Multi-Class Feature Selection with Support Vector Machines

Olivier Chapelle* S. Sathiya Keerthi†

Abstract

We consider feature selection in a multi-class setting where the goal is to find a small set of features for all the classes simultaneously. We develop an embedded method for this problem using the idea of scaling factors. Training involves the solution of a convex program for which we give a scalable algorithm. The method is closely related to extensions of L_1 regularization and recursive feature elimination. These methods are shown to be very effective in text classification.

Key Words: Group Lasso, L_1/L_2 regularization, Support Vector Machines, Feature selection, Text classification, Scaling factors

1. Introduction

Feature selection is an important component of several machine learning applications, e.g. text classification, bioinformatics. It is used to help reduce the load on computational resources and, in cases where there are many noisy features, to help in lifting the performance by eliminating such features.

Guyon and Elisseeff [11] classify feature selection methods into three types: *Filter*, *Wrapper* and *Embedded* methods. Filter methods select features as a preprocessing step; they are cheap, but are not very effective. Wrapper methods use the prediction method as a black box to score subsets of features; they are good, but very expensive. Embedded methods [17], which belong to a middle ground perform feature selection as part of the training process of the prediction method. Linear classifiers that use L_1 regularization on the weights [9, 20] and Recursive Feature Elimination (RFE) [12], a backward elimination method that uses smallness of weights to decide feature removal, fall in this class.

Most feature selection methods suggested in the literature are for binary classification. Many classification problems occurring in practice involve many categories. They are either of a *multi-class* type (assigning exactly one class to each example) or of a *multi-labeled* type (assigning a variable number of classes to each example). In such multi-class and multi-labeled settings[1] it is natural to look for a small common set of features that works well for all the classes. We will refer to this problem as *simultaneous multi-class feature selection*. The need for selecting a small common set of features is also motivated by external constraints. For instance, in text classification, when the average number of features occurring in a document is large it is time consuming to process an individual set of different features for each class. Another example would be the design of a medical diagnosis tool where each feature corresponds to a measurement and is thus expensive to acquire.

It is easy to extend binary filter methods for doing simultaneous multi-class feature subset selection [8, 26]. For example, one can take the information gain values of the individual one-versus-others binary classifiers and combine them (say, via an averaging or max operation) to form a single measure, using which the various features can be ordered. The extension of embedded methods is much less trivial. In this paper we consider the regularized linear classification setting (with SVMs being the particular model taken for implementation) and develop a new embedded method based on scaling factors that addresses the simultaneous feature selection problem. The training process involves a convex program for which we develop a scalable algorithm that works efficiently in a forward selection path tracking mode. This method is closely related to suitable adaptations of L_1 regularization and RFE to deal with simultaneous feature subset selection. In the statistics literature, such L_1 regularization models are also known as Group Lasso [27, 23]. We evaluate these new embedded methods on a number of text classification problems and demonstrate that they are quite superior to a baseline filter method that uses information gain. In parallel works Obozinsky et al [24] and Argyriou et al [1] have developed a similar model for L_1 regularization. They apply the model to multi-task learning and use a block coordinate-wise optimization technique for training.

2. Framework

In this section we present our main ideas for multi-class feature selection. Before doing this we first review Support Vector Machines and two related embedded methods for feature selection, viz. RFE and L_1-SVMs.

2.1 Support Vector Machines

Let us first introduce some notations. In the following, the index i will always run over the training examples $1 \dots n$, j over the features $1 \dots d$, and k over the classes $1 \dots c$. A training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{1 \le i \le n}$ is given, where $\mathbf{x}_i \equiv$

*Yahoo! Research, Santa Clara CA, chap@yahoo-inc.com

†Yahoo! Research, Santa Clara CA, selvarak@yahoo-inc.com

[1] In the rest of the paper we will simply refer to both these types of problems as multi-class problems.

$(x_{i1} \dots x_{id})^\top$ is the d dimensional vector representation of the i-th example and $\mathbf{y}_i \equiv (y_{i1} \dots y_{ic})^\top$ its label vector. $y_{ik} = 1$ if that example belongs to the k-th category and -1 otherwise. The linear classifier for the k-th class uses a d dimensional weight vector, $\mathbf{w}_k$. The j-th element of $\mathbf{w}_k$ will be written as w_{jk}. We will use $\mathbf{w}_k^2$ to denote the square of the Euclidean norm of $\mathbf{w}_k$.

