Text Categorization Based on Regularized Linear Classification Methods

Tong Zhang
Mathematical Sciences Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
tzhang@watson.ibm.com

Frank J. Oles
Mathematical Sciences Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
oles@watson.ibm.com

Abstract

A number of linear classification methods such as the linear least squares fit (LLSF), logistic regression, and support vector machines (SVM's) have been applied to text categorization problems. These methods share the similarity by finding hyperplanes that approximately separate a class of document vectors from its complement. However, support vector machines are so far considered special in that they have been demonstrated to achieve the state of the art performance. It is therefore worthwhile to understand whether such good performance is unique to the SVM design, or if it can also be achieved by other linear classification methods. In this paper, we compare a number of known linear classification methods as well as some variants in the framework of regularized linear systems. We will discuss the statistical and numerical properties of these algorithms, with a focus on text categorization. We will also provide some numerical experiments to illustrate these algorithms on a number of datasets.

1 Background

The text categorization problem is to determine predefined categories for an incoming unlabeled message or document containing text based on information extracted from a training set of labeled messages or documents. Text categorization is an important practical problem for companies that wish to use computers to categorize incoming email, thereby either enabling an automatic machine response to the email or simply assuring that the email reaches the correct human recipient. But, beyond email, text items to be categorized may come from many sources, including the output of voice recognition software, collections of documents (e.g., news stories, patents, or case summaries), and the contents of web pages.

The text categorization problem can be reduced to a set of binary classification problems – one for each category – where for each one wishes to determine a method for predicting in-class versus out-of-class membership. Such supervised learning problems have been widely studied in the past. Recently, many methods developed for classification problems have been applied to text categorization. For example, Apte, Damerau, and Weiss [1] applied an inductive rule learning algorithm, SWAP1, to the text categorization problem. In [25], Yang and Chute proposed a linear least squares fit algorithm to train linear classifiers. Yang also compared a number of statistical methods for text categorization in [24]. The best performances previously reported in the literature are from weighted resampled decision trees in [23] and (linear) support vector machines in [12, 4].

Integral parts of all these approaches are tokenization, feature selection, and creating numeric vector representations of documents. The first step, tokenization, is laid out in detail in Figure 1. This functionality is common to most methods of text categorization. In the tokenization procedure depicted in Figure 1, one or both of Steps 4 and 5 may be omitted, although keeping them may improve performance. If both steps are retained, elimination of stopwords (Step 5) may also be done before stemming (Step 4). Also, the elimination of stopwords (Step 5) may in some instances be subsumed by subsequent feature selection, to be discussed below. For consistency, the same tokenization procedure would be used both (1) for documents used in training to build categorization rules and (2) for incoming documents to be categorized by a system employing the classifiers obtained in the training phase.

After tokenization, each document is represented by a list of word occurrences. A program should then be used to carry out feature selection. (However, feature selection could also be skipped entirely, so that tokens were to be taken by themselves to be the sole features of interest.) We will not take up the specifics of feature selection, but a number of methods of varying degrees of sophistication have been studied in [27]. Feature selection might be done only once for the entire data set, but experience indicates that better results will be obtained by doing feature selection separately for each category, reflecting the intuition that features indicative of category membership will differ as one moves from category to category. Feature selection, under the assumption that a separate set of features is to be selected for each category, is laid out in Figure 2. The output of feature selection would normally be a specification of a separate list of selected features (words) for each category for which we intend to generate a classifier. The specification would necessarily be detailed enough to permit a computer to identify each occurrence of each feature in the tokenized representation of a document.

After feature selection, each document is represented by a vector of word occurrences for each category where each vector component corresponds to a word feature selected for the category in the previous step. Figure 3 shows the steps necessary to use a list of features selected for relevance to a particular category to convert a tokenized representation of a document to a numeric vector representing the document. Because of the vast numbers of different words that may appear in text, generally the numerical vectors of world occurrences one gets are sparse vectors of very high dimensionality. Thus, text categorization necessitates using techniques of supervised two-class categorization problem that are well suited to high-dimensional sparse data.

Formally, a two-class categorization problem is to determine a label $y \in \{-1, 1\}$ associated with a vector x of input variables. A useful method for solving this problem is by using linear discriminant functions, which consist of linear combinations of the input variables. Various techniques have been proposed for determining the weight values for linear discriminant classifiers from a training set of labeled data $(x_1, y_1), \ldots, (x_n, y_n)$. Here, and throughout this document, n is the number of items in a training set. Specifically, we seek a weight vector w and a threshold θ such that $w^T x < \theta$ if its label $y = -1$ and $w^T x \geq \theta$ if its label $y = 1$. Thus, the hyperplane consisting of all x such that $w^T x = \theta$ would approximately separate the in-class vectors from the out-of-class vectors.

The problem just described may readily be converted into one in which the threshold θ is taken to be zero. One does this by embedding all the data into a space with one more dimension, and then translating the original space by some chosen nonzero distance A from its original position. Normally, one takes $A = 1$. Hence, in this conversion, each vector $(z_1, \ldots, z_m)$ is traded in for $(z_1, \ldots, z_m, A)$. For each hyperplane in the original space, there is a unique hyperplane in the larger space that passes through the origin of the larger space. Instead of searching for both an m-dimensional weight vector along with a threshold θ, we can therefore search for an $(m+1)$-dimensional weight vector along with an anticipated threshold of zero.

Under the assumption that the vectors of input variables have been suitably transformed so that we may take $\theta = 0$, the training error rate for a linear classifier with weight vector w is given by

$$\frac{1}{n} \sum_{i=1}^{n} s(w^T x_i y_i), \tag{1}$$

where s is the step function

$$s(z) = \begin{cases} 1 & \text{if } z \leq 0, \\ 0 & \text{if } z > 0. \end{cases} \tag{2}$$

A number of approaches to solving categorization problems by finding linear discriminant functions have been advanced over the years. In the early statistical literature, the weight was obtained by using linear discriminant analysis, which makes the assumption that each class has a Gaussian distribution (cf. [17], chapter 3). Similar to linear discriminant analysis, an approach widely used in statistics (usually for regression rather than classification) is the least squares fit algorithm. Least squares fit has been applied to text categorization problems in [25]. Without any assumption on the underlying distribution, a linear separator can be obtained by using the perceptron scheme that minimizes the training error [15]. Another commonly used approach in statistics for obtaining a linear classifier is logistic regression. Logistic regression is closely related to support vector machines, which have recently gained much popularity.

There have been a long history of using logistic regression in information retrieval, as can be seen from the following partial list [2, 5, 6, 10, 13, 20]. However, for a number of reasons, the method was not used in an effective way for text categorization. As a result, the comparison in [20] suggested negative opinions on the performance of logistic regression. The combination of the following factors could have led to the negative results in [20]. One reason could be the lack of regularization in their formulation. Keep in mind that regularization is important to the success of SVM. We will come back to the issue of regularization in the next section. Another reason could be their choice of the Newton-Raphson method of numerical optimization, which in our experience could become unstable, especially without regularization. In this paper, we instead introduce a stable numerical optimization procedure. A third reason could be the threshold adjustment problem which we will explain in Section 4 — however, this problem does not affect the break-even performance measurement. Furthermore, in the previous studies, logistic regression was typically employed with a relatively small set of features. However, as suggested in [12], it can be helpful to use a very large set of features (in the order of tens of thousands). Our experiments in Section 4 confirm this observation.

This paper is organized as follows. In Section 2, we describe various linear classification algorithms. We discuss issues related to the effectiveness of the algorithms from a statistical point of view. Using this analysis, we introduce a method that is a variant of the linear least squares method and SVM. The presence of large feature sets has introduced some numerical challenges. These issues will be discussed in Section 3, where we investigate the computational aspects of all the algorithms including logistic regression and support vector machines. Some experiments will be given in Section 4 to compare the performance of the different methods. In particular, we illustrate that logistic regression achieves results comparable with support vector machines. We summarize this paper in Section 5.

2 Linear classifiers

We start our discussion with the least squares algorithm, which is based on the following formulation to compute a linear separator $\hat{w}$:

$$\hat{w} = \arg\inf_{w} \frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2. \tag{3}$$

The solution is given by

$$\hat{w} = (\sum_{i=1}^{n} x_i x_i^T)^{-1} (\sum_{i=1}^{n} x_i y_i).$$

One problem with the above formulation is that the matrix $\sum_{i=1}^{n} x_i x_i^T$ may be singular or ill-conditioned. This occurs, for example, when n is less than the dimension of x. Note that in this case, for any $\hat{w}$, there exists infinitely many solutions $\tilde{w}$ of $\tilde{w}^T x_i = \hat{w}^T x_i$ for $i = 1, \dots, n$. This implies that (3) has infinitely many possible solutions $\hat{w}$.

