이미지 데이터베이스에서 인터벌을 이용한 객체분류를 위한 분리 방법

조 준 서[†] · 최 준 수[††]

요 약

정확한 객체 분류를 위한 분리 기준을 지정하기 위한 방법은 의사결정수들(decisions trees) 사이에 주요한 이슈이다. 이 논문은 최적의 분리 지점을 찾기 위해 분류에 대한 새로운 분리방법을 기술하고 있다. 모든 시작점 값들(threshold values)을 검색함으로 제공되는 기존의 분리방법들과 다르게, 이 논문에서는 미리 지정된 인터벌들의 확률을 기반으로 하는 분리방법을 제안하고 있다. 제시한 방법은 사용자가 인터벌의 수를 조정함으로써 트리의 정확도를 통제할 수 있도록 할 수 있으며, 제기된 분리방법을 이미지 객체 인식하기 위한 특징추출을 수치화함으로써 검색되는 일련의 이미지 데이터에 적용하였다.

키워드 : 객체, 분류, 확률, 인터벌, 이미지

Splitting Rules using Intervals for Object Classification in Image Databases

June-Suh Cho[†] · Joonsoo Choi[††]

ABSTRACT

The way to assign a splitting criterion for correct object classification is the main issue in all decisions trees. This paper describes new splitting rules for classification in order to find an optimal split point. Unlike the current splitting rules that are provided by searching all threshold values, this paper proposes the splitting rules that are based on the probabilities of pre assigned intervals. Our methodology provides that user can control the accuracy of tree by adjusting the number of intervals. In addition, we applied the proposed splitting rules to a set of image data that was retrieved by parameterized feature extraction to recognize image objects.

Key Words : Classification, Probability, Interval, Object, Image

1. Introduction

This paper discusses how to recognize image objects using a classification method in image databases. Our method provides simple splitting rules based on the probabilities of intervals. New splitting rules based on the probabilities allow one to search for interesting regions of data.

In prior work[3, 11, 13, 17, 18], most methods of classification repeatedly search for the best split of a subset by searching all possible split points for all variables. In addition, these methods suffer from the raw data to maintain the accuracy. This paper addresses the problem of finding splitting rules for a classifier using a set of pre-assigned intervals for covariates rather than exhaustively searching over all possible splitting values.

In this paper, we propose the method based on new splitting rules from image databases using the probabilities of pre-assigned intervals, which are randomly or manually assigned. The intervals provide the meaningful distribution of objects and the stopping criteria. Our methodology also provides the user with control of the accuracy of the tree by adjusting the number of intervals.

Prominent examples of classification have been based on different splitting criteria such as Gini index[3], entropy[17], Chi-squared test[11], and misclassification rate[13]. These methods do an exhaustive search to find the best split points. The exhaustive search approach has problems with bias in variable selection, tree size and depth, and can be intolerant of small changes in the learning sample. In order to address these problems, we

† 정 회 원 : 한국외국어대학교 경영학부 조교수
†† 정 회 원 : 국민대학교 자연과학대학 컴퓨터학부 부교수
논문접수 : 2005년 3월 21일, 심사완료 : 2005년 10월 7일

propose simple splitting rules based on minimizing the sum of variance and maximizing the difference of probabilities of intervals. In this paper, we focus on the split selection to classify image objects using given rules instead of a tree construction.

The rest of this paper is organized as follows: In Section 2, we review the background of our work. In Section 3, we describe the new splitting rules. In Section 4, we show the results of our experiments. Finally, we conclude this paper in Section 5.

2. Theoretical Background

Most work on image recognition and classification has concentrated on or been based on the object recognition and detection methods. Decision tree algorithms are very important, well-established machine learning technique that has been used for a wide range of applications, especially for classification problems[3, 8, 9, 15, 17]. Many classifiers can be viewed as computing a set of discriminant functions for a data set, one for each class, and assigning to each class the function whose discriminant value is maximum[2]. Examples here include statistical learning methods like CART[3] and C4.5[18].

