
Predicting the Probability of Correct Classification

Gregory Z. Grudic

Department of Computer Science
University of Colorado, Boulder
grudic@cs.colorado.edu

Abstract

We propose a formulation for binary classification, called the Probabilistic CDF algorithm, that both makes a classification prediction, and estimates the probability that the classification is correct. Our model space consists of the widely used basis function models (which includes support vector machines and other kernel classifiers). Our formulation is based on using existing algorithms (such as SVM) to build the classifiers, and therefore achieves state of the art classification accuracy, while at the same time giving accurate estimates *point specific* probabilities of correct classification. We further demonstrate that the Probabilistic CDF algorithm can improve the overall classification accuracy of an existing basis function classifier by appropriate local modification the decision boundary. Our framework is theoretically justified and empirical evaluations show promising results.

1 Introduction

Numerous binary classifier learning algorithms have been proposed in the literature [4]. These have demonstrated excellent classification accuracy on a wide range of complex problem domains. However, most of these classifiers simply give hard *YES* or *NO* outputs, and make no attempt at estimating how accurate individual predictions are. Knowing how accurate individual classification predictions are can be useful in a variety of applications, ranging from medical diagnostics (i.e. how accurate is the prediction that I have cancer) to weather forecasting.

This paper proposes a new formulation for building binary classifiers that, for each new observation (or instance), output both a class prediction as well as a probability that the prediction is correct. The formulation is based on widely used basis function models and includes Support Vector Machines [8], as well as other widely used classifiers [4]. The proposed algorithm, called the *Probabilistic CDF* algorithm, uses existing state of the art classification algorithms (e.g. SVMs [8]) as a starting point, adding estimates of the probability of correct classification without undermining overall classification accuracy. The *Probabilistic CDF* algorithm works by estimating the cumulative distribution functions (*cdf*) of the positive and negative examples in training data, locally around the point being classified. These *cdf*s are then used to obtain local estimates of the correct classification, as well as to locally modify the decision boundary to *improve* the global classification accuracy.

Related work includes Platt’s [7] Sigmoid+SVM algorithm which fits a sigmoid to the data around the decision boundary (and other related algorithms [9]). This sigmoid is used to predict the probability of class membership, by making it a function of the distance of a test example to a boundary. Because this probability is independent of where the point is in input space (only how far it is to the boundary), it cannot adequately model problems such as those depicted in Figure 1a where this probability changes *along* the decision boundary (see Section 4.1 for details). In contrast, the approach proposed here obtains probabilities that do depend on which part of the decision boundary the test point is closest to.

Section 2 presents the theoretical development. Section 3 gives implementation details and the experimental results are presented in Section 4. The conclusion is given in Section 5.

2 Theoretical Formulation

Assume a d dimensional binary classification problem with classes U and V . Elements of U are independently, identically distributed (iid) according to a probability density function $f_u(\mathbf{x})$, where $\mathbf{x} \in \mathfrak{R}^d$. Similarly, elements of V are independently, identically distributed (iid) according to a probability density function $f_v(\mathbf{x})$, where $\mathbf{x} \in \mathfrak{R}^d$. The classes U and V are assumed independent, and the prior probability of class U is given by $\Pr(U)$, while the prior probability of class V is given by $\Pr(V)$, with $\Pr(U) + \Pr(V) = 1$. Finally, we assume that $f_u(\mathbf{x})$ and $f_v(\mathbf{x})$ are bounded, strictly positive and continuous (and integrate to 1 as with standard pdf functions).

