Margin Distribution and Learning Algorithms

Ashutosh Garg ASHUTOSH@US.IBM.COM
IBM Almaden Research Center, San Jose, CA 95123 USA

Dan Roth DANR@UIUC.EDU
Department of Computer Science, University of Illinois, Urbana, IL 61801 USA

Abstract

Recent theoretical results have shown that improved bounds on generalization error of classifiers can be obtained by explicitly taking the observed margin distribution of the training data into account. Currently, algorithms used in practice do not make use of the margin distribution and are driven by optimization with respect to the points that are closest to the hyperplane.

This paper enhances earlier theoretical results and derives a practical data-dependent complexity measure for learning. The new complexity measure is a function of the observed margin distribution of the data, and can be used, as we show, as a model selection criterion. We then present the Margin Distribution Optimization (MDO) learning algorithm, that directly optimizes this complexity measure. Empirical evaluation of MDO demonstrates that it consistently outperforms SVM.

1. Introduction

The study of generalization abilities of learning algorithms and its dependence on sample complexity is one of the fundamental research efforts in learning theory. Understanding the inherent difficulty of learning problems allows one to evaluate the possibility of learning in certain situations, estimate the degree of confidence in the predictions made, and is crucial in understanding, analyzing, and developing improved learning algorithms.

Recent efforts in these directions (Garg et al., 2002; Langford & Shawe-Taylor, 2002) were devoted to developing generalization bounds for linear classifiers which make use of the actual observed margin distribution on the training data, rather than relying only on the distance of the points closest to the hyperplane (the "margin" of the classifier). Similar results have been obtained earlier for a restricted case, when a convex combination of multiple classifiers is used as the classifier (Schapire et al., 1997). At the heart of these results are analysis techniques that can use an appropriately weighted combination of all the data points, weighted according to their distance from the hyperplane.

This paper shows that these theoretical results can be made practical, be used for model selection, and to drive a new learning algorithm.

Building on (Garg et al., 2002), which introduced the data dependent generalizations bounds, we first develop a more realistic version of the bound, that takes into account the classifier's bias term. An important outcome of (Garg et al., 2002), which we slightly modify here, is a data dependent complexity measure for learning which we call the *projection profile* of the data. The projection profile of data sampled according to a distribution $\mathcal{D}$, is the expected amount of error introduced when a classifier h is randomly projected, along with the data, into a k-dimensional space. Our analysis shows that it is captured by the following quantity: $a_k(\mathcal{D}, h) = \int_{x \in \mathcal{D}} u_k(x) d\mathcal{D}$, where

$$u_k(x) = \min\left(3\exp\left(-\frac{(\nu(x)+b)^2 k}{8(2+|(\nu(x))|)^2}\right), \frac{2}{(\nu(x)+b)^2}, 1\right) \tag{1}$$

and $\nu(x)$ is the distance between x and the classifying hyperplane[1] defined by h, a linear classifier for $\mathcal{D}$ and b is the bias of the classifier. The sequence $\mathcal{P}(\mathcal{D}, h) = (a_1(\mathcal{D}, h), a_2(\mathcal{D}, h), \ldots)$ is the *projection profile* of $\mathcal{D}$.

The projection profile turns out to be quite informative, both theoretically and in practice. (Garg et al., 2002) proved its relevance to generalization performance and used it to develop sample complexity bounds (bounds on generalization error) that are more informative than existing bounds for high dimensional learning problems.

The main contribution of this paper is to show that the projection profile of the observed data with respect to a learned classifier can be directly optimized, yielding a new learning

[1] Our analysis does not assume linearly separable data.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

algorithm for linear classifiers, MDO (Margin Distribution Optimization), that attempts to be optimal with respect to the margin distribution based complexity measure. Specifically, we first argue that this complexity measure can be used for model selection. Empirically, we show that the margin distribution of the data with respect to a classifier behaves differently than the margin and observe that it is both better correlated with its accuracy and is more stable in terms of measurements over the training data and expected values. With this as motivation we develop an approximation to Eqn. 1 that is used to drive an algorithm that learns a linear classifier which directly optimizes this measure. We use several data sets to compare the resulting classifiers to those achieved by optimizing the margin, an in SVM, and show that MDO yields better classifiers.

The paper is organized as follows. In the next section we introduce the notations and some definitions that will be used in later sections. Sec. 3 introduces our enhanced version of the data dependent generalization bound based on the margin distribution of the training data and discusses its implications. In Sec. 4 we show that the projection profile is not only meaningful for the purpose of generalization bounds but can also be used as a model selection criterion. The MDO algorithms is introduced in Sec. 5 and experimental results with it are presented in Sec. 6. We conclude by discussing some open issues.