Consider the design of $\mathbf{w}_k$, the weight vector for class k. Support Vector Machines [3] minimize the following objective function[2]

$$\frac{1}{2}\mathbf{w}_k^2 + \frac{C}{2}\sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)). \qquad (1)$$

The loss function ℓ is $\ell(t) = \max(0, 1-t)^p$. We take $p = 2$ in the rest of this paper. SVMs are often used to find non-linear decision boundaries through a *kernel* function, but in this paper we do not employ it. Linear SVMs are effective in many applications such as text classification and bioinformatics [15]. Note that in the linear case, fast methods exist to train SVMs [16]. For instance in that paper, a training time of 10 minutes has been reported for a training set of one million documents.

2.2 Recursive Feature Elimination

One of the standard embedded methods for doing feature selection with SVMs is *Recursive Feature Elimination* (RFE) [12]. It was originally suggested for binary classification. So to begin with let us describe RFE just for one binary classifier, say for the k-th class with weight vector $\mathbf{w}_k$. RFE is based on the idea that the importance of a feature j should be related to the magnitude of its weight $|w_{jk}|$. Combining this idea with a backward elimination gives the following algorithm:

1. Train SVM on the active features.
2. Remove the feature with the smallest $|w_{jk}|$.
3. Go back to step 1 or stop if there are only m features left.

Of course removing one feature at a time in step 2 is time consuming and in practice it is common to remove as much as half of the active features in each iteration. This algorithm is very simple to implement. Also it can be made more efficient by using *seeding*, i.e. when retraining the SVM in step 1, use the weight vector from the previous step as a starting point.

One can think of extending RFE to the multi-class simultaneous feature subset selection problem in various ways. For step 2 of RFE, Chen et al [5] propose using the smallness of $\max_k |w_{jk}|$ as the criterion. One could instead use the smallness of $\sum_k w_{jk}^2$. As we will see in section 2.6 below, this measure has good theoretical support.

[2]A threshold is not included, but can easily be learned by adding a constant component 1 to the examples.

2.3 L_1 Support Vector Machines

Another way of obtaining a sparse solution (i.e. with a few non-zero weight components) is to change the L_2 norm regularizer in (1) by an L_1 norm, $\sum_j |w_{jk}|$. We refer to this model as L_1-Support Vector Machines (L_1-SVM).[3] The use of the L_1 norm tends to give sparse solution. For binary classification this approach has been pioneered by Mangasarian and his colleagues (see for instance [21]). One difficulty with L_1-SVM is that one cannot use standard unconstrained optimization techniques to solve this problem (because of the non differentiability). One can resort to LP solvers [2]. See also [28] for a path tracking method and [9] for a coordinate-wise optimization method.

It is not obvious how the L_1-SVM model can be extended to deal with the simultaneous feature subset selection problem in multi-class settings. Note that replacing the L_2 norm regularizer in (1) by an L_1 norm for each k separately will only lead to the selection of an independent set of different features for that class. Below, in section 2.5 we derive a new L_1-SVM model that addresses the simultaneous feature subset selection problem.

2.4 Feature selection via scaling factors

We now come back to the multi-class case and introduce the framework of scaling factors for doing feature selection.

The goal is to learn c classifications functions $f_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x}$, $1 \le k \le c$ and to do simultaneous feature selection, i.e. find a small set of features which are good for all the classifiers. Suppose we want to find m features. A natural optimization problem to solve is (see also [17, 10]):

$$\min \frac{1}{2}\sum_{k=1}^{c} \mathbf{w}_k^2 + \frac{C}{2}\sum_{i=1}^{n} \ell(y_{ik}\sum_{j=1}^{d} \sqrt{\sigma_j} w_{jk} x_{ij}) \qquad (2)$$

$$\text{subject to the constraints } \sigma_j \in \{0,1\}, \sum \sigma_j = m.$$

The reason for using the square root in (2) will become clear later, but for now it does not change anything since $\sigma_j \in \{0,1\}$. Minimizing this objective function would indeed select m features (those corresponding to $\sigma_j = 1$) and effectively discard the others.

