Instance-Based Penalization Techniques for Classification

Georgi I. Nalbantov[1,2], Jan C. Bioch[2], and Patrick J. F. Groenen[2]

[1] ERIM, Erasmus University Rotterdam
[2] Econometric Institute, Erasmus University Rotterdam
nalbantov@few.eur.nl

Econometric Institute Report EI 2007-01

Abstract. Several instance-based large-margin classifiers have recently been put forward in the literature: Support Hyperplanes, Nearest Convex Hull classifier, and Soft Nearest Neighbor. We examine those techniques from a common fit-versus-complexity framework and study the links between them. Finally, we compare the performance of these techniques vis-a-vis each other and other standard classification methods.

1 Introduction

Recently, three classification methods have been introduced in the literature: Support Hyperplanes (SH) [8], Nearest Convex Hull classifier (NCH) [10] and Soft Nearest Neighbor (SNN) [9]. All of them can be classified as instance-based large-margin penalization classifiers. In the following, we argue why these three techniques should perform well based on their favorable generalization qualities. We specifically look at links between Support Vector Machines (SVM), SH, NCH and SNN and approach them intuitively from a common generalization error-versus-complexity point of view. The instance-based nature of the SH, NCH, and SNN arises from the fact that these classifiers do not output an explicit formulation of a decision boundary between the classes. Rather, the classification of each test point is carried out independently of the classification of other test points.

The paper is organized as follows. First, we briefly revise the role of penalization/capacity control for learners in general and argue that the error-versus-complexity paradigm (see e.g. [4], [15], [13]) could be applied to instance-based techniques. Second, we make an intuitive comparison between SVM, SH, NCH, and SNN (in the so-called separable case). Finally, we present some empirical results and conclude.

2 Penalization in Learning

The need for penalization in learning techniques has long been discussed in both the statistical and artificial intelligence/machines learning/data mining communities. Examples of techniques that explicitly employ some kind of penalization are Ridge Regression, Lasso, Support Vector Machines, Support Vector Regression, etc. See, for example, [4] for a review of such methods. Penalization is referred to the practice of purposefully decreasing the ability of a given learner to cope with certain tasks. This ability is referred to as the learner's capacity (see [15]). Arguably, a decreased learner's capacity is responsible for a better prediction performance by mitigating the problem of overfitting. Data overfitting occurs when a learner fits the training data too well, producing very low amount of errors. The amount of errors is referred to as the empirical risk, the empirical error, or the loss. The main idea behind penalization techniques is that the sum empirical error plus capacity control term should be minimized to achieve good prediction results on new data, or in other words, to achieve good generalization ability. In general, if the empirical error over the training data set is rather small, implying a possible overfitting of the training data, then the capacity of the learner is expected to be high. Thus, the generalization sum – empirical error plus capacity – would be relatively high. Hence the need to put up with some increased empirical error over the training data set, which is to be more than offset by a decrease in the learner's capacity. The latter decrease could come about by explicitly penalizing in some way the class of functions to which a learner belongs.

Instance-based, or lazy classification techniques do not have an explicit rule or a decision boundary derived from the training data with which to classify all new observations, or instances. Rather, a new rule for classifying a test instance is derived each time such an instance is given to the learner. A good example of a lazy technique is k-Nearest Neighbor (kNN).

At first sight, a direct application of the idea for penalization on instance-based learners seems hard to materialize. The reason is that penalization in general is applied to a given class of functions, or learners. In the end, one optimal function out of this class should be chosen to classify *any* test observation. This optimal learner produces a minimal generalization sum. The idea for penalization can however also be applied to instance-based classifiers. In this case the function (taken from a given function class) that is used for the classification of a *particular* test instance should be penalized.

Below we give an intuitive account of three rather new instance-based classification techniques, SH, NCH, and SNN. We approach them from a common generalization framework and discuss the links between them and SVM.

