Learning Limited Dependence Bayesian Classifiers

Mehran Sahami

Gates Building 1A, Room 126
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

Abstract

We present a framework for characterizing Bayesian classification methods. This framework can be thought of as a spectrum of allowable dependence in a given probabilistic model with the Naive Bayes algorithm at the most restrictive end and the learning of full Bayesian networks at the most general extreme. While much work has been carried out along the two ends of this spectrum, there has been surprising little done along the middle. We analyze the assumptions made as one moves along this spectrum and show the tradeoffs between model accuracy and learning speed which become critical to consider in a variety of data mining domains. We then present a general induction algorithm that allows for traversal of this spectrum depending on the available computational power for carrying out induction and show its application in a number of domains with different properties.

Introduction

Recently, work in Bayesian methods for classification has grown enormously (Cooper & Herskovits 1992) (Buntine 1994). Bayesian networks (Pearl 1988) have long been a popular medium for graphically representing the probabilistic dependencies which exist in a domain. It has only been in the past few years, however, that this framework has been employed with the goal of automatically learning the graphical structure of such a network from a store of data (Cooper & Herskovits 1992) (Heckerman, Geiger, & Chickering 1995). In this latter incarnation, such models lend themselves to better understanding of the domain in which they are employed by helping identify dependencies that exist between features in a database as well as being useful for classification tasks. A particularly restrictive model, the Naive Bayesian classifier (Good 1965), has had a longer history as a simple, yet powerful classification technique. The computational efficiency of this classifier has made it the benefactor of a number of research efforts (Kononenko 1991).

Although general Bayesian network learning as well as the Naive Bayesian classifier have both shown success in different domains, each has it shortcomings. Learning in the domain of unrestricted Bayesian networks is often very time consuming and quickly becomes intractable as the number of features in a domain grows. Moreover, inference in such unrestricted models has been shown to be NP-hard (Cooper 1987). Alternatively, the Naive Bayesian classifer, while very efficient for inference, makes very strong independence assumptions that are often violated in practice and can lead to poor predictive generalization. In this work, we seek to identify the limitations of each of these methods, and show how they represent two extremes along a spectrum of data classification algorithms.

Probabilistic Models

To better understand the spectrum we will present shortly for characterizing proabilistic models for classification, it is best to first examine the end points and then naturally generalize.

Bayesian Networks

Bayesian networks are a way to graphically represent the dependencies in a probability distribution by the construction of a directed acyclic graph. These models represent each variable (feature) in a given domain as a node in the graph and dependencies between these variables as arcs connecting the respective nodes. Thus, independencies are represented by the *lack* of an arc connecting particular variables. A node in the network for a variable X_i represents the probability of X_i conditioned on the variables that are immediate parents of X_i, denoted $\Pi(X_i)$. Nodes with no parents simply represent the prior probability for that variable.

In probabilistic classification we would ideally like to determine the probability distribution $P(C|\mathbf{X})$ where C is the class variable and $\mathbf{X}$ is the n-dimensional data vector $(x_1, x_2, ..., x_n)$ that represents an observed instance. If we had this true distribution available to us, we could achieve the theoretically optimal classication

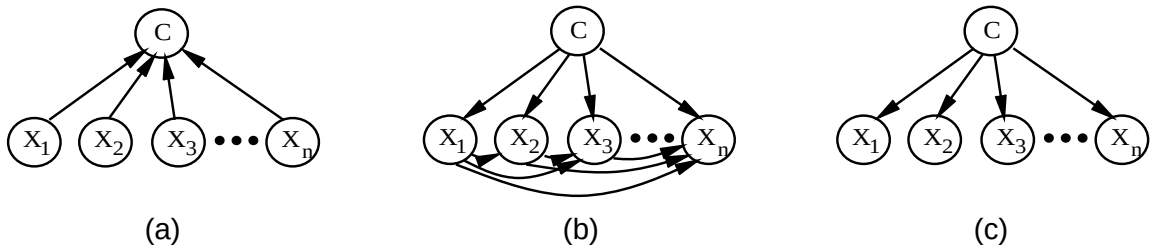

Figure 1: Bayesian networks representing (a) $P(C|\mathbf{X})$, (b) $P(C|\mathbf{X})$ after arc reversal, and (c) Naive Bayes.

by simply classifying each instance $\mathbf{X}$ into the class c_k for which $P(C = c_k|\mathbf{X})$ is maximized. This is known as *Bayes Optimal* classification. This general distribution can be captured in a Bayesian network as shown in Figure 1(a).

