Toward Bayesian Classifiers with Accurate Probabilities

Charles Ling[1] and Huajie Zhang[2]

Department of Computer Science,
The University of Western Ontario
London, Ontario, Canada N6A 5B7
email: {ling, hzhang}@csd.uwo.ca

Abstract. In most data mining applications, accurate rankings and probability estimations are essential. However, many traditional classifiers aim at a high classification accuracy (or low error rate) only, even though they also produce probability estimations. Does high predictive accuracy imply a better ranking and probability estimation? Is there any better evaluation method for those classifiers than the classification accuracy, for the purpose of data mining applications? The answer is the AUC (Area Under the Curve) of ROC (Receiver Operating Characteristics) curves. We show that AUC of ROC provides a more discriminating evaluation for the ranking and probability estimation than the accuracy does. Further, we show that classifiers constructed to maximize the AUC score produce not only higher AUC values, but also higher classification accuracies. Our results are based on experimental comparison between error-based and AUC-based learning algorithms for TAN (Tree-Augmented Naive Bayes).

Key Words: Bayesian network learning, ROC evaluation, ranking and probability estimation, learning algorithms.

1 Introduction

Classification is the most important task in machine learning. In classification, a classifier is built from a set of training examples with class labels. A key performance measure of the classifier is its predictive accuracy (or error rate, which is one (1) minus the accuracy) on the training and testing examples. Predictive error rate is simply the percentage of the number of incorrectly classified examples versus the total number of examples. Many classifiers can also produce probability estimation or confidence of the prediction, but the error rate does not consider how "far-off" (be it 0.45 or 0.01) the prediction is from the target, but only the class with the largest probability estimation. Such error-based classifiers are optimal only under the following two assumptions: First, we care nothing more about the classification results; and second, different types of errors, such as false positive and false negative, are treated as equivalent.

In data mining applications, however, neither of those two assumptions is often true. For example, in direct marketing, we often need to promote the top X% of customers during gradual roll-out, or we often deploy different promotion strategies to customers with different likelihoods of buying some products. To accomplish these tasks, we need more than a classification of buyers and non-buyers. We need at least a ranking of customers in terms of their likelihoods of buying. Thus, a ranking is much more desired than just the classification. Is it true, however, that a classifier with a smaller error rate always has a better ranking?

In addition, the cost of errors, such as false positive and false negative, can be quite different. In direct marketing, for example, the cost of missing a valuable buyer is often much higher than the cost of promoting to a non-buyer. In this case, classifiers that produce small error rates do not optimize the business goal: which customers shall be promoted for the maximum profit?

Last, the probabilities on the prediction are often crucial for optimal decision making. For example, if for an example e, the cost of predicting class i when the true class of e is j is $C(i, j, e)$, the Bayes optimal prediction for e is the class i that minimizes the expected loss:

$$R(i|e) = \sum_j P(j|e)C(i, j, e). \tag{1}$$

$R(i|e)$ is the expected cost of predicting e to be class i. Clearly, Bayes optimal prediction requires accurate probability estimation, more than ranking and classification alone.

If we are aiming at accurate ranking or probability estimations from a classifier, one might naturally think that we must need true ranking or true probabilities in the training examples. In most scenarios, however, that is not possible. Most likely, what we are given is a dataset of examples with class labels. Thus, given only classification labels in training and testing sets, are there better methods than the error rate to evaluate classifiers that also produce probability estimations or ranking, for the purpose of data mining applications?

The answer is the ROC curve! ROC (Receiver Operating Characteristics) (Swets, 1988; Provost, Fawcett, & Kohavi, 1998) compares the classifiers' performance cross the entire range of class distributions and error costs. Details of ROC and related calculations can be in the Appendix, and we only give an intuitive explanation here. Figure 1 shows a plot of four ROCs, each representing one of the four classifiers, A through D. A ROC X is said to *dominate* another ROC Y if X is always above and to the left of Y. This means that that the classifier of X always has a lower expected cost than that of Y, over all possible error costs and class distributions. In this example, A and B dominate D.

