ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-841

Soft Nearest Convex Hull Classifier

Georgi Nalbantov [1] and **Evgueni Smirnov** [1]

Abstract. Consider the classification task of assigning a test instance to one of two or more possible classes. An intuitive way to proceed is to assign the instance to that class, to which the distance is minimal. If one considers the distance to the convex hull of a class as a distance measure, then the resulting classification method is the Nearest Convex Hull (NCH) classifier. There are two key issues with this method per se that severely restrict its applicability, which we solve in this paper: first, how to handle class overlap, and second, how to provide (nonlinear) solutions with better generalization ability. The first problem is handled via using so-called kernel functions and slack variables. The second problem is dealt with using a penalization term that suppresses too complex solutions. We call the resulting method the soft-NCH classifier. In spirit and computationally the method is close to the popular Support Vector Machine (SVM) classifier and can be viewed as an instance-based large-margin classification technique. Advantages of the soft-NCH classifier include its robustness to outliers, good generalization ability and naturally easy handling of multi-class problems. We compare the performance of soft-NCH against state-of-art techniques and report promising results.

1 INTRODUCTION

Both instance-based classifiers and large-margin classifiers have recently gained considerable popularity. This paper puts forward a technique that shares characteristics of these two areas, called the Soft Nearest Convex Hull (soft-NCH) classifier. As the name suggests, the classification rule of the pure Nearest Convex Hull classifier is based on the idea to assign a test instance, or point, $\mathbf{x}$ to that class, which convex hull is closest to $\mathbf{x}$. The implementation of this rather straightforward rule requires solving an optimization task to find the distance from a point to the convex hull of several other points. Standard algorithms for this task have long existed (see, e.g., [1], [4]). One reason why such algorithms have however not been employed for classification purposes is the problem of assigning a label to $\mathbf{x}$ in case it lies inside the convex hulls of two or more classes. This convex-hull overlap results in zero distances from $\mathbf{x}$ to these convex hulls and that is why the classification label of $\mathbf{x}$ cannot be unequivocally determined. Taking this problem as the point of departure, the idea behind the soft-NCH classifier is to handle class overlap by means of using so-called slack distances to convex hulls and/or mapping of the training points from the original input space to a sufficiently large higher-dimensional space via so-called kernel functions. Both of these tools have been borrowed from the popular Support Vector Machine (SVM) classifier. It is the introduction of slack distances that gives rise to the soft version of the NCH classifier.

[1] Department of Knowledge Engineering, Faculty of Humanities and Sciences, Maastricht University, Netherlands, email: {g.nalbantov}{smirnov}@maastrichtuniversity.nl

In order to classify, the NCH classifier computes the distance from a test point to a convex hull. In this process there is no need to compute explicitly the convex hull of any class. Instead, given that the test point lies outside the convex hull of a given class, the approach taken is to compute the maximal distance from the test point to a hyperplane that separates this point from the convex hull. On the surface, it might therefore seem that the soft-NCH classifier actually comprises two SVM classifiers (if there are two classes), applied separately to each class and a given fixed test point. This is not the case however due to one crucial difference : the soft-NCH classifier uses slack variables for the training instances, but not for the test instance $\mathbf{x}$. In this way it is ensured that it is the slack distance from a fixed test point to a convex hull that is measured, and not the slack distance between two classes, one of which consists of the test point only, which would be the SVM-like solution. We note that the decision surface produced by the NCH classifier is in general nonlinear in the original data space, which makes it more "flexible" than the SVM classifier. This arguably results in a more complex decision surface, but at the same time a better fit to the data, *ceteris paribus*. The interplay between these two effects determines which one of these classifiers will have a better generalization ability on a particular classification task.

The NCH classifier can also be thought of as a departure from the 1-Nearest Neighbor (1NN) rule. The point of departure here is the idea to "fill in" the missing points from a given class by (all) points from its convex hull, which helps circumvent the curse of dimensionality problem from which the 1NN rule may suffer ([3], [10]). This makes the NCH classifier less flexible than the 1NN rule, in the sense that it produces more regular decision surfaces. Thus, at least in terms of flexibility, the NCH classifier can be thought of providing an intermediate solution between SVMs and 1NN. One crucial similarity between the 1NN and NCH methods is that in both cases the decision surface between the classes is not explicitly computed. Instead, for any test point the methods output the predicted class that is implied by a decision surface. Since the NCH classifier implements the idea to find the distance to the closest point of a given convex hull, and it has to do so for each hypothetical test instance, the NCH approach can be categorized as being an instance-based large-margin classification approach.