A remedy of this problem is to use a pseudo-inverse [25]. However, one problem of the pseudo-inverse approach is its computational complexity. In order to handle large sparse systems, we need to use iterative algorithms which do not rely on matrix factorization techniques. Therefore in this paper, we use the standard ridge regression method [9] that adds a regularization term to (3):

$$\hat{w} = \arg\inf_w \frac{1}{n} \sum_{i=1}^{n} (w^T x_i y_i - 1)^2 + \lambda w^2, \tag{4}$$

where λ is an appropriately chosen regularization parameter. The solution is given by

$$\hat{w} = (\sum_{i=1}^{n} x_i x_i^T + \lambda n I)^{-1} (\sum_{i=1}^{n} x_i y_i),$$

where I denotes the identity matrix. Note that $\sum_{i=1}^{n} x_i x_i^T + \lambda n I$ is always non-singular, which solves the ill-condition problem.

Statistically, the least squares formulation (4) is often associated with the regression model $y_i = w^T x_i + n_i$, where n_i is assumed to be an iid noise. If we further assume that the noise n_i comes from a zero-mean Gaussian distribution, then the formulation corresponds to a penalized maximum likelihood estimate with a Gaussian prior. Under such a condition, the least squares method can be the optimal estimator in the sense that it corresponds to the optimal Bayesian estimator in the Bayesian framework. However, in other situations, this method is typically sub-optimal. Unfortunately, the Gaussian noise assumption, which is continuous, can only be satisfied approximately for classification problem, since $y_i \in \{-1, 1\}$ is discrete. In statistics, logistic regression (cf. [8] Section 4.5) has often been employed to remedy this problem.

As we have pointed out before, even though there have been considerable interest in applying logistic regression for text categorization, its effectiveness has not been fully explored. Despite the close relationship between logistic regression and support vector machines, no results on text categorization using direct large scale logistic regression have been reported in the literature on the standard Reuters dataset. We shall therefore briefly describe the logistic regression method and its connection to linear SVM's below for completeness.

In logistic regression, we model the conditional probability $P(y = 1|w, x)$ as $1/(\exp(-w^T x)+1)$. It then follows that the likelihood $P(y|w, x) = 1/(\exp(-w^T xy)+1)$ (note that $P(-1|w, x)+P(1|w, x) = 1$). A standard procedure to obtain an estimate of w is by the maximum likelihood estimate which minimizes

$$\hat{w} = \arg\inf_w \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-w^T x_i y_i)). \tag{5}$$

Similar to (3), formulation (5) may be ill-conditioned numerically. The standard method to solve this problem is by adding a scaling of the identity matrix to the Hessian of (5), which is equivalent to the following regularized logistic regression formulation:

$$\hat{w} = \arg\inf_w \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-w^T x_i y_i)) + \lambda w^2, \tag{6}$$

where λ is an appropriately chosen regularization parameter. This regularized formulation also belongs to a class of methods called penalized likelihood (cf. [8], Section 6.5.2), which in the Bayesian framework, can be interpreted as a MAP estimator with prior $P(w) \propto \exp(\lambda w^2)$. Furthermore, we can employ a slightly more general family $a \ln(1 + b \exp(cz))$, instead of the log-likelihood function $\ln(1 + \exp(-z))$ in (6), where both a and c can be absorbed into the regularization parameter λ.

The support vector machine is a method originally proposed by Vapnik [3, 19, 21] that has nice properties from the sample complexity theory. It is designed as a modification of the Perceptron algorithm. Slightly different from our approach of forcing threshold $\theta = 0$, and then compensating by appending a constant component A (usually we take $A = 1$) to each document vector, the standard linear support vector machine (cf. [19], chapter 1) explicitly includes θ into a quadratic formulation as follows:

$$
\begin{aligned}
(\hat{w}, \hat{\theta}) &= \arg\inf_{w,\theta} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda w^2, \qquad (7)\\
\text{s.t.} \quad & y_i(w^T x_i - \theta) \geq 1 - \xi_i, \quad \xi_i \geq 0, \qquad i = 1, \ldots, n.
\end{aligned}
$$

By eliminating ξ_i, the above formula is equivalent to the following formulation:

$$
(\hat{w}, \hat{\theta}) = \arg\inf_{w,\theta} \frac{1}{n} \sum_{i=1}^{n} g(y_i(w^T x_i - \theta)) + \lambda w^2, \tag{8}
$$

where

$$
g(z) = \begin{cases} 1 - z & \text{if } z \leq 1, \\ 0 & \text{if } z > 1. \end{cases} \tag{9}
$$

As a comparison, we plot the SVM loss function $g(z)$, the logistic loss function $\ln(1 + \exp(-z))$, and a modified logistic loss function $\frac{1}{\gamma} \ln(1 + \exp(\gamma(1 - z)))$ in Figure 4. The shapes of all three functions are very similar. Much of the differences about their shapes can be absorbed into a scaling of the regularization parameter λ; the remaining differences are insignificant for text categorization purposes (as we will later demonstrate in our experiments). We shall also note that $g(z) = \lim_{\gamma \to \infty} \frac{1}{\gamma} \ln(1 + \exp(\gamma(1 - z)))$, and hence the support vector machine formulation can be regarded as an approximation and the limiting case of a generalized family of logistic regression formulations.

The standard support vector machine should be solved directly as the quadratic programming problem (7) or in its dual formulation (which we describe later), since the non-smoothness of its corresponding loss function $g(z)$ in (8) introduces difficulties for the direct numerical optimization of (8). However, if the loss function $g(z)$ in (8) can be replaced by a smooth function (as in the case of logistic regression and the least squares fit algorithm), then it will be much easier to be solved directly in its primal formulation.

Thus we shall intentionally replace $g(z)$ by a smoother function, and so we consider the following alternative formulation

$$
\hat{w} = \arg\inf_{w} \frac{1}{n} \sum_{i=1}^{n} h(w^T x_i y_i) + \lambda w^2, \tag{10}
$$

where

$$h(z) = \begin{cases} (1-z)^2 & \text{if } z \leq 1, \\ 0 & \text{if } z > 1. \end{cases} \tag{11}$$

This formulation is sufficiently smooth so that direct numerical optimization of (10) can be performed efficiently (see Section 3). Compared with logistic regression, it has an advantage that no expensive exponential function evaluation is required during the numerical optimization.

Although this particular choice of $h(z)$ is equivalent to an instance of a more general form of support vector machine (see [19], chapter 1), it is interesting to note that the formulation (10) is intentionally excluded from a standard SVM as described both in chapter 1 of [19] and in chapter 10 of [21]. Their argument comes from the impression that $g(z)$ looks to be a better approximation to the step function $s(z)$ than $h(z)$ does. This argument relies on the statistical learning theory basis of SVM that regards $g(z)$ as an approximation to the classification error function $s(z)$.

However, the above reasoning is only partially correct. From the statistical point of view, if we regard the logistic model

$$P(y|w,x) = 1/(\exp(-w^T xy) + 1)$$

as a good approximation to the true conditional probability $P(y|x)$, then by the well-known asymptotic optimality property of the maximum likelihood estimate (that is, the maximum likelihood estimate is an asymptotically most efficient estimator of a distribution dependent parameter among all asymptotically unbiased estimators), an estimator based on (5) is asymptotically more efficient than an estimator based on (1) such as the support vector machine. This suggests that to achieve better efficiency, we need a better distribution model and we need to incorporate such a model into a regularized maximum likelihood estimate, where the regularization is to control numerical stability.

From this point of view, (10) can be a better distribution model of $P(y|w,x)$ than (3) since it captures the heavy tail probability of the histogram of $w^T xy$ presented in the more reasonable logistic model of conditional likelihood. As explained earlier, the Gaussian noise assumption of (3) can only be satisfied approximately. Figure 5 compares the histograms of $\hat{w}^T xy$ on the training set at the estimated projection parameter $\hat{w}$ using the least squares method (4) and the modified least squares method (10). The experiment is done with the "earn" category (the largest category) in the Reuters news story dataset (see Section 4). Not surprisingly, the least squares method computes a $\hat{w}$ that fits a Gaussian-like histogram while the modified least squares method accommodates a heavy tailed distribution which is more appropriate for classification problems.

Since the least squares fit method has a close to the state of the art performance [24], it is reasonable to expect the state of the art performance can be achieved from the better distribution model based on (10). On the other hand, statistical learning theory in its current stage lacks both the concept of efficiency and the concept of distribution modeling, therefore it does not fully explain the superior performance of (10) compared with the standard support vector machine formulation (7) on text categorization problems reported in Section 4.

Due to the above mentioned different philosophies, and to avoid unnecessary confusion with the standard SVM (and a modified SVM we introduce later), it is appropriate to regard (10) as a modified (regularized) least squares method (4) that incorporates the heavy tail distribution aspect of the logistic model into the Gaussian model of the least squares method. Compared with the logistic model, the modified least squares formula has the advantage of numerical efficiency (there is no need to evaluate the time consuming exponential functions during the computation). However, the probability interpretation of the logistic model is often very desirable in many applications.