We consider a general class of binary classifiers of the form

$$y = \sum_{i=1}^k a_i \phi_i(\mathbf{x}) + b \begin{cases} y \geq 0 \Rightarrow \hat{c} = U \\ y < 0 \Rightarrow \hat{c} = V \end{cases} \quad (1)$$

where \hat{c} is the predicted classification of \mathbf{x} ; $\phi_i(\mathbf{x}) : \mathfrak{R}^d \rightarrow \mathfrak{R}$, $i \in \{1, 2, \dots, k\}$, are k basis functions, which we assume are continuous and bounded; $a_i \in \mathfrak{R}$, $i \in \{1, 2, \dots, k\}$, $b \in \mathfrak{R}$; and $y \in \mathfrak{R}$. Therefore, our classification function can be written as:

$$\hat{c} = \text{sgn} [\Phi(\mathbf{x}) \mathbf{a}^T + b]$$

where $\mathbf{a} = (a_1, \dots, a_k)$ and $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$. Given this formulation, we next define two infinite sets in \mathfrak{R}^d :

$$S_u = \left\{ \mathbf{x} \in \mathfrak{R}^d \mid y = \sum_{i=1}^k a_i \phi_i(\mathbf{x}) + b \geq 0 \right\}$$

$$S_v = \left\{ \mathbf{x} \in \mathfrak{R}^d \mid y = \sum_{i=1}^k a_i \phi_i(\mathbf{x}) + b < 0 \right\}$$

The classifier (1) classifies all points $\mathbf{x} \in S_u$ as being part of class U , and all points $\mathbf{x} \in S_v$ as being part of class V . If the basis functions $\phi_i(\mathbf{x})$ are kernel functions, then (1) is a Support Vector Machine (SVM) [8], otherwise (1) belongs to the widely used class of superposition of basis function classifiers [4]. As defined below, the general property of these classifiers exploited here is that they form linear hyperplane boundaries for binary classification in the k dimensional space of basis functions $(\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$.

Given these assumptions, the probability that (1) *correctly* classifies any point $\mathbf{x} \in \mathfrak{R}^d$ is given by:

$$\Pr(\hat{c} = c) = \Pr(U) \Pr(y \geq 0 | U \mapsto \mathbf{x}) + \Pr(V) \Pr(y < 0 | V \mapsto \mathbf{x}) \quad (2)$$

where $U \mapsto \mathbf{x}$ symbolizes that the x was generated by class U (i.e. according to the density function $f_u(\mathbf{x})$), and $V \mapsto \mathbf{x}$ symbolizes that the x was generated by class V (i.e. according to the density function $f_v(\mathbf{x})$). Thus, $\Pr(y \geq 0 | U \mapsto \mathbf{x})$ and $\Pr(y < 0 | V \mapsto \mathbf{x})$ are defined as:

$$\Pr(y \geq 0 | U \mapsto \mathbf{x}) = \int_{S_u} f_u(\mathbf{x}) d\mathbf{x}, \quad \Pr(y < 0 | V \mapsto \mathbf{x}) = \int_{S_v} f_v(\mathbf{x}) d\mathbf{x}$$

It is important to note that equation (2) defines a *global* probability of correct classification for *all* points generated according to the probability density functions $f_u(\mathbf{x})$ and

$f_v(\mathbf{x})$. This paper deals with *local* probabilities of correct classification symbolized by $\Pr(\hat{c} = c | \hat{c} = \Phi(\mathbf{x})\mathbf{a}^T + b)$, and the remainder of this section is devoted to rewriting equation (2) into a form that allows estimation of this local probability. This type of local probability of correct classification has an interesting form in basis function space which can be computationally exploited. To show this, we introduce some notation to simplify the presentation. Let $z_i = \phi_i(\mathbf{x})$ and $\mathbf{z} = \Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$. Let \mathbf{Z}^* be the set of all points $\mathbf{z}^* \in \mathbf{Z}^*$ that fall on the separating hyperplane in basis function space:

$$y = 0 = \sum_{i=1}^k a_i z_i^* + b \quad (3)$$

For each point $\mathbf{z}^* \in \mathbf{Z}^*$ define a perpendicular to this plane in parametric form:

$$\mathbf{z} = \mathbf{a}t + \mathbf{z}^* \quad (4)$$

where $\mathbf{a} = (a_1, \dots, a_k)$ are the normals of the hyperplane (they are also the coefficients in (1)), and $t \in \mathfrak{R}$, with $t = 0$ define points on the hyperplane. Let $\mathbf{X}_{\mathbf{z}^*}$ be the set of all points $\mathbf{x} \in \mathbf{X}_{\mathbf{z}^*}$ that fall on the line in (4). We now define two functions in $\mathbf{z}^* \in \mathbf{Z}^*$ basis function space:

$$g_u(\mathbf{z}^*) = \int_{\mathbf{X}_{\mathbf{z}^*}} f_u(\mathbf{x}) d\mathbf{x}, \quad g_v(\mathbf{z}^*) = \int_{\mathbf{X}_{\mathbf{z}^*}} f_v(\mathbf{x}) d\mathbf{x}$$

