A Randomized Algorithm for Large Scale Almost Linearly Separable Classification Problems with Applications to Text

Krishnan Suresh Kumar Chiranjib Bhattacharyya
Ramesh Hariharan

IISc-CSA-TR-2006-9
http://archive.csa.iisc.ernet.in/TR/2006/9/

Computer Science and Automation
Indian Institute of Science, India

September 2006

A Randomized Algorithm for Large Scale Almost Linearly Separable Classification Problems with Applications to Text

Krishnan Suresh Kumar* Chiranjib Bhattacharya† Ramesh Hariharan‡

September 2006

Abstract

This paper addresses the problem of making Support Vector Machine(SVM) and Linear Programming(LP) formulations scalable for learning linear classifiers. The key contribution of the paper is to show that by using random projections the minimal number of support vectors of linearly separable classification problems is given by $O(\log n)$ with high probability. This fact is used to derive a sampling based scheme, motivated from randomized algorithms, which attains scalability by considering $O(\log n)$ points at a time instead of the entire dataset. The algorithm applies to high dimensional *almost linearly* separable datasets. If on removal of properly chosen $O(\log n)$ points the data becomes linearly separable then it is called almost linearly separable. Many real life text datasets, for example REUTERS-21578 and RCV1, exhibit this property. Experiments done on RCV1 and REUTERS data show that the algorithm can be used to scale up state of the art SVM solvers in terms of memory required and execution time without loss in accuracy. It is to be noted that the algorithm presented here nicely complements existing large scale classification approaches as it can be used to scale up any SVM or LP solver.

*CSA, IISc, Bangalore, India `krishi@csa.iisc.ernet.in`
†CSA, IISc, Bangalore, India `chiru@csa.iisc.ernet.in`
‡Strand Genomics, Bangalore, India `ramesh@strandls.com`

1 Introduction

Classification of text documents is an important task in the modern era of Internet. A text document is often represented by a feature vector which stores the statistics of words and phrases which occur in that document [Sebastiani, 2002]. Using such a representation each text document can be perceived as a point in a high dimension Euclidean space. In recent times Support Vector Machines(SVM) have emerged as a useful tool for classifying points in a Euclidean space. Existing SVM tools are still not capable of handling real world problems which involve many thousands of features and large number of examples.

Consider the RCV1 [Lewis et al., 2004] text dataset which has $\sim 47,000$ features and $\sim 800,000$ examples($\sim$ 2.2GB). State of the art SVM solvers like SVMLight [Joachims, 1999] and LibSVM [Chang and Lin, 2001] do not perform well on such large scale datasets. Most SVM solvers consider the entire dataset in one go thus incurring huge computation and memory overheads. To alleviate this problem, state of the art solvers have efficient heuristics for selecting a small subset of training data points on which most of the computation is done and then they iterate over different subsets. This is known as *decomposition*. Other strategies which are not based on decomposition like ProximalSVM [Fung and Mangasarian, 2001] and L2-SVM-MFN [Keerthi and DeCoste, 2005] are attractive alternatives. These methods focus on improving the speed but do not mention anything about their scalability in terms of memory. A few feature selection algorithms have also been proposed to make the solvers more scalable. Dunja Mladeni and Milic-Frayling [Sept 2002] use a SVM linear classifier to perform feature selection and make SVM algorithms scalable for text applications.

In this work, using a mix of random projections and randomized algorithms we propose an algorithm which iterates over small subsets of the data(document selection) and efficiently solves the SVM problem with high probability. The algorithm presented here is not a SVM solver; instead it can be used to *scale any SVM algorithm in terms of memory usage* and as a consequence of this also achieves improved execution times. Thus work complements most of the other work on large scale SVM training which focus on improving execution time. Hence the algorithm presented here can be used along with any such SVM solver, thus making the latter even more efficient.

Given the dataset $D = \{(x_i, y_i)\}, i = 1 \ldots n$ and $y_i = \{+1, -1\}$, where $x_i \in R^d$ are data points and y_i specify the class labels, the problem of learning the classifier, $y = \text{sign}(w^T x + b)$, can be narrowed down to computing $\{w, b\}$ such that it has good generalization ability. The SVM formulation, which will be called $C - SVM$, for determining $\{w, b\}$ is given by [Vapnik, 1995,

Cortes and Vapnik, 1995]:
C-SVM-1:

$$\begin{array}{lcl} Minimize_{(w,b,\xi)} \ \frac{1}{2}||w||^2 & +C\sum_{i=1}^{n}\xi_i & \\ Subject\ to: \ y_i(w \cdot x_i + b) & \geq & 1-\xi_i, \ i=1\ldots n \\ \xi_i & \geq & 0, \ i=1\ldots n \end{array}$$

At optimality w is given by

$$w = \sum_{i:\alpha_i>0} \alpha_i y_i x_i, \ \ 0 \leq \alpha_i \leq C$$

The set $S = \{x_i|\alpha_i > 0\}$ are called *Support vectors*. Note that S completely determines the solution of $C - SVM$ [Burges, 1998]. The set S may not be unique, though w is. Define a parameter, Δ, to be the minimum cardinality over all S. See that Δ does not change with number of examples (n) and is often much less than n. The key idea in this paper is to iterate over subsets of size more than Δ which may contain S by random sampling thus attaining scalability.

This idea was first proposed in Balcazar et al. [2001a,b]. More generally the $C - SVM$ problem can be seen as an instance of *Abstract optimization problem*(AOP) [Gartner, 1992, Balcazar et al., 2001a,b]. An AOP can be solved by a randomized algorithm by selecting subsets of size greater than the *combinatorial dimension* of the problem [Gartner, 1992]. In some sense the combinatorial dimension captures the notion of number of free variables. For SVM Δ is the combinatorial dimension of the problem [Balcazar et al., 2001a,b]. Apriori the value of Δ is not known but for linearly separable classification problems it can be shown that $2 \leq \Delta \leq d + 1$ and Balcazar et al. [2001a,b] propose to iterate over subsets of size proportional to d^2. Clearly then this algorithm is of limited importance to high dimensional datasets like text, where $d \sim 47,000$, and $n > d > O(\log n)$.

The main contribution of this paper is to show that for well separated datasets, using random projections one can estimate Δ as $O(\log n)$. As a consequence the subset size over which one iterates reduces considerably from d^2 to $\log^2 n$. Using further heuristics we empirically show that the sample size can be reduced to $\log n$. The theory can be extended to *almost linearly separable* datasets which are essentially datasets which become linearly separable when a small number of properly chosen data points are deleted from the dataset, for example, in RCVI dataset, category GSPO becomes separable when 0.1% of datapoints are removed from them. Formally a dataset will be called *almost linearly separable* if after removal of $O(\frac{\log n}{n})$ fraction of data points the problem becomes separable. In such cases also $\Delta = O(\log n)$ turns out to be a good estimate. The idea also applies to Linear Programming(LP) formulation making them suitable for use for text classification. State of the art LP Solvers like LPSolve cannot handle large datasets like RCV1. We show how these solvers can be made scalable using the algorithm. For exact statements please see Section 3. The main advantage of this work is one can take any SVM solver or LP solver and make it scalable, by using it as a black box.

The paper is organized as follows. In the next section we briefly discuss random sampling algorithm as in Balcazar et al. [2001a,b]. Random projections are also reviewed in Section 2. Our contributions are in Section 3. In Section 4 we discuss empirical evaluation with state of the art solvers.

2 Past Work

We first look at a few algorithms which focus on large scale classification. Fung and Mangasarian [2001] presented a SVM formulation called Proximal SVM in which the objective is a non linear least squares function and the inequality constraints are replaced by a system of equations. Finding the best separating hyperplane now involves solving this system of equations. This is done by inverting a $d \times d$ matrix, as a result of which the method is not feasible for datasets like text for which d is very high. Also, the method involves a matrix multiplication $H^T H$ where H is a $n \times (d+1)$ matrix. So the entire data matrix needs to be kept in memory and hence the method is not scalable in terms of memory.