3 Three Instance-Based Classification Methods

Given a data set that is separable by a hyperplane and consists of positive and negative observations, let us assume that we would like to classify a new observation $\mathbf{x}$ using a hyperplane, denoted as h. There are two types of hyperplanes: (a) hyperplanes that classify correctly all training data points (called for short consistent hyperplanes) and (b) hyperplanes that do not classify correctly all training data points (called for short inconsistent hyperplanes). For the sake of

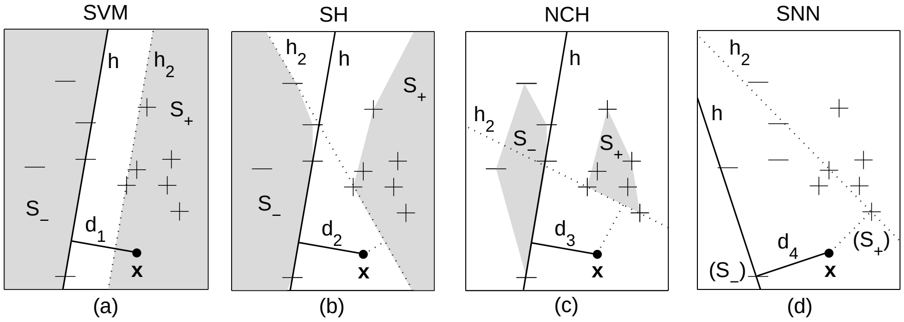

Fig. 1. Binary classification with SVM, SH, NCH, and SNN in Panels (a), (b), (c) and (d), respectively. In all cases, the classification of test point $\mathbf{x}$ is determined using hyperplane h, which is in general different for each method. Equivalently, $\mathbf{x}$ is labeled $+1$ (-1) if it is farther away from set S_- (S_+).

clarity, we consider any hyperplane to be consistent if it does not misclassify any training points.

There are two main factors to be considered in choosing the appropriate h. First, h should not be too close to $\mathbf{x}$. Intuitively speaking, the farther h is from $\mathbf{x}$, the greater the confidence we have in the classification label h assigns to $\mathbf{x}$. Second, h should not make too many mistakes when it classifies the training data. If one chooses h to be extremely far from $\mathbf{x}$, then at one point h will misclassify either all positive or all negative observations. On the other hand, if h classifies correctly all training points, then h might be too close to $\mathbf{x}$, in which case our confidence in the label it assigns to $\mathbf{x}$ is smaller. Thus, in general one cannot have both a big distance between h and $\mathbf{x}$, and a big degree of consistency of h with respect to the training data. A balance between these two desirable properties has unavoidably to be sought. The strife to choose an h that is highly consistent with the training data is referred to as the strife to minimize the empirical risk, empirical error, or training error. The idea to demand h to be as far away from $\mathbf{x}$ as possible can be thought of as a sort of regularization or penalization: the smaller the distance between h and $\mathbf{x}$, the greater the penalty associated with the classification of $\mathbf{x}$. The intuitive assertion here is that the degree of penalization could be proxied by a certain distance. In sum, when classifying a test point $\mathbf{x}$ using a hyperplane, given a separable binary training data set, one is faced with the familiar penalty plus error paradigm (see e.g. [4], [15], [13]). Below we cast four classification methods, SVM, SH, NCH, and SNN, in the light of this paradigm. The hyperplane h with which to classify a new observation $\mathbf{x}$ is in general different for each of these techniques. See Figure 1 for a running toy example.

The h hyperplane in Support Vector Machine classification (see Figure 1a) is defined as the farthest-away from $\mathbf{x}$ consistent hyperplane that is parallel to another consistent hyperplane, h_2, in such a way that the distance between these two hyperplanes (referred to as the "margin") is maximal. Since h is consistent with the training data, the empirical error it makes on the data is zero. The magnitude of the penalty associated with the classification of $\mathbf{x}$ can be considered to be positively related to the inverse of the distance between $\mathbf{x}$ and h ($1/d_1$ in terms of Figure 1a). The (theoretical) instance-based SVM classification algorithm can be stated as follows: first add $\mathbf{x}$ to the data set with -1 label and compute the distance d_1 to h (as defined above). Second, add $\mathbf{x}$ to the data set with $+1$ label and compute the distance d_1^* to h_2. Third, classify $\mathbf{x}$ using h (that is, as -1) if $d_1 > d_1^*$; classify $\mathbf{x}$ using h_2 (as $+1$) if $d_1 < d_1^*$; otherwise, if $d_1 = d_1^*$, the classification of $\mathbf{x}$ is undetermined.

