplicative Updates for Large Margin Classifiers

Fei Sha
University of Pennsylvania

Lawrence K. Saul
University of Pennsylvania, lsaul@cis.upenn.edu

Daniel D. Lee
University of Pennsylvania, ddlee@seas.upenn.edu

Postprint version. Published in *Lecture Notes in Computer Science*, Volume 2777, Learning Theory and Kernel Machines, 2003, pages 188-202.
Publisher URL: http://dx.doi.org/10.1007/b12006

Multiplicative Updates for Large Margin Classifiers

Abstract

Various problems in nonnegative quadratic programming arise in the training of large margin classifiers. We derive multiplicative updates for these problems that converge monotonically to the desired solutions for hard and soft margin classifiers. The updates differ strikingly in form from other multiplicative updates used in machine learning. In this paper, we provide complete proofs of convergence for these updates and extend previous work to incorporate sum and box constraints in addition to nonnegativity.

Comments

Postprint version. Published in *Lecture Notes in Computer Science*, Volume 2777, Learning Theory and Kernel Machines, 2003, pages 188-202.
Publisher URL: http://dx.doi.org/10.1007/b12006

Multiplicative Updates for Large Margin Classifiers

Fei Sha[1], Lawrence K. Saul[1], and Daniel D. Lee[2]

[1] Department of Computer and Information Science
[2] Department of Electrical and Systems Engineering
University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104
{feisha,lsaul,ddlee}@seas.upenn.edu

Abstract. Various problems in nonnegative quadratic programming arise in the training of large margin classifiers. We derive multiplicative updates for these problems that converge monotonically to the desired solutions for hard and soft margin classifiers. The updates differ strikingly in form from other multiplicative updates used in machine learning. In this paper, we provide complete proofs of convergence for these updates and extend previous work to incorporate sum and box constraints in addition to nonnegativity.

1 Introduction

Many problems in machine learning involve optimizations with nonnegativity constraints. Examples include classifi cation by support vector machines [22], density estimation in Bayesian networks [1], and dimensionality reduction by nonnegative matrix factorization [13]. The optimizations for these problems cannot be solved in closed form; thus, iterative learning rules are required that converge in the limit to actual solutions.

The simplest such learning rule is gradient descent. Minimizing an objective function $F(\mathbf{v})$ by gradient descent involves the additive update:

$$v_i \leftarrow v_i - \eta(\partial F/\partial v_i) \ , \tag{1}$$

where $\eta > 0$ is a positive learning rate, and all the elements of the parameter vector $v = (v_1, v_2, \ldots, v_N)$ are updated in parallel. Gradient descent is not particularly well suited to constrained optimizations, however, because the additive update in (1) can lead to violations of the constraints.

For optimizations with nonnegativity constraints, an equally simple but more appropriate learning rule involves the so-called Exponentiated Gradient (EG) [12]:

$$v_i \leftarrow v_i e^{-\eta(\partial F/\partial v_i)} \ . \tag{2}$$

Equation (2) is an example of a multiplicative update. Because the elements of the exponentiated gradient are always positive, this update naturally enforces the nonnegativity constraints on v_i. By taking the logarithm of both sides of (2), we can view the EG update as an additive update[3] in the log domain:

$$\log v_i \leftarrow \log v_i - \eta(\partial F/\partial v_i) \ . \tag{3}$$

[3] This update differs slightly from gradient descent in the variable $u_i = \log v_i$, which would involve the partial derivative $\partial F/\partial u_i = v_i(\partial F/\partial v_i)$ as opposed to what appears in (3).

Multiplicative updates such as EG typically lead to faster convergence than additive updates [12] if the solution $\mathbf{v}^*$ of the optimization problem is sparse, containing a large number of zero elements. Note, moreover, that sparse solutions are more likely to arise in problems with nonnegativity constraints because in these problems minima can emerge at $v_i^* = 0$ without the the precise vanishing of the partial derivative $(\partial F/\partial v_i)|_{\mathbf{v}^*}$ (as would be required in an unconstrained optimization).

The EG update in (2)—like gradient descent in (1)—depends on the explicit introduction of a learning rate $\eta > 0$. The size of the learning rate must be chosen to avoid divergent oscillations (if η is too large) and unacceptably slow convergence (if η is too small). The necessity of choosing a learning rate can be viewed as a consequence of the generality of these learning rules; they do not assume or exploit any structure in the objective function $F(\mathbf{v})$ beyond the fact that it is differentiable.

Not surprisingly, many objective functions in machine learning have structure that can be exploited in their optimizations—and in particular, by multiplicative updates. Such updates need not involve learning rates, and they may also involve intuitions rather different from the connection between EG and gradient descent in (2–3). For example, the Expectation-Maximization (EM) algorithm [2] for hidden Markov models and the generalized iterative scaling (GIS) algorithm [8] for logistic regression can be viewed as multiplicative updates, but unlike the EG update, they can not be cast as simple variants of gradient descent in the log domain.

In this paper, we derive multiplicative updates for the various problems in nonnegative quadratic programming that arise in the training of large margin classifiers [18, 19, 22]. Our multiplicative updates have the property that they lead to monotonic improvement in the loss function for these classifiers. Interestingly, their form is strikingly different from those of other multiplicative updates used in machine learning, including EG, EM, and GIS. A previous, shorter paper [21] presented our updates for nonnegative quadratic programming in their simplest form. This paper has a stronger theoretical component, not only providing complete proofs of convergence, but also deriving extensions that incorporate sum and box constraints in addition to nonnegativity. It also includes the results of experiments on a larger and more difficult data set. The techniques behind this work should be of general interest to researchers in machine learning faced with problems in constrained optimization.