Keerthi and DeCoste [2005] presented an algorithm L2-SVM-MFN which uses a conjugate gradient method to solve the SVM problem and thus does not have to perform any matrix inversion as the previous method. Results in their paper indicate that the algorithm performs very well for large high dimensional datasets like text. Analysis of the algorithm indicates that it accesses the data vectors in a sequential manner and hence does not have to keep the data matrix in main memory, making it scalable in terms of memory.

Our work is closely related to Balcazar et al. [2001a,b]. They propose that d be used as the combinatorial dimension of the problem for the separable case. The dual of the SVM problem, when the data is linearly separable, is the minimum distance between the 2 convex hulls of the positive and negative examples. When the data is not linearly separable, these 2 hulls overlap. This can reduced to the separable case, by condensing the 2 hulls[Bennett and Bredensteiner, 2000]. This is done as follows. Let Z be the set of composed examples z_I where $z_I = \frac{x_{i_1}+\ldots+x_{i_m}}{m}$, where each x_{i_j} is a distinct element of D and all the points defining a z_I have the same label and the label of z_I is the same(For details on this condensation, see their paper). In this case, we have $|Z| \leq {}^nC_m$ and $m+1 \leq \Delta \leq m(d+1)$. It is this aspect of the SVM problem which was used by the authors to develop a randomized algorithm to solve the problem. The algorithm proposed is as follows:

Algorithm 1 RandSVM(D, Δ)

Require: D - Dataset
Require: Δ - Combinatorial dimension.

```
Sample size r = 6Δ^2.
Set weights w(x_i) to be 1 for all examples in D. For any set A ⊆ D, let
  w(A) = Σ_{x_i ∈ A} w(x_i).
repeat
  Select a sample S of size r randomly according to w.
  Use a SVM solver to solve the smaller problem. Let the classifier
    obtained be C.
  Classify the non sampled documents D − S.
  Let V be the set of misclassified documents and let v be the size of V.
  if (w(V) ≤ w(D)/(3Δ)) then
    Double the weights of misclassified documents.
  end if
until v = 0.
Done.
```

The algorithm proceeds in multiple iterations, where in each iteration it picks up a subset of the training data S, such that the size of the subset, r, is greater than the number of support vectors. Any SVM solver can be used to train a classifier C on the sampled subset, which is smaller than the entire data. Based on the classifier C obtained, the sampling probabilities are changed for the training data such that in successive iterations, the support vectors have a higher probability of selection. This process is repeated until the number of misclassified documents $v = 0$. The termination of the algorithm is guaranteed in a probabilistic fashion in Clarkson [1995]. The authors recommend using $m(d+1)$ as an estimate of Δ. This choice of Δ makes the subset size too large for high dimensional datasets, making it impractical.

To overcome this problem we use ideas from random projections [Johnson and Lindenstauss, 1984, Dasgupta and Gupta, 1999, Arriaga and Vempala, 1999]. Consider projecting the data points into a random k dimensional subspace where $k << d$. Using this idea, we give a theoretical bound on the combinatorial dimension Δ which is much lesser than the original data dimension d, in the almost linearly separable case. In practice, it has been observed that Δ is even lower. We then apply this to make the above algorithm scalable(without actually performing any random projection of the data). Appendix A contains a few results for random projections which will be used later.

3 A NEW RANDOMIZED ALGORITHM FOR CLASSIFICATION

The problem with the algorithm **RandSVM** is that the size of the subset sampled is too large and the terminating conditions are very strict, hence it is not useful in practice. In this work, we try to address these problems by using results from work done on random projections and extend the algorithm to handle other problems which are similar to the SVM problem.

In Section 3.1, we discuss the case of linearly separable data and determine a value Δ which is much smaller than the data dimension d. In Section 3.2, we look how the analysis applies to *almost separable* data and present the main result of the paper(Theorem 2). In Section 3.3, we apply the results obtained to text data and present a few other heuristics which make the algorithm practical. In Section 3.4, we discuss the L_1 norm formulation for classification and how the algorithm works for this case.

3.1 Linearly separable data

We start with determining the dimension k of the target space when performing a random projection such that the Euclidean distances and dot products are preserved.

The C-SVM formulation for dataset $D = \{(x_i, y_i), i = 1, \ldots, n\}, x_i \in R^d, y_i \in \{+1, -1\}$ which is linearly separable is given as follows:

C-SVM-2:

$$Minimize_{(w,b)} \ \frac{1}{2}||w||^2$$

$$Subject\ to: \ y_i(w \cdot x_i + b) \ \geq \ 1, \ i = 1 \ldots n$$

By dividing all the constraints by $||w||$, the problem can be reformulated as follows:

C-SVM-2a:

$$Maximize_{(\hat{w},b,l)} l$$

$$\begin{aligned} Subject\ to:\ y_i(\hat{w}\cdot x_i+\hat{b}) &\geq l,\ i=1\dots n \\ ||\hat{w}|| &= 1 \end{aligned}$$

where $\hat{w} = \frac{w}{||w||}$, $\hat{b} = \frac{b}{||w||}$, and $l = \frac{1}{||w||}$. In the SVM problem, we want two parallel hyperplanes $h_1 = \{x_i : w^T x_i + b = 1\}, h_2 = \{x_i : w^T x_i + b = -1\}$ which separate the two classes; the distance between the two hyperplanes l is called the margin . We refer to this as the *margin induced by the middle hyperplane* $h(w$ is the normal of this hyperplane).

The determination of k proceeds as follows. First, for any given value of k, we randomly project the data points onto a k dimensional subspace and show how the margin changes as a function of k. From this, we determine the value $k(k << d)$ which preserves margin with a very high probability. From this, it can be deduced that in the k dimensional subspace, there are at most $k+1$ support vectors. Using the idea of *orthogonal extensions*(definition appears later in this section), we then prove that when the problem is solved in the original space, there are utmost $k+1$ support vectors instead of $d+1$.

Let w' and $x'_i, i = 1, \dots, n$ be the projection of $\hat{w}$ and $x_i, i = 1, \dots, n$ respectively onto a k dimensional subspace (as in **Lemma 2**, **Appendix A**). The classification problem in the projected space with the dataset being $D' = \{(x'_i, y_i), i = 1, \dots, n\}, x'_i \in R^k, y_i \in \{+1, -1\}$ can be written as follows:
C-SVM-2b:

$$Maximize_{(w',\hat{b},l')} l'$$

$$\begin{aligned} Subject\ to:\ y_i(w'\cdot x_i+\hat{b}) &\geq l',\ i=1\dots n \\ ||w'|| &\leq 1 \end{aligned}$$

where $l' = l(1-\gamma)$, γ is the distortion and $0 < \gamma < 1$. Now, given a value of γ, the value of k such that the above problem can be solved with the optimal margin obtained close to the optimal margin for the original problem is given by the following lemma.

Theorem 1 *Let $L = max||x_i||$ and (w^*, b^*, l^*) be the optimal solution for **C-SVM-2a**. Let R be a random $d \times k$ matrix as given in **Lemma 2(Appendix A)**. Let $\widetilde{w} = \frac{R^T w^*}{\sqrt{k}}$ and $x'_i = \frac{R^T x_i}{\sqrt{k}}, i = 1, \dots, n$ and $k \geq \frac{8}{\gamma^2}(1 + \frac{(1+L^2)}{2l^*})^2 \log \frac{4n}{\delta},\ 0 < \gamma < 1,\ 0 < \delta < 1$, then the following bound holds on the optimal margin l_P obtained by solving the problem **C-SVM-2b**:*

$$P(l_P \geq l^*(1-\gamma)) \geq 1 - \delta$$

Proof *From **Corollary 1** of **Lemma 2(Appendix A)**, we have*

$$w^* \cdot x_i - \frac{\epsilon}{2}(1+L^2) \leq \widetilde{w} \cdot x'_i \leq w^* \cdot x_i + \frac{\epsilon}{2}(1+L^2)$$

which holds with probability at least $1 - 4e^{-\epsilon^2 \frac{k}{8}}$*, for some* $\epsilon > 0$*. Consider some example* x_i *with* $y_i = 1$*. Then the following holds with probability at least* $1 - 2e^{-\epsilon^2 \frac{k}{8}}$

$$\widetilde{w} \cdot x'_i + b^* \geq w^* \cdot x_i - \frac{\epsilon}{2}(1+L^2) + b^* \geq l^* - \frac{\epsilon}{2}(1+L^2)$$

