Report No. UIUCDCS-R-2006-2714 **UILU-ENG-2006-1746**

Support Tensor Machines for Text Categorization

by

Deng Cai, Xiaofei He, Ji-Rong Wen, Jiawei Han and Wei-Ying Ma

April 2006

Support Tensor Machines for Text Categorization*

Deng Cai† Xiaofei He‡ Ji-Rong Wen* Jiawei Han† Wei-Ying Ma*

† Department of Computer Science, University of Illinois at Urbana-Champaign

* Microsoft Research Asia

‡ Yahoo! Research Labs

Abstract

We consider the problem of text representation and categorization. Conventionally, a text document is represented by a vector in high dimensional space. Some learning algorithms are then applied in such a vector space for text categorization. Particularly, Support Vector Machine (SVM) has received a lot of attentions due to its effectiveness. In this paper, we propose a new classification algorithm called **Support Tensor Machine** (STM). STM uses **Tensor Space Model** to represent documents. It considers a document as the second order tensor in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$, where $\mathcal{R}^{n_1}$ and $\mathcal{R}^{n_2}$ are two vector spaces. With tensor representation, the number of parameters estimated by STM is much less than the number of parameters estimated by SVM. Therefore, our algorithm is especially suitable for small sample cases. We compared our proposed algorithm with SVM for text categorization on two standard databases. Experimental results show the effectiveness of our algorithm.

1 INTRODUCTION

Text categorization is a task of assigning category labels to new documents based on a set of pre-labelled documents. It is usually considered as a supervised or semi-supervised learning problem. During the last decade, a lot of learning algorithms have been proposed for text categorization, such as Support Vector Machines (SVM) [7], naïve bayes [9], k-nearest neighbors and linear least square fit [19]. Most of these works are based on the Vector Space Model (VSM, [13]). The documents are represented as vectors, and each word corresponds to a dimension. The main reason of the popularity of VSM is probably due to the fact that most of the existing learning algorithms can only take vectors as their inputs, rather than tensors.

* The work was supported in part by the U.S. National Science Foundation NSF IIS-03-08215/IIS-05-13678. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Support Vector Machines (SVM) is a relatively new learning approach introduced by Vapnik in 1995 for solving two-class pattern recognition problem [15]. It is based on the Structural Risk Minimization principle for which error-bound analysis has been theoretically motivated. The method is defined over a vector space where the problem is to find a decision surface that maximizes the margin between the data points in a training set. Previous investigations have demonstrate that SVM can be superior to other learning algorithms such as naïve bayes [18].

In supervised learning settings with many input features, overfitting is usually a potential problem unless there is ample training data. For example, it is well known that for unregularized discriminative models fit via training-error minimization, sample complexity (i.e., the number of training examples needed to learn "well") grows linearly with the Vapnik-Chernovenkis (VC) dimension. Further, the VC dimension for most models grows about linearly in the number of parameters [14], which typically grows at least linearly in the number of input features. All these reasons lead us to consider new representations and corresponding learning algorithms with less number of parameters.

In this paper, we propose a novel supervised learning algorithm for text categorization, which is called **Support Tensor Machine** (STM). Different from most of previous text categorization algorithms which consider a document as a vector in $\mathbb{R}^n$ based on the vector space model, STM considers a document as a second order tensor in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$, where $n_1 \times n_2 \approx n$. For example, a vector $\mathbf{x} \in \mathbb{R}^n$ can be transformed by some means to a second order tensor $\mathbf{X} \in \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. A linear classifier in $\mathbb{R}^n$ can be represented as $\mathbf{a}^T\mathbf{x} + b$ in which there are $n + 1$ ($\approx n_1 \times n_2 + 1$) parameters $(b, a_i, i = 1, \cdots, n)$. Similarly, a linear classifier in the tensor space $\mathcal{R}^n \otimes \mathcal{R}^n$ can be represented as $\mathbf{u}^T\mathbf{X}\mathbf{v} + b$ where $\mathbf{u} \in \mathcal{R}^{n_1}$ and $\mathbf{v} \in \mathcal{R}^{n_2}$. Thus, there are only $n_1 + n_2 + 1$ parameters. This property makes STM especially suitable for small sample cases.

Recently there has been a lot of interests in tensor based approaches to data analysis in high dimensional spaces. Vasilescu and Terzopoulos have proposed a novel face representation algroithm called Tensorface [16]. Tensorface represents the set of face images by a higher-order tensor and extends Singular Value Decomposition (SVD) to higher-order tensor data. Some other researchers have also shown how to extend Principal Component Analysis, Linear Discriminant Analysis, and Locality Preserving Projection to higher order tensor data [2],[6],[20]. Most of previous tensor based learning algorithms are focused on dimensionality reduction. In this paper, we extend SVM based idea to tensor data for classification.

It is worthwhile to highlight several aspects of the proposed approach here:

- While traditional linear classification algorithms like SVM find a classifier in $\mathbb{R}^n$, STM finds a classifier in tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. This leads to structured classification.
- The computation of STM is very simple. It can be obtained by solving two quadratic optimization problems. For each single optimization problem, the computational complexity approximately scales with $\sqrt{n}$, where n is the dimensionality of the document space. There

are few parameters that are independently estimated, so performance in small data sets is very good.

- This paper is primarily focused on the second order tensors. However, the algorithm and the analysis presented here can also be applied to higher order tensors.

