Production Systems and Information Engineering
Volume 3 (2006), pp. 21-38

A PROPOSED METHOD TO HANDLE CLASSIFICATION UNCERTAINTY USING DECISION TREES

NORBERT TÓTH
Budapest University of Technology and Economics, Hungary
Department of Measurement and Information Systems
ntoth@mit.bme.hu

BÉLA PATAKI
Budapest University of Technology and Economics, Hungary
Department of Measurement and Information Systems
pataki@mit.bme.hu

[Received November 2005 and accepted May 2006]

Abstract. A novel method is proposed in this paper to handle the classification uncertainty using decision tree classifiers. The algorithm presented here extends the decision tree framework to give the ability of measuring the confidence of the classification. Using this algorithm a certain number of the input samples are rejected as "risky points" in order to obtain a smaller misclassification rate on the remaining points. The algorithm is being integrated into a Medical Decision Support System where a confindence number to every classification is required.

Keywords: decision tree, classification uncertainty, CART, misclassification, medical decision support system

1. Introduction and Inspiration

Breast cancer is one of the most common form of cancer among women. Every 12th woman suffers from this disease at least once in her lifetime [1]. Since the cause of breast cancer is unknown, early detection is very important. If detected early, the five-year survival rate exceeds 95% [1].

Currently mammography (X-ray examination of the breast) is the most efficient method for early detection. In a mammographic session usually two images are taken of both breasts. Craniocaudal (CC) is a top view, mediolateral (ML) is roughly a side view image of the breast. Radiologists typically notice suspicious-looking structures in one view and then verify their suspicion by checking the corresponding area of the other view of the same breast. The most important mammographic symptoms of breast cancer can be divided into two main classes: microcalcifications (a group of small white calcium spots) and masses (usually approximately round object brighter than its surrounding tissue).

If a global screening were done, a huge number of mammograms (approximately one million images every year in Hungary) would require diagnostics. The main goal is to create a tool that can ease the work of radiologists by filtering out the true negative cases and draw attention to the suspicious ones. Such Medical Decision Support System for Mammography is being developed in cooperation with radiologists in the Budapest University of Technology [2].

In the system several detection algorithms are working parallel to each other, looking for different kinds of abnormalities (e.g. microcalfications, masses) or different kinds of features to detect the same type of abnormality. Since markings (spots that show the location of an abnormality) created by the detection algorithms cannot be 100% certain, a confidence value was introduced to the system. Each marking is accompanied by this value, showing the diagnoses certainty. The higher this value, the more possible is that the marking is a true abnormality. This value is also used by post-processing algorithms to filter out the most likely false positive markers (the ones with the lowest confidence value). Each algorithm is produces this confidence value, although in different ways.

One of the methods uses decision trees to classify a certain number of features at a location of the image [3]. The result of this classification can be normal tissue or abnormality. If the features are classified as abnormal tissue a marking is generated. To generate the confidence value the original decision tree algorithm was modified to handle classification uncertainty.

This paper discusses a novel extension to the original Classification and Regression Tree (CART) framework (proposed by Breiman et al, 1984) [4] to handle classification uncertainty.

2. Extension of the Decision Trees

2.1. Basis of the Current Work

The origin of decision trees dates back to 1963, when the AID (Automatic Interaction Detection) program [5] was developed at the Institute for Social Research, University of Michigan, by Morgan and Sonquist. They proposed a method for fitting trees to predict a quantitative variable. The AID algorithm created regression trees. A modification to the AID was the THAID algorithm [6] in 1973 by Morgan and Messenger which handled nominal or categorical responses. The THAID program created classification trees. Now several decision tree approaches exist, e.g.: CART, ID3, C4.5 [7], C5, THAID CHAID, TREEDISC, etc.

One the most widespread used decision tree framework is the Classification and Regression Trees (CART, 1984) [4] developed by Breiman et al. Their work is based on the original ideas of the AID and the THAID algorithms. We used their work as basis for our enhancements to the decision tree methodology.

2.2. Decision Tree Basics

Decision trees can be used to predict values or classify cases. Because in our work (mammographic image analysis) classification is the main issue; therefore from now on we restrict our discussion to classification trees. Classification trees are used to predict membership of cases or objects in the classes of a categorical dependent variable from their measurements on one or more predictor variables.

Decision tree methods use supervised learning to recursively divide the observations into subcategories in such a way that these subcategories differ from each other as much as possible while each subcategory is as homogenous as possible. The outcome of a decision tree building algorithm is a directional graph connecting nodes. Each node of the graph represents a set of observations. There are two kinds of nodes: “terminal” and “non terminal” Non terminal nodes are also referred as “internal” nodes. These nodes incorporate a “splitting rule”, which is used to split the observations into subcategories. Terminal nodes are also referred as “leaf” nodes. These nodes represent the dependent variable – in our case – the predicted class of the observations (figure 1).

