Perceptron Based Learning with Example Dependent and Noisy Costs

Peter Geibel GEIBEL@CS.TU-BERLIN.DE
Fritz Wysotzki WYSOTZKI@CS.TU-BERLIN.DE
TU Berlin, Fak. IV, ISTI, AI Group, Sekr. FR5-8, Franklinstr. 28/29, D-10587 Berlin, Germany

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). 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 (Margineantu & Dietterich, 2000; Elkan, 2001; Kukar & Kononenko, 1998; Zadrozny & Elkan, 2001; Chan & Stolfo, 1998). 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. (Margineantu & Dietterich, 2000; Elkan, 2001)). 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. (Lenarcik & Piasta, 1998; Zadrozny & Elkan, 2001)). 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 of 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 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 been 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, 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 algo-

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

rithm 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 (Michie et al., 1994; Schulmeister & Wysotzki, 1997; Wysotzki et al., 1997) that brings together the high classification accuracy of neural networks and the interpretability gained from using simple neural models (threshold units). In addition we provide an approach for including example-dependent costs into an arbitrary learning algorithm by using modified example distributions.

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. Experiments on three artificial data sets, and on a credit data set can be found in section 5. In section 6, we discuss the inclusion of costs by resampling the dataset. The conclusions are 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 $X_y = \{\mathbf{x} \,|\, r(\mathbf{x}) = y\}$ be the region where class y is decided. According to Vapnik (1995) the risk of r with respect to the density function p of $(\mathbf{x}, y)$ is given with $p(\mathbf{x}, y) = p(\mathbf{x}|y)P(y)$ as

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

$P(y)$ 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

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

holds. This rule is called the Bayes classifier (e.g. (Duda & Hart, 1973)). 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, and 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. In this case 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 for 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.

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

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.

[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 (Devroye et al., 1996). In the cost-free case, DIPOL performed better than e.g. Backpropagation on several classification problems, see (Michie et al., 1994; Wysotzki et al., 1997).

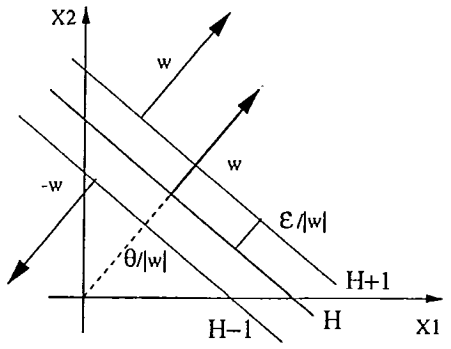

Figure 1. Geometrical interpretation of the margins, 2-dimensional case

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 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. (Duda & Hart, 1973)) 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. (Duda & Hart, 1973)).

Because in many practical cases as in 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 are either non-separable at all (i.e. overlapping), or they are separable but not linearly.

3.1. The Criterion Function

In the following, we will present the approach of Unger and Wysotzki for the linearly non-separable case (Unger & Wysotzki, 1981) extended to the usage of individual costs. Other perceptron algorithms for the linearly non-separable case are discussed in (Yang et al., 2000) and (Duda & Hart, 1973).

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**

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

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 (Unger & Wysotzki, 1981; Duda & Hart, 1973).

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 composition of convex functions. From the convexity of I_ϵ it follows that there exists a unique minimum value.

3.2. The Influence of ϵ

In the case of linearly separable classes, i.e. when the classes have a gap > 0, the criterion function I_ϵ has a minimum of zero. If we set $\epsilon = 0$ in I_ϵ, then the minimization problem for the criterion function has the trivial solution $\bar{\mathbf{w}} = 0$, giving $I_\epsilon(0) = 0$ even if the classes are linearly non-separable. On order to prevent a weight optimization algorithm from converging to $\bar{\mathbf{w}} = 0$, one must choose some $\epsilon > 0$ in order to exclude the zero vector as a trivial solution.

