CLASSIFICATION CONFIDENCE WEIGHTED MAJORITY VOTING USING DECISION TREE CLASSIFIERS

NORBERT TÓTH
Department of Measurement and Information Systems,
University of Technology and Economics,
Magyar tudósok körútja 2., Budapest, H-1117, Hungary
ntoth@mit.bme.hu
http://www.mit.bme.hu/~ntoth

BELA PATAKI
Department of Measurement and Information Systems,
University of Technology and Economics,
Magyar tudósok körútja 2., Budapest, H-1117, Hungary
pataki@mit.bme.hu
http://www.mit.bme.hu/~pataki

Received (Day Month Year)
Revised (Day Month Year)

In this paper a novel method is proposed to combine decision tree classifiers using calculated classification confidence values. This confidence in the classification is based on distance calculation to the relevant decision boundary. It is shown that these values – provided by individual classification trees – can be integrated to derive a consensus decision. The proposed combination scheme – confidence weighted majority voting – possesses attractive features compared to other approaches. There is no need for an auxiliary combiner or gating network, like in the Mixture of Experts structure and the method is not limited to decision trees with axis-parallel splits; it is applicable to any kind of classifiers that use hyperplanes to cluster the input space.

Keywords: decision tree, classification confidence, quadratic programming, density estimation, Bayes rule, confidence interval, combining, ensemble

1. Introduction

Every classifier system can be modeled as a black box whose output is a canonical variable – dependent on the input variables – that represents the class of the input sample. A classification or decision tree is a classifier system that divides the input space into a certain number of sections. These sections have their class label according to the leaf that defines the actual section. If the input vector of the predictor variables falls into a section, the corresponding class label is returned.

Classifier systems most often evaluated in the machine learning literature by their average misclassification or error rate. Classifiers that classify samples more
correctly, have smaller misclassification rate, hence perform better, have better accuracy. This misclassification rate is usually measured on a test set, resulting in an average value, which is a global measure of uncertainty of the classification for every sample in the whole input space.

However many applications – such as a medical decision support system – require not only the cases to be classified, they must be ranked or compared to each other or to an absolute confidence scale. The original idea to rank the cases or provide confidences is to look at the class membership probabilities. In this sense a classifier not only must classify a case it must provide class probability estimates also (or other certainty information that can be used for ranking purposes).

Decision trees [Breiman et al. (1984)] are one of the most widespread used classification models. They possess attractive features, such as good accuracy, comprehensibility and high efficiency in high dimensions and on large datasets. Despite all of these favorable properties the standard form of decision tree classifiers do not provide ranking information, while other classifier models, like neural networks or density estimator based classifiers do.

This first part of this paper presents a method to turn classification or decision trees into probability estimation trees. The second part of the manuscript discusses a method to combine decision tree classifiers based on the calculated classification confidence value.

1.1. Probability Estimation Trees (PETs)

There are numerous approaches to extend the decision tree framework to provide ranking information [Ling and Yan (2003)], [Alvarez et al. (2007)]. The very obvious approach to turn decision trees into probability estimators by using the absolute class frequencies at every leaf:

\[ p_i = \frac{n_i}{N}, \]

where \( n_i \) is the number of the observed samples of class \( c_i \) at a leaf and \( N \) is the total number of samples falling from the test set to the leaf. Although this simple transformation turns a decision tree into a probability estimator tree, the probability estimates will be poor compared too more advanced approaches [Ling and Yan (2003)].

Approaches that aim the improvement of PETs [Ferri et al. (2003)] can be categorized into two groups. The first bunch of methods alter the growth process of the tree to make them more suitable for probability estimation [Provost and Domingos (2003)]. They define different splitting criteria, or pruning techniques. The second group of approaches, that try to obtain better probability estimates without altering the tree, are in focus of this paper. Three methods will be addressed: smoothing techniques, geometric ranking, density estimation at the leafs.

One way to improve the probability estimates provided by decision trees is the use of smoothing techniques [Provost and Domingos (2003)], [Ferri et al. (2003)],
which is one of the most widespread used method to create PETs. Other approaches try to estimate conditional class probabilities, considering the location of input samples at the leaves.

