Perceptron and SVM Learning with Generalized Cost Models

Peter Geibel
Methods of Artificial Intelligence, Sekr. Fr 5–8
Faculty IV, TU Berlin,
Franklinstr. 28/29, D-10587 Berlin, Germany
Tel. +49-30-31425491, Fax. +49-30-31424913
Email geibel@cs.tu-berlin.de

Ulf Brefeld
Knowledge Management Group,
School of Computer Science, Humboldt University, Berlin
Unter den Linden 6, D-10099 Berlin, Germany
Email brefeld@informatik.hu-berlin.de

Fritz Wysotzki
Methods of Artificial Intelligence, Sekr. Fr 5–8
Email wysotzki@cs.tu-berlin.de

Keywords: Machine Learning, SVM, Perceptron, Costs

February 13, 2004

Abstract

Learning algorithms from the fields of artificial neural networks and machine learning, typically, do not take any costs into account or allow only costs depending on the classes of the examples that are used for learning. As an extension of class dependent costs, we consider costs that are example, i.e. feature and class dependent. We derive a cost-sensitive perceptron learning rule for non-separable classes, that can be extended to multi-modal classes (DIPOL) and present a natural cost-sensitive extension of the support vector machine (SVM). We also derive an approach for including example dependent costs into an arbitrary cost-insensitive learning algorithm by sampling according to modified probability distributions.

1 Introduction

The consideration of cost-sensitive learning has received growing attention in the past years [14, 6, 10, 13]. The aim of the inductive construction of classifiers from training sets is to find a hypothesis that minimizes the mean predictive error. If costs are considered, each example not correctly classified by the learned hypothesis may contribute differently to this error. One way to incorporate such costs is the use of a cost matrix, which specifies the misclassification costs in a class dependent manner (e.g. [14, 6]). Using a cost matrix implies that the misclassification costs are the same for each example of the respective class.

The idea we discuss in this paper is to let the cost depend on the single example and not only on the class of the example. This leads to the notion of example dependent costs (e.g. [11]). Besides costs for misclassification, we consider costs for correct classification (gains are expressed as negative costs). Because the individual cost values are obtained together with the training sample, we allow the costs to be corrupted by noise.

One application for example dependent costs is the classification of credit applicants in a bank as either being a "good customer" (the person will pay back the credit) or a "bad customer" (the person will not pay back parts of the credit loan).

The gain or the loss in a single case forms the (mis-) classification cost for that example in a natural way. For a good customer, the cost for correct classification is the negative gain of the bank. I.e. the cost for correct classification is not the same for all customers but depends on the amount of money borrowed. Generally there are no costs to be expected (or a small loss related to the handling expenses) if the customer is rejected, for he or she is incorrectly classified as a bad customer. For a bad customer, the cost for misclassification corresponds to the actual loss that has occured. The cost of correct classification is zero (or small positive if one considers handling expenses of the bank).

As opposed to the construction of a cost matrix that is often given by some expert, we claim that using the example dependent costs directly is more natural and will lead to more accurate classifiers. If the real costs are example dependent as in the credit risk problem, learning with a cost matrix means that in general only an approximation of the real costs is used. When using the classifier based on the cost matrix in the real bank, the real costs as given by the example dependent costs will occur, and not the costs specified by the cost matrix. Therefore using example dependent costs is better than using a cost matrix for theoretical reasons, provided that the learning algorithm

used is able to use the example dependent costs in an appropriate manner[1].

In this paper we consider single neuron perceptron learning and the algorithm DIPOL introduced in [15, 17, 20] that brings together the high classification accuracy of neural networks and the interpretability gained from using simple neural models (threshold units).

Another way of dealing with non-linearly separable, non-separable or multi-modal data is the Support Vector Machine (SVM, [19]). We will demonstrate how to extend the SVM with example dependent costs, and compare its performance to the results obtained using DIPOL.

In order to use cost-insensitive learning algorithms together with example dependent costs, we develop a new sampling strategy for generating an appropriate cost-free training set from the one that contains the costs.

This article is structured as follows. In section 2 the Bayes rule in the case of example dependent costs is discussed. In section 3, the learning rule is derived for a cost-sensitive extension of a perceptron algorithm for non-separable classes. In section 4 the extension of the learning algorithm DIPOL for example dependent costs is described, and in section 5 the extension of the SVM is presented. In section 6, we discuss the inclusion of costs by resampling the dataset. Experiments on two artificial data sets, and on two real world data sets can be found in section 7. The conclusion is presented in section 8.

2 Example Dependent Costs

In the following we consider binary classification problems with classes –1 (negative class) and +1 (positive class). Let $\mathbf{R}$ denote the set of real numbers, and d the dimension of the input vector. For an example $\mathbf{x} \in \mathbf{R}^d$ of class $y \in \{+1, -1\}$, let

- $c_y(\mathbf{x})$ denote the cost of misclassifying $\mathbf{x}$ belonging to class y
- and $g_y(\mathbf{x})$ the cost of classifying $\mathbf{x}$ correctly.

In our framework, gains are expressed as negative costs. I.e. $g_y(\mathbf{x}) < 0$ holds if there is a gain for classifying $\mathbf{x}$ correctly into class y.

Let $r : \mathbf{R}^d \longrightarrow \{+1, -1\}$ be a classifier (decision rule) that assigns $\mathbf{x}$ to a class. Let $X_y = \{\mathbf{x} \,|\, r(\mathbf{x}) = y\}$ be the region where class y is decided by r.

[1] As every classification problem our problem can be restated as a cost prediction, i.e. regression problem with e.g. a quadratic error function. But there is some evidence that classification is easier than regression [4]. In the cost-free case, DIPOL performed better than e.g. Backpropagation on several classification problems, see [15, 20].

The risk of r with respect to the probability density function p of $(\mathbf{x}, y)$ is given with $p(\mathbf{x}, y) = p(\mathbf{x}|y)P(y)$ as

$$R(r) = \sum_{\substack{y_1, y_2 \in \{+1,-1\} \\ y_1 \neq y_2}} \left[\int_{X_{y_1}} g_{y_1}(\mathbf{x}) p(\mathbf{x}|y_1) P(y_1) d\mathbf{x} + \int_{X_{y_1}} c_{y_2}(\mathbf{x}) p(\mathbf{x}|y_2) P(y_2) d\mathbf{x} \right] \quad (1)$$

(see also [19]). $P(y)$ is the prior probability of class y, and $p(\mathbf{x}|y)$ is the class conditional density of class y. The first integral expresses the cost for correct classification, whereas the second integral expresses the cost for misclassification. We assume that the integrals defining R do exist. This is the case if the cost functions are integrable and bounded.