The soft-NCH classifier has a number of advantageous qualities, such as uniqueness of (optimization) solution, avoidance of overfitting, good generalization ability, the possibility to handle nonlinearities with ease via so-called kernel functions and/or slack variables, and the robustness to outliers. The uniqueness-of-solution property stems from the formulation of the optimization problem being solved. Overfitting, or fitting too well to the training data, is overcome via the introduction of a *regularization* term that suppresses too complex potential fits. Nonlinear solutions that are more complex than those already produced in the original data space are produced using kernel functions. The kernel functions provide for the

possibility to map the training data into a higher-dimensional space and to compute distances to convex hulls in this space computationally efficiently, just like in the case of SVMs. The potential overfitting effect in this higher-dimensional space due to the increased complexity/nonlinearily of the (implicit) soft-NCH decision function is counter-weighted by the ability to apply regularization. The robustness-to-outliers property originates from the linear (and not for example quadratic) penalization of the training errors induced in case of class overlap. Another advantage of the soft-NCH as an instance-based method, particularly over SVMs, is its computational efficiency and higher training speed especially for multi-class classification tasks. This occurs because only same-class objects are considered in the estimation of a (soft) distance to a convex hull, and not the entire data set. Furthermore, no additional question arises regarding what type of multi-class approach for combining binary classifiers to use, such as one-against-one or one-against-all. A general disadvantage of the NCH classifier is the requirement to solve k optimization problems to find the predicted class-label of a test point, where k is the number of classes. This property is however tied to the instance-base nature of the method. Nevertheless, each of the k optimization problems, one per class, is significantly easier to solve than a single overall optimization problem that involves all training data points.

We note that one can find parallels not only with SVMs and 1NN, but with other techniques as well. For example, the jump from Linear Discriminant Analysis (LDA) to Quadratic Discriminant Analysis (QDA) resembles in many ways the transition from SVMs to the NCH classifier. More generally, various links between the areas of instance-based methods and large-margin methods have extensively been exploited elsewhere in the literature. Examples include a large-margin nearest neighbor classifier proposed by [8], a kernel nearest-neighbor algorithm of [12], a k-local hyperplane and convex distance nearest neighbor algorithms by [19], and others. We confer also to [20] for a more general discussion on distance-based classification.

The paper is organized as follows. First we provide some intuition behind the soft-NCH classifier and a formal definition of it. Next, we discuss the technical aspects of the classifier – derivation and implementation. Finally, we show some experimental results on simulated and real data sets and then conclude.

2 NEAREST CONVEX HULL CLASSIFIER: DEFINITION AND MOTIVATION

At the outset, consider a binary data set of positive and negative objects $\langle I^+, I^- \rangle$ from $\mathbb{R}^n$. The classification task is to separate the two classes of objects with a decision surface that can generalize well to test data sets. This task is formalized as finding a (target) function $f : \mathbb{R}^n \rightarrow \{-1, 1\}$ such that f will classify correctly unseen observations. The extension to the multi-class case is straightforward. The decision rule of the NCH classifier is defined as follows: *a test point* $\mathbf{x} \in \mathbb{R}^n$ *should be assigned to that class, which convex hull is closest to* $\mathbf{x}$. The convex hull of a class is defined as the smallest convex set that contains the training instances from that class.

Below we split the exposition in two parts. First we consider the case with no class overlap, called the (linearly) separable case, and then we discuss the nonseparable case.

2.1 NCH Classification in the Separable Case

Let us first consider the so-called separable case where two classes are separable by a hyperplane without any misclassification error.

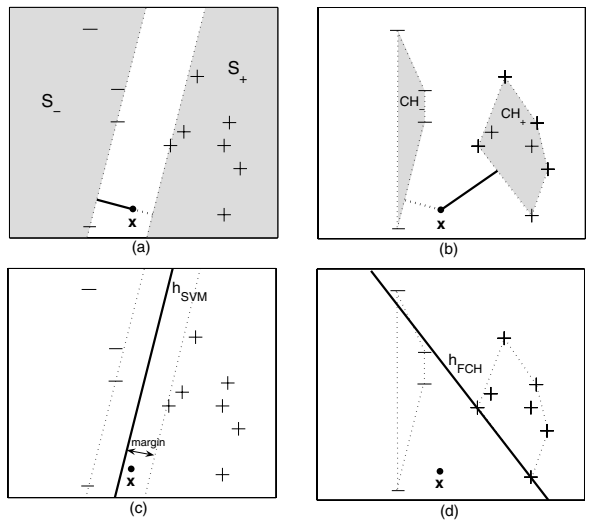

Figure 1. Classification of a test point $\mathbf{x}$ with SVM in Panels (a) and (c), and NCH in Panels (b) and (d) on a binary data set. In Panel (a), the white band has the largest possible width, which is equal to twice the margin, shown in Panel (c). The points to the left and to the right of the band form shaded sets S_- and S_+, respectively. Test point $\mathbf{x}$ receives label $+1$ since it is farther from S_- than S_+. In Panel (b) point $\mathbf{x}$ is classified as -1 since it is farther from the convex hull of the positive points, CH_+, than from the convex hull of the negative points, CH_-.

