A Heuristic Lazy Bayesian Rule Algorithm

Zhihai Wang
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia
zhihai.wang@infotech.monash.edu.au

Geoffrey I. Webb
School of Computer Science and Software Engineering
Monash University
Vic. 3800, Australia
webb@infotech.monash.edu.au

ABSTRACT

LBR has demonstrated outstanding classification accuracy. However, it has high computational overheads when large numbers of instances are classified from a single training set. We compare LBR and the tree-augmented Bayesian classifier, and present a new heuristic LBR classifier that combines elements of the two. It requires less computation than LBR, but demonstrates similar prediction accuracy.

1. INTRODUCTION

The naive Bayesian classifier [1] is known to be optimal and efficient for classification when all the attributes are mutually independent given the class and the required probabilities can be accurately estimated from the training data. Assume X is a finite set of instances, and $A = \{A_1, A_2, \cdots, A_n\}$ is a finite set of n attributes. An instance $x \in X$ is described by a vector $< a_1, a_2, \cdots, a_n >$, where a_i is a value of attribute A_i. C is called the class attribute. Prediction accuracy will be maximized if the predicted class $L(< a_1, a_2, \cdots, a_n >) = argmax_c(P(c| < a_1, a_2, \cdots, a_n >)$. Unfortunately, unless $< a_1, a_2, \cdots, a_n >$ occurs enough times within X, it will not be possible to directly estimate $P(c| < a_1, a_2, \cdots, a_n >)$ from the frequency with which each class $c \in C$ co-occurs with $< a_1, a_2, \cdots, a_n >$ within X. Bayes' theorem provides an equality that might be used to help estimate $P(c| < a_1, a_2, \cdots, a_n >)$ in such a circumstance:

$$P(c_i| < a_1, a_2, \cdots, a_n >) = \frac{P(c_i)P(< a_1, a_2, \cdots, a_n > |c_i)}{P(< a_1, a_2, \cdots, a_n >)}. \quad (1)$$

If the n attributes are mutually independent within each class value, then the probability is directly proportional to:

$$P(c_i| < a_1, a_2, \cdots, a_n >) \propto P(c_i) \prod_{k=1}^{n} P(a_k|c_i). \quad (2)$$

Classification selecting the most probable class as estimated using (1) and (2) is the well-known naive Bayesian classifier. The naive Bayesian classifier has been shown in many domains to be surprisingly accurate compared to alternatives including decision tree learning, rule learning, neural networks, and instance-based learning. Domingos and Pazzani [2] argued that the naive Bayesian classifier is optimal even when the independence assumption is violated, as long as the ranks of the conditional probabilities of classes given an example are correct. However, previous research has shown that semi-naive techniques and Bayesian networks that explicitly adjust the naive strategy to allow for violations of the independence assumption, can improve upon the prediction accuracy of the naive Bayesian classifier in many domains. This suggests that the ranks of conditional probabilities are frequently not correct. One approach to improving the naive Bayesian classifier is to relax the independence assumptions. Kononenko [3] proposed a semi-naive Bayesian classifier, which partitioned the attributes into disjoint groups and assumed independence only between attributes of different groups. Pazzani [4] proposed an algorithm based on the wrapper model for the construction of Cartesian product attributes to improve the naive Bayesian classifier. The naive Bayesian tree learner, $NBTree$[5], combined naive Bayesian classification and decision tree learning. It uses a tree structure to split the instance space into sub-spaces defined by the paths of the tree, and generates one naive Bayesian classifier in each sub-space. $NBTree$ frequently achieves higher accuracy than either a naive Bayesian classifier or a decision tree learner. Although $NBTree$ can alleviate the attribute inter-dependence problem of naive Bayesian classification to some extent, $NBTree$ suffers from the replication and fragment problem as well as the small disjunct problem due to the tree structure. Friedman, Geiger and Goldszmidt [6] compared the naive Bayesian method and Bayesian network, and showed that using unrestricted Bayesian networks did not generally lead to improvements in accuracy and even reduced accuracy in some domains. They presented a compromise representation, called Tree-Augmented naive Bayes (TAN), in which the class node directly points to all attributes nodes and an attribute node can have only at most one additional parent to the class node. Based on this presentation, they utilized the concept of mutual information to efficiently find the best tree-augmented naive Bayesian classifier. Zheng and Webb [7] proposed the lazy Bayesian rule (LBR) learning technique. LBR can be thought of as applying lazy learning techniques to naive Bayesian rule induction. At classification time, for each test example, it builds a most appropriate rule with a conjunction of conditions as its antecedent and a local naive Bayesian classifier as its consequent.