Dividing the above by $||\widetilde{w}||$*, we have*

$$\frac{\widetilde{w} \cdot x'_i + b^*}{||\widetilde{w}||} \geq \frac{l^* - \frac{\epsilon}{2}(1+L^2)}{||\widetilde{w}||}$$

*Note that from **Lemma 1(Appendix A)**, we have*

$$(1-\epsilon)||w^*|| \leq ||\widetilde{w}|| \leq (1+\epsilon)||w^*||$$

which holds with probability at least $1 - 2e^{-\epsilon^2 \frac{k}{8}}$*. Since* $||w^*|| = 1$*, we have*

$$1 - \epsilon \leq ||\widetilde{w}|| \leq 1 + \epsilon$$

Hence we have

$$\begin{aligned}
\frac{\widetilde{w} \cdot x'_i + b^*}{||\widetilde{w}||} &\geq \frac{l^* - \frac{\epsilon}{2}(1+L^2)}{1+\epsilon} \\
&\geq (l^* - \frac{\epsilon}{2}(1+L^2))(1-\epsilon), \\
&= l^*(1 - \epsilon - \frac{\epsilon}{2l^*}(1+L^2)(1-\epsilon)) \\
&\geq l^*(1 - \epsilon - \frac{\epsilon}{2l^*}(1+L^2)) \\
&= l^*(1 - \frac{\epsilon}{2l^*}(2l^* + 1 + L^2)) \\
&= l^*(1 - \epsilon(1 + \frac{1+L^2}{2l^*}))
\end{aligned}$$

This holds with probability at least $1 - 4e^{-\epsilon^2 \frac{k}{8}}$*. A similar result can be derived for a point* x_j *for which* $y_j = -1$*. The above analysis guarantees that by projecting onto a* k *dimensional space, there exists at least one hyperplane* $(\frac{\widetilde{w}}{||\widetilde{w}||}, \frac{b^*}{||\widetilde{w}||})$*, which guarantees a margin of* $l^*(1-\gamma)$ *where*

$$\gamma \leq \epsilon(1 + \frac{1+L^2}{2l^*}) \tag{1}$$

with probability at least $1 - n4e^{-\epsilon^2 \frac{k}{8}}$. *The margin obtained by solving the problem* ***C-SVM-2b****,* l_P *can only be better than this.*

Now we determine the value of k*.*

$$
\begin{aligned}
n4e^{-\frac{\gamma^2}{(1+\frac{1+L^2}{2l^*})^2}\frac{k}{8}} &\leq \delta \\
\Rightarrow k &\geq \frac{8(1+\frac{(1+L^2)}{2l^*})^2}{\gamma^2} \log \frac{4n}{\delta}
\end{aligned}
$$

As seen above, by randomly projecting the points onto a k dimensional subspace, the margin is preserved with a high probability. In a k dimensional space, the number of support vectors is upper bounded by $k+1$. We now show that this upper bound of $k+1$ holds with high probability even when the problem is solved in the original space. We start with the following definition.

Definition *We define an* ***`orthogonal extension`*** *of a* $k-1$*-dimensional flat(a* $k-1$ *dimensional flat is a* $k-1$*-dimensional affine space)* $h_p = (w_p, b)$ *where* $w_p = (w_1, \ldots, w_k)$ *in a subspace* S_k *of dimension* k *to a* $d-1$*-dimensional hyperplane* $h = (\widetilde{w}, b)$ *in d-dimensional space as follows. Let* $R \in R^{d \times d}$ *be a random projection matrix as in* ***Lemma 2****. Let* $\hat{R} \in R^{d \times k}$ *be a another random projection matrix which consists of only the the first* k *columns of* R*. Define* $\hat{x}_i$ *and* x'_i *as follows:*

$$
\begin{aligned}
\hat{x}_i &= R^T x_i, \\
x'_i &= \frac{\hat{R}^T}{\sqrt{k}} x_i, \;\; i = 1, \ldots, n
\end{aligned}
$$

Let $w_p = (w_1, \ldots, w_k)$ *be the optimal hyperplane classifier with margin* l_P *for the points* $x'_1, \ldots, x'_n$ *in the* k *dimensional subspace. Now define* $\widetilde{w}$ *to be all 0's in the last* $d-k$ *coordinates and identical to* w_p *in the first* k *coordinates, that is,*

$$\widetilde{w} = (w_1, \ldots, w_k, 0, \ldots, 0)$$

Orthogonal extensions have the following key property. If (w_p, b) *is a separator with margin* l_p *for the projected points, then its orthogonal extension* $(\widetilde{w}, b)$ *is a separator with margin* l_p *for the original points,that is,*
if,

$$y_i(w_p \cdot x'_i + b) \geq l, \;\; i = 1, \ldots, n$$

then

$$y_i(\widetilde{w} \cdot \hat{x}_i + b) \geq l, \;\; i = 1, \ldots, n$$

As the results of this, the w^* which classifies the original data is given as follows:

$$\begin{aligned} y_i(\widetilde{w} \cdot \hat{x}_i + b) &\geq l, \;\; i = 1, \ldots, n \\ \Rightarrow y_i(\widetilde{w} \cdot R^T x_i + b) &\geq l, \;\; i = 1, \ldots, n \\ \Rightarrow y_i((R\widetilde{w}) \cdot x_i + b) &\geq l, \;\; i = 1, \ldots, n \\ \Rightarrow w^* &= R\widetilde{w} \end{aligned}$$

Let L, l^*, γ, δ and n be as defined in **Theorem 1**.

Claim 1 *Given $k \geq \frac{8}{\gamma^2}(1 + \frac{(1+L^2)}{2l^*})^2 \log \frac{4n}{\delta}$ and n training points with maximum norm L in d dimensional space and separable by a hyperplane with margin l^*, there exists a subset of k training points $x_1 \ldots x_k$ and a hyperplane h satisfying the following conditions:*

1. *h has margin at least $l^*(1-\gamma)$ with probability at least $1-\delta$*
2. *$x_1 \ldots x_k$ are the only training points which lie either on h_1 or on h_2*

Proof *Let w^*, b^* denote the normal to a separating hyperplane with margin l^*, that is, $y_i(w^* \cdot x_i + b^*) \geq l^*$ for all x_i and $||w^*|| = 1$. Consider a random projection of $x_1, \ldots, x_n$ to a k dimensional space and let $w', z_1, \ldots, z_n$ be the projections of $w^*, x_1, \ldots, x_n$, respectively, scaled by $1/\sqrt{k}$. By **Theorem 1**, $y_i(w' \cdot z_i + b^*) \geq l^*(1-\gamma)$ holds for all z_i with probability at least $1-\delta$. Let h be the orthogonal extension of w', b^* to the full d dimensional space. Then h has margin at least $l^*(1-\gamma)$, as required. This shows the first part of the claim.*

To prove the second part, consider the projected training points which lie on w', b^* (that is, they lie on either of the two sandwiching hyperplanes). Barring degeneracies, there are exactly k such points. Clearly, these will be the only points which lie on the orthogonal extension h, by definition.$\square$

From the above analysis, it is seen that if $k << d$, then we can choose a value of $\Delta = k + 1$ and the algorithm **RandSVM** would take on average $O(k \log n)$ iterations to solve the problem [Balcazar et al., 2001a,b].

3.2 Almost separable data

In this section, we look at how the above analysis can be applied to *almost separable* datasets. We call a dataset *almost separable* if by removing a fraction $\kappa = O(\frac{\log n}{n})$ of the points, the dataset becomes linearly separable(Fig 1).

