Aggregating Classifiers With Ordinal Response Structure

Gerhard Tutz & Klaus Hechenbichler
{tutz,hechen}@stat.uni-muenchen.de
Ludwig-Maximilians-Universität München, Akademiestraße 1,
80799 München, Germany

24th July 2003

Abstract

In recent years the introduction of aggregation methods led to many new techniques within the field of prediction and classification. The most important developments, bagging and boosting, have been extensively analyzed for two and multi class problems. While the proposed methods treat the class indicator as a nominal response without any structure, in many applications the class may be considered as an ordered categorical variable. In the present paper variants of bagging and boosting are proposed which make use of the ordinal structure. It is demonstrated how the predictive power is improved by use of appropriate aggregation methods. Comparisons between the methods are based on misclassification rates as well as criteria that take ordinality into account, like absolute or squared distance measures.

1 Introduction

A common problem in multivariate statistics is classification where covariates are used to predict the value of a class variable. Over the years various methods of classification have been proposed ranging from Fisher's linear discriminant analysis over classification trees (Breiman, Friedman, Olshen & Stone, 1984) to support vector machines. Statistical concepts of classification are nicely summarized in McLachlan (1992), Ripley (1996), Hastie, Tibshirani & Friedman (2001).

In recent years the introduction of aggregation methods led to spectacular improvements of standard techniques of classification. Especially the development of bagging (Breiman, 1996; see also Breiman, 1998) and boosting (Freund, 1995; see also Freund & Schapire, 1996) made aggregation methods an area of intensive research. The principle is to use a basic discrimination method not only once but for different (bootstrap) versions of the data set. While bagging samples are based on the unweighted bootstrap, boosting uses weights that depend on the performance in the last sample.

The properties of bagging have been investigated extensively (e.g. Bühlmann & Yu, 2002a; Friedman & Hall, 2000) and further improvements have been proposed (e.g. Bühlmann, 2003; Hothorn & Lausen, 2003). Boosting has been introduced originally in the machine learning community. Many authors contributed to the field from a statistical point of view (e.g. Schapire, Freund, Bartlett & Lee, 1998; Friedman, 1999; Friedman, Hastie & Tibshirani, 2000; Friedman, 2001; Dudoit, Fridlyand & Speed, 2002; Dettling & Bühlmann, 2003; Bühlmann & Yu, 2002b; Bühlmann, 2002; Dettling & Bühlmann, 2003). A detailed overview can be found in Schapire (2002). Extensive studies of comparisons between bagging, boosting and variants have been carried out e.g. by Breiman (1998), Dietterich (2000), Bauer & Kohavi (1999).

A problem that is closely related to classification is how to predict categorical ordered response variables. In usual classification problems the class variable is assumed to have nominal scale level. But if there is ordinal structure within the classes, this additional information should be used for classification. The use of parametric ordinal models has been investigated e.g. by Anderson & Philips (1981) and Rudolfer, Watson & Lesaffre (1995).

The purpose of this paper is to develop aggregation methods that exploit the ordering of the classes. In Section 2 the basic building blocks are considered. In Section 3 classifiers for ordered response categories are built. We distinguish between two methods: Fixed split boosting in which aggregation methods are applied to a fixed dichotomization of the response categories and methods where ordinal aggregation is performed in each iteration step.

2 Aggregating classifiers: bagging and boosting

In multivariate discrimination each object is assumed to come from one out of k classes. On the basis of an observation vector x consisting of p characteristics associated with the object an assessment as to the class is to be made. A predictor or classifier has to be built from past experience. Let $L = \{(Y_i, x_i), i = 1, \ldots, n_L\}$ denote a learning or training set of observed data, where $Y_i \in \{1, \ldots, k\}$ denotes the class and $x_i' = (x_{i1}, \ldots, x_{ip})$ are associated covariates.

A classifier based on learning set L partitions the space X of covariates in the form

$$\begin{aligned} C(.,L) : X &\longrightarrow \{1, \ldots, k\} \\ x &\longrightarrow C(x, L) \end{aligned}$$

where $C(x, L)$ is the predicted class for observation x.

Breiman (1996, 1998) found some benefit by using perturbed versions of the learning set and aggregate the corresponding predictors by plurality voting, where the winning class is the one being predicted by the largest number of predictors. In weighted plurality voting the predicted class for an observation x is given by

$$\text{argmax}_j \sum_{m=1}^{M} c_m I(C(x, L_m) = j)$$

where L_m is the m-th version of the learning set, c_m are weights for learning set L_m and $I(.)$ denotes the indicator function, equaling 1 if the condition in parentheses is true, and 0 otherwise.

The strength of a prediction may be measured by prediction votes (Dudoit, Fridlyand & Speed, 2002) which for observation x are defined by

$$\text{PV}(x) = \frac{\max_j \sum_{m=1}^{M} c_m I(C(x, L_m) = j)}{\sum_{m=1}^{M} c_m} \quad .$$

In the bootstrap aggregating or *bagging* procedure (Breiman, 1996) perturbed learning sets of size n_L are formed by drawing from the learning set L at random. The predictor $C(., L_m)$ is built from the m-th bootstrap sample. Aggregating uses the weights $c_m = 1$. Alternatives to this simple nonparametric bootstrap are the parametric bootstrap (Dudoit, Fridlyand & Speed, 2002) and learning sets based on convex pseudo data (Breiman, 1998).

In *boosting* the data are resampled adaptively and the predictors are aggregated by weighted voting. The discrete AdaBoost procedure proposed by Freund (1995) starts with weights $w_1 = \cdots = w_{n_L} = 1/n_L$. In the following the m-th step of the algorithm is given.

Discrete AdaBoost (m-th step)

1. (a) The current weights $w_1, \ldots, w_{n_L}$ form the resampling probabilities. Based on these probabilities the learning set L_m is sampled from L with replacement.

   (b) Based on L_m the classifier $C(., L_m)$ is built.