The problem with (2) is that this optimization problem is combinatorial and thus difficult to solve. But we can relax the constraint $\sigma_j \in \{0,1\}$ by $\sigma_j \ge 0$. Also, by making the change of variables $w_{jk} \leftarrow w_{jk}\sqrt{\sigma_j}$, we can then rewrite the relaxed version of (2) as

$$\min \frac{1}{2}\sum_{k=1}^{c} \left(\sum_{j=1}^{d} \frac{w_{jk}^2}{\sigma_j} + C\sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)) \right)$$

$$\text{subject to the constraints } \sigma_j \ge 0, \sum \sigma_j = m.$$

[3]In "L_1-SVM", L_1 refers to the norm of the regularizer. In the literature the same acronym has also been used in a different context to refer to the norm of the training errors.

(In the above, the choice $\sigma_j = 0$ will correspond to $w_{jk} = 0 \ \ \forall k$ and feature j being eliminated.) This is a convex problem because the function $(x, y) \mapsto x^2/y$ is jointly convex for $y > 0$.

Instead of having the constraint $\sum \sigma_j = m$, we can introduce a Lagrange multiplier and add $\frac{\lambda}{2} \sum \sigma_j$ in the objective function and get:

$$\min \frac{1}{2} \sum_{k=1}^{c} \left(\sum_{j=1}^{d} \frac{w_{jk}^2}{\sigma_j} + \lambda \sum \sigma_j + C \sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)) \right)$$

$$\text{subject to the constraints } \ \sigma_j \geq 0.$$

Note that the two problems given above are equivalent: for every m, there exists a λ such that the solutions are identical (and vice versa). Dividing this objective function by $\sqrt{\lambda}$, making the change of variable $\sigma_j \leftarrow \sqrt{\lambda}\sigma_j$ and introducing $\tilde{C} = C/\sqrt{\lambda}$, we obtain the following optimization problem:

$$\min \frac{1}{2} \sum_{k=1}^{c} \left(\sum_{j=1}^{d} \frac{w_{jk}^2}{\sigma_j} + \sum \sigma_j + \tilde{C} \sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)) \right) \tag{3}$$

$$\text{subject to the constraints } \ \sigma_j \geq 0.$$

The sparsity is not controlled by m anymore but by $\tilde{C}$. Small values of $\tilde{C}$ will yield sparse solutions.

Once (3) has been optimized, there are two possibilities:

1. Just take the solution (i.e the $\mathbf{w}_k$) as it is. We will refer to this as the L_1-SVM solution, for reasons that will become obvious below in section 2.5.

2. Go back to the original (non relaxed) formulation (2) by fixing $\sigma_j \leftarrow 1_{\sigma_j > 0}$ and optimize (2). This is equivalent to finding the features through the L_1-SVM algorithm and then train a standard SVM on these features. We call this method SSVM (Sparse Support Vector Machines) as it uses the standard SVM formulation but is constrained to give sparse solutions.

The details of the optimization technique used to solve (3) are given in the appendix. The technique uses a combination of a Newton-type algorithm and a path tracking algorithm; see algorithm 1 for a high level description of the algorithm. In terms of training time, it is very efficient because it only involves matrix vector multiplications, where the matrix contains the active features of the support vectors. Moreover, in some applications such as text classification, this matrix is sparse.

For a given $\tilde{C}$, algorithm 1 is an active set method and it converges finitely. Indeed, for a given F, the algorithm finds the minimizer of (3) and the updating of F implies that the function value decreases at the next iteration. This, together with the finite number of choices for F ensures that algorithm 1 stops after a finite number of iterations.

Algorithm 1 Path tracking algorithm

$\tilde{C} \leftarrow \varepsilon$, $F \leftarrow \emptyset$
repeat
  repeat
    Starting from w, σ, minimize (3) under constraints $w_{jk} = 0$ for $j \notin F$ and $\sigma_j \geq 0$.
    $F \leftarrow \{j \in F, \ \sigma_j > 0\}$ % Keep active features
    $F \leftarrow F \cup \{j \notin F, \ ||g_{jk}|| > 2/\tilde{C}\}$ with $g_{jk} = \partial \sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i))/\partial w_{jk}$.
    % Add new features
  until Set F is stabililized
  Record F and w.
  $\tilde{C} \leftarrow 1.5 \ \tilde{C}$
until Convergence or F has reached maximum size.