The h hyperplane in SH classification (see Figure 1b) is defined as the farthest-away from $\mathbf{x}$ consistent hyperplane. Since h is consistent with the training data, the empirical error it makes on the data is zero. The magnitude of the penalty associated with the classification of $\mathbf{x}$ can be considered to be positively related to the inverse of the distance to h ($1/d_2$ in terms of Figure 1b). It can be shown that $d_2 \geq d_1$ always. Therefore, the sum empirical error plus penalty for SH is always smaller than the corresponding sum for SVM, suggesting that SH may possess better generalization ability than SVM. The SH classification algorithm can be stated as follows. First, add $\mathbf{x}$ to the training data set with -1 label and compute the distance d_2 to h. Note that h is consistent with both the original training data and with $\mathbf{x}$. That is, h assigns label -1 to $\mathbf{x}$. Second, add $\mathbf{x}$ to the original data set with $+1$ label and compute the distance d_2^* to h_2. In this case, h_2 is defined as the farthest-away hyperplane from $\mathbf{x}$ that is consistent with both $\mathbf{x}$ and the original training data. Third, classify $\mathbf{x}$ using h (that is, as -1) if $d_2 > d_2^*$; classify $\mathbf{x}$ using h_2 (as $+1$) if $d_2 < d_2^*$; otherwise, if $d_2 = d_2^*$, the classification of $\mathbf{x}$ is undetermined.

The h hyperplane in NCH classification (see Figure 1c) is defined as the farther of two hyperplanes. The first one is the hyperplane farthest away from $\mathbf{x}$ that is consistent with all positive observations and $\mathbf{x}$, where $\mathbf{x}$ has label -1. The second one if the hyperplane farthest away from $\mathbf{x}$ that is consistent with all the negative observations and $\mathbf{x}$, where $\mathbf{x}$ has label $+1$. Effectively, $\mathbf{x}$ is classified as $+1$ (-1) if it is closer to the convex hull of $+1$ (-1) points. The magnitude of the penalty associated with the classification of $\mathbf{x}$ is considered to be positively related to the inverse of the distance from $\mathbf{x}$ to h ($1/d_3$ in terms of Figure 1c). It can be shown that $d_3 \geq d_2 \geq d_1$ always. However, the empirical error on the training set is not guaranteed to be equal to zero. This happens because h should be consistent with at least all positive or all negative observations, and not with both all negative and all positive observations. Thus, the generalization sum training error plus penalty is not guaranteed to be smaller for NCH than for SH or SVM. The NCH classification algorithm can be stated as follows. First, add $\mathbf{x}$ to the training data set with -1 label and compute the distance d_3 to h, the hyperplane that is consistent with all $+1$ points and $\mathbf{x}$. This distance is the distance between $\mathbf{x}$ and the convex hull of the positive points. Second, add $\mathbf{x}$ to the training data set with $+1$ label and compute the distance d_3^* to h_2, the hyperplane that is consistent with all -1 points and $\mathbf{x}$. Third, classify $\mathbf{x}$ using

h (that is, as -1) if $d_3 > d_3^*$; classify $\mathbf{x}$ using h_2 (as $+1$) if $d_3 < d_3^*$; otherwise, if $d_3 = d_3^*$, the classification of $\mathbf{x}$ is undetermined.