2. The learning set is run through the classifier $C(., L_m)$ yielding error indicators $\epsilon_i = 1$ if the i-th observation is classified incorrectly and $\epsilon_i = 0$ otherwise.

3. With $e_m = \sum_{i=1}^{n_L} w_i \epsilon_i$, $b_m = (1 - e_m)/e_m$ and $c_m = \log(b_m)$ the resampling weights are updated for the next step by

$$w_{i,new} = \frac{w_i b_m^{\epsilon_i}}{\sum_{j=1}^{n_L} w_j b_m^{\epsilon_j}} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n_L} w_j \exp(c_m \epsilon_j)} \quad .$$

After M steps the aggregated voting for observation x is obtained by

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \qquad \square$$

While e_m is a weighted sum of errors the parameters $c_m = \log((1-e_m)/e_m)$ are log odds comparing weighted hits to errors. It is easily seen that for the new weighting scheme one has $\sum_{\epsilon_i=1} w_{i,new} = \sum_{\epsilon_i=0} w_{i,new} = 0.5$ and therefore in the next step the resampling probability put on the observations which have been misclassified in the m-th step has sum 0.5 . If $e_m = 0$ then $c_m \to \infty$ is no longer defined. This problem is avoided by adding $1/n_L$ to the denominator of b_m, yielding $c_m = \log((1-e_m)/(e_m + 1/n_L))$.

Step 1 of the algorithm is based on weighted resampling. In alternative versions of boosting observations are not resampled. Instead the classifier is computed by weighting the original observations by weights $w_1, \ldots, w_{n_L}$, that are updated iteratively. Then $C(., L_m)$ should be read as the classifier based on the current weights $w_1, \ldots, w_L$ (in the m-th step). The use of weights instead of resampled observations depends on the availability of program packages which can handle weighted observations.

In the case of two classes it is more common to use binary observations $y_i \in \{1, 0\}$ or $\tilde{y}_i \in \{1, -1\}$ as indicators for class. The latter version is more often used in the machine learning community. The class indicator $Y \in \{1, 2\}$ is transformed to the binary case by using $y_i = 1$ if $Y_i = 1$ and $y_i = 0$ if $Y_i = 2$. The version $\tilde{y}_i \in \{1, -1\}$ is obtained from $\tilde{y}_i = 2y_i - 1$.

Real AdaBoost (Friedman, Hastie & Tibshirani, 2000) uses real valued classifier functions $f(x, L)$ instead of $C(x, L)$ with the convention that $f(x, L) \geq 0$ corresponds to $C(x, L) = 1$ and $f(x, L) < 0$ corresponds to $C(x, L) = 2$.

Real AdaBoost (m-th step)

1. Based on weights $w_1, \ldots, w_{n_L}$ the classifier $C(., L_m)$ is built.
2. The learning set is run through the classifier $C(., L_m)$ yielding estimated class probabilities $p(x_i) = \hat{P}(\tilde{y}_i = 1|x_i)$.
3. Based on these probabilities a real valued classifier is built by

$$f(x_i, L_m) = 0.5 \cdot \log \frac{p(x_i)}{1 - p(x_i)}$$

and the weights are updated for the next step by

$$w_{i,new} = \frac{w_i \exp(-\tilde{y}_i f(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-\tilde{y}_j f(x_j, L_m))} \quad .$$

After M steps the aggregated voting for observation x is obtained by

$$\text{sign}\left(\sum_{m=1}^{M} f(x, L_m)\right) \qquad \square$$

It should be noted that in this version of Real AdaBoost either resampled observations or weighted observations may be used.

The essential term in the updating procedure is $w_i \exp(-\tilde{y}_i f(x_i, L_m))$ which, depending on hits ($\epsilon_i = 0$) and misclassifications ($\epsilon_i = 1$), has the form

$$w_i \exp(-\tilde{y}_i f(x_i, L_m)) = \begin{cases} w_i \exp(-|f(x_i, L_m)|) & \epsilon_i = 0 \\ w_i \exp(|f(x_i, L_m)|) & \epsilon_i = 1 \end{cases} \quad .$$

It is seen that for misclassified observations the weight w_i is increased whereas for correctly classified observations w_i is decreased. In order to ensure existence of $f(x_i, L_m)$, $1/n_L$ is added to numerator and denominator of the fraction, yielding

$$f(x_i, L_m) = 0.5 \cdot \log \frac{p(x_i) + 1/n_L}{1 - p(x_i) + 1/n_L} \quad .$$

3 Ordinal prediction

For the multi class problem $Y \in \{1, \ldots, k\}$ it is assumed that the responses or classes are ordered. For example Y may be the years of survival in a survival study with the last category denoting survival beyond the end of the study. In the following we consider two approaches to exploit the ordinal structure of the response categories. The first approach, fixed split boosting, reduces the problem to binary classification problems by splitting the response categories. After using boosting techniques for the binary classification problems the resulting classifiers are combined to obtain the final classifier. In the second approach the ordinal structure is explicitely used within each boosting iteration.

3.1 Fixed split boosting

In the following the classification procedure is divided into two stages. At the first stage one uses aggregation techniques after splitting the response categories. At the second stage the resulting binary classifiers are combined. Since boosting is used for fixed splits within response categories we refer to the method as *fixed split boosting.*

The reduction to binary problems is made by considering splits within response categories. It works by defining

$$Y^{(r)} = \begin{cases} 1 & Y \in \{1, \ldots, r\} \\ 2 & Y \in \{r+1, \ldots, k\} \end{cases}$$

for $r = 1, \ldots, k-1$. It should be noted that this type of splitting is sensible only for ordered categories.