Thus, $g_u(\mathbf{z}^*)$ integrates the density function of class U along an infinite line in basis function space, while $g_v(\mathbf{z}^*)$ integrates the density function of class V along the same infinite line. Finally, let $\mathbf{X}_{\mathbf{z}^*}^{t_0}$ be the set of all points $\mathbf{x} \in \mathbf{X}_{\mathbf{z}^*}^{t_0}$ that fall on the perpendicular line in (4) with $t \leq t_0$. We now define two *cumulative density functions* for classes U and V along the lines in (4) as follows:

$$G_u(t_0 | \mathbf{z}^*) = \frac{1}{g_u(\mathbf{z}^*)} \int_{\mathbf{X}_{\mathbf{z}^*}^{t_0}} f_u(\mathbf{x}) d\mathbf{x}, \quad G_v(t_0 | \mathbf{z}^*) = \frac{1}{g_v(\mathbf{z}^*)} \int_{\mathbf{X}_{\mathbf{z}^*}^{t_0}} f_v(\mathbf{x}) d\mathbf{x}$$

The above equation allows us to rewrite (2) into a form specific to basis function space. This form is defined in the following theorem:

Theorem 1: *Given the above assumptions, the global probability of correct classification, as defined in equation (2), can be expressed as:*

$$\Pr(\hat{c} = c | t_0 = 0) = \Pr(U) \int_{\mathbf{Z}^*} g_v(\mathbf{z}^*) G_u(t_0 = 0 | \mathbf{z}^*) d\mathbf{z}^* + \Pr(V) \int_{\mathbf{Z}^*} g_v(\mathbf{z}^*) (1 - G_v(t_0 = 0 | \mathbf{z}^*)) d\mathbf{z}^* \quad (5)$$

Proof Sketch: The key to the proof is setting $t_0 = 0$, which defines the hyperplane classification boundary that occurs at $t = 0$. The rest of the proof is follows directly from the above equations and the assumptions on $f_u(\mathbf{x})$ and $f_v(\mathbf{x})$. This completes the proof sketch.

From *Theorem 1* we can directly obtain our goal of local probability estimates $\Pr(\hat{c} = c | \hat{c} = \Phi(\mathbf{x})\mathbf{a}^T + b)$ by removing the integral and looking at a specific point \mathbf{x} . Specifically, we obtain the probability of correct classification for all points $\mathbf{x} \in \mathbf{X}_{\mathbf{z}^*}$ that fall on the subdomain defined by the line perpendicular to the hyperplane (and passing through \mathbf{z}^* as defined in (4)) as follows:

$$\Pr(\hat{c} = c | \hat{c} = \text{sgn}[\Phi(\mathbf{x})\mathbf{a}^T + b], t_0 = 0) = \Pr(U) g_v(\mathbf{z}^*) G_u(t_0 = 0 | \mathbf{z}^*) + \Pr(V) g_v(\mathbf{z}^*) (1 - G_v(t_0 = 0 | \mathbf{z}^*)) \quad (6)$$