The rest of this paper is organized as follows: Section 2 provides a brief description of the algebra of tensors and SVM. In Section 3, we give some descriptions on tensor space model for document representation. The Support Tensor Machine (STM) approach for text categorization is described in Section 4. In Section 5, we give a theoretical justification of STM and its connections to SVM. The experimental results on text databases are presented in Section 6. Finally, we provide some concluding remarks and suggestions for future work in Section 7.

2 PRELIMINARY

In this section, we provide a brief overview of the algebra of tensors and Support Vector Machine. For a detailed treatment please see [8], [15].

2.1 The Algebra of Tensors

A tensor with order k is a real-valued multilinear function on k vector spaces:

$$T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} \to \mathbb{R}$$

The number k is called the *order* of T. A multilinear function is linear as a function of each variable separately. The set of all k-tensors on $\mathbb{R}^{n_i}, i = 1, \cdots, k$, denoted by $\mathcal{T}^k$, is a vector space under the usual operations of pointwise addition and scalar multiplication:

$$(aT)(\mathbf{a}_1, \cdots, \mathbf{a}_k) = a\left(T(\mathbf{a}_1, \cdots, \mathbf{a}_k)\right),$$

$$(T + T')(\mathbf{a}_1, \cdots, \mathbf{a}_k) = T(\mathbf{a}_1, \cdots, \mathbf{a}_k) + T'(\mathbf{a}_1, \cdots, \mathbf{a}_k)$$

where $\mathbf{a}_i \in \mathbb{R}^{n_i}$. Given two tensors $S \in \mathcal{T}^k$ and $T \in \mathcal{T}^l$, define a map:

$$S \otimes T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_{k+l}} \to \mathbb{R}$$

by

$$S \otimes T(\mathbf{a}_1, \cdots, \mathbf{a}_{k+l}) = S(\mathbf{a}_1, \cdots, \mathbf{a}_k)T(\mathbf{a}_{k+1}, \cdots, \mathbf{a}_{k+l})$$

It is immediate from the multilinearity of S and T that $S \otimes T$ depends linearly on each argument $\mathbf{a}_i$ separately, so it is a $(k+l)$-*tensor*, called the tensor product of S and T.

For the first order tensors, they are simply the covectors on $\mathbb{R}^{n_1}$. That is, $\mathcal{T}^1 = \mathcal{R}^{n_1}$, where $\mathcal{R}^{n_1}$ is the dual space of $\mathbb{R}^{n_1}$. The second order tensor space is a product of two first order tensor spaces,

i.e. $\mathcal{T}^2 = \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Let $\mathbf{e}_1, \cdots, \mathbf{e}_{n_1}$ be the standard basis of $\mathbb{R}^{n_1}$, and $\varepsilon_1, \cdots, \varepsilon_{n_1}$ be the dual basis of $\mathcal{R}^{n_1}$ which is formed by coordinate functions with respect to the basis of $\mathbb{R}^{n_1}$. Likewise, let $\widetilde{\mathbf{e}}_1, \cdots, \widetilde{\mathbf{e}}_{n_2}$ be a basis of $\mathbb{R}^{n_2}$, and $\widetilde{\varepsilon}_1, \cdots, \widetilde{\varepsilon}_{n_2}$ be the dual basis of $\mathcal{R}^{n_2}$. We have,

$$\varepsilon_i(\mathbf{e}_j) = \delta_{ij} \text{ and } \widetilde{\varepsilon}_i(\widetilde{\mathbf{e}}_j) = \delta_{ij}$$

where δ_{ij} is the kronecker delta function. Thus, $\{\varepsilon_i \otimes \widetilde{\varepsilon}_j\}$ $(1 \leq i \leq n_1,\ 1 \leq j \leq n_2)$ forms a basis of $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. For any 2-tensor T, we can write it as:

$$T = \sum_{\substack{1 \leq i \leq n_1 \\ 1 \leq j \leq n_2}} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j$$

This shows that every 2-tensor in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ uniquely corresponds to a $n_1 \times n_2$ matrix. Given two vectors $\mathbf{a} = \sum_{k=1}^{n_1} a_k \mathbf{e}_k \in \mathbb{R}^{n_1}$ and $\mathbf{b} = \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l \in \mathbb{R}^{n_2}$, we have

$$\begin{aligned} T(\mathbf{a}, \mathbf{b}) &= \sum_{ij} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j (\sum_{k=1}^{n_1} a_k \mathbf{e}_k, \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\ &= \sum_{ij} T_{ij} \varepsilon_i (\sum_{k=1}^{n_1} a_k \mathbf{e}_k) \widetilde{\varepsilon}_j (\sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\ &= \sum_{ij} T_{ij} a_i b_j \\ &= \mathbf{a}^T T \mathbf{b} \end{aligned}$$

Note that, in this paper our primary interest is focused on the second order tensors. However, the algebra presented here and the algorithm presented in the next section can also be applied to higher order tensors.

2.2 Support Vector Machine

Support Vector Machines are a family of pattern classification algorithms developed by Vapnik [15] and collaborators. SVM training algorithms are based on the idea of *structural risk minimization* rather than *empirical risk minimization*, and give rise to new ways of training polynomial, neural network, and radial basis function (RBF) classifiers. SVM has proven to be effective for many classification tasks [7][12].