Let

$$T(\bar{\mathbf{w}}) = \frac{1}{l}\Big[\sum_{(\mathbf{x},c)\in S_{+1}} c \cdot \sigma(-\bar{\mathbf{w}} \cdot \bar{\mathbf{x}}) + \sum_{(\mathbf{x},c)\in S_{-1}} c \cdot \sigma(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}})\Big] \quad (5)$$

be the *mean misclassification costs* (empirical risk) caused by the hyperplane $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}} = 0$. T is an estimate of R in (1). Because T is not suited for gradient descent, the criterion function I_ϵ is considered. The following proposition holds:

Proposition 3.1 *Let $\epsilon_1, \epsilon_2 > 0$. If w^* minimizes I_{ϵ_1}, then $\frac{\epsilon_2}{\epsilon_1}\bar{\mathbf{w}}^*$ minimizes I_{ϵ_2} and $T(\bar{\mathbf{w}}^*) = T(\frac{\epsilon_2}{\epsilon_1}\bar{\mathbf{w}}^*)$.*

To prove the proposition, we rewrite (4) to $I_\epsilon(\bar{\mathbf{w}}) = \frac{1}{l}[D_\epsilon(\bar{\mathbf{w}}) + \epsilon N_\epsilon(\bar{\mathbf{w}})]$ with $N_\epsilon(\bar{\mathbf{w}}) = \sum_{(\mathbf{x},c)\in S_{+1}} c \cdot \sigma(-\bar{\mathbf{w}} \cdot \bar{\mathbf{x}} + \epsilon) + \sum_{(\mathbf{x},c)\in S_{-1}} c \cdot \sigma(\bar{\mathbf{w}} \cdot \bar{\mathbf{x}} + \epsilon)$. $D_\epsilon(\bar{\mathbf{w}})$ is the sum of the (non-normalized) distances of the misclassified objects to the hyperplane $\bar{\mathbf{w}} \cdot \bar{\mathbf{x}} = 0$. It can be shown that $I_{\epsilon_2}(\frac{\epsilon_2}{\epsilon_1}\bar{\mathbf{w}}) = \frac{1}{N}[\frac{\epsilon_2}{\epsilon_1} D_{\epsilon_1}(\bar{\mathbf{w}}) + \frac{\epsilon_2}{\epsilon_1}(\epsilon_1 N_{\epsilon_1}(\bar{\mathbf{w}}))] = \frac{\epsilon_2}{\epsilon_1} I_{\epsilon_1}(\bar{\mathbf{w}})$ holds. Details of the proof can be found in (Geibel & Wysotzki, 2002).

The essence of the proposition is the fact that with respect to the computed hyperplane, and the mean misclassifications costs T, the choice of the parameter ϵ does not matter, which is good news. It should be noted that $\bar{\mathbf{w}}^*$ and $\frac{\epsilon_2}{\epsilon_1}\bar{\mathbf{w}}^*$ define the same hyperplane.

3.3. The Learning Rule

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

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

To handle the points, in which I_ϵ cannot be differentiated, in (Unger & Wysotzki, 1981) the gradient in (6) is considered as 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 (Unger & Wysotzki, 1981; Clarke, 1983; Nedic & Bertsekas, 2001).

By 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:

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

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 usual conditions for stochastic approximation are imposed: $\lim_{t\to\infty} \gamma_t = 0$, $\sum_{t=0}^{\infty} \gamma_t = \infty$, and $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. The convergence to an optimum in the separable and the non-separable case follows from the results in (Nedic & Bertsekas, 2001).

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

One problem of the derived perceptron algorithm lies in the fact that in general it can be applied successfully only in the case of two classes with unimodal distributions – a counterexample is the XOR problem. In order to deal with multi-class/multi-modal problems, we have extended the learning system DIPOL (Michie et al., 1994; Schulmeister & Wysotzki, 1997; Wysotzki et al., 1997) in order to handle example dependent costs.

Due to the comparison of several algorithms from the field of machine learning, statistical classification and neural networks (excluding SVMs, see e.g. (Vapnik, 1995)), that was performed during the STATLOG project (Michie et al., 1994), DIPOL turned out to be one of the most successful learning algorithms – it performed best on average on all datasets (see (Wysotzki et al., 1997) for more details).

DIPOL can be seen as an extension of the perceptron approach to multiple classes and multi-modal distributions. 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 (Schulmeister & Wysotzki, 1997; Duda & Hart, 1973) for every class. The cluster numbers per class have to be provided by the user or must be adapted by a parameter optimization algorithm.

After the (optional) clustering of the classes, a separating hyperplane is constructed for each pair of classes or

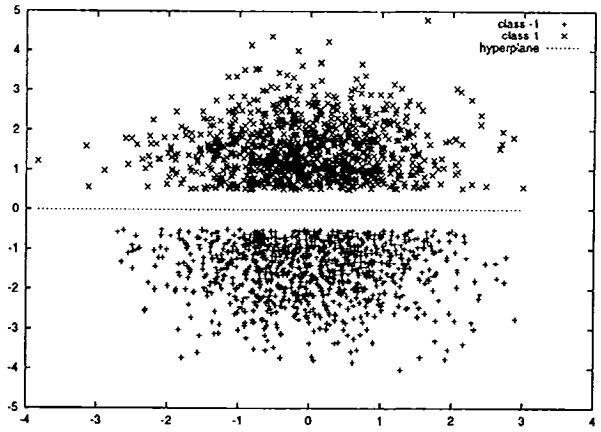

Figure 2. The linearly separable case: resulting hyperplane for $\epsilon = 10$. Horizontal axis: attribute x_1. Vertical axis: attribute x_2.