Mostly the Laplace smoothing correction is used, where the class probability estimates take the form of:

$$p_i = \frac{n_i + 1}{N + C},$$

(2)

where $C$ is the number of classes. The use of Laplace-correction incorporates a $\frac{1}{C}$ prior probability for all classes into the probability estimation (with zero examples all the class probabilities will equal $\frac{1}{C}$), which "pulls" the estimated class probabilities towards $\frac{1}{C}$.

Other approaches try to estimate conditional class probabilities, considering the location of input samples at the leaves.

[Smyth et al. (1995)] proposed the use of kernel density estimation at the leafs of the decision tree to provide probability estimates. They used trees with larger leafs containing more sample points. The major drawback of this approach is that it suffers from the main disadvantage of the kernel density estimator, referred as the "curse-of-dimensionality" [Duda et al. (2001)], which might render the method useless in high dimensions or without a huge number test samples (to estimate the densities).

[Alvarez and Bernard (2005)] proposed a geometric ranking method that is based on the distance calculation of the input sample to the decision boundary induced by the decision tree. Later [Alvarez et al. (2007)] combined the geometric ranking method with the kernel density estimation to provide class probability estimates and at the same time avoiding the "curse-of-dimensionality" coming from the nature of kernel density estimators. They showed that geometric ranking combined with kernel density estimation can outperform other methods of probability estimation.

Their geometric ranking method is limited to axis-parallel decision trees and distance metrics that keep the minimal distance projections invariant to the Euclidean metric, meaning that the minimal distance projection is the same, only the distance value changes, which is not true for example in the case of the widely used Mahalanobis distance metric.

2. Improved Classification Confidence Estimation Using Decision Trees

An improved method is presented in this paper to extend the decision tree framework to provide individual certainty information to all the samples classified. It is based on distance calculation to the decision boundary, density estimation based on the distances and classification confidence calculation. The novelties of the algorithm are summarized in the following points:

- The method is not limited to axis-parallel trees; it can be applied to oblique trees also.
• The method is not limited to distance metrics where the minimal distance projections are invariant, it allows the use of distance metrics – such as the Mahalanobis metric – where the minimal distance projections are not invariant to the selection of the metric.
• Using Bayesian computation the conditional classification probability is estimated.
• Conditional confidence bounds for the estimated conditional classification probabilities are calculated.

2.1. Euclidean Distance from the Decision Boundary

The solution proposed here is applicable to any kind of decision trees, whose decisions at its nodes are linear inequalities defined by hyperplanes in the form of:

$$w^T x \leq b,$$

where $x, w \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Every leaf of the decision tree determine a region in the input space, a polyhedron. This polyhedron is defined as the solution set of a finite number of inequalities that form the decisions of the tree along the decision making path from the root node to leaf node.

The leaves of the decision tree cover the complete input space with polyhedrons. Each of these polyhedrons represents a region with an assigned class label, which is returned if the input point falls into that particular cell or equivalently ”end up” in that leaf.

To get the (Euclidean) distance from the decision boundary we have to measure the shortest distance to the closest section – leaf – with different class label. This
Fig. 2. A two dimensional, two class classification problem. Distance is calculated from the decision boundary induced by an oblique decision tree.

The problem could be tackled by solving a set of constrained quadratic programs [Boyd and Vandenberghe (2001)], one for each leaf with different class label. An input point’s (s) (Euclidean) distance from a different leaf (polyhedron defined by a set of inequalities) $P = \{x|Ax \leq b\}$ in $R^n$ is defined as $\text{dist}(P,s) = \inf\{||x-s||_2 | x \in P\}$. To find the distance of $s$ from $P$, the following quadratic program must be solved:

$$
\begin{align*}
\text{minimize} & \quad f_0 = \frac{1}{2}||x-s||_2^2 \\
\text{subject to} & \quad Ax \leq b,
\end{align*}
$$

where $s$ is constant (input point’s coordinates), $E$ is the identity matrix of appropriate dimension. The matrix $A$ and the vector $b$ contains the constraints (hyperplane parameters) collected from the tree’s decision nodes on the path from the root node to leaf node. The solution of the quadratic program is illustrated in figure 1. $x_0$ is the closest point to the input sample $s$ in $P$, hence the distance between $s$ and $P$ equals the distance between $s$ and $x_0$.