Let $C^{(r)}(., L)$ denote the classifier for the binary class problem defined by $Y^{(r)}$, i.e. for the splitting $\{1, \ldots, r\}, \{r+1, \ldots, k\}$. For fixed r, by using any form of aggregate classifier one obtains the predicted class for observation x by

$$C_{agg}^{(r)}(x) = \text{argmax}_j \sum_{m=1}^{M} c_m^{(r)} I(C^{(r)}(x, L_m^{(r)}) = j) \quad . \tag{1}$$

The first stage aggregate classifier $C_{agg}^{(r)}$ has been designed for a fixed split at r. The combination of the aggregate classifiers $C_{agg}^{(1)}, \ldots, C_{agg}^{(k-1)}$ is based on the second stage aggregation, now by exploiting the ordering of the categories. Thereby one uses that the response $Y \in \{1, \ldots, k\}$ may also be characterized by the binary vector response $(y_1, \ldots, y_k)$ where

$$y_j = \begin{cases} 1 & Y = j \\ 0 & \text{otherwise} \end{cases} \quad .$$

The classifier $C_{agg}^{(r)}$ is a prediction of the binary class indicator $Y^{(r)}$. Let the result of the classifier be transformed into the sequence $\hat{y}_1^{(r)}, \ldots, \hat{y}_k^{(r)}$ of binary variables.

For $C_{agg}^{(r)}(x) = 1$, corresponding to $\hat{Y}(x) \in \{1, \ldots, r\}$, one has

$$\hat{y}_1^{(r)}(x) = \cdots = \hat{y}_r^{(r)}(x) = 1/r, \quad \hat{y}_{r+1}^{(r)}(x) = \cdots = \hat{y}_k^{(r)}(x) = 0 \quad .$$

For $C_{agg}^{(r)}(x) = 2$, corresponding to $\hat{Y}(x) \in \{r+1, \ldots, k\}$, one has

$$\hat{y}_1^{(r)}(x) = \cdots = \hat{y}_r^{(r)}(x) = 0, \quad \hat{y}_{r+1}^{(r)}(x) = \cdots = \hat{y}_k^{(r)}(x) = 1/(k-r) \quad .$$

Thus the classifier $C_{agg}^{(r)}$ yields the binary sequence $(1/r) \cdot (1, 1, \ldots, 1, 0, 0, \ldots, 0)$ or $(1/(k-r)) \cdot (0, 0, \ldots, 0, 1, 1, \ldots, 1)$ where the change from 1 to 0 or 0 to 1 is after the r-th component. We divide these sequences by r or $k-r$, respectively, to take into account the different number of categories within each dichotomization. The final classifier is given by the second stage aggregation

$$C_{agg}(x) = \text{argmax}_j \sum_{r=1}^{k-1} \hat{y}_j^{(r)}(x) \quad . \tag{2}$$

Therefore for observation x the prediction is in the class which is favoured by a weighted majority of splits. Moreover, for each split r the 'score'

$$s^{(r)} = \sum_{r=1}^{k-1} \hat{y}_j^{(r)}(x)$$

is a measure of the accuracy of the prediction (after aggregation by bagging or boosting) which may be standardized to $\tilde{s}^{(r)} = s^{(r)} / (\sum_{t=1}^{k-1} s^{(t)})$.

It should be noted that in the second stage aggregation (2) the accuracy of the binary prediction is not used for weighting. Therefore a weighted form is given by

$$C_{agg,w}(x) = \text{argmax}_j \sum_{r=1}^{k-1} PV^{(r)}(x) \hat{y}_j^{(r)}(x)$$

where the weights

$$PV^{(r)}(x) = \frac{\max_j \sum_{m=1}^{M} c_m^{(r)} I(C^{(r)}(x, L_m^{(r)}) = j)}{\sum_{m=1}^{M} c_m^{(r)}}$$

are the binary prediction votes from the first stage. The corresponding scores which include the precision from the first stage aggregation are

$$s_w^{(r)} = \sum_{r=1}^{k-1} PV^{(r)}(x)\hat{y}_j^{(r)}(x)$$

which may be normalized to

$$\tilde{s}_w^{(r)} = \frac{s_w^{(r)}}{\sum_{t=1}^{k-1} s_w^{(t)}} \quad .$$

Alternatively weights in (2) could be used to reflect the substantial importance of splits instead of using the accuracy of prediction.

3.2 Linking bagging and boosting to the ordinal structure

The procedure suggested in Section 3.1 is similar to the one-against-all approaches which are popular in the machine learning community. In these approaches each response class is compared separately to all other classes and the results of the bagging or boosting of the binary class problems are combined at the end (see also Dettling & Bühlmann, 2003).

The distinct separation of the two stages means that for ordered responses the ordering of the response category is not reflected at the first stage, i.e. in the boosting or bagging procedure. Therefore in the following an algorithm is suggested which connects the weights in boosting to the ordered performance of the classifier.

Ordinal Discrete AdaBoost (m-th step)

1. Based on the current weights $w_1, \ldots, w_{n_L}$ classifiers $C^{(r)}(., L_m)$ are built for all dichotomous splits of the ordinal class variable at value r.

2. The learning set is run through each classifier $C^{(r)}(., L_m)$ yielding the information if the i-th observation is predicted into a class higher or lower than r. The results of the classifiers for different split values r are combined by majority vote (2) yielding the aggregated classifier $C(., L_m)$.

3. Let the error indicators now be given by $\epsilon_i = \frac{|C(x_i, L_m) - Y_i|}{k-1}$. Therefore with the same notation for e_m, b_m and c_m as in simple dichotomous Discrete AdaBoost the weights are updated by

$$w_{i,new} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n} w_j \exp(c_m \epsilon_j)} \quad .$$

After M steps the aggregated voting for observation x is obtained by

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \qquad \square$$

The essential difference to the fixed split procedure in Section 3.1 is in step 2 of the algorithm. The aggregation across splits is now incorporated into the boosting procedure. In addition, the error indicators ϵ_i reflect the distance between the prediction and the true class. Alternatives for the standardized distance criterion based on

$$\epsilon_i = \frac{|C(x_i, L_m) - Y_i|}{k-1} \tag{3}$$

are the unstandardized distance

$$\epsilon_i = |C(x_i, L_m) - Y_i| \tag{4}$$

or the squared (standardized) distance

$$\epsilon_i = \left(\frac{C(x_i, L_m) - Y_i}{k-1}\right)^2.$$

Of course also the simple 0-1-error

$$\epsilon_i = 1 - I(C(x_i, L_m) = Y_i) \tag{5}$$

can be used.