We shall consider SVMs in the binary classification setting. Assume that we have a data set $D = \{\mathbf{x}_i, y_i\}_{i=1}^m$ of labeled examples, where $y_i \in \{-1, 1\}$, and we wish to select, among the infinite number of linear classifiers that separate the data, one that minimizes the generalization error, or at least minimizes an upper bound on it. It is shown that the hyperplane with this property is the one that leaves the maximum margin between the two classes. The discriminant hyperplane can be defined as

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \tag{1}$$

where $\mathbf{w}$ is a vector orthogonal to the hyperplane. Computing the best hyperplane is posed as a constrained optimization problem and solved using quadratic programming techniques. The optimization problem of SVM can be stated as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{2}$$

Given a new data point $\mathbf{x}$ to classify, a label is assigned according to its relationship to the decision boundary, and the corresponding decision function is

$$g(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{3}$$

3 TENSOR SPACE MODEL

Document indexing and representation has been a fundamental problem in information retrieval for many years. Most of previous works are based on the Vector Space Model (VSM, [13]). The documents are represented as vectors, and each word corresponds to a dimension. In this section, we introduce a new Tensor Space Model (TSM) for document representation.

In Tensor Space Model, a document is represented as a tensor. Each element in the tensor corresponds to a feature (word in our case). For a document $\mathbf{x} \in \mathbb{R}^n$, we can convert it to the second order tensor (or matrix) $X \in \mathbb{R}^{n_1 \times n_2}$, where $n_1 \times n_2 \approx n$. Figure 1 shows an example of converting a vector to a tensor. There are two issues about converting a vector to a tensor.

The first one is how to choose the size of the tensor, i.e., how to select n_1 and n_2. In figure 1, we present two possible tensors for a 9-dimensional vector. Suppose $n_1 \geq n_2$, in order to have at least n entries in the tensor while minimizing the size of the tensor, we have $(n_1 - 1) \times n_2 < n \leq n_1 \times n_2$. With such requirement, there are still many choices of n_1 and n_2, especially when n is large. Generally all these (n_1, n_2) combinations can be used. However, it is worth noticing that the number of parameters of a linear function in the tensor space is $n_1 + n_2$. Therefore, one may try to minimize $n_1 + n_2$. In other words, n_1 and n_2 should be as close as possible.

The second issue is how to sort the features in the tensor. In vector space model, we implicitly assume that the features are independent. A linear function in vector space can be written as $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. Clearly, the change of the order of the features has no impact on the function learning. In tensor space model, a linear function can be written as $f(X) = \mathbf{u}^T X \mathbf{v}$. Thus, the independency assumption of the features no longer holds for the learning algorithms in the tensor space model. Different feature sorting will lead to different learning result in the tensor space model.

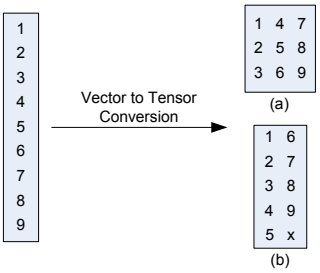

Figure 1: Vector to tensor conversion. 1∼9 denote the positions in the vector and tensor formats. (a) and (b) are two possible tensors. The 'x' in tensor (b) is a padding constant.

In this paper, we empirically sort the features (words) according to their document frequency and then convert the vector into a $n_1 \times n_2$ tensor such that $n_2 = 50$. The better ways of converting a document vector to a document tensor with theoretical guarantee will be left for our future work.

4 SUPPORT TENSOR MACHINES

4.1 The Problem

Given a set of training samples $\{\mathbf{X}_i, y_i\}, i = 1, \cdots, m$, where $\mathbf{X}_i$ is the data point in order-2 tensor space, $\mathbf{X}_i \in \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ and $y_i \in \{-1, 1\}$ is the label associated with $\mathbf{X}_i$. Find a tensor classifier $f(\mathbf{X}) = \mathbf{u}^T \mathbf{X} \mathbf{v} + b$ such that the two classes can be separated with maximum margin.

4.2 The Algorithm

STM is a tensor generalization of SVM. The algorithmic procedure is formally stated below:

1. **Initialization:** Let $\mathbf{u} = (1, \cdots, 1)^T$.
2. **Computing v:** Let $\mathbf{x}_i = \mathbf{X}_i^T \mathbf{u}$ and $\beta_1 = \|\mathbf{u}\|^2$, $\mathbf{v}$ can be computed by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{v},b,\xi} \quad & \frac{1}{2}\beta_1 \mathbf{v}^T \mathbf{v} + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{v}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{4}$$

   Note: The optimization problem (4) is the same as (2) in the standard SVM algorithm. Thus, any computational method for SVM can also be used here.

3. **Computing u:** Once $\mathbf{v}$ is obtained, let $\tilde{\mathbf{x}}_i = X_i\mathbf{v}$ and $\beta_2 = \|\mathbf{v}\|^2$. $\mathbf{u}$ can be computed by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u},b,\xi} \quad & \frac{1}{2}\beta_2\mathbf{u}^T\mathbf{u} + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{u}^T\tilde{\mathbf{x}}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{5}$$

Note: As above, the optimization problem (5) is the same as (2) in the standard SVM algorithm and we can use the computational methods for SVM to solve (5).

4. **Iteratively computing u and v:** By step 2 and 3, we can iteratively compute $\mathbf{u}$ and $\mathbf{v}$ until they tend to converge.
Note: The convergence proof is given in Section 5.2.