The C-SVM formulation when the data is not linearly separable(and *almost separable*) was given in **C-SVM-1**. This problem can be reformulated as follows:

$$Minimize_{(w,b,\xi)} \sum_{i=1}^{n} \xi_i$$

$$\begin{aligned} Subject\ to: \ y_i(w \cdot x_i + b) &\geq l - \xi_i, \ i = 1 \ldots n \\ ||w|| &\leq \frac{1}{l} \\ \xi_i &\geq 0, \ i = 1 \ldots n \end{aligned}$$

This formulation is known as the *Generalized Optimal Hyperplane* formulation. Here l depends on the value of C in the C-formulation. At optimality, the margin $l^* = l$. The following theorem proves a result for almost separable data similar to the one proved in **Claim 1** for separable data.

Theorem 2 *Given $k \geq \frac{8}{\gamma^2}(1 + \frac{(1+L^2)}{2l^*})^2 \log \frac{4n}{\delta} + \kappa n$, l^* being the margin at optimality, l the lower bound on l^* as in the Generalized Optimal Hyperplane formulation and $\kappa = O(\frac{\log n}{n})$, there exists a subset of k training points $x_1 \ldots x_k$ and a hyperplane h satisfying the following conditions:*

1. *h has margin at least $l(1-\gamma)$ with probability at least $1-\delta$*
2. *$\frac{8(1+\frac{(1+L^2)}{2l^*})^2}{\gamma^2} \log \frac{4n}{\delta}$ points lie on the planes h_1 or on h_2*
3. *$x_1, \ldots, x_k$ are the only points which form the support vectors of the h*

Proof *Let the optimal solution for the generalized optimal hyperplane formulation be (w^*, b^*, ξ^*). $w^* = \sum_{i:\alpha_i>0} \alpha_i y_i x_i$, and $l^* = \frac{1}{||w^*||}$ as mentioned before. The set of support vectors can be split into to 2 disjoint sets defined as follows:*

$$\begin{aligned} SV_1 &= \{x_i : \alpha_i > 0 \ and \ \xi_i^* = 0\} \\ SV_2 &= \{x_i : \alpha_i > 0 \ and \ \xi_i^* > 0\} \end{aligned}$$

*Now, consider removing the points in SV_2 from the dataset. Then the dataset becomes linearly separable with margin l^*. Using an analysis similar to **Theorem 1**, and the fact that $l^* \geq l$, we have the proof for the first 2 conditions.*

When all the points in SV_2 are added back, at most all these points are added to the set of support vectors and the margin does not change. The

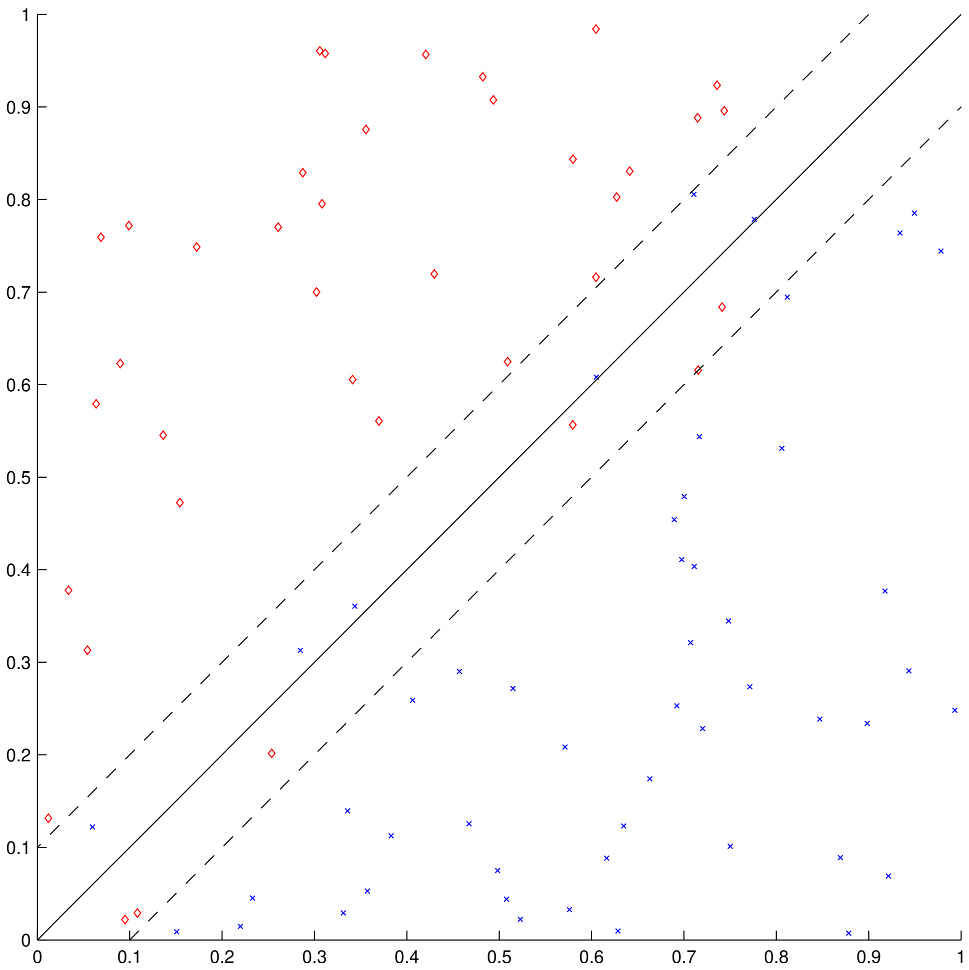

Figure 1: The figure shows the *almost separability* property for a dataset. By removing a small fraction of the points the data becomes linearly separable. At optimality the margin is $2l$. The middle solid line is the classifier $h : w \cdot x_i + b = 0$. The 2 sandwiching hyperplanes(dashed lines) are $h_1 : w \cdot x_i + b = 1$ and $h_2 : w \cdot x_i + b = -1$.

margin not changing is guaranteed by the fact that for proving the conditions 1 and 2, we have assumed the worst possible margin, and any value lower than this would violate the constraints of the problem. This proves condition 3. $\square$

Hence the *combinatorial dimension* Δ of the problem is given by:

$$\begin{aligned}\Delta &= \frac{8}{\gamma^2}(1+\frac{(1+L^2)}{2l^*})^2 \log\frac{4n}{\delta} + \kappa n \\ &= \frac{8}{\gamma^2}(1+\frac{(1+L^2)}{2l^*})^2 \log\frac{4n}{\delta} + O(\log n) \end{aligned} \quad (2)$$

Using this value of Δ, RandSVM can be applied to solve the problem.

The problem with this is determining the value of κ. To handle this, the $\nu - SVM$ formulation can be used. The formulation is as follows:

$$Minimize_{(w,b,\rho,\xi)}\ \frac{1}{2}||w||^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

$$\begin{aligned} Subject\ to:\ y_i(w^T x_i + b) &\geq \rho - \xi_i,\ i = 1\ldots n \\ \xi_i &\geq 0,\ i = 1\ldots n \\ \rho &\geq 0 \end{aligned}$$

Here ν is a user-chosen parameter between 0 and 1. The parameter ν sets a bound on the number of support vectors and training errors. The bounds are as follows:

$$\begin{aligned} \nu &\geq \frac{\text{Number of Training Errors}}{n} \\ \nu &\leq \frac{\text{Number of Support Vectors}}{n} \end{aligned}$$

Hence, given the value of the ν parameter, a value of of $\Delta \geq \frac{8(1+\frac{(1+L^2)}{2l^*})^2}{\gamma^2}\log\frac{4n}{\delta} + \nu n$ can be chosen and the problem solved on this subset.

3.3 Applicability of the above analysis to text data

Now, as mentioned before this analysis is useful when the value of l is large. In the case of linearly non-separable data, this implies that the first term in Equation 2 is small when the l is chosen to be sufficiently large, that is, if C is chosen to be small. But at the same time, increasing l increases the the second term. Hence C should be chosen reasonably. Else the value Δ won't be much less than the value of n, and the algorithm would not be very useful.