2 Nonnegative Quadratic Programming

We begin by studying the problem of nonnegative quadratic programming in its simplest form. Consider the minimization of the objective function

$$F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} + \mathbf{b}^{\mathrm{T}}\mathbf{v} \ , \tag{4}$$

subject to the constraints that $v_i \geq 0$ for all i. We assume that the matrix $\mathbf{A}$ is symmetric and semipositive definite, so that the objective function $F(\mathbf{v})$ is bounded below, and its optimization is convex. Due to the nonnegativity constraints, however, there does not exist an analytical solution for the global minimum (or minima), and an iterative solution is needed.

2.1 Multiplicative Updates

Our multiplicative updates are expressed in terms of the positive and negative components of the matrix $\mathbf{A}$ in (4). Let $\mathbf{A}^+$ and $\mathbf{A}^-$ denote the *nonnegative* matrices:

$$A_{ij}^+ = \begin{cases} A_{ij} \text{ if } A_{ij} > 0, \\ 0 \quad \text{otherwise,} \end{cases} \text{ and } A_{ij}^- = \begin{cases} |A_{ij}| \text{ if } A_{ij} < 0, \\ 0 \quad \text{otherwise.} \end{cases} \tag{5}$$

It follows that $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$. In terms of these nonnegative matrices, the objective function can be decomposed as the combination of three terms, which we write as

$$F(\mathbf{v}) = F_a(\mathbf{v}) + F_b(\mathbf{v}) - F_c(\mathbf{v}) \tag{6}$$

for reasons that will become clear shortly. We use the first and third terms in (6) to "split" the quadratic piece of $F(\mathbf{v})$, and the second term to capture the linear piece:

$$F_a(\mathbf{v}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{A}^+\mathbf{v} \ , \tag{7}$$

$$F_b(\mathbf{v}) = \mathbf{b}^T\mathbf{v} \ , \tag{8}$$

$$F_c(\mathbf{v}) = \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{A}^-\mathbf{v} \ . \tag{9}$$

The gradient of $F(\mathbf{v})$ can be similarly decomposed in terms of contributions from these three pieces. We have chosen our notation in (4) and (8) so that $b_i = \partial F_b/\partial v_i$; for the quadratic terms in the objective function, we define the corresponding derivatives:

$$a_i = \frac{\partial F_a}{\partial v_i} = (\mathbf{A}^+\mathbf{v})_i \ , \tag{10}$$

$$c_i = \frac{\partial F_c}{\partial v_i} = (\mathbf{A}^-\mathbf{v})_i \ . \tag{11}$$

Note that these partial derivatives are themselves nonnegative:[4] that is, $a_i \geq 0$ and $c_i \geq 0$. In terms of these derivatives, our updates take the form:

$$v_i \longleftarrow v_i \left[\frac{-b_i + \sqrt{b_i^2 + 4a_ic_i}}{2a_i} \right] \ . \tag{12}$$

These updates are meant to be applied in parallel to all the elements of $\mathbf{v}$. They are remarkably simple to implement, as they neither involve a learning rate nor other heuristic

[4] Some of our proofs in the appendices rely additionally on the *positivity* of $a_i = (\mathbf{A}^+\mathbf{v})_i$ and $c_i = (\mathbf{A}^-\mathbf{v})_i$ whenever $\mathbf{v}$ has no zero elements. If $\mathbf{A}^+$ and $\mathbf{A}^-$ are defined as in (5), then these terms will be strictly positive as long as the matrix $\mathbf{A}$ has at least one positive and negative element in each row. If $\mathbf{A}$ does not satisfy this condition, then $\mathbf{A}^+$ and $\mathbf{A}^-$ can be redefined (for example, by adding a small positive number to each element) so that the proofs remain valid. In this case, the multiplicative updates are not changed in form, merely the definitions of $\mathbf{A}^+$ and $\mathbf{A}^-$. Alternatively, in certain of these degenerate cases, the proofs can be modified while keeping the original decomposition in (5).

criteria that must be tuned to ensure convergence. As we show later, moreover, these updates are guaranteed to decrease the value of $F(\mathbf{v})$ at each iteration.

The reader will recognize the factor multiplying v_i on the right hand side of (12) as the quadratic formula for the positive root of the polynomial $a_i z^2 + b_i z - c_i$. This factor is guaranteed to be nonnegative, as we observed earlier that $a_i \geq 0$ and $c_i \geq 0$. The updates thus naturally enforce the nonnegativity constraints on v_i. The updates are notable for their absence of a learning rate, but even beyond this, their basic form is strikingly different than the EG update in (2). How does this seemingly mysterious combination of partial derivatives [17] serve to minimize the objective function $F(\mathbf{v})$?

An intuition for these multiplicative updates can be gained by examining their fixed points. One fixed point for (12) occurs at $v_i^* = 0$; the other occurs when the positive root of the polynomial $a_i z^2 + b_i z - c_i = 0$ is located at $z = 1$, since in this case the multiplicative factor in (12) is equal to unity. The latter condition, together with the definitions in (6–11), implies that $(\partial F/\partial v_i)|_{\mathbf{v}^*} = a_i + b_i - c_i = 0$. Thus the two criteria for fixed points are either (i) $v_i^* = 0$, or (ii) $(\partial F/\partial v_i)|_{\mathbf{v}^*} = 0$. These are ultimately the same criteria as the EG update in (2).

Further intuition is gained by considering the effects of the multiplicative update away from its fixed points. Although the partial derivative $\partial F/\partial v_i$ does not appear explicitly in (12), there is a close link between the sign of this derivative and the effect of the update on v_i. In particular, using the fact that $\partial F/\partial v_i = a_i + b_i - c_i$, it is easy to show that the update decreases v_i if $\partial F/\partial v_i > 0$ and increases v_i if $\partial F/\partial v_i < 0$. Thus, the multiplicative update in (12) moves each element v_i in the same direction as the EG update in (2), though not in general by the same amount.