In order to minimize the risk $R(r)$, an example $\mathbf{x}$ is assigned to class +1, if

$$0 \leq (c_{+1}(\mathbf{x}) - g_{+1}(\mathbf{x}))p(\mathbf{x}|{+1})P(+1) - (c_{-1}(\mathbf{x}) - g_{-1}(\mathbf{x}))p(\mathbf{x}|{-1})P(-1) \quad (2)$$

holds. We assume $c_y(\mathbf{x}) - g_y(\mathbf{x}) \geq 0$ for every example $\mathbf{x}$, i.e. there is a benefit for classifying $\mathbf{x}$ correctly.

From (2) it follows that the classification of examples depends on the *differences* of the costs for misclassification and correct classification, not on their actual values. Therefore we will assume $g_y(\mathbf{x}) = 0$ and $c_y(\mathbf{x}) > 0$ without loss of generality. This means for the credit risk problem that for good customers the cost of correct classification is set to zero. The misclassification cost of good customers is defined as the gain that is lost in this case.

The Bayes classifier (see e.g. [5]) for this simplified problem can be stated as

$$r^*(\mathbf{x}) = \text{sign}[c_{+1}(\mathbf{x})p(\mathbf{x}|{+1})P(+1) - c_{-1}(\mathbf{x})p(\mathbf{x}|{-1})P(-1)]. \quad (3)$$

We define sign(0) = +1 though the assignment of the class is arbitrary for the case $c_{+1}(\mathbf{x})p(\mathbf{x}|{+1})P(+1) - c_{-1}(\mathbf{x})p(\mathbf{x}|{-1})P(-1) = 0$.

Given a training sample $(\mathbf{x}^{(1)}, y^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(l)}, y^{(l)}, c^{(l)})$ with $c^{(i)} = c_{y^{(i)}}(x^{(i)})$, the empirical risk is defined by

$$R_{\text{emp}}(r) = \frac{1}{l} \sum Q(\mathbf{x}^{(i)}, y^{(i)}, r).$$

If the example is misclassified, it holds that $Q(\mathbf{x}^{(i)}, y^{(i)}, r) = c^{(i)}$; otherwise, it holds $Q(\mathbf{x}^{(i)}, y^{(i)}, r) = 0$. In our case, R_{emp} corresponds to the mean misclassification costs defined using the example dependent costs.

Proposition 2.1 ([19]) *If both cost functions are bounded by a constant B, then it holds with a probability of at least $1-\eta$*

$$R(r) \leq R_{\text{emp}}(r) + B\sqrt{\frac{h(ln\frac{2l}{h}+1) - ln\frac{\eta}{4}}{l}},$$

where h is the VC-dimension of the hypothesis space of r.

This result from [19] (p. 80) holds in our case, since the only assumption Vapnik made on the loss function is its non-negativity and boundedness.

Let $\bar{c}_{+1}$ and $\bar{c}_{-1}$ be the *mean* misclassification costs for the given distributions. Let r^+ be the Bayes optimal decision rule with respect to the class dependent costs $\bar{c}_{+1}$ and $\bar{c}_{-1}$. Then it is easy to see that $R(r^*) \leq R(r^+)$, where $R(r^*)$ (defined in (3)) and $R(r^+)$ are both evaluated with respect to the example dependent costs. This means that because the example dependent costs can be considered to be the real costs occuring, their usage can lead to decreased misclassification costs. Of course this is only possible if the learning algorithm is able to incorporate example dependent costs.

2.1 Noisy Costs

If the cost values are obtained together with the training sample, they may be corrupted due to measurement errors. This means that the cost values are prone to noise. A probabilistic noise model for the costs can be included into the definition of the risk (1) by considering a common distribution of $(\mathbf{x}, y, c)$ where c is the cost. In the case of a continuous random variable c, equation (1) can be reformulated (with $g_y = 0$) to

$$R(r) = \sum_{y_1 \neq y_2} \int_{X_{y_1}} [\int_{\mathbf{R}} c\ p(c|\mathbf{x}, y_2) p(\mathbf{x}|y_2) P(y_2) dc] d\mathbf{x}\,,$$

where $p(c|\mathbf{x}, y)$ is the probability density function of the cost given $\mathbf{x}$ and y.

It is easy to see that the cost functions c_y can be obtained as the expected value of the costs, i.e.

$$c_y(\mathbf{x}) := E[c\,|\mathbf{x}, y] = \int_{\mathbf{R}} c\ p(c|\mathbf{x}, y) dc \tag{4}$$

where we assume that the expected value exists. In the learning algorithms presented in the next sections, it is not necessary to compute (4) or estimate it before learning starts, because the necessary averaging is done by the learning algorithms.

3 Perceptrons

Now we assume, that a training sample $(\mathbf{x}^{(1)}, y^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(l)}, y^{(l)}, c^{(l)})$ is given with example dependent cost values $c^{(i)}$. We allow the cost values to be noisy, but for the moment, we will require them to be positive. In the following we derive a cost-sensitive perceptron learning rule for linearly non-separable classes, that is based on a non-differentiable error function. A perceptron (e.g. [5]) can be seen as representing a parameterized function defined by a vector $\mathbf{w} = (w_1, \ldots, w_n)^T$ of *weights* and a threshold θ. The vector $\bar{\mathbf{w}} = (w_1, \ldots, w_n, -\theta)^T$ is called the extended weight vector, whereas $\bar{\mathbf{x}} = (x_1, \ldots, x_n, 1)^T$ is called the extended input vector. We denote their scalar product as $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}$. The output function $y : R^d \longrightarrow \{-1, 1\}$ of the perceptron is defined by $y(\mathbf{x}) = \text{sign}(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}})$.

A weight vector having *zero* costs can be found in the *linearly separable case*, where a class separating hyperplane exists, by choosing an initial weight vector, and adding or subtracting examples that are not correctly classified (for details see e.g. [5]).

Because in many practical cases as the credit risk problem the classes are *not* linearly separable, we are interested in the behaviour of the algorithm for linearly non-separable classes. If the classes are linearly non-separable, they can either be non-separable at all (i.e. overlapping), or they are separable but not linearly separable.

3.1 The Criterion Function

In the following we will present the approach of Unger and Wysotzki for the linearly non-separable case [18] extended to the usage of individual costs. Other perceptron algorithms for the linearly non-separable case are discussed in [21, 5].

Let the step function σ be defined by $\sigma(u) = 1$ for $u \geq 0$, and $\sigma(u) = 0$ if $u < 0$. In the following, σ will be used as a function that indicates a classification error.