It is possible to derive kernel methods (see [19]) for all algorithms mentioned in this section by using the dual formulation (see Section 3). However, for text categorization problems, there is not sufficient evidence to show that kernel methods help. For example, results from [4] and from this paper with plain linear classifiers are slightly better than those of [12] with kernels. Note that the effectiveness of kernels relies on the assumption that high order word correlations (word pairs, word triples, etc) convey more information than single words. However, some of this correlation can already be captured by linear combination of word occurrences. We have conducted some preliminary experiments which failed to demonstrate statistically significant evidence that high order methods such as using word pairs or kernels are helpful. In addition, kernel classifiers are much more complex computationally. We will thus avoid these methods in the paper.

3 Numerical algorithms

For text categorization problems, input vector x_i is usually very sparse and of large dimensionality. The traditional numerical techniques based on small dense problems for (4) and (6) may not be suitable, especially if they require some kind of matrix factorization. In this section, we study iterative methods for solving large sparse systems associated with schemes discussed in this paper.

In the following, we use i to denote the index of training samples, and j to denote the feature index. For example: x_{ij} is the j-th component of the i-th training sample x_i, while w_j is the j-th component of the weight vector.

3.1 A column relaxation algorithm

We consider a more general formulation

$$\hat{w} = \arg\inf_w \frac{1}{n}\sum_{i=1}^{n} f(w^T x_i y_i) + \lambda w^2, \qquad (12)$$

where f is a relatively smooth function which has a continuous first order derivative and a non-negative piecewise continuous second order derivative. In this case, the formulation in (12) is convex, thus it has a unique local minimum which is also the global minimum. Methods investigated in this section are based on the following generic relaxation algorithm (for example, see [7]):

Algorithm 1 *(Gauss-Seidel)*

let $w = 0$ and $r_i = w^T x_i = 0$
for $k = 1, 2, \ldots$
  for $j = 1, \ldots, d$
    find Δw_j by approximately minimizing
      $\frac{1}{n} \sum_i f(r_i + \Delta w_j x_{ij} y_i) + \lambda (w_j + \Delta w_j)^2 \qquad (*)$
    update r: $r_i = r_i + \Delta w_j x_{ij} y_i \qquad (i = 1, \ldots, n)$
    update w: $w_j = w_j + \Delta w_j$
  end
end

In the following, we will specialize Algorithm 1 to produce computational routines for solving the regularized linear least squares fit formulation (i.e. ridge regression) in (4), the modified least squares formulation in (10), and the regularized logistic regression formulation in (6).

For the regularized linear least squares fit formulation (4), we obtain Algorithm 2 where $(*)$ is solved exactly. Note that for linear systems such as (4), the conjugate gradient (CG) procedure is often preferred over the Gauss-Seidel method. In addition, a preconditioned CG with a symmetrized Gauss-Seidel method as the preconditioner usually performs better. In fact, without taking advantage of possible special structures (if any), this combination is the method of choice as a general purpose symmetric large sparse linear system solver, For more details, see relevant chapters in [7]. Even though CG can also be used for nonlinear optimization problems, due to a number of reasons, it is not very suitable to systems considered in this paper. For simplicity and consistent with other methods, we shall use Algorithm 2 to solve (4) in our experiments.

Algorithm 2 *(CLS)*

let $w = 0$ and $r_i = 0$
for $k = 1, 2, \ldots$
  for $j = 1, \ldots, d$
    $\Delta w_j = -(\sum_i (r_i - 1) x_{ij} y_i + \lambda n w_j) / (\sum_i x_{ij}^2 + \lambda n)$
    update r: $r_i = r_i + \Delta w_j x_{ij} y_i \qquad (i = 1, \ldots, n)$
    update w: $w_j = w_j + \Delta w_j$
  end
end

Instead of Algorithm 1, one may also apply Newton's method directly to (12), as suggested in [8] and used in [20]. The resulting linear system from Newton's method can be solved by an iterative solver such as the Gauss-Seidel iteration or CG. However, a properly implemented line search method, which can be complicated, is required to guarantee convergence. Such a line search method can also result in small step sizes, which slows down the convergence. As a comparison, Algorithm 1 has the advantage of simplicity and guaranteed convergence, as long as we update Δw_j so that the objective function (12) decreases in $(*)$

every time. For this purpose, we rewrite (*) by Taylor expansion: $\exists 0 < \eta < 1$ such that

$$\begin{aligned}&[\frac{1}{n}\sum_i f(r_i + \Delta w_j x_{ij} y_i) + \lambda(w_j + \Delta w_j)^2] - [\frac{1}{n}\sum_i f(r_i) + \lambda(w_j)^2] \\ =&[\frac{1}{n}\sum_i f'(r_i) x_{ij} y_i + 2\lambda w_j]\Delta w_j + \frac{1}{2}[\frac{1}{n}\sum_i [f''(r_i + \eta \Delta w_j x_{ij} y_i) x_{ij}^2 + 2\lambda]\Delta w_j^2.\end{aligned}$$

This equality implies that $\forall \Delta_j > 0$, if we let

$$\Delta v_j = -\frac{\frac{1}{n}\sum_i f'(r_i) x_{ij} y_i + 2\lambda w_j}{\frac{1}{n}\sum_i F(r_i, \Delta_j |x_{ij}|) x_{ij}^2 + 2\lambda},$$

where F is an arbitrary function such that $F(r_i, \eta) \geq \sup_{|\Delta r_i| \leq \eta} f''(r_i + \Delta r_i)$, and let

$$\Delta w_j = \begin{cases} -\Delta_j & \Delta v_j < -\Delta_j \\ \Delta v_j & \Delta v_j \in [-\Delta_j, \Delta_j] \\ \Delta_j & \Delta v_j > \Delta_j \end{cases},$$

then the objective function always decreases after step (*). This property ensures the convergence of Algorithm 1.

We would like to apply this idea to (10). To do so, it is helpful to further enhance the smoothness of f by introducing a continuation parameter $c \in [0, 1]$, so that $f(x) = f_0(x)$ and the smoothness of $f_c''(x)$ decreases as c decreases:

$$f_c(x) = \begin{cases} (x-1)^2 & x \leq 1 \\ c(x-1)^2 & x > 1. \end{cases} \tag{13}$$

Algorithm 1 should then be modified so that at each step k, a different c_k is chosen (so that $1 = c_1 \geq c_2 \geq \cdots \geq c_K = 0$), and the function f shall be replaced by f_{c_k}. Note that this introduction of a continuation parameter is not required in order for Algorithm 1 to converge. However, in our experience, this simple modification accelerates the rate of convergence:

Algorithm 3 *(CMLS)*

let $w = 0$ and $r_i = 0$
pick a decreasing sequence of $1 = c_1 \geq c_2 \geq \cdots \geq c_K = 0$
pick $\Delta_j > 0$ for $j = 1, \ldots, d$
for $k = 1, 2, \ldots, K$
  define function $C_k(r_i) = 2(r_i - 1)$ if $r_i \leq 1$ and $C_k(r_i) = 2c_k(r_i - 1)$ otherwise
  define function $F_k(x, y) = 2$ if $x \leq 1 + |y|$ and $F_k(x, y) = 2c_k$ otherwise
  for $j = 1, \ldots, d$
    $\Delta v_j = -[\sum_i C_k(r_i) x_{ij} y_i + 2\lambda n w_j]/[\sum_i F_k(r_i, \Delta_j x_{ij}) x_{ij}^2 + 2\lambda n]$
    $\Delta w_j = \min(\max(\Delta v_j, -\Delta_j), \Delta_j)$
    update r: $r_i = r_i + \Delta w_j x_{ij} y_i$ $(i = 1, \ldots, n)$
    update w: $w_j = w_j + \Delta w_j$
    update Δ_j: $\Delta_j = 2|\Delta w_j| + \epsilon$
  end
end

For logistic regression (6), we obtain the following algorithm with a specific choice of F:
Algorithm 4 *(CLG)*

let $w = 0$ and $r_i = 0$
pick $\Delta_j > 0$ for $j = 1, \ldots, d$
for $k = 1, 2, \ldots, K$
  define function $F_k(x, y) = \min(0.25, e^{|y|}/(2 + e^x + e^{-x}))$
  for $j = 1, \ldots, d$
    $\Delta v_j = -[\sum_i -\frac{1}{\exp(r_i)+1} x_{ij} y_i + 2\lambda n w_j]/[\sum_i F_k(r_i, \Delta_j x_{ij}) x_{ij}^2 + 2\lambda n]$
    $\Delta w_j = \min(\max(\Delta v_j, -\Delta_j), \Delta_j)$
    update r: $r_i = r_i + \Delta w_j x_{ij} y_i \qquad (i = 1, \ldots, n)$
    update w: $w_j = w_j + \Delta w_j$
    update Δ_j: $\Delta_j = 2|\Delta w_j| + \epsilon$
  end
end

In Algorithm 4, we can also use other formulations of $F_k(x, y)$, as long as the computation can be organized efficiently.