5 JUSTIFICATIONS

In this section, we provide a justification of our STM algorithm.

5.1 Large Margin Classifier in Tensor Space

Suppose we have a set of order-2 tensors $\mathbf{X}_1, \cdots, \mathbf{X}_m \in \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. A linear classifier in the tensor space can be naturally represented as follows:

$$f(\mathbf{X}) = sign\left(\mathbf{u}^T\mathbf{X}\mathbf{v} + b\right), \quad \mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2} \tag{6}$$

Equation (6) can be rewritten through matrix inner product as follows:

$$f(\mathbf{X}) = sign\left(< \mathbf{X}, \mathbf{u}\mathbf{v}^T > +b\right), \quad \mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2} \tag{7}$$

Thus, the optimization problem in the tensor space is reduced to the following:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v},b,\xi} \quad & \frac{1}{2}\|\mathbf{u}\mathbf{v}^T\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{u}^T\mathbf{X}_i\mathbf{v} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{8}$$

We will now switch to a Lagrangian formulation of the problem. We introduce positive Lagrange multipliers $\alpha_i, \mu_i, i = 1, \cdots, m$, one for each of the inequality constraints (8). This gives Lagrangian:

$$\begin{aligned} L_P \quad = \quad & \frac{1}{2}\|\mathbf{u}\mathbf{v}^T\|^2 + C\sum_i \xi_i - \sum_i \alpha_i y_i\left(\mathbf{u}^T\mathbf{X}_i\mathbf{v} + b\right) \\ & + \sum_i \alpha_i - \sum_i \alpha_i\xi_i - \sum_i \mu_i\xi_i \end{aligned}$$

Note that

$$\begin{aligned}\frac{1}{2}\|\mathbf{u}\mathbf{v}^T\|^2 &= \frac{1}{2}trace\left(\mathbf{u}\mathbf{v}^T\mathbf{v}\mathbf{u}^T\right)\\ &= \frac{1}{2}\left(\mathbf{v}^T\mathbf{v}\right)trace\left(\mathbf{u}\mathbf{u}^T\right)\\ &= \frac{1}{2}\left(\mathbf{v}^T\mathbf{v}\right)\left(\mathbf{u}^T\mathbf{u}\right)\end{aligned}$$

Thus, we have:

$$\begin{aligned}L_P &= \frac{1}{2}\left(\mathbf{v}^T\mathbf{v}\right)\left(\mathbf{u}^T\mathbf{u}\right) + C\sum_i \xi_i - \sum_i \alpha_i y_i\left(\mathbf{u}^T\mathbf{X}_i\mathbf{v} + b\right)\\ &+ \sum_i \alpha_i - \sum_i \alpha_i\xi_i - \sum_i \mu_i\xi_i\end{aligned}$$

Requiring that the gradient of L_P with respect to $\mathbf{u}$, $\mathbf{v}$, b and ξ_i vanish give the conditions:

$$\mathbf{u} = \frac{\sum_i \alpha_i y_i \mathbf{X}_i\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \tag{9}$$

$$\mathbf{v} = \frac{\sum_i \alpha_i y_i \mathbf{u}^T\mathbf{X}_i}{\mathbf{u}^T\mathbf{u}} \tag{10}$$

$$\sum_i \alpha_i y_i = 0 \tag{11}$$

$$C - \alpha_i - \mu_i = 0, \quad i = 1, \cdots, m \tag{12}$$

From Equations (9) and (10), we see that $\mathbf{u}$ and $\mathbf{v}$ are dependent on each other, and can not be solved independently. In the following, we describe a simple yet effective computational method to solve this optimization problem.

We first fix $\mathbf{u}$. Let $\beta_1 = \|\mathbf{u}\|^2$ and $\mathbf{x}_i = \mathbf{X}_i^T\mathbf{u}$. Thus, the optimization problem (8) can be rewritten as follows:

$$\begin{aligned}\min_{\mathbf{v},b,\xi} \quad & \frac{1}{2}\beta_1\|\mathbf{v}\|^2 + C\sum_{i=1}^m \xi_i\\ \text{subject to} \quad & y_i(\mathbf{v}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\\ & \xi_i \geq 0, \quad i = 1, \cdots, m.\end{aligned} \tag{13}$$

It is clear that the new optimization problem (13) is identical to the standard SVM optimization problem. Thus, we can use the same computational methods of SVM to solve (13), such as [5][10][11].

Once $\mathbf{v}$ is obtained, let $\beta_2 = \|\mathbf{v}\|^2$ and $\tilde{\mathbf{x}}_i = \mathbf{X}\mathbf{v}$. Thus, $\mathbf{u}$ can be obtained by solving the following optimization problem:

$$\begin{aligned}\min_{\mathbf{u},b,\xi} \quad & \frac{1}{2}\beta_2\|\mathbf{u}\|^2 + C\sum_{i=1}^m \xi_i\\ \text{subject to} \quad & y_i(\mathbf{u}^T\tilde{\mathbf{x}}_i + b) \geq 1 - \xi_i,\\ & \xi_i \geq 0, \quad i = 1, \cdots, m.\end{aligned} \tag{14}$$