It has been observed that this particular property does hold for text data. In the experiments we conducted, the value of Δ was usually upper bounded by $0.1n$ as a result of which this algorithm is practically useful in the case of text data.

3.3.1 Determination of combinatorial dimension

The value of k depends on the l which is not available unless the generalized optimal hyperplane formulation is used. Since most solvers solve either the C-SVM or the nu-SVM formulation, this implies that k cannot be determined using **Theorem 1**. To overcome this problem, we solve for k as a function of ϵ where ϵ is the maximum allowed distortion in the L_2 norms of the vectors upon projection. The lower the required distortion, the higher the value of k. For text data, all the data points are normalized to length 1, that is, $L = 1$. From the proof for **Theorem 1**(Equation 1),

$$\epsilon \geq \frac{\gamma}{1 + \frac{1+L^2}{2l^*}}$$

Now from **Theorem 2**, we have:

$$\begin{aligned} k &\geq \frac{8}{\gamma^2}(1 + \frac{(1+L^2)}{2l^*})^2 \log \frac{4n}{\delta} + O(\log n) \\ &\geq \frac{16}{\gamma^2}(1 + \frac{(1+L^2)}{2l^*})^2 \log \frac{4n}{\delta}) \end{aligned}$$

Hence, by choosing k as follows:

$$k \geq \frac{16}{\epsilon^2} \log \frac{4n}{\delta} \tag{3}$$

, the problem can be solved.

3.3.2 Some heuristics

Sample Size: Consider the algorithm **RandSVM**. The sample size in the algorithm is chosen as $r = 6\Delta^2$(where $\Delta = k$). Even with a reduced value of Δ, the value of $r = 6\Delta^2$ for the sample size is very large(sometimes larger than the dataset size). For example, in the RCV1 dataset, the smallest value of Δ we chose is around 3000. The number of documents is around $8,00,000+$. Hence using the sample size of $6\Delta^2$ is not useful. As the result

of which the algorithm still remains inefficient in practice. To overcome this problem, we chose a value $\Delta = k$. The disadvantage of doing so is that we do not have(currently) an asymptotic(as in Balcazar et al. [2001a,b]) bound on the number of iterations. But we have empirical validation for this heuristic.
Stopping criterion: In Balcazar et al. [2001a,b], the stopping criterion for the algorithm is that the number of violators is zero. We noticed that this criterion was too strict for practical purposes. Using this critetion results in too many iterations and does not provide us with any advantage with regards to classification accuracy. Hence we use a stopping criterion which is the training error acceptable. This is specified using a parameter err which is used as follows: if the violators form a fraction of entire dataset less than err, then the algorithm terminates. There are other termination criterion that can also be used, for example, the maximum number of iterations. We have tested the algorithm only with the first stopping criterion.

The algorithm with these heuristics is given as **RandSVM-Mod**.

Algorithm 2 RandSVM-Mod(D, ϵ, κ, err, W)

Require: D - Dataset
Require: ϵ, κ - Parameters used to determine sample size
Require: err - Stopping criterion; training error acceptable.
Require: W - weighting function, specifies how the weights are updated.
Determine sample size r using parameters ϵ, κ, and $|D|$.
Let P_1 be the initial distribution(uniform) of the training data.
Let $i = 1$
repeat
    Select a sample S of size r based on P_i.
    Use a SVM solver to solve the smaller problem. Let the classifier obtained be C.
    Classify the non sampled documents $D - S$.
    Let V be the set of misclassified documents and $|V| = v$.
    Change the weights of the misclassified documents using W.
    $i = i + 1$.
until $v/|D| \leq$ err.
Done

3.4 Randomized linear programming

The L_2 norm SVM formulation has an equivalent L_1 norm formulation [Bennett and Bredensteiner, 2000]. The L_1 norm formulation can be additionally used as a feature selection mechanism. This formulation is as follows:

$$Minimize_{(w,b,\xi)} \sum_{j=1}^{d} |w_j| + C \sum_{i=1}^{n} \xi_i$$

$$\begin{aligned} Subject\ to: \ y_i(w \cdot x_i + b) &\geq 1 - \xi_i, \ i = 1 \ldots n \\ \xi_i &\geq 0, \ i = 1 \ldots n \end{aligned}$$

By using the following substitution for w,

$$w = u - v, \quad |w| = u + v, \quad u_j, v_j \geq 0, \ j = 1 \ldots d$$

the problem is reformulated into the following LP problem before it is solved:

$$Minimize_{u,v,b,\xi} \sum_{j=1}^{d} (u_j + v_j) + C \sum_{i=1}^{n} \xi_i$$

$$\begin{aligned} Subject\ to: \ y_i((u - v) \cdot x_i + b) &\geq 1 - \xi_i \ i = 1 \ldots n \\ 0 &\leq u_j, v_j, \ j = 1 \ldots d \\ \xi_i &\geq 0, \ i = 1 \ldots n \end{aligned}$$

This problem can be solved using any LP solver. But it has been observed that state of the art LP solvers like *LPSolve* cannot do the learning even for datasets consisting of just $2,00,000$ examples(high dimensional data). Hence the applicability of this formulation was limited.

Even though the support vectors determined by solving this formulation are different from those determined by solving the L_2-norm formulation; when the data is sparse and almost separable, an analysis similar to the one done in the previous sections can be used to bound the combinatorial dimension of the problem. Hence this formulation can be solved by **RandSVM-Mod**, except in this case, the solver used to solve the subproblem is a LP solver.

4 Experiments

In this section, we empirically validate the claims made before. To start with we do experiments on a synthetic dataset which is almost separable where we compare SVMLight and RandSVM(with SVMLight as the solver). We show that choosing Δ as given here is enough to solve the problem. Then we look at the execution time and memory required by both the algorithms.

The second set of experiments are on real life text datasets, REUTERS-21578 and RCV1. We show that real life text datasets like RCV1 do satisfy the *almost separability* property. Then we compare RandSVM with state of the art solvers (SVMLight, LibSVM and LPSolve) with respect to memory, execution time, how these two performance measures change with the increase in data size and finally, classification accuracy.

We would like to mention that we have not compared RandSVM with L2-SVM-MFN because the code for the latter is not available. We tried comparing RandSVM with ProximalSVM, but ProximalSVM did not run even on the synthetic dataset. Hence we do not present any results with respect to ProximalSVM.

4.1 Randomized SVM

4.1.1 Dataset

The experiments were conducted using three datasets, a synthetic dataset, the REUTERS-21578 and RCV1. REUTERS-21578 was used to verify the correctness of the algorithm, that is, to ensure that the algorithm builds classifiers which are as good as the ones constructed using the entire corpora. The RCV1 dataset was used to determine the scaling properties of the algorithm apart from the classification accuracy.

Synthetic dataset The synthetic dataset consists of 1,00,000 training documents, half of which have label +1 and the other half have label -1. Each of the examples has 5000 features. The dataset was generated by selecting the features for the +1 and -1 classes from $N(0.1, 1)$ and $N(-0.1, 1)$ respectively. The dataset is 99% separable. The test set has the same properties as the training set but consists of 5000 documents. This set was used to compare the performance of SVMLight and RandSVM(with SVMLight as the solver).

REUTERS-21578 The REUTERS-21578 dataset consists of 21578 documents. This was split into a training set consisting of 15000 documents and a test set consisting of the remaining documents. The experiments were conducted on 5 different categories [Chakrabarti et al., 2002]: acq, crude, earn, interest and money-fx. For a given category, the training dataset was generated as follows: let x be the number of positive documents for a category in the dataset. Out of these, $.8x$ documents were randomly chosen and were included as part of the training set. The remaining training examples(negative examples) were chosen randomly from the negative examples. This mechanism was used to ensure that the positive documents were well represented in the training data, as the number of positive documents was much smaller than the number of training documents. Also, by using this method some positive examples are included in the test set, without which the precision and recall measures would be zero. This was done 3 times for each category so that we have 3 different training sets.

The test data was randomly split into 10 sets of equal(or almost equal) size, while ensuring that the ratio of positive to negative documents in all sets were equal(or almost equal).