Note that for any point $\mathbf{x} \in \mathbf{X}_{\mathbf{z}^*}$, the probability of correct classification is the same because classification is defined by the hard threshold $\hat{c} = \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b)$. Here we note that $t_0 = 0$ may not be the optimal boundary for separating the classes in the subdomain $\mathbf{x} \in \mathbf{X}_{\mathbf{z}^*}$. In fact, every line $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$ in basis function space may have a different optimal separating point. We can find the optimal point t^* for the perpendicular line at \mathbf{z}^* (and hence for any specific point $\mathbf{x} \in \mathfrak{R}^d$) as follows

$$\max_{t^*} \{ \Pr(\hat{c} = c | \hat{c} = \text{sgn}[\Phi(\mathbf{x})\mathbf{a}^T + b], t_0 = t^*) \} = \max_{t^*} \{ \Pr(U) g_v(\mathbf{z}^*) G_u(t_0 = t^* | \mathbf{z}^*) + \Pr(V) g_v(\mathbf{z}^*) (1 - G_v(t_0 = t^* | \mathbf{z}^*)) \} \quad (7)$$

Therefore, the optimal separating boundary changes from that defined in equation (1), and is given by:

$$y = \sum_{i=1}^k a_i \varphi_i(\mathbf{x}) + b \begin{cases} y \geq t^* \Rightarrow \hat{c} = U \\ y < t^* \Rightarrow \hat{c} = V \end{cases} \quad (8)$$

The optimal overall probability of correct classification given this locally modified boundary $t = t^*$ is give by:

$$\Pr(\hat{c} = c | t_0 = t^*) = \frac{\Pr(U) \int_{\mathbf{z}^*} g_v(\mathbf{z}^*) G_u(t_0 = t^* | \mathbf{z}^*) d\mathbf{z}^* + \Pr(V) \int_{\mathbf{z}^*} g_v(\mathbf{z}^*) (1 - G_v(t_0 = t^* | \mathbf{z}^*)) d\mathbf{z}^*}{\Pr(U) \int_{\mathbf{z}^*} g_v(\mathbf{z}^*) G_u(t_0 = t^* | \mathbf{z}^*) d\mathbf{z}^* + \Pr(V) \int_{\mathbf{z}^*} g_v(\mathbf{z}^*) (1 - G_v(t_0 = t^* | \mathbf{z}^*)) d\mathbf{z}^*} \quad (9)$$

To complete the formulation we present the following theorem:

Theorem 2: *Given the above assumptions, and assuming that $\Pr(U)$, $\Pr(V)$, g_u , G_u , g_v and G_v are known exactly, the locally modified boundary produced a classifier that is at least as good as the original boundary - i.e. $\forall \mathbf{x} \in \mathbb{R}^d$, $\Pr(\hat{c} = c | t = t^*) \geq \Pr(\hat{c} = c | t = 0)$.*

Proof: From equation (7), at any point $\mathbf{x} \in \mathbb{R}^d$, $\Pr(\hat{c} = c | \hat{c} = \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*) \geq \Pr(\hat{c} = c | \hat{c} = \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = 0)$. It therefor trivially follows that this is true for all points $\mathbf{x} \in \mathbb{R}^d$. This completes the proof.

The importance of the above *Theorem 2* is that a classifier with a locally modified boundary can improve on the hyperplane defined in equation (1). This is supported experimentally in the results section.

In this paper we implement an algorithm that estimates t^* for every point \mathbf{x} by maximizing (7), and then uses this maximum to estimate the local probability of correct classification $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*)$. The numerical implementation of this is presented next.

3 Numerical Implementation of the Algorithm

We assume a set of N training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$, where $y_i \in U, V$. If $y_i = U$, then \mathbf{x}_i was generated according to the density function $f_u(\mathbf{x})$, otherwise \mathbf{x}_i was generated according to the density function $f_v(\mathbf{x})$ (see previous section for details on distribution assumptions on classes U and V).

In order to use (7) to both classify and obtain a local probability of correct classification, we first need to estimate $\Pr(U)$, $\Pr(V)$, g_u , G_u , g_v and G_v from the training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$. The prior class probabilities $\Pr(U)$ and $\Pr(V)$ can be estimated by simply calculating their proportions in the training data. However, estimations of g_u , G_u , g_v and G_v are more difficult and require the use of data points that were not used to construct the classifier defined in (1) (otherwise the probabilities come biased by the training set). In the next two sections, we define how g_u , G_u , g_v and G_v are estimated using examples independent from those used to obtain the classifier (1), and then we show how cross-validation is used to obtain these independent points from the training set.