However, often there is no clear dominating relation between two ROCs. For example, curves A and B are not dominating each other in the whole range. In those situations, the AUC (Area Under the Curve) of the ROC is a good "summary" for comparing the two ROCs. Clearly, if one ROC dominates the other, its AUC must be larger. Intuitively, if the AUC of one ROC is larger than the AUC of the other ROC, its probability estimation would likely to be better too. Bradley (1997) show that the AUC of ROC is a proper metric for the quality of classifiers averaged across all possible probability thresholds.

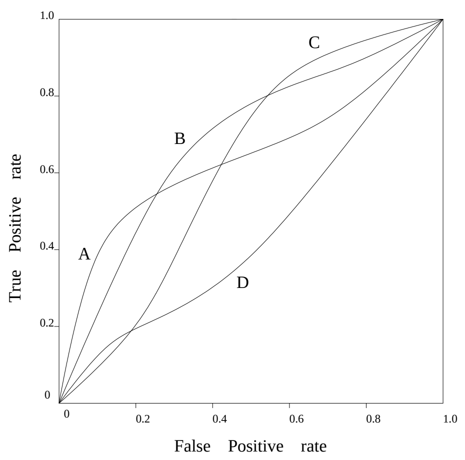

Fig. 1. An example of four ROC curves.

We believe that the AUC of the ROC should always be used as a more discriminating evaluation method than the error rate for classifiers that produce probabilities. For example, if we have two classifiers, 1 and 2, both producing probability for a set of 10 testing examples. Assume that both classifiers classify 5 of the 10 examples as positive, and the other 5 negative. If we rank the testing examples according to increasing probability, we get the two rank lists as in Table 1.

Clearly, both classifiers produce an error rate of 20% (one false positive and one false negative), thus the two classifiers are equivalent in terms of the error

Classifier 1	$- - - - +\| - + + + +$
Classifier 2	$+ - - - -\| + + + + -$

Table 1. An Example in which two classifiers have the same classification accuracy, but different AUC values.

rate. However, intuition tells us that Classifier 1 is better than Classifier 2, since overall positive examples are ranked higher in classifier 1 than 2. If we calculate AUC of the ROC, we obtain that AUC of classifier 1 is $\frac{24}{25}$, and AUC of classifier 2 is $\frac{16}{25}$. Clearly, AUC tells us that classifier 1 is indeed better than 2.

For a perfect classifier, which ranks all positive examples with higher probability than any negative examples, the AUC would be 1. In the worst case (the reverse of above), the AUC is equal to 0. If the ranking is random, then the AUC is 0.5. Note, however, that AUC of ROC is a global measure of a classifier, and is not always better than error rate. Counter examples can be found in Section 3.1.

Since AUC of ROC is a better evaluation method than the error rate for data mining algorithms that also produce probabilities, the next natural question is: Can we construct classifiers using AUC directly, rather than using error based matrix (such as entropy, error rate in cross-validation)? Clearly, we can use AUC in constructing most popular learning algorithms, such as Bayesian networks and decision trees. For example, in decision-tree learning, instead of using information gain (ratio) to choose the best attribute at the top of the (sub)tree, we can choose an attribute that produces the maximum AUC for the current tree under construction. Such decision trees would maximize AUC on the training data, and hopefully, on the testing as well (if overfitting is prevented).

In this paper, we study how to use AUC to construct Bayesian classifiers, and compare this strategy to several previous algorithms that based on error rate. We choose Bayesian networks since they are essentially approximating the underlying probability distribution, so it is more likely for AUC-constructed Bayesian networks to produce larger AUC values.

The rest of the paper is organized as follows: Section 2 reviews necessary background about Bayesian networks, and in particular, TAN, which our new AUC-based learning algorithm will construct. Section 3 proposes a new learning algorithm for AUC-based TAN learning, and compares it to three other learning algorithms by empirical experiments.

