

Effective Shrinkage of Large Multi-class Linear SVM Models for Text Categorization

Jianxiong Dong
Yahoo Inc
jxd@yahoo-inc.com

C. Y. Suen, Adam Krzyżak
CENPARMI, Concordia University
suen@cenparmi.concordia.ca, krzyzak@cs.concordia.ca

Abstract

When linear support vector machines (SVMs) are applied to multi-class text categorization in industry, the size of the linear SVM model is very large, usually greater than several gigabytes. As a result, the model cannot directly fit into the computer memory and the classification process is slow. In this paper, a novel method based on vector norm is proposed to shrink the model size significantly without sacrificing the classification accuracy. Also, we propose a cache-efficient implementation of multi-class linear SVMs in the classification phase. Our experimental results have shown that on Yahoo-Korea dataset the proposed method can shrink the model size from 5.2 gigabytes to 260 megabytes and the efficient implementation of linear SVM has obtained a speedup factor of 44.

1. Introduction

Linear support vector machines [1, 2] have been active research topics in text categorization in the past few years. Compared with other classifiers in text categorization, linear SVMs have achieved competitive generalization performance [1]. Most research focuses on the efficient algorithms of training linear SVMs [2] on a large dataset. The issues of linear SVMs in the testing phase are seldom addressed. When linear SVMs are used in the classification of web contents for many categories and bag-of-words are used as features, the SVM model size is usually linearly proportional to the number of categories and the total size of index terms. In the web content classification, the size of index terms usually amounts to several millions. As a result, the model size of linear SVMs for multi categories is large (several gigabytes) and cannot fit into the computer memory. Therefore, it is important to reduce the size of models without sacrificing the accuracy so that linear SVMs

can be widely applied to large-scale text categorization in real world.

Several strategies can be used to shrink the size of the model. One method is to perform feature selection [6] before linear SVMs are trained. However, most feature selections in text categorization do not directly minimize the classification errors. An aggressive feature selection may result in the performance degradation of linear SVMs since it can handle very large feature sets and is not sensitive to the problem of “curse of dimensionality” [5] which occurs on the conventional methods such as decision tree and nearest neighboring. In [1] the experimental results showed that the low-ranked features by information gains still contain useful information and are relevant to some extent. The other method is to select features based on SVM’s weights. The term ranking scores are represented in the SVM weights. In [3], for linear binary SVM classification, the corresponding SVM’s weight magnitude of each feature is used for feature ranking. In [4], the authors combined the weight magnitude and the mean difference of each feature in positive and negative classes for feature ranking.

In this paper, we propose a novel method based on vector norm constructed in the parameter space of multi-class linear SVMs to rank the term. Based on term ranking, we can significantly shrink the model while maintaining the accuracy. Also, we propose a cache-efficient implementation of multi-class linear SVMs based on shrunk model. On the Yahoo-Korea dataset, a speedup factor of 44 has been achieved.

The rest of the paper is organized as follows. In Section 2 we formally describe the problem and construct the vector norm based on parameter space which is used to shrink the model. Section 3 a cache-efficient algorithm is proposed to implement linear SVMs in the testing phase. In section 4 extensive experimental results have shown the effectiveness of the proposed methods and we also compare the performance of different implementations of multi-class linear SVMs. Section 5

summarizes our findings.

2. Model shrinkage based on term ranking in parameteric space

In text categorization, the original feature space consists of the unique terms that occur in documents. The algorithm in [2] which uses L2-norm loss function is applied to train linear SVMs. The algorithm performs optimization on the primal space, rather than the dual space. As a result, for each binary SVM, the number of free parameters is equal to the total size of unique terms plus one (a bias term). For multi-classes, one-against-others method is used. After the training, each linear SVM becomes a hyperplane, represented by

$$f_i = \sum_{i=1}^n w_{ij}x_i + b_i \quad (1)$$

where $i = 1, \dots, m$, m is the number of classes. b_i is the bias term for class i and n is the total size of unique terms of the training set. w_{ij} is the weight for term j in binary linear SVM for class i . Intuitively, the larger magnitude the weight is, the more contribution it makes to the decision. For binary SVM, the L2 norm of weight vector in (1) is related to the margin of SVM. The larger the margin is, the better generalization SVM can achieve to some extent.

It can also be rewritten in the matrix form below:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1n} \\ w_{21} & \dots & w_{2n} \\ \vdots & \vdots & \vdots \\ w_{m1} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \quad (2)$$

Let \mathbf{W} denote the weight matrix on the right side in eq. (2) and $\mathbf{W}(:, j)$ denote the j th column vector associated with term j . For the matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, the induced norm is defined by

$$\|\mathbf{W}\|_p = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{W}\mathbf{x}\|_p}{\|\mathbf{x}\|_p} \quad (3)$$

where $\|\cdot\|_p$ corresponds to p -norm of a vector. From the above definition, the following equations [7] hold

$$\begin{aligned} \|\mathbf{W}\mathbf{x}\|_2 &\leq \|\mathbf{W}\|_2 \|\mathbf{x}\|_2 \\ \frac{1}{\sqrt{m}} \|\mathbf{W}\|_1 &\leq \|\mathbf{W}\|_2 \leq \sqrt{n} \|\mathbf{W}\|_1 \\ \|\mathbf{W}\|_1 &= \max_{1 \leq j \leq n} \sum_{i=1}^m |w_{ij}| \end{aligned} \quad (4)$$