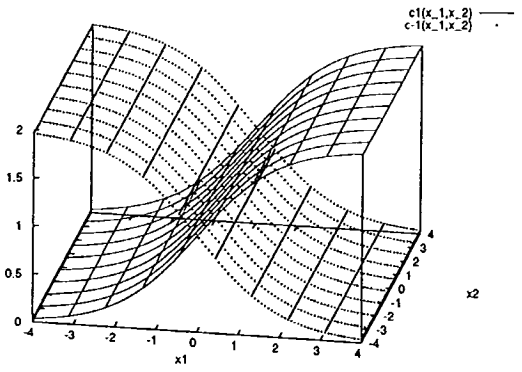

Figure 3. Cost functions in the separable and non-separable case

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. 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 the 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 (Michie et al., 1994; Schulmeister & Wysotzki, 1997; Wysotzki et al., 1997), 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. 3.3 for the experiments.

5. Experiments

In our first experiment, we used the perceptron algorithm for the linearly non-separable case (sect. 3.3) with individual costs. 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 means $(0.0, \pm 1.0)^T$. The covariance matrix for both classes is the unit matrix. The distribution of class $+1$ was modified in order to prevent the generation of examples with an x_2-value smaller than 0.5. Accordingly, for class -1, no examples with an x_2-value larger than -0.5 were generated. The resulting gap between the classes has a width of 1. The dataset and a resulting hyperplane (i.e. line) is depicted in fig. 2.

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 $c_{-1}(x_1, x_2) = 2\frac{1}{1+e^{x_1}}$. The cost functions are shown in fig. 3.

The algorithm stops, when it has found a hyperplane that perfectly separates the classes *and* when the margin hyperplanes fit into the gap. This means that the margin hyperplanes are required to separate the classes, too. We have found empirically, that choosing a larger value of ϵ causes the solution hyperplane to maximize the margin, similar to support vector machines (see fig. 2). This effect cannot be explained from the criterion function I_ϵ alone, but has to be explained by the learning algorithm, for details see (Geibel & Wysotzki, 2002).

5.1. The Non-Separable Case

For our experiments with a non-separable dataset, we used the same class dependent distributions as in the separable case, but no gap. The cost functions were identical to the cost functions used for the separable case, see fig. 3.

The dataset together with the resulting hyperplane for $\epsilon = 0.1$ is depicted in fig. 4 (dashed line). Other ϵ-values produced similar results which is due to prop. 3.1. Without costs, a line close to the x_1-axis was produced (fig. 4, drawn line). With class dependent misclassification costs, lines are produced that are almost parallel to the x_1-axis and that are shifted

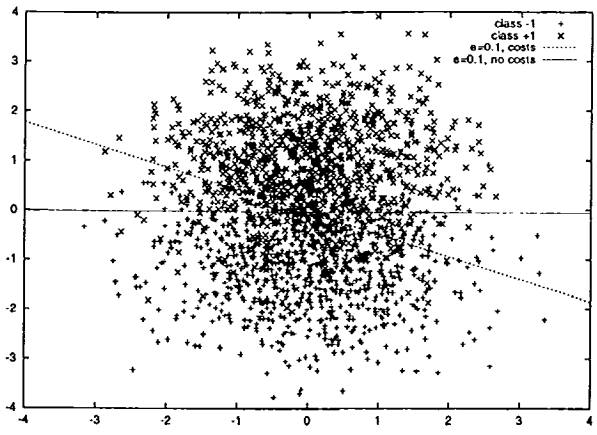

Figure 4. Results for the non-separable case. Horizontal axis: attribute x_1. Vertical axis: attribute x_2.

