# MULTIPLICATIVE UPDATES FOR NON-NEGATIVE KERNEL SVM

*Vamsi K. Potluru*[*2], *Sergey M. Plis*[†2], *Morten Mørup*[3], *Vince D. Calhoun*[1,2,4], *Terran Lane*[2]

[1]Electrical and Computer Engineering Department, University of New Mexico,New Mexico, USA, 87131
[2]Computer Science Department, University of New Mexico, New Mexico, USA, 87131
[3]DTU Informatics, Technical University of Denmark, Denmark
[4]Mind Research Network, New Mexico, USA, 87131

**ABSTRACT**

We present multiplicative updates for solving hard and soft margin support vector machines (SVM) with non-negative kernels. They follow as a natural extension of the updates for non-negative matrix factorization. No additional parameter setting, such as choosing learning, rate is required. Experiments demonstrate rapid convergence to good classifiers. We analyze the rates of asymptotic convergence of the updates and establish tight bounds. We test the performance on several datasets using various non-negative kernels and report equivalent generalization errors to that of a standard SVM.

***Index Terms*—** NMF, SVM, multiplicative updates

## 1. INTRODUCTION

Support vector machines (SVM) are now routinely used for many classification problems in machine learning [1] 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 [2].

SVM can be trained using variants of the gradient descent method applied to the NQP. Although these methods can be quite efficient [3], their drawback is the requirement of setting the learning rate. Subset selection methods are an alternative approach to solving the SVM NQP problem [2]. 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 [2], 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 SVM in [4]. 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.

In this paper we reformulate the dual SVM problem and demonstrate a connection to the non-negative matrix factorization (NMF) algorithm [**?**]. NMF employs multiplicative updates and is very successful in practice due to its independence from the learning rate parameter, low computational complexity and the ease of implementation. The new formulation allows us to devise multiplicative updates for solving SVM with non-negative kernels (the output value of the kernel function is greater or equal to zero). The requirement of a non-negative kernel is not very restrictive since their set includes many popular kernels, such as Gaussian, polynomial of even degree etc. The new updates possess all of the good properties of the NMF algorithm, such as independence from hyper-parameters, low computational complexity and the ease of implementation. Furthermore, the new algorithm converges faster than the previous multiplicative solution of the SVM problem from [4] both asymptotically (a proof is provided) and in practice. We also show how to solve the SVM problem with soft margin using the new algorithm.

## 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 [**?**]. The constraint of non-negativity (all elements are $\geq 0$) 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:

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}. \tag{1}$$

*Thanks to NIBIB grants 1 R01 EB 000840 and 1R01 EB 005846
†Thanks to NIMH grant 1 R01 MH076282-01.

Lee and Seung [?] 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:

$$E = \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F \qquad (2)$$

$$\boldsymbol{W} = \boldsymbol{W} \odot \frac{\boldsymbol{X}\boldsymbol{H}^T}{\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T}, \qquad \boldsymbol{H} = \boldsymbol{H} \odot \frac{\boldsymbol{W}^T\boldsymbol{X}}{\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H}}, \qquad (3)$$

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 [?]. In [?] it is proved that when the algorithm iterates using the updates (3), $\boldsymbol{W}$ and $\boldsymbol{H}$ monotonically decrease the cost function.

## 3. SVM AS NMF

Let the set of labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with binary class labels $y_i = \pm 1$ correspond 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 $\Phi$, 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.

The dual quadratic optimization problem for SVM [1] is given by minimizing the following loss function:

$$S(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \qquad (4)$$
$$\text{subject to } \alpha_i \geq 0, i \in \{1..n\},$$

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 $\Phi$ by performing all operations only in the original data space on $x_i$ and $x_j$, thus defining a Hilbert space $\Phi$.