The above intuitions are useful, but insufficient to establish that the updates converge to global minima of $F(\mathbf{v})$. In Appendix A, we prove the following theorem:

Theorem 1. *The function $F(\mathbf{v})$ in (4) decreases monotonically to the value of its global minimum under the multiplicative updates in (12).*

The proof of this theorem relies on the construction of an auxiliary function which provides an upper bound on $F(\mathbf{v})$. While many algorithms in machine learning [2, 5, 8, 9, 14, 17] are derived from auxiliary functions, the proof in this paper introduces a particular inequality for nonnegative matrices that we have not seen in previous work.

2.2 Sum Constraint

The multiplicative updates in (12) can be extended to incorporate additional constraints beyond nonnegativity. One such constraint is a linear equality of the form:

$$\sum_i \beta_i v_i = \beta \ , \tag{13}$$

with constant coefficients β_i and constant sum β. We will refer to such a constraint as a sum constraint. In what follows, we assume that the feasible region resulting from the sum and nonnegativity constraints is not empty.

The sum constraint in (13) is enforced by introducing a Lagrange multiplier λ at each iteration of the multiplicative updates. In particular, to incorporate the sum constraint, we consider the "extended" objective function:

$$F(\mathbf{v}, \lambda) \;=\; \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} + \mathbf{b}^{\mathrm{T}}\mathbf{v} + \lambda\left(\sum\nolimits_i \beta_i v_i - \beta\right) \;\; . \tag{14}$$

Suppose that λ is fixed to some value; then its overall effect in (14) is to alter the coefficients of the term that is linear in v_i by an amount $\lambda\beta_i$. As shorthand notation to capture this effect in the multiplicative update, we let

$$r_i(\lambda) \;=\; \frac{-(b_i + \lambda\beta_i) + \sqrt{(b_i + \lambda\beta_i)^2 + 4a_i c_i}}{2a_i} \tag{15}$$

denote the positive root of the polynomial $a_i z^2 + (b_i + \lambda\beta_i)z - c_i$, where as in the previous section, $a_i = (\mathbf{A}^{+}\mathbf{v})_i$ and $c_i = (\mathbf{A}^{-}\mathbf{v})_i$. Then, a multiplicative update that decreases $F(\mathbf{v}, \lambda)$ for fixed λ is given by:

$$v_i \longleftarrow v_i r_i(\lambda) \;\; . \tag{16}$$

At each iteration, the unknown λ should be chosen so that the updated values of v_i satisfy the sum constraint in (13). In terms of the existing values of v_i, this is done by choosing λ to satisfy:

$$\sum_i \beta_i v_i r_i(\lambda) = \beta \;\; . \tag{17}$$

If λ is chosen at each iteration[5] to enforce the constraint, such that $F(\mathbf{v}) = F(\mathbf{v}, \lambda)$, then the multiplicative updates in (16) will serve to minimize $F(\mathbf{v})$ while preserving both the sum and nonnegativity constraints on v_i. The proof of convergence in Appendix A generalizes in a straightforward way to this case. Though impossible to solve (17) for λ in closed form, a simple iterative procedure exists to compute a solution. In particular, in Appendix B, we prove the following:

Theorem 2. *Equation (17) has a unique solution, λ^*, that can be computed by iterating:*

$$\lambda \leftarrow \frac{1}{R}\left(\sum\nolimits_i \beta_i v_i r_i(\lambda) - \beta\right) + \lambda \;\; , \tag{18}$$

where R is any positive constant satisfying $|\sum_i \beta_i v_i \frac{dr_i}{d\lambda}| < R$ for all λ.

2.3 Box Constraints

The multiplicative updates in (12) can also be extended to incorporate constraints of the form $v_i \leq \kappa$ for all i, where κ is a constant. These are referred to as box constraints, since they bound v_i from both above and below. A more general constraint is that each v_i has a different upper bound κ_i. By linearly rescaling the variables v_i, however, this problem can be transformed into the previous one. Interestingly, though box constraints

[5] The required value of λ changes from one iteration to the next (though eventually it stabilizes).

are nonlinear, their handling is fairly trivial and indeed much simpler than enforcing the linear sum constraint in the previous section. The simplest way to enforce the box constraints is to clip the output of the updates in (12), such as:

$$v_i \longleftarrow \min \left\{ \kappa, v_i \left[\frac{-b_i + \sqrt{b_i^2 + 4a_i c_i}}{2a_i} \right] \right\} \quad . \tag{19}$$

This clipped update is also guaranteed to decrease the objective function $F(\mathbf{v})$ in (4) if it results in a change of v_i. It is not straightforward, however, to combine this clipped update with the sum constraint in the previous section. Thus, for problems in which both sum and box constraints are active, we have developed an additional approach.

Recall that the multiplicative update in (12) increases v_i if $\partial F/\partial v_i < 0$ and decreases v_i if $\partial F/\partial v_i > 0$. Since increasing v_i is equivalent to decreasing $(\kappa - v_i)$, we devise a multiplicative update that "operates" on v_i if $\partial F/\partial v_i \geq 0$ and on $(\kappa - v_i)$ if $\partial F/\partial v_i < 0$. In particular, at each iteration, we define the new variables:

$$\hat{v}_i = \begin{cases} v_i & \text{if } \frac{\partial F}{\partial v_i} \geq 0, \\ \kappa - v_i & \text{otherwise.} \end{cases} \tag{20}$$