2 Review of Learning Simple Bayesian Networks

Bayesian networks (BNs) are probabilistic models that combine the probability theory and the graph theory. They represent causal and probabilistic relations (by arcs and conditional probability tables) among random variables (nodes) that are governed by the probability theory. Probabilistic inference can be made directly from Bayesian networks. Bayesian networks have been widely used in many applications, because they provide intuitive and causal representations of our real-world applications, and they are supported by a rigorous theoretical foundation.

Bayesian networks have often been used in classification. Assume A_1, A_2,$\cdots$, A_n are n attributes. An example e is represented by a vector $(a_1, a_2, , \cdots, a_n)$, where a_i is the value of A_i. Let C represent the classification variable that corresponds to the class, and c represent the value that C takes.

From the Bayes Rule, the probability of an example $e = (a_1, a_2, \cdots, a_n)$ being class c is

$$p(c|e) = \frac{p(a_1, a_2, \cdots, a_n|c)p(c)}{p(a_1, a_2, \cdots, a_n)}$$

e is classified into the most probable class c, i.e.,

$$g(e) = \max_c p(c|a_1, a_2, \cdots, a_n) \tag{2}$$

where $g(e)$ is called a Bayesian classifier.

The term $p(c|a_1, a_2, \cdots, a_n)$ is difficult to estimate. Assume all attributes are independent given the value of the class variable; that is,

$$p(a_1, a_2, \cdots, a_n|c) = \prod_{i=1}^{n} p(a_i|c)$$

the resulting $g(e)$ is then:

$$g(e) = \max_c p(c) \prod_{i=1}^{n} p(a_i|c) \tag{3}$$

$g(e)$ is called a Naive Bayesian classifier, or simply Naive Bayes (NB). Figure 2 (a) shows an example of Naive Bayes.

Because values of $p(a_i|c)$ can be estimated easily from the training examples, Naive Bayes is easy to construct. It is also, however, surprisingly effective (Kononenko, 1990) in classification. Unfortunately, the independence assumption is rarely true in real-world applications. Indeed, Naive Bayes is found to work poorly for regression problems (Frank, Trigg, Holmes, & Witten, 2000), and produces poor probability estimations (Bennett, 2000; Monti & Cooper, 1999). Therefore, researchers have extended the structure of Naive Bayes to represent dependencies among attributes[1]. Tree Augmented Naive Bayes (TAN) is such an extension, in which the classification node points directly to all attributes (as in Naive Bayes), but an attribute can have one parent from another attribute. Figure 2 (b) shows an example of TAN. TAN is a specific case of general Augmented Naive Bayesian networks (or simply ANB), where the classification node also points directly to all attributes, but where there is no limitation on arcs among attributes (except that they do not form any directed cycle). The general ANB is as powerful as general Bayesian networks.

[1] Most of the extensions, however, were aimed at improving the predictive accuracy, not at better probability estimations.

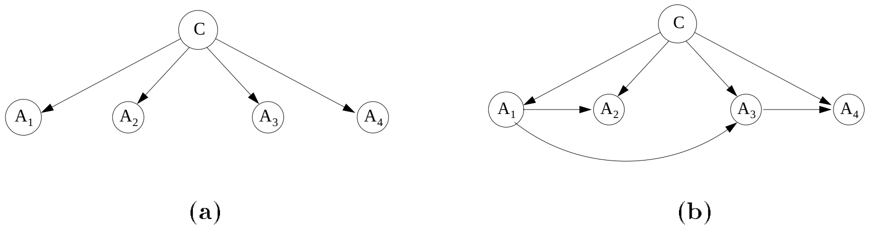

Fig. 2. (a) an example of Naive Bayes (b) an example of TAN

TAN is a nice trade-off between complexity of learning and representation power. In the past, a number of learning algorithms have been published on learning TAN. Some algorithms, such as ones by Chow and Liu (1968), Friedman, Geiger, and Goldszmidt (1997) are based on independence among attributes. Others, such as (Keogh & Pazzani, 1999), are based on minimizing the error rate. There is no previous work that learns TAN based on the AUC of the ROC. The questions we will answer in this paper are: Do error-based learning algorithms also produce accurate probability estimations? What about learning TAN by maximizing directly AUC? How do we compare these three learning algorithms (CI-based, error-based, and AUC-based) in terms of the error rate and the AUC?