The algorithm is demonstrated on a two dimensional dataset – containing 2 classes – classified by an oblique decision tree. The distance from the decision boundary is visualized as a meshgrid in figure 2.

2.2. Mahalanobis Distance from the Decision Boundary

Using the method proposed above – to calculate the distance from the decision boundary induced by a decision tree – allows us to use different distance metrics rather than the Euclidean metric, such as the Mahalanobis metric. These metrics are not limited to those where the minimal distance projection is invariant. This means that the minimal distance point in a polyhedron to a point $s$, which is the minimal distance projection of $s$ to $P$, is different for the Euclidean and the Mahalanobis metric. The Mahalanobis distance was introduced by P. C. Mahalanobis [Mahalanobis (1936)],[Duda et al. (2001)]. It differs from the Euclidean distance in that it takes into account the correlations among the input variables, and is scale invariant. The Mahalanobis distance between two random vectors $x$ and $y$ with the same distribution and the covariance matrix $\Sigma$:

$$
\text{dist}(x,y) = \sqrt{\text{tr}(\Sigma^{-1}(x-y)(x-y)^T)}
$$

where $\Sigma$ is the covariance matrix.
Fig. 3. Two class classification and a decision tree defining four hyperplanes (thick lines). Distance is calculated from the decision boundary and the contour levels are illustrated with thin lines. Euclidean distance to the left, Mahalanobis distance to the right.

If the covariance matrix is the identity matrix the Mahalanobis distance reduces to the Euclidean distance. If the covariance matrix is diagonal the Mahalanobis distance is the normalized Euclidean distance between the vectors.

Using the Mahalanobis distance as a distance metric when calculating the classification certainty comes from the hypothesis that in directions of greater deviation greater Euclidean distance is required for the same certainty of the classification. This is illustrated in figure 3. The “square class” data points show greater deviation in the horizontal axis than the vertical axis. This indicates that we need a greater distance from decision boundaries in that direction for the same certainty.

Using Mahalanobis distance means that in greater deviation directions (in this case the horizontal axis) the distance from the decision boundaries will raise slower than in lower deviation directions (in this case the vertical axis). Figure 3 illustrates that the use of the Mahalanobis distance results in more concentrated contour lines in the center. Another difference is that in Mahalanobis sense there is only one point that has maximal distance from the decision boundary, while in Euclidean sense more points (a central horizontal line) is at maximal distance from the decision boundaries. Figure 4 shows that the gradient of the distance function is smaller along the horizontal axis than the vertical axis using Mahalanobis distance instead of Euclidean distance. Using Euclidean distance the gradient is the same in both – in fact all the – directions.

However the method proposed above to calculate the shortest distance to the decision boundary allows the use of the Mahalanobis distance metric, it requires a covariance matrix. There are different options for choosing or determining the matrix, each with its own explanation and hypothesis. Possible choices might be:

1. During the tree building process estimate a global covariance matrix from the training data, including all the data from different classes. When calculating an
input sample’s distance from the decision boundary use this global covariance matrix.

(2) During the tree building process estimate a different covariance matrix from the training data for each input class. When calculating an input sample’s distance from the decision boundary, first use the tree to classify the sample, and then use the corresponding covariance matrix that belongs to that particular class to calculate the distance.

(3) After the tree building process estimate a locally global covariance matrix for each of the tree’s leaves using all the training data that fall into that leaf. When calculating an input sample’s distance from the decision boundary first determine the leaf that it belongs to, and then use the covariance matrix belonging to that particular leaf.

(4) After the tree building process estimate a local covariance matrix for each class at each of the tree’s leaves. When calculating an input sample’s distance from the decision boundary determine the leaf, and the class and then use the appropriate covariance matrix.