Note that since v_i and $\hat{v}_i$ are linearly related, minimizing $F(\mathbf{v})$ is equivalent to minimizing the quadratic form $\hat{F}(\hat{\mathbf{v}})$ obtained by the change of variables in (20). Let

$$\hat{F}(\hat{\mathbf{v}}) = \frac{1}{2}\hat{\mathbf{v}}^{\mathrm{T}}\hat{\mathbf{A}}\hat{\mathbf{v}} + \hat{\mathbf{b}}^{\mathrm{T}}\hat{\mathbf{v}} \quad , \tag{21}$$

where the coefficients $\hat{A}_{ij}$ and $\hat{b}_i$ are chosen such that $\hat{F}(\hat{\mathbf{v}}) - F(\mathbf{v})$ is a constant that does not depend on $\mathbf{v}$. To compute these coefficients, we let $s_i = \mathrm{sgn}(\partial F/\partial v_i)$ denote the sign of $\partial F/\partial v_i$, with $\mathrm{sgn}(0)$ equal to 1. Then v_i can be expressed in terms of $\hat{v}_i$ as:

$$v_i = \hat{v}_i s_i + \kappa(1 - s_i)/2 \quad . \tag{22}$$

The coefficients $\hat{A}_{ij}$ and $\hat{b}_i$ are obtained by substituting (22) into (4) and extracting the coefficients of the terms $\hat{v}_i\hat{v}_j$ and $\hat{v}_i$, respectively. This gives:

$$\hat{A}_{ij} = s_i s_j A_{ij} \quad , \tag{23}$$

$$\hat{b}_i = b_i s_i + (\kappa/2)\sum\nolimits_j s_i(1 - s_j)A_{ij} \quad . \tag{24}$$

By constructing $\hat{F}(\hat{\mathbf{v}})$ in this way, we ensure that all the elements of its gradient are nonnegative: $\partial\hat{F}/\partial\hat{v}_i \geq 0$. Thus, if we define matrices $\hat{A}^{\pm}$ from $\hat{A}$ using the same construction as in (5), and if we define $\hat{a}_i = (\hat{A}^{+}\hat{\mathbf{v}})_i$ and $\hat{c}_i = (\hat{A}^{-}\hat{\mathbf{v}})_i$, then the multiplicative update

$$\hat{v}_i \longleftarrow \hat{v}_i \left[\frac{-\hat{b}_i + \sqrt{\hat{b}_i^2 + 4\hat{a}_i\hat{c}_i}}{2\hat{a}_i} \right] \tag{25}$$

will decrease $\hat{F}(\hat{\mathbf{v}})$ by driving all the variables $\hat{v}_i$ toward zero. Note that by decreasing $\hat{F}(\hat{\mathbf{v}})$, the update also decreases $F(\mathbf{v})$ since the two differ only by a constant.

Moreover, by enforcing the nonnegativity constraint on $\hat{v}_i$, the update enforces the box constraint on v_i: in particular, either v_i decreases toward zero or increases toward κ. To distinguish this scheme from the clipped multiplicative update in (19), we will refer to (25) as the "flipped" multiplicative update, noting that the effect of the change of variables in (20) is simply to reverse the direction of the multiplicative update.

Sum and box constraints can be jointly enforced by combining the ideas in this section and the previous one. In this case, at each iteration the change of variables in (20) is first used to obtain a quadratic form $\hat{F}(\hat{\mathbf{v}})$ with $\partial \hat{F}/\partial \hat{v}_i \geq 0$ for all i. Next, the sum constraint in (13) is rewritten in terms of the variables $\hat{v}_i$. Finally, the multiplicative update in (16) is applied to the variables $\hat{v}_i$.

3 Large Margin Classifiers

The problems in nonnegative quadratic programming studied in Section 2 arise in the training of large margin classifiers. These classifiers—though decades old—have generated renewed interest due to their underlying role in support vector machines (SVMs). SVMs use kernel methods to map inputs into a higher, potentially infinite, dimensional feature space. The maximum margin hyperplane in this feature space is then computed as the decision boundary between classes. SVMs have been applied successfully to many problems in machine learning and statistical pattern recognition [7, 18, 19, 22].

Computing the maximum margin hyperplane in SVMs gives rise to a quadratic programming problem with nonnegativity constraints. There is a large literature on iterative algorithms for nonnegative quadratic programming in general and for SVMs as a special case [3, 7, 18, 19]. In this section, we will begin by briefly reviewing the constrained optimizations that arise in SVMs, then show how the multiplicative updates from Section 2 are applied to these problems. Finally, we will compare the updates to other algorithms for training SVMs.

3.1 Costs and Constraints

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote a training set of labeled examples with binary labels $y_i = \pm 1$, and let $K(\mathbf{x}_i, \mathbf{x}_j)$ denote the kernel used to compute dot products between inputs in the feature space. We consider first the realizable setting where the examples are linearly separable in the feature space generated by the choice of kernel. The decision rule of the maximum margin classifier is given by the signed function:

$$y = \mathrm{sgn}\left(K(\mathbf{w}^*, \mathbf{x}) + b^*\right) \quad , \tag{26}$$

where $\mathbf{w}^*$ is the normal vector to the maximum margin hyperplane and $|b^*|$ is its distance from the origin. The vector $\mathbf{w}^*$ can be written as a weighted sum over examples,

$$\mathbf{w}^* = \sum_i \alpha_i^* y_i \mathbf{x}_i \quad , \tag{27}$$

where the coefficients α_i^* are non-zero only for the so-called support vectors that lie closest to the hyperplane. By convention, the scale of $\mathbf{w}^*$ is set by requiring

$$y_i[K(\mathbf{w}^*, \mathbf{x_i}) + b^*] \;\geq\; 1 \tag{28}$$

for all examples in the training set, with equality holding only for support vectors.

A special case of the above occurs when we simply set $b^* = 0$, constraining the separating hyperplane to pass through the origin. In this case, the weight vector $\mathbf{w}^*$ for the maximum margin classifier (assuming the examples remain linearly separable) is obtained by minimizing the loss function:

$$L(\alpha) = \frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i \ , \tag{29}$$

subject to the nonnegativity constraints $\alpha_i \geq 0$. The coefficients α_i^* of the weight vector are determined by the minimum of this loss function, and the solution satisfies $y_i K(\mathbf{w}^*, \mathbf{x}_i) \geq 1$ for all examples in the training set.