While ϵ_i refers to the error for single observations the weighted error $e_m = \sum_{i=1}^{n_L} w_i \epsilon_i$ refers to the total data set. The weights $c_m = \log((1-e_m)/e_m)$ which are based on e_m are essentially constructed for two class problems. For k class problems some adaptation is suggested which depends on the type of error which is used. For the simple 0-1-error (5) in k class problems the error rate that can always be obtained is $(k-1)/k$, simply by classifying all observations into the dominant class. Adaptation is based on

$$c_m = \log\left(\frac{(1-e_m)/(1/k)}{e_m/(1-1/k)}\right) = \log\left(\frac{(1-e_m)(k-1)}{e_m}\right) \quad .$$

Thus for $e_m \to (k-1)/k$ one has $c_m \to 0$. Of course this correction is also relevant for nominal multi class versions of the common AdaBoost algorithms. For the simple distance (4) with range $[0, k-1]$ one computes that if all observations are classified into one of k groups with equal sample sizes the corresponding error (with equal weights $w_i = 1/n_L$) is given by $(k-1)/2$. Thus an appropriate choice for c_m has the familiar form

$$c_m = \log\left(\frac{(1-e_m)/((k-1)/2)}{e_m/((k-1)/2)}\right) = \log\left(\frac{1-e_m}{e_m}\right) \quad .$$

The same value results for the standardized form given in the description of the algorithm.

Since in the same way as for the common Discrete AdaBoost c_m might deteriorate ($c_m \to \infty$), we add a term $1/(n_L(k-1))$ to the denominator of b_m. This guarantees that $c_m = \log((1-e_m)/(e_m + 1/(n_L(k-1))))$ is properly defined.

The construction of an ordinal Real AdaBoost is again based on the dichotomous problem of distinguishing between classes $\{1,\ldots,r\}$ and $\{r+1,\ldots,k\}$. For each dichotomization probabilities $p^{(r)}(x)=\hat{P}(Y\leq r|x)$ are provided by a dichotomous classifier. From these probabilities one obtains a sequence of scores $\hat{y}^{(r)}(x)$ for each class by

$$\hat{y}_1^{(r)}(x)=\cdots=\hat{y}_r^{(r)}(x)=p^{(r)}(x),\quad \hat{y}_{r+1}^{(r)}(x)=\cdots=\hat{y}_k^{(r)}(x)=1-p^{(r)}(x)\quad .$$

For the aggregation across splits one considers the value

$$\hat{y}_j(x)=\sum_{r=1}^{k-1}\hat{y}_j^{(r)}(x) \tag{6}$$

which reflects the strength of prediction in class j. A version of Ordinal Real AdaBoost, that is based on these scores, is given in the following.

Ordinal Real AdaBoost (m-th step)

1. Based on weights $w_1,\ldots,w_{n_L}$ a classifier (6) for ordered classes $1,\ldots,k$ is built.
2. The learning set is run through the classifier yielding scores $\hat{y}_j(x_i)$, $j=1,\ldots,k$. Following the criterion in simple Real AdaBoost a variant for the multi class situation is given by

$$f_j(x_i,L_m)=0.5\cdot\log\frac{\hat{y}_j(x_i)}{\left(\prod_{l\neq j}^{k}\hat{y}_l(x_i)\right)^{\frac{1}{k-1}}}\quad .$$

3. The weights are updated for the next step by

$$w_{i,new}=\frac{w_i\exp(-f_{Y_i}(x_i,L_m))}{\sum_{j=1}^{n_L}w_j\exp(-f_{Y_j}(x_j,L_m))}\quad .$$

After M steps the aggregated voting for observation x is obtained by

$$\mathrm{argmax}_j\left(\sum_{m=1}^{M}f_j(x,L_m)\right)\qquad\qquad\square$$

The sign of $f_j(x_i,L_m)$ depends on the ratio of the score for class j compared to the average score of all other classes. Again in order to ensure existence of $f_j(x_i,L_m)$, $\hat{y}_j(x_i)$ is replaced by $\hat{y}_j(x_i)+1/n_L$.

This algorithm uses the ordinal structure in the aggregation step of the binary classifiers. The criterion for updating the weights is based on misclassification but not on distance to the true class. Therefore the technique is referred to as Ordinal Real AdaBoost using the misclassification criterion. The updating criterion $f_j(x, L_m)$ can also be used for multi class problems in simple nominal Real AdaBoost.

Based on this algorithm a variant may be derived, that takes ordinality into account also in the updating step for the weights. One possible implementation is given by

$$f_j(x_i, L_m) = 0.5 \cdot \log \frac{\hat{y}_j(x_i)}{\frac{1}{\sum_{l \neq j}^{k} d_{il}} \sum_{l \neq j}^{k} d_{il} \hat{y}_l(x_i)}$$

where d_{ij} is the distance between true class and class j for observation i. This criterion still shows a close relationship to Real AdaBoost, but class probabilities are weighted according to the distance between current and true class. Thus high probabilities for classes far away from the true class are penalized stronger than high probabilities for neighbourhood classes. This technique is referred to as Ordinal Real AdaBoost using the distance criterion.

Again the aggregated voting for observation x is obtained after M steps by

$$\text{argmax}_j \left(\sum_{m=1}^{M} f_j(x, L_m) \right) \quad .$$

4 Empirical studies

4.1 Study design

In this section we compare the different ordinal approaches to simpler alternative methods, that either do not use the ordinal information within the data or do not use aggregation techniques. The first one is a simple classification tree, called nominal CART in the following tables. Here we build a tree by means of the deviance criterion and let it grow up to a maximum depth. Afterwards it is pruned on the basis of resubstitution misclassification rates until a fixed tree size (given by the user and dependent on the data set) is reached. An approach which uses the ordering of the classes, but still without bagging or boosting, is to build a tree for every dichotomization separately and aggregate them according to (2). This method is called ordinal CART. In the same way two bagging variants are considered: A nominal approach, where every tree predicts the multi class target variable and the final result is obtained by a majority vote of these predictions and ordinal bagging, in which bagging is applied to fixed splits. The results are aggregated over dichotomizations according to (2) and over the bagging cycles. The nominal methods are considered as baseline for possible improvements by ordinal bagging or ordinal boosting.