Each approach has its advantages and disadvantages. It is the subject of further research which method to use, but "it seems" that there is no global optimum, each approach can be superior to the others in certain conditions, it depends on the structure of the data and tree itself. In the example above the second method was used, at the tree building different covariances were estimated for the two different class data.

2.3. Class Density Estimates

The previously proposed algorithm provides us the minimum distance values from the relevant decision boundary for the input samples. It is a natural hypothesis to assume that the samples farther away from the decision boundary are classified by greater confidence in contrast to those that are close to the decision boundary also noted by Alvarez [Alvarez and Bernard (2005)]. In other words we expect the majority of the classification errors to occur in the proximity of the classification
boundary. With the distance values we can rank a set of cases, saying which has a greater classification confidence. However we are interested in the classification confidence value for each individual input sample.

To get class probability estimates we can use a set of training samples and density estimation [Silverman (1986)] to approximate the probability density of the true and false classification based on the sample's distance calculated from the decision boundary. The proposed method uses kernel density estimation [Silverman (1986)] with gaussian kernel. The bandwidth is estimated using Silverman’s “rule of thumb” [Silverman (1986)]. The zero distance boundary conditions are handled with the reflection technique [Silverman (1986)].

\[ h = \left( \frac{4\hat{\sigma}^5}{3n} \right)^{-\frac{1}{5}}, \]  

(6)

where \( n \) is the number data samples and \( \hat{\sigma} \) is the estimated standard deviation of the data.

In figure 5 the class-conditional probability densities are shown for two leaves of a decision tree. These densities are multiplied by the prior probabilities – which are data estimated – of the classes at the particular leaf. It can be seen in the figure that the leaf to the left provide inferior classification performance compared to the leaf to the right. It has the class-conditional probability densities ”closer to each other” which results in increased uncertainty (if the class-conditional densities equal at a certain distance it means both classes are equally probable). The leaf to the right has a greater classification confidence because class B’s density is far bigger than class A’s at all distances.

2.4. Conditional Classification Probability Based on Bayesian Computation

Since we have got the class (conditional) probability density estimates (for each leaf), now the conditional probability of the class of the input sample – based on its distance from the decision boundary – can be computed with a Bayesian computation [Duda et al. (2001)]. The Bayes formula can be easily adapted to true (correct) and false classification probabilities. No matter how many classes we have, we can measure the misclassification rate on a training set counting the points correctly classified and incorrectly classified. In this case we derive two classes (from all the classes) when we measure the performance of the decision tree. One class contains the correctly classified points and the other contains the incorrectly classified points. We can estimate priors for these from the training data. The Bayes formula then reduces to the form:

\[ P(true|D) = \frac{p(D|true)P(true)}{p(D|true)P(true) + p(D|false)P(false)}, \]  

(7)
Fig. 5. Class conditional density estimates at two leaves. The first leaf (central area, yellow) contains mostly class A samples, the second (right half pane, green) contains mostly class B points.

where

\[
P(\text{true}) = \frac{\text{number of correctly classified points}}{\text{total number of data points}} \\
P(\text{false}) = 1 - P(\text{true}).
\]  

(8)

\(D\) is the input point’s distance from it’s relevant decision boundary, \(p(D|\text{true})\) and \(p(D|\text{false})\) are the conditional density estimates of the correct and false classification. \(P(\text{true}|D)\) can be calculated globally (using all the input samples’ distances) or locally for every leaf of the tree (using separate data at each leaf). This way more accurate classification certainty information can be obtained.

In figure 6 to the left the true and false conditional classification densities of a decision tree are shown. The conditional Bayes probability of the correct classification is calculated using formula 7. It can be seen that the probability of the correct classification is increasing with the distance, since misclassifications occur near the decision boundary.
2.5. Confidence Interval for the Conditional Classification Probability

When evaluating the performance of classifiers we are curious about the error (misclassification) rate or the true (correct) classification rate. If a classifier’s true (correct), but unknown, classification rate is $p$ and if $k$ of the $n$ test samples are correctly classified, than $k$ has the binomial distribution. The fraction of the test samples correctly classified $\hat{p}$ is the maximum-likelihood estimate for $p$.