The optimization is only slightly changed when we no longer constrain the separating hyperplane to pass through the origin. In this case, the coefficients α_i^* in (27) are again obtained by minimizing (29), but subject to the sum constraint

$$\sum_i y_i \alpha_i = 0 \ , \tag{30}$$

in addition to the nonnegativity constraints $\alpha_i \geq 0$. The threshold b^* does not appear in (29), but it can be computed from the optimal weight vector using (28). The classifier with $b^* \neq 0$ resulting from the additional sum constraint is guaranteed to have an equal or larger margin than the one whose separating hyperplane is constrained to pass through the origin.

In the realizable setting, there is no need to relax or soften the constraints in (28), and the resulting classifiers are known as hard margin classifiers. For examples that are not linearly separable, one must relax these constraints while attempting to minimize the required degree of slack. The resulting classifiers are known as soft margin classifiers. If a one-norm is used to penalize slack, then the problem for computing the optimal classifier is hardly changed from the realizable setting. In this case, the weight vector $\mathbf{w}^*$ is again obtained by minimizing (29), but now subject to the box constraints

$$0 \leq \alpha_i \leq \kappa \tag{31}$$

in addition to the sum constraint in (30) and the nonnegativity constraints $\alpha_i \geq 0$. The constant κ is a free parameter that measures the penalty per unit slack in the margin constraints (28). The determination of the bias b^* is somewhat more complicated for soft margin classifiers; further details can be found in standard treatments [19].

3.2 Multiplicative Margin Maximization ($\mathbf{M^3}$)

The optimizations required for large margin classifiers are special cases of the problems in nonnegative quadratic programming considered in Section 2. In particular, the loss function in (29) is a special case of (4) with $A_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ and $b_i = -1$. The simplest SVM occurs in the realizable setting where the separating hyperplane is constrained to pass through the origin. This SVM can be trained by the update rule:

$$\alpha_i \longleftarrow \alpha_i \left[\frac{1 + \sqrt{1 + 4(\mathbf{A}^+\alpha)_i(\mathbf{A}^-\alpha)_i}}{2(\mathbf{A}^+\alpha)_i}\right] \ , \tag{32}$$

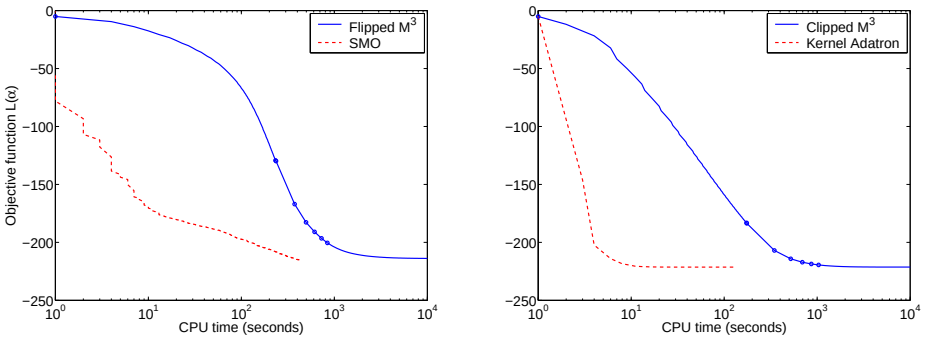

Fig. 1. Plots of the objective $L(\alpha)$ in (29) versus training time on USPS handwritten digits. Left: SMO and the flipped M^3 update in (25). Right: Kernel Adatron and the clipped M^3 update in (19). The circles mark batches of one hundred M^3 updates.

where $\mathbf{A}^{\pm}$ are defined as in (5). For hard margin classifiers whose separating hyperplanes do not pass through the origin, the sum constraint in (30) corresponds to the constraint in Section 2.2; for soft margin classifiers, the box constraints in (31) corresponds to the constraints in Section 2.3. The multiplicative updates in these sections are easily specialized to SVMs, and we refer to this general framework for training large margin classifiers as Multiplicative Margin Maximization (M^3).

We have applied M^3 algorithms for SVMs to three well-known data sets. The first two are the sonar [11] and breast cancer [15] data sets from the UCI Machine Learning Repository. These are small data sets containing 208 and 683 examples, respectively. The third data set is the collection of USPS handwritten digits [20]; this is a larger data set, containing 7291 examples for training. All these data sets have been benchmarked using SVMs. Using M^3 algorithms, we obtained large margin classifiers with similar error rates on training and test sets. This is not too surprising since the same maximum margin hyperplane was being computed for all the benchmarks. Arguably, then, the interesting comparisons are not in terms of error rates, but in terms of other criteria. These are discussed next.

3.3 Comparison to Other Approaches

A large number of algorithms have been investigated for nonnegative quadratic programming in SVMs. We have not attempted an exhaustive comparison, but instead have focused on similarly motivated approaches (such as EG) and on competing approaches that represent the state-of-the-art for large applications.

EG updates have been used to train SVMs [6]. These updates share many of the advantages of M^3 updates: natural handling of nonnegativity constraints, ease of implementation, and simple parallelization. A drawback of EG updates is the need to choose a learning rate and the lack of theoretical guidance for choosing it; this issue does not arise in M^3 updates, which additionally provide a guarantee of monotonic convergence. Note that both EG and M^3 updates are complicated by the sum constraint in (30). It

