A Reformulation of Support Vector Machines for General Confidence Functions

Yuhong Guo[1] and Dale Schuurmans[2]

[1] Department of Computer & Information Sciences
Temple University
www.cis.temple.edu/˜yuhong
[2] Department of Computing Science
University of Alberta
www.cs.ualberta.ca/˜dale

Abstract. We present a generalized view of support vector machines that does not rely on a Euclidean geometric interpretation nor even positive semidefinite kernels. We base our development instead on the *confidence matrix*—the matrix normally determined by the direct (Hadamard) product of the kernel matrix with the label outer-product matrix. It turns out that alternative forms of confidence matrices are possible, and indeed useful. By focusing on the confidence matrix instead of the underlying kernel, we can derive an intuitive principle for optimizing example weights to yield robust classifiers. Our principle initially recovers the standard quadratic SVM training criterion, which is only convex for kernel-derived confidence measures. However, given our generalized view, we are then able to derive a principled relaxation of the SVM criterion that yields a convex upper bound. This relaxation is *always* convex and can be solved with a linear program. Our new training procedure obtains similar generalization performance to standard SVMs on kernel-derived confidence functions, but achieves even better results with indefinite confidence functions.

1 Introduction

Support vector machines were originally derived from purely geometric principles [13, 1]: given a labeled training set, one attempts to solve for a consistent linear discriminant that maximizes the minimum Euclidean distance between any data point and the decision hyperplane. Specifically, given $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_t, y_t)$, $y \in \{-1, +1\}$, the goal is to determine a $(\mathbf{w}, b)$ such that $\min_i y_i(\mathbf{w}^\top \mathbf{x}_i + b)/\|\mathbf{w}\|$ is maximized. Vapnik [13] famously proposed this principle and formulated a convex quadratic program for efficiently solving it. With the addition of slack variables the dual form of this quadratic program can be written

$$\min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_i \alpha_i \quad \text{subject to} \quad 0 \leq \boldsymbol{\alpha} \leq \beta, \ \boldsymbol{\alpha}^\top \mathbf{y} = 0 \qquad (1)$$

where the dual variables $\boldsymbol{\alpha}$ behave as weights on the training examples.

One of the key insights behind the support vector machine approach is that the training vectors appear only as inner products in both training and classification, and therefore can be abstracted away by a general kernel function. In this case, the kernel function, $k(\mathbf{x}_i, \mathbf{x}_j)$, simply reports inner products $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$ in some arbitrary feature (Hilbert) space. Combining the kernel abstraction with the ν-SVM formulation of [12, 4] one can re-express (1) in the more general form

$$\min_{\boldsymbol{\alpha}} \; \boldsymbol{\alpha}^\top (K \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\alpha} \quad \text{subject to} \quad 0 \le \boldsymbol{\alpha} \le \beta, \; \boldsymbol{\alpha}^\top \mathbf{y} = 0, \; \boldsymbol{\alpha}^\top \mathbf{e} = 1 \tag{2}$$

where K is the *kernel matrix*, $K_{ij} = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$, the matrix $\mathbf{y}\mathbf{y}^\top$ is the *label matrix*, the vector $\mathbf{e}$ consists of all 1's, and $\circ$ denotes componentwise matrix multiplication (Hadamard product).

Although (2) appears to be a very general formulation of the weight training problem for $\boldsymbol{\alpha}$, it is in fact quite restrictive: for (2) to be convex, the combined matrix $K \circ \mathbf{y}\mathbf{y}^\top$ must be positive semidefinite, implying that K itself must be conditionally positive semidefinite.[3] Thus, it is commonly assumed that support vector machines should be applied on conditionally positive semidefinite kernels K.