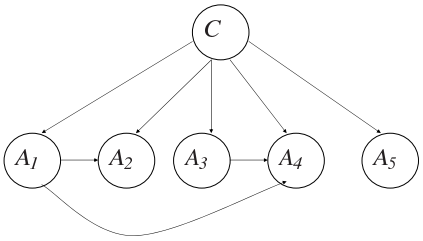

Figure 1: An example of a tree-augmented Bayesian network

Among these approaches of relaxing the attribute independence assumption, LBR has demonstrated remarkably low classification error rate. Zheng and Webb [7] experimentally compared LBR with a naive Bayesian classifier, a decision tree classifier, a Bayesian tree learning algorithm, a constructive Bayesian classifier, a selective naive Bayesian classifier, and a lazy decision tree algorithm in a wide variety of natural domains. In their extensive experiments, LBR obtained lower error than all the alternative algorithms. However, LBR is computationally inefficient if large numbers of objects are to classified from a single training set. In this paper, we compare the LBR and TAN techniques. A heuristic strategy for selecting attribute values to form the antecedent of a lazy Bayesian rule will be presented, which can be thought of as an application of TAN. Experimental comparisons and analysis of this heuristic lazy learning of Bayesian rules algorithm with the naive Bayesian classifier, LBR and TAN show that the heuristic algorithm has the almost same prediction accuracy as LBR with much lower computational requirements.

2. TAN AND LBR

Bayesian networks have been a popular medium for graphically representing and manipulating attribute interdependencies. Bayesian networks are directed acyclic graphs (DAG) that allow for efficient and effective representation of joint probability distributions over a set of random variables. Each vertex in the graph represents a random variable, and each edge represents direct correlations between the variables. Each variable is independent of its non-descendants given its parents in the graph. Bayesian networks provide a kind of direct and clear representation for the dependencies among the variables or attributes. A tree-augmented Bayesian network is a restricted form of Bayesian network [8], which can be defined by the following conditions:

- Each attribute has the class attribute as a parent;
- Attributes may have at most one other attribute as a parent.

Fig. 1 shows an example of a tree-augmented Bayesian network.

In a tree-augmented Bayesian network, a node without any parent, other than the class node, is called an *orphan*. Given a tree-augmented Bayesian network, if we extend arcs from node A_k to every orphan node A_i, then node A_k is said to be a *super parent*. For any node v, we denote its parents by $Parents(v)$. If v is an orphan, then $Parents(v) = \emptyset$.

LBR uses lazy learning to learn at classification time a single Bayesian rule for each instance to be classified. LBR is similar to $LazyDT$ (Lazy Decision Tree learning algorithms) [9], which can be considered to generate decision rules at classification time. For each instance to be classified, $LazyDT$ builds one rule that is most appropriate to the instance by using an entropy measurement. The antecedent of the rule is a conjunction of conditions in the form of attribute-value pairs. The consequent of the rule is the class to be predicted, being the majority class of the training instances that satisfy the antecedent of the rule. LBR can be considered as a combination of the two techniques $NBTree$ and $LazyDT$. Before classifying a test instance, it generates a rule (called a Bayesian rule) that is most appropriate to the test instance. Alternatively, it can be viewed as a lazy approach to classification using the following variant of Bayes theorem,

