Improving Multi-Class Support Vector Machines Training

Jyothi Bellary, and Dr. Keshava Reddy Eddula

Abstract— In this paper we address the issues related to scalability of Named Entity Recognition using Multi Class Support Vector Machines Solution using high-dimensional input space, by looking at the training computational time factor. This paper presents mathematical foundation of multi-class support vector machines and the evolution of the problem formulation aiming to reduce the training time by improving the optimization algorithms. We then have a look at a new cutting plane algorithm that accelerates the training process while achieving good out-of-the-box performance in linear time. The cutting plane algorithm is first implemented as a standalone java executable in order to study its effectiveness in reducing the computational time.

Keywords— Computational Time, Multi-Class Classification, Support Vector Machines, Training.

I. Introduction

SUPPORT Vector Machine (SVM) is a new and very promising classification technique developed by Vapnik and his group at AT&T Bell Laboratories [1, 7, 8, 9]. Standard SVM training has $O(m^3)$ time and $O(m^2)$ space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. Binary (two-class) classification using support vector machines (SVMs) is a very well developed technique [1] [2]. Due to various complexities, a direct solution of multiclass problems using a single SVM formulation is usually avoided. The better approach is to use a combination of several binary SVM classifiers to solve a given multiclass problem. Popular methods for doing this are: one-versus-all method using winner-takes-all strategy (WTA SVM); one-versus-one method implemented by Max-Wins Voting (MWV SVM); DAGSVM [5]; and error-correcting codes [2]. Hastie and Tibshirani [3] proposed a good general strategy called *pairwise coupling* for combining posterior probabilities provided by individual binary classifiers in order to do multiclass classification. Since SVMs do not naturally give out posterior probabilities, they suggested a particular way of generating these probabilities from the binary SVM outputs and then used these probabilities together with pairwise coupling to do multi-class classification. Hastie and Tibshirani did a quick empirical evaluation of this method against MWV SVM and found that the two methods give comparable generalization performances. Platt [6] criticized Hastie and Tibshirani's method of generating posterior class probabilities for a binary SVM, and suggested the use of a properly designed sigmoid applied to the SVM output to form these probabilities.

Jyothi Bellary is with Aditya College of Engineering, Madanapalle (corresponding author's phone: +919985295046 ; e-mail: jyothibellary@gmil.com).

Dr. E Keshava Reddy, is with Jawaharlal Nehru Technological University Anantapur, Ananthapuramu, Andhra Pradesh , India. (e-mail:keshava_e@rediffmail.com).

II. Algorithms for Training SVM

A. Basic All-Together Multi-Class SVM

The Basic All-Together Multi-Class SVM idea is similar to One-Against-All approach. It constructs k two-class rules where the jth function $w_j^T \emptyset(x) + b$ separates training vectors of the class j from other vectors. There are k decision functions but all are obtained by solving one problem. Similar to the non-separable binary SVM case, the objective of the machine is to maximize the margin separating the different classes while minimizing the classification error of each data point as represented by the slack variable. For a k-class problem with n training points, the multi-class support vector machine can be formulated as the minimization of

$$Q(\mathbf{w}, \mathbf{b}, \xi) = \frac{1}{2} \sum_{j=1}^{k} w_i^T w_i + C \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq y_i}}^{k} \xi_{ij} \quad (1)$$

Subject to

$$w_{y_i}^T \emptyset(x_i) + b_{y_i} \geq w_j^T \emptyset(x_i) + b_j + 1 - \xi_{ij}$$

$$\xi_{ij} \geq 0, \; i = 1 \ldots . n \;\; and \;\; j \in \{1..k\}, \;\; j \neq y_i \quad (2)$$

Where x_i is the input vector for data point i, y_i is the correct class for data point i, ξ_{ij} is the slack variable associated with data point i relative to class j (i.e. the error measure for misclassifying i as belonging to class j), and C is the regularization parameter determining the trade-off between the maximization of the margin and the minimization of the classification error.

Figure 1 is an illustration of different slack variables relative to individual classes, which is the basic way of formulating the Multi-class SVM Problem.