Let S_{+1} contain all examples from class $+1$ together with their cost value. S_{-1} is defined accordingly. For the derivation of the learning algorithm, we consider the **criterion function**

$$\begin{aligned} I_\epsilon(\bar{\mathbf{w}}) &= \frac{1}{l}\left[\sum_{(\mathbf{x},c)\in S_{+1}} c\,(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\sigma(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\right. \\ &\quad \left. + \sum_{(\mathbf{x},c)\in S_{-1}} c\,(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\sigma(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\right] \end{aligned} \tag{5}$$

that is to be minimized. The parameter $\epsilon > 0$ denotes a *margin* for classification. Each correctly classified example must have a geometrical distance of at least $\frac{\epsilon}{|\mathbf{w}|}$ to the hyperplane. The margin is introduced in order to exclude the zero weight vector as a minimizer of (5), see [18, 5].

The situation of the criterion function is depicted in fig. 1. In addition to the original hyperplane $H : \bar{\mathbf{w}} \cdot \bar{\mathbf{x}} = 0$, there exist two margin hyperplanes $H_{+1} : \bar{\mathbf{w}} \cdot \bar{\mathbf{x}} - \epsilon = 0$ and $H_{-1} : -\bar{\mathbf{w}} \cdot \bar{\mathbf{x}} - \epsilon = 0$. The hyperplane H_{+1} is now responsible for the classification of the class $+1$ examples, whereas H_{-1} is responsible for class -1 ones. Because H_{+1} is shifted into the class $+1$ region, it causes at least as many errors for class $+1$ as H does. For class -1 the corresponding holds.

It is relatively easy to see that I_ϵ is a convex function by considering the convex function $h(z) := k\, z\sigma(z)$ (where k is some constant), and the sum and composition of convex functions. From the convexity of I_ϵ it follows that there exists a unique minimum value.

It can be shown that the choice of an $\epsilon > 0$ is not critical, because the hyperplanes minimizing the criterion function are identical with respect to the empirical risk for every $\epsilon > 0$, see also [7].

3.2 The Learning Rule

By differentiating the criterion function I_ϵ, we derive the learning rule. The gradient of I_ϵ is given by

$$\nabla_{\bar{w}} I_\epsilon(\bar{\mathbf{w}}) = \frac{1}{l}\left[\sum_{(\mathbf{x},c)\in S_{+1}} -c\,\bar{\mathbf{x}}\sigma(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon) + \sum_{(\mathbf{x},c)\in S_{-1}} c\,\bar{\mathbf{x}}\sigma(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\right] \qquad (6)$$

To handle the points, in which I_ϵ cannot be differentiated, in [18] the gradient in (6) is considered to be a *subgradient*. For a subgradient a in a point $\bar{\mathbf{w}}$, the condition $I_\epsilon(\bar{\mathbf{w}}') \geq I_\epsilon(\bar{\mathbf{w}}) + a \cdot (\bar{\mathbf{w}}' - \bar{\mathbf{w}})$ for all $\bar{\mathbf{w}}'$ is required. The subgradient is defined for convex functions, and can be used for incremental learning and stochastic approximation (see [18, 2, 16]).

Considering the gradient for a single example, the following *incremental* rule can be derived. For learning, we start with an arbitrary initialisation $\bar{\mathbf{w}}(0)$. The following weight update rule is used when encountering an example

$(\mathbf{x}, y)$ with cost c at time (learning step) t:

$$\bar{\mathbf{w}}(t+1) = \begin{cases} \bar{\mathbf{w}}(t) + \gamma_t c\, \bar{\mathbf{x}} & \text{if } y = +1 \text{ and} \\ & \bar{\mathbf{w}}(t) \cdot \bar{\mathbf{x}} - \epsilon \leq 0 \\ \bar{\mathbf{w}}(t) - \gamma_t c\, \bar{\mathbf{x}} & \text{if } y = -1 \text{ and} \\ & \bar{\mathbf{w}}(t) \cdot \bar{\mathbf{x}} + \epsilon \geq 0 \\ \bar{\mathbf{w}}(t) & \text{else} \end{cases} \tag{7}$$

We assume either a randomized or a cyclic presentation of the training examples.

In order to guarantee convergence to a minimum and to prevent oscillations, for the factors γ_t the following conditions for stochastic approximation are imposed: $\lim_{t\to\infty} \gamma_t = 0$, $\sum_{t=0}^{\infty} \gamma_t = \infty$, $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. A possible choice is $\gamma_t = \frac{1}{t}$. The convergence to an optimum in the separable and the non-separable case follows from the results in [16].

If the cost value c is negative due to noise in the data, the example could just be ignored. This corresponds to modifying the density $p(\mathbf{x}, y, c)$ which is in general *not* desirable. Alternatively, the learning rule (7) must be modified in order to *mis*classify the current example. This can be achieved by using the modified update conditions $\text{sign}(c)\bar{\mathbf{w}}(t) \cdot \bar{\mathbf{x}} - \epsilon \leq 0$ and $\text{sign}(c)\bar{\mathbf{w}}(t) \cdot \bar{\mathbf{x}} + \epsilon \geq 0$ in (7). This means that an example with negative cost is treated as if it belongs to the other class.

4 Multiple and Disjunctive Classes

In order to deal with multi-class/multimodal problems (e.g. XOR), we have extended the learning system DIPOL [15, 17, 20] in order to handle example dependent costs.

The aim of the STATLOG project (see [15]) was the comparison of several algorithms from the fields of machine learning, statistical classification and neural networks (excluding SVMs). DIPOL turned out to be one of the most successful learning algorithms – it performed best on average on all datasets (see [20] for more details).

DIPOL can be seen as an extension of the perceptron approach to multiple classes and multi-modal distributions. A learning problem with two classes, where both classes have a bimodal distribution (i.e. there exist two clusters), is shown in fig. 2 (together with the hyperplanes learned by DIPOL). If a class possesses a multi-modal distribution (disjunctive classes), the clusters are determined by DIPOL in a preprocessing step using a minimum-variance clustering algorithm (see [17, 5]) for every class.

In the case of $N \geq 2$ classes, each example is described by an N-place cost vector describing the $N-1$ possibilities of misclassification and the cost for correct classification.

After the (optional) clustering of the classes, a separating hyperplane is constructed for each pair of classes or clusters if they belong to different classes. When creating a hyperplane for a pair of classes or clusters, respectively, all examples belonging to other classes/clusters are *not* taken into account. Of course, for clusters originating from the same class in the training set, no hyperplane has to be constructed.

The construction of a hyperplane for a pair of clusters or classes has two phases. In the weight initialization phase, a least squares problem is solved using standard regression techniques. To one of the classes, the target value +1 is assigned, whereas the other class has the target value –1. The weights of the regression hyperplane are used as initial values for the second phase in which a gradient descent is performed. In contrast to the approach described in section 3.2, DIPOL uses a sequence of γ_t that tends to zero in an exponential manner, yielding a much faster convergence and also good learning results because of the in general good initialization of the weights by the regression step.

After the construction of the hyperplanes, the whole feature space is divided into decision regions each belonging to a single class, or cluster respectively. For classification of a new example $\mathbf{x}$, it is determined in which region of the feature space it lies, i.e. a region belonging to a cluster of a class y. The class y of the respective region defined by a subset of the hyperplanes is the classification result for $\mathbf{x}$.