$$P(C_i|V_1 \wedge V_2) = P(C_i|V_2)P(V_1|C_i \wedge V_2)/P(V_1|V_2) \quad (3)$$

Here any test instance can be described by a conjunction of attribute values V, and V_1 and V_2 are any two conjunctions of attribute values such that each v_i from belongs to exactly one of V_1 or V_2. At classification time, for each instance to be classified, the attribute values in V are allocated to V_1 and V_2 in a manner that is expected to minimize estimation error. The antecedent of a Bayesian rule is the conjunction of attribute-value pairs from the set V_2. The consequent is a local naive Bayesian classifier created from those training instances that satisfy the antecedent of the rule. THis local naive Bayesian classifier only uses those attributes that belong to the set V_1. During the generation of a Bayesian rule, the test instance to be classified is used to guide the selection of attributes for creating attribute-value pairs. The values in the attribute-value pairs are always the same as the corresponding attribute values of the test instance. The objective is to grow the antecedent of a Bayesian rule that ultimately decreases the errors of the local naive Bayesian classifier in the consequent of the rule. Leave-one-out cross validation is used to select the attribute values to be moved to the left of a lazy Bayesian rule. The structure of a Bayesian network for a lazy Bayesian rule is shown in Fig. 2, here $V_1 = A_1, A_2, \cdots, A_k$ and $V_2 = A_{k+1}, A_{k+2}, \cdots, A_n$. The general form of this lazy Bayesian rule can be simply expressed as $(A_{k+1} \wedge A_{k+2} \wedge \cdots \wedge A_n) \rightarrow NaiveBayesClassifier(A_1, A_2, \cdots, A_k)$. Both LBR and TAN can be viewed as variants of naive Bayes that relax the attribute independence assumption. TAN relaxes this assumption by allowing each attribute to depend upon at most one other attribute in addition to the class. LBR allows an attribute to depend upon many other attributes, but all attributes depend upon the same set of other attributes.

3. DESCRIPTION OF HEURISTIC LAZY BAYESIAN RULE ALGORITHM

The principle cause of LBR's inefficiency when large numbers of instances are to be classified is the selection for each such instance of the attributes to place in the antecedent of the rule. Our strategy in the new algorithm is to move as much of this computation to training time as possible, performing as much of the computation as possible once only at the time when the training data is first analysed. To this end we seek at training time to identify attributes that should not be considered as candidates for inclusion in an

Table 1: The heuristic lazy Bayesian rule algorithm

ALGORITHM: HLBR $(X, V, C, test, \alpha)$
INPUT: 1) X is the set of training instances,
2) V is the set of attributes,
3) C is the set of class values,
4) T is a test instance,
5) α is the significance level.
OUTPUT: a predicted class for the test instance.

$Candidates = \emptyset$; /* The candidate atrributes */
$GlobalNB = NB$ trained using X, V and C;
$Errors$ = leave-one-out errors on X of $LocalNB$;
FOR each attribute a DO
$ThisErrors$ = leave-one-out errors on X of LBR with a as the antecedent;
IF $ThisErrors < Errors$
THEN $Candidates = Candidates + a$;
FOR each instance $test \in T$ DO
$Cond = true$;
$BestNB = GlobalNB$;
$BestErrors = Errors$;
REPEAT
FOR each A in $Candidates$ whose value v_A on $test$ isn't missing DO
X_{subset} = instances in X with $(A = v_A)$;
$TempNB = NB$ trained using $(V - \{A\})$ on X_{subset};
$TempErrors$ = (leave-one-out errors on X_{subset} of $TempNB$)
+ (errors from $Errors$ for instances in $(X - X_{subset})$);
IF (($TempErrors < BestErrors$) AND ($TempErrors$ is significantly lower than $Errors$)
THEN $BestErrors = TempErrors$;
$BestNB = TempNB$;
$ABest = A$;
IF (an $ABest$ is found)
THEN $Cond = Cond \wedge (ABest = v_{ABest})$;
$LocalNB = BestNB$;
$X_{training} = X_{subset}$ corresponding to $ABest$;
$V = V - \{ABest\}$;
$Errors$ = leave-one-out errors on $X_{training}$ of $LocalNB$;
ELSE EXIT from the REPEAT loop;
Classify $test$ using $LocalNB$;

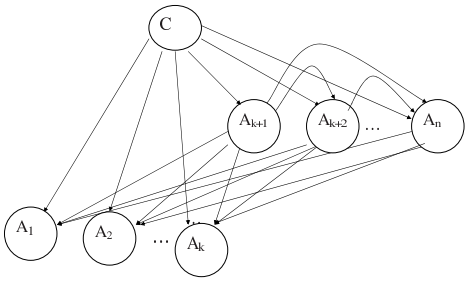

Figure 2: The structure of a Bayesian network for an example LBR

antecedent at classification time. To achieve this we perform leave-one-out cross validation for each attribute assessing the error when lazy Bayesian rules are formed using that and only that attribute in the antecedent. We restrict the candidates for consideration at classification time to those for which the cross validation error on this test is less than the cross validation error of naive Bayes. Our reasoning is that if there are harmful interdependencies between this and other attributes then this test will succeed. If there are no such harmful interdependencies then we should not consider the attribute as a candidate for inclusion in an antecedent. The heuristic lazy Bayesian rule algorithm is described in Table 1.