There are many splitting rules for decision trees, and advantages and disadvantages for each rule. Therefore, there is no best splitting rule that can be applied to all problem types or purposes. These methodologies are summarized as follows:

CART[3] is used to build a decision tree whose questions minimize the impurity of the subsets at that point in the tree. As long as new splits can be found, the decision tree keeps on growing. These splits improve the ability of the tree to separate objects of a training set into classes. Each of the candidate subtrees is used to classify the objects in the test set. The tree with the lowest overall error rate is declared the winner.

C4.5[18] produces trees with varying number of branches per node. It uses a criterion called information gain (entropy) to compare potential splits. Another area in which C4.5 differs substantively from CART is in its approach to pruning. C4.5 prunes the tree it has grown without reference to any data beyond the training set.

CHAID[11], which descended from an earlier automatic interaction detection(AID), is the oldest of the algorithms for detecting statistical relationships between variables. The algorithm searches for a way to use the input variables to split the training data into two or more child nodes. A Chi-squared test is used to measure if the proportion of classes is significantly different. In CHAID, the tree keeps growing until no more splits are available. This leads to differences in classification that are statistically significant. Another difference is that CHAID is restricted to categorical variables. Continuous variables must be broken into ranges or replaced with classes.

QUEST[14] is a method for computing binary classification trees based on univariate or linear combination splits for categorical predictors or ordered predictors. A unique feature is that its attribute selection method has negligible bias. If all the attributes are uninformative with respect to a class attribute, then each has approximately the same chance of being selected to split a node.

These methodologies repeatedly search for the best possible split of a subset by searching all candidate split points for all variables. However, our approach uses intervals. The user manually or randomly assigns intervals for each attribute, and is less than the number of all possible split points. Exhaustive search methods have trouble with the tree size and depth. In this paper, our approach is to find the best split points with a simple, small classifier, and provide good accuracy of classification to recognize the objects.

In the following section, we specifically examine the classification with only the two-class situation. The criteria used by traditional trees are summarized as follows:

We assume that there are two classes, 0 and 1. We give definitions as follows:

n = the total number of objects in the parent node
n_L = the number of objects in the left bucket
n_R = the number of objects in the right bucket
n^1 = the number objects of class 1 in the parent node
n^0 = the number objects of class 0 in the parent node
n_L^1 = the number of objects of class 1 in the left bucket
n_R^1 = the number of objects of class 1 in the right bucket
n_L^0 = the number of objects of class 0 in the left bucket
n_R^0 = the number of objects of class 0 in the right bucket

$$P_L=\frac{n_L}{n},\quad P_R=\frac{n_R}{n},\quad P_L^1=\frac{n_L^1}{n_L},\quad P_R^1=\frac{n_R^1}{n_R},$$

$$P_L^0=\frac{n_L^0}{n_L},\quad P_R^0=\frac{n_R^0}{n_R},$$

Split point = A split point is a point P, which divides a set S into two exclusive subsets, S_1 and S_2, such that the attribute value of all points in S_1 is $<$ the attribute value of the split point P and the attribute value of all points in S_2 is $\geq$ attribute value of the split point P.

3. The Proposed Methods

General approaches to split selection have been proposed in the statistical literature[11, 3, 18, 14, 19]. Most approaches examine all possible binary splits of the data along each predictor variable to select the split. If X is an ordered variable, this approach searches over all possible values c for splits as follows:

$$X \le c$$

In order to split the predictor variables, our approach assigns the range of each attribute variable into some number of uniformly sized intervals.

Our splitting rules compute probabilities based on the intervals, take the intervals, which have the highest probability, then add up the intervals to select a split point by given split rules. Intervals need to give meaningful distribution of objects. To give meaningful distribution, intervals are merged to find a split point. The size of the intervals also provides the stopping criteria.

However, we have a limitation of the number of intervals. If we assign 1, 2, or small intervals, our splitting rules may not provide the best split point and the performance of our method may be worse than other methods. To avoid this limitation, we need to find a reasonable interval size on datasets.

The advantage of our approach is that the number of intervals can be controlled by the user. In addition, we stop splitting the subset if the sample size in some bucket is less than a user-specified value or there is no significantly different variable under the splitting rules.

In the context of probabilistic object recognition, we compute the probabilities of the intervals. Given a split with left and right buckets, the probabilities of class 0 and class 1 on the left and the right satisfy $p_L^0 + p_L^1 = 1$, $p_R^0 + p_R^1$ respectively.