Another possibility is to do alternating optimization on w and σ [1]. Such an optimization usually has a slower convergence. But it has the advantage that it is easy to implement: the optimization on w can be done using a standard SVM solver after rescaling each input component by $\sqrt{\sigma_j}$, while the optimal value of σ_j can be found in closed form (see below). This approach is closely related to the MM algorithm [14] whose convergence can be established using the Lyapunov method [18].

2.5 Multi-Class L_1-SVM

For fixed w_{jk} the objective function in (3) can be minimized in closed form with respect to σ_j. Indeed, the optimal value of σ_j is given by $\sigma_j = \sqrt{\sum w_{jk}^2}$. Plugging this value in (3), we obtain

$$\sum_{j=1}^{d} \sqrt{\sum_{k=1}^{c} w_{jk}^2} + \frac{\tilde{C}}{2} \sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)), \tag{4}$$

which can be seen as a multiclass extension of L_1-SVM. Indeed, when there is only a single binary classification problem, the regularizer in (4) is just the L_1 norm of the weight vector. In the multi-class case (4) the optimization is more worrisome, not only because the objective function is not differentiable, but also because linear programming techniques cannot be employed. Some techniques [24, 23, 1] based on block coordinate optimization have however been proposed recently to solve objective functions with the same regularizers as in (4), sometimes referred to as an L_1/L_2 regularizer

2.6 Multi-Class RFE

Another possibility to deal with (2) is the following. Instead of optimizing on σ, fix all of them to the value 1 and decide which one to keep based on the gradient. More precisely, let $T(w, \sigma)$ denote the objective function in (3) (with $\tilde{C} = C$) and define $V(\sigma) = \min_w T(w, \sigma)$. Solving problem (2) is actually equivalent to minimizing V over $\sigma_j \in \{0, 1\}, \sum \sigma_j = m$. One can do something similar

to RFE: start with all the features (i.e. all $\sigma_j = 1$) and put one of the σ_j to 0 such that V is minimized. Keep doing that until m features are left. An approximate way of selecting the feature $\hat{\jmath}$ to suppress is to make a linear approximation of V and select the smallest component of the gradient of V: $\hat{\jmath} = \arg\max \left.\frac{\partial V}{\partial \sigma_j}\right|_{\sigma=1}$. From the definition of V we get,

$$\left.\frac{\partial V}{\partial \sigma_j}\right|_{\sigma_j=1} = \underbrace{\frac{\partial T}{\partial w}}_{=0}\frac{\partial w}{\partial \sigma_j} + \frac{\partial T}{\partial \sigma_j} = \frac{1}{2}(-\sum_k w_{jk}^2 + 1).$$

So $\hat{\jmath} = \arg\min \sum_k w_{jk}^2$. The criterion is very intuitive: it removes features for which the weights are small.

3. Experiments

The performance evaluation was done on several text classification problems in a multi-labeled setting: each class was considered as an independent binary problem. We considered the multi-labeled case because several of our datasets are of this type.

3.1 Algorithms

We evaluated the different algorithms described in the previous section, i.e., RFE, L_1-SVM, and SSVM and compared them to the Information Gain (IG) method, which is known to be one of the best filter methods for text classification [26]. Even though IG has a natural extension to the multi-class scenario, there are different ways of using it for the multi-labeled scenario. We selected the features using the sum of the information gains of all the binary problems as the overall IG criterion. Thus, the score of feature j is:

$$\sum_{k=1}^{c} -H(P(y_{ik}=1)) + P(x_{ij}>0)H(P(y_{ik}=1|x_{ij}>0)) + P(x_{ij}=0)H(P(y_{ik}=1|x_{ij}=0)), \quad (5)$$

where H is the binary entropy function, $H(t) = t\log(t) + (1-t)\log(1-t)$ and P denotes the empirical frequency over $i = 1 \ldots n$.

For all algorithms, the thresholds were further optimized after training to maximize the F-measure on the training set. This is the SCut strategy described in [25] except that we did not use a validation set. For evaluation we used the *microaveraged* F-measure [19, section 5.3].

For RFE, SSVM and IG, the C parameter of the SVM was fixed to 10. This large value corresponds closely to the hard margin SVM. For SSVM and L_1-SVM, $\tilde{C}$ was gradually increased from a small value (corresponding to a small number of features) to higher values until 5000 features are selected. For RFE, at each step of the algorithm, half of the features were discarded.