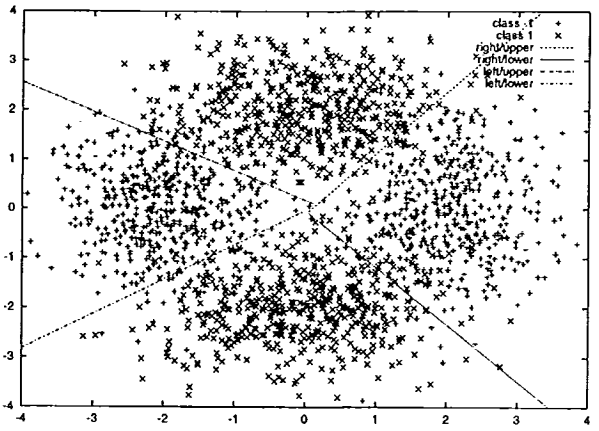

Figure 5. Training set and hyperplanes generated by DIPOL for the multimodal non-separable case. Horizontal axis: attribute x_1. Vertical axis: attribute x_2.

into the class region of the less dangerous class (not displayed in fig. 4). In contrast, our selection of the individual cost functions caused a *rotation* of the line, see fig. 4. 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.*

5.2. DIPOL

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. 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.

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. 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. 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.

5.3. 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 ((Michie et al., 1994), 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 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.

This means that in the case of the German Credit Dataset we achieved a 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.

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

$$B_y = \int_{\mathbf{R}^d} c_y(x)p(x|y)dx\,. \qquad (8)$$

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

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

I.e. we now consider the new class conditional densities

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

and new priors

$$P'(y) = P(y)\frac{B_y}{B_{+1}P(+1) + B_{-1}P(-1)}\,.$$

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 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.

It holds for the compound density

$$p'(\mathbf{x}, y) = p'(\mathbf{x}|y)P'(y) = \frac{c_y(\mathbf{x})}{b}p(\mathbf{x}, y). \qquad (9)$$

Therefore, in each of the $\lfloor Nl \rfloor$ 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 $T(r) \approx bT'(r)$, where T is evaluated with respect to the given sample, and $T'(r)$ is evaluated with respect to the generated cost-free sample. I.e. 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 described e.g. in (Chan & Stolfo, 1998), and to the extension of the METACOST-approach for example dependent costs (Zadrozny & Elkan, 2001). We will compare their performances in future experiments.

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 cost-sensitive learning rule for linearly non-separable classes 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.

Independently from the perceptron framework, we have discussed the inclusion of example dependent costs into a cost-insensitive learning algorithm by resampling the original examples in the training set according to their costs. This way example dependent costs can be incorporated into an arbitrary cost-insensitive learning algorithm.

Future work will include the evaluation of the sampling method described in sect. 6, and a comparison to regression based approaches for predicting the cost directly.

References

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

Clarke, F. H. (1983). *Optimization and nonsmooth analysis.* Canadian Math. Soc. Series of Monographs and Advanced Texts. John Wiley & Sons.

Devroye, L., Györfi, L., & Gabor, L. (1996). *A probabilistic theory of pattern recognition.* Springer-Verlag.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Elkan, C. (2001). The foundations of Cost-Sensitive learning. *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)* (pp. 973–978). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Geibel, P., & Wysotzki, F. (2002). *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).

Kukar, M., & Kononenko, I. (1998). Cost-sensitive learning with neural networks. *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)* (pp. 445–449). Chichester: John Wiley & Sons.

Lenarcik, A., & Piasta, Z. (1998). Rough classifiers sensitive to costs varying from object to object. *Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing (RSCTC-98)* (pp. 222–230). Berlin: Springer.

Margineantu, D. D., & Dietterich, T. G. (2000). Bootstrap methods for the cost-sensitive evaluation of classifiers. *Proc. 17th International Conf. on Machine Learning* (pp. 583–590). Morgan Kaufmann, San Francisco, CA.

Michie, D., Spiegelhalter, D. H., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification.* Series in Artificial Intelligence. Ellis Horwood.

Nedic, A., & Bertsekas, D. (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 109–138.

Schulmeister, B., & Wysotzki, F. (1997). Dipol - a hybrid piecewise linear classifier. In R. Nakeiazadeh and C. C. Taylor (Eds.), *Machine learning and statistics: The interface*, 133–151. Wiley.

Unger, S., & Wysotzki, F. (1981). *Lernfähige Klassifizierungssysteme (Classifier Systems that are able to Learn).* Berlin: Akademie-Verlag.

Vapnik, V. N. (1995). *The nature of statistical learning theory.* New York: Springer.

Wysotzki, F., Müller, W., & Schulmeister, B. (1997). Automatic construction of decision trees and neural nets for classification using statistical considerations. In G. DellaRiccia, H.-J. Lenz and R. Kruse (Eds.), *Learning, networks and statistics*, no. 382 in CISM Courses and Lectures. Springer.

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

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)* (pp. 204–214). New York: ACM Press.