3.1 Numerical Estimation of g_u , G_u , g_v and G_v

From equation (4), we know that for every point \mathbf{x} , there exists a line in basis function space $(\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$, defined by $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$, where at $t = 0$ the line passes through the hyperplane. If infinitely many points from the distribution f_v and f_u are located exactly on this line, then the task of estimating G_u and G_v is complete, because these cumulative distributions can be empirically estimated arbitrarily well [2]. However, it is likely this will not happen in practice (i.e. the number of sample points N is limited), and therefore we must use points generated from f_v and f_u that are *close* to the line $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$ in basis function space. To calculate the distance, denoted by β_i , to the line $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$ for any other point \mathbf{x}_i , we must first project it into basis function space $\mathbf{z}_{xi} = (\phi_1(\mathbf{x}_i), \dots, \phi_k(\mathbf{x}_i))$. Then we calculate the distance of \mathbf{z}_{xi} to the line $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$ using standard plane geometry:

$$\beta_i = |\mathbf{z}_{xi} - (\mathbf{z}^* - \gamma\mathbf{a})| \quad (10)$$

where $\gamma = (\mathbf{a}^t(\mathbf{z}_{xi} - \mathbf{z}^*)) / (\mathbf{a}^t\mathbf{a})$. Therefore, for all points \mathbf{x}_i which were not used to construct the discriminating hyperplane (3), we calculated its distance β_i to perpendicular

line containing the point \mathbf{x} being classified. This is done for points belonging to both classes U and V , and we note the location where each point intersects the perpendicular line. This is given by the y values of equation (1) which are referenced by y_i (corresponding to point \mathbf{x}_i which is distance β_i away). Thus, given a set of points \mathbf{x}_i in class U that are within distance β_{max} of our line $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$, we calculate an empirical (sometimes called Kaplan-Meier) cumulative distribution function (*cdf*) [2]. This is our estimate \hat{G}_u of G_u . We do the same thing for points \mathbf{x}_i in class V , which gives our estimate \hat{G}_v of G_v . However, in order for these estimates of G_u and G_v to be accurate, we need to make sure we have a statistically significant sample size of both classes - or equivalently, we need to make sure β_{max} is big enough to include enough points. The important property of Kaplan-Meier *cdf* is that we exploit is that, given a confidence level of $100(1 - \alpha)\%$ it returns a range of maximum and minimum *cdf* estimates. Therefore, by randomly dividing the points in each class into two, we can use cross validation to decide when the first *cdf* of one set is within the $100(1 - \alpha)\%$ confidence interval of the second. When β_{max} is large enough so that this is true for points in both classes U and V , we are $100(1 - \alpha)\%$ confident that our estimates of G_u and G_v are accurate. Finally, to estimate g_u we count the number of examples of class U that are within β_{max} and divide this by the total number points from class U . A similar calculation is done to estimate g_v . Given this descriptions, we now state the following theorem:

Theorem 3: *Assume that points \mathbf{x}_i were not used to estimate the hyperplane (1). Let \hat{g}_u , \hat{G}_u , \hat{g}_v and \hat{G}_v be estimates of g_u , G_u , g_v and G_v respectively, obtained as defined above. Then, as the number of training examples approaches infinity, $N \rightarrow \infty$, these estimates approach the true values: i.e. $|\hat{g}_u = g_u| \rightarrow 0$, $|\hat{G}_u = G_u| \rightarrow 0$, $|\hat{g}_v = g_v| \rightarrow 0$ and $|\hat{G}_v = G_v| \rightarrow 0$.*

Proof Sketch: The proof is based on the fact that if infinitely many points are projected onto $\mathbf{z} = \mathbf{a}t + \mathbf{z}^*$, then the Kaplan-Meier *cdf* will approach the true *cdf* (see [2]). Furthermore, because estimates of g_u and g_v are obtained by calculating ratios, statistical sampling theory tells us that as the number of points used to approximate g_u and g_v goes to infinity, the error in their estimates goes to zero (see [6]). This completes the proof sketch.