DIPOL can be trained using the criterion function I_ϵ or using a quadratic error function, e.g. [15, 17, 20]. It incorporates incremental gradient descent (sect. 3.2) as well as a modified batch mode procedure, where the learning rate decays exponentially.

In the new version of DIPOL, example dependent costs can be included in every step:

- In the clustering step, the costs can be used as an additional attribute, possibly yielding a finer clustering of the data.

- In the regression step, the costs are used as target values. For the class, that is considered to be the –1-class, the costs are multiplied by –1. In the case of $N > 2$ classes, the appropriate entries of the cost vector are considered.

- In the gradient descent phase, the costs are incorporated as described in section 3.2.

In the next section, an alternative approach for the cost-sensitive construction of linear hyperplanes is considered.

5 Support Vector Machines

5.1 SVMs with Example Dependent Costs

DIPOL constructs a classifier by dividing the given input space into regions belonging to different classes. The classes are separated by hyperplanes computed with the algorithm in sect. 3. In the SVM approach, hyperplanes are not computed by gradient descent but by directly solving an optimization problem, see below. More complex classifiers are formed by an implicit transformation of the given input space into a so called feature space by using kernel functions.

Given a training sample $(\mathbf{x}^{(1)}, y^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(l)}, y^{(l)}, c^{(l)})$, the optimization problem of a standard soft margin support vector machine (SVM) [19, 3] can be stated as

$$\begin{array}{lrcl} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}|\mathbf{w}|^2 + C\sum\limits_{i=1}^{l}\xi_i^k & & \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}\cdot\mathbf{x}^{(i)}+b\right) & \geq & 1-\xi_i \\ & \xi_i & \geq & 0, \end{array} \tag{8}$$

where the regularization constant $C > 0$ determines the trade-off between the complexity term $\frac{1}{2}|\mathbf{w}|^2$ and the sum. It holds that $b = -\theta$. The sum takes all examples into account for which the corresponding pattern $\mathbf{x}^{(i)}$ has a geometrical margin of less than $\frac{1}{|\mathbf{w}|}$, and a functional margin of less than 1. For such an example, the slack variable $\xi_i > 0$ denotes the difference to the required functional margin. Different values of k lead to different versions of the soft margin SVM, see e.g. [3].

For k=1, the sum of the ξ_i can be seen as an upper bound of the empirical risk. Hence we can extend the optimization problem (8) in a natural way by weighting the slack variables ξ_i with the corresponding costs $c^{(i)}$. This leads for $k = 1$ to the cost-sensitive optimization problem[2]

$$\begin{array}{lrcl} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}|\mathbf{w}|^2 + C\sum\limits_{i=1}^{l}c^{(i)}\,\xi_i & & \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}\cdot\mathbf{x}^{(i)}+b\right) & \geq & 1-\xi_i \\ & \xi_i & \geq & 0. \end{array} \tag{9}$$

[2]A similar approach is taken in [9] for modeling concept drift. There, the weights correspond to the recency of the examples.

Introducing non-negative Lagrangian multipliers α_i, $\mu_i \geq 0$, $i = 1, \ldots, l$, we can rewrite the optimization problem (9), and obtain the following primal Lagrangian

$$\begin{aligned} L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \; & \frac{1}{2}|\mathbf{w}|^2 + C \sum_{i=1}^{l} c^{(i)} \, \xi_i \\ & - \sum_{i=1}^{l} \alpha_i \left[y^{(i)} \left(\mathbf{w} \cdot \mathbf{x}^{(i)} + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{l} \mu_i \, \xi_i. \end{aligned}$$

Substituting the derivatives with respect to $\mathbf{w}$, b and $\boldsymbol{\xi}$ into the primal, we obtain the dual Langragian that has to be maximized with respect to the α_i,

$$L_D(\boldsymbol{\alpha}) \;=\; \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \, \alpha_j \, y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}. \tag{10}$$

Equation (10) defines the 1-norm soft margin SVM. Note that the example dependent costs do not occur in L_D, but restrict the α_i by the so called box constraints

$$\forall i \quad 0 \leq \alpha_i \leq c^{(i)} \, C$$

that depend on the cost value for the respective example and therefore limit its possible influence. The box constraints can be derived from the optimization problem, see e.g. [3].

If the optimal decision function is not a linear function of the data, we map the input data to some other Euclidean Space $\mathcal{H}$ (possibly of infinite dimension), the feature space, by means of a mapping $\boldsymbol{\phi} : \mathbf{R}^d \to \mathcal{H}$. Substituting the mapped data into the optimization problem leads to the dual Lagrangian

$$L_D(\boldsymbol{\alpha}) \;=\; \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \, \alpha_j \, y^{(i)} y^{(j)} \boldsymbol{\phi}(\mathbf{x}^{(i)}) \cdot \boldsymbol{\phi}(\mathbf{x}^{(j)}). \tag{11}$$

By means of kernel functions $K : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$, with the property $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$, we are able to evaluate the inner product in $\mathcal{H}$ without explicitly knowing $\boldsymbol{\phi}$.

For $k = 2$ (the 2-norm soft margin SVM) analogous results can be obtained where the dual Lagrangian depends directly on the individual costs:

$$L_D(\alpha) \;=\; \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \, K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \frac{1}{2} \sum_{i=1}^{l} \frac{\alpha_i^2}{c^{(i)} C}.$$

5.2 Relation to perceptron learning

In order to show the relationship between the criterion function I_ϵ in (5) and the learning problem of the SVM consider the case $k = 1$. In the limit $C \to \infty$ the sum of the ξ_i of the objective function is minimized. By means of the non-negativity function $(\cdot)_+$, with $(u)_+ = u$ if $u > 0$ and $(u)_+ = 0$ otherwise, both constraints can be integrated into the single inequality

$$\begin{aligned} \xi_i &\geq \left(1 - y^{(i)}\left[\mathbf{w}\cdot\mathbf{x}^{(i)} + b\right]\right)_+ && (12)\\ &= \left(1 - y^{(i)}\left[\mathbf{w}\cdot\mathbf{x}^{(i)} + b\right]\right)\sigma\left(1 - y^{(i)}\left[\mathbf{w}\cdot\mathbf{x}^{(i)} + b\right]\right), && (13)\end{aligned}$$

where we used $(u)_+ = u\,\sigma(u)$ in order to indicate classification errors. Note that equality in (12) holds for all patterns. Substituting (13) into the term $\sum_{i=1}^{l} c^{(i)}\,\xi_i$ of the objective function leads to the minimization problem

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{l} c^{(i)}\left(1 - y^{(i)}\left[\mathbf{w}\cdot\mathbf{x}^{(i)} + b\right]\right)\sigma\left(1 - y^{(i)}\left[\mathbf{w}\cdot\mathbf{x}^{(i)} + b\right]\right). \qquad (14)$$

Using the sets $S_{\pm 1}$ and the extended vectors $\bar{\mathbf{w}}$ and $\bar{\mathbf{x}}$ equation (14) becomes

$$\min_{\mathbf{w},b} \sum_{(\mathbf{x},c)\in S_{+1}} c\,(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}} + 1)\,\sigma\,(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}} + 1) + \sum_{(\mathbf{x},c)\in S_{-1}} c\,(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}} + 1)\,\sigma\,(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}} + 1),$$

which is equivalent to the I_ϵ criterion function in (5) with $\epsilon = 1$.