Structurally, the NCH classifier can be viewed as an intermediate method between the popular SVM classifier ([18]) and the 1-Nearest Neighbor classifier. Therefore, we first draw an intuitive comparison between NCH and SVM, and then between NCH and 1NN. See Figure 1 for an illustrative binary classification example. Panels (a) and (c) refer to SVM classification, and Panels (b) and (d) refer to NCH classification. To make the comparison between SVM and NCH classification more appealing, we describe SVM classification from an instance-based point of view. That is, we focus on the classification process of a particular test point. As it turns out, in both the SVM and NCH cases a test point is classified according to the proximity to particular sets of points. In the NCH case these sets are the convex hulls of the positive and negative classes, denoted by CH_+ and CH_- respectively, and in the SVM case these sets are areas to the left and to the right of the so-called *margin* band, denoted by S_- and S_+ (defined below).

In SVM classification, the target function is a hyperplane of the form $\mathbf{w}'\mathbf{x} + b = 0$, where $\mathbf{w}$ is a vector of coefficients and b is the intercept, and $\mathbf{w}'$ is the transpose of (column vector) $\mathbf{w}$. The SVM hyperplane $\mathbf{w}^{*\prime}\mathbf{x} + b^* = 0$ (denoted as h_{SVM} in Figure 1) is the hyperplane that separates the classes with the widest margin, where a margin is defined as the distance between a (separating) hyperplane and the closest point to it from the training data set. In terms of Figure 1, Panel (a), the width of the white band is equal to twice the margin. The margin itself is shown in Panel (c). The closest point to h_{SVM} is defined to lie on the hyperplane $\mathbf{w}^{*\prime}\mathbf{x} + b^* = 1$ if this point is positively labeled, or on the hyperplane $\mathbf{w}^{*\prime}\mathbf{x} + b^* = -1$ if this closest point is negatively labeled. For all points $\mathbf{x}$ that do no lie inside the margin it holds that either $\mathbf{w}^{*\prime}\mathbf{x}+b^* \leq -1$ or $\mathbf{w}^{*\prime}\mathbf{x}+b^* \geq 1$. The former set of points is defined as S_-, and the latter set of points is defined as S_+. For any test point $\mathbf{x}$, the SVM classification rule can be formulated as follows: a test point $\mathbf{x}$ is classified as $+1$ if it is closer to set S_+ than set S_-; otherwise $\mathbf{x}$ receives label -1. It has been argued (see, e.g., [17], [6], [18]) that SVM classification

searches for a balance between empirical error (or, the goodness-of-fit over the training data) and complexity of the function class that fits the data. Quantitatively, the complexity can be (inversely) related to the distance between sets S_+ and S_-, that is, twice the margin: the larger the margin, the lower the associated complexity and the simpler the classification model. Note that in the separable case at hand in Figure 1, the empirical training error of h_{SVM} is zero since it fits the data perfectly. In principle, the higher the function-class complexity, the lower the empirical error. The tradeoff between empirical error and complexity can intuitively be approached from an instance-based viewpoint as well. In this case, complexity is imputed in the classification of each separate test instance. Thus, the larger the distance from a test object $\mathbf{x}$ to the farther one of the two sets S_+ and S_-, the lower the complexity associated with the classification of $\mathbf{x}$.

Unlike SVMs, in NCH classification the target function is implicit. A test point $\mathbf{x}$ receives classification according to its proximity to the convex hull of the positive instances, CH_+, and the convex hull of the negative instances, CH_-, which results in an overall implicit nonlinear decision surface (for illustration, see Figure 4, upper left panel). The NCH classification rule can be formulated as follows: if $\mathbf{x}$ is closer to set CH_+ than CH_-, it receives label $+1$; otherwise $\mathbf{x}$ receives label -1. Analogously to SVMs, the NCH classifier can also be considered intuitively from a fit-versus-complexity standpoint. In NCH classification one can consider the distance to the farther one of the two sets CH_+ and CH_- as a proxy for the complexity associated with the classification of test point $\mathbf{x}$: the larger the distance, the lower this complexity. Note that this distance is always as big as or bigger than the distance from $\mathbf{x}$ to the farther of sets S_+ and S_-. This property holds since the convex hull of the $+1$ (-1) points is a subset of S_+ (S_-), as can be seen in Figure 1. The decision surface produced by NCH in the original data space is in general nonlinear, implying that the method is more adaptive than the (linear) SVMs. From generalization ability viewpoint, the better empirical fit is however counterbalanced by the increased flexibility of the resulting NCH decision surface. As a consequence, it is not clear a priori whether NCH or SVM will strike a better balance between fit and complexity.

The NCH classifier could also be compared and contrasted with the 1-Nearest Neighbor rule in terms of the fit-versus-complexity interplay. The 1NN suffers from the curse of dimensionality and may exhibit severe bias in high dimensional problems ([10]), due to the fact that the training data set is generally too small to sufficiently populate the input space. The 1NN rule is highly adaptive though, as it is able to produce an extremely irregular decision surface that separates the classes of any data set without an error (unless it contains points from different classes with the same input values). One may think of this high flexibility as giving rise to a rather high complexity associated with the classification of any test point. The NCH approach in a way softens the curse of dimensionality by introducing hypothetical points – the points from the convex hulls of the classes – with which to populate the input space in an artificial way. This "cure" is however accompanied by worsened flexibility in the sense of a more regular overall decision surface. The extent of interplay between these two effects would ultimately determine which method would prevail for a given task.