The SNN classification can also be presented along similar lines as SVM, SH, and NCH. In the separable case, SNN is equivalent to the classical First Nearest Neighbor (1NN) classifier. The h hyperplane in 1NN classification (see Figure 1d) is the farther of two hyperplanes. The first one is farthest away from $\mathbf{x}$ hyperplane that is consistent with the closest positive observation and $\mathbf{x}$, where $\mathbf{x}$ has label -1. The second hyperplane is the farthest away from $\mathbf{x}$ hyperplane that is consistent with the closest negative observation and $\mathbf{x}$, where $\mathbf{x}$ has label $+1$. Effectively, $\mathbf{x}$ is classified as $+1$ (-1) if its closest training point has label $+1$ (-1). The magnitude of the penalty associated with the classification of $\mathbf{x}$ is considered to be positively related to the inverse of the distance from $\mathbf{x}$ to h ($1/d_4$ in terms of Figure 1d). It can be shown that $d_4 \geq d_3 \geq d_2 \geq d_1$ always, suggesting (somewhat counterintuitively) that 1NN provides for the greatest penalization among the four techniques under consideration. However, the empirical error in 1NN on the training data set is certainly not guaranteed to be equal to zero. In fact, h is not even guaranteed to be consistent with either all positive or all negative points, as the case is in NCH classification, as well as in SH and SVM classification. Thus, the h hyperplane in 1NN is likely to commit the greatest amount of errors on the training data set as compared to SVM, SH and NCH. Consequently, the generalizability sum empirical error plus penalty may turn out to be the highest. Note however that it could also turn out to be the lowest for some $\mathbf{x}$, in which case 1NN exhibits the highest generalization ability. The 1NN classification algorithm can be (theoretically) stated as follows. First, add $\mathbf{x}$ to the training data set with label -1 and compute the distance d_4 to h, the hyperplane that is consistent with $\mathbf{x}$ and the closest positive point. Second, add $\mathbf{x}$ to the training data set with $+1$ label and compute the distance d_4^* to h_2, the hyperplane that is consistent with $\mathbf{x}$ and the closest negative point. Third, classify $\mathbf{x}$ using h (that is, as -1) if $d_4 > d_4^*$; classify $\mathbf{x}$ using h_2 (as $+1$) if $d_4 < d_4^*$; otherwise, if $d_4 = d_4^*$, the classification of $\mathbf{x}$ is undetermined.

4 Alternative Specifications

In the separable case, there is an alternative, but equivalent, formulation of the SVM, SH, NCH, and SNN techniques in terms of distances to sets as opposed to distances to hyperplanes. The corresponding sets for each technique are depicted in Figure 1 as shaded areas. A common classification rule for all methods can be defined as follows: a new point $\mathbf{x}$ should be classified as -1 if it is farther from set S_+ than from set S_-; $\mathbf{x}$ should be classified as $+1$ if it is farther from set S_- than from set S_+; otherwise, if the distance to both S_+ and S_- is the same, the class of $\mathbf{x}$ is undetermined as $\mathbf{x}$ lies on the decision boundary. Sets S_+ and S_- are defined differently for each method.

For SVM, set S_+ is defined as the set of all points that are classified as $+1$ by all hyperplanes that lie inside the SVM margin. Set S_- is similarly defined as the set of all points that are classified as -1 by all hyperplanes that lie inside the SVM margin.

For SH, set S_+ is the set of all points classified as $+1$ by all hyperplanes that are consistent with the training data. The latter include all hyperplanes that lie inside the SVM margin plus all the rest of the consistent hyperplanes. Analogically, set S_- is defined as the set of all points that are classified as -1 by all consistent hyperplanes. The collection of all consistent hyperplanes is referred to in the literature as the version space ([7]) of hyperplanes with respect to a given training data set. A conservative version-space classification rule is to classify a test point $\mathbf{x}$ only if all consistent hyperplanes assign one and the same classification label to it ([14]), or in other words if $\mathbf{x}$ belongs to either S_+ or S_-.

For the NCH classifier, set S_+ is the set of all points that are classified as $+1$ by all hyperplanes that are consistent with the positively-labeled data points. In other words, S_+ is the convex hull of the positive observations. Set S_- is defined as the set of all points that are classified as -1 by all hyperplanes that are consistent with the negatively-labeled data points. Thus, S_- is the convex hull of the negative points.