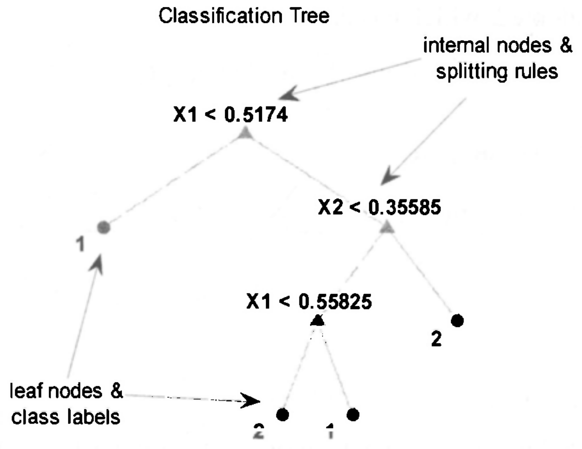

Figure 1. Sample classification tree with 3 splits and 2 classes.

There are various ways to grow a decision tree. For example there are several options to implement the splitting rule or the selection of the right sized tree. For our work we implemented CART algorithm [4] and used it as a basis for further development. Properties of the tree growth algorithm:

- Splitting rule

A splitting rule deals with observations reaching that specific node. It is used to divide that group of observations into subgroups. We use 2-way binary (smaller / bigger) splits on one variable. At each node a single variable is tested if bigger or smaller than a certain threshold value. At each node impurity is defined to measure the homogeneousness of the node. The best split is the one that creates the purest nodes. Given a node t with estimated class probabilities $p(j|t)$, $j=1..Nc$, where Nc is the number of classes, a measure of node impurity for given t

$$i(t)=f[p(1|t), \dots, p(Nc|t)] \tag{2.2.1}$$

is defined and an exhaustive search is made to find the split that most reduces tree impurity. This split maximizes the impurity gain

$$\Delta i(t)=i(t)-pLi(tL)-pRi(tR), \tag{2.2.2}$$

where pL and pR is portion of observations falling to left or right child node (tL, tR) according to split. To measure node impurity the Gini diversity index [4] was adopted, which has the form

$$i(t)=\sum_{j\neq i} p(j\mid t)p(i\mid t) \tag{2.2.3}$$

and can be rewritten as

$$i(t)=(\sum_j p(j\mid t))^2-\sum_j p^2(j\mid t)=1-\sum_j p^2(j\mid t). \tag{2.2.4}$$

The Gini index ensure that for any split s the impurity can only decrease: $\Delta i(s,t)\geq 0$.

- Determining the right sized tree

In general, bigger trees having more splits, give better classification rate on the training data. However they tend to overfit, giving worse classification rates on the test data. To determine the right sized tree – that gives the best error rate (R) on the test data and avoids overfitting – there are 2 options. The first is to stop splitting according to a certain criterion. According Breiman's [4] and our experiments as well this is not recommended. The better way to determine the right sized tree is to grow a tree that is much too large and than "prune" it upwards iteratively until we reach the root node. After this test sample error estimates (R) are used to select best subtree that has minimal error on the test data. We implemented the Minimal Cost Complexity (MCC) pruning algorithm [4]. In MCC pruning a cost-complexity measure is introduced:

$$R\alpha(Tt)=R(Tt)+\alpha|Tt|, \tag{2.2.5}$$

where α is the cost-complexity parameter (real number), Tt is the subbranch starting at node t (if t=1, than $T1=T$ the original tree) and $|Tt|$ is number of terminal nodes on the subbranch Tt. The higher the value of the α parameter the greater the cost of more complicated trees. In this sense the tree complexity is defined by the number of its terminal or leaf nodes. To get a series of pruned subtrees we start from the original tree and we perform a "weakest-link cutting" This is done in the following way:

set

$$R\alpha(\{t\})=R(\{t\})+\alpha \tag{2.2.6}$$

and

$$R\alpha(Tt)=R(Tt)+\alpha|Tt|. \tag{2.2.7}$$

As long as $R\alpha(Tt)< R\alpha(\{t\})$ the branch Tt has a smaller cost-complexity than the single node $\{t\}$. In other words it is "worth" to keep this node expanded. At a critical value of α the two cost-complexities become equal, than keeping only a single node $\{t\}$ instead of an expanded branch Tt is preferable. To find this critical α, the following equation must be solved:

$$\alpha=\frac{R(\{t\})-R(Tt)}{|\mathrm{T}|\text{-}1}. \tag{2.2.8}$$

This critical α value has to be calculated for all internal nodes of the tree, and than the smallest is the "weakest-link" This means that node is the one that – if we increase α – has to be "closed" to get better cost-complexity value for the entire tree T. Closing means to prune the tree at that location, to replace the branch Tt with the single node t.

Using this method we get a series of smaller and smaller subtrees according to the increasing value of α. To select the best tree we can use test sample or cross-validation error estimates. We used 10-fold cross-validation [4] to estimate the misclassification rate. In this sense the best tree is the smallest one that has minimal cross-validation (or test sample) error (figure 2).