3.2 Obtaining Independent Training Examples

As discussed above, our estimates of g_u , G_u , g_v and G_v assume that we use points that were NOT used to construct the classification model (1). In order to obtain these independent points, we follow the M -fold cross-validation procedure defined in [7]. Namely, the training data is split into M folds and M models are built using a different set of $M - 1$ folds to build each model. The points in the fold not used to construct the model are then used to obtain values y_i (corresponding to point \mathbf{x}_i in the training set). Thus unbiased values of y_i are obtained for each training example, which are in turn used to estimate the Kaplan-Meier *cdf* used in the previous section.

3.3 Algorithm Summary

The final Probabilistic CDF model is defined by: 1) a single basis function model $\mathbf{a} = (a_1, \dots, a_k)$, $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$ and b as defined in (1); a set of values y_1, \dots, y_n (see (1)) for each training point input $\mathbf{x}_1, \dots, \mathbf{x}_N$ obtained via cross validation as described in [7] (note that y_i is called f_i in [7]); and finally a vector $\mathbf{z}_{xi} = (\phi_1(\mathbf{x}_i), \dots, \phi_k(\mathbf{x}_i))$ for each training input. For each test point \mathbf{x} , we calculate $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = 0)$ and $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*)$ as follows (note that the *cdf* functions G_u , and G_v are obtained using the *ecdf* function in the *matlab statistics toolbox*):

1. *Project into basis function space.* Calculate $(\phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))$ the distances

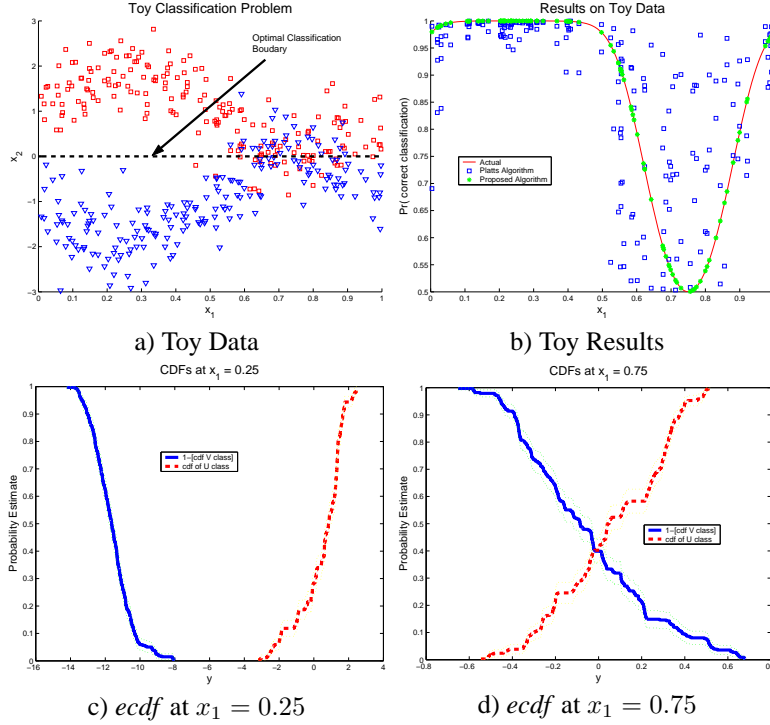


Figure 1: Toy Binary Classification Problem. See text for description.

β_1, \dots, β_N of this point (in basis function space) to each of the training points \mathbf{z}_{xi} using equation (10).

2. *Find β_{max} .* Choose a window size β_{max} that gives $100(1 - \alpha)\%$ confidence in estimates of G_u and G_v (as defined in Section 3.1).
3. *Estimate probabilities.* Estimate g_u , G_u , g_v and G_v as defined in Section 3.1 and use these to estimate $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = 0)$ and $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*)$ using equations (6) and (7) respectively.

4 Experimental Results

In all of the experiments below, the basis function classifier used is a support vector machine [8]. The LIBSVM implementation is used (www.csie.ntu.edu.tw/~cjlin/libsvm/) and α confidence is set to 0.05 (α defined in Section 3.1).