The update formula for Δ_j is to make sure that a sufficiently large step can be taken. ϵ is a tolerance parameter than can be set to 0.1 for text categorization problems. The initial choice of Δ_j is unimportant, we can set it to be either 1 or ∞. The standard Newton's procedure can also be used initially to obtain estimates of Δ_j. Other update formulae for Δ_j can be used. For example, a formula $\Delta_j = \max(2|\Delta w_j|, \Delta_j/2)$ gives a method that is very similar to a standard trust-region update. Theoretically, this trust-region-like update is appealing because $\Delta_j \to 0$ as $k \to \infty$, and hence each inner column-wise update asymptotically behaves like Newton's method.

The above algorithms are written in the form of fixed iteration algorithms where K is picked beforehand (usually 100 is sufficient for text categorization purposes). It is not difficult to impose an explicit stopping criterion which checks the convergence of the parameter by some measure after each iteration. As an example, for text categorization applications, if the change of $\sum_i |\Delta r_i|$ is less than 0.1 percent of $1 + \sum_i |r_i|$, we can terminate the algorithm.

3.2 A row relaxation algorithm

We introduce a dual form of (12), and suggest a class of numerical algorithms to solve the dual formulation. By using the dual formulation, we can obtain a representation of w in a so-called Reproducing Kernel Hilbert Space (RKHS) [22]. Such a representation allows a linear classifier to learn non-linear functions in the original input space, and thus is considered a major advantage of kernel methods including recently popularized support vector machines and Gaussian processes (dual ridge regression). Although we have mentioned in Section 2 that we are unable to observe any statistically significant evidence that kernel methods are superior for general text-categorization tasks, they could be helpful for special problems. It is thus useful to know that methods such as logistic regression and modified least squares share the same advantage of SVM in terms of kernel representation.

To proceed, we introduce a dual form of (12) by considering an auxiliary variable ζ_i for each data x_i:

$$(\hat{w}, \hat{\zeta}) = \arg\inf_w \sup_\zeta \frac{1}{n} \sum_{i=1}^{n} [-k(\zeta_i) + \zeta_i w^T x_i y_i] + \lambda w^2, \tag{14}$$

where $k(\cdot)$ is the Legendre transform of $f(\cdot)$: $k(v) = \sup_u (uv - f(u))$. It is well known that k is convex. By switching the order of $\inf_w$ and $\sup_\zeta$, which is valid for the above minimax convex programming problem (a proof is given in Appendix A), we obtain

$$\hat{\zeta} = \arg\sup_\zeta \frac{1}{n} \sum_{i=1}^{n} [-k(\zeta_i) + \zeta_i w^T x_i y_i] + \lambda w^2, \tag{15}$$

where w is minimized at $w = -\frac{1}{2\lambda n} \sum_i \zeta_i x_i y_i$. Substituting into the above equation, we obtain

$$\hat{\zeta} = \arg\inf_\zeta \sum_{i=1}^{n} k(\zeta_i) + \frac{1}{4\lambda n} (\sum_{i=1}^{n} \zeta_i x_i y_i)^2, \tag{16}$$

and

$$\hat{w} = -\frac{1}{2\lambda n} \sum_i \hat{\zeta}_i x_i y_i. \tag{17}$$

Similar to Algorithm 1 which solves the primal problem (12), the following generic relaxation algorithm solves the dual problem (16):

Algorithm 5 *(Dual Gauss-Seidel)*

let $\zeta = 0$ and $v_j = 0$ for $j = 1, \ldots, d$
for $k = 1, 2, \ldots$
  for $i = 1, \ldots, n$
    find $\Delta\zeta_i$ by approximately minimizing
      $k(\zeta_i + \Delta\zeta_i) + \frac{1}{4\lambda n}(2\Delta\zeta_i v^T x_i y_i + \Delta\zeta_i^2 x_i^2)$ $(**)$
    update v: $v_j = v_j + \Delta\zeta_i x_{ij} y_i$ $(j = 1, \ldots, d)$
    update ζ: $\zeta_i = \zeta_i + \Delta\zeta_i$
  end
end
let $w = -\frac{1}{2\lambda n} v$.

The dual formulation (16) also leads to kernel methods which embed the feature vector x into a Hilbert space. Instead of computing w in the associated Hilbert space which is computationally infeasible, we may choose to keep the dual variable ζ. In order to compute $v^T x_i = \sum_k \zeta_k x_k^T x_i y_i$ in $(**)$, we replace the inner product $x_k^T x_i$ by a kernel function $K(x_k, x_i)$ just like in a kernel SVM [21].

An important implementation issue for Algorithm 5 is that the data ordering can have a significant effect on the rate of convergence (the feature ordering does not appear to have a very noticeable effect on Algorithm 5). More specifically, if we order the data points such that members in each class are grouped together, then we may experience a very slow convergence. On the other hand, the dual algorithm appears to work very well with a random data ordering. Intuitively, this phenomenon is not too surprising, since if class members are grouped together, then as we go through the data in one class, we tend to over-fit the portion of data until we encounter some data in the other class. However, we still do not have a deep understanding of the data ordering issue. Future study may potentially lead to a systematic method of data re-ordering to accelerate convergence.

An attractive property of the dual formulation is that after the computation of x_i^2 (which can also be pre-computed and stored in memory) and $v^T x_i y_i$, the update step $(**)$ becomes a small system which can be efficiently solved in constant time (approximately to any pre-determined numerical accuracy) by a line-search algorithm. However, the rate of convergence of a dual (row-wise) algorithm can be different than that of a primal (column-wise) algorithm. In general, for $f(\cdot)$ in (12) that is sufficiently smooth, we have not found any sufficient evidence that the dual algorithm is more efficient.

On the other hand, the dual formulation is computationally very useful for an SVM due to the non-smooth loss function in its primal form. For the standard SVM (8), a good method for text categorization is SMO [4], which is based on the dual formulation described in chapter 12 of [19]. One complication of the dual SVM (8) which employs an explicit threshold θ is the associated equality constraint $\sum_i \zeta_i y_i = 0$ on the dual variable, which has to be specially treated in a numerical optimization routine. For example, in SMO, this is done by modifying a pair of dual variable components simultaneously at each step to enforce the constraint $\sum_i \zeta_i y_i = 0$. Heuristics are used to select pairs of dual variables to optimize. Our experience with SMO indicates that a non-negligible amount of time is spent on the heuristic dual pair selection procedure, rather than numerical optimization.

A simple remedy of the problem is to set $\theta = 0$ in a standard SVM and append a constant feature as we do for all other algorithms in this paper. This modification leads to a variant of SVM without the dual equality constraint. Specifically, its dual formulation is given by (16) with $k(z)$ defined as

$$k(z) = \begin{cases} z & -1 \le z \le 0, \\ +\infty & \text{otherwise.} \end{cases}$$

The $+\infty$ value is effectively a simple constraint for each dual variable: $-1 \le \zeta_i \le 0$. Algorithm 5 can be applied to this formulation with $(**)$ given by the following update:

$$\Delta\zeta_i = -\max(\zeta_i, \min(1 + \zeta_i, \eta \frac{2\lambda n + v^T x_i y_i}{x_i^2})). \tag{18}$$

This procedure was also mentioned by Platt in Chapter 12 of [19], although he dismissed it as inferior to a standard SVM. We shall later present experimental evidence contrary to Platt's opinion. In fact, the method has also been successfully employed in [11]. To distinguish this procedure from a standard SVM, we shall call it modified SVM in this paper. The update (18) with $\eta = 1$ corresponds to the exact minimization of the dual objective functional in $(**)$. However, in practice, we observe that it is useful to use a smaller update step with $0 < \eta < 1$.

For the modified least squares formulation (10), $k(z)$ is defined only for $z \leq 0$ ($k(z) = +\infty$ for $z > 0$): $k(z) = z^2/4 + z$. We thus obtain an instance of Algorithm 5 with $(**)$ replaced by the following the closed form solution of $(**)$:

$$\Delta\zeta_i = \min(-\zeta_i, -\frac{(2+\zeta_i)\lambda n + v^T x_i y_i}{\lambda n + x_i^2}).$$

For logistic regression (6), the Legendre dual of f is $k(z) = -z\ln(-z) + (1+z)\ln(1+z)$ where $k(\cdot)$ is defined only in $[-1, 0]$. The solution of $(**)$ can be obtained by solving the following equation:

$$\ln \frac{1+\zeta_i+\Delta\zeta_i}{-(\zeta_i+\Delta\zeta_i)} + \frac{1}{2\lambda n}(v^T x_i y_i + x_i^2 \Delta\zeta_i) = 0.$$

4 Experimental Results

In this section, we compare the following methods already discussed in the paper: the regularized linear least squares fit method (4) denoted as *Lin Reg*, the modified least squares formulation (10) denoted as *Mod Least Squares*, the regularized logistic regression (6) denoted as *Logistic Reg*, the (linear) support vector machine (8) denoted as *SVM*, and the modified SVM corresponding to (18) denoted as *Mod SVM*. For comparison purposes, we also include results of Naive Bayes [14] as a baseline method.