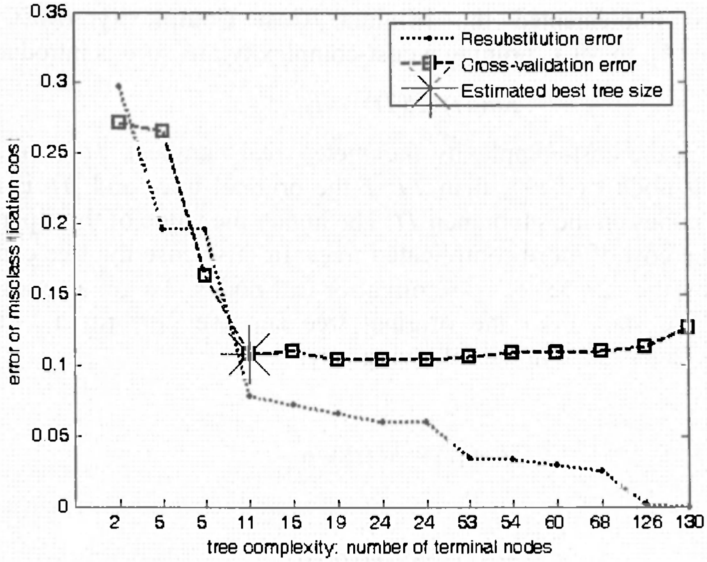

Figure 2. The best tree is the one that has minimal cross-validation error. Resubstitution error means the error on the learning data.

Decision trees produced by the CART algorithm have some favorable properties compared to other methods. They are easily interpretable and can be used to classify data very quickly. Another good property is that the decision boundary can be easily identified. In the next sections we will introduce an algorithm to provide a confidence value to the classification result of the tree. This algorithm makes use of the clear structure of the decision trees and the explicitly defined decision boundary.

2.3. Dealing with Classification Uncertainty

A classification tree 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. Dealing with the classification uncertainty or classification confidence the main assumption is that the confidence of the classification is proportional to the distance from the closest decision boundary that splits between different class labels.

According to the previous assumption, to get a classification certainty value we need to measure the shortest distance to the closest decision boundary that splits between different classes, or equally the shortest distance to the closest section with different class label.

The proposed algorithm to measure the shortest distance from the closest decision boundary that splits between different classes is the following:

1) First the actual data is classified using the decision tree: a leaf node is reached, which defines a section in the input space and an output label.

2) To get the distances to the other sections the input data point is projected to all of the decision boundaries. The projection rules are calculated only once (see 2.4 determining projection rules), right after the tree growth process and stored together with the tree structure.

   If the input space contains *N* variables than the decision boundaries of a section are maximum *N*-1 dimensional hyper planes (figure 3, 4). A 0 dimensional boundary only exists if all variables in the input space exist in the path from the root node to the leaf node.

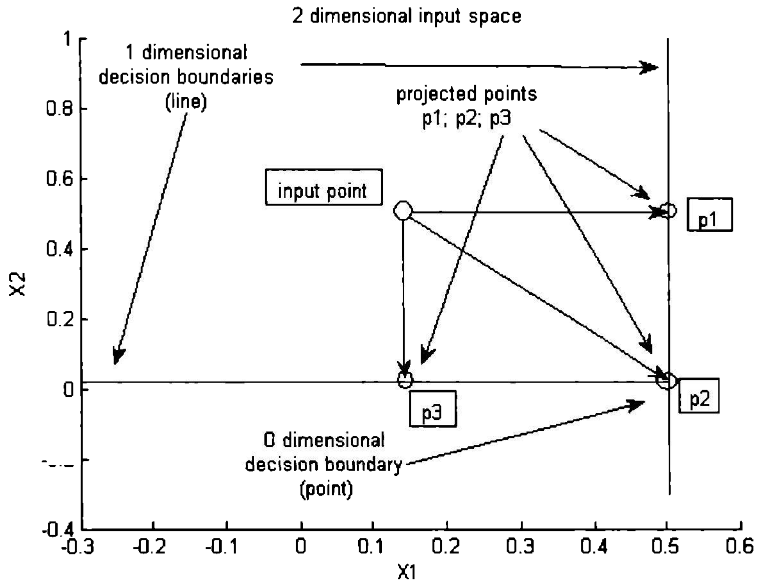
2 dimensional input space
1 dimensional decision boundaries (line)
projected points p1; p2; p3
input point
p1
p2
p3
0 dimensional decision boundary (point)
X1
X2
1
0.8
0.6
0.4
0.2
0
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

Figure 3. Sample decision boundaries in 2 dimensional input space and the projected points of the input data point.