Table 1: 41 semantic categories from Reuters-21578 used in our experiments

category	**ModeApte**		**category**	**ModeApte**	
	Train	Test		Train	Test
earn	2673	1040	ipi	27	9
acq	1435	620	nat-gas	22	11
crude	223	98	veg-oil	19	11
trade	225	73	tin	17	10
money-fx	176	69	cotton	15	9
interest	140	57	bop	15	8
ship	107	35	wpi	12	8
sugar	90	24	pet-chem	13	6
coffee	89	21	livestock	13	5
gold	70	20	gas	10	8
money-supply	70	17	orange	12	6
gnp	49	14	retail	15	1
cpi	45	15	strategic-metal	9	6
cocoa	41	12	housing	13	1
alum	29	16	zinc	8	4
grain	38	7	lumber	7	4
copper	31	13	fuel	4	7
jobs	32	10	carcass	6	5
reserves	30	8	heat	6	4
rubber	29	9	lei	8	2
iron-steel	26	11			

Again, we can use the standard SVM computational methods to solve this optimization problem. Thus, $\mathbf{v}$ and $\mathbf{u}$ can be obtained by iteratively solving the optimization problems (13) and (14). In our experiments, $\mathbf{u}$ is initially set to the vector of all ones.

By comparing the optimization problems of SVM (2) and STM (8), also noting that X_i in (8) is converted from $\mathbf{x}_i$ in (2) through the "vector-to-tensor" conversion described in Section 3, STM can be thought of as a special case of SVM with the following constraint:

$$w_{n_1(j-1)+i} = u_i v_j \tag{15}$$

For n-dimensional documents, the $\mathbf{w}$ in decision function of SVM is also n-dimensional, so there are $n+1$ ($\approx n_1 \times n_2 + 1$) parameters for SVM. For STM, there are only $n_1 + n_2 + 1$ parameters. Therefore, STM is much more computationally tractable and especially suitable for small sample cases.

5.2 Convergence Proof

In this section, we provide a convergence proof of the iterative computational method described above. We have the following theorem:

Theorem 1 *The iterative procedure to solve the optimization problems (13) and (14) will monotonically decreases the objective function value in (8), and hence the STM algorithm converges.*

Proof Define:

$$f(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\|\mathbf{u}\mathbf{v}^T\|^2 + C\sum_{i=1}^{m}\xi_i$$

Let $\mathbf{u}_0$ be the initial value. Fixing $\mathbf{u}_0$, we get $\mathbf{v}_0$ by solving the optimization problem (13). Likewise, fixing $\mathbf{v}_0$, we get $\mathbf{u}_1$ by solving the optimization problem (14).

Notice that the optimization problem of SVM is convex, so the solution of SVM is globally optimum [1][4]. Specifically, the solutions of equations (13) and (14) are globally optimum. Thus, we have:

$$f(\mathbf{u}_0, \mathbf{v}_0) \geq f(\mathbf{u}_1, \mathbf{v}_0)$$

Finally, we get:

$$f(\mathbf{u}_0, \mathbf{v}_0) \geq f(\mathbf{u}_1, \mathbf{v}_0) \geq f(\mathbf{u}_1, \mathbf{v}_1) \geq f(\mathbf{u}_2, \mathbf{v}_1) \geq \cdots$$

Since f is bounded from below by 0, it converges. ∎

5.3 From Matrix to High Order Tensor

The STM algorithm described above takes order-2 tensors, *i.e.*, matrices, as input data. However, the algorithm can also be extended to high order tensors. In this section, we briefly describe the STM algorithm for high order tensors.

Let (T_i, y_i), $i = 1, \cdots, m$ denote the training samples, where $T_i \in \mathcal{R}^{n_1} \otimes \cdots \otimes \mathcal{R}^{n_k}$. The decision function of STM is:

$$f(T) = T(\mathbf{a}^1, \mathbf{a}^2, \cdots, \mathbf{a}^k) + b$$

$$\mathbf{a}^1 \in \mathbb{R}^{n_1}, \mathbf{a}^2 \in \mathbb{R}^{n_2}, \cdots, \mathbf{a}^k \in \mathbb{R}^{n_k}$$

where

$$T(\mathbf{a}^1, \mathbf{a}^2, \cdots, \mathbf{a}^k) = \sum_{\substack{1 \leq i_1 \leq n_1 \\ \vdots \\ 1 \leq i_k \leq n_k}} T_{i_1,\cdots,i_k} a^1_{i_1} \times \cdots \times a^k_{i_k}$$

As before, $\mathbf{a}^1, \cdots, \mathbf{a}^k$ can also be computed iteratively.

We first introduce the l-mode product of a tensor T and a vector $\mathbf{a}$, which we denote as $T \times_l \mathbf{a}$. The result of l-mode product of a tensor $T \in \mathcal{R}^{n_1} \otimes \cdots \otimes \mathcal{R}^{n_k}$ and a vector $\mathbf{a} \in \mathbb{R}^{n_l}, 1 \leq l \leq k$ will