3 Learning TAN with Accurate Probability Estimations

3.1 Error Rate vs AUC

As we have seen, the AUC of ROC can be viewed as a better measure than the error rate for classifiers that rank or produce probability, but a larger AUC does not always imply a lower error rate. Table 2 shows such a counter example. It is easy to obtain the AUC of Classifier 1 is 0.889, and 0.694 for Classifier 2. However, the error rate of Classifier 1 is 33.3%, but only 16.7% for Classifier 2.

Classifier 1	$- - - - + + \mid - - + + + +$
Classifier 2	$+ - - - - - \mid + + + + + -$

Table 2. A counter example in which one classifier has higher AUC but lower classification accuracy.

We will compare three different TAN learning algorithms in terms of the error rate and AUC values: one is CI-based, the second one is error-based, and the third is AUC-based. We will discuss each of these algorithms below.

3.2 CI-based TAN Learning Algorithms

CI (conditional independence) based learning algorithms have been studied by many researchers (Friedman et al., 1997). The basic idea is to detect the conditional dependence between two nodes by performing conditional independence tests based on the data, and then search for a network using the detected dependencies. Essentially, CI-based learning algorithms attempt to directly approximate the underlying probability distribution. Intuitively, they may tend to produce more accurate probability estimation.

Chow and Liu (1968)'s TAN learning algorithm is a typical CI-based algorithm. The idea here is to estimate dependencies between each pair of attributes, and find a maximum spanning tree based on the estimations in building the TAN. Friedman et al. (1997) extend (Chow & Liu, 1968)'s algorithm by using conditional mutual information between two attributes given the class variable. This function is defined as

$$I_P(X;Y|Z) = \sum_{x,y,z} P(x,y,z) log \frac{P(x,y|z)}{P(x|z)P(y|z)}$$

Friedman et al. (1997)'s algorithm is described below, and is used in our comparison. We refer to this algorithm as CI-TAN in our paper.

1. Compute $I_P(A_i, A_j|C)$ between each pair of variables, $i \neq j$.
2. Build a complete undirected graph in which the nodes are the attributes A_1, $\cdots$, A_n. Annotate the weight of an edge connecting A_i to A_j by $I_P(A_i; A_j|C)$.
3. Build a maximum weighted spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
5. Construct a TAN model by adding a node labeled by C and adding an arc from C to each A_i.

3.3 Error-based TAN Learning Algorithms

Error-based algorithms of learning Bayesian networks search for the network that minimizes the classification error (or maximizes the accuracy). They belong to scored-based algorithms since the accuracy is the score to maximize. In learning TAN, the SuperParent algorithm is a recent algorithm that searches for arcs by maximizing the accuracy via cross validation (Keogh & Pazzani, 1999). A node is called a SuperParent if we extend arcs from it to every orphan (nodes without parent). The algorithm SuperParent is depicted below.

1. Initialize network to Naive Bayes.

2. Evaluate the current classifier by its classification accuracy.
3. Consider making each node a SuperParent. Let A_{sp} be the SuperParent which increases accuracy the most.
4. Consider an arc from A_{sp} to each orphan. If the best such arc improves accuracy, then keep it and go to 2; else return the current classifier.

Since error-based algorithms aim directly at higher classification accuracy, intuitively, it may tend to produce a model with higher classification accuracy than CI-based algorithms. This has been verified by results in (Keogh & Pazzani, 1999), as well as in our experiment (see Section 3.5). However, the most intriguing question is: Would an AUC-based SuperParent algorithm produce better probability estimation measured by AUC, compared to error-based SuperParent algorithm?