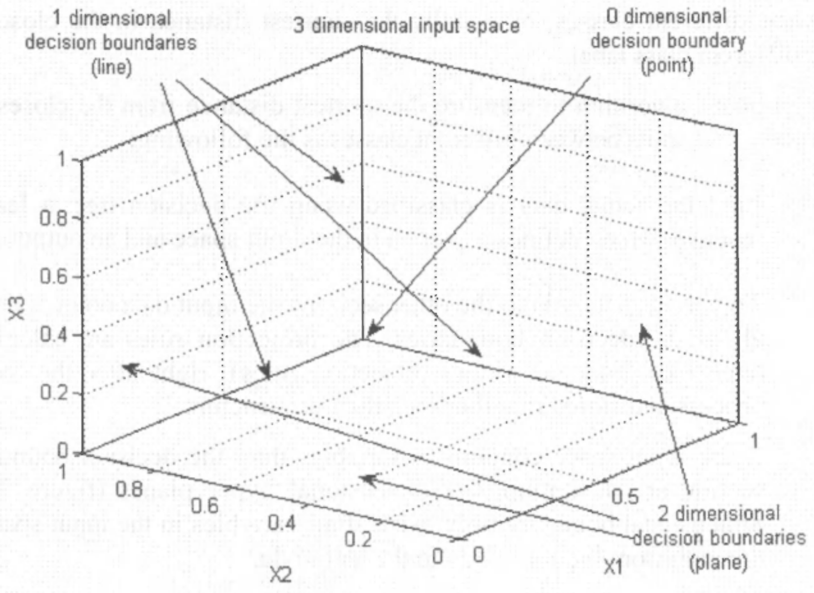
1 dimensional decision boundaries (line)
3 dimensional input space
0 dimensional decision boundary (point)
2 dimensional decision boundaries (plane)
X3
X2
X1
1
0.8
0.6
0.4
0.2
0
0.5

Figure 4. Sample decision boundaries in 3 dimensional input space.

3) The distance between the projected points and the input point is calculated.

4) Take out the projected point that has minimal distance from the input point.

5) Check if that projected point is on a decision plane that splits between different classes (see 2.5 checking projected points).

6) If yes, the output certainty value is the distance between the projected point and the input data. If no, take out the next projected point with minimal distance and repeat steps 5 and 6.

The algorithm described above returns the shortest distance to the closest decision boundary that splits between different classes.

2.4. Determining the Projection Rules

A set of critical points that we call "projection rules" has to be determined for each leaf node. These critical points will be the closest points on the boundary of the actual input space section. We call these "projection rules" because most of these points are not fully defined, they are maximum *N*-1 dimensional hyper planes (the input space is *N* dimensional).

To determine the projection points for a certain leaf the following algorithm is proposed:

Initialize a boundary matrix that will contain the boundary values for the actual input space section marked by the leaf node. This matrix has equal number of rows to the number of input variables in the decision tree. The matrix has 2 columns, because each variable can border the actual segment from above and from below. Initialize the border matrix with infs (abbreviation for infinite) as if no border was present to the actual segment to any direction. Than we go from the leaf node up to the root node taking out the splitting variables and split values. When we take out a split we also check if we came up from a smaller child or from a bigger child. When a split value is taken out we insert it into the boundary matrix into the row identified by the variable and into column 1 if it was a smaller child, and into column 2 if it was a bigger child. If we encounter a split that already in the boundary matrix we skip that because that is not a boundary to the section marked by the leaf node.

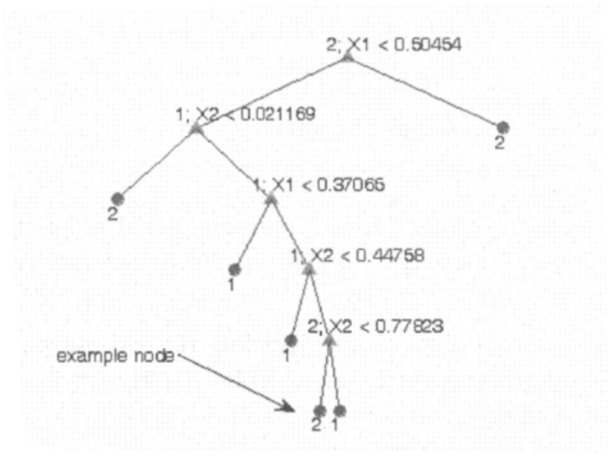

Figure 5. Sample tree with an example node.