Table 2: 56 semantic categories from TDT2 used in our experiments

category	**doc num**	**category**	**doc num**	**category**	**doc num**
20001	1844	20087	98	20007	28
20015	1828	20096	76	20041	28
20002	1222	20021	74	20064	28
20013	811	20026	72	20089	25
20070	441	20008	71	20034	25
20044	407	20056	66	20004	21
20076	272	20037	65	20063	19
20071	238	20065	63	20043	18
20012	226	20005	58	20083	17
20023	167	20074	56	20078	16
20048	160	20009	52	20072	13
20033	145	20091	51	20029	13
20039	141	20031	49	20093	12
20086	140	20024	47	20084	12
20032	131	20042	35	20028	12
20047	123	20020	34	20050	11
20019	123	20011	33	20085	10
20077	120	20022	31	20053	10
20018	104	20017	29		

be a new tensor $B \in \mathcal{R}^{n_1} \otimes \cdots \otimes \mathcal{R}^{n_{l-1}} \otimes \mathcal{R}^{n_{l+1}} \otimes \cdots \mathcal{R}^{n_k}$, where

$$B_{i_1,\cdots,i_{l-1},i_{l+1},\cdots,i_k} = \sum_{i_l=1}^{n_l} T_{i_1,\cdots,i_{l-1},i_l,i_{l+1},\cdots,i_k} \cdot a_{i_l}$$

Thus, the decision function in higher order tensor space can also be written as:

$$f(T) = T \times_1 \mathbf{a}^1 \times_2 \mathbf{a}^2 \cdots \times_k \mathbf{a}^k + b$$

The optimization problem of STM in high order tensors is:

$$\begin{aligned} \min_{\mathbf{a}^1,\cdots,\mathbf{a}^k,b,\xi} \quad & \frac{1}{2}\|\mathbf{a}^1 \otimes \cdots \otimes \mathbf{a}^k\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(T_i(\mathbf{a}^1, \mathbf{a}^2, \cdots, \mathbf{a}^k) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{16}$$

Here $\|\mathbf{a}^1 \otimes \cdots \otimes \mathbf{a}^k\|$ denotes the tensor norm of $\mathbf{a}^1 \otimes \cdots \otimes \mathbf{a}^k$ [8].

First, to compute $\mathbf{a}^1$, we fix $\mathbf{a}^2, \cdots, \mathbf{a}^k$. Let $\beta_2 = \|\mathbf{a}^2\|^2, \cdots, \beta_k = \|\mathbf{a}^k\|^2$. We then define

Table 3: Performance comparison on Reuters-21578

Train&Test split	Method	micro F1	macro F1
5% Train	SVM	.8490	.3355
	STM	**.8666**	**.4818**
10% Train	SVM	.8841	.4553
	STM	**.8908**	**.5735**
30% Train	SVM	**.9225**	.6618
	STM	.9196	**.7357**
50% Train	SVM	**.9355**	.7247
	STM	.9282	**.7826**
ModApte	SVM	**.9368**	.7356
	STM	.9321	**.7877**

Table 4: Statistical significance tests on Reuters-21578

Train&Test split	sysA	sysB	s-test	S-test	T-test
5% Train	STM	SVM	$\gg$	$\gg$	$\gg$
10% Train	STM	SVM	$>$	$\gg$	$\gg$
30% Train	STM	SVM	$<$	$\gg$	$\gg$
50% Train	STM	SVM	$<$	$\gg$	$\gg$
ModApte	STM	SVM	$<$	$>$	$\gg$

"$\gg$" or "$\ll$" means P-value ≤ 0.01

"$>$" or "$<$" means $0.01 <$ P-value ≤ 0.05

"$\sim$" means P-value > 0.05

$\mathbf{t}_i = T_i \times_2 \mathbf{a}^2 \cdots \times_k \mathbf{a}^k$. Thus, the optimization problem (16) can be reduced as follows:

$$\begin{aligned} \min_{\mathbf{a}^1, b, \xi} \quad & \frac{1}{2}\beta_2 \cdots \beta_k \|\mathbf{a}^1\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i({\mathbf{a}^1}^T \mathbf{t}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{17}$$

Again, we can use the standard SVM computational methods to solve this optimization problem. Once $\mathbf{a}^1$ is computed, we can fix $\mathbf{a}^1, \mathbf{a}^3, \cdots, \mathbf{a}^k$ to compute $\mathbf{a}^2$. So on, all the $\mathbf{a}^i$ can be computed in such iterative manner.

6 EXPERIMENTS

In this section, several experiments were performed to show the effectiveness of our proposed algorithm. Two standard document collections were used in our experiments: Reuters-21578 and TDT2. We compared our proposed algorithm with Support Vector Machines.

6.1 Data Corpora

Reuters-21578 corpus[1] contains 21578 documents in 135 categories. The ModApte version of Reuters-21578 is used in our experiments. Those documents with multiple category labels are discarded, and the categories with more than 10 documents are kept. It left us with 8213 documents in 41 categories as described in Table 1. For ModeApte split, there are 5899 training documents and 2314 testing documents. After preprocessing, this corpus contains 18933 distinct terms.

The TDT2 corpus[2] consists of data collected during the first half of 1998 and taken from six sources, including two newswires (APW, NYT), two radio programs (VOA, PRI) and two television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this dataset, we also removed those documents appearing in two or more categories and use the categories which contain more than 10 documents thus leaving us with 10021 documents in 56 categories as described in Table 2. After preprocessing, this corpus contains 36771 distinct terms.

Each document is represented as a term-frequency vector and each document vector is normalized to 1. We simply removed the stop words, and no further preprocessing was done. For STM, we empirically sort the features (words) according to their document frequency and then convert the vector into a $n_1 \times n_2$ tensor such that $n_2 = 50$.