$$P(k) = \binom{n}{k} p^k (1 - p)^{n-k}. \quad (9)$$

The fraction of the test samples correctly classified $\hat{p}$ is the maximum-likelihood estimate for $p$.

$$\hat{p} = \frac{k}{n}. \quad (10)$$

For the binomial proportion $p$ confidence intervals can be calculated using the $F$ distribution. Define the 100(1-$\alpha$)% confidence interval for $p$ as $(\Phi, \Psi)$. The lower bound can be calculated as:

$$\Phi = \begin{cases} 0 & \text{if } k = 0 \\ \frac{1}{k + (n-k+1)F_{2(n-k+1),2k, 1-\alpha/2}} & \text{if } k \neq 0. \quad (11) \end{cases}$$

The higher bound can be calculated as:

$$\Psi = \begin{cases} 1 & \text{if } k = n \\ \frac{(k+1)F_{2(k+1),2(n-k),1-\alpha/2}}{n-k+(k+1)F_{2(k+1),2(n-k),1-\alpha/2}} & \text{if } k \neq n. \quad (12) \end{cases}$$

The width of the confidence interval is a function of $n$, $k$, and $\alpha$. The significance level $\alpha$ depends on our choice. The smaller we choose, the wider the confidence interval will be. $n$ is the number of the test samples. Increasing $n$ will narrow the
confidence interval. $k$ is the number of samples correctly classified, it depends on $n$ and follows a binomial distribution. $k$ characterizes the classifier, the closer it gets to $n$, the better classifier performs.

Confidence intervals for the Bayes classification probabilities for each test sample can also be constructed. The method proposed in this paper provides the empirical classification performance $\hat{p}$, but this classification probability is a function of the distance from the decision boundary: $D$. To calculate the confidence interval at any distance, an approximation of the number of test samples $n(D)$ must be obtained at the distance $D$. This is a problem, because $D$ is a continuous variable, and since only a limited number of test samples are available, for most of the values of $D$ the number of samples will most probably equal 0. $D$ can be approximated from the densities of the test data points:

$$n(D) = (\# x_i \text{ in } [x-h, x+h]) \approx \sum_{i=0}^{n} K \left( \frac{D - D_i}{h} \right) \approx nh\hat{f}(D),$$

where $h$ is from Silverman's rule of thumb [Silverman (1986)]. Using $n(D)$ now it is possible to construct confidence intervals for every approximated classification performance $\hat{p}$ at distance $D$.

Figure 6 to the right shows the 95% confidence interval for a sample decision tree. It can be seen in the figure that the confidence interval narrows as we move farther from the decision boundary because of the number of test samples at those distances increases and hypothetical probability of the correct classification also increases. At a certain distance – around $D = 0.06$ – the confidence interval starts to drop. Its explanation is that the hypothetical probability of the correct classification reaches its maximum at one ($\hat{p} = 1$), and the number of test samples are decreasing with the distance from around $D = 0.05$.

### 3. Confidence Weighted Majority Voting to Combine the Classifiers

In classification problems the ultimate goal is to create a classifier or classifier system that achieves the best performance for the application at hand. The traditional approach was to create or train different kind of classifiers (or the same kind with different parameters) and select the one that performs the best. During the evaluation of the trained classifiers for the same classification task it has been observed that however they provide different accuracies, the set of patterns misclassified might not necessarily overlap [Kittler et al. (1998)]. This suggested that different classifiers provide different information about the patterns to be classified, which if somehow could be integrated, a classifier system could be created that outperforms the best individual classifier in the trained classifier set. Such classifier system can be created using bagging, which is in focus later in this paper.

Another use of classifier ensembles is the decomposition of the training data into – not necessarily separate – subsets. During training for every subset unique
classifier is trained. In the classification process an input point is classified using all the classifiers and the results of the individual experts are aggregated according to specific rules. Such classifier system is the Mixtures of Experts structure (MOE) [Jordan (1994)], evaluated later in this paper.