In this study, we consider two kinds of splitting rules. One is to select a best split point by minimizing the sum of variances of intervals, another is that maximizing the difference of probabilities of intervals. We assume that the variance of an interval is the sum of the individual variances. In the following equation, q_L and q_R represent the fraction of class 1 that goes to the left and right buckets. We consider a splitting rule as follows:

$$\left[1 - \left(q_L p_L^1 + q_R p_R^1\right)\right]$$

This criterion to split a node is obtained by considering the proportion of responses in left and right buckets. Since n^1 is fixed at the parent node, it can be represented as follows:

$$\left(1 - \left[q_L p_L^1 + q_R p_R^1\right]\right) n^1 = n_L p_L^1 p_L^0 + n_R p_R^1 p_R^0$$

where $q_L = \frac{n_L^1}{n^1}, q_R = \frac{n_R^1}{n_1}, n^1 = n_L^1 + n_R^1$

Based on the above equation, we can represent that minimizing the sum of variances is equivalent to maximizing the sum of weighted probabilities in the left and the right bucket as shown in equation (1).

$$\min_c \left[1 - \left(q_L p_L^1 + q_R p_R^1\right)\right] \equiv \max_c \left[q_L p_L^1 + q_R p_R^1\right] \quad (1)$$

Since $q_L + q_R = 1$, we can rewrite the sum of weighted probabilities as follows:

$$q_L p_L^1 + q_R p_R^1 = (1-\alpha)\left[\frac{n n_L^1}{n_L (n - n_L)}\right] + \frac{n_R^1 - n_L^1}{n - n_L}$$

where, $\alpha = \frac{n_R^1}{n^1}, n = n_L + n_R, n^1 = n_L^1 + n_R^1, n_L^1 \le n_L \le n - n_L^1$

This expression is maximized by increasing p_L^1 or p_R^1 when q_L or q_R is neither class 0 nor class 1. Another criterion for splitting is maximizing the difference of probabilities between the left and the right bucket. This approach is based on the search for interesting regions of the data regardless of non-interesting regions. This criterion can be expressed as shown in equation (2).

$$\max_c \left| p_L^1 - p_R^1 \right| \quad (2)$$

As we discussed above, the algorithm to select a best split point under each splitting rule and the method to combine these two rules are summarized in this section. In addition, the splitting rules are calculated based on class 1.The hybrid splitting rule is to compromise the advantages and disadvantage in minimizing the variances and maximizing the difference of probabilities in the left and the right bucket. Our approach for object recognition is based on the hybrid splitting rule.

Splitting rule 1

1. Split a predictor variable into I_i intervals,

$$1 \le i \le L.$$

For each interval I_i, calculate $p_i^1 = P(class1 \mid x \in I_i)$ and $q_i^1 = P(class1 \mid class1 \ in \ pareant \ node, x \in I_i)$

2. From the interval, which has the highest probability, add up the intervals to find a split point c by maximizing the sum of weighted probabilities of each interval as shown in equation (3).

$$d_I = \max_c \left| q_L p_L^1 + q_R p_R^1 \right| \quad (3)$$

Splitting rule 2

1. For each interval I_i, calculate $p_i^1 = P(class1 \mid x \in I_i)$
2. From the interval which has the highest probability, add up the intervals to select a split point c by maximizing as shown in equation (4).

$$d_{II} = \max_c \left| p_L^1 - p_R^1 \right| \quad (4)$$

Hybrid splitting rule

1. For each interval I_i, calculate $p_i^1 = P(class1 \mid x \in I_i)$ and $q_i^1 = P(class1 \mid class1 \ in \ pareant \ node, x \in I_i)$
2. Calculate two statistics d_I, d_{II} and give a rank to them according to the grouping of the interval.
3. The average ranks for each group are given by two splitting rules. When the average ranks are the same, the priority is given to the first splitting rule because there is still a chance to split the other predictor variables later.
4. Take a split point **c** with the highest rank.