Lastly, for the 1NN classifier, which is the hard-margin version of the SNN classifier, the S_+ set consists of just one point: the closest to $\mathbf{x}$ positively-labeled point. Set S_- also consists of just one point: the closest to $\mathbf{x}$ negatively-labeled point.

5 Estimation

We now review the estimation of SVM, SH, NCH, and SNN. Further details can be found, e.g., in [2], [15], [8], [10], [9]. We examine a common setup for the four techniques: a binary classification data set $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, where each $\mathbf{x}_i$ is an n-dimensional vector of values for the predictor variables and each y_i is either a $+1$ or a -1 observation label. The classification task is: given a test point $\mathbf{x}$, output its predicted label. Each of the techniques solves an optimization problem to find an optimal hyperplane, $\mathbf{w}^{*\prime}\mathbf{x} + b^* = 0$, with which to classify the test observation in the way presented in Section 3. Here $\mathbf{w}$ is a vector of hyperplane coefficients, b is the intercept, and the asterisk (*) indicates optimal values.

5.1 Support Vector Machines

SVM solve the classification task by maximizing the so-called margin between the classes. In the separable case, the margin is equal to the distance between the convex hulls of the two classes at the optimal SVM solution ([15]). Formally, the margin is equal to the distance between hyperplanes $\mathbf{w}'\mathbf{x} + b = -1$ and $\mathbf{w}'\mathbf{x} + b = 1$, presented already as h and h_2 in Figure 1a. Thus, the margin equals $2/||\mathbf{w}||$. Maximizing the margin is equivalent to minimizing the term $||\mathbf{w}||^2/2 = \mathbf{w}'\mathbf{w}/2$. Formally, to find the SVM hyperplane h, one solves the following optimization problem:

$$
\begin{aligned}
&\min_{\mathbf{w},b} && \frac{1}{2}\mathbf{w}'\mathbf{w} \\
&\text{s.t.} && y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1,\ i = 1, 2, \ldots, l.
\end{aligned} \tag{1}
$$

If there is no hyperplane that is able to separate the classes, so-called slack variables ξ_i are introduced. This case is referred to as the nonseparable case or the class-overlapping case. Then, problem (1) becomes:

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

where $C > 0$ is a manually adjustable constant that regulates the trade-off between the penalty term $\mathbf{w}'\mathbf{w}/2$ and the loss $\sum_{i=1}^{l}\xi_i$.

Optimization problem (2) can be dualized as:

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

where the α_i's are the Lagrange multipliers associated with (2). The advantage of the dual is that different nonlinear mappings $\mathbf{x} \rightarrow \phi(\mathbf{x})$ of the data can easily handled. Thus, if one first transforms the data into a higher-dimensional space, where the coordinates of the data points are given by $\phi(\mathbf{x})$ instead of $\mathbf{x}$, then the dot product $\mathbf{x}_i'\mathbf{x}_j$ will appear as $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ in the dual optimization problem. There exist so-called kernel functions $\kappa(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ that compute this dot product efficiently, without explicitly carrying the transformation mapping. Popular kernels are the 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 $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$. The mapping $\mathbf{x} \rightarrow \phi(\mathbf{x})$ when the RBF kernel is used corresponds to a mapping into an infinite-dimensional space. The manually-adjustable γ parameter of the RBF kernel determines the proximity of any two points in this infinite-dimensional space.

5.2 Support Hyperplanes

In the separable case, the h hyperplane in SH classification, with which to classify test point $\mathbf{x}$, can be found as the solution of the following optimization problem:

$$
\begin{aligned}
&\min_{\mathbf{w},b,y_{l+1}} && \frac{1}{2}\mathbf{w}'\mathbf{w} \\
&\text{s.t.} && y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0,\ i = 1, 2, \ldots, l \\
& && y_{l+1}(\mathbf{w}'\mathbf{x} + b) = 1, y_{l+1} \in \{-1, 1\}.
\end{aligned} \tag{4}
$$

This problem is partially combinatorial due to the constraint that the predicted label of $\mathbf{x}$, y_{l+1}, can take on only two values. Therefore, one usually solves two separate optimization subproblems: in the first one $y_{l+1} = +1$, and in the second one $y_{l+1} = -1$. The value of y_{l+1} that minimizes the objective function in (4) is the predicted label of $\mathbf{x}$. Note that the distance between $\mathbf{x}$ and h is defined as $1/\sqrt{\mathbf{w}^{*\prime}\mathbf{w}^{*}}$ by the equality constraint $y_{l+1}(\mathbf{w}'\mathbf{x} + b) = 1$.