3.4 AUC-based SuperParent algorithm

AUC-based Bayesian learning algorithms simply use AUC as the score to maximize via cross-validation during the search for arcs in learning TAN. Intuitively, since we search the best TAN structure based on higher AUC values, the resulting TAN would tend to produce higher AUC, or more accurate probability estimations.

We extend SuperParent algorithm in which AUC is used for evaluating current network instead of classification accuracy. This algorithm is called AUC-SuperParent in our paper. The algorithm is described below.

1. Initialize network to Naive Bayes.
2. Evaluate the current classifier in terms of its AUC.
3. Consider making each node a SuperParent. Let A_{sp} be the SuperParent which increases AUC the most.
4. Consider an arc from A_{sp} to each orphan. If the best such arc improves AUC, then keep it and go to 2; else return the current classifier.

3.5 Empirical Comparisons

The questions we are interested in are: Do AUC-based algorithms really result in classifiers with higher AUC? If AUC is a better and more discriminating measure than the error rate, would AUC-based learning algorithms also produce a lower error rate?

We will answer those questions by empirical experiments. We use twelve datasets from the UCI repository (Merz, Murphy, & Aha, 1997) to conduct our experiments. Table 3 lists the properties of the datasets we use in our experiment. Most of these datasets were also used in comparing SuperParent and TAN published by Keogh and Pazzani (1999).

We also included Naive Bayes in our experiment, since other algorithms are derived from it. Our experiments follow the procedure below:

Dataset	Attributes	Class	Instances
Australia	14	2	690
breast	10	2	683
cars	7	2	700
dermatology	34	6	366
ecoli	7	8	336
hepatitis	4	2	320
import	24	2	204
iris	5	3	150
pima	8	2	768
segment	19	7	1540
vehicle	18	4	846
vote	16	2	435

Table 3. Descriptions of the datasets used in our experiments.

1. The continuous attributes in the dataset are discretized by Fayyad and Irani (1993)'s entropy-based method.
2. For each dataset, run Naive Bayes, CI-based TAN (Friedman et al., 1997), SuperParent (Keogh & Pazzani, 1999), and our AUC-SuperParent with the 5-fold cross-validation, and obtain the AUC and classification accuracy on the testing set unused in the training.
3. Repeat 20 times and obtain an average AUC and the classification accuracy on the testing data.

Table 4 shows the experimental results of AUC values of four learning algorithms: Naive Bayes, CI-based TAN, SuperParent, and AUC-SuperParent. They are represented as NB, CI-TAN, SP, and AUC-SP in the table.

Dataset	**NB**	**CI-TAN**	**SP**	**AUC-SP**
Australia	75.2±0.30	75.2±0.48	74.7±0.44	76.2±0.48
breast	47.6±1.19	43.3±1.11	42.9±1.10	42.2±0.92
cars	93.7±0.17	93.4±0.24	96.1±0.16	97.1±0.12
dermatology	99.9±0.01	99.2±0.47	99.3±0.46	99.6±0.33
ecoli	99.3±0.06	98.9±0.08	99.2±0.07	99.2±0.08
hepatitis	62.5±0.70	62.4±0.60	61.3±0.73	62.5±0.70
import	99.2±0.10	97.8±0.19	99.0±0.14	99.2±0.11
iris	96.5±0.67	97.1±0.18	97.3±0.19	97.9±0.18
pima	77.4±0.44	75.8±0.51	77.2±0.46	76.7±0.50
segment	95.3±0.05	97.0±0.05	97.2±0.06	97.0±0.05
vehicle	91.1±0.49	92.4±0.71	93.3±0.39	93.2±0.42
vote	86.2±0.54	86.0±0.56	86.4±0.55	85.4±0.55

Table 4. Comparison of AUC for four algorithms.

We use a simple threshold, 1%, on average AUC values to judge if an algorithm outperforms another. The comparison of the four algorithms on these datasets is presented in Table 5. In the table, i-j-k means that the algorithm at the corresponding row wins in i datasets, loses in j datasets, and ties in k datasets, compared to the algorithm at the corresponding column.