After we find split rules, we should consider how to assign stopping criteria. For the stopping criteria, we consider two stopping rules. One is to allow splitting to continue until all leaf nodes are pure or contain no more than a specified minimum number of objects. Another is to allow splitting to continue until all leaf nodes are pure or contain no more objects than a specified minimum fraction of the sizes of one or more classes.

According to the split rules, a tree grows until it meets the stopping criteria. At that moment, we can get the optimal tree. In order to assign stopping criteria, we directly assign the size of objects at leaf nodes. In terms of the stopping rule, we assign the default size of objects at each leaf node to stop splitting at that node.

Split Points Selection with an Example

In this section, we describe how to select split points based on our split rules.

1. We assume that we have eight feature parameter vectors, which have 27 objects for class 0 and 14 objects for class 1 among 41 objects. In addition, a default class for calculation is 1.
2. We assign the intervals into feature parameter vectors such as roundness, form factor, aspect ratio, surface regularity, angle of second moment, entropy, contrast, and mean.
3. For example, we have a feature parameter vector for roundness, and we assigned five intervals in a vector as shown in Figure 3.1. We also assigned five intervals in other feature parameter vectors. We calculate variances and differences of a feature vector for roundness based on given equations as follow:

$d_{I1} = \left|3/5 - 11/36\right| = 0.294$ $\quad d_{II1} = \left(3/5 \times 3/14 + 11/36 \times 11/14\right) = 0.3685$

$d_{I2} = \left|3/8 - 11/33\right| = 0.0417$ $\quad d_{II2} = \left(3/8 \times 3/14 + 11/33 \times 11/14\right) = 0.3422$

$d_{I3} = \left|3/16 - 11/25\right| = 0.2525$ $\quad d_{II3} = \left(3/16 \times 3/14 + 11/25 \times 11/14\right) = 0.3858$

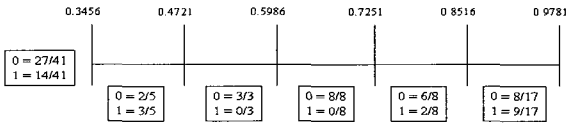
$d_{I4} = \left|5/24 - 9/17\right| = 0.3211$ $\quad d_{II1} = \left(5/24 \times 5/14 + 9/17 \times 9/14\right) = 0.4148$

We also calculate variances and differences of other feature parameter vectors. At each feature vector, we perform the same calculation as above.

4. After calculating the probabilities, we find a best split point among feature parameters. Among feature parameters, we select a feature parameter based on split rules as follow:

Roundness : $d_{Ir} = 0.3211$, $d_{IIr} = 0.4148$
Surface regularity : $d_{Is} = 0.3143$, $d_{IIs} = 0.3541$
Aspect ratio : $d_{Ia} = 0.2944$, $d_{IIa} = 0.3708$
Angle of second moment : $d_{Ias} = 0.1051$, $d_{IIas} = 0.3460$
Contrast : $d_{Ic} = 0.2944$, $d_{IIc} = 0.3687$
Entropy : $d_{Ie} = 0.0494$, $d_{IIe} = 0.3430$
Mean : $d_{IM} = 0.1857$, $d_{IIM} = 0.3541$

From the above examples, d_{I4} and d_{II4} of roundness produce the maximum values based on split rules. (Figure 3.2) shows an example of a feature parameter vector to find a split point by split rules.



$d_I = |P^1_L - P^1_R| = |5/24 - 9/17| = 0.3211$

$d_{II} = (q^1_L P^1_L + q^1_R P^1_R) = (5/14*5/24) + (9/14*9/17) = 0.4148$

(Figure 3.1) Example of the splitting rules; intervals and probabilities.

4. Experimental Results

To enable setup of a classification rule, we assume that training data is provided. What we would like to know is the proportion of errors made by this rule when it is up and running and classifying new objects without the benefit of knowing the true classifications. In this section, we describe the data sets and the results of object recognition by the proposed method.

Our image database consists of 356 randomly selected objects of images from current online shopping electronic catalogs on the Internet. The image objects are categorized by semantics, such as cups, and plates. In addition, our image database contains images which are scaled, rotated, and different posed objects.