In the nonseparable case, SH introduce slack variables ξ_i, similarly to SVM. As a result, the nonseparable version of (4) becomes:

$$\begin{aligned} \min_{\mathbf{w},b,y_{l+1},\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}'\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, \ldots, l \\ & y_{l+1}(\mathbf{w}'\mathbf{x}_{l+1} + b) = 1, y_{l+1} \in \{-1, 1\}. \end{aligned} \tag{5}$$

As in (4), two separate optimization problems have to be solved to determine the optimal y_{l+1}. Each of these two subproblems can be dualized as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \alpha_{l+1} - \tfrac{1}{2}\textstyle\sum_{i,j=1}^{l+1}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i'\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ i = 1, 2, \ldots, l, \text{ and } \textstyle\sum_{i=1}^{l+1} y_i\alpha_i = 0. \end{aligned} \tag{6}$$

Similarly to SVM, different kernels can be substituted for the dot product $\mathbf{x}_i'\mathbf{x}_j$.

5.3 Nearest Convex Hull Classifier

The optimization problem for the NCH classifier is almost identical to the SH one. The only difference is that in each of the two optimization subproblems observations from only one class are considered. This property enables NCH to handle the multi-class classification case with ease, unlike SVM and SH. In the two-class problem at hand, let us denote with S_+ the set of observations that belong to the positive class and with S_- the set of observations that belong to the negative class. Next, two optimization problems are solved, one per each class k:

$$\begin{aligned} \min_{\mathbf{w}_k,b_k} \quad & \frac{1}{2}\mathbf{w}_k'\mathbf{w}_k \\ \text{s.t.} \quad & \mathbf{w}_k'\mathbf{x}_i + b_k \geq 0,\ i \in S_k \\ & -(\mathbf{w}_k'\mathbf{x} + b_k) = 1. \end{aligned} \tag{7}$$

The distance from $\mathbf{x}$ to the k^{th} class is defined as $1/\sqrt{\mathbf{w}_k^{*\prime}\mathbf{w}_k^{*}}$ by the equality constraint in (7). The class associated with the smallest such distance is assigned to the test point $\mathbf{x}$. Notice that this distance is inversely related to the objective function $\mathbf{w}_k'\mathbf{w}_k/2$. Therefore, the class k that achieves the maximal value for this objective function should be assigned to $\mathbf{x}$.

In the nonseparable case, each of the optimization subproblems is expressed as:

$$\begin{aligned} \min_{\mathbf{w}_k, b_k, \boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}_k'\mathbf{w}_k + C\sum_{i \in S_k} \xi_i \\ \text{s.t.} \quad & \mathbf{w}_k'\mathbf{x}_i + b_k \geq 0 - \xi_i,\ \xi_i \geq 0,\ i \in S_k \\ & -(\mathbf{w}_k'\mathbf{x} + b_k) = 1, \end{aligned} \tag{8}$$

where the ξ's are slack variables. In dual form, (8) becomes:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \alpha_{l_k+1} - \tfrac{1}{2}\textstyle\sum_{i,j=1}^{l_k+1} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i'\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ i = 1, 2, \ldots, l_k,\ \text{and } \textstyle\sum_{i=1}^{l_k+1} y_i \alpha_i = 0, \end{aligned} \tag{9}$$

allowing for the employment of kernel functions, as in SVM and SH. Here $i = 1, 2, \ldots, l_k$ denotes the elements of class k.