Table 1: Properties of the datasets used in this paper, viz. the number of training documents (n), the number of features (d) and the number of classes (c). For these datasets marked *, only the c largest classes were retained.

	n	d	c	Description
SECTOR	6412	55197	105	Industry sector [22]
20NG	15935	62061	20	20 Newsgroups [22]
APPAREL	22148	37144	22	Yahoo! Shopping
WEBKB	5505	3000	7	[22]
REUTERS^{*1}	9228	12317	45	ModApte split [22]
RCV1*	11575	47236	52	Half size [19]
OHSUMED[1]	22926	30689	23	

3.2 Datasets

We considered 7 datasets, which are summarized in table 1. Most of them were randomly split into a training set and a test set in a 2:1 ratio. The exact number of training documents can be seen in table 1. Note that the first 4 datasets are multi-class, while the 3 others are multi-labeled. All documents were coded with TFIDF and normalized to norm 1. We used a version of the WEBKB dataset where 3000 features have already been preselected by information gain.

3.3 Results

Performances of the different methods as a function of the number of features are plotted in figure 1. To have a more quantitative evaluation, we also computed the number of features that each method needs to keep in order to achieve a performance 3% below the performance of an SVM trained on all the features (right end of the blue and green curves in figure 1). This was done by computing the intersects of the different curves with a horizontal line at the 3% loss level. We summarize in table 2 the relative feature set size with respect to information gain. For instance a 7 means that a given method has been able to find a feature set size 7 times smaller than IG (for the same performance).

Overall, SSVM is the best performing method. On some datasets, especially SECTOR and APPAREL, IG is quite weak. Despite its simplicity RFE achieved good results. It is noteworthy that L_1-SVM achieves better results on some datasets than a standard SVM trained on all the features (e.g., APPAREL and OHSUMED). However this result in favor L_1-SVM has to be taken with care because it is possible that the value of $C = 10$ that we used for SVM is sub-optimal. Note also that on 20NG the peak performance of L_1-SVM was worse than the SVM trained on all features.