Although the restriction to conditional positive semidefiniteness might not appear onerous, it is actually problematic in many natural situations. First, as [11] notes, verifying that a putative kernel function $k(\cdot, \cdot)$ is conditionally positive semidefinite can be a significant challenge. Second, as many authors note [2, 3, 7, 9–11, 14] using indefinite kernels and only approximately optimizing (2) can often yield similar or even better results than using conventional positive semidefinite kernels. (A frequently used example is the hyperbolic tangent "kernel" $\tanh(a\langle \mathbf{x}_i, \mathbf{x}_j\rangle + b)$.) Third, adding conditional positive semidefiniteness as a constraint causes difficulty when attempting to *learn* a kernel (similarity measure) directly from data.

In fact, it is this third difficulty that is the main motivation for this research. We are interested in *learning* similarity measures from data that we can then use to train accurate classifiers. One can easily devise natural ways of doing this (we elaborate on one approach below), but unfortunately in these cases ensuring positive semidefiniteness ranges from hard to impossible. To date, most successful attempts at learning conditionally positive semidefinite kernels have been reduced to taking convex combinations of known conditionally positive semidefinite kernels [8, 5]. But we would like to consider a wider range of techniques for learning similarities, and therefore we seek to generalize (2) to exploit general similarity matrices. Our goal is to develop an efficient weight optimization procedure for $\boldsymbol{\alpha}$ that does not require a positive semidefinite matrix $K \circ \mathbf{y}\mathbf{y}^\top$, while still preserving the desirable generalization and sparseness properties achieved by standard SVM training.

Below in Section 2 we show how the standard kernel classifier can be generalized to consider more abstract *confidence functions* $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$ that play the same role as the usual kernel-label combination $y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$. We then briefly outline some natural approaches for learning confidence functions from data in Section 3. The approach we propose there is very simple, but effective. Nevertheless, it suffers from the drawback

[3] A symmetric matrix K is conditionally positive semidefinite if $\mathbf{z}^\top K \mathbf{z} \ge 0$ for all $\mathbf{z}$ such that $\mathbf{z}^\top \mathbf{e} = 0$. K need only be conditionally positive semidefinite to ensure $K \circ \mathbf{y}\mathbf{y}^\top$ is positive semidefinite because of the assumption $\boldsymbol{\alpha}^\top \mathbf{y} = 0$. That is, if $(\boldsymbol{\alpha} \circ \mathbf{y})^\top \mathbf{e} = \boldsymbol{\alpha}^\top \mathbf{y} = 0$, then we immediately obtain $\boldsymbol{\alpha}^\top (K \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\alpha} = (\boldsymbol{\alpha} \circ \mathbf{y})^\top K (\boldsymbol{\alpha} \circ \mathbf{y}) \ge 0$.

of not being able to ensure a positive semidefinite matrix for optimization. Section 4 then outlines our main development. Given the general confidence function viewpoint, we derive an $\boldsymbol{\alpha}$-weight optimization procedure from intuitive, strictly *non*-geometric first principles. The first procedure we derive simply recovers the classical quadratic objective, but from a new perspective. With this re-derivation in hand, we are then able to formulate a novel relaxation of the standard SVM objective that is both principled while also being guaranteed to be convex. Finally, in Section 6 we present experimental results with this new training principle, showing similar performance to standard SVM training with standard kernel functions, but obtaining stronger performance using indefinite confidence functions learned from data.

2 Confidence function classification

Our goal is to develop a learning and classification scheme that can be expressed more abstractly than the usual formulation in terms of $y_i y_j k(\mathbf{x}_i \mathbf{x}_j)$. We do this via the notion of a *confidence function*, $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$, which expresses a numerical confidence that the label pair $y_i y_j$ is in fact correct for the input pair $\mathbf{x}_i$ and $\mathbf{x}_j$. A large confidence value expresses certainty that the label pair is correct, while a small value correspondingly expresses a lack of confidence that the label pair is correct (or certainty that the label pair is wrong). We make no other assumptions about the confidence function, although it is usually presumed to be symmetric: $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) = c(y_j y_i | \mathbf{x}_j \mathbf{x}_i)$.