2. Preliminaries

We study a binary classification problem $f : \mathbb{R}^n \rightarrow \{-1, 1\}$, a mapping from a n dimensional space to class labels $\{-1, 1\}$. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ denotes a sample set of m examples. The hypothesis $h \in \mathbb{R}^n$ is an n-dimensional linear classifier and b is the bias term. That is, for an example $x \in \mathbb{R}^n$, the hypothesis predicts $\widehat{y}(x) = \text{sign}(h^T x + b)$.

We use n to denote the original (high) dimensionality of the data, k to denote the (smaller) dimension into which projections are made and m to denote the sample size of the data. The subscript will refer to the index of the example in the sample set and the superscript will refer to particular dimension that is under consideration.

Definition 2.1 Under 0-1 loss function, the *empirical error* $\widehat{E}$ of h over a sample set S and the *expected error* $\overline{E}$ are given resp. by,

$$\widehat{E}(h, S) = \frac{1}{m} \sum_i I(\widehat{y}(x_i) \neq y_i);$$

$$\overline{E}(h, S) = E_x \left[I(\widehat{y}(x) \neq f(x)) \right],$$

where $I(\cdot)$ is the indicator function which is 1 when its argument is true and 0 otherwise. The expectation E_x is over the distribution of data.

We denote by $||\cdot||$ the L_2 norm of a vector. We will assume w.l.o.g that all data points come from the surface of unit sphere (ie. $\forall x, ||x|| = 1$), and that $||h|| = 1$ (The non unity norm of classifier and data will be accounted for by normalizing the bias b.)

Let $\nu(x) = h^T x$ denote the signed distance of the sample point x from the classifier h. When x_j refers to the j^{th} sample from S, we denote it by $\nu_j = \nu(x_j) = h^T x_j$. With this notation (omitting the classifier h) the classification rule reduces simply to $\text{sign}(\nu(x) + b)$.

Our analysis of margin distribution based bounds is based on results in the area of random projection of high dimensional data, developed in (Johnson & Lindenstrauss, 1984), and further studied in several other works, e.g., (Arriaga & Vempala, 1999). Briefly, the method of random projection shows that with high probability, when n dimensional data is projected down to a lower dimensional space of dimension k, using a random $k \times n$ matrix, relative distances between points are *almost* preserved. We will use this to show that if the data is separable with large margin in a high dimensional space, then it can be projected down to a low dimensional space without incurring much error. Next we give the definition of random matrix:

Definition 2.2 (**Random Matrix**) A *random projection matrix* R is a $k \times n$ matrix whose each entry $r_{ij} \sim N(0, 1/k)$. For $x \in \mathbb{R}^n$, we denote by $x' = Rx \in \mathbb{R}^k$ the projection of x from an n to a k-dimensional space using projection matrix R.

Similarly, for a classifier $h \in \mathbb{R}^n$, h' denotes its projection to a k dimensional space via R, S' denotes the set of points which are the projections of the sample S, and $\nu'_j = (h')^T x'_j$, the signed distance in the projected space.

3. A Margin Distribution based Bound

The decision of the classifier h is based on the sign of $\nu(x) + b = h^T x + b$. Since both h and x are normalized, $|\nu(x)|$ can be thought of as the geometric distance between x and the hyperplane orthogonal to h that passes through the origin. Given a distribution on data points x, this induces a distribution on their distance from the hyperplane induced by h, which we refer to as the *margin distribution*.

Note that this is different from the margin of the sample set S with respect to a classifier h, traditionally defined in the learning community as the distance of the point which is closest to the hyperplane.

$$\text{margin of } S \text{ w.r.t. } h \text{ is} \quad \gamma(S, h) = \min_{i=1}^{m} |h^T x_i + b|.$$

In (Garg et al., 2002), we have developed a data dependent margin bound for the case when the classification is done as $\widehat{y} = \text{sign}(\nu(x))$. Although theoretically one can redefine $x \equiv [x, 1]$ and $h \equiv [h, 1]$ and obtain a similar result, it turns out that in practice this may not be a good idea. The bias term b is related to the distance of the hyperplane from the origin and h is the slope of the hyperplane. Proving the bound, as well as developing the algorithmic approach (based on Eqn. 3 below), require considering a projected version of the hyperplane into a lower dimensional space, k, in a way that does not significantly effects the classification performance. At times, however, the absolute value of b may be much larger than the individual components of h (even after normalization with the norm of h). In these cases, b will contribute more to the noise associated with the projection and it may not be a good idea to project b. Due to this observation, the algorithmic approach that is based on Eqn. 3 necessitates that we prove a slightly modified version of the result given in (Garg et al., 2002).