It can be inferred from the above equations that the L2 matrix norm should not be significantly changed if

the matrix \mathbf{W} is shrunk. When \mathbf{W} is large, it is difficult to estimate the change of its norm. But L2 matrix norm can be bounded by L1 matrix norm. L1 matrix norm provides us a good hint: remove some columns with small norms. Intuitively, we rank the column vectors of \mathbf{W} based on its L2-norm and remove low-rank column vectors. The reason of choosing L2 norm is that the SVM objective function minimizes the L2-norm of the weight vector under some constraints [2]. Details of the algorithm are given below:

Multi-class SVM Model shrinkage based on L2-norm

Input: Weight matrix W and a shrinking ratio t which controls the size of remaining column vectors.

Output: A sub-matrix of size $m \times k$, where k is set to nt .

1. Calculate the L2-norm of column vectors of \mathbf{W} .
2. Sort vector norms in a descending order.
3. Select the top k column vectors to construct a sub-matrix.

The computational cost of the above algorithms is very low. We can heuristically interpret the vector norm of $\mathbf{W}(:, j)$ as the decision strength of a term j .

3. A good implementation of multi-class linear SVMs

In text categorization, the input feature vector x in eq. (2) is very sparse. Although the feature dimension is very high (more than several millions), the average number of unique terms in a web document is low, usually between 200 and 300. The matrix \mathbf{W} could be dense or sparse. If we calculate the dot product of class weight vector and input feature vector one by one, the weight vector will not be accessed contiguously in memory address space. As a result, the ratio of cache misses will be high. In fact, eq.(2) can be calculated in the form of

$$\mathbf{f} = \sum_i^n x_i \mathbf{W}(:, j) \quad (5)$$

That is, the decision vector \mathbf{f} can be treated as linear combination of column vectors of matrix \mathbf{W} . In case of row-major matrix memory layout such as C language, if \mathbf{W} is transposed and stored in the memory, the column vectors can be accessed in a contiguous address space and the number of cache misses can be reduced significantly.

4. Experimental results

In this section, we evaluated the shrinking performance of different p -norms described in Section 2 and compared the proposed norm-based shrinking method with the traditional uniform thresholding method. We also compare the performance of different implementations of linear SVMs in the testing phase.

In our experiments, Yahoo-Korea dataset, collected at Yahoo, consists of 460554 training web documents with 450 categories. The size of documents on the testing set is 1000. The total size of term indexes is 3,052,939. For the feature extraction, a binary term vector was created which indicates whether each term appears on the page or not, and was normalized into the unit length. The term frequency and document frequency are not taken into account in this representation since the early experiments in [8] showed good performance in this representation.

We train each binary SVM for the corresponding class based on one-against-others method using the algorithm in [2]. After the training, a sigmoid function is used to fit the outputs of each binary SVM using the method in [9] so that binary SVMs generate comparable probabilistic outputs across categories. The probabilistic outputs help to define the uniform confidence and combine the different classification methods.

The experiments were conducted on a Linux server which ran RHEL4 with AMD Opeteron 852 and 64 gigabyte RAM.

4.1. Performance comparisons for different p -norms

After the training, the weight matrix W of size 450×3052939 takes up 5200 megabytes. When the proposed shrinking method is applied, the results are shown in Table 1, where t is defined in the algorithm in Section 2. It can be observed from Table 1 that L2 norm performs best when the size of shrunk matrix becomes smaller and max norm (that is, $p = \infty$) performs worst. When t is 1.00, we can see that without shrinking the original training accuracy is 85.72%. When t is 0.05 (that is, removing 95% column vectors), the accuracy for the shrunk matrix for L2 norm is 84.35%. Compared with the original one, the accuracy is slightly degraded.

4.2. Performance comparisons with uniform thresholding method

Traditionally, in order to shrink the matrix, we like to convert the dense weight matrix into a sparse one by a cut-off value: if the absolute value of a weight element

Table 1. Training accuracy (%) for different p -norms vs. shrinking ratio t

| Ratio t | L1 norm | L2 norm | max norm |
|-----------|---------|---------|----------|
| 1.00 | 85.72 | 85.72 | 85.72 |
| 0.75 | 85.71 | 85.71 | 85.71 |
| 0.50 | 85.64 | 85.65 | 85.64 |
| 0.25 | 85.42 | 85.43 | 85.39 |
| 0.15 | 85.17 | 85.20 | 85.08 |
| 0.10 | 84.92 | 84.95* | 84.75 |
| 0.05 | 84.19 | 84.35 | 83.95 |
| 0.025 | 82.42 | 83.32* | 82.53 |
| 0.01 | 79.18 | 80.61* | 78.02 |
| 0.005 | 74.27 | 77.18* | 68.79 |
| 0.003 | 68.14 | 74.08* | 63.34 |
| 0.001 | 40.28 | 61.93* | 55.91 |

is less than a threshold β , the element will be set to zero. In this section, its performance is compared with the proposed method based on L2 norm and the results are shown in