Although the notion of a pairwise confidence function might seem peculiar, it is in fact exactly what the $y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$ values provide to the SVM. In particular, if we make the analogy $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$ and assume $y \in \{-1, +1\}$, one can see that $y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$ behaves as a simple form of confidence function: the value is relatively large if either $y_i = y_j$ and $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar under the kernel k, or if $y_i \neq y_j$ and $\mathbf{x}_i$ and $\mathbf{x}_j$ are *dissimilar* under the kernel. We therefore refer to the matrix $C = K \circ \mathbf{y}\mathbf{y}^\top$ as the *confidence matrix*.

Proposition 1. *If the entries of the confidence matrix $K \circ \mathbf{y}\mathbf{y}^\top$ are strictly positive, then the training data is linearly separable in the feature space defined by K.*

This proposition clearly shows that high confidence values translate into an accurate classifier on the training data. In fact, it is confidences, not similarities, that lie at the heart of support vector machines: The SVM methodology can be recast strictly in terms of confidence functions, abstracting away the notion of a kernel entirely, without giving up anything (except the Euclidean geometric interpretation). To illustrate, consider the standard SVM classifier: Assuming a vector of training example weights, $\boldsymbol{\alpha}$, has already been obtained from the quadratic minimization (2), the standard classification rule can be rewritten strictly in terms of confidence values

$$\begin{aligned}\hat{y} \;=\; \text{sign}\Big(\Big(\sum_j \alpha_j y_j k(\mathbf{x}, \mathbf{x}_j)\Big) + b\Big) &= \arg\max_y \left(\Big(\sum_j \alpha_j y y_j k(\mathbf{x}, \mathbf{x}_j)\Big) + by\right) \\ &= \arg\max_y \left(\Big(\sum_j \alpha_j c(y y_j | \mathbf{x}\mathbf{x}_j)\Big) + by\right) \quad (3)\end{aligned}$$

Thus a test example $\mathbf{x}$ is classified by choosing the label y that exhibits the largest weighted confidence when paired against the training data.

Quite obviously, the SVM training algorithm itself can also be expressed strictly in terms of a confidence matrix over training data.

$$\min_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}^\top C \boldsymbol{\alpha} \quad \text{subject to} \quad 0 \leq \boldsymbol{\alpha} \leq \beta, \ \boldsymbol{\alpha}^\top \mathbf{y} = 0, \ \boldsymbol{\alpha}^\top \mathbf{e} = 1 \tag{4}$$

This is just a rewriting of (2) with the substitution $C = K \circ \mathbf{y}\mathbf{y}^\top$, which does not change the fact that the problem is convex if and only if C is positive semidefinite. However, the formulation (4) is still instructive. Apparently the SVM criterion is attempting to reweight the training data to *minimize* expected confidence. Why? Below we argue that this is in fact an incorrect view of (4), and suggest that, alternatively, (4) can be interpreted as attempting to *maximize* the robustness of the classifier against changes in the training labels. With this different view, we can then formulate an alternative training criterion—a relaxation of (4)—that still attempts to maximize robustness, but is convex for *any* confidence matrix C. This allows us to advance our goal of *learning* confidence functions from data, while still being able to use SVM training of the example weights without having to ensure positive semidefiniteness.

Before turning to the interpretation and relaxation of (4), we first briefly discuss approaches that can be explored to learning confidence functions.

3 Learning confidence functions

There are many natural ways to consider learning a confidence function $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$ from training data. A straightforward approach is to explore any known similarity learning techniques to learn an arbitrary kernel matrix and then obtain a confidence function by combining the y terms. Alternatively, one can also learn the confidence function directly. One of such simple techniques has been explored in [6]. Given training labels, one can just straightforwardly learn to predict label pairs $y_i y_j$ given their corresponding input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. Concretely, given examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_t, y_t)$, it is easy to form the set of training pairs $\{(\mathbf{x}_i \mathbf{x}_j, y_i y_j)\}$ from the original data, which doubles the number of input features and squares the number of training labels and classes. (Subsampling can always be used to reduce the size of this training set.) Given such pairwise training data, standard probabilistic models can be learned to predict the probability of a label pair given the input vectors.