Theorem 3.1 *Let $S = \{(x_1, y_1), \ldots, (x_{2m}, y_{2m})\}$ be a set of n-dimensional labeled examples and h a linear classifier with bias term b. Then, for all constants $0 < \delta < 1; 0 < k$, with probability at least $1 - 4\delta$, the expected error of h is bounded by*

$$\overline{E}(h) \leq \widehat{E}(S, h) + \min_k \left\{ \mu_k + 2\sqrt{\frac{(k+2)\ln\frac{me}{k+2} + \ln\frac{2}{\delta}}{2m}} \right\} \quad (2)$$

with μ_k redefined as

$$\mu_k = \frac{2}{m\delta} \sum_{j=1}^{2m} \min\left\{ 3\exp\left(-\frac{(\nu_j + b)^2 k}{8(2 + |\nu_j|)^2}\right), \frac{2}{k(\nu_j + b)^2}, 1 \right\}.$$

Note that by choosing $b = 0$, the above result reduces to the one given in (Garg et al., 2002). Although one can prove the above theorem following the steps given in (Garg et al., 2002), we give the sketch of the proof as it is instrumental in developing an intuition for the algorithm.

The bound given in Eqn. 2 has two main components. The first component, μ_k, captures the distortion incurred due to the random projection to dimension k; the second term follows directly from VC theory for a classifier in k dimensional space. The random projection theorem (Arriaga & Vempala, 1999) states that relative distances are (almost) preserved when projecting to lower dimensional space. Therefore, we first argue that, with high probability the image, under projection, of data points that are far from h in the original space, will still be far in the projected (k dimensional) space. The first term quantifies the penalty incurred due to data points whose images will not be consistent with the image of h. That is, this term bounds the additional error incurred due to projection to k dimensional space. Once the data lies in the lower dimensional space, we can bound the expected error of the classifier on the data as a function of the dimension of the space, number of samples and the empirical error there (that is, the first component). Decreasing the dimension of the projected space implies increasing the contribution of the first term, while the VC-dimension based term decreases. To get the optimal bound, one has to balance these two quantities and choose the dimension k of the projected space so that the generalization error is minimized. The following lemma is the key in proving the above theorem; it quantifies the penalty incurred when projecting the data down to k dimensional space. For proof, please refer to (Garg et al., 2002).

Lemma 3.2 *Let h be an n-dimensional classifier, $x \in \mathbb{R}^n$ a sample point, such that $||h|| = ||x|| = 1$, and $\nu = h^T x$. Let $R \in \mathbb{R}^{k \times n}$* **be** *a random projection matrix (Def. 2.2), with $h' = Rh, x' = Rx$. Let the classification is done as $\text{sign}(h^T x + b)$. Then the probability of misclassifying x, relative to its classification in the original space, due to the random projection, is*

$$P\left[\text{sign}(h^T x + b) \neq \text{sign}(h'^T x' + b)\right] \leq \min\left\{ 3\exp\left(-\frac{(\nu + b)^2 k}{8(2 + |\nu|)^2}\right), \frac{2}{k(\nu + b)^2}, 1 \right\}.$$

The above lemma establishes a bound on the additional classification error that is incurred when projecting the sample down from an n to a k dimensional space. Note that in this formulation, unlike the one in (Garg et al., 2002)) we only project h while keeping b as in the original space. It can be shown that in many cases this bound is tighter than the one given in (Garg et al., 2002).

Once the above result is established, one can prove the theorem by making use of the symmetrization lemma (Anthony & Bartlett, 1999) and standard VC-dimension arguments (Vapnik, 1998). See (Garg et al., 2002) for details.

3.1. Existing Learning Algorithms

In this section, we attempt to provide some intuition to the significance of considering the margin distribution when selecting a classifier. We note that the discussion is general although done in the context of linear classifiers. It has been shown that any classifier can be mapped to a linear classifier in high dimensional space. This suggests that analysis of linear classifiers is general enough and can give meaningful insights. The selection of a classifier from among a set of classifiers that behaves well on the training data (with respect to some complexity measure) is based Vapnik's (Vapnik, 1998) structural risk minimization principle. The key question then is, what is the measure that should guide this selection.