It is insightful to apply arc reversal (Shachter 1986) to the network in Figure 1(a) to produce the equivalent dependence structure in Figure 1(b). Here, $\Pi(X_i) = \{C, X_1, ..., X_{i-1}\}$. Now, we can see that the true complexity in such an unrestricted model (i.e. no independencies) comes from the large number of feature dependence arcs which are present in the model.

Since Bayesian networks allow for the modeling of arbitrarily complex dependencies between features, we can think of these models as lying at the most general end of a *feature dependence spectrum*. Thus, Bayesian networks have much representational power at the cost of computationally expensive learning and inference.

Naive Bayes

The Naive Bayesian classifier represents the most restrictive extreme in our spectrum of probabilistic classification techniques. As a Bayesian approach, it predicts the class c_k that maximizes $P(C = c_k|\mathbf{X})$, for a data vector $\mathbf{X}$, under the restrictive assumption that each feature X_i is conditionally independent of every other feature given the class label. In other words: $P(\mathbf{X}|C = c_k) = \prod_i P(X_i|C = c_k)$

The Naive Bayesian model is shown in Figure 1(c). In contrast to Figure 1(b), we see that the Naive Bayesian model allows for no arcs between feature nodes. We can think of the Naive Bayesian algorithm as being at the most restrictive end of the feature dependence spectrum, in that it strictly allows no dependencies between features given the class label.

We now formalize our notion of the spectrum of feature dependence in Bayesian classification by introducing the notion of k-dependence Bayesian classifiers. The proofs of the propositions we give subsequently are straight-forward and are omitted for brevity.

Definition 1 *A* k-dependence Bayesian classifier *is a Bayesian network which contains the structure of the Naive Bayesian classifier and allows each feature* X_i *to have a maximum of* k *feature nodes as parents. In other words,* $\Pi(X_i) = \{C, \mathbf{X_{d_i}}\}$ *where* $\mathbf{X_{d_i}}$ *is a set of at most* k *feature nodes, and* $\Pi(C) = \emptyset$.

Proposition 1 *The Naive Bayesian classifier is a* 0*-dependence Bayesian classifier.*

Proposition 2 *The full Bayesian classifier (i.e. no independencies) is a* $(N-1)$*-dependence Bayesian classifier, where* N *is the number of domain features.*

Geiger (1992) has defined the related notion of a *conditional dependence tree*. This notion is captured in our general framework as a 1-dependence Bayesian classifier. Friedman & Goldszmidt (1996) have also developed an algorithm, named TAN, which is similar to Geiger's method for inducing conditional trees. These algorithms generate optimal 1-dependence Bayesian classifiers, but provide no method to generalize to higher degrees of feature dependence.

By varying the value of k we can define models that smoothly move along the spectrum of feature dependence. If k is large enough to capture all feature dependencies that exist in a database, then we would expect a classifier to achieve optimal Bayesian accuracy if the "right" dependencies are set in the model[1].

The KDB Algorithm

We presently give an algorithm which allows us to construct classifiers at arbitrary points (values of k) along the feature dependence spectrum, while also capturing much of the computational efficiency of the Naive Bayesian model. Thus we present an alternative to the general trend in Bayesian network learning algorithms which do an expensive search through the space of network structures (Heckerman, Geiger, & Chickering 1995) or feature dependencies (Pazzani 1995).