4.1 Some Implementation issues

A number of feature selection methods for text categorization were compared in [27]. In our experiments, we employ a method similar to the information gain (IG) approach described in [27]. However, we replace the entropy scoring in the IG criterion by Gini index. The reason for our choice is that we do not remove stopwords in the preprocessing step. In our experience, Gini index seems to remove stopwords more effectively than the entropy function. However, the difference is rather small. We didn't experiment with other methods described in [27].

In our experiments, we select 10000 features as the input to the algorithms, ignoring features occurring less than 3 times as suggested in [12]. Each feature is a binary word occurrence $\in \{0, 1\}$, indicating whether the word occurs or not in the document. A word in the title is regarded as a different feature than the same word in the body. More complicated representations such as TFIDF weighting schemes [12] can also be employed. In this work, we didn't compare the effectiveness of these different approaches.

Our implementation of Naive Bayes is based on the multinomial model formulation described in [14]. However, we replace the Laplacian smoothing where each probability count is increased by 1, by a λ-smoothing where each probability count is increased by λ. This formulation can be considered as a regularization method to avoid the zero-denominator problem. In our experience, the replacement of Laplacian smoothing with $\lambda < 1$ smoothing makes a significant difference in practice. $\lambda = 1$ is almost never a good choice. For certain data such as IndustrySector (see below for description), the Laplacian smoothing gives a classification accuracy of below 60%, which is significantly worse than the 85% average reported in Table 5 with $\lambda = 10^{-2}$.

For methods described in this paper, our implementation is based on algorithms described in Section 3. The default number of iterations K is chosen to be 100. The algorithms can be further accelerated with feature selection inside the iterations of the algorithms: we can truncate small column weight variables to zero, and ignore the corresponding column iteration. In addition to computation speed up, this may also enhance the classification performance. *Lin Reg* (4) is solved by Algorithm 2, and *Logistic Reg* (6) is solved by Algorithm 4. For *Mod Least Squares* (10), we choose $c_k = \max(0, 1 - k/50)$ in Algorithm 3. The standard SVM (8) is solved by an implementation of the SMO algorithm ([19], chapter 12). The modified SVM is solved by Algorithm 5 with $(**)$ given by (18), where we choose $\eta = 0.1$.

On the Reuters dataset (see below), the regularization parameter λ for each algorithm is determined by a 5-fold cross-validation with the training set. We fix this parameter to be identical for all categories. We try values of λ at 10^{-k} with $k = 0, \ldots, 6$. The particular optimal choices of λ we obtain are summarized in Table 1. For other datasets, we simply use the λ computed on the Reuters dataset. Our experience indicates that the difference of using this choice and using the optimal value for each specific dataset is relatively small.

	Naive Bayes	Lin Reg	Mod Least Squares	Logistic Reg	SVM	Mod SVM
λ	10^{-2}	10^{-3}	10^{-3}	10^{-4}	10^{-3}	10^{-3}

Table 1: Values of the regularization parameter

Due to different formulations, termination criteria and parameter choices, it is difficult to make rigorous timing comparisons for different algorithms. However, to have a rough idea, in our implementation, *Naive Bayes* is typically an order of magnitude faster than the other algorithms. *Lin Reg* is slower than NB but faster than the others due to its simplicity. *Mod Least Squares* and *Mod SVM* have similar speed, which is about five minutes on Reuters using a Pentium 400 Mhz Linux PC with 10000 features. This timing, which includes feature selection, corresponds to approximately 2.5 seconds per category. Both logistic regression and the standard SVM using SMO are noticeably slower in our particular implementation.

4.2 Classification performance

In text categorization, it frequently happens that each category contains only a small percentage of documents. When this happens, the standard classification error measurement is often not very indicative, because one can get a low error rate simply by saying no item is in any class, which is not likely to result in a very useful categorizer. In text categorization, the standard performance measures of a classification method are the precision and recall instead of classification error:

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \times 100$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \times 100$$

If a classification algorithm contains a parameter that can be adjusted to facilitate a trade-off between precision and recall, then one can compute the break-even point (BEP), where precision equals recall, as an evaluation criterion for the performance of a classification algorithm [26]. Another widely used single number metric is the F_1 metric defined as the harmonic mean of the precision and the recall [24].

The standard dataset for comparing text categorization algorithms is the Reuters set of news stories, publicly available at

http://www.research.att.com/~lewis/reuters21578.html.

We use Reuters-21578 with ModApte split to partition the documents into training and validation sets. This dataset contains 118 non-empty classes (in addition, there are some documents that are not categorized), with 9603 documents in the training set and 3299 documents in the test set. The micro-averaged (that is, the precision and the recall are computed by using statistics summed over the confusion matrices of all classes) performances of the algorithms over all 118 classes are reported in Table 2. Each BEP is computed by first adjusting the linear decision threshold individually for each category to obtain a BEP for the category, and then micro-averaged over all 118 classes. Our SVM result is consistent with the previously reported results in [4, 12]. The Naive Bayes result is consistent with [4].

It is interestingly to note that for our particular experimental setup, we have obtained a very good performance from the least squares method. This can be partially explained by using the statistical learning theory argument of SVM in [21]. From the learning theory point of view, an SVM can perform well if there exists a linear separator w with a small 2-norm that separates in-class data from out-of-class data with a relative large margin: $w^T xy \geq 1$ shall be valid for most data. On the other hand, the SVM learning bounds require $\|x\|_2$ small for x, which means that $|w^T x| \leq \|w\|_2 \|x\|_2$ should also be small. Therefore the overall effect is that $w^T x$ is clustered around a positive point for in-class data and clustered around a negative point for out-of-class data. Since the least squares method computes a weight w so that $w^T x$ is clustered around 1 for in-class data and clustered around -1 for negative data, therefore it can perform reasonably well as long as an SVM is suitable for the problem.

In our experiments, we have noticed that, except for Naive Bayes, all methods find hyperplanes of comparable quality. This is reflected by their similar break-even points. However, we also found that the F_1 numbers with unmodified computed thresholds are usually suboptimal for all methods. We found this is because for many small categories in the Reuters dataset, the computed thresholds are biased towards causing fewer errors for the dominant out-of-class data — this effect can be compensated by lowering the computed thresholds. In our experiments, we adjust the threshold after the hyperplane is computed to compensate for this problem (we do this by finding a value that minimizes the training error). This phenomenon has also been observed in [16], where Platt found that fitting a sigmoid to the output of SVM can enhance the performance on Reuters — the effect of such a sigmoid fitting is equivalent to using a different threshold.

It is unclear what is the best procedure to adjust the threshold after the hyperplane is computed. This topic requires further investigation. Since the BEP metric is independent of different threshold choices, therefore it can be regarded as a pure measure of the quality of the computed hyperplane itself. It can also be regarded as a lower-bound estimate of the best F_1 metric we can obtain if we can find the "ideal" threshold. We include break-even points in Table 2. The results of modified least squares and naive Bayes indicate that a good choice of threshold can potentially lead to a better F_1 value (implying a smaller classification error) than the BEP value. On the other hand, logistic regression is more likely to end up with a suboptimal threshold, which is reflected by its much better BEP measure than its F_1 measure. In addition to micro-average results, as argued in [26], it is also useful to report macro-averaged F_1 values over categories, which correspond to (un-weighted) averages of F_1 values for each class. Figure 6 shows the macro-averaged F_1 values over top 5, 10, 20, 40, 60, 80, 100, and all 118 categories. We also include F_1 values for the 10 largest categories of the Reuters dataset in Table 3, where the best number for each category is highlighted.

	Naive Bayes	Lin Reg	Mod Least Squares	Logistic Reg	SVM	Mod SVM
precision	77.0	87.1	89.2	88.0	89.2	89.4
recall	76.9	84.9	85.3	84.9	84.0	83.7
F_1	77.0	86.0	87.2	86.4	86.5	86.5
BEP	75.8	86.3	86.9	86.9	86.5	86.7

Table 2: Binary classification performance on Reuters (all 118 classes)

	Naive Bayes	Lin Reg	Mod Least Squares	Logistic Reg	SVM	Mod SVM
earn	96.6	97.1	**98.4**	**98.4**	98.1	98.1
acq	91.7	93.2	**95.4**	95.2	95.3	94.5
money-fx	70.0	73.2	**76.0**	75.2	74.4	74.5
grain	76.6	**91.6**	90.3	88.4	89.6	90.6
crude	84.1	**86.2**	84.9	85.9	84.8	84.0
trade	52.3	70.8	**76.3**	72.9	73.4	74.8
interest	68.2	75.2	75.7	**78.1**	75.9	74.7
ship	76.4	80.7	83.6	81.9	82.4	**83.8**
wheat	58.1	**89.6**	88.5	88.2	88.9	**89.6**
corn	52.4	**89.3**	88.1	88.7	86.2	86.7

Table 3: F_1 performance on Reuters (top 10 classes)

Figure 7 shows the break-even performance of logistic regression at different feature sizes. As we can see, the performance improves as the number of features increases. This suggests that excessive feature selection is not beneficial for regularized linear classification methods. However, if we eliminate the feature selection step so that all stop-words (as well as noisy features occurring less than 3 times) are present, then the performance degrades to a break-even point of about 85. Note that in this study, we have used the BEP metric to avoid complications of threshold adjustment discussed earlier.