A cartoon example is depicted in Fig. 1. Training data with

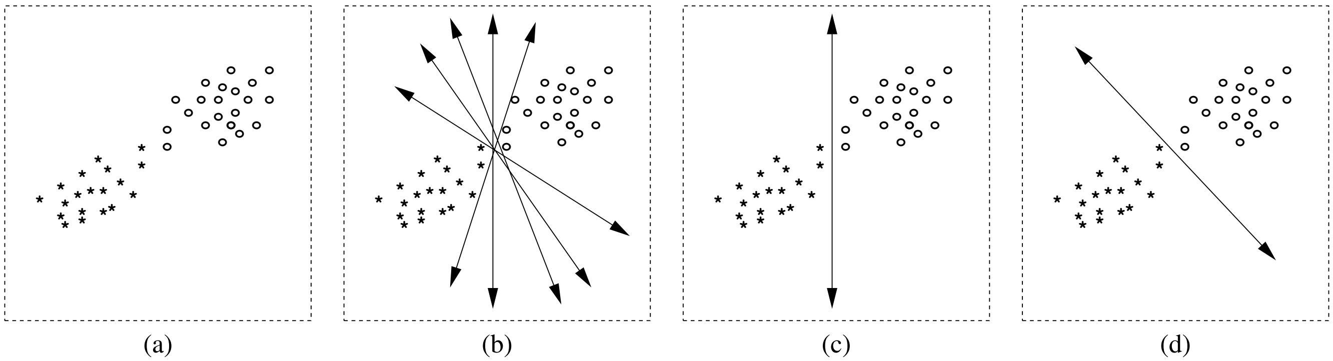


Figure 1. (a) A cartoon showing the training data in two dimensional space with “$*$” corresponding to positive class and “o” corresponding to negative class. (b) Linear classifiers that can classify the data without error. (c) The hyperplane that learned by a large margin classifier such as SVM. (d) This hyperplane may be a better choice than the one learned by SVM, since more data points are further apart from the hyperplane.

“$*$” corresponds to positive and with “o” corresponds to negative data. Clearly, there are a number of linear classifiers that can classify the training data perfectly. Fig. 1(b) shows some of these classifiers. All these classifiers have zero error on the training data and as such any selection criteria based on training error only will not distinguish between them.

One of the popular measure that guides the selection of classifier is the margin of a hyperplane with respect to its training data. Clearly, however, this measure is sensitive since it typically depends on a small number of extreme points. This can be significant not only in the case of noisy data, but also in cases of linearly separable data, since this selection may effect performance on future data. In the context of large margin classifiers such as SVM, researchers tried to get around this problem by introducing slack variables and ignoring some of the closest points in an ad hoc fashion, but this extreme notion of margin still drives the optimization. The cartoon in Fig. 1(d) shows a hyperplane that was chosen by taking into account all the data points rather than those that are closest to the hyperplane. Intuitively, this looks like a better choice than the one in Fig. 1(c) (which would be chosen by a large margin classifier) , by the virtue of being a more stable measure. In the next section, we argue that one can use projection profile as a model selection criterion and show that this hyperplane is the one that would actually be selected if this model selection criteria is used.

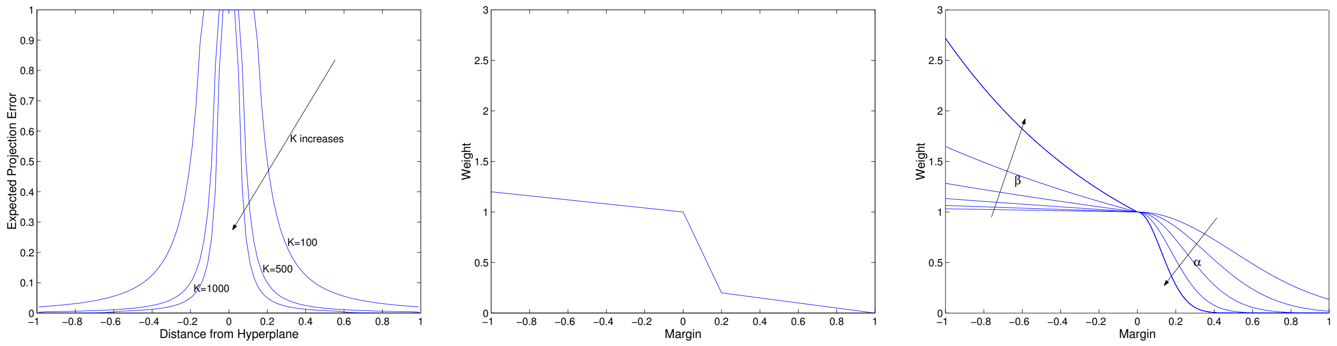
Dataset	C:Margin-E	C:ProjProfile-E
Pima	0.3734	0.7913
Breast Cancer	0.4558	0.6162
Ionosphere	0.1753	0.3490
Credit	0.1215	0.3147

Table 1. The first column (C:Margin-E) gives the correlation coefficient between the Margin of the training data and the error on the test set. The second column (C:ProjProfile-E) gives the correlation coefficient between the weighted margin according to the cost function given in Eqn. 5 and the error on test set; all done for four datasets from UCI ML Repository. When computing the margin we look at only correctly classified points.

4. Algorithmic Implications

The expected probability of error for a k-dimensional image of a point x that is at distance $\nu(x)$ from an n-dimensional hyperplane (where the expectation is with respect to selecting a random projection matrix) is given by the projection profile in Eqn. 1. This expression measures the contribution of a point to the generalization error as a function of its distance from the hyperplane (for a fixed k). Figure 2(a) shows this term (Eqn. 1) for different values of k. All points contribute to the error, and the relative contribution of a point decays as a function of its distance from the hyperplane. Hypothetically, given a probability distribution over the instance space and a fixed classifier, one can compute the margin distribution which can then be used to compute the projection profile of the data.