Many standard probabilistic methods, especially discriminative methods, can be used for learning pairwise predictors. For example, the logistic regression classifiers can be trained to maximize the conditional likelihood $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$ of the observed label pairs given the conjoined vector of inputs $\mathbf{x}_i \mathbf{x}_j$. Once learned, a pairwise model would classify test inputs $\mathbf{x}$ by maximizing the product $\hat{y} = \arg\max_y \prod_j P(y y_j | \mathbf{x}\mathbf{x}_j)$.[4] Clearly, this is equivalent to using a confidence function $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) = \log P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$ and classifying with respect to uniform example weights $\boldsymbol{\alpha}$. Surprisingly, [6] obtained good classification results with this simple approach. In this paper, we are interested in

[4] [6] also considered other techniques for classification, including correlating the predictions on the test data in a transductive manner, but we do not pursue these extensions here.

the connection to support vector machines and attempt to improve the basic method by optimizing the $\boldsymbol{\alpha}$-weights.

Note that, as a confidence measure, using a log probability model, $\log P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$, is a very natural choice. It can be trained easily from the available data, and performs quite well in practice. However, using a log probability model for a confidence function raises a significant challenge: $\log P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$ is always non-positive and therefore any confidence matrix C it produces, since it is strictly non-positive, cannot be positive semidefinite. This raises the same difficulty one faces with non-positive semidefinite kernels, which motivates us to reformulate the quadratic optimization criterion (4), so that convexity can be achieved more generally while preserving the effective generalization properties.

4 Optimizing training example weights: An alternative view

Given a confidence classifier (3) it is natural to consider adjusting the training example weights $\boldsymbol{\alpha}$ to improve accuracy. At first glance, the quadratic minimization criterion (4) used by SVMs appears to be adjusting the example weights to *minimize* the confidence of the training labels y_i. However, we can argue that this interpretation is misleading. In fact, standard kernel-based confidence functions have a special property that masks a key issue: how confidences change when a training label is flipped. For the classifier (3), it is not the absolute confidence that counts, but rather the *relative* confidences between the correct label and the incorrect label. That is, we would like the confidence of a correct label to be larger than the confidence of a wrong label. For kernel-based confidence functions it turns out that the relationship between the relative confidences is greatly restricted.

Observation 1 *Let $\overline{y}$ denote a label flip, $-y$. If $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) = y_i y_j k(\mathbf{x}_i \mathbf{x}_j)$ then*

$$\sum_j \alpha_j c(y y_j | \mathbf{x} \mathbf{x}_j) + by = -\sum_j \alpha_j c(\overline{y} y_j | \mathbf{x} \mathbf{x}_j) - b\overline{y} \tag{5}$$

However, the relationship (5) is obviously not true in general. For example, it is violated by any probabilistic confidence function defined by $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) = \log P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$.

Thus for kernel-based confidence functions, the confidence in the opposite label is always just the negation of the confidence in the current label.

We now show how the concept of minimizing sensitivity to label flips on the training data recovers the classical SVM training criterion. Consider a training example $(\mathbf{x}_i, y_i)$ and an arbitrary current set of weights $\boldsymbol{\alpha}$. The current confidence in the training label y_i is

$$\Big(\sum_j \alpha_j c(y_i y_j | \mathbf{x}_i \mathbf{x}_j) \Big) + b y_i$$

Now consider the change in the confidence in y_i that would result if a single training label y_k was actually mistaken. That is, if the current value of y_k is incorrect and should have been given the opposite sign, then the mistake we are making in y_i's confidence is

$$\Delta_k c(y_i) = \alpha_k c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) - \alpha_k c(\overline{y_k} y_i | \mathbf{x}_k \mathbf{x}_i) \tag{6}$$

$$= 2\alpha_k c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) \tag{7}$$

Note that (7) holds only under the kernel-based restriction (5), but in this special case the confidence penalty is just twice the original confidence. If (5) does not hold, then (6) can be used. The sum of the local confidence changes measures the overall sensitivity of the classification of training label y_i to possible mislabelings of other data points