The boundary matrix for the example node (figure 5.):

$$BM = \begin{bmatrix} 0.504 & 0.370 \\ 0.778 & 0.447 \end{bmatrix}. \tag{2.4.1}$$

Now the boundary matrix is extended with a column containing infs (referring to infinites). The extended boundary matrix:

$$BMe = \begin{bmatrix} 0.504 & 0.370 & \text{inf} \\ 0.778 & 0.447 & \text{inf} \end{bmatrix}. \tag{2.4.2}$$

To get all the critical points we have to get all the permutations of the elements of the extended boundary matrix, keeping the order of the variables. The points from the matrix *BMe*:

$$\begin{aligned} &P1=(0.504, 0.778) \\ &P2=(0.504, 0.447) \\ &P3=(0.370, 0.778) \\ &P4=(0.370, 0.447) \\ &P5=(\text{inf}, 0.778) \\ &P6=(\text{inf}, 0.447) \\ &P7=(0.504, \text{inf}) \\ &P8=(0.370, \text{inf}). \end{aligned} \tag{2.4.3}$$

The first 4 critical points are fully defined, they are actual points (0 dimensional hyper planes). The rest of them are not fully defined they are (in this case) 1 dimensional hyper planes: lines (figure 6).

These projection rules depend only on the decision tree and independent from the input data. As a result they have to be calculated only once, which saves a considerable amount of time.

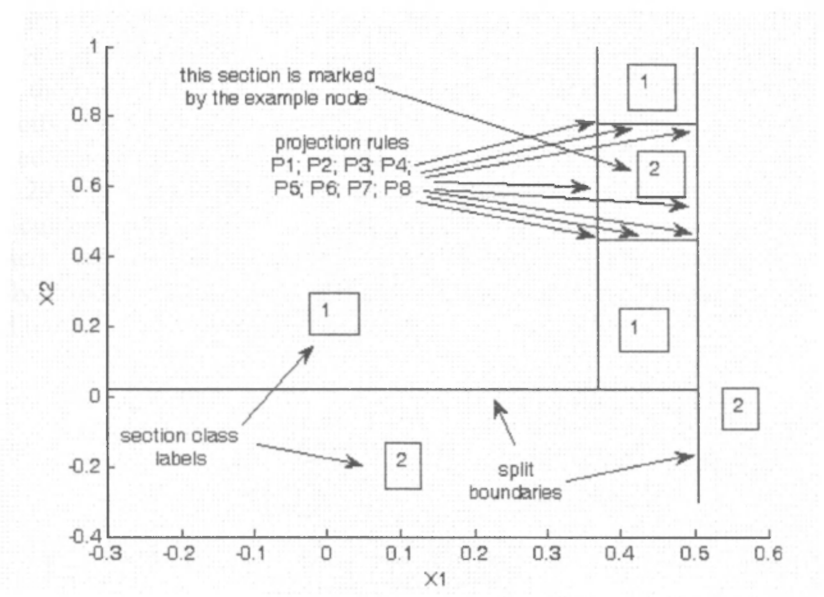
this section is marked by the example node
projection rules P1; P2; P3; P4; P5; P6; P7; P8
section class labels
split boundaries
1
2
1
1
2
2
X1
X2

Figure 6. The 8 critical points (projection rules) for the example node.

We use these critical points (projection rules) to project the input data point to section boundaries. The projection technically means to insert the coordinates of the input point into the projection rules replacing the infs. We have to project the input point with the rules from *each* leaf node. These projected points will be the closest boundary points to the input point. However it can not be known if a projected point on a section boundary that splits between different classes. This has to be determined individually for each projected point (see 2.5 checking projected points).

2.5. Checking Projected Points

A projected point is not necessarily on a boundary that splits between different classes. Checking each side of a decision boundary in the worst case requires 2^*N*-1 points to be classified by the tree if we checked all sections around the projected point. However we only have to determine the class of the section on the other side of the boundary in the direction of the projection. We do not have to check the surrounding points because those are not the closest border points to the corresponding area. Figure 7 shows a simplified illustration.

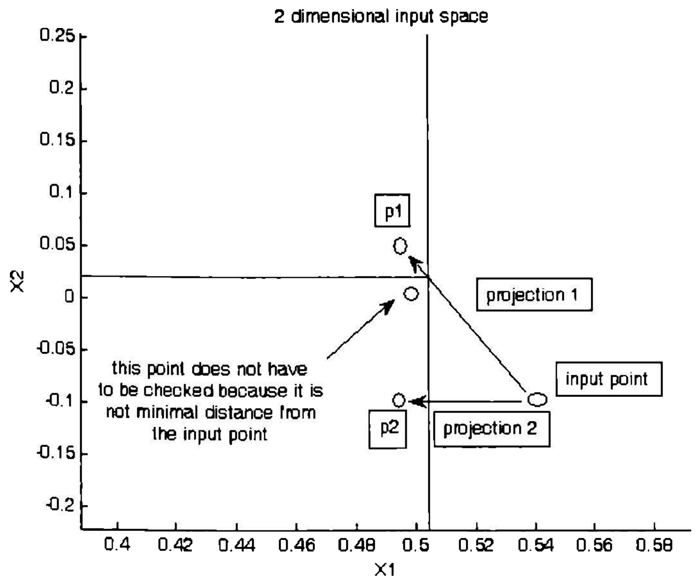
2 dimensional input space
p1
projection 1
input point
this point does not have to be checked because it is not minimal distance from the input point
p2
projection 2
X1
X2

Figure 7. Checking projected points. The point marked does not have to be checked because *p*2 is in the same section and is closer to the input point (using the algorithm given in section 2.3 it is already checked by the time we get to checking *p*1).