5.4 Soft Nearest Neighbor

In the separable case, SNN is equivalent to the 1NN classifier. Instead of computing the distances between $\mathbf{x}$ and all data points to determine the nearest neighbor of $\mathbf{x}$ however, SNN take a different approach. Observe that the distance to the nearest neighboring point is equal to the maximal radius of a (hyper)sphere with center $\mathbf{x}$ that does not contain any training data points. To find this radius r, one solves the following optimization problem:

$$\begin{aligned} \max \quad & r^2 \\ \text{s.t.} \quad & r^2 \leq ||\mathbf{x}_i - \mathbf{x}||^2,\ i = 1, 2, \ldots, l. \end{aligned} \tag{10}$$

In SNN classification, one first finds the distances between $\mathbf{x}$ and the closest point from each of the two (or, in general k) classes. Point $\mathbf{x}$ is then assigned to the class, which such point is closer/closest to $\mathbf{x}$. Denoting with S_+ and S_- the sets of positive and negative observations, respectively, SNN thus solve one optimization problem per each class k, of the form:

$$\begin{aligned} \max \quad & r^2 \\ \text{s.t.} \quad & r^2 \leq ||\mathbf{x}_i - \mathbf{x}||^2,\ i \in S_k. \end{aligned} \tag{11}$$

The class that produces the minimal value for the objective function $\mathbf{R}^2$ of (11) is then assigned to point $\mathbf{x}$. Similarly to the SVM, SH and NCH approaches, one can introduce slack variables ξ_i. In this case (11) becomes:

$$\begin{aligned} \max \quad & \mathbf{R}^2 - C \sum_{i \in S_k} \xi_i && (12) \\ \text{s.t.} \quad & \mathbf{R}^2 \leq \ ||\mathbf{x}_i - \mathbf{x}||^2 + \ \xi_i, \ \xi_i \geq 0, \ i \in S_k. \end{aligned}$$

The $C > 0$ parameter controls the trade-off between the length of the radius and amount of training errors. A training error occurs if a point lies inside the hypersphere. Each of the k quadratic optimization problems (12) can be expressed in dual form as:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{i \in S_k} \alpha_i (\mathbf{x}_i' \mathbf{x}_i - 2(\mathbf{x}_i' \mathbf{x}) + \mathbf{x}' \mathbf{x}) && (13) \\ \text{s.t.} \ & 0 \leq \alpha_i \leq C, \ i \in S_k, \ \text{and} \sum_{i \in S_k} \alpha_i = 1. \end{aligned}$$

This formulation allows for the employment of different kernels, which can replace the dot products $\mathbf{x}_i'\mathbf{x}_i$, $\mathbf{x}_i'\mathbf{x}$ and $\mathbf{x}'\mathbf{x}$. Notice that unlike (12), (13) is a linear programming problem.

6 Comparison Results

The basic optimization algorithms for SH, NCH, and SNN classification, (6), (9) and (13) respectively, are implemented via a modification of the freely available LIBSVM software ([3]). We tested the performance of SH, NCH, and SNN on several small- to middle-sized data sets that are freely available from the SlatLog and UCI repositories ([11]) and have been analyzed by many researchers and practitioners (e.g. [1], [5], [6], [12] and others): *Sonar*, *Voting*, *Wisconsin Breast Cancer* (W.B.C.), *Heart*, *Australian Credit Approval* (A.C.A.), and *Hepatitis* (Hep.). Detailed information on these data sets can be found on the web sites of the respective repositories. We stop short of carrying out an extensive experimental study, since this falls out of the main scope of the paper. Furthermore, large data sets are harder to handle due to the instance-based nature of the SH, NCH, and SNN classifiers.

We compare the results of SH, NCH and SNN 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 [16] for additional information on these classifiers and their implementation. We measure model performance by the leave-one-out (LOO) accuracy rate. For our purposes – comparison between the 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, because it involves a random split of the data into several parts.