We can see that the average AUC of AUC-SuperParent is slightly better (3 wins, 1 loss, and 8 ties) than error-based SuperParent and CI-TAN, and significantly better than Naive Bayes. There are four datasets in which AUC-SuperParent is higher than that of Naive Bayes for more than one percent, but only one dataset in which the reverse happens. Overall, AUC-SuperParent is best regarding the AUC evaluation. That confirms that using AUC directly to build Bayesian networks will result in a network with more accurate ranking or probability estimation. The tables also show that CI-based TAN does not perform

Algorithms	**NB**	**CI-TAN**	**SP**	**AUC-SP**
NB				
CI-TAN	2-3-7			
SP	3-2-7	3-1-8		
AUC-SP	5-1-6	3-1-8	3-1-8	

Table 5. Comparison of algorithms in terms of AUC. i-j-k in the table means that the algorithm at the corresponding row wins i times, loses j times, and ties k times, compared to the algorithm at the corresponding column.

well; it is even slightly worse than Naive Bayes. CI-based TAN is constructed with the goal of a good fit for conditional independence of all attributes with the data, not necessarily a good fit of the classification accuracy or its probability estimation, which is what we actually care to measure. This shows indirectly that if we want to learn a Bayesian network for a certain goal, the best bet is to search the network that maximizes a score of that goal. Ranking and probability estimations are important for data mining, thus, data mining models should be constructed to maximize the AUC value, not the predictive accuracy.

Since AUC is a more discriminating evaluation method compared to the accuracy, one may expect that the AUC-SuperParent would also have a higher predictive accuracy compared to error-based SuperParent. Table 6 shows the experimental results of the four learning algorithms on the classification accuracy on the testing set.

When we set the threshold to be 0.5%, the comparison of the four algorithms in terms of their classification accuracies is shown in Table 7. It indicates that AUC-SuperParent is significantly better, in terms of predictive accuracy, than Naive Bayes (6 wins, 1 loss, and 5 ties), CI-TAN (9 wins, 0 loss, and 3 ties), and SuperParent (5 wins, 2 losses, and 5 ties).

As we discussed earlier, AUC is a more discriminating evaluation criterion than the accuracy, thus, to build a TAN for the purpose of high classification

Dataset	NB	CI-TAN	SP	AUC-SP
Australia	76.1±0.39	76.7±0.32	76.0±0.30	76.6±0.39
breast	68.3±0.36	73.3±0.37	74.8±0.34	74.6±0.33
cars	86.1±0.29	85.4±0.37	90.0±0.27	90.8±0.25
dermatology	98.3±0.14	97.7±0.17	98.5±0.13	98.2±0.13
ecoli	96.9±0.20	96.1±0.23	96.8±0.21	96.7±0.21
hepatitis	71.0±0.48	70.5±0.42	70.3±0.48	71.0±0.48
import	97.0±0.24	93.6±0.37	96.7±0.28	96.7±0.23
iris	91.4±0.45	91.2±0.48	91.6±0.47	91.3±0.49
pima	71.9±0.40	70.5±0.46	71.5±0.39	71.0±0.44
segment	73.1±0.21	82.3±0.17	82.6±0.18	81.9±0.19
vehicle	82.0±0.26	89.3±0.23	89.4±0.22	90.0±0.23
vote	76.0±0.55	78.6±0.61	77.0±0.60	79.9±0.52

Table 6. Comparison of the classification accuracy.

Algorithms	NB	CI-TAN	SP	AUC-SP
NB				
CI-TAN	5-5-2			
SP	5-1-6	6-2-4		
AUC-SP	6-1-5	9-0-3	5-2-5	

Table 7. Comparison of algorithms in terms of classification accuracy.

accuracy, we should probably use AUC as well for the measure to maximize during the search of network structures.