RCV1 This dataset consists of 8,00,000+ documents. This was split into training sets of sizes 50,000, 1,00,000 to 7,00,000 in steps of 1,00,000. For each category, the training set and test sets were created as above, except for a slight modification in the creation of the training set. For some categories, the number of positive examples exceeded some of the training set sizes. Hence the selection of positive training examples was done as follows: if x is the number of positive examples for a category, and n is the training set size(not the entire dataset size), then randomly choose $.8x$ or $.5n$ positive documents, whichever is smaller. This ensures, the positive documents are not over-represented in the training data (in the extreme, the entire training set would consist of only positive documents).

10 categories were chosen for the experiments. 4 of these categories are the top categories in the *topics* hierarchy - MCAT, GCAT, ECAT, CCAT. The remaining six categories chosen are those which have very few misclassified documents - GSPO, E121, E13, E142, E132 and M143.

4.1.2 Experiments with the synthetic dataset

The experiments were run on a Intel Dual Core 2.8GHz machine with 2GB of memory. For the synthetic dataset, SVMLight was used as the solver in the experiments. The value of the C parameter was chosen using 10 fold cross validation. The tuned C value chosen for SVMLight and RandSVM(with SVMLight) were 0.0001 and 1 respectively. RandSVM was not used with C values below 1 because it was noticed that the number of support vectors were too large (around 30% of the data) and there is not much difference in the accuracies attained. We would like to make an important point here: RandSVM is independent of the value of C as long as it is chosen reasonably(that is, the value of C should not be too low or too high, else a large number of support vectors, that is, greater than $O(\log n)$ will be found). The parameter δ was chosen to be $\frac{1}{n}, n = 100000$. The parameter ϵ was chosen to have 3 different values $0.20, 0.25$ and 0.30.

Table 1 shows a comparison between RandSVM and SVMLight. It indicates that RandSVM outperforms SVMLight both in terms of time and memory while attaining the same classification performance.

Performance Metric	SVMLight $C = 0.0001$	RandSVM, $C = 1$ $\epsilon = 0.20$	$\epsilon = 0.25$	$\epsilon = 0.30$
Memory(in MB)	376/370	37/36	33.5/32	31.4/30
No. of SVs	35956	1931/3657	1720/2769	1574/2268
Time(in seconds)	873	271	254	245
F1	0.987	0.958	0.969	0.950

Table 1: This table shows a comparison between SVMLight and RandSVM(using SVMLight, C-SVM), on the synthetic dataset. Each memory entry must be read as Virtual Memory/Resident Memory. Each entry for the number of support vectors under RandSVM columns must be read as No. of Support Vectors/Sample Size.

4.1.3 Experiments with RCV1 and REUTERS-21578

Methodology The experiments were run on a Intel Dual Core 2.8GHz machine with 2GB of memory. 2 SVM solvers were used in the experiments. For the C-SVM formulation, SVMLight was used. The C value chosen was 100 for RCV1 dataset and 50 for REUTERS dataset. For the $\nu-$SVM formulation, LibSVM was used with $\nu = 0.06$. Since LibSVM was taking a lot of time to solve even the sampled problem, experiments with RandLibSVM were restricted on datasets of sizes 50000, 100000 and 200000 and 4 categories only:CCAT, MCAT, GCAT and ECAT.

The parameter δ was chosen to be $\frac{1}{n}$, where n is the training set size(and not the entire dataset size). The parameter ϵ was chosen to be $0.10, 0.15, 0.20$.

Almost separability Table 2 shows the comparison between the number of support vectors, number of bounded support vectors for SVMLight and RandSVM($\epsilon = 0.10$) for all the chosen categories of RCV1 dataset, with the largest training set size($7,00,000$ documents). As seen from the numbers, text data does exhibit the *almost separability* property. This implies that **RandSVM-Mod** can be applied to text classification problem.

Another important observation is that the number of support vectors attained using RandSVM is much lesser than the number of support vectors obtained using SVMLight. It seems this happens because as the dataset size grows larger, SVMLight relaxes the conditions on the solutions and terminates the problem without reaching the optimal value. This disparity is especially predominant when the category in consideration is a difficult category to learn, as in the case of the first 4 categories in the table.

Category	Rand-SV	Rand-BSV	SVM-SV	SVM-BSV
CCAT	6922	61	23438	4072
MCAT	4519	60	5910	20
GCAT	5415	8	18799	1760
ECAT	5871	59	19475	5409
C183	1200	0	5510	48
E121	261	2	1606	119
E132	172	0	526	9
E142	116	0	515	7
GSPO	1540	0	3064	90
M131	1272	29	7400	1005
M143	1037	1	4921	177

Table 2: This table shows the comparison between the number of support vectors and bounded support vectors as obtained by SVMLight and RandSVM(using SVMLight, $C-$SVM, $\epsilon = 0.10$) for the largest training dataset size used for each of the algorithms, for all the chosen categories of RCV1 dataset. The largest training set used for RandSVM consisted of $7,00,000$ documents and the largest dataset used for SVMLight consisted of $2,00,000$ documents as SVMLight never completed for datasets of larger sizes.

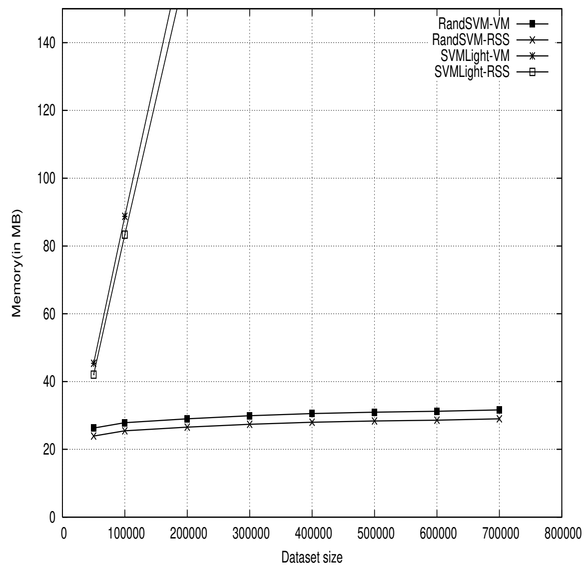

Figure 2: This graph shows the comparison of memory required by SVMLight and RandSVM(using SVMLight, $C-$SVM, $\epsilon = 0.10$) for different dataset sizes. The numbers for each dataset size is an average over all categories and all attempts for each category. VMS denotes the total virtual memory size and RSS denotes the amount of physical memory actually used.

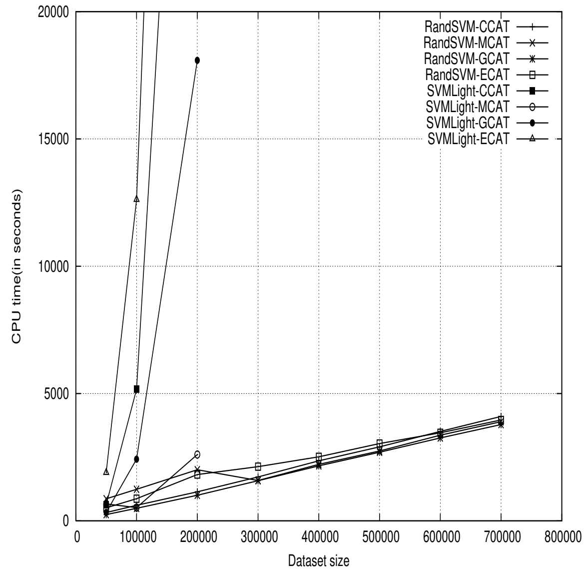

Figure 3: This graph shows a comparison of execution times between SVM-Light and RandSVM(using SVMLight, $C-$SVM, $\epsilon = 0.10$) for 4 categories – CCAT, MCAT, GCAT, ECAT. The numbers for a category and particular dataset size is an average of all the 3 attempts done for it.