As we described in [5], we have used image parameters such as surface regularity, roundness, form factor, aspect ratio, angle of orientation, contrast, and mean since these are image feature parameters. Especially, shape parameters distinctly represent image objects in our image database. In order to achieve minimum rates of error for object recognition, we performed experiments using feature parameters of images. We performed recognition with image objects in the training data sets and we evaluated the test data set.

To perform the experiments, we have raw datasets from the randomly selected objects. We performed classification based on a training set of image databases as in Figures 4.1 represent tree structure for plate and cup for dinnerware. It shows image parameters and decision variables to get to leaf nodes. We assigned the number of intervals as 30% of the total number of objects and the minimum size of intervals as 5% of the total number of objects.

We used 60% of the total number of objects for the training data set, and 40% of the total number of objects for the test data sets using Bootstrap [7] method because of the size of data. Figures 4.1 shows classification trees based on the training set. This figure shows image feature parameters, which are used to split subsets. Experiments have been implemented using Matlab for evaluation of the split rules.

The probability of leaf nodes represents the distribution of objects, which belong to the class. For example, if a leaf node shows 1.0, it means that a leaf node has 100% number of objects corresponding to a class. On the other hand, 0.0 means that a leaf node does not have any objects of a class. It represents the ratio of misclassified to classified objects in each leaf node.

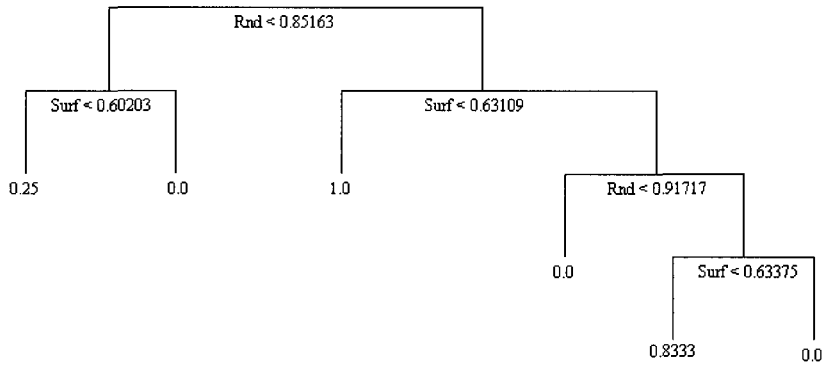
4.1 Comparison with Other Methods

In this section, we describe the experimental methodologies used in each of the next following sections. Typical measures to evaluate quality of classification trees include tree size and tree depth on training and test sets. These experiments were performed to evaluate our method compared to the exhaustive search methods such as CART[3] and S-plus tree[6].

4.2 Tree Size and Depth

Optimality of a decision tree may be measured in terms of size and depth[15]. It should be clear that it is desirable to build optimal trees in terms of one or more of these criteria.

<Table 4.1> shows a comparison between our method and other exhaustive search methods.[15, 13] mentioned that smaller and shallower decision trees imply better comprehensibility and computational efficiency. Comprehensibility typically decreases with increase in the tree size and complexity. Shallow trees are also more cost-effective, as



(Figure 4.1) Example of the tree structure for Plate and Cup. The probabilities of classification are given below each node.

〈Table 4.1〉 The classification tree size, and tree depth using our method compare to the exhaustive search methods on ten test sets. In experiments, our method used the number of intervals as 30% of the total number of objects and those numbers are averaged.

Methods	Measurements	
	Avg. Tree Size	Avg. Tree Depth
Our method	10.2	4.7
CART	16.8	5.1
S-plus tree	9.8	5.2

the depth of a tree is a measure of its classification cost. Tree size represents the number of leaf nodes, because the tree size starts at the root node and increases by one with each added split. In this experiment, we compare tree size and depth with those resulting from the exhaustive search methods.

<Table 4.1> gives the number of levels for each classifier on raw datasets. Ten test datasets were gathered from our image database. We classified, computed, and averaged those numbers. Comparing the tree size of our method and other methods, our method is smaller than CART, but similar to S-plus tree. In addition, our method is shallower than S-plus tree and CART. Therefore, the results show that our method is better than CART and S-plus tree. Overall quality of our classification method is better than other exhaustive search methods. We conclude that our method provides reasonable tree size and depth.