Checking technically means to "push" the projected point further in the projection direction to an ε distance into the corresponding section and than use the tree to classify the point (figure 7, *p*1 and *p*2).

3. Test Results and Conclusion

3.1. Test Result on 2 Dimensional Data

The algorithm is first demonstrated on a 2 dimensional dataset. The input data is shown in figure 8. The input points are marked with an 'x' or ' ' according to their class label. After the decision tree is grown (using cross-validation and MCC pruning), the input space is covered with a grid and the distance from the decision boundaries are calculated in the grid's points using the algorithm presented in section 2.3. Figure 9 displays the distance from the decision boundaries.

There were N=1354 data points in the sample dataset, containing roughly equal number of class 1 and class 2 members. 1248 points were correctly and 106 were incorrectly classified by the tree. This gives 7.8% misclassification error rate.

Using the introduced algorithm we calculate the certainty value (the distance from the decision boundaries) for each input point. We determine a certainty threshold such that if the certainty for a given input point is smaller than this threshold the tree rejects the classification. For this application the threshold is defined to keep 76% of the correctly classified samples (figure 10). In this case 90% percent of the incorrectly classified cases are filtered out (96 points out of 106). 960 cases are classified out of the total 1354, which is around 71% of the total number of input points. 29% percent (394) of the input samples are considered as risky, meaning the calculated classification certainty value was under the threshold. From the classified ones 10 are misclassified, which means 0.1% misclassification rate. Visually this means the method filters out the risky cases inside a "safety lane" around the decision boundaries (figure 11).

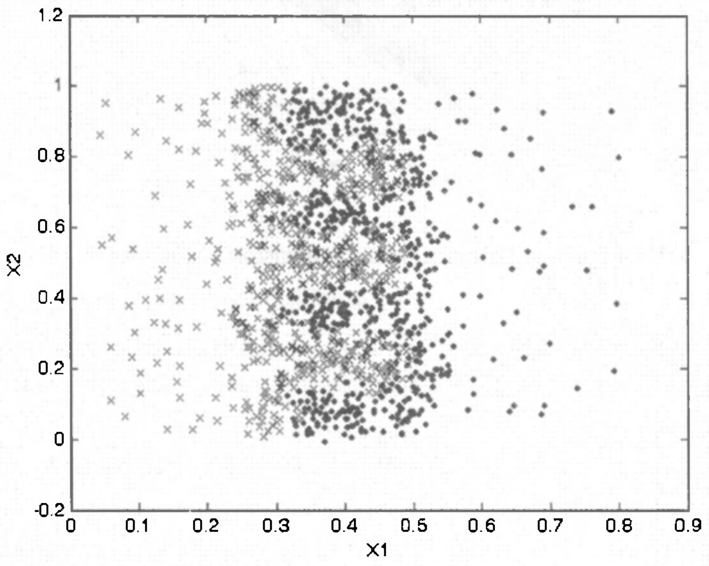

Figure 8. A 2 dimensional sample dataset. The input points are marked with an 'x' or '.' according to their class label.

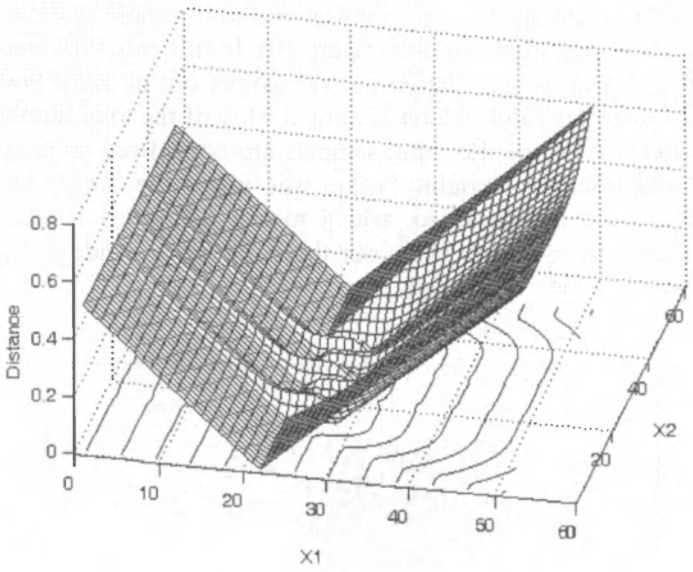

Figure 9. Distance from the decision boundaries.