Memory Fig 2 shows the comparison of memory required between SVM-Light and RandSVM($\epsilon = 0.10$). In the plot, VMS is Total Virtual Memory(including swap) and RSS depicts the amount of physical memory actually used. The values are obtained by taking an average over all categories and all attempts for each training dataset size. As expected, the memory required by SVMLight increases very rapidly as the dataset size increases. In the case of RandSVM, the increase in memory size is only logarithmic in the dataset size. Hence, there is a very little increase the memory required when the dataset size increases.

Execution time Fig 3 compares the execution time obtained by SVM-Light and RandSVM($\epsilon = 0.10$) for 4 of the categories of the RCV1 dataset. The categories chosen are the CCAT, MCAT, GCAT and ECAT. As seen in the plot, RandSVM outperforms SVMLight for all dataset sizes. This implies that considering the entire dataset for solving the problem far outweighs the overheads involved in sampling and multiple iterations of the RandSVM. Fig 4 shows the execution time obtained by RandSVM(using LibSVM, $nu-$SVM formulation). The numbers for LibSVM have not been included in the graph as it was observed that for a dataset size of just 100000, the solver did not terminate even after a long time.

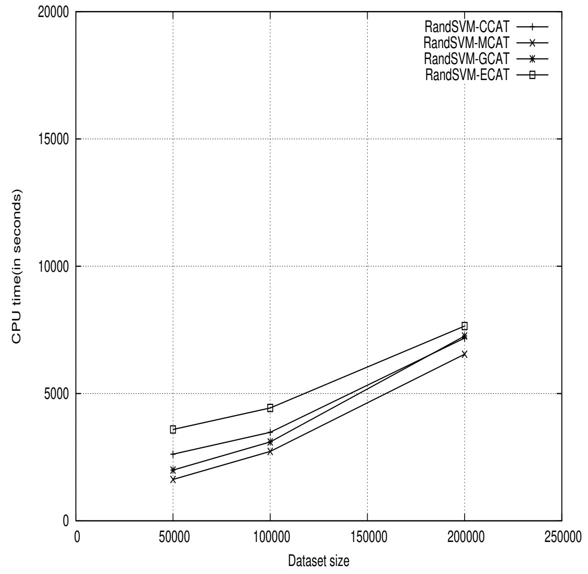

Figure 4: This graph shows a comparison of execution times between RandSVM(using LibSVM, $nu-$SVM, $\epsilon = 0.10$) for 4 categories – CCAT, MCAT, GCAT, ECAT. The numbers for a category and particular dataset size is an average of all the 3 attempts done for it.

Another important observation is that the rate at which execution time of RandSVM increases with dataset size is much smaller than for SVMLight. This happens because, as the dataset size increases, the size of sample chosen doesn't increase by much.

Accuracy Fig 5 and Fig 6 show the macroaveraged F1 attained by RandSVM($\epsilon =$ 0.10) and SVMLight for RCV1 and REUTERS-21578 respectively. The accuracies obtained are quite similar . This is an empirical verification of the correctness of the algorithm

Sensitivity with respect to ϵ In these experiments we test the sensitivity of RandSVM with respect to ϵ. We performed the experiments with one category of the RCV1 dataset, CCAT. The dataset sizes chosen were 50000, 100000 and 200000. The value of δ was chosen as in the previous case. The results are shown in Table 3. As noticed, the classification performance of RandSVM is not affected when ϵ is increased from 0.1 to 0.2 whereas the time and memory required are definitely reduced. Similar performance was noticed for other categories of the dataset.

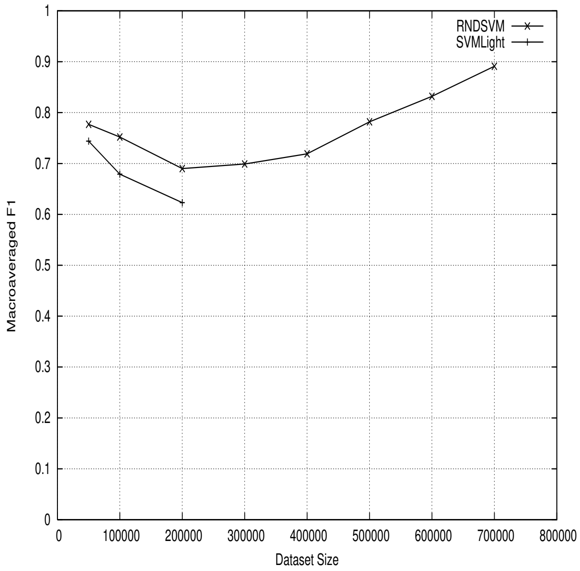

Figure 5: This graph shows a comparison of the Macroaveraged F1 between SVMLight and RandSVM(using SVMLight, $C-$SVM, $\epsilon = 0.10$) . The macroaverage was taken for all categories and all attempts for each category.

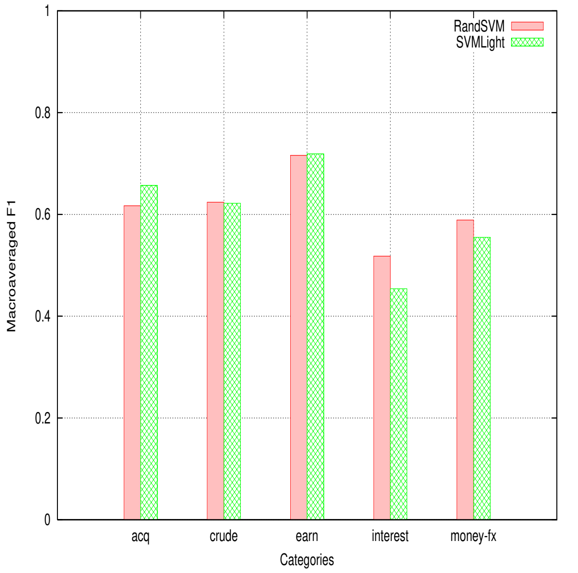

Figure 6: This graph shows a comparison of of the Macroaveraged F1 between SVMLight and RandSVM(using SVMLight, $C-$SVM, $\epsilon = 0.10$) for 5 categories of REUTERS dataset. The macroaverage was taken over all attempts for each category.

Performance Metric	Dataset Size	SVMLight	RandSVM		
			$\epsilon = 0.10$	$\epsilon = 0.15$	$\epsilon = 0.20$
Memory(in MB)	50000	45.38/42.05	26.29/23.91	5.93/4.52	4.46/3.08
	100000	88.74/83.36	27.88/25.46	6.13/4.76	4.60/3.21
	200000	172.56/163.02	29.00/26.54	6.34/4.97	4.75/3.36
Time(in seconds)	50000	636.39	326	49	36
	100000	5167.49	1141.67	101	72.67
	200000	45098.11	1723.67	201.67	149
F1	50000	0.878	0.882	0.894	0.891
	100000	0.884	0.889	0.893	0.894
	200000	0.894	0.886	0.892	0.891

Table 3: This table shows how the performance of RandSVM(using SVM-Light, C-SVM), changes with respect to the parameter ϵ. The learning was done on the CCAT category of the RCV1 dataset. Each memory entry must be read as Virtual Memory/Resident Memory.

Dataset	Split 1	Split 2	Split 3
Train-2000	108282	139640	139469
Train-2500	162351	162362	162296
Train-3000	183264	183278	183396

Table 4: Sizes of the datasets used for experiments with Randomized LP.

4.2 Randomized LP

Dataset The RCV1 dataset was used for experiments with L_1 norm formulation. The solver used to solve the linear program was LPSolve. Since this solver cannot handle large datasets efficiently, the training set sizes were kept below 2,00,000.

Three datasets of different sizes were created called Train-2000, Train-2500 and Train-3000. These datasets were generated like the Train-1600 dataset in Dunja Mladeni and Milic-Frayling [Sept 2002]. Dataset Train-X was created as follows. The topics hierarchy in the dataset consists of 103 categories. For each of these categories with more than X positive examples, X were randomly chosen. For the rest of the categories, all the documents were made part of the training set. 3 such datasets were created for each Train-X using different samples. The number of documents in each of the datasets is given in Table 4. The experiments were conducted for 11 different categories: C183, E121, E13, E132, E142, G154, GHEA, GOBIT.