Figure 8 shows the break-even performance of logistic regression with different values of regularization parameter λ. In theory, the performance of logistic regression is unpredictable as $\lambda \to 0$. This is because if the problem is linearly separable, then the solution of non-regularized logistic regression is ill-defined: let $\hat{w}$ linearly separate the data so that $\hat{w}^T x_i y_i > 0$ for all i, then for a positive scaling parameter γ, it is easy to verify that the logistic objective function $\sum_i \ln(1 + \exp(-\gamma \hat{w}^T x_i y_i))$ decreases to zero as $\gamma \to +\infty$. This implies that the optimal solution of the non-regularized logistic regression cannot be finite (i.e. ill-defined) if the data is linearly separable (this situation happens in Reuters). However, Figure 8 implies that the performance degradation of logistic regression as $\lambda \to 0$ is not severe. We do not have a definitive explanation for this phenomenon. However, we conjecture that it could be due to the fact that we use a relatively small number of iterations, and at each iteration the update step size is small, with the consequence that the weight vector does not get large.

Although when $\lambda \to 0$, an SVM is well behaved for linearly separable data, it is ill-behaved for linearly non-separable data. However, interestingly, even when $\lambda \to 0$, the dual algorithms are relatively well-behaved if we use a fixed number of iterations. For *Mod SVM* and *Mod Least Squares*, the micro-averaged BEPs are reduced to about 85 as $\lambda \to 0$, when we fix the number of iterations to be 100 in the dual algorithms. The reason for this good behavior is due to the slow convergence of the dual algorithms when $\lambda \to 0$. For example, in (18), when $\lambda \to 0$, $\Delta\zeta_i$ becomes proportionally small, so that ζ/λ does not become large within a reasonable number of iterations. However, it can be shown that for non-separable data with $\lambda \to 0$, the optimal value of ζ/λ cannot be finite. Therefore in this case, the slow convergence of a dual algorithm avoids the bad scenario of infinite ζ/λ. Consequently, the primal algorithm for solving (10) leads to a poorer result (around 80 BEP) than that of the corresponding dual algorithm (around 85 BEP), even though the primal algorithm converges faster (which is verified by comparing the objective function values in (10) with the computed weights). In our experience, the least squares method degrades more severely in the limit of $\lambda \to 0$, with a micro-averaged break-even point of around 70. However, if we use only 100 features instead of 10000 features, then the least squares method achieves a break-even point of about 83 in the limit of $\lambda \to 0$.

In Table 4, we give comparative results on another data set (AS400) which consists of records of call center inquiries from IBM customers. This data, categorized into 36 classes, is partitioned into a training set of 59557 items and a test set of 14235 items. The characteristics of this data is rather different than that of the Reuters. The BEP performances of the algorithms are more or less consistent. However, the F_1 metric again demonstrates that the choice of linear threshold can significantly affect the performance of an algorithm, even when the algorithm has a good break-even point.

	Naive Bayes	Lin Reg	Mod Least Squares	Logistic Reg	SVM	Mod SVM
precision	66.1	78.5	77.7	76.3	78.9	78.7
recall	74.9	64.0	70.9	71.4	63.8	63.6
F_1	70.2	70.5	74.1	73.8	70.6	70.4
BEP	69.1	71.7	74.4	73.8	73.9	73.8

Table 4: Binary classification performance on AS400

Even though for some text categorization problems such as Reuters, each document can be multiply classified, in many other applications, documents are singly classified. In this case, one is usually interested in treating the problem directly as a single multi-class problem, rather than many separate binary-class problems. The overall classification accuracy defined as the percentage of correctly predicted category, is a useful performance measure for this type of problems. We compare the algorithms on the following datasets. The first three datasets are available at *http://www.cs.cmu.edu/ TextLearning/datasets.html*.

- 20NewsGroup, which is a collection of 20,000 messages, from 20 different newsgroups, with one thousand messages from each newsgroup.
- IndustrySector, which is a collection of about 10000 web pages of companies from various economic sectors. There are 105 categories.
- WebKB, which contains about 8000 web pages collected from computer science departments of various universities in January 1997. We use the version with the following seven categories: "student", "faculty", "staff", "department", "course", "project", and "other".
- GRANT, which contains about 5000 grant proposals. There are 42 categories corresponding to areas of the proposals.
- IBMweb, which contains about 7000 web pages collected from www.ibm.com, categorized into 42 classes.

We use the same choices for the regularization parameter λ as listed in Table 1 for the various algorithms. We observe the performance difference of this default choice is close to that of the optimal parameter for the above datasets. This implies that parameters we use can be reasonably good across a variety of datasets, which can partially eliminate the necessity of cross-validation parameter selection, if such a procedure is undesirable due to the extra computational cost. Unlike the binary classification case, we do not perform any threshold adjustment in these experiments. To obtain a multi-class categorizer, we compute a linear classifier with weight w_l for each category l by treating it as a binary problem; we then predict the class of a data-point x as the label l with the largest value of $w_l^T x$. For Naive Bayes, this gives a method we call *Naive Bayes-2*. For comparison purposes, we have also implemented a direct multi-class Naive Bayes method as described in [14], which we call *Naive Bayes-1*. Although there are studies suggesting certain coding schemes (such as error-correcting output code) to convert a binary classifier into a multi-class categorizer, we have found no statistically significant improvement by using any of such coding schemes in our implementation.

We select 10000 features for all algorithms except *Naive Bayes-1*. No feature selection is performed for *Naive Bayes-1*. The consistent improvement of *Naive Bayes-2* over *Naive Bayes-1* is surprising. We observe that the improvement is not entirely due to feature selection. However, we still do not have a good explanation for this phenomenon. For each dataset, we generate five random training-test splits, with training set size four times that of test set size. Table 5 reports classification accuracies of the algorithms with the five random splits for each dataset. The result is in a format of *mean* $\pm$ *standard deviation*. The relative performances of the algorithms are rather consistent (except logistic regression is worse than linear regression on IndustrySector).

In conclusion, Naive Bayes is consistently worse than the other methods. There also seems to be a statistically significant difference to indicate that linear regression is worse than the remaining algorithms (except Naive Bayes). The differences among other algorithms are not significant enough to draw any firm conclusion. We also observe that the differences among the algorithms are more significant on some datasets than others. It is unclear how to characterize a dataset in order to predict how significant are the performances of different algorithms.

method	20NewsGroup	IndustrySector	WebKB	GRANT	IBMweb
Naive Bayes-1	89.7 ± 0.5	84.8 ± 0.5	65.0 ± 0.7	59.4 ± 1.9	77.2 ± 0.4
Naive Bayes-2	92.4 ± 0.4	91.0 ± 0.6	68.7 ± 1.1	64.2 ± 1.3	79.6 ± 0.7
Lin Reg	94.1 ± 0.2	93.4 ± 0.5	83.8 ± 0.3	67.0 ± 0.8	85.7 ± 0.5
Mod Least Squares	94.5 ± 0.3	93.6 ± 0.4	88.7 ± 0.5	70.2 ± 1.2	86.2 ± 0.7
Logistic Reg	94.4 ± 0.3	92.3 ± 0.9	89.0 ± 0.5	70.6 ± 1.2	86.2 ± 0.6
SVM	94.8 ± 0.3	93.6 ± 0.5	88.4 ± 0.5	70.0 ± 1.2	86.1 ± 0.4
Mod SVM	94.7 ± 0.3	93.6 ± 0.4	88.5 ± 0.5	69.8 ± 1.2	85.8 ± 0.7

Table 5: Multi-class classification accuracy

5 Summary

In this paper, we have applied the logistic regression model and a modification of the least squares fit method for text categorization. We have also compared them with other linear classification methods such as support vector machines.

Statistical properties of different methods were discussed in Section 2, providing useful insights for the various methods. We argued that the text categorization application requires numerical iterative algorithms that are suitable for solving large sparse systems associated with the proposed formulations. Numerical issues were discussed in Section 3. Specifically, we introduced a column-wise relaxation algorithm and a row-wise relaxation algorithm to solve these large sparse systems. In our experience, the proposed algorithms are rather efficient. However, it may be possible to further enhance these algorithms, for example, by using a quasi-Newton acceleration on top of the iterations. The insufficient smoothness of some formulations described in this paper may cause trouble in such a quasi-Newton enhancement. It is therefore still an open question on how significant an improvement we can obtain.

Experimental comparisons of the algorithms on a number of datasets were reported. Naive Bayes is consistently worse than the other algorithms. Although the regularized linear least squares method has a performance very close to the state of the art, this (relatively small) difference appears to be statistically significant. We have also demonstrated that an appropriately implemented regularized logistic regression can perform as well as an SVM, which is regarded as the state of the art in text categorization. Furthermore, logistic regression has the advantage of yielding a probability model, which can be useful in many applications.