NCH has the property that the extent of proximity to a given class is determined without taking into consideration objects from other classes. This property contrasts with the SVM approach, where the sets S_+ and S_- are not created independently of each other. A similar parallel can be drawn between LDA and QDA methods. In LDA, one first determines the Mahalanobis distances from $\mathbf{x}$ to the centers of the classes using a common pooled covariance matrix and then classifies $\mathbf{x}$ accordingly. In QDA, one uses a separate covariance matrix for each class. Analogically, the NCH classifier first determines the Euclidean distance from $\mathbf{x}$ to the convex hulls of each of the classes and then classifies $\mathbf{x}$ accordingly. In sum, loosely speaking one may think of the shift from SVM to NCH as resembling the shift from LDA to QDA.

2.2 NCH Classification in the Nonseparable Case

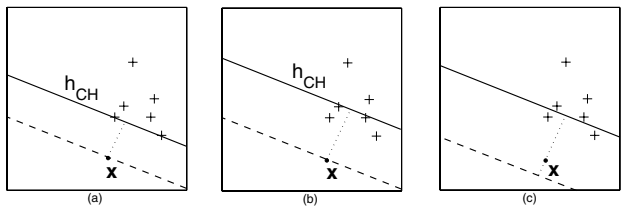

Figure 2. Classification of a test point $\mathbf{x}$ with (hard) NCH in Panel (a), soft NCH in Panel (b), and soft SVM in Panel (c). In Panel (b), the hyperplane from Panel (a) is "pushed" further away from $\mathbf{x}$ and inside the convex hull of the positive class, resulting in a soft distance from $\mathbf{x}$ to this convex hull. In Panel (c), the distance between the two parallel hyperplanes represents the so-called (twice) SVM margin. In this case the slack is applied to both $\mathbf{x}$ and the points from the positive class, unlike the soft NCH, where no slack is applied to $\mathbf{x}$.

In the so-called nonseparable case the convex hulls of the two classes intersect each other. As a result, there exist test points that belong to both convex hulls simultaneously and cannot therefore be assigned to a particular class. To deal with this situation, we introduce so-called slack distances to convex hulls, in a way not dissimilar to the soft SVM approach, and call the resulting approach the soft-NCH classifier. Consider Figure 2, which shows the solutions for the separable NCH case, the nonseparable NCH case, and the nonseparable SVM case. The difference between Panels (a) and (b) of the figure is the increased, or soft, distance in Panel (b) to the convex hull of the class in question (here, the positive class). Figuratively speaking, the hyperplane h_{CH} in Panel (a) is pushed further away from the test point and inside the convex hull of the positive class in Panel (b). This increased distance gives automatically rise to points from the positive class appearing on the wrong side of the hyperplane h_{CH}. We consider these points as "errors" and associate a linear loss with each of them that is proportional to the distance between any such point and h_{CH}. The introduction of a slack distance allows to find a positive (soft) distance to a given convex hull even if the test point lies inside the convex hull. This step is crucial as it allows to deal with class overlap: even when a test instance lies inside two convex hulls, implying that the distance to both hulls is equal (to zero), unique class assignment is possible as the soft distances to the convex hulls will in general be different. Notice that the hyperplane that passes through the test point in Panel (a) preserves its position also in Panel (b). In this way it is ensured that it is the slack distance to the convex hull that is being measured, and not some other distorted distance. An example of such distorted distance is given in Panel (c) of the figure, which shows the soft SVM solution to the two-class problem where one of the classes is the positive class, and the other class consists of the test point only. As the standard SVM classifier does not explicitly differentiate between the classes, both points from the positive class *and* the test point are associated with a slack, which

leads to a solution that does not represent the slack distance between a fixed test point and a convex hull, but the slack distance between two classes, one of which consists of the test point only.

3 ESTIMATION

3.1 Separable case

Consider a data set of l objects from k different groups, or classes. Let l_k denote the number of objects in the k^{th} class. According to NCH, a test point $\mathbf{x}$ is assigned to that class, to which the distance is minimal. In the separable case, the distance to a class is defined as the distance to the convex hull of the objects from that class. The algorithm for classifying $\mathbf{x}$ can be described as follows (see Figure (3)): first, compute the distance from $\mathbf{x}$ to the convex hull of each of the k classes; second, assign to $\mathbf{x}$ the label of the closest class. Formally, to find the distance from a test point $\mathbf{x}$ to the convex hull of the nearest class, one has to solve a quadratic optimization problem, as for instance described in [4], p.398. However, in order to be able to employ kernels easily, we formulate a quadratic optimization problem that has to be solved for each class k, which is similar to the SVM optimization problem:

$$\begin{aligned} \min_{\mathbf{w}_k, b_k} \quad & \frac{1}{2}\mathbf{w}_k'\mathbf{w}_k \\ \text{s.t.} \quad & \mathbf{w}_k'\mathbf{x}_i + b_k \geq 0,\ i = 1, 2, \ldots, l_k \\ & -(\mathbf{w}_k'\mathbf{x} + b_k) = 1 \end{aligned} \tag{1}$$