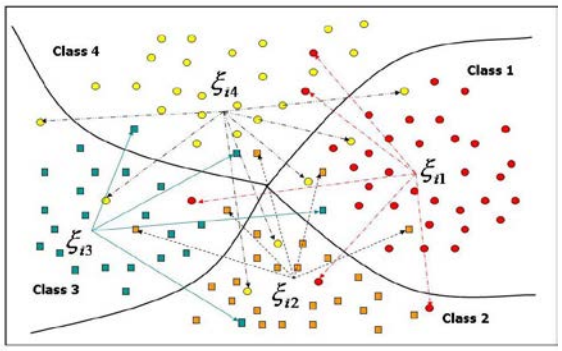

Fig. 1-Multi-Class SVM Error Representation

In the basic multi-class SVM formulation, the machine needs to minimize *k* x *n* slack variables, in addition to maximizing *k* margins. The multi-class classification decision function is defined by $argmax_{j=1\ldots k}\ w_j^T \emptyset(x_i) + b_j$, i.e., a data point *x* is classified as the class *j* whose weights maximize the classification score for the point *x*. The constrained problem in (1) and (2) can be transformed into its unconstrained equivalent formulation by introducing the non-negative Lagrange multipliers α_{ij} and β_{ij}:

$$Q(w,b,\xi,\alpha,\beta) = \frac{1}{2}\sum_{j=1}^{k} w_j^T w_j + C\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k}\xi_{ij}$$

$$-\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k}\alpha_{ij}\left(\left(w_{y_i}-w_j\right)^T\phi(x_i)+b_{y_i}+b_j-1+\xi_{ij}\right)$$

$$-\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k}\beta_{ij}\,\xi_{ij}$$

$$=\frac{1}{2}\sum_{j=1}^{k} w_j^T w_j-\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k} z_{ij}\left(w_j^T\ \phi(x_i)+b_j-1\right)$$

$$-\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k}(\alpha_{ij}+\beta_{ij}-C)\,\xi_{ij} \qquad (3)$$

Where

$$z_{ij} = \sum_{\substack{j=1\\ j\neq y_i}}^{k}\alpha_{ij} \quad for\ j = y_i$$

$$= -\alpha_{ij}\ \text{otherwise} \qquad (4)$$

and the conditions for optimality are :

$$\alpha_{ij}\left(\left(w_{y_i}-w_j\right)^T\phi(x_i)+b_{y_i}+b_j-1+\xi_{ij}\right)=0$$

$$for\ j\neq y_i, j=1\ldots n \qquad (5)$$

$$\beta_{ij}\xi_{ij} = 0\ for\ j\neq y_i, j=1\ldots k, i=1\ldots n \qquad (6)$$

in addition to $Q(w,b,\xi,\alpha,\beta)$ being minimized in **w**,**b**,**ξ** (derivatives equal to zero).

The dual formulation is obtained by reducing (3) to (6) using the kernel function $K(x,y) = \phi(x)^T\phi(y)$. The dual formulation is to maximize:

$$Q(\alpha) = \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq y_i}}^{k}\alpha_{ij} - \frac{1}{2}\sum_{i,l=1}^{n}\sum_{j=1}^{k} z_{ij}z_{lj}\,K(x_i,x_l) \qquad (7)$$

subject to:

$$\sum_{i=1}^{n} z_{ij} = 0\ for\ j\neq y_i, j=1\ldots n \qquad (8)$$

$$0\leq\alpha_{ij}\leq C\ for\ j\neq y_i, j=1\ldots k, i=1\ldots n \qquad (9)$$

Finally, the decision function for class *j* is given by:

$$f_j(x) = \sum_{i=1}^{n} z_{ij}K(x_i,x) + b_j \qquad (10)$$

and the classification task for data point x is to find class *j* to satisfy $argmax_{j=1\ldots k} f_j(x)$.

By examining the dual formulation of the basic multi-class SVM learning problem as defined in (7) to (9), we observe the following:

- Number of variables α_{ij} in the optimization problem is equal to the number of training points *n* times the number of classes *k*, i.e., *n* x *k* variables.
- Number of constraints to be satisfied z_{ij} is equal to number of training points *n*.
- The upper limit for the weight variables α_{ij} is the regularization parameter *C*.