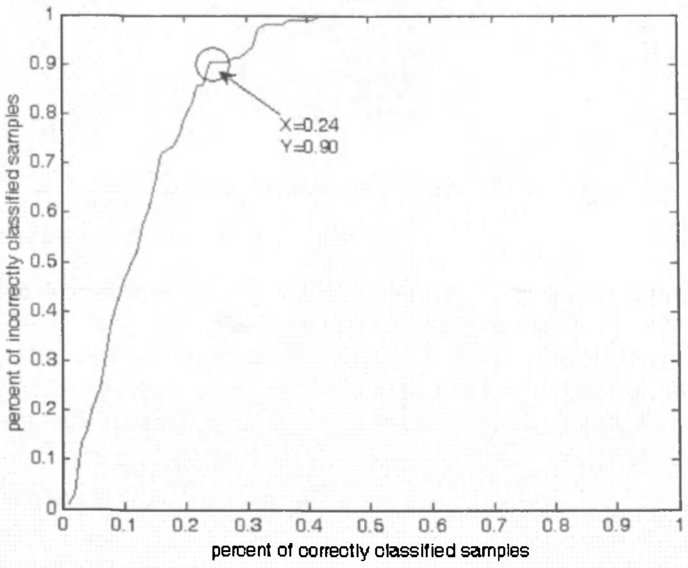

Figure 10. Defining the threshold value to keep 76% of the correctly classified samples.

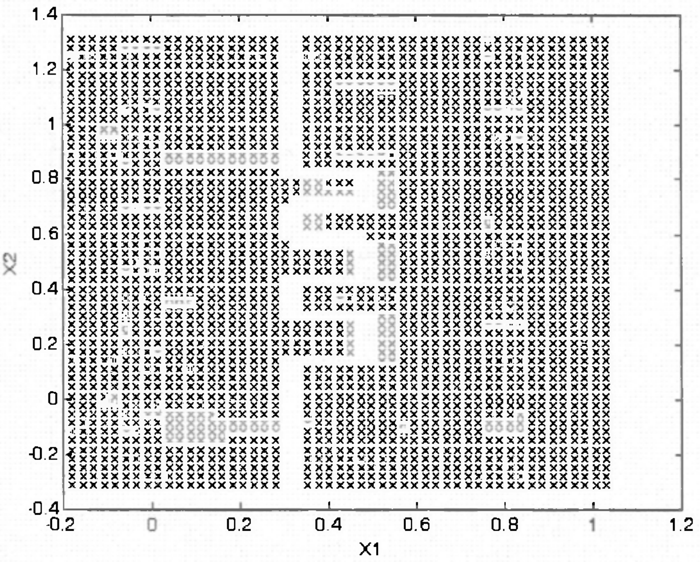

Figure 11. The "safety lane" around the decision boundary.

3.2. Test Results on 9 Dimensional Data

The algorithm is now demonstrated on a 9 dimensional data set. This dataset is the "breast-cancer-wisconsin" dataset downloaded from the UCI Machine Learning Repository [8].

2 experiments were made with different trees. The database contained N=699 data points.

In the first experiment the tree misclassifies 33 points out of the 699, which gives 4.9% misclassification rate. The certainty threshold is determined to keep 90% percent of the correctly classified samples (figure 12). This case the tree rejects 12.7% of the input points but the misclassification rate on the remaining points reduces to 1.5%.

In the second experiment the tree misclassifies 23 points out of the 699, which gives 3.3% misclassification rate. The certainty threshold is determined to keep 93% percent of the correctly classified samples (figure 13). This case the tree rejects 8% of the input points and the misclassification rate on the remaining points reduces to 2%.

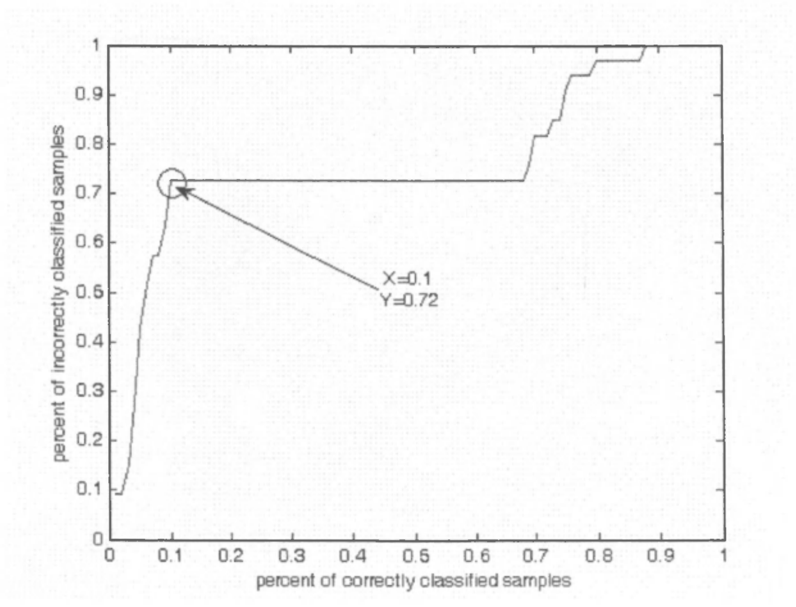

Figure 12. Defining the threshold value to keep 90% of the correctly classified samples. 72% of the misclassified samples are filtered out.