4. EXPERIMENTS

We compare the classification performance of four learning algorithms: the naive Bayesian classifier, *LBR*, *TAN* and our heuristic lazy Bayesian rule algorithm (*HLBR*). We use the naive Bayes classifier implemented in the Weka system, simply called Naive. We implemented a lazy Bayesian rule (*LBR*) learning algorithm and a tree-augmented Bayesian network (*TAN*) learning algorithm in the Weka system. All the experiments are run in the Weka system [10].

Thirty-five natural domains are used in the experiments shown in Table 2. Twenty-nine of these are all the data sets used in [7], the remaining are six larger data sets (German, Mfeat-mor, Satellite, Segment, Sign, and Vehicle). In Table 2, "Size" means the number of instances in a data set. "Class" means the number of values of a class attribute. "Attr." means the number of attributes, not including the class attribute. The error rate of each classifier on each domain is obtained by running 10-fold cross validation, and the random seed for 10-fold cross validation takes on the Weka default value. We also use the Weka default discretization method "weka.filters.DiscretizeFilter," an implementation of MDL discretization [11], as the discretization method for continuous values.

All experimental results for the error rates of the algorithms are shown in Table 3. The final two rows present the mean error across all data sets and the geometric mean error ratio. The latter measure is the geometric mean of the ratio for each data set of the error of respective algorithm divided by the error of HLBR. The geometric mean is used as the appropriate average for ratios. The average is at best a crude measure of overall performance as error rates on different data sets are incommensurable. The error ratio attempts to correct this problem by standardising the outcomes. Both

Table 2: Descriptions of Data

	Domain	Size	Class	Atts.
1	Annealing Processes	898	6	38
2	Audiology	226	24	69
3	Breast Cancer(Wisconsin)	699	2	9
4	Chess (KR-vs-KP)	3196	2	36
5	Credit Screening(Australia)	690	2	15
6	Echocardiogram	74	2	6
7	Germany	1000	2	20
8	Glass Identification	214	7	10
9	Heart Disease(Cleveland)	303	2	13
10	Hepatitis Prognosis	155	2	19
11	Horse Colic	368	2	22
12	House Votes 84	435	2	16
13	Hypothyroid Diagnosis	3163	2	25
14	Iris Classification	150	3	4
15	Labor Negotiations	57	2	16
16	LED 24(noise level=10%)	1000	10	24
17	Liver Disorders(bupa)	345	2	6
18	Lung Cancer	32	3	56
19	Lymphography	148	4	18
20	Mfeat-mor	2000	10	6
21	Pima Indians Diabetes	768	2	8
22	Post-Operative Patient	90	3	8
23	Primary Tumor	339	22	17
24	Promoter Gene Sequences	106	2	57
25	Satellite	6435	6	36
26	Segment	2310	7	19
27	Sign	12546	3	8
28	Solar Flare	1389	2	9
29	Sonar Classification	208	2	60
30	Soybean Large	683	19	35
31	Splice Junction Gene Seq.	3177	3	60
32	Tic-Tac-Toe End Game	958	2	9
33	Vehicle	846	4	18
34	Wine Recognition	178	3	13
35	Zoology	101	7	16