5.3 Convergence

Lin showed in [12] that the 2-norm SVM approximates the Bayes rule in the limit $l \to \infty$. For that purpose he treats the SVM optimization problem as the following regularization problem in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$

$$\begin{aligned} \min_{h,b,\xi} \quad & \frac{1}{l}\sum_{i=1}^{l}\xi_i^2 + \lambda|h|^2_{\mathcal{H}_K} \\ \text{s.t.} \quad & \xi_i \geq 1 - y^{(i)} f(\mathbf{x}^{(i)}) \\ & \xi_i \geq 0, \end{aligned}$$

with $f(\mathbf{x}) = h(\mathbf{x}) + b$ and an appropriate trade-off $\lambda = \frac{1}{2lC}$. The corresponding regularization problem with example dependent costs can be stated as

$$\min_{h,b,\xi} \quad \frac{1}{l}\sum_{i=1}^{l} c^{(i)}\,(\underbrace{1 - y^{(i)} f(\mathbf{x}^{(i)})}_{\equiv\xi_i})_+^2 + \lambda\|h\|^2_{\mathcal{H}_K}, \qquad (15)$$

where we integrated the constraints by means of the non-negativity function $(\cdot)_+$ into a single inequality which is substituted into the objective function.

In the limit $l \to \infty$ the upper bound of the empirical risk in (15) converges to the expectation

$$E_{\mathbf{X},Y}\left[c_Y(\mathbf{X})(1 - Y f(\mathbf{X}))_+^2\right], \tag{16}$$

where we introduced random variables $\mathbf{X}$ and Y. Minimizing $E_Y[\cdot]$ of the equivalent expression

$$E_{\mathbf{X}}\left[E_Y[c_Y(\mathbf{X})(1 - Y f(\mathbf{X}))_+^2 | \mathbf{X}]\right]$$

for every fixed $\mathbf{X} = \mathbf{x}$ leads to the minimization of

$$g \;=\; c_{-1}(\mathbf{x})(1 + f(\mathbf{x}))_+^2\, p(-1|\mathbf{x}) + c_{+1}(\mathbf{x})(1 - f(\mathbf{x}))_+^2\, p(+1|\mathbf{x}), \tag{17}$$

with $c_{y'}(\mathbf{x}^{(i)}) = c^{(i)}$ if $y' = y^{(i)}$ and 0 otherwise.

It can be shown that the range of the optimal function lies in the interval $f^{opt}(\mathbf{x}) \in [-1, +1]$. Therefore (17) remains non-negative for all $\mathbf{x}$ and we can drop the non-negativity function $(\cdot)_+$. By setting $z := f(\mathbf{x})$ and solving $\frac{\partial g}{\partial z} = 0$, we derive the optimal decision function

$$f^{opt}(\mathbf{x}) = \frac{c_{+1}(\mathbf{x})p(+1|\mathbf{x}) - c_{-1}(\mathbf{x})p(-1|\mathbf{x})}{c_{+1}(\mathbf{x})p(+1|\mathbf{x}) + c_{-1}(\mathbf{x})p(-1|\mathbf{x})}.$$

Proposition 5.1 *In the case $k = 2$, $sign(f^{opt}(\mathbf{x}))$ is a minimizer of R, and it minimizes* (16)*. Moreover it holds*

$$sign(f^{opt}) \equiv r^*.$$

where r^* is defined in eq. (3).

Therefore we conjecture from proposition 5.1 that SVM learning approximates the Bayes rule for large training sets. For $k = 1$ the corresponding cannot be shown.

6 Re-Sampling

Example dependent costs can be included into a cost-insensitive learning algorithm by re-sampling the given training set. First we define the mean costs for each class by

$$\bar{c}_y = \int_{\mathbf{R}^d} c_y(\mathbf{x}) p(\mathbf{x}|y) d\mathbf{x}\,. \tag{18}$$

We define the global mean cost $b = \bar{c}_{+1}P(+1) + \bar{c}_{-1}P(-1)$. From the cost-sensitive definition of the risk in (1) it follows that

$$\begin{aligned}\frac{R(r)}{b} &= \int_{X_{+1}} \frac{c_{-1}(\mathbf{x})p(\mathbf{x}|-1)}{\bar{c}_{-1}} \frac{\bar{c}_{-1}P(-1)}{b} d\mathbf{x} \\ &+ \int_{X_{-1}} \frac{c_{+1}(\mathbf{x})p(\mathbf{x}|+1)}{\bar{c}_{+1}} \frac{\bar{c}_{+1}P(+1)}{b} d\mathbf{x}.\end{aligned}$$

This means that we now consider the new class conditional densities

$$p'(\mathbf{x}|y) = \frac{1}{\bar{c}_y} c_y(\mathbf{x})p(\mathbf{x}|y)$$

derived from the compound density

$$p'(\mathbf{x}, y) = p'(\mathbf{x}|y)P'(y) = \frac{c_y(\mathbf{x})}{b} p(\mathbf{x}, y). \tag{19}$$

and the new priors

$$P'(y) = P(y) \frac{\bar{c}_y}{\bar{c}_{+1}P(+1) + \bar{c}_{-1}P(-1)}\,.$$

It is easy to see that $\int p'(\mathbf{x}|y)d\mathbf{x} = 1$ holds, as well as $P'(+1) + P'(-1) = 1$.

Because b is a constant, minimizing the cost-sensitive risk $R(r)$ is equivalent to minimizing the cost-free risk

$$\begin{aligned}\frac{R(r)}{b} = R'(r) &= \int_{X_{+1}} p'(\mathbf{x}|-1)P'(-1)d\mathbf{x} \\ &+ \int_{X_{-1}} p'(\mathbf{x}|+1)P'(+1)d\mathbf{x}.\end{aligned}$$

In order to minimize R', we have to draw a new training sample from the given training sample. Assume that a training sample $(\mathbf{x}^{(1)}, y^{(1)}, c^{(1)})$, ... , $(\mathbf{x}^{(l)}, y^{(l)}, c^{(l)})$ of size l is given. Let C_y be the total cost for class y in the sample. Based on the given sample, we form a second sample of size lN by random sampling from the given training set, where $N > 0$ is a fixed real number.