The bound given in Thm. 3.1 depends only on the projection profile of the data and the projection dimension and is independent of the particular classifier. Of all the classifiers with the same training error, the one with the least projection profile k value will have the best generalization performance. Therefore, the Theorem suggests a model selection criterion. Of all the classifiers, choose the one that optimizes the projection profile and not merely the margin. A similar argument, but for a slightly restricted problem, was presented by (Mason et al., 2000). They argued that when learning a convex combination of classifiers, instead of maximizing the margin one should maximize a



(a)

(b)

(c)

Figure 2. (a)The contribution of data points to the generalization error as a function of their distance from the hyperplane. (b) The contribution of data points to the generalization error as a function of their distance from the hyperplane as given by (Mason et al., 2000) (c) The weight given to the data points by the MDO algorithm as a function of there margin.

weighted function of the distance of the points from the learned classifier. This was used to develop margin distribution based generalization bound for *convex combination* of classifiers. The weighting function that was proposed is shown in Fig. 2(b). The results of their algorithm showed that a weighted margin distribution (for convex combination of classifiers) is a better complexity measure than the extreme notion of margin which has been typically used in large margin classification algorithms like SVM. Although the problem addressed there is different from ours, interestingly, the form of their weighting function (shown in Fig. 2(b), discussed later) is very similar to ours.

In our work, instead of working with the actual projection profile, we work with a simpler form of it given in Eqn. 3.

$$u(x) = \exp(-(\nu(x) + b)^2 k) \tag{3}$$

The advantage of using the projection profile as a model selection criteria is evident from the experiments presented in Table 1. In particular, we took four data sets from UCI ML repository. For each data set, we randomly divided it into a test and a training data set. A classifier was learned on the training set and was evaluated on the test data. This experiment was repeated 200 times (each time choosing a different subset as training data). Each time we computed the actual margin of the SVM that was used as the initial guess and the weighted margin according to a cost function, given in Eqn. 5, for this classifier. Table 1 gives the correlation between the generalization error of the classifier (its performance on the test data) and these quantities. The first column in Table 1 gives the correlation between the error on test data and the margin on training data. The second column gives the correlation between the error on the test data and the weighted margin (projection profile) computed on the training data. Note that here we are looking at the absolute value of the correlation coefficient. The correlation results highlights the fact that our new cost function is a better predictor of generalization error. Further evidence is presented in Table 2. It presents empirical evidence to show that the cost function given in Eqn. 5 is a more stable measure than the margin. The first column in Table 2 gives the correlation between the margin (which is computed as the distance of the closest correctly classified point from the hyperplane) on the training and test data over 200 runs. The second column gives the correlation between the weighted margin (projection profile) computed on the training data and test data. Again, a high correlation in the latter case indicates the stability of this new measure.

5. The MDOAlgorithm

In this section, we introduce a new learning algorithm, the *Margin Distribution Optimization* (MDO) algorithm. MDO is driven by optimizing with respect to the simplified form of projection profile given in Eqn. 3. The learning algorithm attempts to find a classifier that minimizes a cost function which happens to be the projection profile of the data. The cost function in Eqn. 3 determines the contribution we want our selected classifier to give to the data points. In order to choose a classifier that optimizes with respect to this measure, though, we need to change it in order to take into account misclassified points. We do that by defining a new cost function which explicitly gives higher weights to misclassified points. Thus, the weight given to an example x_i with true class label y_i and learned hyperplane h with bias b is given by

$$W(x_i) = \begin{cases} e^{-\alpha(\nu(x_i)+b)^2} & \text{if } y_i(\nu(x_i)+b) \geq 0, \\ e^{-\beta y_i(\nu(x_i)+b)} & \text{if } y_i(\nu(x_i)+b) < 0. \end{cases} \tag{4}$$

That is, for correct classification, the weight is the one given by the projection profile. However, for misclassified points, the example is exponentially weighted. The constants α, β allows one to control the weight of these two

Dataset	C:Margin Tr-Te	C:ProjProfile Tr-Te
Pima	0.5763	0.8225
Breast Cancer	0.2782	0.5375
Ionosphere	0.4312	0.4618
Credit	0.3208	0.6201

Table 2. The first column (C:Margin Tr-Te) gives the correlation coefficient between the margin on the training data and the margin on the test data. The second column (C:ProjProfile Tr-Te) gives the correlation coefficient between weighted margin according to cost function given in Eqn. 5 on the training data and on the test data for four datasets from UCI ML Repository. When computing the margin we look at only correctly classified points.

terms. α should be thought of as the optimal projection dimension and could be optimized (although at this point our algorithms determines it heuristically). Fig. 2(c) shows a family of weight functions as a function of the margin $\nu(x) + b$ for an example. In this figure, a positive margin means correct classification and negative margin corresponds to misclassification. The weight given to points which were classified with good margin is very low while points which are either misclassified or very close to the margin get higher weights. Minimizing this cost function will essentially try to maximize the weighted margin while reducing the misclassification error. Note the close resemblance to the weighting function proposed by Mason et. al. for the convex combination case, shown in Fig. 2(b).