6.2 Evaluation Metric

The classification performance is evaluated by comparing the predicted label of each testing document with that provided by the document corpus. The standard recall, precision and F_1 measure are used here [17]. Recall is defined to be the ratio of correct assignments by the classifier divided by the total number of correct assignments. Precision is the ratio of correct assignments by the classifier divided by the total number of the classifier's assignments. The F_1 measure combines recall (r) and precision (p) with an equal weight in the following form:

$$F_1(r, p) = \frac{2rp}{r + p}$$

These scores can be computed for the binary decisions on each individual category first and then be averaged over categories. Or, they can be computed globally over all the $n \times m$ binary decisions where n is the number of total test documents, and m is the number of categories in consideration. The former way is called *macro-averaging* and the latter way is called *micro-averaging*. It is understood that the micro-averaged scores tend to be dominated by the classifier's performance on common categories, and the macro-averaged scores are more influenced by the performance on rare categories. Providing both kinds of scores is more informative than providing either alone.

[1] Reuters-21578 corpus is at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

[2] Nist Topic Detection and Tracking corpus is at `http://www.nist.gov/speech/tests/tdt/tdt98/index.html`

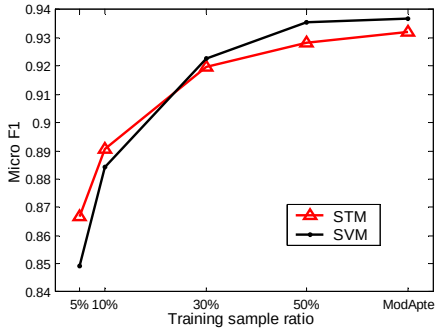

Figure 2: Micro-averaged F1 on Reuters-21578

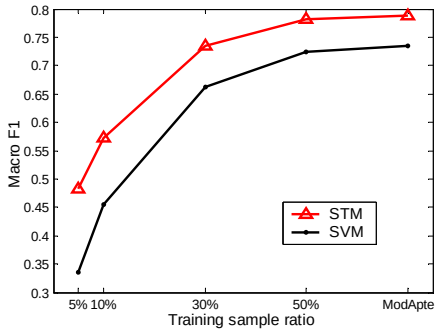

Figure 3: Macro-averaged F1 on Reuters-21578

We also use a set of significance tests for comparing two classification methods with various performance measures. These significance tests are:

- *Micro sign test (s-test)*: A sign test designed for comparing two systems, A and B, based on their decisions on all the document/category pairs.
- *Macro sign test (S-test)*: A sign test for comparing two systems, A and B, using the paired F_1 values for individual categories.
- *Macro t-test (T-test)*: A t-test for comparing two systems, A and B, using the paired F_1 values for individual categories.

For more details about these significance tests, please refer to [18].

6.3 Experimental Results

We used the LIBSVM system [3] and tested it with the linear model, since previous researches [18] show that linear SVM is effective enough for text categorization.

The dataset was randomly split into training and testing sets. In order to examine the effectiveness of the proposed algorithm with different size of the training set, we ran several tests that the

Table 5: Performance comparison on TDT2

Train&Test split	Method	micro F1	macro F1
5% Train	SVM	.8881	.6477
	STM	**.9064**	**.7507**
10% Train	SVM	.9292	.7692
	STM	**.9343**	**.8317**
30% Train	SVM	**.9602**	.9063
	STM	.9563	**.9152**
50% Train	SVM	**.9684**	.9307
	STM	.9640	**.9402**

Table 6: Statistical significance tests on TDT2

Train&Test split	sysA	sysB	s-test	S-test	T-test
5% Train	STM	SVM	$\gg$	$\gg$	$\gg$
10% Train	STM	SVM	$\gg$	$\gg$	$\gg$
30% Train	STM	SVM	$\ll$	$\sim$	$\sim$
50% Train	STM	SVM	$\ll$	$>$	$>$

training set contains $5\%, 10\%, 30\%$ and 50% documents, for both Reuters-21578 and TDT2. In all these splits, we kept at least two documents in every category of the training set. For each test, we averaged the results over 10 random splits. Moreover, for Reuters-21578 dataset, we also tested on the ModApte split which contains around 70% training sample.

Table 3 and 5 show the classification results on two datasets. Table 4 and 6 summarized the statistical significance tests. Figure (2,3,4,5) show the performance with respect to the training set size.

As can be seen from the above results, when the training set is small (5% and 10%), STM outperforms SVM on both micro-averaged F_1 and macro-averaged F_1. As the number of training samples increases, STM performed better than SVM on macro-averaged F_1 but worse on micro-averaged F_1.

As we know, the micro-level measure is dominated by the performance of the classifiers on large categories, while the macro-level measure is more sensitive to the performance of the classifiers on small categories [18]. The experimental results show the greater performance of STM on small training sample cases over SVM.

To get a more detailed picture of the performance difference between STM and SVM over different size of training samples, we plot the performance curves of STM and SVM over different categories in Figure 6 and 7. The categories are sorted by the number of training samples. For those categories with the same number of training samples, we averaged their F_1 scores. We can