In addition we consider eight different boosting versions: The simple nominal multi class Discrete and Real AdaBoost, which do not use the ordinal structure within the data, are used for comparison only. The new methods are the ordinal boosting techniques: On the one hand we distinguish between real and discrete methods, on the other hand between three kinds of aggregation. Fixed split means that every single dichotomization is boosted separately and combined at the end. Ordinal AdaBoost, which combines the results from dichotomizations in every boosting cycle, is based on the simple 0-1-misclassification-criterion (5) or on the standardized distance criterion (3). In all of these methods boosting can be done either by sampling with current weights or by using weighted trees. Since we found better results with random sampling, only these results are shown.

The evaluation of the methods is based on several measures of accuracy. As criterion for the accuracy of the prediction we take the raw misclassification error rate $\frac{1}{n}\sum_{i=1}^{n} 1_{\{y_i \neq \hat{y}_i\}}$. In the case of ordinal class structure measures should take into account that a larger distance is a more severe error than a wrong classification into a neighbour class. Therefore we use two measures which reflect the distance in different ways: The mean absolute value (here called *mean abs*) of the differences $\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$ and the mean squared difference (here called *mean squ*) $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ which penalizes larger differences even harder.

In the measures of accuracy one has to distinguish between resubstitution error and validation (or test) error. Resubstitution means that we build the classification rule by using the whole dataset and measure the quality of the prediction on the same data. Resubstitution error is used to examine how fast misclassification error can be lowered by the different techniques, but because of its bias it is not an appropriate measure for prediction accuracy. Therefore we divide the data set at random into two parts consisting of one respectively two thirds of the observations. From the larger (learning) data set the classification model is built and the observations of the smaller (validation) data set are predicted. We use 50 different random splits into learning and validation set and give the mean over these splits.

When aggregating classifiers one has to choose the number of cycles, which means the number of different classifiers that are combined in one bagging or boosting run. As standard we use a number of 50 cycles in this study. The last parameter that may be chosen is the (fixed) tree size, that is defined as the number of terminal nodes of each tree. The values which have been used are given with the results.

4.2 Datasets

4.2.1 Scapula Data

The scapula data are part of a dissertation (Feistl & Penning, 2001) written at the *Institut für Rechtsmedizin der LMU München*. The aim was to predict age of dead bodies only by means of the scapula. Therefore a lot of measures, implying angles, lengths, descriptions of the surface, etc. were provided. We preselected 15 important covariates to predict age, which was splitted into 8 distinct ordinal classes. Each class covers ten years. The data set consists of 153 complete observations.

4.2.2 Consumer-satisfaction of car-owners

The data are based upon a poll from a german car company and were published in Fahrmeir, Hamerle & Tutz (1996). In 1983 questionnaires were sent to 2000 customers, who had bought a new car approximately three months earlier. The point of interest was the degree of satisfaction, reasons for the particular choice, consumer profile, etc. Participation was voluntary of course. Only 1182 persons answered the questions and after removing forms with missing values only 793 questionnaires remained. Each form contained 46 questions, which resulted in a dataset of 46 covariates with 793 observations each. The degree of satisfaction, which was originally partitioned into 10 ordinal classes, was used as class variable. As many of these classes consist of very few observations, they have been aggregated to only four distinct ordered classes.

4.3 Results

4.3.1 Scapula Data

In the nominal approach we use trees with 15 terminal nodes. This seems necessary for a problem with eight classes and after all 15 covariates. All ordinal approaches are performed with trees of size five, because only two class problems are treated.

The interpretation of Table 1 leads to the following conclusions: In resubstitution all kinds of error measurements are improved by using aggregated classifiers. In comparison to fixed split boosting, which reduces the error to zero, and nominal boosting variants, which get at least close to zero, other ordinal boosting methods perform worse. However, they are still superior to bagging.

The more interesting results are based on validation since here the predictive power of the classifier is tested. Concerning the misclassification error there are only slight differences between the classifiers. However, the more important measures for problems with ordinal classes are the distance measures. Both, the absolute and the squared distances, show similar behaviour, but the distinction between the classifiers is more pronounced for the squared error. The results of CART are improved by all aggregation methods. Especially fixed split boosting, but also some other ordinal boosting methods and ordinal bagging perform very well. For example the mean squared distance 2.365 of the classification tree is reduced to 1.215 by discrete fixed split boosting.

evaluation	method	criterion	misclass	mean abs	mean squ
resubstitution	CART (nominal)		0.418	0.608	1.183
	CART (ordinal)		0.379	0.484	0.784
	bagging (nominal)		0.176	0.327	0.824
	bagging (ordinal)		0.294	0.386	0.621
	boosting (nominal)	discrete	0.046	0.046	0.046
	boosting (nominal)	real	0.026	0.026	0.026
	boosting (ordinal)	discrete (fixed split)	0.000	0.000	0.000
	boosting (ordinal)	real (fixed split)	0.000	0.000	0.000
	boosting (ordinal)	discrete (misclass)	0.118	0.118	0.118
	boosting (ordinal)	real (misclass)	0.373	0.399	0.451
	boosting (ordinal)	discrete (distance)	0.176	0.176	0.176
	boosting (ordinal)	real (distance)	0.405	0.418	0.444
test	CART (nominal)		0.676	1.085	2.365
	CART (ordinal)		0.652	0.995	2.112
	bagging (nominal)		0.663	0.982	1.925
	bagging (ordinal)		0.628	0.828	1.375
	boosting (nominal)	discrete	0.649	0.932	1.747
	boosting (nominal)	real	0.646	0.904	1.619
	boosting (ordinal)	discrete (fixed split)	0.629	0.799	1.215
	boosting (ordinal)	real (fixed split)	0.646	0.818	1.244
	boosting (ordinal)	discrete (misclass)	0.635	0.884	1.633
	boosting (ordinal)	real (misclass)	0.681	0.864	1.300
	boosting (ordinal)	discrete (distance)	0.611	0.841	1.473
	boosting (ordinal)	real (distance)	0.691	0.878	1.330

Table 1: Resubstitution and test error for scapula data

Figures 1 to 3 show the performance of four selected classification techniques. While the y-axis describes one of the three measures of accuracy, the x-axis marks the number of computed cycles. These figures show the development of the predictive power across runs of the algorithms. Again one sees that there is not much difference in the values of the misclassification error and that especially fixed split boosting shows dominating performance according to distance measures.