3.1. Combining decision tree classifiers using classification confidence.

The lower classification confidence value calculated for every input sample – discussed in the previous sections – can be used to form classifier ensembles containing decision trees that provide this value in addition to the predicted class. The proposed method:

1. Classify every input sample using all the decision trees.
2. Calculate the input sample’s distance from every classifier’s relevant decision boundary.
3. Using the distance from the decision boundaries, get the the lower classification confidence value for every classifier.
4. Use the lower classification confidence value to weigh the classifiers output, and do weighted majority voting.
5. If all the classifier’s classification confidence is zero, then use the classifier whose decision boundary is the closest to the input sample.

Step five is necessary to ensure the classification of input points far from every classifier, where no training data were available. In this case all confidences would be zero and no final output class could be created. The hypothesis behind this step is that points far from every classifier could be best classified by the closest one.

3.2. An Illustrative Example

In this illustrative example two different datasets were integrated forming one complex classification problem, a puzzle and a ring (figure 7). It will be shown how two different classification trees trained each for one of the subproblems could be combined using the proposed method. The two datasets are located in different input space regions. For each dataset a unique decision tree was trained and the distance–confidence profiles were estimated – as described in the previous sections – characteristic to the classifiers. Figure 8 to left shows the confidence values of the classifiers in the input space. It can be seen in the figure that the confidence drops in the proximity of the decision boundaries and at far distances from the classifiers where were only a few training points available. The final aggregated output of the created classifier system is shown in figure 8 to the right. Misclassifications – marked with stars – occur at two locations:

1. Around the boundaries of each classifier system.
2. Where the classification confidence of the two classifiers are around equal.
Fig. 7. Two datasets, and two decision trees grown, one for each. The two datasets are located in different input space regions. The solid lines represent the decision hyperplanes of the two sub classifiers.

Fig. 8. Confidence weighted majority voting combination of two classifiers. The confidence levels are shown to the left, and the classified data to the right. Stars represent the misclassifications.

The proposed method is applicable to cases whether the classifier regions overlap or not. Both scenarios are evaluated in the following sections.

3.3. Why Classification Accuracy is Inappropriate to Use to Form the Weights of the Classifiers

In the algorithm the lower classification confidence is used as weights for the voting system. It might also be a natural assumption to use the correct classification probability. The reasoning behind the presented approach is that using the confidence value instead of the correct classification probability eliminates the following difficulties:

(1) At a certain distance the confidence value is a function of the number of data
samples that were available during training. This value incorporates more information than the correct classification probability. The more training samples available to confidence value will be more closer to the correct classification probability. If only a few samples are available at a specific distance, the lower classification confidence will be lower, which means that the estimated correct classification probability is not "that reliable".

(2) At higher distances from the decision boundary the estimated correct classification probability is likely to be one. This is a natural assumption, because decision boundaries are put between samples with different classes. Consider the scenario when two classifiers are combined and each operates in a different input space region. If an input sample falls into the region of one of the classifiers, than the other classifier will estimate a correct classification probability of one, while the other – whose region the sample fell – will estimate a correct classification probably less than one. The problem is that the other classifier will suppress the classifier – whose region the data sample fell – during voting, however its correct classification probability is totally unreliable, because it did not have any training data samples at this distance. Using confidence values this difficulty can be overcome, because if the data samples are low during training the calculated lower confidence value will depart from the estimated correct classification value as expected.
Fig. 10. Artificial region of competence boundary induced around the training data. If an input sample falls outside this boundary than the sample’s distance from the decision boundary is defined as the maximum distance from decision boundary and from the region of competence boundary.

3.4. The Problem with Decision Boundaries that Extend Outside of Region of Competence

Classifiers are trained on training data, hence they are supposed to operate in that particular input region the training data occupies (region of competence). The proposed method that enables classifiers to calculate the classification confidence for every input sample is a self-calculated value that the classifier provides to state how certain is the predicted classification. This value is a function of the number of training points available and the achieved accuracy of the classifier at a specific distance from the decision boundary.