The distance between hyperplane $\mathbf{w}_k'\mathbf{x} + b_k = 0$ and $\mathbf{x}$ is defined

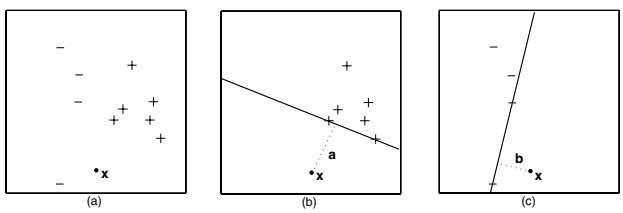

Figure 3. Classification of a test point $\mathbf{x}$ with NCH on the binary data set in Panel (a) in two steps. At stage one (Panel (b)), a test point $\mathbf{x}$ is added to a data set that contains only the positive class, and the distance **a** from $\mathbf{x}$ to the convex hull of this class is computed. At stage two (Panel (c)), $\mathbf{x}$ is added to a data set that contains only the negative class, and the distance **b** from $\mathbf{x}$ to the convex hull of this class is computed. If $\mathbf{a} > \mathbf{b}$ ($\mathbf{a} < \mathbf{b}$), then $\mathbf{x}$ is assigned to the negative (positive) class.

as $1/\sqrt{\mathbf{w}_k'\mathbf{w}_k}$ by the last constraint of (1). This distance is maximal when $\frac{1}{2}\mathbf{w}_k'\mathbf{w}_k$ is minimal. At the optimum, it represents the distance from $\mathbf{x}$ to the convex hull of class k. The role of the first l_k inequality constraints is to ensure that the hyperplane classifies correctly each point that belongs to class k. Effectively, for each of the k classes, the l_k same-class objects are assigned label 1, and the test point is assigned label -1. Eventually, $\mathbf{x}$ is assigned to that class to which the distance is minimal, that is, which corresponding value for the objective function in (1) is maximal.

3.2 Nonseparable case

The solution for $\mathbf{w}_k$ of optimization problem (1) differs from $\mathbf{0}$ for a given k only if the test point lies outside the convex hull of class k. Therefore, a complication arises if some of the convex hulls overlap. Then a test point could lie simultaneously in two or more convex hulls and its classification label would be undetermined. To cope with these situations, so-called slack variables can be introduced, similarly to the SVM approach, which gives rise to the soft Nearest Convex Hull classifier. Consequently, the nonseparable version of optimization problem (1) that has to be solved for each class k becomes:

$$\begin{aligned} \min_{\mathbf{w}_k, b_k, \boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}_k'\mathbf{w}_k + C\sum_{i=1}^{l_k}\xi_i \\ \text{s.t.} \quad & \mathbf{w}_k'\mathbf{x}_i + b_k \geq 0 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, \ldots, l_k \\ & -(\mathbf{w}_k'\mathbf{x} + b_k) = 1. \end{aligned} \tag{2}$$

Note that in (2) the points that are incorrectly classified are penalized linearly via the term $\sum_{i=1}^{l_k}\xi_i$. If one prefers a quadratic penalization of the classification errors, then the sum of squared errors $\sum_{i=1}^{l_k}\xi_i^2$ should be substituted for $\sum_{i=1}^{l_k}\xi_i$ in (2). One can go even further and extend the NCH algorithm in a way analogical to LS-SVM ([9]) by imposing in (2) that constraints $\mathbf{w}_k'\mathbf{x}_i + b_k \geq 0 - \xi_i$ hold as equalities, on top of substituting $\sum_{i=1}^{l_k}\xi_i^2$ for $\sum_{i=1}^{l_k}\xi_i$. A closely related approach to finding soft distances between convex hulls in case of class overlap, which uses the concept of *reduced* convex hulls, can be found in [2]. One key difference between this approach and soft NCH is analogous to the difference between soft SVM and soft NCH discussed in Section 2.2: soft NCH explicitly removes the slack variable associated with the test point, via the equality constraint in (2). It can be shown that soft NCH can be transformed into a reduced convex classifier by imposing an additional constraint $\alpha_{l_k+1} = 1$ in the dual of (2) below, (3).