With *n* x *k* variables, the quadratic optimization problem would require $O(n^3k^3)$ computational time to solve – assuming the quadratic optimizer used runs in third power order of the size of its input expressed in number of variables. This is the main issue with the multi-class SVM training. One can easily see that the training time would become prohibitive when a large number of training data points and classes is used for the learning task.

B. Improving SVM Training Time

From the discussion of the basic multi-class formulation, we see that any improvement in training time would require tackling one of the sources of delay by either: reducing the number of data points, reducing the data points' dimensionality, and/or lowering the number of variables and constraints for the optimization problem. Each of these possibilities has been the subject of research activities aiming to improve SVM training. In this paper, we will focus on the approach which is to accelerate training by lowering the number of variables and/or constraints for the optimization task. Crammer and Singer (2001) reduce the number of optimization variables by reducing the number of slack variables α_{ij} to α_i = max(α_{ij}) for *j*=1, .., *k*. In other words, it reduces the size of the optimization problem by considering only the highest slack for each data point across all classes,

thereby having one slack variable per data point. The primal and dual formulations are derived in the same way as for the basic SVM formulation. The mathematical proof is provided in (Crammer and Singer 2001) and extended in (Abe2005) to include bias terms. We will simply provide the initial optimization problem and the final dual formulation in order to contrast it with the basic formulation. Using the *n* slack formulation, the optimization problem is:

$$Q(\mathbf{w}, \mathbf{b}, \xi) = \frac{1}{2}\sum_{j=1}^{k} w_i^T w_i + C\sum_{i=1}^{n}\xi_i \quad (11)$$

Subject to

$$(w_{y_i}^T - w_j^T)\emptyset(x_i) + b_{y_i} - b_j \geq 1 - \xi_{ij} \quad (12)$$

$$\xi_{ij} \geq 0,\ i = 1 \ldots . n \quad and \quad j \in \{1..k\},\ j \neq y_i$$

The dual formulation is to maximize

$$Q(\alpha) = \sum_{i=1}^{n}\xi_i - \frac{1}{2}\sum_{i,l=1}^{n}\sum_{j=1}^{k} Z_{ij}Z_{lj}\alpha_i\alpha_l K(x_i, x_l)$$

Subject to (13)

$$\sum_{i=1}^{n} Z_{ij}\alpha_i = 0 \text{ for } j \neq y_i, j = 1 \ldots k \quad (14)$$

$$0 \leq (n-1)\alpha_i \leq C \text{ for } j \neq y_i, j = 1..k, i = 1..n \quad (15)$$

and the decision function for class j is given by

$$f_j(x) = \sum_{i=1}^{n} Z_{ij}\alpha_j K(x_i, x) + b_j \quad (16)$$

Comparing the basic formulation to that of (Crammer and Singer 2001), we observe that the reduction of the number of slack variables from *n* x *k* to *n* did not increase the number of constraints and the final optimization problem is much simpler that the basic SVM problem. However, using a large number of data points *n*, the optimization time remains high as it requires $O(n3)$ to complete. The optimization algorithm in (Crammer and Singer 2001) is implemented in *SVM-Multiclass* (Tsochantaridis et al. 2004). In order to further improve the training time, one may attempt to reduce the number of data points considered by the optimization algorithm. This may be achieved by approximating the total accurate solution by one that attempts to reach a close accuracy by using a smaller number of data points. Cutting plane algorithms are developed to approximate convex optimization problems by finding data points that approach the optimal solution and discarding the other points.