Because of (19), in each of the $\lfloor lN \rfloor$ independent sampling steps, the probability of including example i in this step into the new sample should be determined by

$$\frac{c^{(i)}}{C_{+1} + C_{-1}}$$

i.e. an example is chosen according to its contribution to the total cost of the fixed training set. Note that $\frac{C_{+1}+C_{-1}}{l} \approx b$ holds. Because of $R(r) = bR'(r)$, it holds $R_{\text{emp}}(r) \approx b\, R'_{\text{emp}}(r)$, where R_{emp} is evaluated with respect to the given sample, and $R'_{\text{emp}}(r)$ is evaluated with respect to the generated cost-free sample. This means that a learning algorithm that tries to minimize the expected cost-free risk by minimizing the mean cost-free risk will minimize the expected cost for the original problem. From the new training set, a classifier for the cost-sensitive problem can be learned with a cost-insensitive learning algorithm.

Our approach is related to the resampling approach for class dependent costs described e.g. in [1], and to the extension of the METACOST-approach for example dependent costs [22]. We will compare their performances in future experiments.

7 Experiments

7.1 The uni-modal case

If the classes are linearly separable, each separating hyperplane also minimizes the cost-sensitive criterion function I_ϵ. We therefore do not present results for the linearly separable case here. In our first experiment, we used the perceptron algorithm for the linearly non-separable case (sect. 3.2), that is part of DIPOL, and for the extended SVM with a radial basis function kernel.

We have constructed an artificial data set with two attributes x_1 and x_2. For each class, 1000 randomly chosen examples were generated using a modified Gaussian distribution with mean $(0.0, \pm 1.0)^T$. The covariance matrix for both classes is the unit matrix.

The individual costs of class +1 are defined using the function $c_{+1}(x_1, x_2) = 2\frac{1}{1+e^{-x_1}}$. The costs of the class –1 examples were defined in a similar way by the function $c_{-1}(x_1, x_2) = 2\frac{1}{1+e^{x_1}}$. This means that for $x_1 > 0$ the +1-examples have larger misclassification costs, whereas for $x_1 < 0$ the -1-examples have larger costs. The cost functions are shown in fig. 3 (left). The dataset together with the resulting hyperplane for $\epsilon = 0.1$ is depicted in fig. 4 (left, bold line). Other ϵ-values produced similar results. Without costs, a line close to the x_1-axis was produced (fig. 4, left, dashed line). With only class dependent misclassification costs, lines are produced that are almost parallel to the x_1 axis and that are shifted into the class region of the less dangerous class (not displayed in fig. 4). For the case of example dependent costs, analogous results are achieved by the extended SVM (fig. 4, right).

Our selection of the individual cost functions caused a *rotation* of the class boundary, see fig. 4. This effect cannot be reached using cost matrices alone. Our approach is therefore a genuine extension of previous approaches for including costs, which rely on class dependent costs or cost matrices.

7.2 The multi-modal case

For the multi-modal case, we have created the artificial dataset that is shown in fig. 5. Each class consists of two modes, each defined by a Gaussian distribution.

For class +1, we have chosen a constant cost $c_{+1}(x_1, x_2) = 1.0$. For class -1 we have chosen a variable cost, that depends only on the x_1-value, namely $c_{-1}(x_1, x_2) = 2\frac{1}{1+e^{-x_1}}$. This means, that the examples of the left cluster of class -1 (with $x_1 < 0$) have smaller costs compared to the class +1 examples, and the examples of the right cluster (with $x_1 > 0$) have larger costs. The cost functions are shown in fig. 3 (right).

For learning, the augmented version of DIPOL was provided with the 2000 training examples together with their individual costs. The result of the learning algorithm is displayed in fig. 5. For reasons of symmetry, the separating hyperplanes that would be generated *without* individual costs must coincide with one of the bisecting lines of the coordinate system. It is obvious in fig. 5, that this is not the case for the hyperplanes that DIPOL has produced for the dataset *with* the individual costs: The left region of class -1 is a little bit smaller, the right region is a little bit larger compared to learning without costs. Both results are according to the intuition.

The solution of the extended SVM with a radial basis function kernel results in the same shift of the class regions. Due to a higher sensitivity to outliers the decision boundary is curved in contrast to the piecewise linear hyperplanes generated by DIPOL.

7.3 German Credit Data Set

In order to apply our approach to a real world domain, we also conducted experiments on the German Credit Data Set ([15], chapter 9) from the STATLOG project (the dataset can be downloaded from the UCI repository). The data set has 700 examples of class "good customer" (class +1) and 300 examples of class "bad customer" (class -1). Each example is described by 24 attributes. Because the data set does not come with example dependent costs, we assumed the following cost model: If a good customer is incorrectly classified as a bad customer, we assumed the cost of $0.1\frac{duration}{12} \cdot amount$, where *duration* is the duration of the credit in months, and *amount* is the

credit amount. We assumed an effective yearly interest rate of $0.1 = 10\%$ for every credit, because the actual interest rates are not given in the data set. If a bad customer is incorrectly classified as a good customer, we assumed that 75% of the whole credit amount is lost (normally a customer will pay back at least part of the money). In the following, we will consider these costs as the real costs of the single cases.

In our experiments we wanted to compare the results using example dependent costs with the results when a cost matrix is used. We constructed the cost matrix $\begin{pmatrix} 0 & 6.27 \\ 29.51 & 0 \end{pmatrix}$, where 6.27 is the average cost for the class $+1$ examples, and 29.51 is the average cost for the class -1 examples (the credit amounts were normalized to lie in the interval [0,100]).

In our experiment we used cross validation to find the optimal parameter settings (cluster numbers) for DIPOL, i.e. the optimal cluster numbers, and to estimate the mean *predictive* cost R_{emp} using the 10%-test sets. When using the individual costs, the estimated mean predictive cost was 3.67 (the default cost is 4.38).

In a second cross validation experiment, we determined the optimal cluster numbers when using the cost matrix for learning and for evaluation. For these optimal cluster numbers, we performed a second cross validation run, where the classifier is constructed using the cost matrix for the respective training set, but evaluated on the respective test set using the example dependent costs. Remember, that we assumed the example dependent costs as described above to be the real costs for each case. This second experiment leads to an estimated mean predictive cost of 3.98. Using the matrix for learning and the individual costs for model selection produced somewhat better results.

This means that in the case of the German Credit Dataset, we achieved a 7.8% reduction in cost using example dependent costs instead of a cost matrix. The classifiers constructed using the cost matrix alone performed worse than the classifiers constructed using the example dependent costs.