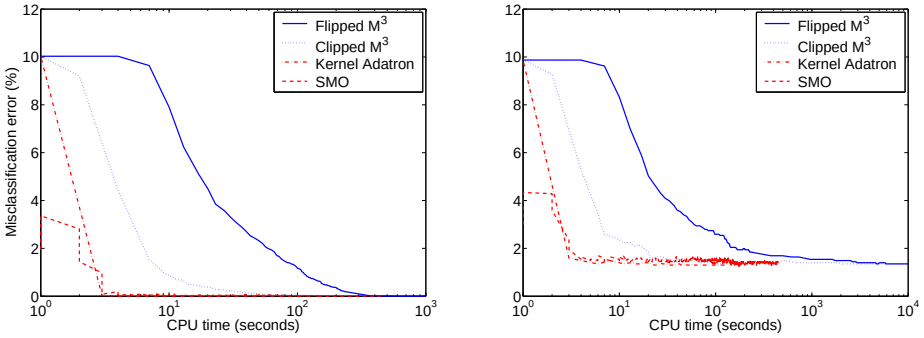

Fig. 2. Left: Percentage of misclassification against training time of the flipped M ^{3} algorithm and SMO on USPS training data. Right: Percentage of misclassification against training time of the clipped M^3 algorithm and Kernel Adatron algorithm on the testing data.

is worth pointing out, however, that for many problems, the bias term in large margin classifiers (though well motivated) does not have a significant effect on generalization.

EG and M^3 updates for training SVMs are both applied in parallel to all the coefficients α_i that appear in the loss function (29). Subset methods constitute a fundamentally different approach to nonnegative quadratic programming. These methods split the variables at each iteration into two sets: a *fixed* set in which the variables are held constant, and a *working* set in which the variables are optimized by an internal subroutine. At the end of each iteration, a heuristic is used to transfer variables between the two sets and improve the objective function.

Two subset methods have been widely used for training SVMs. The first is the method of sequential minimal optimization (SMO) [16], which updates only two coefficients of the weight vector per iteration. In this case, there exists an analytical solution for the updates, so that one avoids the expense of an iterative optimization within each iteration of the main loop. SMO enforces the sum and box constraints for soft margin classifiers. If the sum constraint is lifted, then it is possible to update the coefficients of the weight vector sequentially, one at a time, with an adaptive learning rate that ensures monotonic convergence. This approach is known as the Kernel Adatron [4, 10].

SMO and Kernel Adatron are among the most viable methods for training SVMs on large data sets. Figures 1 and 2 compare the amount of CPU time for these approaches and two types of M^3 updates (flipped and clipped) on the USPS data set of handwritten digits. SVMs were trained to distinguish the digit "2" from the rest of the digits. A Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2}$ was used, with $\sigma = 6.0$, and digit images were smoothed prior to training and testing. The slack penalty was $\kappa = 10$. Parameters ensuring rapid convergence of SMO [16] and Kernel Adatron [4, 10] were set as in previous implementations. Note that SMO and the flipped M^3 updates in (25) enforce both sum and box constraints, while the Kernel Adatron and the clipped M^3 updates in (19) enforce only box constraints. On this data set, the figures show that the M^3 updates take one to two orders of magnitude longer to converge to solutions of similar quality, as

measured either by the minimum value of the objective function, $L(\alpha)$, or by the error rates on the training and test sets.

From these results, it appears that the main utility of the M^3 updates lies in their application to small data sets [21], where computation time is not a primary concern. The simple, parallel form of M^3 updates makes them easy to implement in higher-level languages, such as MATLAB. The parallel form, however, also has its drawbacks. Parallel M^3 updates require more computation per iteration than subset methods, involving the whole Gram matrix for each update. Also, on large data sets with redundant inputs, subset methods such as SMO and Kernel Adatron appear to have the same advantages over M^3 updates as on-line learning algorithms have over batch algorithms. A final drawback of M^3 updates, in their simplest form, is that they cannot set a variable directly to zero. Despite these issues, however, we believe M^3 updates provide an attractive starting point for experimenting with large margin classifiers.

4 Conclusion

In this paper, we have derived multiplicative updates for nonnegative quadratic programming by exploiting hidden structure in the objective function. Several interesting questions remain. For example, the decomposition of $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$ could be achieved in many ways; the particular scheme in (5) is probably not optimal. The optimal decomposition and resulting performance of the algorithm are likely to depend on specific properties of the data set. Of potential interest is the algorithm's behavior on data sets (such as text documents[6]) with sparse inputs and unbalanced numbers of positive and negative examples. The updates could also be used as part of a subset method, instead of being applied completely in parallel. This might be one way to accelerate the M^3 updates for SVMs, by removing the need to multiply by the entire Gram matrix at each iteration. Finally, we are interested in nonnegative quadratic programming problems that arise on graphs—in particular, problems with connections to inference and learning in probabilistic graphical models.

References

1. E. Bauer, D. Koller, and Y. Singer. Update rules for parameter estimation in Bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI*, 1997.
2. L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.
3. C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.
4. C. Campbell and N. Cristianini. Simple learning algorithms for training support vector machines. Technical report, University of Bristol., 1998.
5. M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaBoost, and Bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

[6] We are grateful to the anonymous reviewers for suggesting this line of research.

6. N. Cristianini, C. Campbell, and J. Shawe-Taylor. Multiplicative updatings for support vector machines. In *Proceedings of ESANN'99*, pages 189–194, 1999.
7. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.
8. J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.
9. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–37, 1977.
10. T. Friess, N. Cristianini, and C. Campbell. The Kernel Adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proc. 15th International Conference on Machine Learning*, 1998.
11. R. P. Gorman and T. J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1(1):75–89, 1988.
12. J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.
13. D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999.
14. D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural and Information Processing Systems*, volume 13, Cambridge, MA, 2001. MIT Press.
15. O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.
16. J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
17. L. K. Saul and D. D. Lee. Multiplicative updates for classification by mixture models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural and Information Processing Systems*, volume 14, Cambridge, MA, 2002. MIT Press.
18. B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1999.
19. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
20. B. Schölkopf, K Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classiers. *IEEE Trans. on Signal Processing*, 45(11), 1997.
21. F. Sha, L. K. Saul, and D. D. Lee. Multiplicative updates for nonnegative quadratic programming in support vector machines. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural and Information Processing Systems*, volume 15, Cambridge, MA, 2003. MIT Press.
22. V. Vapnik. *Statistical Learning Theory*. Wiley, N.Y., 1998.