This value has a correct, interpretable meaning in the region of competence of the classifiers. However classifier decision boundaries might extend outside from the region of competence and in that case this classification confidence value loses it’s meaning. This phenomenon is coming from the fact that this value is function of the classifier accuracy and the available training data points only through a one dimensional projection onto the decision boundary. Classifier decision boundaries that extend outside the region of competence of the classifier can induce false classification certainties in regions where were no training data points at all.

Figure 10 shows the puzzle classification problem and the decision boundary of a decision tree. The decision boundary extends outside the region of the training samples. This means that for input samples that fall far away from the training samples, but close to the decision boundary would get a small calculated distance value (figure 11, left image). Since classification confidence depends on the calculated distances, these samples would be classified with high confidence (figure 11, left image), although they are outliers. There were no training data in their region,
therefore classification from a tree that was grown in a different region should not be trusted.

We can overcome the problem of extending decision boundaries with a correction after the distance from the decision boundary is calculated. In this solution an artificial decision boundary (figure 10) is created around the training data during training. Later when the distance of an input sample is calculated from the decision boundary, and the sample is outside decision boundary, the distance is defined as the maximum of the distance from the decision boundary and the distance from the artificially induced region of competence boundary:

\[ D(x) = \max\{d(x, \text{decision boundary}); d(x, \text{region of competence boundary})\}. \]  

(14)

This solution overcomes the problem induced by the decision boundaries that extend outside the region of the training data used to form the classifier. Data samples that fall far from the training samples, but close to an extending decision boundary will get high distance values (figure 3.14, right image), thanks to the correction to the distance value by equation (14). This solution is a simple, although effective approach, defining a rectangular box around the training data to define the region of competence of the classifier. However it is possible that using a more sophisticated method to define the region of competence (e.g. using the convex hull of the training data or taking into account the outliers) might provide better result. This area is subject to further research.

3.5. Comparison to the Mixtures of Experts Structure

In this experiment the proposed method to combine decision tree classifiers is compared to the Mixtures of Experts (MOE) structure [Jordan (1994)]. In the MOE structure a gating network is used to weigh the output of the different classifiers (experts) according to the input sample.
In figure 12 comparison of the weights produced by the MOE structure and the proposed system for the puzzle–ring classification problem (see figure 7) are shown. The same way two decision trees were trained for each dataset. The gating network of the MOE structure were trained using data from each dataset. The left figure shows the gating functions of the MOE structure in the input space. The output of the MOE structure is the weighted output – according to the gating network – of the classifiers or experts in the structure. In this example the output is the classifier that has greater weight, since there are only two classifiers. It can be seen that similar results can be achieved – with the proposed method – to the MOE structure. The two classifier selection boundaries from the different methods (the proposed and the MOE) share similar characteristics. However there are major benefits of the proposed classification confidence based weighting voting scheme over the MOE structure:

1. The proposed method does not require a gating network. The classification confidence, that act as a gating value, is calculated by the individual classifiers. It is a natural consequence that there is no need to train this network.
2. The proposed method provides confidence values at far better resolution inside the regions of the classifiers forming the ensemble. The weights of the proposed system incorporate direct and detailed information about the performance of the classifiers in their competitive region. This especially true around the decision boundaries of the classifiers, where the confidence drops in contrast to the MOE structure. This way the proposed system can provide valuable classification certainty information attached to the predicted class inside the competitive regions also, where the MOE would only give a gating value of one.
3. If a new classifier (or expert) is added to the structure the MOE system needs to be retrained. On the contrary the proposed system allows to us to dynamically add new classifiers to the system, without the need to retrain anything. The only requirement that the new classifier must also supply it’s classification confidence value attached to the predicted class.

3.6. Comparison to Majority Voting During Bagging

The proposed method to combine decision tree classifiers is now evaluated during bagging, which is a common way of creating classifier ensembles from the same training data [Breiman (1994)]. Multiple classifiers are created using the bootstrap replicates of the original training set. In case of canonical outputs plurality voting is used to form the aggregated class prediction. It has been shown that bagging can give substantial classification accuracy improvement, however the key element is the instability of the predictors. This means bagging performs best if the classifiers trained on different bootstrap samples are different. During bagging the trained classifiers operate in the same input space region.