$$\Delta c(y_i) = \sum_k \Delta_k c(y_i) \tag{8}$$

The smaller this value, the less likely y_i is to be misclassified due to a mislabeling of some other data point. That is, the sensitivity to label flips should be minimized if the classifier is to be made more robust. Nevertheless, there might be a trade-off between the sensitivities of different training examples. Therefore, as a final step, we consider minimizing the overall *weighted* sensitivity of the training labels. This yields the minimization objective

$$\sum_i \alpha_i \Delta c(y_i) = \sum_i \alpha_i \sum_k \alpha_k \Big(c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) - c(\overline{y_k} y_i | \mathbf{x}_k \mathbf{x}_i) \Big) \tag{9}$$

$$= 2 \sum_{ik} \alpha_i \alpha_k c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) \tag{10}$$

Again, (10) only holds under the kernel-based restriction (5), but if this is violated, the more general form (9) can be used.

Therefore, if we are using a kernel-based confidence function, we recover exactly the same training criterion as the standard SVM (4). What is interesting about this derivation is that it does not require any reasoning about Euclidean geometry or even feature (Hilbert) spaces. The argument is only about adjusting the example weights to reduce the sensitivity of the classifier (3) to any potential mistakes in the training labels. That is, from the perspective of $\boldsymbol{\alpha}$-weight optimization, it is only the reflection property (5) and the desire to minimize sensitivity to mislabeled training examples that yields the same minimization objective as standard SVMs. The remaining constraints in (4) are also easily justified in this context: It is natural to assume that the example weights form a convex combination, and therefore $0 \leq \boldsymbol{\alpha}$, $\boldsymbol{\alpha}^\top \mathbf{e} = 1$. It is also natural to preserve class balance in the re-weighting, hence $\boldsymbol{\alpha}^\top \mathbf{y} = 0$. Finally, as a regularization principle, it makes sense to limit the magnitude of the largest weights so that too few examples do not dominate the classifier, hence $\boldsymbol{\alpha} \leq \beta$.

Of course, re-deriving an old criterion from alternative principles is not a significant contribution. However, what is important about this perspective is that it immediately suggests principled alternatives to the SVM criterion that can still reduce sensitivity to potential training label changes. Our goal is to reformulate the objective to avoid a quadratic form, since this prevents effective optimization on indefinite confidence functions, which are the confidence functions we are most interested in (Section 3). It turns out that just a minor adjustment to (10) yields just such a procedure.

Given the goal of minimizing the sensitivity to training label changes, previously we sought to minimize the *weighted* sensitivity, using the same weights being optimized, which leads to the quadratic form (10) and the optimization problem (4). However, rather than minimize weighted sensitivity, one could instead be more conservative and attempt to minimize the *maximum* sensitivity of any label in the training set. That is, we would like to adjust the example weights so that the worst sensitivity of any training label y_i to potential mislabelings of other examples is minimized. This suggestion immediately yields our proposal for a new training criterion

$$\min_{\boldsymbol{\alpha}} \max_i \sum_k \alpha_k \Big(c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) - c(\overline{y_k} y_i | \mathbf{x}_k \mathbf{x}_i) \Big) \quad \text{subject to} \quad 0 \le \boldsymbol{\alpha} \le \beta,\ \boldsymbol{\alpha}^\top \mathbf{y} = 0,\ \boldsymbol{\alpha}^\top \mathbf{e} = 1 \tag{11}$$

Once $\boldsymbol{\alpha}$ has been optimized, the offset b can be chosen to minimize training error.

Proposition 2. *The objective (11) is convex for any confidence function c, and moreover is an upper bound on (4).*

The proof of this proposition is obvious. The minimization objective is a maximum of linear functions of $\boldsymbol{\alpha}$, and hence is convex. Given the constraints $0 \le \boldsymbol{\alpha}$, $\boldsymbol{\alpha}^\top \mathbf{e} = 1$ we immediately have $\max_i f(i) \ge \sum_i \alpha_i f(i)$.