Table 1. Leave-one-out accuracy rates (in %) of the Support Hyperplanes (SH), Nearest Convex Hull (NCH) and Soft Nearest Neighbor (SNN) classifiers 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

	SH	SH	SH	NCH	NCH	NCH	SNN	SNN	SNN	SVM	SVM	SVM								
	rbf	2p	lin	rbf	2p	lin	rbf	2p	lin	rbf	2p	lin	NB	LR	LDA	QDA	MLP	*k*NN	DS	C4.5
Sonar	**91.35**	87.98	79.80	**91.35**	90.38	87.98	88.46	76.92	87.50	88.94	82.21	80.77	67.30	73.08	75.48	74.88	81.25	86.54	73.08	71.15
Voting	96.77	96.31	96.77	95.85	85.48	95.85	94.47	94.01	93.78	96.54	96.31	96.77	90.32	96.54	95.85	94.24	94.93	93.32	95.85	**97.00**
W.B.C.	**97.42**	96.85	97.00	**97.42**	97.14	97.28	**97.42**	97.28	97.28	97.00	96.85	96.85	95.99	96.14	95.99	91.42	94.99	97.00	92.42	95.28
Heart	**85.56**	81.90	**85.56**	**85.56**	82.59	84.07	85.19	80.74	**85.56**	**85.56**	81.11	85.56	82.96	83.70	83.70	81.48	78.89	84.44	76.30	75.19
A.C.A.	**87.39**	86.70	86.80	86.38	85.36	86.09	85.80	85.51	85.65	**87.39**	79.86	87.10	77.10	86.38	85.80	85.22	84.78	85.94	85.51	83.77
Hep.	**87.70**	86.45	86.45	85.16	84.52	84.52	87.10	85.16	85.16	**86.45**	86.45	86.45	83.23	83.87	85.81	83.87	79.35	85.81	79.35	80.00

Table 1 presents performance results for all methods considered. Some methods, namely kNN, SH, NCH, SNN 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 instance-based penalization classifiers SH, NCH and SNN perform quite well on all data sets. Most notably, SH achieve best accuracy rates on five data sets. NCH replicate this success three times. SVM also perform best on three data sets. The SNN classifier achieves best accuracy rate on just two data sets, but five times out of six performs better than its direct competitor, kNN. 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.

7 Conclusion

We have studied from a common generalization perspective three classification methods recently introduced in the literature: Support Hyperplanes, Nearest Convex Hull classifier and Soft Nearest Neighbor. In addition, we have compared them to the popular Support Vector Machines. A common theme in SH, NCH, and SNN is their instance-based nature. In addition, these methods strive to find a balance between learner's capacity and learner's fit over the training data. Last but not least, the techniques can be kernelized, which places them also in the realm of kernel methods. We have provided a rather intuitive treatment of these techniques and the generalization framework from which they are approached. Further research could concentrate on more detailed such treatment and on the derivation of theoretical test-error bounds. Extensive experiments with different loss functions, such as the quadratic one, have also to be carried out. Last but not least, ways to improve the computational speed can also be explored.

References

1. L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
2. C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
3. 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`.
4. Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer-Verlag New York, Inc., 2001.
5. 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.

6. 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.
7. T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.
8. G. I. Nalbantov, J. C. Bioch, and P. J. F. Groenen. Classification with support hyperplanes. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006: 17th European Conference on Machine Learning, Berlin, Germany*, Lecture Notes in Computer Science, pages 703–710. Springer Berlin / Heidelberg, 2006.
9. G. I. Nalbantov, J. C. Bioch, and P. J. F. Groenen. Soft nearest neighbor. Econometric institute technical report, Erasmus University Rottedam, 2006. To Appear.
10. G. I. Nalbantov, P. J. F. Groenen, and J. C. Bioch. Nearest convex hull classification. Econometric institute technical report, Erasmus University Rottedam, 2006. To Appear.
11. D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html` University of California, Irvine, Dept. of Information and Computer Sciences.
12. 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.
13. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
14. E.N. Smirnov, I.G. Sprinkhuizen-Kuyper, G.I. Nalbantov, and S. Vanderlooy. Version space support vector machines. In A. Perini G. Brewka, S. Coradeschi and P. Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence*, ECAI 2006, pages 809–810. IOS Press, Amsterdam, The Netherlands, 2006.
15. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995. 2nd edition, 2000.
16. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufman, San Francisco, 2005. 2nd edition.