Each of the k (primal) optimization problems pertaining to (2) can be expressed in dual form[2] as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \alpha_{l_k+1} - \frac{1}{2}\sum_{i,j=1}^{l_k+1}\alpha_i\alpha_j y_i y_j (\mathbf{x}_i'\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ i = 1, 2, \ldots, l_k, \\ & \sum_{i=1}^{l_k+1} y_i\alpha_i = 0, \end{aligned} \tag{3}$$

where the α's are the Lagrange multipliers associated with the respective k^{th} primal problem. Here α_{l_k+1} is the Lagrange multiplier associated with the equality constraint $-(\mathbf{w}_k'\mathbf{x} + b_k) = 1$. In each problem $y_i = 1, i = 1, 2, \ldots, l_k$ and $y_{l_k+1} = -1$. The advantage of the dual formulation (3) is that different kernels can be employed to replace the inner product $\mathbf{x}_i'\mathbf{x}_j$ in (3) in order to obtain nonlinear decision boundaries, just like in the SVM case. Three popular kernels are linear $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i'\mathbf{x}_j$, polynomial of degree d $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i'\mathbf{x}_j + 1)^d$ and the Radial Basis Function (RBF) kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2)$, where the manually-adjustable γ parameter determines the proximity between $\mathbf{x}_i$ and $\mathbf{x}_j$.

A total of k NCH optimization problems have to be solved to determine the class of any test point $\mathbf{x}$. This property provides for the fact that the NCH decision boundary is in general implicit and nonlinear, even in case the original data is not mapped into a higher-dimensional space via a kernel. Figure 4 demonstrates that this property does not hold in general for Support Vector Machines, for instance. This figure also illustrates that the NCH decision boundary appears to be less sensitive to the choice of kernel and kernel parameters than the respective SVM boundary.

[2] The derivation of the dual problem resembles the one used in SVM (see, e.g., [6]).

Table 1. Leave-one-out accuracy rates (in %) of the Nearest Convex Hull classifier as well as some standard methods on several data sets. Rbf, 2p and lin stand for Radial Basis Function, second-degree polynomial and linear kernel, respectively.

	NCH rbf	NCH 2p	NCH lin	SVM rbf	SVM 2p	SVM lin	NB	LR	LDA	QDA	MLP	kNN	DS	C4.5
Sonar	**91.4**	90.4	88.0	88.9	82.2	80.8	67.3	73.1	75.5	74.9	81.3	86.5	73.1	71.2
Voting	95.9	95.4	95.9	96.5	96.3	96.8	90.3	96.5	95.9	94.2	94.9	93.3	95.9	**97.0**
W.B.C.	**97.4**	97.1	97.3	97.0	96.9	96.9	96.0	96.1	96.0	91.4	95.0	97.0	92.4	95.3
Heart	**85.6**	82.6	84.1	**85.6**	81.1	85.6	83.0	83.7	83.7	81.5	78.9	84.4	76.3	75.2
A.C.A.	86.4	85.4	86.1	**87.4**	79.9	87.1	77.1	86.4	85.8	85.2	84.8	85.9	85.5	83.8
Hep.	85.2	84.5	84.5	**86.5**	86.5	86.5	83.2	83.9	85.8	83.9	79.4	85.8	79.4	80.0
Haberman	**76.2**	73.9	73.9	75.8	75.2	75.2	75.2	73.9	73.2	73.5	72.2	74.2	74.2	75.5
B.Cancer	74.5	75.2	75.2	**76.9**	75.9	71.3	72.7	68.5	36.7	44.8	69.6	75.2	72.0	70.3
Colic	84.2	81.3	83.7	84.5	**84.8**	84.5	75.8	79.6	68.5	67.9	78.5	81.3	81.5	83.4
CreditG	71.0	72.6	74.7	70.0	70.0	75.0	**75.1**	**75.1**	67.1	68.2	72.6	74.3	70.0	70.9

Being instance-based, the NCH classifier is characterized by high computational burden associated with the classification of test points that are not in the original data set, as compared to methods that compute a decision surface explicitly, as SVM's for instance. Therefore, tasks for which test points arrive quickly after each other will require greater offsetting computational power. On the other hand, the NCH approach may lead to a faster determination of the manually-adjustable parameters, such as C (used in SVM's as well). This is most evident if one estimates such parameters using the leave-one-out procedure (say, in a two-class problem). At each step, one of the test points is left for testing, and a model is built using the remaining points in the data set. If the date set is balanced (the number of instances per class is the same), then the NCH has to solve two quadratic optimization problems of size $N/2$, where N is the total number of instances in the training set; in contrast, SVM's have to solve one quadratic optimization problem (quite similar to those NCH solves) of size N. Current nonlinear SVM solvers scale more than linearly (in time) with the number of instances, so NCH will be faster overall. This effect fades out if one estimates the parameters using a cross-validation procedure, where the number of folds is not high. In this case, SVM's have the benefit of being presented with test instances from each of the cross-validation test folders en bloc, and use a previously computed explicit decision rule to classify them quickly.

Technically speaking, in case the convex hulls do not overlap, NCH could be solved using the standard SVM optimization formulation (see, e.g., [18], [6]). In this case one searches for the widest margin between each of the k classes and a test point $\mathbf{x}$. This margin represents the distance from $\mathbf{x}$ to the convex hull of the k^{th} class. The class for which the margin is smallest is the winning one. The standard nonseparable-case SVM formulation cannot however be automatically applied to the nonseparable NCH case, since the equality constraint in (2) will not be satisfied in general.