As a practical matter, (11) can be solved by a simple linear program

$$\min_{\boldsymbol{\alpha},\delta} \ \delta \ \text{ subject to } \ \delta \ge \sum_k \alpha_k \Big(c(y_k y_i | \mathbf{x}_k \mathbf{x}_i) - c(\overline{y_k} y_i | \mathbf{x}_k \mathbf{x}_i) \Big) \quad \forall i$$
$$0 \le \boldsymbol{\alpha} \le \beta,\ \boldsymbol{\alpha}^\top \mathbf{y} = 0,\ \boldsymbol{\alpha}^\top \mathbf{e} = 1 \tag{12}$$

This formulation produces a convex relaxation of the ν-SVM criterion for *any* confidence function $c(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$, and provides an alternative option for using indefinite kernels.

5 Related Work

Most work related to our proposed approach concerns learning SVMs with indefinite kernels [10, 2, 3, 11]; although in this paper we address the slightly more general problem of learning from confidence functions—a superset of confidences derived from indefinite kernels.

As noted, if the kernel matrix is not conditionally positive semidefinite, the standard SVM training problem (2) becomes non-convex, hence hard to optimize. To overcome this difficulty, some authors have suggested using the indefinite kernel directly but instead solve an approximate form of the SVM training problem [11, 9, 7]. Most, however, propose to transform the indefinite kernel into a positive semidefinite kernel and then apply standard SVM training. Such transformation methods include "denoise" (neglect the negative eigenvalues), "flip" (flip the sign of the negative eigenvalues) and "shift" (shift all eigenvalues by a positive constant to make all positive); see [14] for details. A limitation of using such simple transformation methods is that valuable information about the data can be lost in the transformation process. Therefore, more recently, a number of papers have begun to pursue a middle ground strategy, where the original indefinite kernel is used in training, while the SVM objective is perturbed as little as possible to maintain a convex objective $[10, 2, 3]$. These papers proceed by fixing the original indefinite kernel K_0 and then modifying the training objective as follows.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \min_{K} \quad & \boldsymbol{\alpha}^\top \mathbf{e} - \frac{1}{2}\boldsymbol{\alpha}^\top (K \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\alpha} + \rho \|K - K_0\|^2 \\ \text{subject to} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0,\ 0 \le \boldsymbol{\alpha} \le \beta,\ K \succeq 0 \end{aligned} \tag{13}$$

We provide some comparison of our approach to these different techniques below.

6 Experimental results

We implemented the new weight optimization scheme based on linear programming (12) and compared it both to standard SVM quadratic minimization (4) and to a simple approach of using uniform weights. In addition, for *indefinite* confidence measures, we compared our proposed method to the existing kernel transformation methods mentioned in Section 5 above. Note that training based on (4) can only be efficiently performed when the kernel is positive semidefinite. Furthermore, note that although using uniform weights appears to be naive, for high quality confidence measures such as those learned by training reasonable probability models, uniform weighting can still achieve highly competitive generalization performance—a fact that will be revealed below.

We compared these different weight optimization schemes on a variety of confidence functions, including those defined by standard positive semidefinite kernels (linear dot product and RBF), as well as the sigmoid kernel $\tanh$, and two probabilistic confidence models trained using naive Bayes and logistic regression respectively. In our result tables below, we will denote the results produced by the proposed weight optimization scheme (12) as $\boldsymbol{\alpha}1$, for the uniform weighting scheme as $\boldsymbol{\alpha}0$, and for the standard SVM quadratic minimization scheme (4) as $\boldsymbol{\alpha}2$. We compared the test accuracy of these various algorithms on a set of two-class UCI data sets. All of our experimental results are averages over 5 times repeats with training size equal to 100 or 4/5ths of the data size.