A Proof of Theorem 1

The proof of monotonic convergence is based on the derivation of an auxiliary function. Similar proofs have been used for many models in statistical learning [5, 8, 9, 14, 17].

A.1 Monotonic Convergence

An auxiliary function $G(\tilde{\mathbf{v}}, \mathbf{v})$ has the two crucial properties that $F(\tilde{\mathbf{v}}) \leq G(\tilde{\mathbf{v}}, \mathbf{v})$ and $F(\mathbf{v}) = G(\mathbf{v}, \mathbf{v})$ for all nonnegative $\tilde{\mathbf{v}}$,$\mathbf{v}$. From such an auxiliary function, we can derive

the update rule $\mathbf{v}' = \arg\min_{\tilde{\mathbf{v}}} G(\tilde{\mathbf{v}}, \mathbf{v})$ which never increases (and generally decreases) the objective function $F(\mathbf{v})$:

$$F(\mathbf{v}') \leq G(\mathbf{v}', \mathbf{v}) \leq G(\mathbf{v}, \mathbf{v}) = F(\mathbf{v}) \ . \tag{33}$$

By iterating this procedure, we obtain a series of estimates that improve the objective function. For nonnegative quadratic programming, we derive an auxiliary function $G(\tilde{\mathbf{v}}, \mathbf{v})$ by decomposing $F(\mathbf{v})$ in (4) into three terms and then bounding each term separately:

$$F(\mathbf{v}) = \frac{1}{2}\sum_{ij} A^+_{ij} v_i v_j - \frac{1}{2}\sum_{ij} A^-_{ij} v_i v_j + \sum_i b_i v_i \ , \tag{34}$$

$$G(\tilde{\mathbf{v}}, \mathbf{v}) = \frac{1}{2}\sum_i \frac{(\mathbf{A}^+\mathbf{v})_i}{v_i}\tilde{v}_i^2 - \frac{1}{2}\sum_{ij} A^-_{ij} v_i v_j \left(1 + \log\frac{\tilde{v}_i \tilde{v}_j}{v_i v_j}\right) + \sum_i b_i \tilde{v}_i \ . \tag{35}$$

In the following, we show that $F(\tilde{\mathbf{v}}) \leq G(\tilde{\mathbf{v}}, \mathbf{v})$. We begin by focusing on the first term on the right hand side of (34–35) and establishing that:

$$\frac{1}{2}\sum_{ij} A^+_{ij} \tilde{v}_i \tilde{v}_j \ \leq \ \frac{1}{2}\sum_i \frac{(\mathbf{A}^+\mathbf{v})_i}{v_i}\tilde{v}_i^2 \ . \tag{36}$$

To this end, let δ_{ij} denote the Kronecker delta function, and let $\mathbf{K}$ be the diagonal matrix with elements

$$K_{ij} \ = \ \delta_{ij}\frac{(\mathbf{A}^+\mathbf{v})_i}{v_i} \ . \tag{37}$$

Equation (36) is equivalent to the statement that the matrix $(\mathbf{K} - \mathbf{A}^+)$ is semipositive definite. To show this, we consider the matrix $\mathbf{M}$ whose elements

$$M_{ij} \ = \ v_i(K_{ij} - A^+_{ij})v_j \tag{38}$$

are obtained by rescaling componentwise the elements of $(\mathbf{K} - \mathbf{A}^+)$. Thus, $(\mathbf{K} - \mathbf{A}^+)$ is semipositive definite if $\mathbf{M}$ is semipositive definite. We note that for all vectors $\mathbf{u}$:

$$\mathbf{u}^T\mathbf{M}\mathbf{u} = \sum_{ij} u_i M_{ij} u_j \tag{39}$$

$$= \sum_{ij} u_i u_j v_j \delta_{ij} (\mathbf{A}^+\mathbf{v})_i - \sum_{ij} v_i v_j u_i u_j A^+_{ij} \tag{40}$$

$$= \sum_{ij} v_i v_j A^+_{ij} u_i^2 - \sum_{ij} v_i v_j u_i u_j A^+_{ij} \tag{41}$$

$$= \sum_{ij} v_i v_j A^+_{ij} \left[\frac{1}{2}u_i^2 + \frac{1}{2}u_j^2 - u_i u_j\right] \tag{42}$$

$$= \frac{1}{2}\sum_{ij} A^+_{ij} v_i v_j (u_i - u_j)^2 \tag{43}$$

$$\geq 0 \ . \tag{44}$$

Thus, $(\mathbf{K}-\mathbf{A}^{+})$ is semipositive definite, proving the bound in (36). To bound the second term on the right hand side of (34), we use the inequality: $z \geq 1 + \log z$. The second term on the right hand side of (35) is obtained by substituting $z = \tilde{v}_i \tilde{v}_j / (v_i v_j)$ into this inequality:

$$\tilde{v}_i \tilde{v}_j \geq v_i v_j \left(1 + \log \frac{\tilde{v}_i \tilde{v}_j}{v_i v_j}\right) \quad . \tag{45}$$

Combining (36) and (45), and noting that the third terms on the right hand sides of (34–35) are the same, we have shown that:

$$F(\tilde{\mathbf{v}}) \leq G(\tilde{\mathbf{v}}, \mathbf{v}) \quad . \tag{46}$$

It is easy to verify that $F(\mathbf{v}) = G(\mathbf{v}, \mathbf{v})$. Therefore, $G(\tilde{\mathbf{v}}, \mathbf{v})$ is an auxiliary function, and we can use it to improve $F(\mathbf{v})$. Note that $G(\tilde{\mathbf{v}}, \mathbf{v})$ diverges as $\tilde{v}_i \to 0$; thus, except in degenerate cases, its minimum occurs at positively valued $\tilde{v}_i$. The minimization of $G(\tilde{\mathbf{v}}, \mathbf{v})$ is performed by setting its derivative with respect to $\tilde{v}_i$ to zero, leading to the multiplicative updates in (12). Minimizing $G(\tilde{\mathbf{v}}, \mathbf{v})$ with box constraints on $\tilde{v}_i$ leads to the clipped multiplicative updates in (19).