4 EXPERIMENTS ON SIMULATED DATA

We first evaluate the performance of the soft-NCH classifier in an artificial setting using a simulated data set. The next section focuses on real data sets. Here we examine the generalization performance of the NCH classifier on a popularized simulated two-class problem that has been analyzed by many classification methods in [11]. The data set consists of 100 training instanced per class in a 2D input space. Each class is generated from a mixture of Gaussians, which results in a nonlinear Bayes-optimal decision surface between the classes. Refer to [11] for a full description of the data set and its generation.

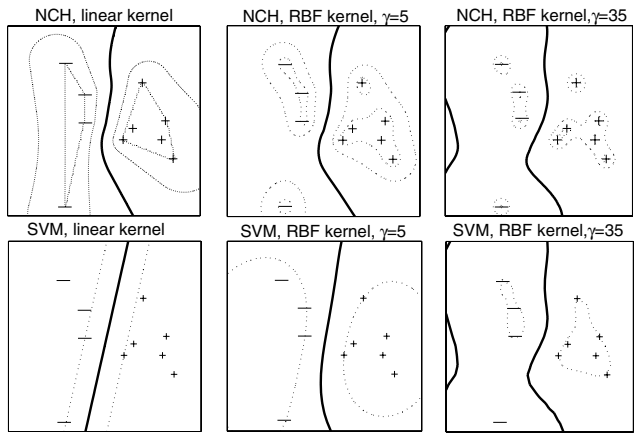

Figure 4. Decision boundaries for NCH and SVM using the linear and RBF kernels on a linearly separable data set. The dashed contours for the NCH method are iso-curves along which the ratio of the distances to the two convex hulls is constant.

The data set together with the leave-one-out NCH-optimal (using the RBF kernel) and the Bayes-optimal separation surfaces are shown in Figure 5. Although the leave-one-out test error of the NCH classier, 0.228, is quite close to the minimal Bayes error, 0.21, the lowest-possible test error of the NCH method is still lower than 0.228 and can be found by tuning the parameters C and γ according to performance on a test set rather than using a cross-validation procedure. We have not however carried out such a brute-force exercise in order to avoid unrealistically optimistic test-error results.

5 EXPERIMENTS ON UCI AND SlatLog DATA SETS

The basic optimization algorithm for Nearest Convex Hull classification (3) is implemented via a modification of the freely available LIBSVM software ([7]). We tested the performance of NCH on several small- to middle-sized data sets that are freely available from the SlatLog and UCI repositories ([15]) and have been analyzed by many researchers and practitioners (e.g. [5], [13], [14], [16] and others): *Sonar*, *Voting*, *Wisconsin Breast Cancer* (W.B.C.), *Heart*, *Australian Credit Approval* (A.C.A.), *Hepatitis* (Hep.), *Haberman*, *Breast Cancer* (B.Cancer), *Colic*, and *German Credit* (CreditG). Detailed information on these data sets can be found on the web sites of the respective repositories. The chosen benchmark data sets have numeric

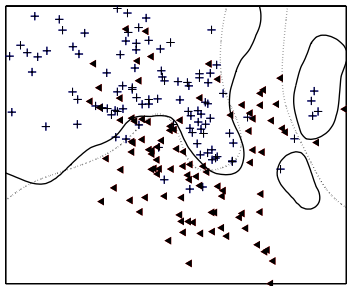

Figure 5. The NCH-optimal (using the RBF kernel) decision boundary found using a leave-one-out cross-validation procedure for tuning the C and γ parameters in solid contour and Bayes-optimal decision boundary in dashed contour on a synthetic data set introduced in [11]. The Bayes error is 0.210, and the test error of the NCH classifier is 0.228. The instances of the two classes are denoted with pluses and triangles. The data set is balanced and consists of 200 instances in total.

attributes, which avoids obtaining favorable results that are due to data-preprocessing rather than qualities of the applied classification techniques. On the other hand, distance-based classifiers are naturally more suitable to numeric domains.

We compare the results of NCH to those of several state-of-art techniques: Support Vector Machines (SVM), Linear and Quadratic Discriminant Analysis (LDA and QDA), Logistic Regression (LR), Multi-layer Perceptron (MLP), k-Nearest Neighbor (kNN), Naive Bayes classifier (NB) and two types of Decision Trees – Decision Stump (DS) and C4.5. The experiments for the NB, LR, MLP, kNN, DS and C4.5 methods have been carried out with the WEKA learning environment using default model parameters, except for kNN. We refer to [21] for additional information on these classifiers and their implementation. We measure model performance by the leave-one-out (LOO) accuracy rate. Because we aim at comparing several methods, LOO seems to be more suitable than the more general k-fold cross-validation (CV), because it always yields one and the same error rate estimate for a given model, unlike the CV method (which involves a random split of the data into several parts).