Finally, there are a number of open issues revealed in our study. One interesting problem is how to find a good threshold given a computed hyperplane. Another interesting issue is the effect of changing the regularization parameter λ, as well as a simple method to select a good value of λ given a dataset. We have only briefly touched this issue in this paper. Behaviors of different algorithms as $\lambda \to 0$ were examined in Section 4. In addition, our experience suggests that fixing λ at a default value leads to reasonable performance across a variety of datasets (compared with the baseline Naive Bayes). However, it would still be useful to carefully investigate related issues such as the impact of certain dataset characteristics (e.g. the training sample size, or the average length of a document, etc) on the choice of λ.

The feature selection issue has also been investigated in our experiments. We showed that a moderate amount of feature selection is useful. Although, this issue has been discussed in many previous works, further studies on the effect of feature selection on regularization methods, such as those discussed in the paper, would be valuable.

We have also observed that the performance differences among the algorithms are data dependent. It would be useful to find dataset characteristics to explain this variation. Such a characterization could provide valuable insights on how different algorithms behave and why they perform well or poorly with a particular data.

A Validity of the dual formulation

We would like to show that it is valid to interchange the order of $\inf_w$ and $\sup_\zeta$ in (14). Let

$$L(w,\zeta) = \frac{1}{n}\sum_{i=1}^{n}[-k(\zeta_i) + \zeta_i(w^T x_i y_i)] + \lambda w^2.$$

Then we need to show that there exists $(\hat{w},\hat{\zeta})$ such that

$$L(\hat{w},\hat{\zeta}) = \inf_w \sup_\zeta L(w,\zeta) = \sup_\zeta \inf_w L(w,\zeta). \tag{19}$$

It is well-known (for example, see [18]) that the duality gap G is non-negative, where

$$G = \inf_w \sup_\zeta L(w,\zeta) - \sup_\zeta \inf_w L(w,\zeta).$$

Furthermore, equation (19) has a solution if $G = 0$. We need to find $(\hat{w},\hat{\zeta})$ such that

$$L(\hat{w},\hat{\zeta}) = \inf_w L(w,\hat{\zeta}) = \sup_\zeta L(\hat{w},\zeta). \tag{20}$$

In this case, $(\hat{w},\hat{\zeta})$ is called a saddle point.

To construct a saddle point, we consider $\hat{w}$ that minimizes the primal problem, which satisfies the following estimation equation:

$$\frac{1}{n}\sum_{i=1}^{n} f'(\hat{w}^T x_i y_i) x_i y_i + 2\lambda \hat{w} = 0.$$

Now, let $\hat{\zeta}_i = f'(\hat{w}^T x_i y_i)$, we obtain:

$$\hat{w} = -\frac{1}{2\lambda n}\sum_{i=1}^{n} \hat{\zeta}_i x_i y_i.$$

This implies that $\hat{w}$ achieves the minimum of

$$\frac{1}{n} w^T \sum_{i=1}^{n} \hat{\zeta}_i x_i + \lambda w^2.$$

That is,

$$L(\hat{w}, \hat{\zeta}) = \inf_w L(w, \hat{\zeta}). \tag{21}$$

Also by the definition of $\hat{\zeta}_i$ as the derivative of $f(\cdot)$ at $\hat{w}^T x_i y_i$, we know that $\hat{\zeta}_i$ achieves the maximum of

$$\zeta_i w^T x_i y_i - k(\zeta_i).$$

That is,

$$L(\hat{w}, \hat{\zeta}) = \sup_\zeta L(\hat{w}, \zeta). \tag{22}$$

Combining (21) and (22), we obtain (19), which implies the interchangeability of the order of $\inf_w$ and $\sup_\zeta$ in (14).

References

[1] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12:233–251, 1994.

[2] W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *SGIR 92*, pages 198–210, 1992.

[3] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[4] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.

[5] N. Fuhr and U. Pfeifer. Combining model-oriented and description-oriented approaches for probabilistic indexing. In *SIGIR 91*, pages 46–56, 1991.

[6] F. C. Gey. Inferring probability of relevance using the method of logistic regression. In *SIGIR 94*, pages 222–231, 1994.

[7] G. Golub and C. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[8] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. Chapman and Hall Ltd., London, 1990.

[9] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[10] D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Symposium on Document Analysis and Information Retrieval*, pages 301–315, 1995.

[11] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7:95–114, 2000.

[12] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learing, ECML-98*, pages 137–142, 1998.

[13] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR 94*, pages 3–12, 1994.

[14] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[15] M. Minsky and S. Papert. *Perceptrons*. MIT press, Cambridge, MA, expanded edition edition, 1990.

[16] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[17] B. Ripley. *Pattern recognition and neural networks*. Cambridge university press, 1996.

[18] R. T. Rockafellar. *Convex analysis*. Princeton University Press, Princeton, NJ, 1970.

[19] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods : Support Vector Learning*. The MIT press, 1999.

[20] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *SIGIR 95*, pages 229 – 237, 1995.

[21] V. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.

[22] G. Wahba. Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods : Support Vector Learning*, chapter 6. The MIT press, 1999.

[23] S. Weiss, C. Apte, F. Damerau, D. Johnson, F. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14:63–69, 1999.

[24] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, 1:69–90, 1999.

[25] Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12:252–277, 1994.

[26] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR 99*, pages 42–49, 1999.

[27] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

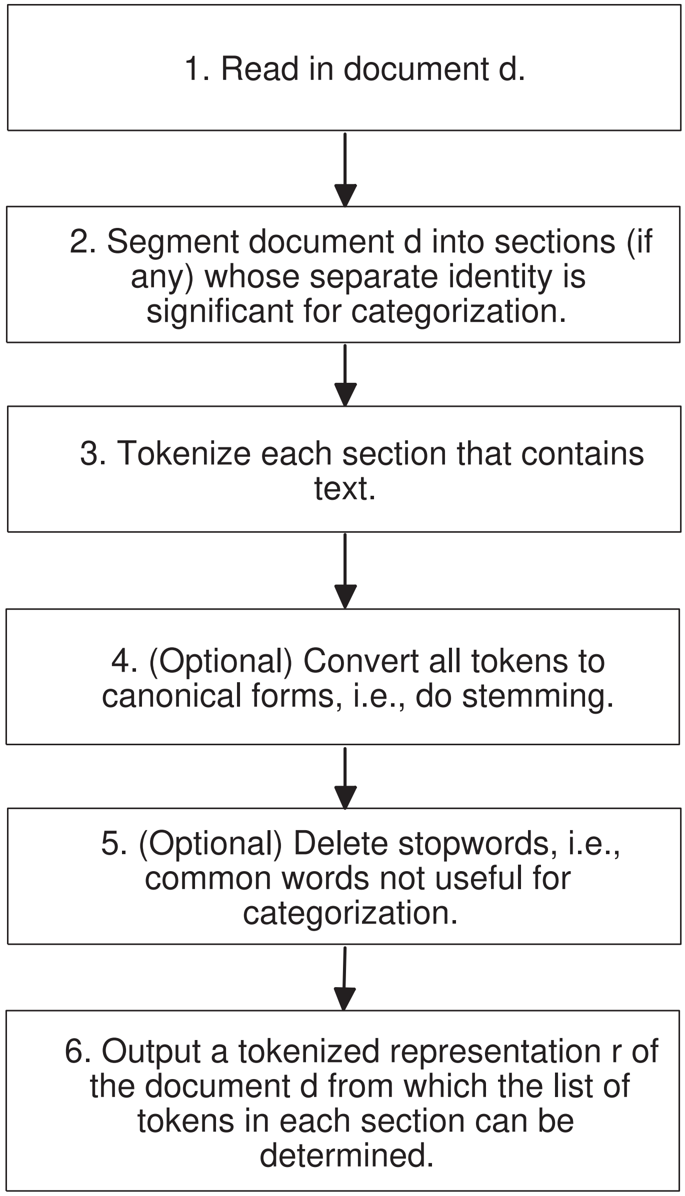

Figure 1: Document tokenization.

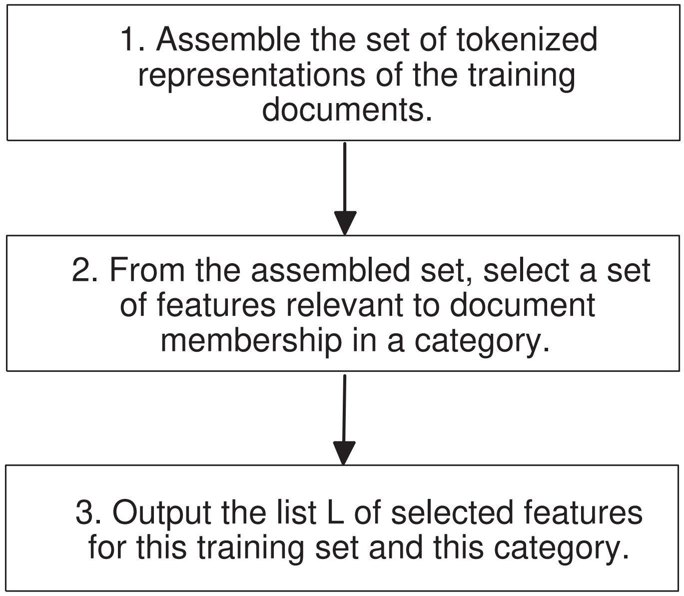

Figure 2: Feature selection.