[1]Downloaded from `http://www.kyb.mpg.de/bs/people/pgehler/rap/index.html`

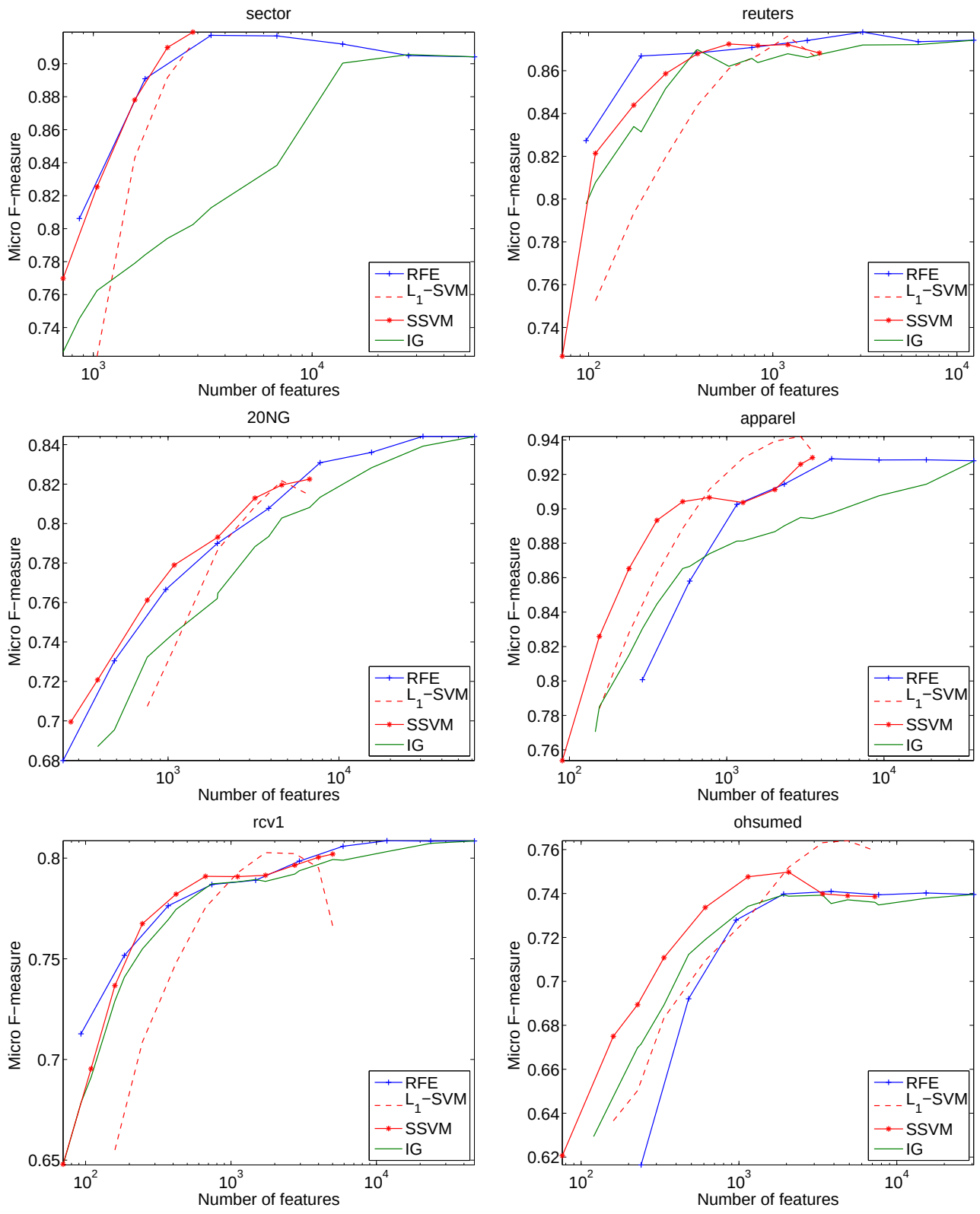

Figure 1: Microaverage F-measure as a function of the number of selected features on different datasets. `webkb` is shown on figure 2

Table 2: Relative reduction of the number of features in comparison to IG. The number of features was selected such that there is 3% loss in performance relative to the SVM trained on all the features. (Thus, higher the number above 1, better is the performance over IG.) Last column: number of features for SSVM.

	IG	RFE	L_1-SVM	SSVM	# Feat
SECTOR	1	7.08	5.92	7.29	1536
20NG	1	1.84	2.78	2.34	4528
APPAREL	1	5.19	8.76	13.36	429
WEBKB	1	0.78	0.42	1.06	83
REUTERS	1	1.68	0.57	1.31	189
RCV1	1	1.00	0.79	1.46	443
OHSUMED	1	0.71	0.74	1.45	400
Mean	1	2.61	2.85	4.04	

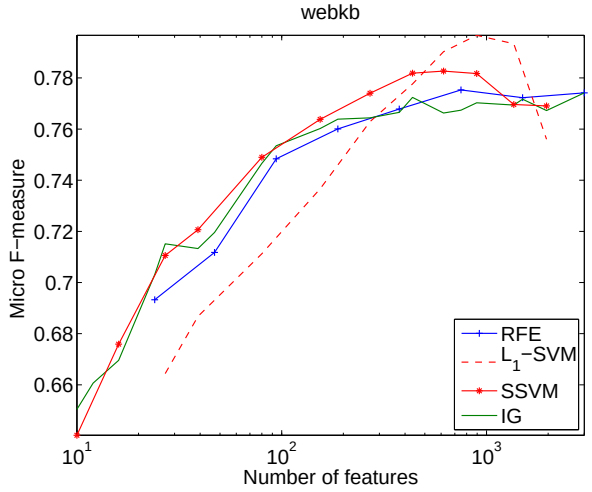

Figure 2: Continuation of figure 1.

We conclude this section with some running time analysis of the different algorithms. We take the OHSUMED dataset for this purpose. As it can be seen from figure 3, running times of L_1-SVM and SSVM are roughly linear with respect to the number of selected features. For selecting 5000 features the typical running times for L_1-SVM and SSVM are in the order of a couple of hours. Note from the results in table 2 that, when SSVM is used, for most datasets we can expect accurate results with less than 1000 features. Therefore SSVM is very affordable. RFE is a much faster method since it requires only a few standard SVM trainings and, even when all the features are present, standard SVM training can be done very fast [16]. In general, RFE turns out to be an order of magnitude faster than SSVM and L_1-SVM. Thus, if training time happens to be a bottleneck for SSVM, RFE could be the method of choice given its relatively good results.