In the first study we compared how the different weight optimization schemes performed using the positive semidefinite confidence functions determined by the linear and RBF kernels respectively. Table 1 shows that the new weight optimization scheme (12) achieves comparable generalization performance to standard quadratic training (4). The uniform weighting strategy is generally inferior in the linear kernel case (being dominated on all data sets except placing second on *flare* and *german*). Although uniform weighting performs relatively better in the RBF kernel case, the results are still not comparable to the linear and quadratic weighting schemes. The reason is that the confidence functions are only weakly informative here, and simply averaging them still yields a sensitive classifier. It is encouraging to note that our convex relaxation retains most of the benefit of the original quadratic objective in this case.

We also compared our proposed approach to existing methods for learning with indefinite kernels. Table 2 shows the comparative results we obtained using indefinite kernels produced by $\tanh$ functions. Interestingly, the straightforward *denoise* and *flip* transformations yield very good results using tanh kernels, while *shift* produces very poor performance. The sophisticated *reg-svm* technique, based on (13) [10, 2, 3], requires much more involved training, yet produces very similar results to the simple linear optimization scheme we propose.

Table 1. *Comparison of the test accuracy of different $\boldsymbol{\alpha}$-weight optimizers on UCI data sets using the positive semidefinite confidence functions defined by linear (L) and RBF ($k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2)$) kernels. Here, $\boldsymbol{\alpha}$0 denotes the results for uniform weighting; $\boldsymbol{\alpha}$1 for our proposed linear optimization (12); and $\boldsymbol{\alpha}$2 for standard quadratic SVM optimization (4).*

	L-α0	**L-α1**	**L-α2**	**RBF-α0**	**RBF-α1**	**RBF-α2**
australian	0.613	0.815	0.843	0.729	0.784	0.713
breast	0.928	0.969	0.970	0.947	0.971	0.962
cleve	0.617	0.794	0.802	0.817	0.819	0.792
corral	0.571	0.900	0.929	0.907	0.936	1.000
crx	0.546	0.850	0.837	0.649	0.770	0.677
diabetes	0.651	0.754	0.705	0.741	0.705	0.730
flare	0.829	0.829	0.760	0.800	0.815	0.829
german	0.700	0.705	0.666	0.498	0.628	0.700
glass2	0.756	0.781	0.819	0.892	0.867	0.864
heart	0.780	0.824	0.835	0.764	0.794	0.774
hepatitis	0.813	0.863	0.863	0.725	0.825	0.825
mofn-3-7	0.779	0.781	0.805	0.704	0.776	0.857
pima	0.651	0.763	0.736	0.772	0.701	0.694
vote	0.615	0.931	0.935	0.881	0.913	0.875
average	0.703	0.826	0.822	0.773	0.807	0.807

Another interesting test of the method is on using *learned* indefinite confidence functions, such as those determined by an estimated probability model $\log P(y_i y_j | \mathbf{x}_i \mathbf{x}_j)$. In these cases, the quadratic objective is non-convex and cannot be solved by standard quadratic optimizers. However, as mentioned, our relaxation remains convex. [11] suggests more sophisticated approach for training in these cases, but their methods are substantially more technical than the simple technique proposed here. Table 3 shows the results of our linear weight optimization procedure and the uniform weighting on the indefinite confidence functions. Noticeably the probabilistic (trained) confidence functions yield better results than the ones produced in the previous tables by standard SVMs with both linear and RBF kernels. Moreover, even uniform weighting already achieves good results for these confidence functions, since the confidence functions obtained using LR and NB are very informative.

The main benefit of the new approach is the ability to reliably optimize example weights for a wider range of confidence functions. We believe this is a useful advantage over SVM training because most natural confidence functions, in particular learned confidence functions, are not usually positive semidefinite but have a wider potential for generalization improvement over using fixed kernels, as our results suggest.