Figures 4 to 7 again show the performance of the same four selected methods. Here the focus is on single observations of the data set. On the x-axis the observations are given ordered by class label. The y-axis shows the median of the predicted class over all 50 different partitions into learning and validation set. Often the large differences between prediction and true value are found for the same observations. This shows that there are some observations that are particularly hard to classify.

4.3.2 Consumer-satisfaction of car-owners

In the nominal approach we use trees with 10 terminal nodes as this classification problem copes with only four classes. All ordinal approaches are performed with trees of size five, because only two class problems are treated.

Table 2 can be interpreted as follows: In the same way as for the scapula data all kinds of error measurements concerning resubstitution are improved

evaluation	method	criterion	misclass	mean abs	mean squ
resubstitution	CART (nominal)		0.632	0.803	1.146
	CART (ordinal)		0.573	0.691	0.928
	bagging (nominal)		0.520	0.646	0.903
	bagging (ordinal)		0.517	0.585	0.721
	boosting (nominal)	discrete	0.266	0.346	0.507
	boosting (nominal)	real	0.270	0.333	0.464
	boosting (ordinal)	discrete (fixed split)	0.240	0.253	0.281
	boosting (ordinal)	real (fixed split)	0.102	0.106	0.113
	boosting (ordinal)	discrete (misclass)	0.494	0.541	0.634
	boosting (ordinal)	real (misclass)	0.575	0.579	0.586
	boosting (ordinal)	discrete (distance)	0.415	0.453	0.528
	boosting (ordinal)	real (distance)	0.588	0.595	0.610
test	CART (nominal)		0.653	0.858	1.292
	CART (ordinal)		0.635	0.787	1.107
	bagging (nominal)		0.626	0.797	1.148
	bagging (ordinal)		0.626	0.725	0.925
	boosting (nominal)	discrete	0.621	0.810	1.213
	boosting (nominal)	real	0.611	0.786	1.154
	boosting (ordinal)	discrete (fixed split)	0.613	0.769	1.105
	boosting (ordinal)	real (fixed split)	0.623	0.808	1.212
	boosting (ordinal)	discrete (misclass)	0.614	0.737	0.992
	boosting (ordinal)	real (misclass)	0.628	0.714	0.886
	boosting (ordinal)	discrete (distance)	0.609	0.799	1.217
	boosting (ordinal)	real (distance)	0.630	0.720	0.899

Table 2: Resubstitution and test error for consumer-satisfaction of car-owners

by using aggregated classifiers. Especially fixed split boosting reduces the error close to zero, while other ordinal boosting methods perform worse. However, according to distance measures they are still superior to bagging. All in all the improvement by using aggregated classifiers is not as convincing as for the scapula data.

The validation results, which are better indicators for the predictive power of the classifier, show similar effects as for the scapula data. The most important difference between the results concerning both data sets is the fact, that different ordinal methods seem to dominate. The misclassification error shows only slight differences between the classifiers and no superior technique can be pointed out. Concerning the more important measures of distance, the results of CART are improved at least a bit by all aggregation methods. But fixed split boosting, which led to optimal results for the scapula data, is by far not the best procedure here. Instead especially real variants of Ordinal AdaBoost perform very well, just as ordinal bagging.

5 Concluding remarks

The concept to combine aggregating classifiers with techniques for ordinal data structure led to new methods we refer to as Ordinal AdaBoost. Different updating criteria and discrete as well as real valued boosting methods can be used. Another suggestion is to boost dichotomizations only and combine these single results, which we call fixed split boosting. Like boosting also bagging can be combined with ordinal structure. All in all many different combinations are derived and compared in terms of their accuracy, which is measured by raw misclassification error but also by distance measures in order to take into account the ordinal structure.

We found promising results for two empirical data sets, since all ordinal techniques improve the performance of a simple CART tree. There seems to be no dominating method as for different data sets the best results occur by different methods. Although ordinal bagging shows satisfying results for both data sets, it is outperformed by at least one of the ordinal boosting techniques. While in one data set fixed split boosting led to the best results, the other data set shows superior performance of Ordinal AdaBoost.

These findings suggest that further research seems to be a worthwhile task, as variations of these methods are still possible. Especially the transfer of Real AdaBoost from nominal to ordinal data structure is arbitrary and other criteria might be used.

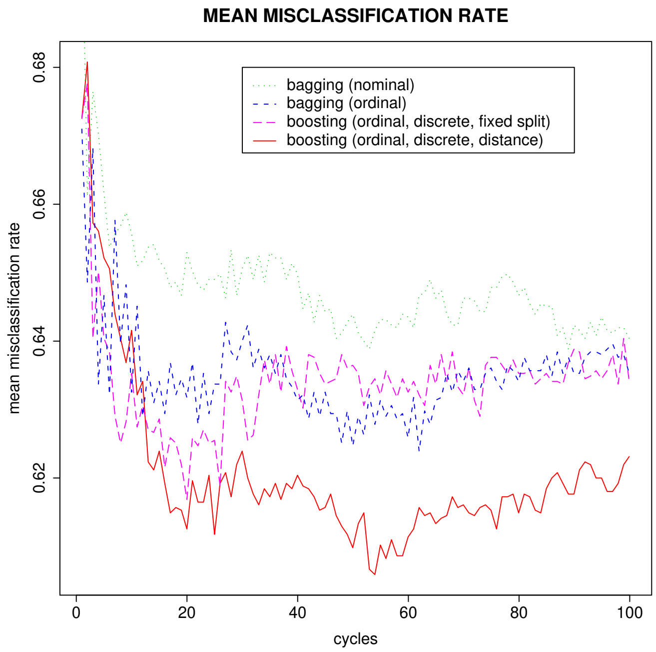

Figure 1: Mean misclassification rate (test error) across cycles for four selected classification techniques