Table 3: Average Error Rate for Each Data Set

	Domain	Naive	LBR	TAN	HLBR
1	Annealing Processes	5.46	5.46	4.01	5.46
2	Audiology	29.20	29.20	29.20	29.20
3	Breast Cancer(Wisconsin)	2.58	2.58	2.58	2.58
4	Chess (KR-vs-KP)	12.36	3.57	5.07	3.85
5	Credit Screening(Australia)	15.07	14.64	14.35	14.64
6	Echocardiogram	27.48	27.48	28.24	27.48
7	Germany	24.60	24.70	24.80	24.60
8	Glass Identification	11.68	9.81	6.07	9.81
9	Heart Disease(Cleveland)	16.50	16.50	16.50	16.50
10	Hepatitis Prognosis	16.13	16.13	16.13	16.13
11	Horse Colic	20.11	19.29	18.48	19.29
12	House Votes 84	9.89	7.13	6.90	7.13
13	Hypothyroid Diagnosis	2.94	2.78	2.88	2.90
14	Iris Classification	6.67	6.67	6.00	6.67
15	Labor Negotiations	3.51	3.51	3.51	3.51
16	LED 24(noise level=10%)	24.50	24.70	24.50	24.70
17	Liver Disorders(bupa)	36.81	36.81	40.29	36.52
18	Lung Cancer	46.88	43.75	50.00	43.75
19	Lymphography	14.19	14.19	15.54	14.19
20	Mfeat-mor	30.65	29.95	30.10	29.90
21	Pima Indians Diabetes	25.00	24.87	25.39	24.87
22	Post-Operative Patient	28.89	28.89	30.00	28.89
23	Primary Tumor	48.97	49.85	49.85	49.85
24	Promoter Gene Sequences	8.49	8.49	8.49	8.49
25	Satellite	18.90	13.27	12.63	13.40
26	Segment	11.08	6.41	6.28	6.45
27	Sign	38.58	20.93	26.85	20.98
28	Solar Flare	18.57	15.69	16.85	15.62
29	Sonar Classification	25.48	25.96	23.56	28.37
30	Soybean Large	7.17	7.17	7.03	7.17
31	Splice Junction Gene Seq.	4.66	4.06	4.69	4.25
32	Tic-Tac-Toe End Game	29.54	14.61	28.81	13.99
33	Vehicle	39.48	31.44	31.68	31.44
34	Wine Recognition	3.37	3.37	3.37	3.37
35	Zoology	5.94	5.94	5.94	5.94
	Mean	**19.18**	**17.14**	**17.90**	**17.20**
	Geo. Mean	**1.14**	**0.99**	**1.02**	**1.00**

Table 4: Comparison of *LBR* to others

	WIN	LOSS	DRAW	p
Naive	16	4	15	0.012
TAN	15	11	9	0.557
HLBR	7	5	23	0.774

Table 5: Comparison of TAN to others

	WIN	LOSS	DRAW	p
Naive	17	9	9	0.168
LBR	11	15	9	0.557
HLBR	12	14	9	0.845

measures suggest that all of LBR, TAN, and HLBR enjoy substantially lower error than naive Bayes. The differences between LBR, TAN, and HLBR are much smaller, ordered from lowest to highest error LBR, then HLBR, then TAN.

The win/loss/draw record provides a more robust evaluation of relative performance over a large number of data sets. Tables 4, 5, and 6 present the win/loss/draw records for *LBR*, *TAN* and *HLBR*, respectively. This is a record of the number of data sets for which the nominated algorithm achieves lower, higher, and equal error to the comparison algorithm, measured to two decimal places. The final column presents the outcome of a two-tailed sign test. This is the probability that the observed outcome or more extreme would be obtained by chance if wins and losses were equiprobable. *LBR* and *HLBR* both achieve lower error than naive Bayes with frequency that is statistically significant at the 0.05 level. No win/loss/draw record indicates a significant difference in performance. This suggests that *LBR*, *HLBR* and *TAN* demonstrate comparable levels of error rate. *LBR* has a higher error rate than *TAN* in eleven data sets, and lower error rate in fifteen. *HLBR* has a higher error rate than *TAN* in twelve data sets, and lower error rate in fourteen. *LBR* has a lower error rate than the naive Bayes classifier in sixteen out of the thirty-five data sets, and a higher error rate in only four data sets. *HLBR* has a lower error rate than the naive Bayes classifier in seventeen out of the thirty-five data sets, and a higher error rate in only three data sets.

These results suggest that HLBR performs, in general, at a similar level of prediction accuracy to *LBR*. This comparable accuracy is obtained with far lower computation than LBR.

