Learning Linear Classifiers Sensitive to Example Dependent and Noisy Costs*

Peter Geibel, Ulf Brefeld, and Fritz Wysotzki

TU Berlin, Fak. IV, ISTI, AI Group, Sekr. FR5-8
TU Berlin
Franklinstr. 28/29, D-10587 Berlin, Germany
Email {geibel|brefeld|wysotzki}@cs.tu-berlin.de

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).

1 Introduction

The consideration of cost-sensitive learning has received growing attention in the past years ([9, 4, 6, 8]). 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. [9, 4]). 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. [7]). 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 to 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 corresponds to the gain of the bank expressed as negative costs. This cost will

* *Proc. of the International Symposium on Intelligent Data Analysis*, ©Springer, 2003.

in general depend on the credit loan, the interest rate etc. Generally there are no costs to be expected (or a small loss related to the handling expenses) if the customer is rejected, for he is incorrectly classified as a bad customer[1]. For a bad customer the misclassification cost is simply the actual loss that has occured. The costs 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, we claim that using the example costs directly is more natural and will lead to more accurate classifiers.

In this paper we consider single neuron perceptron learning and the algorithm DIPOL introduced in [10, 12, 15] 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 multimodal data is the Support Vector Machine (SVM, [14]). We will demonstrate how to extend the SVM with example dependent costs, and compare its performance to the results obtained using DIPOL.

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. Experiments on two artificial data sets, and on a credit data set can be found in section 6. The conclusion is presented in section 7.

2 Example Dependent Costs

In the following we consider binary classification problems with classes –1 (negative class) and +1 (positive class). 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. $g_y(\mathbf{x}) < 0$ if there is a gain for classifying $\mathbf{x}$ correctly into class y. $\mathbf{R}$ denotes the set of real numbers. d is the dimension of the input vector.

Let $r : \mathbf{R}^d \longrightarrow \{+1, -1\}$ be a classifier (decision rule) that assigns $\mathbf{x}$ to a class. Let $X_y = \{\mathbf{x} \,|\, r(\mathbf{x}) = y\}$ the region where class y is decided. According to [14] 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

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

[1] Where it is assumed that the money can be given to another customer without loss of time.

$P(y)$ 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 exist. This is the case if the cost functions are integrable and bounded.

The risk $R(r)$ is minimized if $\mathbf{x}$ is assigned to class +1, if

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

holds. This rule is called the Bayes classifier (see e.g. [3]). We assume $c_y(\mathbf{x}) - g_y(\mathbf{x}) > 0$ for every example $\mathbf{x}$, i.e. there is a real 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. E.g. in the credit risk problem, 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).

2.1 Noisy Costs

If the cost values are obtained together with the training sample, they may be corrupted due to measurement errors. I.e. 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. (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's easy to see that the cost functions c_y can be obtained as the expected value of the costs, i.e.

$$c_y(\mathbf{x}) := E_c[c\, p(c|\mathbf{x}, y)] \qquad (3)$$

where we assume that the expected value exists. In the learning algorithms presented in the next sections, it's not necessary to compute (3) or estimate it before learning starts.

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. [3]) 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 $y(\mathbf{x}) = \text{sign}(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}})$. We define $\text{sign}(0) = 1$.

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. [3]).

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 ([13]) extended to the usage of individual costs. Other perceptron algorithms for the linearly non-separable case are discussed in [16, 3].

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**

$$I_\epsilon(\bar{\mathbf{w}}) = \frac{1}{l}\left[\sum_{(\mathbf{x},c)\in S_{+1}} c\,(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\sigma(-\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon) + \sum_{(\mathbf{x},c)\in S_{-1}} c\,(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\sigma(\bar{\mathbf{w}}\cdot\bar{\mathbf{x}}+\epsilon)\right] \tag{4}$$

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 (4), see [13, 3].

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 much errors for class $+1$ as H does. For class -1 the corresponding holds.

It's 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

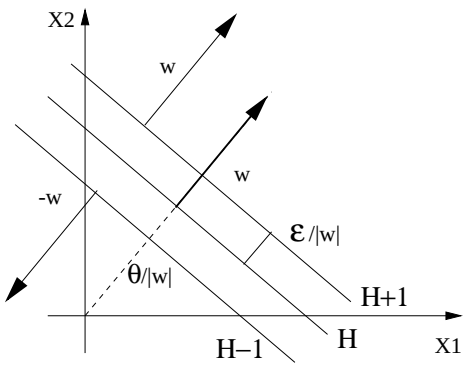

Fig. 1. Geometrical interpretation of the margins, 2-dimensional case

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 for every $\epsilon > 0$, see also [5].