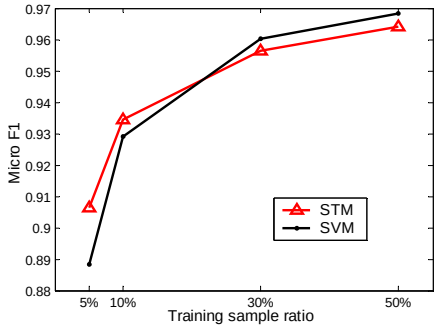

Figure 4: Micro-averaged F1 on TDT2 dataset

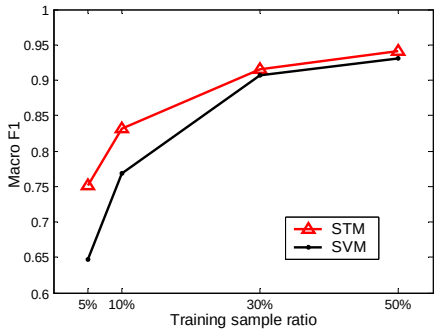

Figure 5: Macro-averaged F1 on TDT2 dataset

see that STM is better than SVM on the left side of the figures (corresponding to small training set) but worse than SVM on the right side of the figures. This observation indicates that our STM algorithm is especially suitable for small sample problems. This is due to the fact that the number of parameters need to be estimated in STM is n_1+n_2+1 which can be much smaller than $n_1 \times n_2+1$ in SVM.

7 Conclusions

In this paper we have introduced a tensor framework for document representation and classification. In particular, we have proposed a new classification algorithm called **Support Tensor Machines** (STM) for learning a linear classifier in tensor space. Our experimental results on Reuters-21578 and TDT2 databases demonstrate that STM is especially suitable for small sample cases. This is due to the fact that the number of parameters estimated by STM is much less than that estimated by standard SVM.

There are several interesting problems that we are going to explore in the future work:

1. In this paper, we empirically construct the tensor. The better ways of converting a document vector to a document tensor with theoretical guarantee need to be studied.

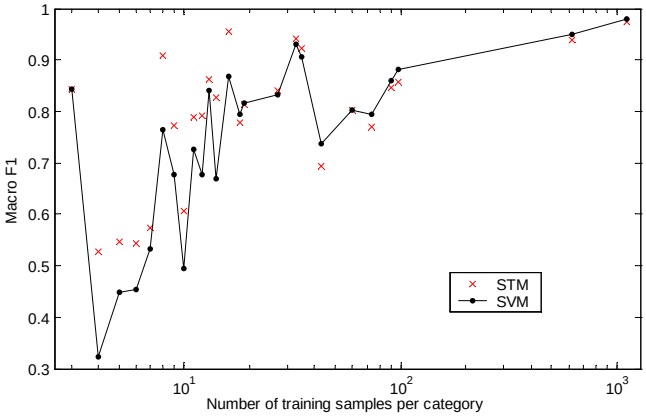

Figure 6: Performance curves of all categories on Reuters-21578 dataset (30% training samples case)

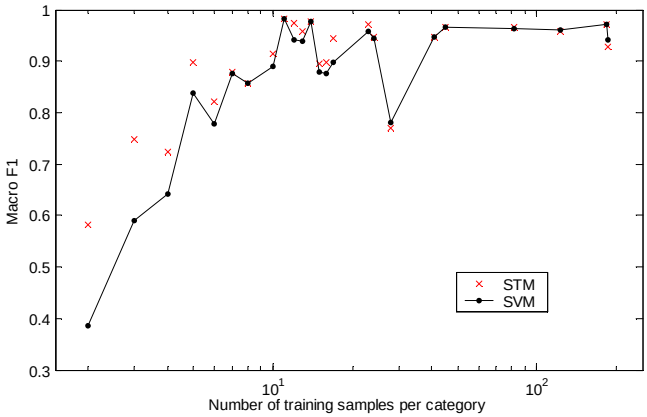

Figure 7: Performance curves of all categories on TDT2 dataset (10% training samples case)

2. STM is a linear method. Thus, it fails to discover the nonlinear structure of the data space. It remains unclear how to generalized our algorithm to nonlinear case. A possible way of nonlinear generalization is to use kernel techniques.

3. In this paper, we use a iterative computational method for solving the optimization problem of STM. We expect that there exists more efficient computational methods.

References

[1] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[2] Deng Cai, Xiaofei He, and Jiawei Han. Subspace learning based on tensor analysis. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2005-2572, May, 2005.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] R. Fletcher. *Practical methods of optimization*. John Wiley and Sons, 2nd edition edition, 1987.

[5] Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade SVM. In *Advances in Neural Information Processing Systems 17*, 2004.

[6] Xiaofei He, Deng Cai, and Partha Niyogi. Tensor subspace analysis. In *Advances in Neural Information Processing Systems 18*, 2005.

[7] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML'98*, 1998.

[8] John M. Lee. *Introduction to Smooth Manifolds*. Springer-Verlag New York, 2002.

[9] 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.

[10] John Platt. Using sparseness and analytic qp to speed training of support vector machines. In *Advances in Neural Information Processing Systems 11*, 1998.

[11] John C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[12] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *ECCV'02*, 2002.

[13] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for information retrieval. *Communications of the ACM*, 18(11):613–620, 1975.

[14] V. N. Vapnik. *Estimation of dependences based on empirical data*. Springer-Verlag, 1982.

[15] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[16] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[17] Yiming Yang. An evaluation of statistical approaches to text categorization. *Jounal of Information Retrieval*, 1(1/2):67–88, 1999.

[18] Yiming Yang and Xin Liu. A re-examination of text categoriztion methods. In *SIGIR'99*, 1999.

[19] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *Proc. 2003 Int. Conf. on Research and Development in Information Retrieval (SIGIR'03)*, pages 267–273, Toronto, Canada, Aug. 2003.

[20] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. In *Advances in Neural Information Processing Systems 17*, 2004.