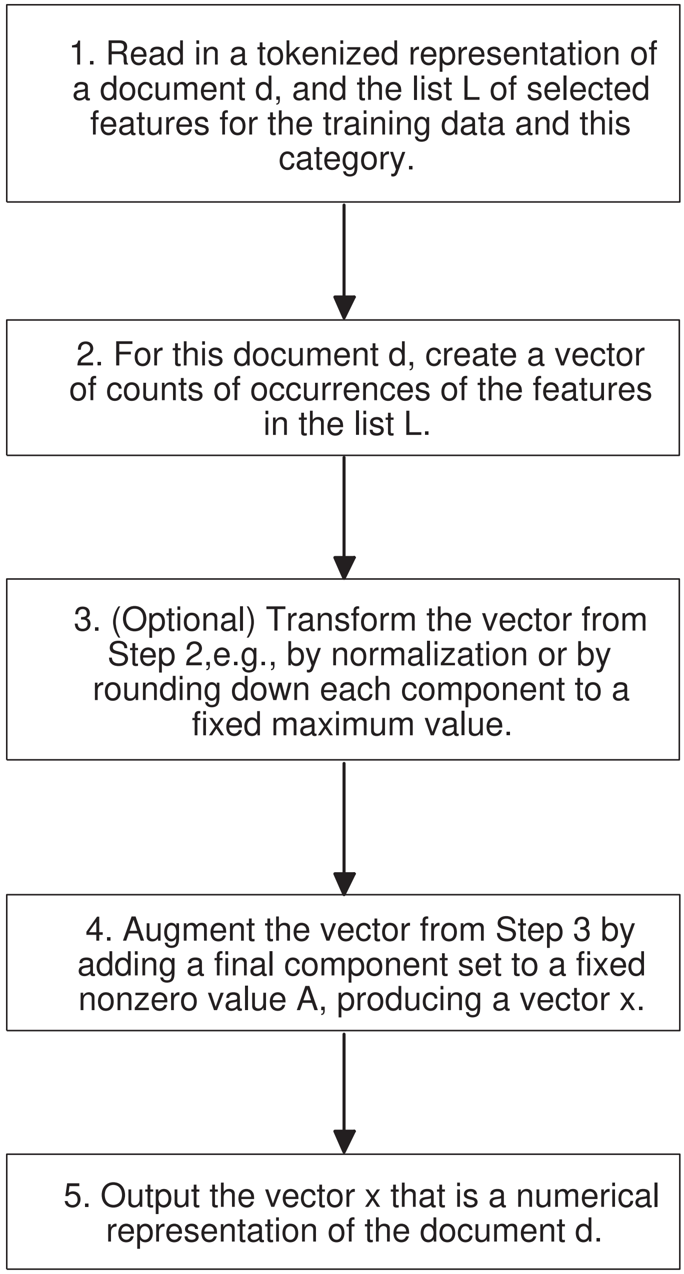

Figure 3: Creating a vector to represent a document using the selected features.

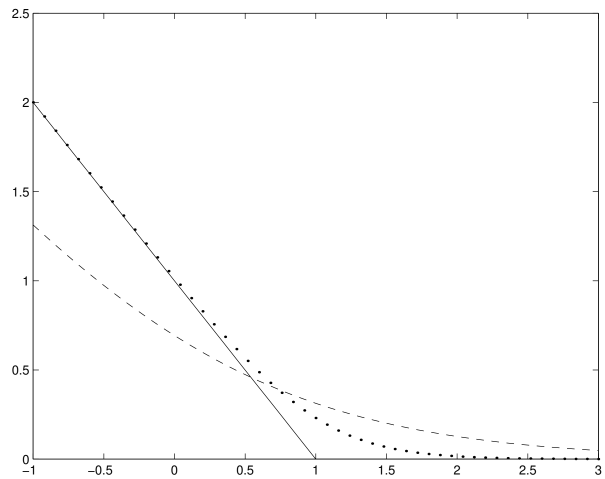

Figure 4: loss functions: g(z) = 'solid'; $\ln(1 + \exp(-z))$ = 'dash'; $\frac{1}{3}\ln(1 + \exp(3(1 - z)))$ = 'dotted'.

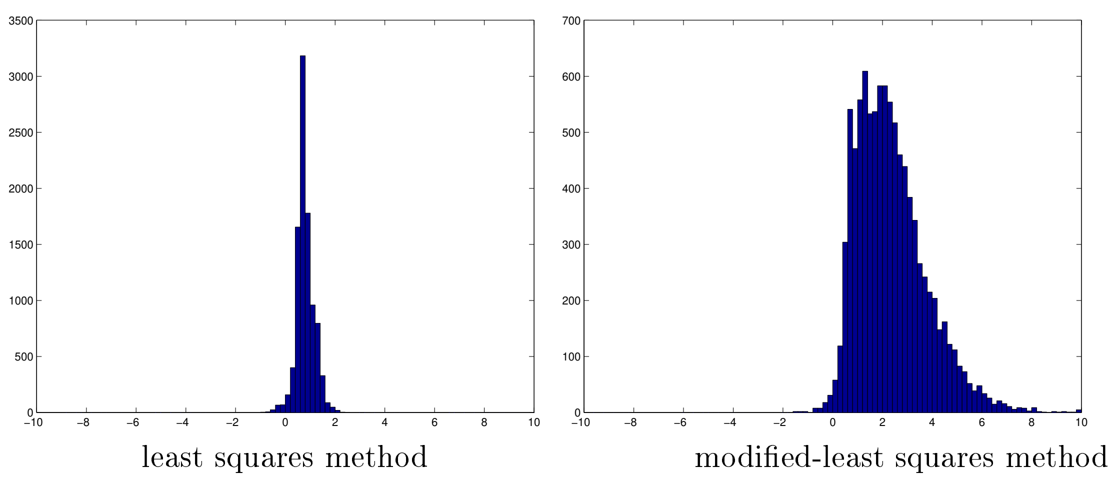

Figure 5: projected histogram of $\hat{w}^T xy$

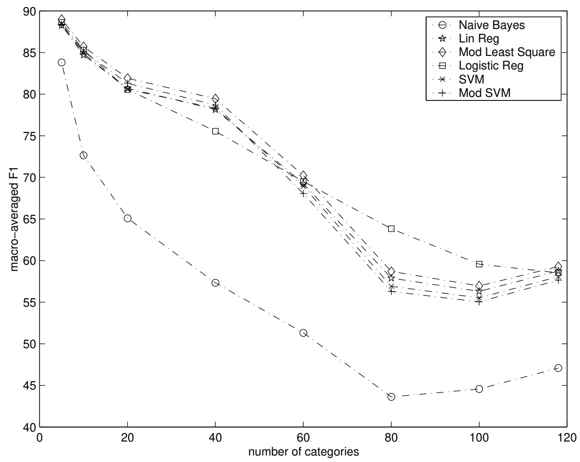

Figure 6: Macro averaged F_1 measure

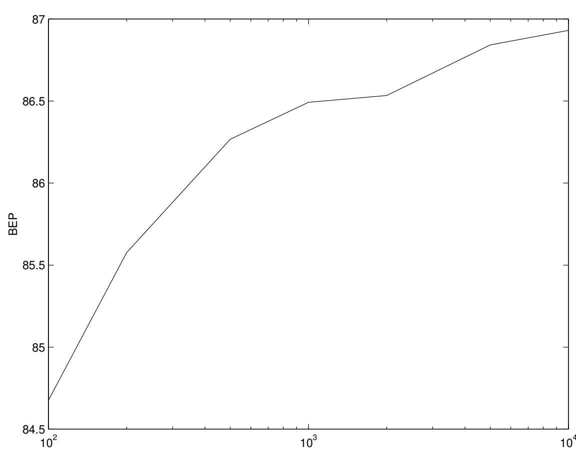

Figure 7: Break-even performance of logistic regression against feature size

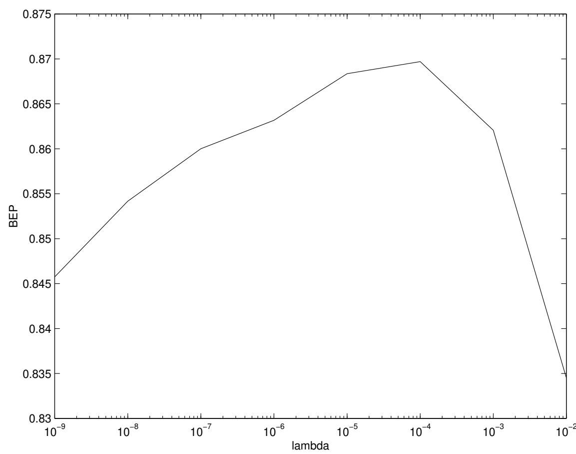

Figure 8: Break-even performance of logistic regression against λ