4.1 Toy Example

The toy problem considered is the two dimensional binary classification problem depicted in Figure 1. The two classes have equal prior probabilities. This is a linear problem where $\phi_1 = x_1$ and $\phi_2 = x_2$, and the optimal separating boundary is given by $\phi_2 = x_2 = 0$. The analytically obtained $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*)$ is shown in the solid line in Figure 1b. Note that the toy problem is almost completely separable at $x_1 = 0.25$ and the two classes *cannot* be separated $x_1 = 0.75$. This is supported by the empirical cdf function at $x_1 = 0.25$ and $x_1 = 0.75$ which are given in Figure 1c and 1d respectively. In Figure 1c the cdfs of the two classes don't cross, and are thus completely separable, while in Figure 1d they cross at about 0.5, meaning that $\Pr(\hat{c} = c | \text{sgn}(\Phi(\mathbf{x})\mathbf{a}^T + b), t = t^*) = 0.5$ (i.e. equal chance of being right and wrong). Figure 1b shows the probability predictions of

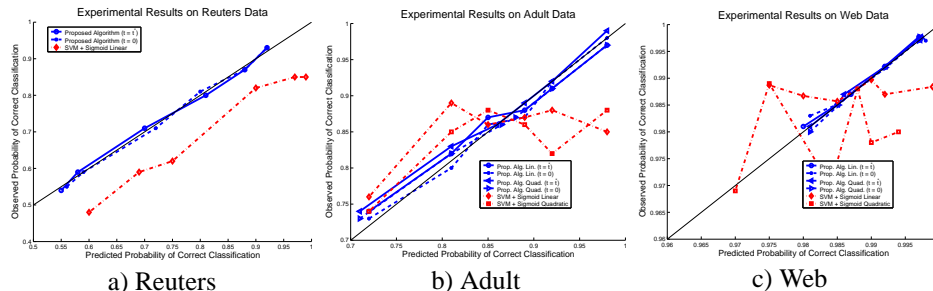


Figure 2: Summary of Results on Reuters, Adult, and Web data

the Probabilistic CDF model proposed here, which closely follow the analytically derived probabilities. Figure 1b also shows the results obtained using the SVM+sigmoid model in [7], demonstrating that this approach cannot effectively model probabilities of this toy data (this is because the SVM+sigmoid model models probabilities as distance to boundary and independent of \mathbf{x} , a condition that this toy data violates).

4.2 Three Large Real Data Sets

In order to assess the proposed algorithm's ability to predict the probability of correct classification we evaluated on three large data sets used in [7]; Reuters news article, the UCI Adult benchmark, and Web data. We used the *exactly* same experimental setup as [7], however, we also kept track of the predicted probabilities of correct classification for all the algorithms, and use a standard method assessing these probabilities [3]. More specifically, the predicted probabilities \hat{p}_i for the test data were sorted and put into 6 bins, each with an equal number of predictions. The predicted correct probability in each bin was the average of all the predictions in that bin. The actual correct probability was obtained by calculating the *ACTUAL* error rate on the test data in each bin. Plots of the predicted versus observed probability results are given in Figure 2. Notice that the proposed algorithm, both for optimal values of $t = t^*$ and the suboptimal values $t = 0$, was able to effectively predict the true probabilities of correct prediction (i.e. lie on the 45 degree line in the plots). In contrast, the SVM + Sigmoid (both linear and quadratic) [7], did poorly at predicting the probability of correct classification. In fact, for both the Adult and Web data, the predicted probabilities were not well correlated to the actual. Finally, the global accuracy rates obtained on the data sets are given in Table 1. Note that the proposed algorithm compares favorably to the results reported in [7]. It is also interesting to note that the two versions of the algorithm, $t = t^*$ and $t = 0$, get very similar results, with the locally optimized decision boundary (i.e. $t = t^*$) giving only slightly better results.