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 get:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\left(\sum_{ij \in A}\rho_{ij} - 2\sum_{\substack{i \in B \\ j \in A}}\rho_{ij} + \sum_{ij \in B}\rho_{ij}\right) - \sum_{i=1}^{n}\alpha_i \qquad (5)$$
$$\text{subject to } \alpha_i \geq 0, i \in \{1..n\}.$$

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

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\Phi(\boldsymbol{X}_A)\boldsymbol{\alpha}_A - \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B\|_2^2 - \sum_{i \in \{A,B\}} \alpha_i \qquad (6)$$
$$\text{subject to } \alpha_i \geq 0,$$

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 $\Phi$ applied to a matrix corresponds to mapping each individual column vector of the matrix using $\Phi$ and stacking them to generate the new matrix. The vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$ contain coefficients of the support vectors of the two groups $A, B$ respectively. We will use the vector $\boldsymbol{\alpha}$ to denote the concatenation of vectors $\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_B$. Expression (6) resembles NMF with an additional term in the objective [?]. The above formulation enables other metrics $D(\Phi(\boldsymbol{X}_{\boldsymbol{A}})\boldsymbol{\alpha}_A || \Phi(\boldsymbol{X}_B)\boldsymbol{\alpha}_B)$ than least squares for SVM such as more general Bregman divergence [5]. However, to be computationally efficient the metric used has to admit the use of the kernel trick.

## 4. MULTIPLICATIVE ALGORITHM

In this paper, we focus on kernel functions which are non-negative. A kernel function is non-negative when its output value is greater than or equal to zero for all possible inputs in its domain. We note that quite a few of the commonly used kernels are non-negative like Gaussian, polynomials of even degree, etc. We take the derivative of the objective (6) with respect to $\boldsymbol{\alpha}_A$:

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

We slightly abuse notation to define a matrix kernel 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 $\boldsymbol{\Phi}$ for all values of $(i,j)$ in range. We note that the derivative has a positive and a negative component. Similarly, we take the derivative with respect to $\boldsymbol{\alpha}_B$. Recalling the updates for NMF from previous section, we write down the multiplicative updates for this problem (6):

$$\begin{aligned}\boldsymbol{\alpha}_A &= \boldsymbol{\alpha}_A \odot \frac{K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}_B + \mathbf{1}}{K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}_A} \\ \boldsymbol{\alpha}_B &= \boldsymbol{\alpha}_B \odot \frac{K(\boldsymbol{X}_B, \boldsymbol{X}_A)\boldsymbol{\alpha}_A + \mathbf{1}}{K(\boldsymbol{X}_B, \boldsymbol{X}_B)\boldsymbol{\alpha}_B},\end{aligned} \qquad (7)$$

where $\mathbf{1}$ is an appropriately sized vector of ones and $\odot$ denotes Hadamard product as before. We call this new algorithm Multiplicative Updates for Non-negative Kernel SVM (*MUNK*).

The convergence of the above updates follow from the proof of convergence of the regular NMF updates [**?**]. Furthermore, since the Hessian of the joint problem of estimating $\boldsymbol{\alpha}_A$ and $\boldsymbol{\alpha}_B$ is positive semi-definite the alternating updates have no local minima only the global minimum.

## 5. SOFT MARGIN

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:

$$\alpha_i = \min\{\alpha_i, l\} \tag{8}$$

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

The dual problem for soft margin SVM is given by:

$$\min_{\boldsymbol{\alpha}} S(\boldsymbol{\alpha}), \quad \text{subject to } 0 \leq \alpha_i \leq C, i \in \{1..n\}, \tag{9}$$

The parameter $C$ is a regularization term, which provides a way to avoid overfitting. Soft margin SVM involves box constraints that can be handled by the above formulation. At each update of $\boldsymbol{\alpha}$, we implement a step given by (8) to ensure the box constraint is satisfied. This corresponds to potentially reducing the step size of the multiplicative update of an element and since the problem is convex this will still guarantee monotonic decrease of the objective.

## 6. ASYMPTOTIC CONVERGENCE

Sha et al. [4] observed a rapid decay of non-support vector coefficients in the $M^3$ algorithm and performed an analysis of the rate of asymptotic convergence. They perturb one of the non-support vector coefficients, e.g. $\boldsymbol{\alpha}_i$, away from the fixed point to some nonzero value $\delta\boldsymbol{\alpha}_i$ and fix all the remaining values. Applying their multiplicative update 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:

$$\gamma_i^{M^3} \leq \left[1 + \frac{1}{2}\frac{(d_i - d)d}{l_i l}\right]^{-1} \tag{10}$$

We perform a similar analysis for rate of asymptotic convergence of the multiplicative updates of the MUNK algorithm. 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:

| 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 $M^3$, MUNK (M) and Kernel Adatron (KA) algorithms. Polynomial kernels of degree 4 and 6 and Gaussian kernels of $\sigma$ 1 and 3 were used.

$$\gamma_i^{MUNK} \leq \left[1 + \frac{(d_i - d)d}{l_i l}\right]^{-1} \tag{11}$$

The proof sketch can be found in appendix. We note that our bound is tighter compared to the $M^3$ algorithm as $\gamma_i^{MUNK} \leq \gamma_i^{M^3}$.