4 Conclusion

In this paper, we investigate for TAN learning algorithms for the purpose of accurate probability estimation, often required in many applications of data mining. We show that AUC of ROC is a more discriminating measure of the quality of ranking or probability estimation. We also propose a new algorithm, AUC-SuperParent, for learning TAN by directly using AUC as the search criterion. By empirical experiments, we have obtained the following interesting results:

- AUC-based Bayesian network learning algorithms tend to produce more accurate ranking and probability estimation, than the error-based algorithms.
- AUC-based Bayesian network learning algorithms tend to produce significantly higher classification accuracy than the error-based algorithms.

We can thus conclude that AUC of ROC should be used both as an evaluation criterion, and as a scoring function, for data mining algorithms.

In our future research, we will study other methods for improving probability estimation such as smoothing, binning, and bagging. Another direction we are working on is to study other learning algorithms using AUC, such as AUC-based decision-tree learning algorithm.

5 Appendix

We review the basics of ROC and AUC here. See Provost and Fawcett (1997), Provost et al. (1998) for more details.

Let $\{P, N\}$ be the positive and negative instance classes, and let $\{\tilde{P}, \tilde{N}\}$ be the classifications produced by a classifier. Let $P(P|I)$ be the posterior probability that an instance I is positive. The true positive rate, TP, of a classifier is:

$$TP = P(\tilde{P}|P) \approx \frac{\text{positives correctly classified}}{\text{total positives}}.$$

The false positive rate, FP, of a classifier is:

$$FP = P(\tilde{P}|N) \approx \frac{\text{negatives incorrectly classified}}{\text{total negatives}}.$$

On a ROC graph, TP is plotted on the Y axis and FP is plotted on the X axis. In the ROC space, each classifier with a given class distribution and cost matrix is represented by a point (FP, TP) on the ROC. For a model that produces a continuous output, i.e., the probability estimation for Bayesian networks, TP and FP can vary as the threshold on the output varies between its extremes (0 and 1). The resulting curve is called the ROC curve. A ROC curve illustrates the tradeoff available with a given model, in which a point is recorded for a different cost and class distribution. When a ROC dominates another, then its classifier always has a lower expected cost than the other over all possible error costs and class distributions. However, if two ROCs do not dominate each other, then the AUC of the ROC can be used as a rough measure for the expected cost. Hand and Till (2001) showed that the AUC is equivalent to the probability that a randomly chosen negative example will have a smaller estimated probability of belonging to the positive class than a randomly chosen positive example. Therefore, the larger the AUC, the more likely that a negative example will not be misclassified. They give a simple formula for calculating AUC.

Bibliography

Bennett, P. N. (2000). Assessing the calibration of Naive Bayes' posterior estimates. In *Technical Report No. CMU-CS00-155.*

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Patten Recognition*, *30*, 1145–1159.

Chow, C. K., & Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, *14*, 462–467.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1027. Morgan Kaufmann.

Frank, E., Trigg, L., Holmes, G., & Witten, I. H. (2000). Naive Bayes for regression. *Machine Learning*, *41(1)*, 5–15.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Keogh, E. J., & Pazzani, M. J. (1999). Learning augmented Naive Bayes classifiers. In *Proceedings of the Seventh International Workshop on AI and Statistics.*

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In Wielinga, B. (Ed.), *Current Trends in Knowledge Acquisition.* IOS Press.

Merz, C., Murphy, P., & Aha, D. (1997). UCI repository of machine learning databases. In *Dept of ICS, University of California, Irvine.* http://www.www.ics.uci.edu/mlearn/MLRepository.html.

Monti, S., & Cooper, G. F. (1999). A Bayesian network classifier that combines a finite mixture model and a Naive Bayes model. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pp. 447–456. Morgan Kaufmann.

Provost, F., & Fawcett, T. (1997). Analysis and visulization of classifier performance: comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 43–48. AAAI Press.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International conference on Machine Learning*, pp. 445–453. Morgan Kaufmann.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*, 1285–1293.