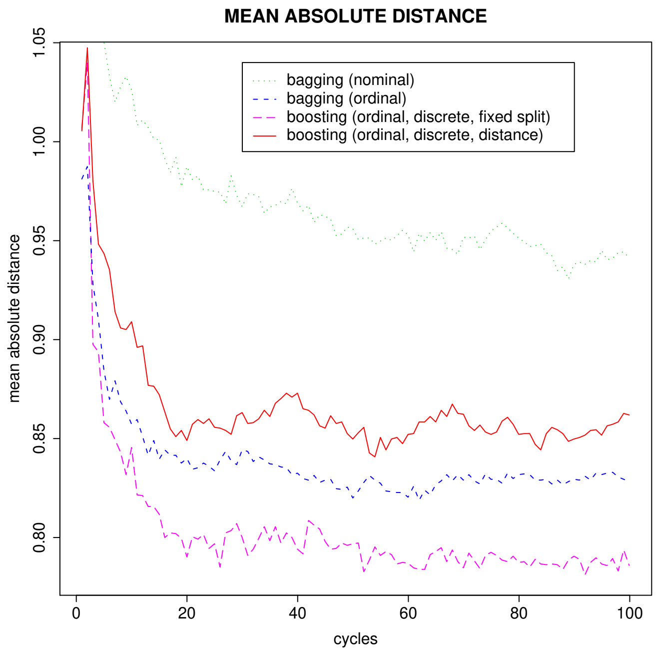

Figure 2: Mean absolute distance (test error) across cycles for four selected classification techniques

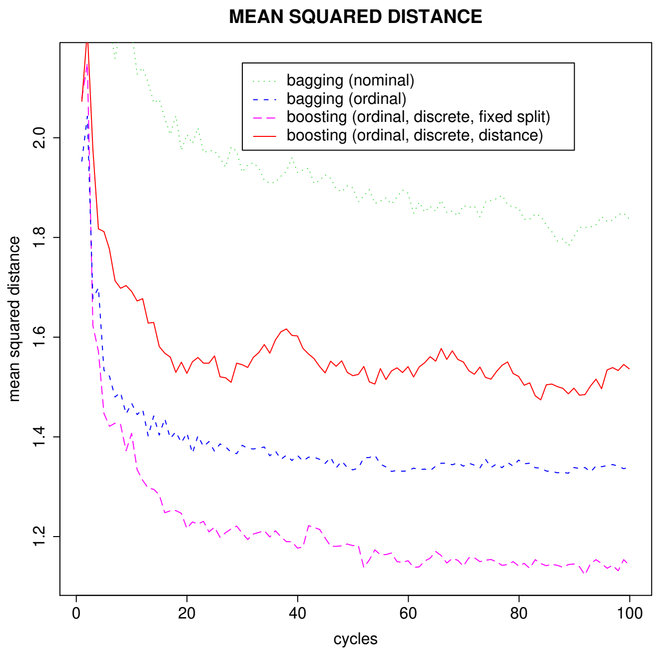

Figure 3: Mean squared distance (test error) across cycles for four selected classification techniques

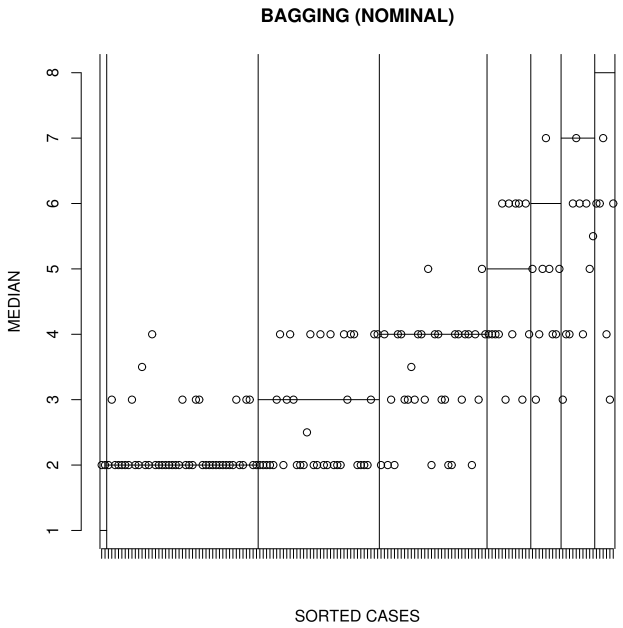

Figure 4: Median of predicted class for every observation of the data set (vertical lines distinguish between classes, horizontal lines show the true class within limits; applied method: nominal bagging)

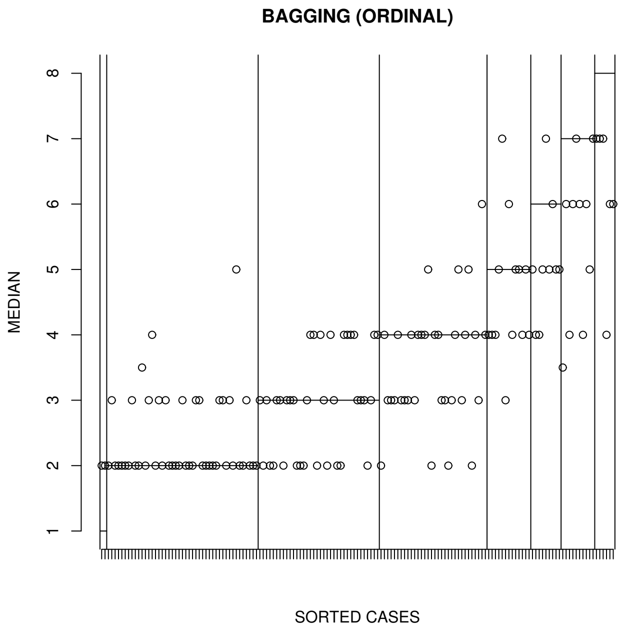

Figure 5: Median of predicted class for every observation of the data set (vertical lines distinguish between classes, horizontal lines show the true class within limits; applied method: ordinal bagging)