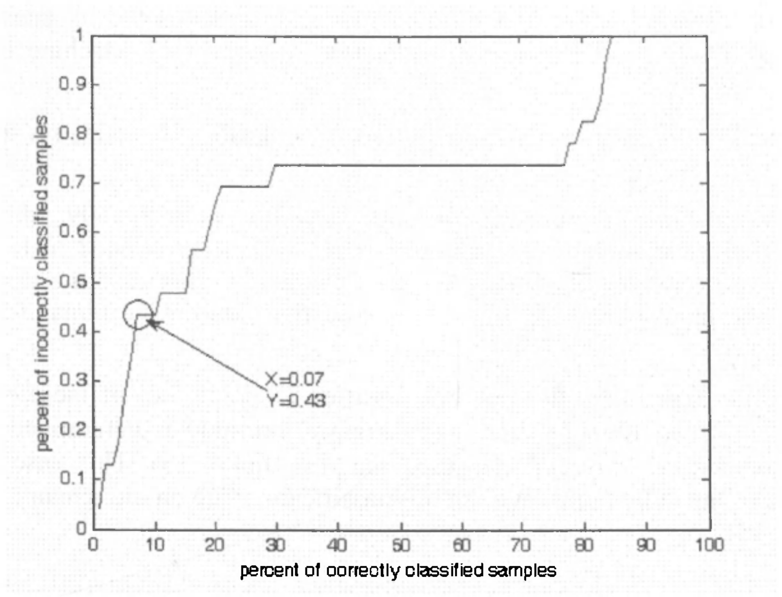

Figure 13. Defining the threshold value to keep 93% of the correctly classified samples. 43% of the misclassified samples are filtered out.

3.3. Conclusion and Future Work

A method was presented in this paper to extend the decision tree framework. The proposed extension gives the possibility to determine classification certainty.

The method proposed was tested on two sample datasets. A certainty threshold was determined to filter out the most "risky" classifications with low certainty values. Results indicate that the proposed algorithm can significantly reduce the misclassification error, in the cost of rejecting a portion of the input points, considering them as "risky" points. The algorithm takes advantage of the clear structure of the decision trees and the explicitly defined decision boundaries. The projection rules have to be determined only once after the tree growth process. Calculating the distance from the relevant decision boundaries involves projecting the input point, than measuring the distance to these projection points (section 2.4) and finally checking these points for class changes (section 2.5). This results in a reasonably fast algorithm and gives accurate distance information.

The method is being integrated into the Medical Decision Support system that is under development in the Budapest University of Technology. Currently results on mammographic data are very preliminary and will be published during the next year.

In the presented examples the key parameter when using the method is the value of the classification certainty threshold. This parameter controls the balance between the rejection rate and the classification certainty. In the demonstrative examples above this threshold was determined manually. Current research focuses on finding a method to automatically determine this threshold, which is optimal in certain means.

Acknowledgements

This work was sponsored by Research and Development Secretariat of the Hungarian Ministry of Education under contract IKTA 102/2001 and by the Hungarian Fund for Scientific Research (OTKA) under contract T046771.

REFERENCES

[1] HIGHNAM, R., BRADY, M.: *Mammographic Image Analysis*. Springer, 1999.

[2] HORVÁTH, G., VALYON, J., STRAUSZ, GY., PATAKI, B., SRAGNER, L., LASZTOVICZA, L., SZÉKELY, N.: *Intelligent Advisory System for Screening Mammography*. Proc. of the IEEE Instrumentation and Measurement Technology Conference, IMTC '2004. Como, Italy, May 18-20. Vol.3. pp. 2071-2077.

[3] SZÉKELY, N., TÓTH, B., PATAKI A.: *Hybrid System for Detecting Masses in Mammographic Images.* Proceedings of IMTC/04, 21th IEEE Instrumentation and Measurement Technology Conference, Como, Italy, 18-20 May 2004, pp. 2065-2070.

[4] BREIMAN, L., FRIEDMAN, J., OLSHEN, R., STONE, C.: *Classification And Regression Trees.* Chapman & Hall, 1984.

[5] MORGAN, J.N., SONQUIST, J.A.: *Problems in the Analysis of Survey Data, and a Proposal.* Journal of the American Statistical Association, 2963, 58, 415-35.

[6] MORGAN, J.N., MESSENGER, R.C.: *THAID- a Sequential Analysis Program for the Analysis of Nominal Scale Dependent Variables.* Technical Report, Survey Research Center, Institute for Social Research, Univeristy of Michigan, 1973.

[7] QUINLAN, R.J.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993, his website: http://www.rulequest.com

[8] *UCI Machine Learning Repository*, http://www.ics.uci.edu/~mlearn/MLRepository.html