## 7. EXPERIMENTS

In order to demonstrate the practical applicability of the 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 [6]. 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 training and test sets. The vectors $\boldsymbol{\alpha}$ were initialized the same in all algorithms. Different kernels involving polynomial and radial basis functions were applied to the dataset. For comparison we also provide results for the $M^3$ and Kernel-Adatron (KA) [3] algorithms. Misclassification rates on the test datasets are shown in Table 1. They match previously reported error rates on this dataset [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 the same classification error rates.

MUNK is slightly faster per iteration than $M^3$ 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 MUNK and $M^3$ algorithms on the Breast and Sonar sets in Figure 1. The result agrees

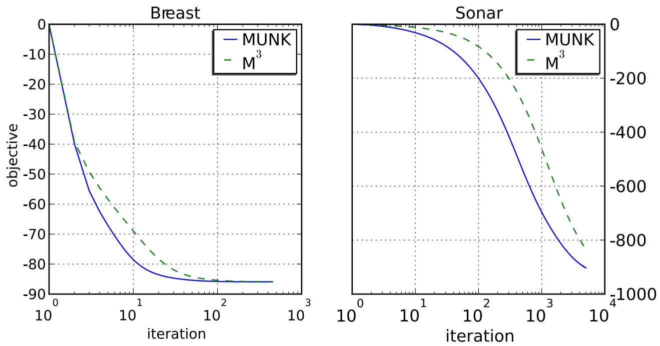


**Fig. 1**. Convergence of the objective with iterations when training with Gaussian kernel ($\sigma = 3$). Lower curve means faster convergence. Note that $x$ axis is logarithmic, indicating a multiplicative speedup for MUNK over a wide operating range.

with the theoretically shown upper bound: MUNK converges about twice as fast as M$^3$.

### 8. CONCLUSIONS

We have derived simple multiplicative update rules for solving the maximum-margin classifier problem in SVMs with non-negative kernels. No additional parameter tuning is required and the convergence is guaranteed. The updates are straight-forward to implement. The updates could also be used as part of a subset method which could potentially speed up MUNK algorithm. MUNK 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 drawback of M$^3$ in its inability to directly set a variable to zero. However, we have shown MUNK 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 with a non-negative kernel and NMF. 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.

### 9. REFERENCES

[1] 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.

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

[3] 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*. 1998, pp. 188–196, Morgan Kaufmann, San Francisco, CA.

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

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

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

## Appendix

Let the fixed point be $\boldsymbol{\alpha}^*$ and $K(\boldsymbol{X}_A, \boldsymbol{X}_A)\boldsymbol{\alpha}^*_A$ be denoted by $\boldsymbol{z}^+$ and $K(\boldsymbol{X}_A, \boldsymbol{X}_B)\boldsymbol{\alpha}^*_B$ by $\boldsymbol{z}^-$. If we choose an $i$th 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 $\gamma_i$. We then have:

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

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

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

where $k_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

We now obtain a bound on the denominator:

$$\begin{aligned} \boldsymbol{z}^+_i &= \sum_{j \in A} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{\alpha}^*_j \leq \max_{k \in A} K(\boldsymbol{x}_i, \boldsymbol{x}_k) \sum_{j \in A} \boldsymbol{\alpha}^*_j \\ &\leq \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} \max_{k \in A} \sqrt{K(\boldsymbol{x}_k, \boldsymbol{x}_k)} K(\boldsymbol{w}, \boldsymbol{w}) \end{aligned}$$

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

We do a similar analysis by perturbing an $i$th non-support vector coefficient from group B. Combining the analysis, the lower bound is:

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