C. SVM Structural 1-Slack Formulation

We performed a number of experiments using both single class and multi-class cases in order to identify ways to improve the multi-class training time which aimed to identify the scalability issues with All-Together multi-class SVM. Using *SVM-Light* (Joachims 1998a, 2002) for the single class experiments and *SVM-Multiclass* (Crammer and Singer 2001;

Tsochantaridis et al. 2004) for multi-class problems, we observe that the number of support vectors generated in both cases is $O(n^{0.8})$. In fact, *SVM-Multiclass* uses *SVM Light*'s quadratic optimizer, where the input to the optimizer is a realization of (Crammer and Singer 2001). The $O(n^{0.8})$ is an experimental observation using the same training dataset for binary and multi-class training. In general, the worst case estimate for the number of support vectors is $O(n)$, where all training points are potentially support vectors. . The training time using *SVM-Light* and *SVM-Multiclass* is $O(n^2)$, where the multi-class case is $O(k^2)$ slower than the single case case, *k* being the number of classes. With the availability of the improved binary SVM implementation in *SVM-Perf* (Joachims 2006) which reduces the training time to linear time, and knowing that the multi-class solution can be built using the single class one, we analyzed the improved binary implementation in *SVM-Perf* in order to investigate ways to extend the solution to support multi-class training. *SVM-Perf* (Joachims 2006) is based on a newer SVM formulation than that of (Crammer and Singer 2001) which attempts to reduce the number of slack variables from *n* variables to just one. The new formulation is referred to as 1-slack formulation.

The 1-slack structural formulation is (excluding the bias terms):

$$Q(w, \xi) = \frac{1}{2}\sum_{j=1}^{k} w_j^T w_j + C\xi \quad (17)$$

$$\forall c \in \{0,1\}^n : \frac{1}{n} w^T \sum_{i=1}^{n} c_i y_i x_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi \quad (18)$$

$$where\ \xi = \sum_{i=1}^{n}\xi_i$$

Such that

In this formulation, one common slack variable ξ which constitutes an upper bound on all training errors is used. However, the number of constraints in this case is 2^n, one for each possible vector $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in \{0,1\}^n$. Each constraint vector corresponds to the sum of a subset of constraints from the *n*-slack formulation in (11) and (12). The new constraint vectors constitute the input to the quadratic optimizer.

SVM-Perf (Joachims 2006) uses a binary cutting plane algorithm in order to reduce the number of constraints in the problem while providing an approximate solution in a constant number of iterations. Proof of equivalence of the 1-slack formulation to the *n*-slack formulation and convergence of the cutting plane algorithm are provided in (Joachims 2006). *SVM-Perf* binary training algorithm is the following:

Algorithm 1 : Algorithm for training binary classification SVM

1: Input: S={(x_1, y_1),...,(x_n, y_n)},y∈ {-1,1 },C ε
2: C=$\emptyset$ (C is the set of constraints for input to the optimizer)
3: repeat $(\mathbf{W}, \xi) \leftarrow argmin_{W,\xi \geq 0}\{\frac{1}{2}W^T W + C\,\xi\}$
4: such that $\forall c \in C$: $\frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i x_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi$
5: for i=1……n do
6. $c_i = 1$ $y_i(W^T x_i) < 1$
0 Otherwise
7: end of for
8: $C \leftarrow C \cup \{c\}$
9: until $\frac{1}{n}\sum_{i=1}^{n} c_i - \frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i (W^T x_i) \leq \xi + \varepsilon$
10: return (W, ξ)

D. SVM – PerfMulti : New Multi-Class Instantiation

In the general case, the SVM structural 1-slack formulation(Joachims 2005) is the following (unbiased formulation(unbiased formulation):

$$Q(\mathbf{w},\xi) = \frac{1}{2}\sum_{j=1}^{k} w_i^T w_i + C\xi \quad (19)$$

Such that

$$\forall \bar{y}' \in \bar{Y} \backslash \bar{y} : w^T[\varphi(\bar{x},\bar{y})] \geq \Delta(\bar{y}',\bar{y}) - \xi \quad (20)$$

where

$$\bar{y} \in \bar{Y} = (y_1, y_2, \ldots . y_k)$$

is a set of possible k labels and $\varphi(\bar{x},\bar{y}')$ is a function that describes the match between $(x_1,\ldots,x_n)$ and $(\bar{y}_1..\bar{y}_n)$. The objective is to maximize $w^T\varphi(\bar{x},\bar{y}')$ where

$$\varphi(\bar{x},\bar{y}') = \sum_{i=1}^{n} y'_i x_i \quad (21)$$

Designing the structure of the function $\varphi(\bar{x},\bar{y}')$ and a suitable training loss function $\Delta(\bar{y}',\bar{y})$ for a given problem such that the *argmax* is computed efficiently is the main objective for a particular instantiation of *SVM-Struct V3.0* and is left to the designer of the solution. The general algorithm to solve a quadratic optimization problem using the multivariate SVM 1-slack formulation (Joachims 2005) is the following:

Algorithm 2: Algorithm for solving multivariate quadratic optimization problems

1: Input: $\bar{x}$={($x_1,\ldots x_n$)}, $\bar{y}$={($y_1,\ldots y_k$)}, y∈ {-1, 1 }, C, ε

2: C=∅ (C is the set of constraints for input to the optimizer)

3: repeat $(\mathbf{W}, \xi) \leftarrow argmin_{\mathbf{W},\xi\geq 0}\{\frac{1}{2}W^TW + C\,\xi\}$

4: such that $\forall c \in C : \frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i x_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi$

5: for i=1……n do

6: $c_i = \begin{matrix} 1 & y_i(W^T x_i) < 1 \\ 0 & \text{Otherwise} \end{matrix}$

7: end of

8: $C \leftarrow C \cup \{c\}$

9: until $\frac{1}{n}\sum_{i=1}^{n} c_i - \frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i (W^T x_i) \leq \xi + \varepsilon$

10: return (W, ξ)

Tsochantaridis et al. (2004) show that the algorithm terminates after a polynomial number of iterations. The set of constraints **C** is iteratively filled with the most violated constraints found in the input training dataset. Algorithm 1 is an instantiation of Algorithm 2 for the binary classification case.

E. Accelerated Cutting Plane Algorithm

The 1-slack SVM structural formulation expedites the optimization process by reducing the number of variables to be optimized while shifting the decision on how to prepare the quadratic optimizer's input data to the solution designer. Using an error rate loss function, *SVM-Perf* builds one constraint vector for all data points and adds it to the input data vectors for the optimization process. The individual constraint value is zero if the point is correctly classified, or one otherwise. Inspired by the improved training time of *SVM-Perf* as compared to that of *SVM-Light*, we develop a cutting plane algorithm for handling multiple classes at the same time using a loss function based on error rate. In this section we describe a new multi-class instantiation of *SVM-Struct* V3.0 (Tsochantaridis et al. 2005) – *SVM-PerfMulti*. Using the SVM 1-slack formulation, we introduce a cutting plane algorithm that identifies the most violated constraints to be used for the quadratic optimization. The cutting plane algorithm is inspired by the geometrical intuition behind support vector machines illustrated in Figure 1 and Figure 2.

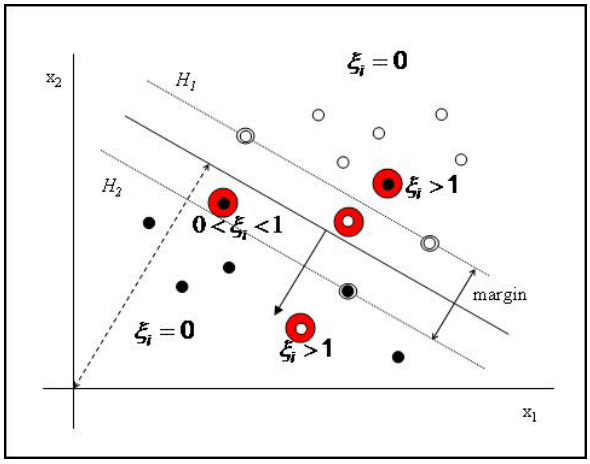

Fig. 2 – SVM Non-Linearly Separable case

Considering the general non-linearly separable case, the objective of the optimization problem is to:

- Maximize the margin(s) separating the hyperplanes
- Minimize the slack error for each data point.

In addition to the optimization objectives, the algorithm also attempts to boost the classification performance of the machine. The cutting plane algorithm iteratively increases the gap between the positive and negative examples for each class, which refines the input to the quadratic optimizer during consecutive learning iterations and accentuates the impact of the scarce positive examples.