Given a training set S of size m, the optimization problem can be formulated as: Find (h, b) such that the following cost function is minimized.

$$\begin{aligned}\mathcal{L}(h,b) &= \sum_{i=1}^{m} I(\widehat{y}_i = y_i) e^{-\alpha(\nu(x_i)+b)^2} \\ &\quad + \sum_{i=1}^{m} I(\widehat{y}_i \neq y_i) e^{-\beta y_i(\nu(x_i)+b)} \qquad (5)\end{aligned}$$

Here $I(\cdot)$ is an indicator function which is 1 when its argument is true otherwise 0. α is directly proportional to the concept of projection dimension and β is a related to the tradeoff between the misclassification error and the projection profile. There are a few observations that needs to be made before we proceed further. In most practical cases, learning is done with non-separable training data. Standard learning algorithms like SVM handle this case by introducing slack variables and the optimal values of the slack variables are chosen by a search procedure. This makes the performance of the algorithm dependent on the particular search method. Our formulation automatically takes care of misclassified points. While minimizing the above cost function, it gives larger penalty to points which are in error as against the ones which are correctly classified. The amount of error that will be tolerated can be controlled by choosing α, β appropriately. One possible way to choose these parameters is by evaluating the learned classifier over some hold out set. In our case, we observed that in most cases $\alpha = \frac{1}{(\overline{\nu}+b)^2}$ and $\beta = \frac{1}{(\overline{\nu}+b)}$ gave good results where $\overline{\nu}$ is an estimate of average margin given by $\overline{\nu} = \sum |\nu_i|/m$ for a data set of size m for some h.

The main computational difficulty in our algorithm lies in the non-convexity of the objective function. In general, one would like to avoid solving a non-convex problem as it is hard to obtain a globally optimal solution and there is no closed form expression. Nevertheless we show that gradient descent methods are quiet useful here. Due to non-convexity, there are a number of local saddle points and one may not reach a global optimal point. This makes the choice of starting point extremely critical. In our work, we suggest to use SVM learning algorithm to choose the initial starting classifier. Once an initial classifier is obtained, the algorithm uses gradient descent over the cost function given in Eqn 5. The algorithm stops once it has reached a local minima of the cost function. Two important points that deserves attention are (1) the norm of the classifier h and (2) the non-differentiability of the cost function. In general, by increasing the norm of the classifier, we can decrease the value of the cost function (as the margin will increase). Therefore, we don't take into account the norm of the classifier h in the cost function and at each iteration, after obtaining the updated vector h, we re-normalize it. The second issue is more tricky. The derivative of the cost function is not defined for those x_i which have 0 margin i.e. $\nu(x_i) + b = 0$. At such points, we compute both the left and right derivatives. All the gradient directions are then evaluated and those which lead to maximum decrease in the cost function is selected. The algorithm terminates if no such gradient direction are found.

The algorithm given in Fig. 7 gives the implementation details. It starts with learning a classifier using SVM learning algorithm obtaining (h, b). We assume[2] that $||h|| = 1$. α, β are then initialized (as above). MAX_CONST - maximum number of points with zero margin that will be considered at any given iteration of the algorithm. Since at points with zero margin, we are evaluating both the left and right derivative, the number of directions considered is exponential in the number of points with zero margin, and as such we limit the number of such points (for which both gradient directions are considered) to MAX_CONST. MDO is the main procedure. If there are no points with 0 margin, the algorithm simply computes the gradients as:

$$\begin{aligned}\frac{\partial \mathcal{L}(h,b)}{\partial h_k} &= -2\alpha \sum_{i=1}^{m} I(\widehat{y}_i = y_i)(\nu(x_i) + b) x_i^k e^{-\alpha(\nu(x_i)+b)^2} \\ &\quad -\beta \sum_{i=1}^{m} I(\widehat{y}_i \neq y_i) y_i x_i^k e^{-\beta y_i(\nu(x_i)+b)} \qquad (6)\end{aligned}$$

[2] If $||h|| \neq 1$, define $h = h/||h||$, $b = b/||h||$. The classification result will be unchanged.