Table 1 presents performance results for all methods considered. Some methods, namely kNN, NCH and SVM, require tuning of model parameters. In these cases, we report only the highest LOO accuracy rate obtained by performing a grid search for tuning the necessary parameters. Overall, the NCH classifier performs quite well on all data sets, and achieves best accuracy rates on four data sets. Its performance is comparable to that of SVMs, and is better than SVMs in some cases. The rest of the techniques show relatively less favorable and more volatile results. For example, the C4.5 classifier performs best on the *Voting* data set, but achieves rather low accuracy rates on two other data sets – *Sonar* and *Heart*. Note that not all data sets are equally easy to handle. For instance, the performance variation over all classifiers on the *Voting* and *Breast Cancer* data sets is rather low, whereas on the *Sonar* data set it is quite substantial.

6 CONCLUSION

We have introduced a new technique that can be considered as a type of an instance-based large-margin classifier, called Soft Nearest Convex Hull classifier (soft-NCH). The soft-NCH assigns a test observation to the class, which convex hull is closest. Convex-hull overlap is handled via the introduction of slack variables and/or kernels. The soft-NCH method induces an implicit and generally nonlinear decision surface between the classes. One of the advantages of soft-NCH is that an extension from binary to multi-class classification tasks can be carried out in a straightforward way. Others are its alleged robustness to outliers and good generalization qualities. A potential weak point of soft-NCH, which also holds for SVMs, is that it is not clear a priori which type of kernel and what value of the tuning parameters should be used. Furthermore, we do not address the issue of attribute selection and the estimation of class-membership probabilities. Further research could also concentrate on the application of soft-NCH to more domains, on faster implementation suitable for analyzing large-scale data sets, and on the derivation of theoretical test-error bounds.

REFERENCES

[1] Kristin P. Bennett and Erin J. Bredensteiner, 'Duality and geometry in SVM classifiers', in *Proceeding of the 17th International Conference on Machine Learning*, pp. 57–64. Morgan Kaufmann, San Francisco, CA, (2000).

[2] Kristin P. Bennett and Colin Campbell, 'Support vector machines: hype or hallelujah?', *SIGKDD Explor. Newsl.*, **2**(2), 1–13, (2000).

[3] Goldstein J. Ramakrishnan R. Beyer, K. and U. Shaft, 'When is "nearest neighbor" meaningful?', in *Proceeddings of the 7th International Conference on Database Theory*, pp. 217–235, (1999).

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[5] L. Breiman, 'Bagging predictors', *Machine Learning*, **24**, 123–140, (1996).

[6] C.J.C. Burges, 'A tutorial on support vector machines for pattern recognition', *Data Mining and Knowledge Discovery*, **2**, 121–167, (1998).

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2006. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] Gunopulos D. Domeniconi, C. and Jing Peng, 'Large margin nearest neighbor classifiers', *IEEE Transactions on Neural Networks*, **16**, (2005).

[9] Tony Van Gestel, Johan A. K. Suykens, Bart Baesens, Stijn Viaene, Jan Vanthienen, Guido Dedene, Bart De Moor, and Joos Vandewalle, 'Benchmarking least squares support vector machine classifiers.', *Machine Learning*, **54**(1), 5–32, (2004).

[10] T. Hastie and R. Tibshirani, 'Discriminant adaptive nearest neighbor classification', *IEEE Trans. Pattern Anal. Mach. Intell.*, **18**, 607–616, (1996).

[11] Trevor Hastie, Robert Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag New York, Inc., 2001.

[12] Liang Ji Kai Yu and Xuegong Zhang, 'Kernel nearest-neighbor algorithm', *Neural Process. Lett.*, **14**, 147–156, (2002).

[13] R. D. King, C. Feng, and A. Sutherland, 'Statlog: comparison of classification algorithms on large real-world problems', *Applied Artificial Intelligence*, **9(3)**, 289–334, (1995).

[14] T.S. Lim, W.Y. Loh, and Y.S. Shih, 'A comparison of prediction accuracy, complexity, and training time for thirtythree old and new classification algorithms', *Machine Learning*, **40**, 203–228, (1995).

[15] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

[16] Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff, 'Tree induction vs. logistic regression: A learning-curve analysis', *Journal of Machine Learning Research*, **4**, 211–255, (2003).

[17] John Shawe-Taylor and Nello Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, NY, USA, 2004.

[18] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., 1995. 2nd edition, 2000.

[19] P. Vincent and Y. Bengio, 'K-local hyperplane and convex distance nearest-neighbor algorithms', in *Advances in Neural Information Processing Systems*, eds., T. G. Dietterich, S. Becker, and Z. Ghahramani, volume 14, pp. 985–992. MIT Press, (2002).

[20] Ulrike von Luxburg and Olivier Bousquet, 'Distance–based classification with lipschitz functions.', *Journal of Machine Learning Research*, **5**, 669–695, (2004).

[21] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufman, San Francisco, 2005. 2nd edition.