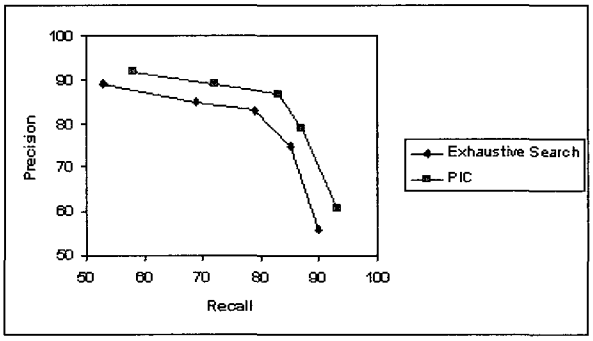
4.3 Precision and Recall

The receiver operating characteristic (ROC) curve enables us to analyze and visualize classification performance separately from assumptions about class distributions and error costs[17, 1, 2]. (Figures 4.2) and (Figures 4.3) show the ROC curves for our approach and exhaustive search approach, on training and test datasets respectively. Each of the points on the curves is for some specific value of the threshold, which is a probability of leaf node, for thresholds 95%, 85%, 75%, 65%, and 55%.

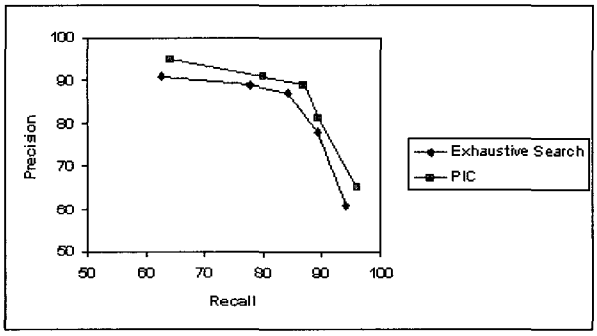
In order to evaluate the effectiveness, five sets were run through the classifier, and precision and recall scores averaged in <Table 4.2>. We compare precision, recall, and F-score compared to CART and S-plus tree. In prior work, the method in[12] experimented a method compared to CART and OC1. This method produced a higher precision score for the "Australian" dataset related to credit card. The precision of this method was 3% better than CART and 4% better than OC1. OC1 method in [16] shows comparison between CART and C4.5 on several

〈Table 4.2〉 Performance evaluation with Precision, Recall, and F-score with other methods on Training and Test sets. Our method used the number of intervals as 30% of the total number of objects.

Methods	Precision		Recall		F-score	
	Training	Test	Training	Test	Training	Test
Our method	85.9%	93.8%	91.6%	92.5%	.881	.931
CART	81.4%	87.2%	86.5%	89.1%	.839	.881
S-plus tree	79.5%	83.9%	85.9%	87.2%	.826	.855



(Figure 4.2) Precision vs Recall(60/40 split). The receiver operating characteristic curves on the Training datasets.



(Figure 4.3) Precision vs Recall(60/40 split). The receiver operating characteristic curves on the Test datasets.

different real-world datasets such as "Breast Cancer Diagnosis" and "Diabetes Diagnosis". The precision of OC1 was 1% better than others on "Breast Cancer Diagnosis" dataset, and 1% better than CART and 3% better than C4.5 on "Diabetes Diagnosis" dataset. Buntine [4] experimented with a classification method on several datasets such as "Medical data", "Pole Balancing data", and "Voting data". This method was compared to CART and C4.5. The precision of a method was from 2% to 3% better than others. CART[3] also experimented the comparison of the precision with nearest neighbor classification, CART produced 6% higher precision than nearest neighbor classification on "Waveform" data set.

In prior art, the improvement of the precision is from 1% to 6% or more differences of precision compared to other methods. As we can see from the table, the difference of precision score between our method and others is more than 4% in training sets and 6% in test sets, which is considered significant improvement in precision.

To evaluate the capability of classification, we calculated F-score to combine precision and recall. F-score shows that our method is much closer to 1 than other methods. Comparing F-score of our method and others, our method has a good capability of classifying objects. It is worth mentioning that the training and test data shown here both have an operating point on the ROC curve for which average Precision and Recall are 80% or above. The method demonstrates higher precision and recall scores than the exhaustive search method.