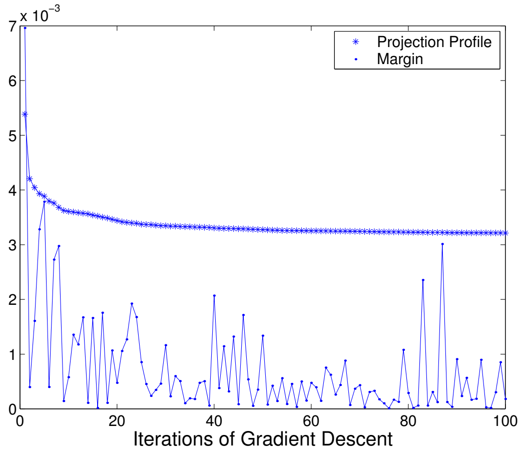


Figure 3. The plot gives the experimental results for the breast cancer dataset from the UCI ML repository. It gives the projection profile "$*$" and margin "$\cdot$" of the learned classifier on the training data as the algorithm iterates.

These derivatives are then used by procedure *Update* to obtain the updated values of (h, b) given by (h', b'). While updating, we make sure that cost function is decreasing at each iteration. However, if there are points with 0 margin, we randomly select of MAX_CONST such points, and at those we compute all the possible gradients (computing both left and right derivatives). The other points with zero margin are ignored. Once we have the derivative of the cost function, the same update procedure, as above, is called. While doing the experiments, we observed that the algorithm typically converges in a small number of steps.

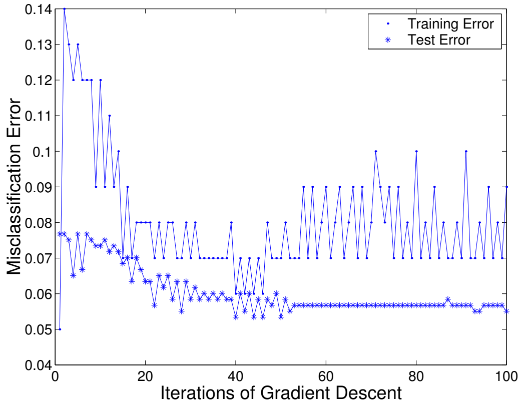


Figure 4. The plot gives the experimental results for the breast cancer dataset from the UCI ML repository. This plot gives the training error "$\cdot$" and test error "$*$" at each iteration. With the iterations, training error increased from the starting point - 0th iteration while test error went down.

Datasets	SVM	MDO	% Reduction
Credit (UCI)	94.04	95.83	30.03
Breast (UCI)	91.71	92.93	14.72
Ionosphere (UCI)	94.47	95.55	19.53
Heart (UCI)	79.72	80.94	6.02
Liver (UCI)	83.31	84.13	5.99
Mushroom (UCI)	96.1	96.42	8.21
peace & piece	89.26	91.72	22.91
being & begin	96.04	96.75	17.92

Table 3. Results comparing the performance of SVM and MDO. The first six datasets are from UCI ML repository. The last two datasets are from the context sensitive spelling correction problem. The column SVM gives the % correct classification for test data when a classifier learned using SVM was used and column MDO gives the same for MDO. The last column gives the percentage reduction in error from SVM to MDO. As seen, in all cases, MDO's results are comparable or better than the ones obtained using SVM. These results are averaged over 100 runs.

6. Results

In this section, we provide the empirical comparison of the MDO with SVM. Note that although, in our experiments, MDO's starting point is the SVM classifier, the results obtained by MDO could be worse than SVM. Indeed, the selection of the new classifier is driven by our new optimization function and it could lead to larger error. Interestingly, when doing the experiments we observed that in many cases we obtained better generalization at the cost of slightly poor performance on the training data. To understand this we analyzed in detail the results for the breast cancer dataset from UCI ML repository. In Fig. 3 we plot the margin and the value of the cost function given in Eqn. 5 over the gradient descent iterations. As expected, while the cost function of MDO improved (decreased in value) the margin deteriorated (since we started with SVM, the margin at 0^{th} iteration is maximal). Fig. 4 is even more revealing. We show how the training and test error changed with the iterations of the gradient descent. Interestingly, the error on the training error went up from 5% at 0^{th} iteration to 9%. The algorithm traded the increase in training error with the decrease in projection profile. The second plot gives the test error and as evident, the test error went down with more iterations. The gap between the training error and test error also decreased which shows that the new classifier generalized well and the cost function proposed in Eqn. 5 is a better measure of learning complexity.

In Table 3, we compare the classification performance of MDO and SVM on a number of datasets from UCI ML repository and on two real world problems related to context sensitive spelling correction. The spelling data is taken from the *pruned* data set of (Golding & Roth, 1999). Two particular examples of context sensitive spelling correction, *being* & *begin* and *peace* & *piece*, are considered. For details on the dataset see (Murphy, 1994). In all experiments, data was randomly divided into training (60%), and test set (40%). The experiments were repeated 100 times each time choosing a different subset (randomly chosen) of training and test data. It is evident that except for a few case (in which the improvement is not statistically significant), in most cases MDO outperforms SVM.