We also compared the performance of the cost-sensitive extension of DIPOL to the performance of DIPOL on a dataset obtained from resampling. In each cross validation run, we replaced the training set of that run with a resampled cost-free version of the same size (900 examples). The mean predictive costs averaged over 10 cross validation runs were 3.72 for the optimal parameter setting. This means that there is no significant decrease in performance compared to 3.67 in the case of example dependent costs. When using oversampled training sets with 9000 examples, the costs are 3.61, i.e. even slightly better.

The extended SVM generated similar results for the usage of the cost

matrix and the example dependent costs respectively, i.e. we found no substantially increased performance. The reason is presumably that the results for DIPOL, the SVM, and other learning algorithms are not much better than the default rule, see [15], though DIPOL and SVM perform comparably well.

7.4 The KDD-98 dataset

The KDD-98 dataset [8] contains informations on persons that were mailed during a campaign with requests to donate to a charity. In the dataset there are two classes. The first class consists of persons that received a mail but did not give some money (non-donators). The second class consists of those persons that received a mail and spent some money (donators). The learned classifier is intended to be used to decide whom to send a request in a future campaign.

For a non-donator, the misclassification cost corresponds to the cost of the mail that is estimated as \$0.68. The cost for correct classification is zero which captures the case of deciding not to send a letter. For donators, the cost for misclassification is also set to zero, because in this case no letter would have been sent. The costs (i.e. gain) for correct classification is the donation amount minus the cost for the mail. This value varies between \$–0.32 and \$–199.32.

For learning with the training set that contains 95413 examples, we used a normalized version of the dataset, as described in section 2 (zero cost for correct classification). For testing we used the validation dataset containing 96386 examples and the original cost functions giving the estimated total gain of the future mailing campaign. The default gain when mailing to every person is \$10560.

Using DIPOL together with example dependent costs, the gain was \$12163 while the extended 2-norm SVM with a radial basis function kernel reaches \$12374. We also created resampled data from the original dataset. Using this dataset, we achieved an increased gain of \$12883 with a "vanilla" SVM in contrast to \$14045 with DIPOL (averaged over 10 resampled datasets). Note that the latter result is only slightly worse than the winner of the KDD-98 competition (\$14712).

In [23] slightly better results are reported for a modified sampling strategy[3] that is combined with averaging classifiers ("costing"). We assume that our results can be improved too by using a multi-classifier approach. Obviously, the inclusion of duplicates that can occur during resampling does not

[3]The strategy in [23] was developed independently from ours.

play a negative role for DIPOL.

We conjecture that for DIPOL the increase in gain is due to the fact that in the resampled dataset, the two classes are rather balanced in contrast to the original dataset. Seemingly, this makes learning easier for DIPOL. First investigations have shown that in the case of the resampled data set, the regression step already produces very accurate hyperplanes. For the original dataset, this is not always the case.

A general advantage of DIPOL over the resampling strategy is the treatment of more than two classes. While DIPOL can be applied in this case without problems, there is no straightforward way to extend the resampling strategy.

8 Conclusion

In this article we discussed a natural cost-sensitive extension of perceptron learning with example dependent costs for correct classification and misclassification. We stated an appropriate criterion function and derived a cost-sensitive learning rule for linearly non-separable classes from it, that is a natural extension of the cost-insensitive perceptron learning rule for separable classes.

We showed that the Bayes rule only depends on differences between costs for correct classification and for misclassification. This allows us to define a simplified learning problem where the costs for correct classification are assumed to be zero. In addition to costs for correct and incorrect classification, it would be possible to consider example dependent costs for rejection, too.

The usage of example dependent costs instead of class dependent costs leads to a decreased misclassification cost in practical applications, e.g. credit risk assignment.

Experiments with the extended SVM approach verified the results of perceptron learning. Its main advantage lies in a lower error at the expense of non-interpretable decision boundaries. The piecewise linear classifier of DIPOL can easily be transformed to disjunctive rules with linear inequalities.

With respect to the resampling strategy presented in this paper, we have shown that it performs well and may even lead to an increased performance. The theoretical properties of the approach have to be investigated in future work. For example there is the possibility of including duplicates of single examples that lead to a bias.

Acknowledgments. We thank Bianca Zadrozny for providing a version of KDD-98 dataset with a reduced number of attributes.

References

[1] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Knowledge Discovery and Data Mining (Proc. KDD98)*, pages 164–168, 1998.

[2] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Math. Soc. Series of Monographs and Advanced Texts. John Wiley & Sons, 1983.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods).* Cambridge University Press, 2000.

[4] L. Devroye, L. Györfi, and L. Gabor. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[6] Charles Elkan. The foundations of Cost-Sensitive learning. In Bernhard Nebel, editor, *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 973–978, San Francisco, CA, August 4–10 2001. Morgan Kaufmann Publishers, Inc.

[7] P. Geibel and F. Wysotzki. Using costs varying from object to object to construct linear and piecewise linear classifiers. Technical Report 2002-5, TU Berlin, Fak. IV (WWW http://ki.cs.tu-berlin.de/∼geibel/publications.html), 2002.

[8] S. Hettich and S.D. Bay. The UCI KDD archive. University of California, Irvine. `http://kdd.ics.uci-edu/`.

[9] R. Klinkenberg and S. Rüping. Concept drift and the importance of examples. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining – Theoretical Aspects and Applications*. Springer, 2003.

[10] M. Kukar and I. Kononenko. Cost-sensitive learning with neural networks. In Henri Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 445–449, Chichester, 1998. John Wiley & Sons.

[11] A. Lenarcik and Z. Piasta. Rough classifiers sensitive to costs varying from object to object. In Lech Polkowski and Andrzej Skowron, editors, *Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing (RSCTC-98)*, volume 1424 of *LNAI*, pages 222–230, Berlin, June 22–26 1998. Springer.

[12] Y. Lin. Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 6(3):259–275, 2002.

[13] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1-3):191–202, 2002.

[14] D. D. Margineantu and T. G. Dietterich. Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 583–590. Morgan Kaufmann, San Francisco, CA, 2000.

[15] D. Michie, D. H. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Series in Artificial Intelligence. Ellis Horwood, 1994.

[16] A. Nedic and D.P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, pages 109–138, 2001.

[17] B. Schulmeister and F. Wysotzki. Dipol - a hybrid piecewise linear classifier. In R. Nakeiazadeh and C. C. Taylor, editors, *Machine Learning and Statistics: The Interface*, pages 133–151. Wiley, 1997.

[18] S. Unger and F. Wysotzki. *Lernfähige Klassifizierungssysteme (Classifier Systems that are able to Learn)*. Akademie-Verlag, Berlin, 1981.

[19] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[20] F. Wysotzki, W. Müller, and B. Schulmeister. Automatic construction of decision trees and neural nets for classification using statistical considerations. In G. DellaRiccia, H.-J. Lenz, and R. Kruse, editors, *Learning, Networks and Statistics*, number 382 in CISM Courses and Lectures. Springer, 1997.