Lines 6 to 8 of algorithm aim to satisfy line 4 of Algorithm 2 by finding the set of most violated constraints that $\bar{y}'$ maximizes

$$\varphi(\bar{x},\bar{y}') = \sum_{i=1}^{n} y'_i x_i$$

The *SVM-PerfMulti* (Habib 2008) cutting plane algorithm identifies the most violated constraints by maximizing the difference between the classification score of a data point relative to its own class and the best score among all other classes. If the difference is greater than a loss measure threshold, the data point is considered to be correctly classi_ fied and therefore its associated constraint is not violated. Otherwise, the constraint is violated. In other words, the

greater the difference in classification score between the correct classification and the next best, the more we consider than the trained model is capable of correctly classifying unknown data points. Since the cutting plane decision is based on a slack error criterion, it falls in the category of a slack-rescaling algorithm. However, we use the margin-rescaling option during the optimization process, where individual weights are scaled by a fixed margin factor that is independent of the current loss value.

Algorithm 3: Algorithm for training multi-class classification SVM

1: Input: S={(x_1, y_1),...,(x_n, y_n,)},y∈ {1..k },C,ε,a

2:C=∅ (C is the set of constraints for input to the optimizer)

3: l=initil example loss value=100.0 / n

4: repeat

$$(\mathbf{W},\xi) \leftarrow argmin\ \{\frac{1}{2}\sum_{j=1}^{k} W_j^T W_j + C\,\xi\}$$

5:such that

$$\forall c \in \{0,1\}^n : \frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i x_i \ \geq\ \frac{1}{n}\sum_{i=1}^{n} c_i - \xi$$

Where $\mathbf{W=\{W_1....W_k\}}$

6: for i=1……n do

7: $c_i = 1 \quad \left(W_{y_i}^T x_i\right) - max_{\substack{j\neq y \\ j=1..k}} \left\{w_j^T x_i\right\} \leq l$

= 0 Otherwise

8: end for

9: $C \leftarrow C \cup \{c\}$

10: $l \leftarrow l + a \times max_{i=1..n}\{W_{y_i}^T x_i\}$

11. until $\frac{1}{n}\sum_{i=1}^{n} c_i - \frac{1}{n} W^T \sum_{i=1}^{n} c_i y_i (W^T x_i) \leq \xi + \varepsilon$

12: return (W, ξ)

The initial loss measure threshold is equivalent to that of a total loss, i.e., 100% loss distributed among all data points. The loss threshold is increased after each iteration thereby separating the correct vs. incorrect classifications by a larger distance. We use a heuristic increment based on a fraction of the highest correctly classified score.

To find those positions that maximize line 4 of Algorithm 2 – $\bar{y}' \leftarrow argmax_{\bar{y}\in\bar{Y}}\{\Delta(\bar{y}', \bar{y}) + W^T\Psi(\bar{x}, \bar{y}')\}$– a sufficient condition is to maximize $\Psi(\bar{x}, \bar{y}')$ by classifying the input vectors using the trained model after each optimization cycle. If an input example is incorrectly classified, its associated constraint would be considered violated. In this case, finding the most violated constraints' criteria would be:

$$c_i = 1 \quad \left(w_{y_i}^T x_i\right) \leq max_{\substack{j\neq y_i \\ j=1..k}} \left(w_j^T x_i\right)$$

$$= 0 \quad \text{otherwise} \qquad (22)$$

We will now show that all constraints found by the criteria in the above eqution – all incorrectly classified points – are also found by line 7 of Algorithm 3 where a constraint is considered violated if:

$$\left(w_{y_i}^T x_i\right) - max_{\substack{j\neq y_i \\ j=1..k}} \left(w_j^T x_i\right) \leq l$$

i.e. $$\left(w_{y_i}^T x_i\right) \leq max_{\substack{j\neq y_i \\ j=1..k}} \left(w_j^T x_i\right) + l \qquad (23)$$