7. Conclusions

We presented a new, practical, margin distribution based complexity measure for learning classifiers that we derived by enhancing a recently developed method for developing margin distribution based generalization bounds. We have shown that this theoretically justified complexity measure can be used robustly as a model selection criterion and, most importantly, used it to drive a new learning algorithm for linear classifiers, that is selected by optimizing with respect to this complexity measure. The results are based on a novel use of the margin distribution of the data relative to the learned classifier, that is different than the typical use of the notion of margin in machine learning. Consequently, the resulting algorithm is not sensitive to small number of samples in determining the optimal hyperplane.

Although the bound presented here is tighter than existing bounds and is sometimes informative for real problems, it is still loose and more research is needed to match observed performance on real data. Algorithmically, although we have given an implementation of the new algorithm MDO, one of the main direction of future research is to study it further, as well as to investigate other algorithmic implications of the ideas presented here.

Acknowledgments: This research is supported by NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168 and an IBM fellowship to Ashutosh Garg.

```
Here we initialize the constants and obtain initial
estimate of the classifier
Global :
Training Set: $S = \{(x_1, y_1), ..., (x_m, y_m)\}$
Classifier:   Learn $(h, b)$ using linear SVM
              Where $h, b \; : \; y_i(h^T x_i + b) > 0$ (for all
              the correctly classified training examples.)
Constants:    MAX_CONST,$\alpha$,$\beta$,$c^{\inf}$
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
This is the main function
Procedure MDO($h, b, S$)
 For all $(x_i, y_i) \in S$, $f_h(x_i) = y_i(h^T x_i + b)$
 $C(h, b) = \sum_{i:f_h(x_i) \geq 0} e^{-\alpha f_h(x_i)^2} + \sum_{i:f_h(x_i) < 0} e^{-\beta f_h(x_i)}$
 $A \; := \; \{(x_i, y_i) \in S : f_h(x_i) = 0\}$ (points with $0$ margin)
 if ($A = \phi$)        No points with $0$ margin
   $g_h := -\nabla_h C(h, b)$;  $g_b := -\nabla_b C(h, b)$
   Update ($h', b', h, b, g_h, g_b$)
 else          $A$ is a set of points with $0$ margin
   if ($|A| >$MAX_CONST) then $B$ is random subset
   of $A$ of size MAX_CONST, else $B = A$
   $N = |B|$
   for each $(b_1, ..., b_N) \in \{\pm 1\}^N$
    $\{g_h^{(b_1,..,b_N)}, g_b^{(b_1,..,b_N)}\} = -\{\nabla_h^{(b_1,..,b_N)}, \nabla_b^{(b_1,..,b_N)}\} C(h, b)$;
    where $\nabla^{(b_1,...,b_N)}$ means the left or right derivative at
    the $i^{th}$ example in $B$ according as $b_i = +1$ or $-1$.
    Update ($h', b', h, b, g_h, g_b$)
    $c^{(b_1,...,b_N)} = C(h, b) - C(h', b')$
   end
   Choose $h', b'$ which had the maximum reduction in cost
 end
 if $C(h, b) - C(h', b') < c^{\inf}$, then BREAK
 else $h = h'$; $b = b'$; MDO($h, b, S$)
End
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
This procedure gives the updated parameters based on the
old parameters and gradients
Procedure Update($h', b', h, b, g_h, g_b$)
 $h' = h + \epsilon g_h, b' = b + \epsilon g_b$; $h' = h'/||h'||, \; b' = b'/||h'||$
 where $\epsilon$ is maximum positive real number
 such that $C(h, b) - C(h', b') \geq 0$.
end
```

Figure 5. Sketch of the Margin Distribution Optimization (MDO) Algorithm

References

Anthony, M., & Bartlett, P. L. (1999). *Neural network learning: Theoretical foundations*. Cambridge.

Arriaga, R. I., & Vempala, S. (1999). An algorithmic theory of learning: Robust concepts and random projection. *Proc. of the 40th Foundations of Computer Science* (pp. 616–623).

Garg, A., Har-Peled, S., & Roth, D. (2002). On generalization bounds, projection profile, and margin distribution. *Proc. of the International Conference on Machine Learning*.

Golding, A. R., & Roth, D. (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, *34*, 107–130.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Conference in modern analysis and probability* (pp. 189–206).

Langford, J., & Shawe-Taylor, J. (2002). Pac-Bayes and margins. *Advances in Neural Information Processing Systems*.

Mason, L., Bartlett, P., & Baxter, J. (2000). Improved generalization through explicit optimization of margins. *Machine Learning*, *38-3*, 243–255.

Murphy, P. M. (1994). *UCI repository of machine learning databases [http://www.ics.uci.edu/ mlearn]* (Technical Report). Irvine, CA: University of California.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. *Proc. 14th International Conference on Machine Learning* (pp. 322–330). Morgan Kaufmann.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: John Wiley and Sons Inc.