[21] J. Yang, R. Parekh, and V. Honavar. Comparison of performance of variants of single-layer perceptron algorithms on non-separable data. *Neural, Parallel and Scientific Computation*, 8:415–438, 2000.

[22] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In Foster Provost and Ramakrishnan Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 204–214, New York, August 26–29 2001. ACM Press.

[23] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the 2003 IEEE International Conference on Data Mining*, 2003. to appear.

List of Figures

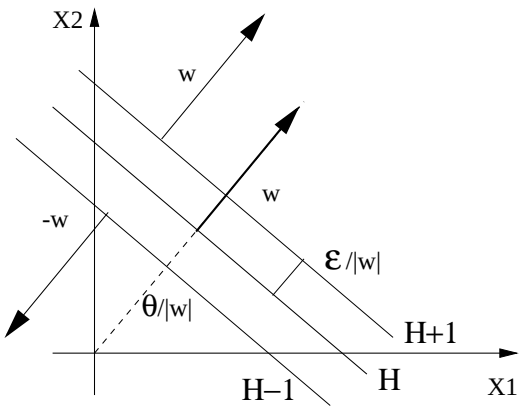

Figure 1: Geometrical interpretation of the margins, 2-dimensional case

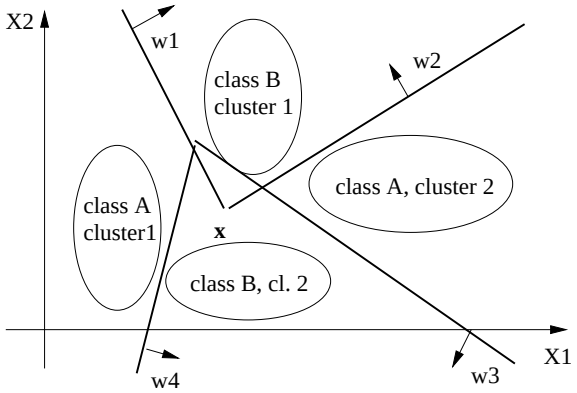

Figure 2: A two class, multi-modal learning problem with hyperplanes generated by DIPOL.

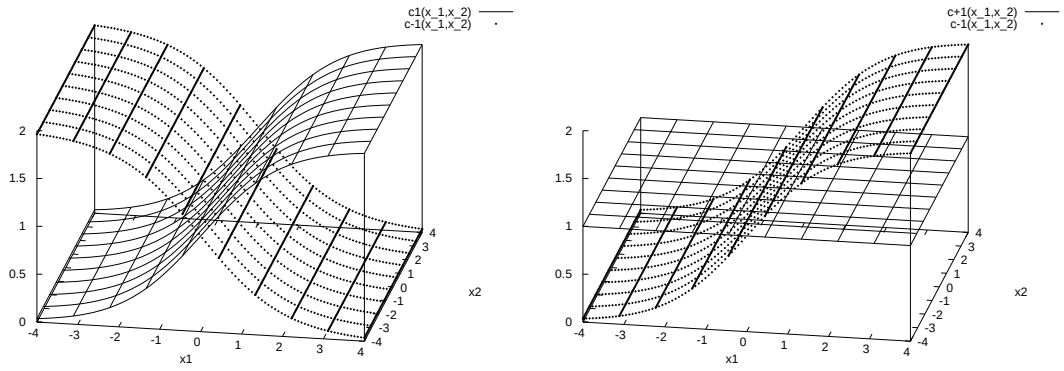

Figure 3: Cost functions in the non-separable case (left), and the multi-modal case (right).

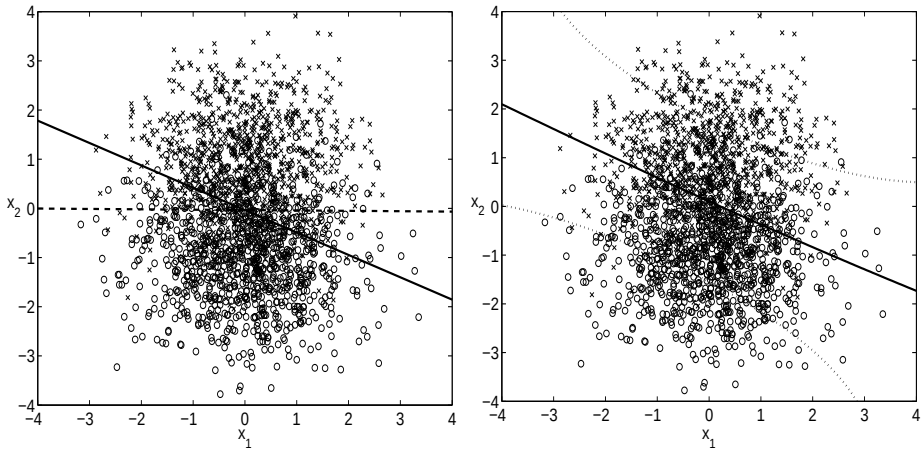

Figure 4: Results for the non-separable case. The two classes $+1$ and -1 are visualized by crosses and circles respectively. Hyperplanes (bold) generated by DIPOL (left), and the class boundary for the extended SVM (right). The DIPOL solution for the cost-free case (dashed) and the margin of the SVM (dotted) are shown additionally.

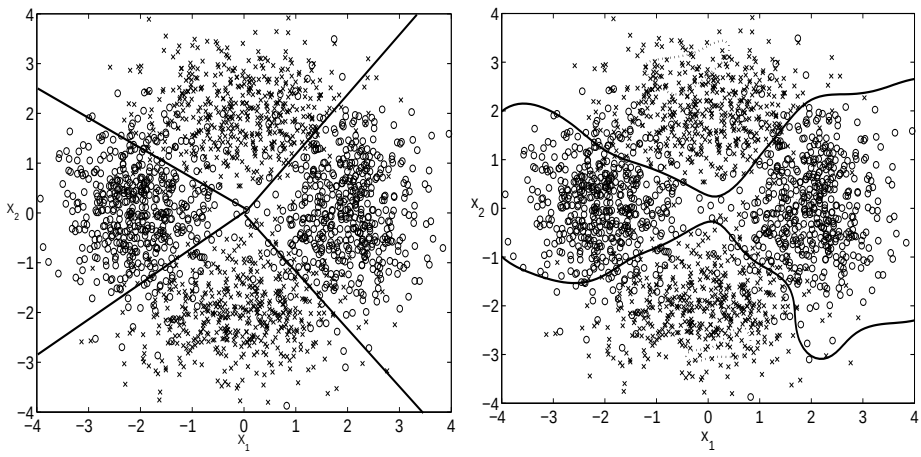

Figure 5: Results for the multi-modal non-separable case. Bold lines visualize the learned hyperplanes by DIPOL (left) and the extended SVM (right).