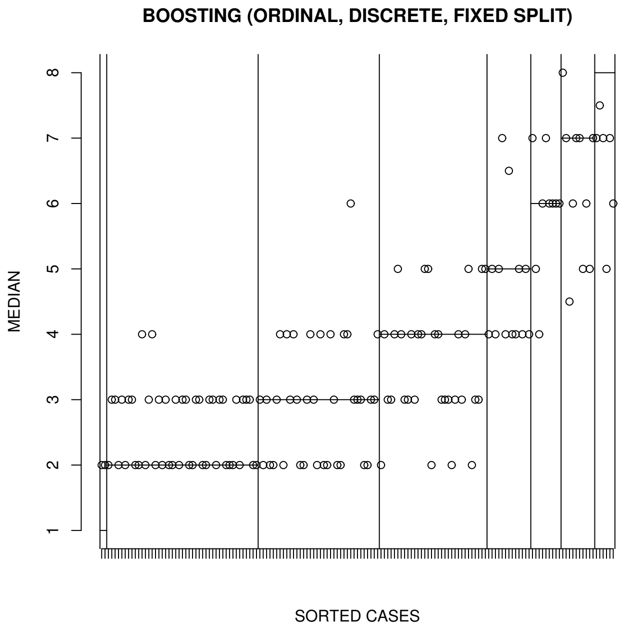

Figure 6: Median of predicted class for every observation of the data set (vertical lines distinguish between classes, horizontal lines show the true class within limits; applied method: discrete fixed split boosting)

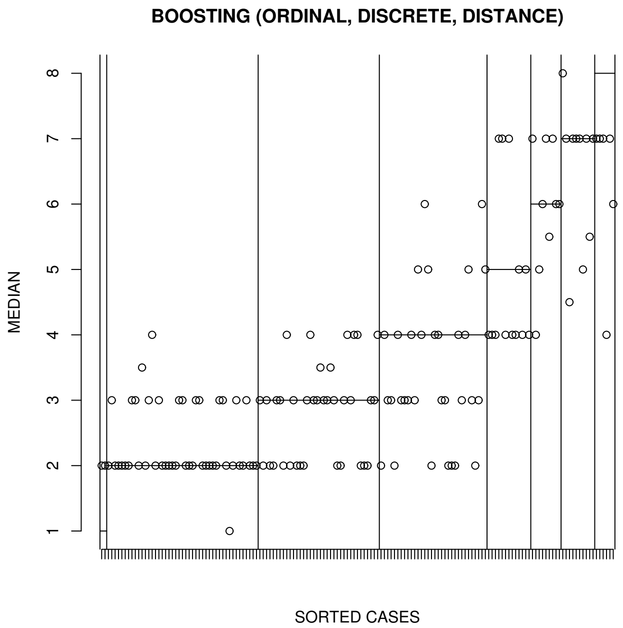

Figure 7: Median of predicted class for every observation of the data set (vertical lines distinguish between classes, horizontal lines show the true class within limits; applied method: Ordinal Discrete AdaBoost using distance criterion)

References

Anderson, J. and Philips, P. (1981). Regression, discrimination and measurement models for ordered categorical variables. *Applied Statistics* **30**, 22–31.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning* **36**, 105–142.

Bühlmann, P. (2002). Consistency for l_2-boosting and matching pursuit with trees and tree-type basis functions. Research Report 109, ETH Zürich.

Bühlmann, P. (2003). Bagging, subagging and bragging for improving some prediction algorithms. To appear in: Recent Advances and Trends in Nonparametric Statistics.

Bühlmann, P. and Yu, B. (2002a). Analyzing bagging. *Annals of Statistics* **30**, 927–961.

Bühlmann, P. and Yu, B. (2002b). Boosting with the l_2-loss: Regression and classification. To appear in: Journal of the American Statistical Association.

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

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics* **26**, 801–849.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Boca Raton - London - New York - Washington D.C.: Chapman & Hall.

Clark, L. and Pregibon, D. (1992). Tree-based models. In J. Chambers & T. Hastie (Eds.), *Statistical Models in S*, pp. 377–419. Pacific Grove: Wadsworth & Brooks.

Dettling, M. and Bühlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics* **19**, 1061–1069.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning* **40**, 139–157.

Dudoit, S., Fridlyand, J., and Speed, T. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association* **97**, 77–87.

Fahrmeir, L., Hamerle, A., and Tutz, G. (1996). *Multivariate statistische Verfahren*. Berlin - New York: de Gruyter.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information And Computation* **121**, 256–285.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. Machine Learning: Proceedings of the Thirteenth International Conference.

Friedman, J. (1999). Stochastic gradient boosting. Technical Report, Stanford University.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. Technical Report, Stanford University.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. Technical Report, Stanford University.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 337–374.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. New York: Springer.

Hothorn, T. and Lausen, B. (2003). Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition* **36**, 1303–1309.

McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge: University Press.

Rudolfer, S., Watson, P., and Lesaffre, E. (1995). Are ordinal models useful for classification? a revised analysis. *Journal of Statistical Computation and Simulation* **52**, 105–132.

Schapire, R. (2002). The boosting approach to machine learning: An overview. MSRI Workshop on Nonlinear Estimation and Classification.

Schapire, R., Freund, Y., Bartlett, P., and Lee, W. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* **26**, 1651–1686.

Tutz, G. (2000). *Die Analyse kategorialer Datensätze*. München: Oldenbourg.