If the correct classification score for data point *i* is less than the maximum score for all incorrect classes, $\left(w_{y_i}^T x_i\right) \leq max_{\substack{j\neq y_i \\ j=1..k}} \left(w_j^T x_i\right)$, it will also be less than the maximum score plus a loss threshold *l*, $\left(w_{y_i}^T x_i\right) \leq max_{\substack{j\neq y_i \\ j=1..k}} \left(w_j^T x_i\right) + l$ This means that all violated constraints that need to be found in order to satisfy line 4 of Algorithm 2 are also found by line 7 of Algorithm 3, even if no loss threshold comparison is performed (*l*=0). However, adding the loss threshold comparison will cause a subset of the correctly classified data points to be flagged as incorrectly classified and added to the violated constraints. The loss threshold term *l* is therefore making the correct classification criteria more strict and requiring that the correct example is as far as possible for all other classes. Moreover, by incrementing *l* after each optimization cycle, the classification is further refined and the separation between classes is widened.

III. BOOSTING CLASSIFICATION PERFORMANCE

We hypothesize that the effect of the stricter correct classification decision has the effect of boosting the classification performance. Widening the gap between the correct and incorrect classification for an example has the effect of boosting the weight of the positive examples for the correct class. As part of the investigative experiments , one of the single class experiments using *SVM-Light* assessed the effect of boosting the weights of positive examples by some multiplier factor. Using a preset boosting factor led to improved performance measures up to a certain level, up to a certain point after which performance decreased with higher boosting factors. In Algorithm 3, we do not apply a preset boosting factor but rather use a fraction of the maximum correctly classified score to increase the loss threshold measure. This is a heuristic value indicating the highest sphere of correct scores. One may use other heuristic measures, such as the current value of the loss function $\Delta(\bar{y}', \bar{y})$. However, we observed that using the maximum correct score led to the best and more consistent results with different experimental datasets.

Algorithm 3 falls under the category of *slack-rescaling* algorithms because the violation criteria is based on the difference between the correctly classified score and the incorrect ones. We use slack rescaling for finding the most violated constraints yet use margin rescaling for the

optimization phase. According to the learning constraint in (15), all learned weights are limited by the value of the regularization parameter *C* which governs the trade-off between the slack minimization and the margin maximization. Using the performance boosting mechanism in Algorithm 3 leads to a faster stabilization of both optimization objectives thereby causing the trained model to reach a good out-of-the-box performance independent of the value of *C* when binary features are used.

IV. Reducing Memory Requirements

Using the SVM structural formulation for either binary or multi-class learning, the training time is improved by combining feature vectors into vector(s) of most violated constraints. The generated vectors require larger memory as the size of each constraint vector is O(*f*) in the binary case and O(*kf*) in the multi-class case, where *f* is the number of features in the training set and *k* is the number of classes. E.g., for a training set with 1,000,000 features and 10 classes – and assuming 8-bytes per feature for the feature number and its weight – a binary constraint vector may need up to 8MB of memory while a multi-class vector may need up to 80MB. These estimates constitute a worse-case scenario, where all features are represented in each vector for all classes. In practice, using the JNLPBA-04 training dataset with over a million features and 11 classes, the multi-class constraint vector size was about 0.5MB. Although the support vector size in multi-class training could reach O(*k*) multiples of the corresponding size in the binary case, experiments found that the multi-class overall memory requirements using *SVM-PerfMulti* do not approach this worst case possibility. Since the memory needed depends on the number of constraints – or in other words, the number of learning iterations – reducing the number of iterations will lead to a lowered memory consumption. Algorithm 3 accelerates the learning process thereby reducing the total number of iterations (and support vectors). In order to ensure that all memory allocations performed during the learning process are necessary, we performed extensive process time and memory profiling as part of the empirical analysis. We identified one area of improvement in *SVM-Struct* V3.0 where the trained model is copied at the end of the learning iterations. With the high amount of memory needed for a trained model using a large high-dimensional dataset, experiments using larger data sizes failed due to lack of memory and the overall program aborts without saving the trained model although the actual learning has been completed. Another area of improvement that we identified is to alter the way that *SVM-Light* shrinks the working set of constraint vectors by using a negative number of iterations-to-shrink. This alteration has the effect of lowering the number of support vectors based on inconsistency independent of how long the support vector has been deactivated. The final effect is a reduced number of support vectors in the working set.