Table 2. *Comparison of the test accuracy of different methods using indefinite kernels produced by* $\tanh$ *(*$k(\mathbf{x}_i, \mathbf{x}_j) = tanh(0.001 \cdot \mathbf{x}_i\mathbf{x}_j - 1)$*). Here, the column headings refer to the various techniques reviewed in Section 5, except that* $\boldsymbol{\alpha}$*1 refers to our proposed method based on (12).*

	denoise	flip	shift	reg-svm	$\boldsymbol{\alpha}1$
australian	0.852	0.852	0.554	0.782	0.844
breast	0.967	0.967	0.650	0.965	0.969
cleve	0.778	0.767	0.541	0.798	0.763
corral	0.893	0.893	0.429	0.871	0.879
crx	0.842	0.841	0.429	0.828	0.879
diabetes	0.758	0.759	0.651	0.759	0.699
flare	0.829	0.698	0.829	0.828	0.829
german	0.722	0.723	0.700	0.726	0.695
glass2	0.768	0.765	0.540	0.781	0.781
heart	0.827	0.827	0.553	0.837	0.815
hepatitis	0.875	0.875	0.813	0.813	0.825
mofn-3-7	1.000	1.000	0.779	0.816	0.813
pima	0.756	0.763	0.651	0.766	0.735
vote	0.935	0.935	0.615	0.928	0.921
average	0.843	0.833	0.624	0.821	0.818

7 Conclusion

We have introduced a simple generalization of support vector machines based on the notion of a *confidence function* $c(y_i y_j|\mathbf{x}_i\mathbf{x}_j)$. This view allows us to think of SVM training as attempting to minimize the sensitivity of the classifier to perturbations of the training labels. From this perspective, we can not only re-derive the standard SVM objective without appealing to Euclidean geometry, we can also devise a new training objective that is convex for arbitrary, not just positive semidefinite, confidence functions. Of course, other optimization objectives are possible, and perhaps superior ones could still be developed. An important research direction is to develop a generalization theory for our relaxed training procedure that is analogous to the theory that has already been developed for SVMs.

Acknowledgments The authors gratefully acknowledge support from Temple University, AICML, NSERC, MITACS and the Canada Research Chairs program.

References

1. B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings 5th Annual Conference on Computational Learning Theory (COLT)*, 1992.
2. J. Chen and J. Ye. Training SVM with indefinite kernels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.
3. Y. Chen, M. Gupta, and B. Recht. Learning kernels from indefinite similarities. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

Table 3. *Comparison of the test accuracy of different α-weight optimizers on UCI data sets, using the indefinite confidence functions produced by pairwise naive Bayes (NB), and logistic regression (LR). Here, α0 denotes the results for uniform weighting; and α1 for our proposed linear optimization (12).*

	LR-α0	**LR-α1**	**NB-α0**	**NB-α1**
australian	0.850	0.850	0.846	0.847
breast	0.959	0.933	0.972	0.969
cleve	0.802	0.804	0.840	0.846
corral	0.871	0.893	0.900	0.900
crx	0.845	0.845	0.842	0.836
diabetes	0.744	0.741	0.761	0.755
flare	0.820	0.820	0.831	0.821
german	0.719	0.718	0.714	0.713
glass2	0.806	0.813	0.873	0.895
heart	0.813	0.812	0.822	0.827
hepatitis	0.900	0.900	0.925	0.938
mofn-3-7	0.911	0.903	0.887	0.924
pima	0.743	0.743	0.758	0.759
vote	0.918	0.918	0.919	0.912
average	0.836	0.835	0.849	0.853

4. D.J. Crisp and C.J.C. Burges. A geometric interpretation of *v*-SVM classifiers. In *Advances in Neural Information Processing Systems (NIPS-00)*, 2000.
5. N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel target-alignment. In *Advances in Neural Information Processing Systems (NIPS-01)*, 2001.
6. Y. Guo, R. Greiner, and D. Schuurmans. Learning coordination classifiers. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.
7. B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:NO.4, 2005.
8. G. Lanckriet, N. Cristianini, P. Bartlett, L Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5, 2004.
9. H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. 2003.
10. R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems (NIPS-07)*, 2007.
11. C.S. Ong, X. Mary, S. Canu, and A.J. Smola. Learning with non-positive kernels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.
12. B. Schoelkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.
13. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
14. G. Wu, Y. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.