Table 2. Training accuracy (%) based on the uniform thresholding method

| Threshold (β) | Shrinking ratio t | Accuracy (%) |
|-----------------------|---------------------|--------------|
| 0.01 | 0.795 | 85.61 |
| 0.05 | 0.381 | 84.35 |
| 0.07 | 0.255 | 83.32 |
| 0.1 | 0.147 | 81.52 |
| 0.185 | 0.05 | 76.71 |
| 0.2 | 0.044 | 75.86 |

For the same shrinking ratio t of size 0.05, our proposed shrinking method based on L2 norm obtains a training accuracy of 84.35% whereas the accuracy for uniform thresholding method is only 76.71%.

4.3. How the proposed shrinking method affects classifier confidence and testing accuracy

For the proposed shrinking method, it is necessary to investigate how it affects the classifier confidence and testing accuracy. Since our classifier outputs are transformed as probabilistic ones [9], we use the average absolute difference of those outputs on the training and testing sets to measure the confidence change.

It can be observed that when the shrinking ratio is 0.05, there are no significant differences of classifier outputs and the performance of the proposed method is almost the same as the original one.

Table 3. Confidence differences and testing accuracy for L2-norm shrinking

| Shrinking ratio t | 0.05 | 0.01 | 0.003 | 0.001 |
|----------------------|--------|-------|-------|-------|
| Train_conf_diff | 0.026 | 0.1 | 0.236 | 0.42 |
| Test_conf_diff | 0.0077 | 0.056 | 0.16 | 0.298 |
| Testing accuracy (%) | 60 | 59 | 55 | 45 |

4.4. Performance comparisons of different implementations

In this subsection, we compare the performance of several implementations of linear SVMs. When the shrinking ratio is set to 0.05, we obtained a dense shrunk weight matrix of size 450×152637 . If each weight is represented as a float point format (four bytes), the size of the total memory usage is about 260 megabytes while the original one is about 5240 megabytes. (In case of sparse-matrix vector multiplication, we convert the dense matrix into the sparse one based on uniform thresholding method. The threshold is set to 0.001. As a result, the number of non-zero weights is 55182519. The results show that for the shrunk dense matrix, about 80 ($=55182519/(450 * 152637)$) percent of the weights are nonzero. In this case the data layout of the traditional compressed row storage for sparse matrix does not have any advantage over the dense one since this representation takes up more memory than the dense one. For the above three implementations, the performances are shown below:

Table 4. Performance comparisons of different implementations

| Methods | Speed (examples per second) |
|-----------------|-----------------------------|
| Method_A | 50 |
| Method_B | 250 |
| Proposed method | 2204 |

where Method_A: Sparse-matrix vector multiplication; Method_B: Cache-inefficient dense-matrix vector multiplication. The above results have shown that the data layout and cache-efficient access patterns can boost the performance dramatically.

5. Conclusions

This paper presents a new method based on L2-norm defined on parameter space to effectively shrink the large SVM model for web document categoriza-

tion. The method can significantly reduce the memory usage while keeping the accuracy. The experimental results have shown that L2-norm shrinking performs best among several vector norms and cache-efficient implementation increases the classification speed substantially.

References

- [1] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the European Conference On Machine Learning*, no. 1398, in Lecture Notes in Computer Science, pp. 137–142, Chermnitz, DE. Springer Verlag, Heidelberg, DE 1998.
- [2] S. S. Keerthi and D. DeCoste, “A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs,” *The Journal of Machine Learning Research*, vol. 6, pp. 341–361, December 2005.
- [3] I. Guyon, J. Weston, S. Barnhill and V. Vapnik, “Gene Selection for Cancer Classification using Support Vector Machines,” *Machine Learning*, vol. 46, pp. 389–422, 2002.
- [4] Z.G. Fan and B.L. Lu, “Fast Recognition of Multi-view Faces with Feature Selection,” *Proceedings of 10th IEEE International Conference on Computer Vision*, vol. 1, pp.76-81, 2005.
- [5] R.O. Duda, P.E. Hart and D.G. Stork, *Pattern Classification*. 2nd Edition. Wiley-interscience, October 2000.
- [6] Y. Yang and J. Pedersen, “A comparative study on feature selection in text categorization,” In J.D. H. Fisher, editor. *The Fourteenth International Conference on Machine Learning (ICML’97)*, pp. 412–420, Morgan Kaufmann, 1997.
- [7] G. Golub and Charles F. Van Loan. *Matrix Computations - Third Edition*. The Johns Hopkins University Press, Baltimore, 1996.
- [8] S.T. Dumais, J. Platt, D. Heckerman and M. Sahami, “Inductive learning algorithms and representations for text categorization,” *Proceedings of the 7th International Conference on Information and Knowledge Management*, pp. 148–155, 1998.
- [9] H.-T. Lin, C.-J. Lin and R.C. Weng, A note on Platt’s probabilistic outputs for support vector machines. *Machine Learning*, vol. 28, pp. 267–276, 2007.