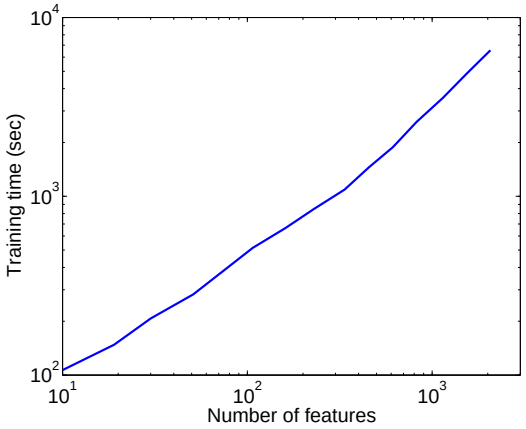

Figure 3: Total training time for L_1-SVM as a function of the number of features on OHSUMED. The relation is roughly linear. SSVM requires about the same time as L_1-SVM since it uses the feature set found by L_1-SVM and the final SVM training step is in comparison very fast. The total training time for RFE is 802 seconds.

4. Discussion

The main idea of this paper concerns the suitable setting up of the regularization part of the training objective to achieve simultaneous multi-class feature selection. Although in (2) we used a sum of individual one-versus-rest loss functions, that is not restrictive; we could as well employ other loss functions that are suited for a given situation. For instance, for purely multi-class problems we can use a joint loss function involving the weight vectors of all classes together. For the multi-labeled setting more sophisticated loss functions based on ranking such as the one in [6] can be used in association with our regularization methods.

The parallel works of Obozinski et al and Argyriou et al [24, 1] use the regularizer in (4), but with the motivation of using it for multi-task learning. Block coordinate optimization combined with path tracking is employed in [24] for training. While these methods are purely tied to L_1 regularization similar to our L_1-SVM method, our SSVM method is an interesting idea that uses L_1 regularization as a feature selection method for L_2-SVM solution; the idea of scaling factors makes this connection neatly via the developments in (2)-(3). The performance difference between L_1-SVM and L_2-SVM (using all features) depends on the dataset. For instance, on 20NG L_2-SVM is clearly superior, while on OHSUMED L_1-SVM is quite better. Therefore, from a practical perspective SSVM offers a very useful addition because it achieves the performance of L_2-SVM, but with a much smaller number of features. L_1-SVM and SSVM together (with one or the other chosen by validation) provide a powerful overall feature selection method.

5. Conclusion

In this paper we have proposed a new class of embedded methods for simultaneous feature selection in multi-class and multi-labeled text classification. The methods do effective feature selection: on some datasets, the feature set size was an order of magniture smaller than the one selected by Information Gain (for the same performance level). They are also computationally efficient and the structure of the optimization method allows ample scope for enhancing efficiency using parallel processing techniques.

Finally it is noteworthy that the framework of scaling factors that was used for deriving the new embedded methods is general and new algorithms for feature selection can easily be derived from it.

References

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, Cambridge, MA, 2007.

[2] J. Bi, K. Bennett, M. Embrechts, and C. Breneman. Dimensionality reduction via sparse support vector machine. *Journal of Machine Learning Research*, 3, 2003.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Availabe at `http://www.stanford.edu/~boyd/cvxbook`.

[5] X. Chen, X. Zeng, and D. van Alphen. Multi-class feature selection for texture classification. *Pattern Recognition Letters*, 27(14):1685–1691, 2006.

[6] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

[7] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 1987.

[8] G. Forman. A pitfall and solution in multi-class feature selection for text classification. In *International Conference on Machine Learning*, 2004.

[9] A. Genkin, D. D. Lewis, and D. Madigan. Large scale bayesian logistic regression for text categorization. *Technometrics*, 49:291–304, 2003.

[10] Y. Grandvalet and S. Canu. Adaptive Scaling for Feature Selection in SVMs. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[12] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1/3):389, 2002.

[13] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, pages 1391–1415, 2004.

[14] D. R. Hunter and K. Lange. A tutorial on mm algorithms. *The American Statistician*, 58:30–37, 2004.

[15] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, number 1398 in Lectures Notes in Artificial Intelligence, pages 137–142. Springer Verlag, 1998.

[16] S. S. Keerthi and D. M. DeCoste. A modified finite Newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, 2005.

[17] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*, pages 137–165. Springer-Verlag, 2006.

[18] K. Lange. A gradient algorithm locally equivalent to the em algorithm. *Journal of the Royal Statistical Society, Series B*, 57:425–437, 1995.