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

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

To handle the points, in which I_ϵ cannot be differentiated, in [13] the gradient in (5) 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 [13, 1, 11]).

Considering the gradient for a single example, the following *incremental* rule can be designed. 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:

$$\bar{\mathbf{w}}(t+1) = \begin{cases} \bar{\mathbf{w}}(t) + \gamma_t c\,\bar{\mathbf{x}} & \text{if } y=+1 \text{ and} \\ & \bar{\mathbf{w}}(t)\cdot\bar{\mathbf{x}} - \epsilon \leq 0 \\ \bar{\mathbf{w}}(t) - \gamma_t c\,\bar{\mathbf{x}} & \text{if } y=-1 \text{ and} \\ & \bar{\mathbf{w}}(t)\cdot\bar{\mathbf{x}} + \epsilon \geq 0 \\ \bar{\mathbf{w}}(t) & \text{else} \end{cases} \tag{6}$$

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: $\lim_{t\to\infty} \gamma_t = 0$, $\sum_{t=0}^{\infty} \gamma_t = \infty$, $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. This algorithm for linearly non-separable distributions will converge to a solution vector in the linearly separable case (for a discussion see [5]).

If the cost value c is negative due to noise in the data, the example can just be ignored. This corresponds to modifying the density $p(\mathbf{x}, y, c)$ which is in general *not* desirable. Alternatively, the learning rule (6) 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 (6). 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 ([10, 12, 15]) in order to handle example dependent costs. DIPOL can be seen as an extension of the perceptron approach to multiple classes and multi-modal distributions. If a class possesses a multimodal distribution (disjunctive classes), the clusters are determined by DIPOL in a preprocessing step using a minimum-variance clustering algorithm (see [12, 3]) for every class.

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.

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 $\mathbf{x}$.

In contrast to the version of DIPOL described in [10, 12, 15], that uses a quadratic criterion function and a modified gradient descent algorithm, we used the criterion function I_ϵ and the incremental learning rule in sect. 3.2 for the experiments.

5 Support Vector Machines

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) [14, 2] can be stated as

$$\begin{array}{ll} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}|\mathbf{w}|^2 + C\sum\limits_{i=1}^{l}\xi_i^k \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}\cdot\mathbf{x}^{(i)}+b\right) \geq 1-\xi_i \\ & \xi_i \geq 0, \end{array} \tag{7}$$

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 $b = -\theta$. 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. [2].

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 (7) 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

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

Introducing non-negative Lagrangian multipliers $\alpha_i,\ \mu_i \geq 0,\ i = 1, \ldots, l$, we can rewrite the optimization problem (8), 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,

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\,\alpha_j\,y^{(i)}y^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)}. \tag{9}$$

Equation (9) 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

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

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. [2].

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 $\boldsymbol{\phi} : \mathbf{R}^d \rightarrow \mathcal{H}$. Substituting the mapped data into the optimization problem leads to the dual Lagrangian

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

By means of kernel functions $K : \mathbf{R}^d \times \mathbf{R}^d \rightarrow \mathbf{R}$, 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 $\boldsymbol{\phi}$. Furthermore it is always possible to achieve linear separability in the feature space if we use i.e. radial basis function kernels.

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

In order to show the relationship between perceptron and support vector learning, note that for $k = 1$ the ξ_i in (8) correspond to $-y^{(i)} \bar{\mathbf{w}} \cdot \bar{\mathbf{x}}^{(i)} + \epsilon$ of the criterion function I_ϵ in (4) for patterns misclassified by $H_{y^{(i)}}$. Thus, in the limit $C \rightarrow \infty$, both methods are equivalent for $\epsilon = 1$.

6 Experiments

6.1 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 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 $c_{-1}(x_1, x_2) = 2\frac{1}{1+e^{x_1}}$. I.e. for $x_1 > 0$ the $+1$-examples have larger misclassification costs, whereas for $x_1 < 0$ the -1-examples have larger costs.

The dataset together with the resulting hyperplane for $\epsilon = 0.1$ is depicted in fig. 2 (left, bold line). Other ϵ-values produced similar results. Without costs, a line close to the x_1-axis was produced (fig. 2, left, dashed line). With 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