Therefore, we conclude that experimental results show that our approach provided better precision and recall scores with fewer splits, and our method demonstrates good capability of classification compared to other methods.

5. Concluding Remarks

In this paper, we have introduced a method to perform classification that are new splitting rules based on the probabilities of pre-assigned intervals, generated from binary tree splits to find split points.

The proposed method described how to find the maximal decision criteria based on new splitting rules. In addition, our method allows users to control the accuracy of a tree by adjusting the number of intervals and objects. However, we have a limitation of the small number of intervals. To avoid this problem, we experimented and selected the number of intervals as 30% of the total number of objects.

According to experimental results, our method demonstrates higher precision and recall scores against the exhaustive search methods. The results show that our method properly classified objects in image databases.

Our methodologies allowed one to search images based on image objects. Experimental results show that our approach provides reasonable accuracy with fewer splits, provides smaller and shallower tree. In addition, the experiments show that this probabilistic approach is suitable for solving object recognition problems.

References

[1] J. R. Beck and E. K. Schultz. The use of ROC curves in test performance evaluation. Arch Pathol Lab Med, 1986.

[2] P. Bradely. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 1997.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and Regression Trees. Wadsworth, 1984.

[4] Wray Buntine. Learning Classification Trees. *Statistics and Computing*, 1992.

[5] June-Suh Cho. Feature Extraction of Shape of Image Objects in Content-based Image Retrieval. *The KIPS Transactions : Part B,* 2003.

[6] L. A. Clark and D. Pregibon. Tree-based models, in J. M. Chambers and T. J. Hastie(eds), Statistical Models in S. Chapman and Hall, New York, 1993.

[7] B. Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association,* 1983.

[8] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. BOAT-Optimistic Decision Tree Construction. *In SIGMOD'99*, 1999.

[9] Johannes Gehrke, Raghu Ramakrishnan, and Venkatesh Ganti. RainForest - A Framework for Fast Decision Tree Construction of Large Datasets. *In Proceedings or the 24th VLDB Conference*, New York, 1998.

[10] Rodney M. Goodman and Padhraic J. Smyth. Decision tree design from a communication theory standpoint. *IEEE Transactions on Information Theory*, 1988.

[11] D. M. Hawkins and G. V. Kass. Automatic Interaction Detection. Cambridge University Press, 1982.

[12] George H. John. Robust linear discriminant trees. In *The 5th International Workshop on Artificial Intelligence and Statistics*, 1995.

[13] T. Lim and W. Loh. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. Dept. of Statistics in University of Wisconsin Technical Report 979, 1999.

[14] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica Sinica,* 1997.

[15] Sreerama K. Murthy. On Growing Better Decision Trees From Data. Ph.D. Thesis in Johns Hopkins University, 1995.

[16] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 1994.

[17] Foster Provost, Tom Fawcett, and Ron Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms. *In International Conference on Machine Learning,* 1998.

[18] J. R. Quinlan. C4.5 : Programs for Machine Learning. Morgan Kaufmann, 1993.

[19] R. Rastogi and K. Shim. PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. *In Proceedings of the 24nd VLDB Conference*, 1998.

조 준 서

email : jscho@hufs.ac.kr
1989년 경희대학교(학사)
1993년 New York University Computer Science(이학석사)
2001년 Rutgers University Computer Information Systems(경영학박사)
2000년~2001년 IBM T. J. Watson Research Center 연구원
2002년~2003년 CALTECH
2003년~현재 한국외국어대학교 경영학부 조교수
관심분야 : Multimedia Database, e-Business, Ubiquitous Computing

최 준 수

email : jschoi@kookmin.ac.kr
1984년 서울대학교 전기공학과(학사)
1986년 한국과학기술원 전산학과 (이학석사)
1986년~1990년 한국통신 전임연구원
1995년 New York University Computer Science(전산학박사)
1995년~1996년 한국통신 선임연구원
1996년~현재 국민대학교 자연과학대학 컴퓨터학부 부교수
관심분야 : Design and Analysis of Algorithms, Computer Graphics, Mobile Communication