[19] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[20] D. Madigan, A. Genkin, D. D. Lewis, S. Argamon, D. Fradkin, and L. Ye. Author identification on the large scale. In *Classification Society of North America*, 2005.

[21] O. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

[22] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[23] L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. Technical Report TR131, ETH Zürich, 2006.

[24] G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. Technical report, Department of Statistics, University of California, Berkeley, 2006.

[25] Y. Yang. A study on thresholding strategies for text categorization. In *Proceedings of SIGIR*, pages 137–145, 2001.

[26] Y. Yang and J. Pedersen. A comparitive study on feature selection in text categorization. In *International Conference on Machine Learning*, 1997.

[27] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[28] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems*, 2003.

Appendix

The optimization of (3) has been implemented as an interior point method: a log barrier on the scaling factors has been added to the objective: $-t\sum\log(\sigma_j)$. Depending on the value of t, this will force more or less the σ to be away from 0. We will describe later how to minimize this modified objective function for a given value of t. Usually [4], t is initialized to a large value, the new unconstrained objective function is minimized, then t is decreased and this process iterates until t has reached a sufficiently small value.

We could have done exactly this procedure, but given that the optimal solution is likely to be sparse, some time is wasted on optimizing weights and scaling features which will zero at the end anyway. Instead, we implemented a path tracking approach [13]: $\tilde{C}$ is increased from a small value to its final value. Given the solution and the set of active features for a given $\tilde{C}$, the new solution for a larger $\tilde{C}$ can be found efficiently because the set of active features is likely not to change too much. Pseudo-code is given in algorithm 1. During this process, there is no need to vary t since the gradual change in $\tilde{C}$ already makes the optimization well behaved. Thus t is fixed to a relatively small value, $t = 10^{-3}$.

Two important ingredients in this algorithm are the criterion to add and remove features. If t were 0, one could simply remove the features corresponding to $\sigma_j = 0$. But because of the log barrier, all the σ_j will be at least equal to $2t$. That is the reason why we decided to have a relative threshold with respect to t and remove the features such that $\sigma_j < 10t$.

On the other hand, adding features is based on the link with L_1-SVM (4). One can indeed show that adding an inactive feature j will decrease the objective function if and only if

$$\sum_k \left(\frac{\tilde{C}}{2} \frac{\partial}{\partial w_{jk}} \sum_{i=1}^{n} \ell(y_{ik}(\mathbf{w}_k \cdot \mathbf{x}_i)) \right)^2 > 1.$$

Let us now the study the unconstrained minimization of (3) with the log barrier. To do so we used the the Levenberg-Marquardt algorithm[4], which is basically a Newton-type algorithm where a ridge is added to the Hessian in order to limit the step size. The expensive part of this algorithm is to solve the linear system $(H+\lambda I)x = g$, where H is the Hessian of the objective function and g its gradient. To do so, we used a *linear conjugate gradient* algorithm with a maximum of 20 iterations.

The Hessian wrt to $(w_{11}, w_{21}, \ldots, w_{dc}, \sigma_1, \ldots, \sigma_d)$ is

$$\begin{pmatrix} \ddots & & 0 & \vdots \\ & \tilde{C}X_{sv_k}^\top X_{sv_k} + D^{-1}(\sigma) & & D(\mathbf{w}_k)D^{-2}(\sigma) \\ 0 & & \ddots & \vdots \\ \ldots & D(\mathbf{w}_k)D^{-2}(\sigma) & \ldots & D(\sum_k w_{jk}^2 + t\sigma_j)D^{-3}(\sigma) \end{pmatrix},$$

where X_{sv_k} is the matrix containing the support vectors of the k-th classifier and $D(v)$ stands for the diagonal matrix with vector v on the diagonal. One can see that most blocks are diagonals and that the bulk of the calculation in the Hessian vector product are multiplications by either X or $X^\top$. The sparsity of those matrices is the key element for our algorithm to be fast.

Finally, we used a preconditioner which is equal to the Hessian, but where $X^\top X$ is replaced by its diagonal elements. Thus the preconditioner is diagonal in each block. In preconditioned conjugate gradient, one needs to invert the preconditioner efficiently. This is the case here because the blocks are only on the diagonal and last row and column.

[4] We could also have used nonlinear conjugate gradient. [7, Algorithm 5.2.7]