V. Conclusion

Having addressed ways to reduce the necessary online memory needed for the learning process, additional reduction may be achieved by using a different medium to store examples and/or support vectors. In the future work, we describe a database supported architecture to alleviate the online memory needs as well as provide a user friendly framework for SVM learning and classification.

References

[1] Boser, B., Guyon, I., Vapnik, V.: An training algorithm for optimal margin classifiers. In: *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, ACM (1992) 144–152.

[2] Dietterich, T., Bakiri, G.: Solving multiclass problem via error-correcting output code. *Journal of Artificial Intelligence Research*, Vol. 2 (1995) 263–286.

[3] Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: Jordan, M.I., Kearns, M.J., Solla, A.S. (eds.): *Advances in Neural Information Processing Systems 10*. MIT Press (1998).

[4] Vapnik, V.: *Statistical Learning Theory*. Wiley Interscience (1998).

[5] Platt, J., Cristanini, N., Shawe-Taylor, J.: Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems 12*. MIT Press (2000) 543–557.

[6] Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Smola, A.J., Bartlett, P., Sch¨olkopf, B., Schuurmans, D. (eds.): *Advances in Large Margin Classifiers*. MIT Press (1999) 61–74.

[7] C.J.C. Burges. Simpli_ed support vector decision rules, 1996.

[8] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:1{25, 1995.

[9] V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, 1995. http://dx.doi.org/10.1007/978-1-4757-2440-0

[10] S. Abe, *Support Vector Machines for Pattern Classification*, Springer-Verlag, London, 2005.

[11] K. Crammer and Y. Singer, *On the Algorithmic Implementation of Multi-class SVMs*, Journal of Machine Learning Research, 2 (2001), pp. 265–292.

[12] M. S. Habib and J. Kalita, *Language and Domain-Independent Named Entity Recognition: Experiment using SVM and High- Dimensional Features*, in J. Kalita and J. Mattoon, eds., *Proc. of the 4th Biotechnology and Bioinformatics Symposium (BIOT-2007)*, Colorado Springs, CO, 2007, pp. 69-73.

[13] C.-W. Hsu and C.-C. Lin, *A Comparison of Methods for Multi- Class Support Vector Machines*, IEEE Transactions on Neural Networks, 13 (2002), pp. 415-425. http://dx.doi.org/10.1109/72.991427

[14] T. Joachims, *Training Linear SVMs in Linear Time*, *Proc. of the ACM Conference on Knowledge Discovery and DataMining (KDD)*, 2006.

[15] T. Joachims, *Learning to Classify Text Using Support Vector Machine*, Kluwer Academic, Norwell, MA, 2002. http://dx.doi.org/10.1007/978-1-4615-0907-3

Jyothi Bellary - pursuing Ph.D from Jawaharlal Nehru Technological University Anantapur, Ananthapuramu. Life Member of ISTE, Member of IAENG, Member of IACSIT. Working as an associate professor, Department of CSE, Aditya College of Engineering, Madanapalle, Chittoor District, Andhra Pradesh, India. Mail id : jyothibellary@gmail.com. Presented and published 3 research papers in National and International Journals and 4 National and International Conferences.

Dr.E Keshava Reddy – Presently Professor of Department of Mathematics and Controller of Examinations, Jawaharlal Nehru Technological University Anantapur, Ananthapuramu. Guided two Ph.D students and one M.Phil student. Adjudicates 9 Ph.D Theses and 8 M.Phil Theses. Many research papers published in National and International Journals. Presented many papers in National and International Conferences. Delivered many guest lectures at various colleges and Universities in the country. Life member of ISTE, IACSIT, Andhra Pradesh Society for for Mathematical Sciences, Indian Mathematical Society, Calcutta Mathematical Society, Allahabad Mathematical Society, Indian Science Congress Association, Marathwada Mathematical Society and Indian Science Congress. Authored 8 books for undergraduate courses. Mail-id : keshava_e@rediffmail.com