A.2 Convergence to Global Minimum

The equality in (46) is satisfied if and only if $\tilde{\mathbf{v}} = \mathbf{v}$, implying that the update rule has reached a fixed point. The existence of an auxiliary function guarantees monotonic convergence to a fixed point, though not (in general) convergence to a global minimum. The optimization in nonnegative quadratic programming, however, is convex. Using this, one can prove the second statement of Theorem 1, namely that under the multiplicative updates, the objective function $F(\mathbf{v})$ converges to the value of its global minimum.

Let $\mathbf{v}^*$ be a fixed point that emerges from iteratively applying the update in (12) to an initial vector $\mathbf{v}$ with no zero elements. By examining the sign of the gradient at $\mathbf{v}^*$, we can show that $F(\mathbf{v}^*)$ represents a global minimum. In particular, if $v_i^* \neq 0$, then as shown in Section 2.1, it follows that $(\partial F / \partial v_i)|_{\mathbf{v}^*} = 0$. Alternatively, if $v_i^* = 0$, we can show that $(\partial F / \partial v_i)|_{\mathbf{v}^*} \geq 0$. Together, these are precisely the conditions of the Kuhn-Tucker Theorem, establishing that $F(\mathbf{v})$ attains its global minimum value at $\mathbf{v}^*$.

To prove the latter statement, we suppose that $v_i^* = 0$ occurs at a "reachable" fixed point and show that $(\partial F / \partial v_i)|_{\mathbf{v}^*} < 0$ leads to a contradiction. If the partial derivative is negative at $v_i^* = 0$, then by continuity there exists an $\epsilon > 0$ such that $(\partial F / \partial v_i)|_{\mathbf{v}'} < 0$ for all $\mathbf{v}'$ such that $|\mathbf{v}' - \mathbf{v}^*| < \epsilon$. As observed in Section 2.1, the multiplicative update increases v_i if $\partial F / \partial v_i < 0$; thus in this region, the update will push v_i to larger and larger values until it escapes from the ϵ-region. This leaves only one scenario in which $v_i^* = 0$ could emerge as a reachable fixed point—namely, if an update from outside the ϵ-region sets v_i directly to zero. Examining the update rule, we see that this cannot happen if the terms $a_i = (\mathbf{A}^{+}\mathbf{v})_i$ and $c_i = (\mathbf{A}^{-}\mathbf{v})_i$ are strictly positive when $\mathbf{v}$ does not contain any zero elements. This is easily guaranteed by construction of the matrices A_{ij}^{+} and A_{ij}^{-}; see the footnote in Section 2.1. Thus, we have a contradiction.

B Proof of Theorem 2

First, we show that (17) has a unique solution. Computing the derivative of the left hand side with respect to λ gives:

$$\sum_i \beta_i v_i r_i'(\lambda) \;=\; \sum_i \frac{\beta_i^2 v_i}{2a_i}\left[-1 + \frac{b_i + \lambda\beta_i}{\sqrt{(b_i+\lambda\beta_i)^2 + 4a_i c_i}}\right] \quad . \tag{47}$$

Every term in this sum is strictly negative if $a_i > 0$ and $c_i > 0$, which can be assumed without loss of generality. (See the footnote in Section 2.1.) Thus, the sum as a whole is always negative, implying that the left hand side of (17) decreases monotonically with λ. The existence of a solution to (17) is implied by the assumption of a non-empty feasible region, while the monotonicity of the left hand side establishes uniqueness.

To prove convergence of (18), let λ^t denote the value after t iterations of the update rule, and let λ^* denote its fi xed point. The proof is typical of fi xed point theorems in that we will show each update moves λ^{t+1} closer to λ^* than λ^t. To begin, note that:

$$\left|\lambda^{t+1} - \lambda^*\right| = \left|\lambda^t - \lambda^* + \frac{1}{R}\left(\textstyle\sum_i \beta_i v_i [r_i(\lambda^t) - r_i(\lambda^*)]\right)\right| \quad . \tag{48}$$

Consider the fi nal term on the right hand side of (48). By Taylor's Theorem, there exists a value $\bar{\lambda}$ between λ^t and λ^* such that:

$$\sum_i \beta_i v_i [r_i(\lambda^t) - r_i(\lambda^*)] \;= \sum_i \beta_i v_i (\lambda^t - \lambda^*) r_i'(\bar{\lambda}) \quad . \tag{49}$$

Substituting (49) into the right hand side of (48) and collecting terms, we obtain a mapping of the form:

$$\left|\lambda^{t+1} - \lambda^*\right| = |\lambda^t - \lambda^*| \left|1 + \frac{1}{R}\textstyle\sum_i \beta_i v_i r_i'(\bar{\lambda})\right| \quad . \tag{50}$$

We now exploit the fact that the left hand side of (17) decreases monotonically with λ, and that its derivative is negative and bounded below. In particular, by setting

$$R = \sum_i \frac{\beta_i^2 v_i}{a_i} \tag{51}$$

and appealing to the form of the derivative in (47), it is easily shown that the factor multiplying $|\lambda^t - \lambda^*|$ in (50) is less than unity. It follows that (48) is a contraction mapping, with $|\lambda^{t+1} - \lambda^*| < |\lambda^t - \lambda^*|$. In practice, we have found the iterative procedure in (18) to work well for solving (17), though more sophisticated root-fi nding methods are certainly possible.