The runtimes on all datasets of *LBR* and *HLBR* are shown in Table 7. Both *LBR* and *HLBR* were run on a dual-processor 1.7GHz Pentium 4 Linux computer with 2GB RAM. Runtimes less than one second are recorded as 1 second. Note that there is considerable variance in run times on the machine on which the experiments were run. The run time of LBR was higher than that of HLBR on 19 data sets and lower on 8. We calculated the ratio derived by dividing the run time of LBR by the run time of HLBR for each data set. The appropriate form of average for such ratio values is the geometric mean. The geometric mean was 1.4, indicating a substantial average advantage to HLBR.

Table 6: Comparison of HLBR to others

	WIN	LOSS	DRAW	p
Naive	17	3	15	0.002
LBR	5	7	23	0.774
TAN	14	12	9	0.845

Table 7: Runtime of LBR and HLBR (Unit: Seconds)

	Domain	LBR	HLBR
1	Annealing Processes	177	94
2	Audiology	1028	470
3	Breast Cancer(Wisconsin)	16	3
4	Chess (KR-vs-KP)	18468	6516
5	Credit Screening(Australia)	66	40
6	Echocardiogram	1	1
7	Germany	164	56
8	Glass Identification	5	2
9	Heart Disease(Cleveland)	6	4
10	Hepatitis Prognosis	4	5
11	Horse Colic	15	18
12	House Votes 84	26	28
13	Hypothyroid Diagnosis	3905	399
14	Iris Classification	1	1
15	Labor Negotiations	1	1
16	LED 24(noise level=10%)	696	186
17	Liver Disorders(bupa)	2	2
18	Lung Cancer	3	20
19	Lymphography	5	5
20	Mfeat-mor	156	117
21	Pima Indians Diabetes	25	9
22	Post-Operative Patient	1	1
23	Primary Tumor	192	50
24	Promoter Gene Sequences	16	131
25	Satellite	48923	40624
26	Segment	4652	896
27	Sign	11821	9670
28	Solar Flare	103	94
29	Sonar Classification	32	155
30	Soybean Large	1172	247
31	Splice Junction Gene Seq.	12406	4391
32	Tic-Tac-Toe End Game	34	36
33	Vehicle	106	116
34	Wine Recognition	3	3
35	Zoology	5	5

5. CONCLUSIONS

We present a heuristic variant of the lazy Bayesian rules classifier. HLBR seeks to reduce classification time when there are large numbers of instances to be classified by identifying some attributes that should never be considered as candidates for inclusion in the antecedent of a lazy Bayesian rule. Our experimental results suggest that HLBR is successful in this aim while also managing to retain a similar level of classification accuracy to the original LBR.

6. REFERENCES

[1] Mitchell, T. M.: Machine Learning. New York: The

McGraw-Hill Companies, Inc.. (1997) 154-199

[2] Domingos, P., Pazzani, M.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In: Proceedings of the Thirteenth International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers, Inc. (1996) 105-112

[3] Kononenko, I.: Semi-Naive Bayesian Classifier. In: Proceedings of European Conference on Artificial Intelligence, (1991) 206-219

[4] Pazzani, M.: Constructive Induction of Cartesian Product Attributes. Information, Statistics and Induction in Science. Melbourne, Australia. (1996)

[5] Kohavi, R.: Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybird. In: Simoudis, E., Han, J.-W., Fayyad, U. M. (eds.): Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI Press. (1996) 202-207

[6] Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning, 29 (1997) 131–163

[7] Zheng, Z., Webb, G. I.: Lazy learning of Bayesian Rules. Machine Learning. Boston: Kluwer Academic Publishers.(2000) 1-35

[8] Keogh, E. J., Pazzani, M. J.: Learning Augmented Bayesian Classifiers: A Comparison of Distribution-Based and Classification-Based Approaches. In: Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics. (1999) 225-230

[9] Friedman, N., Kohavi, R., Yun, Y.: Lazy Decision Tree. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. Menlo Park, CA: The AAAI Press. (1996) 717-724

[10] Witten, I. H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Seattle, WA: Morgan Kaufmann Publishers. (2000)

[11] Fayyad, U. M., Irani, K. B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. (1993) 1022-1027