The test set was generated as in the previous section.

Category	Memory (in MB)	Time(in seconds)
C183	159	3917
E121	130	3402
E13	159	4273
E132	130	3215
E142	114	3212
G154	159	4586
GHEA	161	3842
GOBIT	152	2760
GSPO	181	4184
M131	162	5373
M143	158	4327

Table 5: This table shows the memory required in MBs and training time in seconds for the largest dataset used for RandSVM with LPSolve for Train-3000. Each of the values is the maximum of 3 attempts. Since LPSolve could not handle any of the datasets, it has not been included in the table

.

Methodology The experiments were conducted on a Intel Xeon Quad processor machine with 4GB of memory. The LP solver used in the experiments was LPSolve. All the experiments were conducted using only the randomized algorithm, as it was observed that if LPSolve was used with the entire dataset, it required a very large amount of memory and did not finish even after a few days. The C value was chosen to be 100 for these experiments.

Memory and Execution time Table 5 shows the memory and execution time for all categories using Randomized LP solver for the Train-3000 dataset. If the entire dataset is submitted to LPSolve, it required around 1.4-1.6GB of memory and it did not finish even after running for a few days. Thus it is obvious LPSolve can't solve problems of even such a moderate size($<$ 2,00,0000) documents. But by using RandSVM, the problems are quite easily solved. This clearly is a very big gain.

5 Conclusion and Open Problems

In this paper, we proposed an algorithm using ideas from random projections and randomized algorithms which iterates over small subsets of the data and efficiently solves the SVM problem with high probability. We also introduced the concept of an *almost separable* dataset and showed that real life text datasets satisfy this property. As seen from the experiments the algorithm works very well for such datasets and has good scalability. The biggest advantage of this algorithm is that it can scale up any SVM solver without doing any kind of modifications to the code of the solver, that is, it uses any SVM solver as a black box. And since the implementation of the algorithm is very simple, scalability is obtained without much effort.

The work presented here throws up some interesting open problems. The most important problem that needs to be addressed is the asymptotic bound on the number of iterations that the algorithm will take. A second area of interest is tying up the stopping criterion with the theory. Thirdly, we would like to look at applying the method to non-separable data in general using kernels.

A Some Results from Random Projections

Here we review a few lemmas from random projections [Arriaga and Vempala, 1999]. The following lemma discusses how the L_2 norm of a vector is preserved when it is projected on a random subspace.

Lemma 1 *Let $R = (r_{ij})$ be a random $d \times k$ matrix, such that each entry (r_{ij}) is chosen independently according to $N(0,1)$. For any fixed vector $u \in R^d$, and any $\epsilon > 0$, let $u' = \frac{R^T u}{\sqrt{k}}$. Then $E[||u'||^2] = ||u||^2$ and the following bounds hold:*

$$(1-\epsilon)||u||^2 \leq ||u'||^2 \leq (1+\epsilon)||u||^2$$

with probability at least $1 - 2e^{(\epsilon^2 - \epsilon^3)\frac{k}{4}}$.

The following theorem and its corollary show the change in the Euclidean distance between 2 points and the dot products when they are projected onto a lower dimensional space [Arriaga and Vempala, 1999].

Lemma 2 *Let $u, v \in R^d$. Let $u' = \frac{R^T u}{\sqrt{k}}$ and $v' = \frac{R^T u}{\sqrt{k}}$ be the projections of u and v to R^k via a random matrix R whose entries are chosen independently from $N(0,1)$ or $U(-1,1)$. Then for any $\epsilon > 0$, the following bounds hold*

$$(1-\epsilon)\|u - v\|^2 \leq \|u' - v'\|^2$$

with probability at least $1 - e^{-(\epsilon^2 - \epsilon^3)\frac{k}{4}}$ and

$$\|u' - v'\|^2 \leq (1+\epsilon)\|u - v\|^2$$

with probability at least $1 - e^{-(\epsilon^2 - \epsilon^3)\frac{k}{4}}$.

A corollary of the above theorem shows how well the dot products are preserved upon projection(This is a slight modification of the corollary given in Arriaga and Vempala [1999]).

Corollary 1 *Let u, v be vectors in R^d s.t. $\|u\| \leq L_1, \|v\| \leq L_2$. Let R be a random matrix whose entries are chosen independently from either $N(0,1)$ or $U(-1,1)$. Define $u' = \frac{R^T u}{\sqrt{k}}$ and $v' = \frac{R^T v}{\sqrt{k}}$. Then for any $\epsilon > 0$, the following holds with probability at least $1 - 4e^{-\epsilon^2 \frac{k}{8}}$*

$$u \cdot v - \frac{\epsilon}{2}(L_1^2 + L_2^2) \leq u' \cdot v' \leq u \cdot v + \frac{\epsilon}{2}(L_1^2 + L_2^2)$$

Proof *For the vectors u and v, let the event E_1 be*

$$(1-\epsilon)||u-v||^2 \leq ||u'-v'||^2 \leq (1+\epsilon)||u-v||^2$$

and E_2 be

$$(1-\epsilon)||u+v||^2 \leq ||u'+v'||^2 \leq (1+\epsilon)||u+v||^2$$

Hence from the previous lemma:

$$P(E_1 \text{ and } E_2) \geq 1 - 4e^{-(\epsilon^2-\epsilon^3)\frac{k}{4}}$$

Now,

$$\begin{aligned} u' \cdot v' &= \frac{1}{4}(||u'+v'||^2) - ||u'-v'||^2) \\ &\leq \frac{1}{4}((1+\epsilon)||u+v||^2 - (1-\epsilon)||u-v||^2) \\ &= u \cdot v + \frac{\epsilon}{2}(||u||^2 + ||v||^2) \\ \Rightarrow u' \cdot v' &\leq u \cdot v + \frac{\epsilon}{2}(L_1^2 + L_2^2) \end{aligned}$$

The above inequality holds with probability greater than or equal to $1 - 2e^{-(\epsilon^2-\epsilon^3)\frac{k}{4}}$ Similarly,

$$u' \cdot v' \geq u \cdot v - \frac{\epsilon}{2}(L_1^2 + L_2^2)$$

holds with probability greater than or equal to $1-2e^{-(\epsilon^2-\epsilon^3)\frac{k}{4}}$. Hence the proof.

References

R. I. Arriaga and S. Vempala. An algorithmic theory of learning: Random concepts and random projections. In *Proceedings of the 40th Foundations of Computer Science*, 1999.

Jose L. Balcazar, Yang Dai, and Osamu Watanabe. A random sampling technique for training support vector machines. In *ALT*. Springer, 2001a.

Jose L. Balcazar, Yang Dai, and Osamu Watanabe. Provably fast training algorithms for support vector machines. In *ICDM*, pages 43–50, 2001b.

K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 57–64, San Francisco, California, 2000. Morgan Kaufmann Publishers.

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

S. Chakrabarti, S. Roy, and M. V. Soundalgekar. Fast and accurate classification via multiple linear projections. In *VLDB*, 2002.

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.

Kenneth L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42(2):488–499, 1995.

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

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical Report TR-99-006, Berkeley, CA, 1999. URL `citeseer.ist.psu.edu/dasgupta99elementary.html`.

Marko Grobelnik Dunja Mladeni, Janez Brank and Natasa Milic-Frayling. Feature extraction using linear support vector machines. In *Proceedings of the 3rd Int. Conf. on Data Mining Methods and Databases for Engineering, Finance, and Other Fields, Bologna, Italy*, Sept 2002.

Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 77–86, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-391-X. doi: http://doi.acm.org/10.1145/502512.502527.

Bernd Gartner. A subexponential algorithm for abstract optimization problems. In *Proceedings 33rd Symposium on Foundations of Computer Science, IEEE CS Press*, 1992.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

W. Johnson and J. Lindenstauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 1984.

S. S. Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, March 2005.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004. ISSN 1533-7928.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/505282.505283.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995. ISBN 0-387-94559-8.