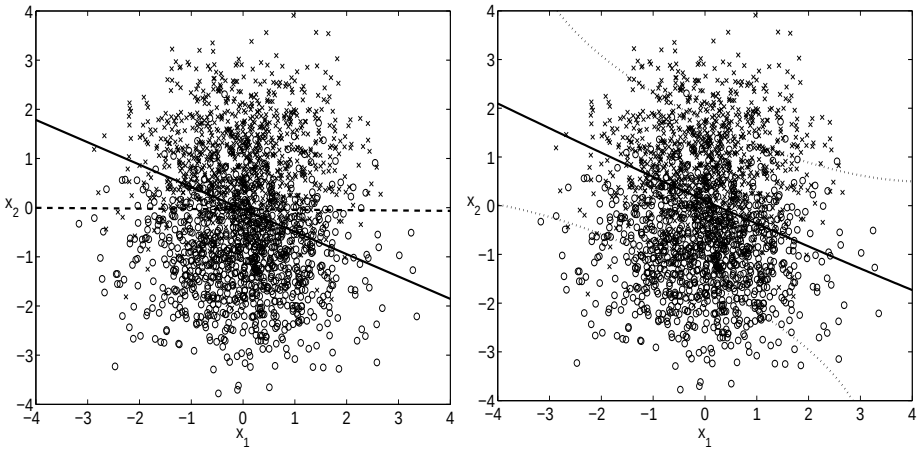

Fig. 2. Results for the non-seperable case. The two classes $+1$ and -1 are visualized by crosses and circles respectively. Hyperplanes (bold) generated by DIPOL (left), and the class boundary for the extended SVM (right). The DIPOL solution for the cost-free case (dashed) and the margin of the SVM (dotted) are shown additionally.

(not displayed in fig. 2). Analogous results are achieved by the extended SVM (fig. 2, right).

Our selection of the individual cost functions caused a *rotation* of the class boundary, see fig. 2. This effect cannot be reached using cost matrices alone. I.e. our approach is a genuine extension of previous approaches for including costs, which rely on class dependent costs or cost matrices.

6.2 The multi-modal case

To test an augmented version of DIPOL that is capable of using individual costs for learning, we have created the artificial dataset that is shown in fig. 3. 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.

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. 3. It is clear that, for reasons of symmetry, the separating hyperplanes that would be generated *without* individual costs must coincide with one of the bisecting lines. It is obvious in fig. 3, 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

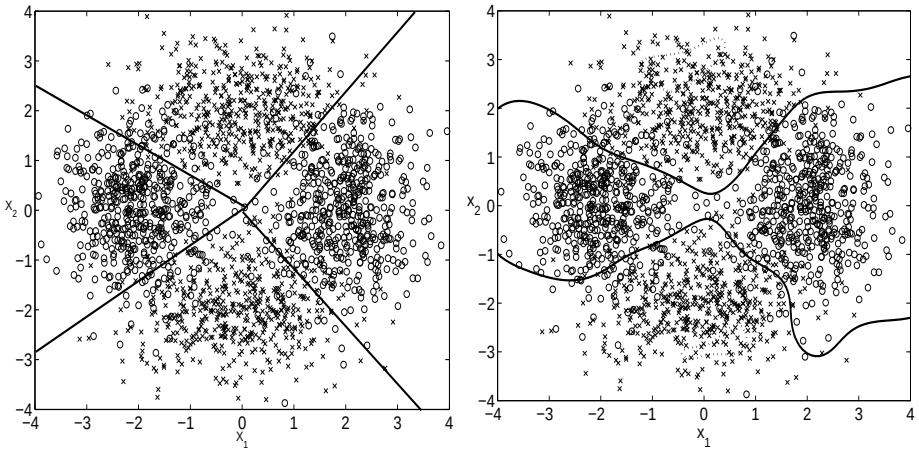

Fig. 3. Results for the multi-modal non-seperable case. Bold lines visualize the learned hyperplanes by DIPOL (left) and the extended SVM (right).

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.

6.3 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 ([10], 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 T using the 10%-test sets. When using the individual costs, the estimated mean predictive cost was 3.67.

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.

I.e. 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.

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 [10], though DIPOL and SVM perform comparably well.

7 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 costs-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 generalisation 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. Compared to the SVM, the gradient descent described in sect. 3.2 is very slow in general. The original version of DIPOL described in [10, 12, 15] incorporates

a much faster modified learning procedure (e.g. [5]) that leads to results similar to those presented in this article.

References

1. F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Math. Soc. Series of Monographs and Advanced Texts. John Wiley & Sons, 1983.
2. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods)*. Cambridge University Press, 2000.
3. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
4. 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.
5. 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.
6. 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.
7. 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.
8. Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1-3):191–202, 2002.
9. Dragos D. Margineantu and Thomas 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.
10. D. Michie, D. H. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Series in Artificial Intelligence. Ellis Horwood, 1994.
11. A. Nedic and D.P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, pages 109–138, 2001.
12. 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.
13. S. Unger and F. Wysotzki. *Lernfähige Klassifizierungssysteme (Classifier Systems that are able to Learn)*. Akademie-Verlag, Berlin, 1981.
14. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
15. 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.
16. 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.