4.3 Evaluation on Smaller Data Sets

In order to determine the overall accuracy of the proposed algorithm, we evaluated it on six widely used small data sets: Ionosphere, Sonar, Breast Cancer, Pima diabetes, and Twonorm data sets used in [5], as well as the votes data used in [1]. The experimental setup was identical to that described in [5] and [1]. These data sets allow direct comparison with state-of-the-art machine learning algorithms: minimax probability machines [5], Support Vector Machines [8], and Random Forests [1]. Table 2 gives accuracy results for these algorithms and the two versions of the algorithm proposed here ($t_0 = 0$ and $t = t^*$). The numbers in brackets in the columns associated with the proposed algorithm are the average prediction accuracies for all test points. Three interesting observations can be made from Table 2. First, as predicted by Theorem 2, the proposed algorithm that uses the locally optimal threshold (i.e. $t = t^*$) outperforms the algorithm which does not (i.e. $t_0 = 0$). Second, the proposed algorithm with $t_0 = t^*$ performs as well as or better than previously published results. And finally, the average prediction accuracies for the pro-

	PA Lin ($t_0 = 0$)	PA Lin ($t_0 = t^*$)	PA Qaud ($t_0 = 0$)	PA Qaud ($t_0 = t^*$)	Lin SVM	Quad SVM
Reuters	72.5%	72.9%	-	-	70.8%	-
Adult	72.5%	72.9%	72.1%	72.2%	85.0%	84.3%
Web	98.7%	99.0%	98.0%	98.7%	98.77%	97.9%

Table 1: Accuracy rates on Reuters, Adult, and Web data.

Dataset	Prop. Alg. ($t = 0$)	Prop. Alg. ($t = t^*$)	MPM Linear	MPM Quadratic	SVM Linear	SVM Quadratic	Random Forests
Ionosphere	89.2% (88.2%)	93.5% (94.2%)	85.4%	93.0%	87.8%	91.5%	92.9%
Sonar	81.0% (80.5%)	90.1% (88.3%)	75.1%	89.8%	75.9%	86.7%	84.1%
Breast Cancer	95.9% (95.0%)	98.3% (97.5%)	97.2%	97.3%	92.6%	98.5%	97.1%
Pima diabetes	71.8% (71.2%)	76.1% (77.2%)	73.8%	74.6%	70.1%	75.3%	66.9%
Twonorm	96.3% (96.0%)	96.9% (96.1%)	96.3%	97.2%	95.6%	97.4%	96.1%
Votes	90.2% (92.0%)	95.1% (96.0%)	-	-	-	-	95.9%

Table 2: Accuracy results on widely used small datasets. The numbers in brackets in the columns associated with the proposed algorithm (both versions ($t_0 = 0$) and ($t_0 = t^*$)) indicated the average predicted accuracy rates.

posed algorithms closely match the true accuracy, once more supporting the claim that the algorithm is able to predict how good individual predictions are.

5 Conclusion

We proposed a new formulation for building classifier models that, for each new observation (or instance), output both a class prediction as well as a probability that the prediction is correct. The formulation is based on the widely used basis function models (examples of which include Support Vector Machines [8] as well as other widely used classifiers [4]). Theoretical support is given showing that, in the limit of infinite data, our formulation gives perfect predictions on how accurate specific class estimates are. Experimental evidence supports this claim, as well demonstrating that the proposed algorithm gives global error rates that are competitive with state of the art algorithms which make no attempt to predict how accurate specific model outputs are. This paper leaves open a number of interesting questions for future research. These include analysis, both theoretical and experimental, of how the formulation is affected by the choice of the basis functions.

References

- [1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [2] D.R. Cox and D. Oakes. *Analysis of Survival Data*. Chapman and Hall, London, 1984.
- [3] A. P. Dawid. Calibration-based empirical probability. *The Annals of Statistics*, Vol. 13, No. 4. (Dec., 1985), pp. 1251–1274, 1985.
- [4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.
- [5] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.
- [6] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer, 1998.
- [7] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, 2000.
- [8] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, 1998.
- [9] Ting-Fan Wu, Chih-Jen Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.