In this experiment the plurality voting is replaced by the certainty weighted
4. Summary

A novel method is presented in this paper to extend decision tree classifiers to provide a unique classification confidence value for each input sample classified. This classification confidence value is a function of the predicted accuracy and the number of training samples that were available during training at a specific location in the input space. The method is applicable to every classifier system using hyperplanes to cluster the input space. It is not restricted to axis-parallel trees (while previous methods are), it can be used with classifiers utilizing oblique hyperplanes.

It was also shown that this confidence value can be effectively used to form classifier ensembles. The proposed method to combine these kind of classifiers – that provide a classification confidence value attached to their predicted classification –
Fig. 13. Comparison of majority voting and certainty weighted voting during bagging. Results are demonstrated on three datasets (puzzle, ring, and Breast Cancer Wisconsin respectively). All values are in percentages.

is derived from the majority vote rule by forming weights from the classification confidence value. The proposed voting scheme adaptively weights the classifiers in the ensemble according to their predicted classification confidence, hence creating an input dependent, adaptive, self organized combiner architecture, which is:

(1) Input dependent: every classifier predicts its own classification confidence, which is based on the input sample’s distance from the decision boundary, hence unique to every input sample at different distances.

(2) Adaptive: since the weights during the combination of the predicted classes are formed from the classification confidences of the classifiers in the ensemble, and these classification confidences are dependent on the input pattern, it is clear that the voting weights will also be the function of the input sample.

(3) Self organized: again, voting weights are formed from classification confidences provided by the classifiers in the ensemble. As highlighted previously this architecture does not require a separate combiner network or logic. There is no need train or form this network (or logic), and the architecture also allows us to add new classifiers to the ensemble dynamically without the need to retrain or reform this kind of combiner logic. If a new classifier is inserted into the
ensemble no modification is required to any of the components in the system.

The proposed classification certainty weighted majority voting was compared to two widespread used methods to create classifier ensembles:

(1) the Mixtures of Experts structure (MOE), and
(2) Bagging.

These two methods follow different approaches to create and combine classifier ensembles. The MOE structure is developed to partition the data in the input space and assign classifiers to these regions. If an input sample is classified, the classifier(s) that are most competent in the region get the highest weight in the final voting process. This accomplished by the use of a separate gating network that assigns the weights to the classifiers according to input pattern.

The bagging approach follows a different path. From the same training dataset bootstrap replicates are created and the classifiers are trained on these replicated datasets. In this case every classifier operates in the same input space region and majority voting is used to derive final classification.

During the comparisons it was shown that the proposed method is applicable in both cases, whether there is overlapping between the classifiers or not. It was shown that the proposed method can mimic the MOE structure in behavior, because the provided classification confidence values can be regarded as gating functions that partition the input space. It was shown that the classification confidence functions posses’ significant advantages over the MOE structure’s gating functions.

During the comparison to bagging it was shown that when the classifier outputs are highly correlated and the classifiers occupy the same input region, the method is still applicable. In this case the proposed system still resulted in smaller misclassification errors, however the improvement was not significant. These evaluations show that the proposed classifier combiner scheme can be effectively applied without the need to check the independence or the overlapping of the classifiers.

References

**Biography**

Norbert Tóth was born in Budapest, Hungary on March 23, 1980. He received the M.Sc. degree in Electrical Engineering from the Faculty of Electrical Engineering and Informatics, Budapest University of technology and Economics, (BUTE), Hungary, in 2003 and is currently working toward the Ph.D. degree at the Department of Measurement and Information Systems, BUTE. His research interests include decision support systems, pattern classification, and decision uncertainty.

Béla Pataki was born in Budapest, Hungary. He received the M.Sc., Technical Doctorate, and Ph.D. degrees from the Budapest University of Technology and Economics, (BUTE), Budapest, Hungary, in 1978, 1994, and 1997, respectively, all in electrical engineering. He is currently an Associate Professor at the Department of Measurement and Information Systems, BUTE. His current research topics include intelligent signal analysis and processing, medical image processing, and decision support systems.