[1]The question becomes one of determining if the model has allowed for enough dependencies to represent the *Markov Blanket* (Pearl 1988) of each feature. We refer the interested reader to Friedman & Goldszmidt (1996) and Koller & Sahami (1996) for more details

Our algorithm, called KDB, is supplied with both a database of pre-classified instances, DB, and the k value for the maximum allowable degree of feature dependence. It outputs a k-dependence Bayesian classifier with conditional probability tables determined from the input data. The algorithm is as follows:

1. For each feature X_i, compute mutual information, $I(X_i; C)$, where C is the class.
2. Compute class conditional mutual information $I(X_i; X_j|C)$, for each pair of features X_i and X_j, where $i \neq j$.
3. Let the used variable list, S, be empty.
4. Let the Bayesian network being constructed, BN, begin with a single class node, C.
5. Repeat until S includes all domain features
   - 5.1. Select feature X_{max} which is not in S and has the largest value $I(X_{max}; C)$.
   - 5.2. Add a node to BN representing X_{max}.
   - 5.3. Add an arc from C to X_{max} in BN.
   - 5.4. Add $m = \min(|S|, k)$ arcs from m distinct features X_j in S with the highest value for $I(X_{max}; X_j|C)$.
   - 5.5. Add X_{max} to S.
6. Compute the conditional probabilility tables infered by the structure of BN by using counts from DB, and output BN.

In this description of the algorithm, Step 5.4 requires that we add m parents to each new feature added to the model. To make the algorithm more robust, we also consider a variant where we change Step 5.4 to: Consider m distinct features X_j in S with the highest value for $I(X_{max}; X_j|C)$, and only add arcs from X_j to X_{max} if $I(X_{max}; X_j|C) > \theta$, where θ is a mutual information threshold. This allows more flexibility by not forcing the inclusion of dependencies that do not appear to exist when the value of k is set too high.

Another feature of our algorithm which makes it very suitable for data mining domains is its relatively small computational complexity. Computing the actual network structure, BN, requires $O(n^2mcv^2)$ time (dominated by Step 2) and calculating the conditional probability tables within the network takes $O(n(m + v^k))$ time, where n is the number of features, m is the number of data instances, c is the number of classes, and v is the maximum number of discrete values that a feature may take. In many domains, v will be small and k is a user-set parameter, so the algorithm will scale linearly with m, the amount of data in DB. Moreover, classifying an instance using the learned model only requires $O(nck)$ time.

We have recently become aware that Ezawa & Schuermann (1995) also have an algorithm similar in flavor to ours, but with some important differences, which attempts to discover feature dependencies directly using mutual information, as opposed to employing a general search for network structure.

Dataset	No. Classes	No. Features	Training Set Size	Testing Set Size
Corral	2	6	128	10-fold CV
LED7	10	7	3200	10-fold CV
Chess	2	36	3196	10-fold CV
DNA	3	180*	3186	10-fold CV
Vote	2	48*	435	10-fold CV
Text	3	3440	1084	454

Table 1: Datasets. *Denotes Boolean encoding.

Dataset	k	Accuracy KDB-orig	Accuracy KDB-θ
Corral	0	$88.4 \pm 10.5\%$	$88.4 \pm 10.5\%$
	1	$100.0 \pm 0.0\%$	$100.0 \pm 0.0\%$
	2	$96.7 \pm 5.8\%$	$96.7 \pm 5.8\%$
	3	$88.4 \pm 17.2\%$	$100.0 \pm 0.0\%$
LED7	0	$72.9 \pm 2.1\%$	$72.9 \pm 2.1\%$
	1	$73.1 \pm 3.9\%$	$73.0 \pm 2.9\%$
	2	$73.5 \pm 2.3\%$	$72.9 \pm 2.4\%$
	3	$73.2 \pm 2.3\%$	$73.4 \pm 1.3\%$
Chess	0	$86.2 \pm 1.9\%$	$86.2 \pm 1.9\%$
	1	$93.9 \pm 1.3\%$	$93.8 \pm 1.4\%$
	2	$95.1 \pm 1.2\%$	$95.5 \pm 1.6\%$
	3	$94.9 \pm 1.1\%$	$95.3 \pm 1.2\%$
DNA	0	$94.0 \pm 0.9\%$	$94.0 \pm 0.9\%$
	1	$94.0 \pm 1.6\%$	$94.1 \pm 1.1\%$
	2	$95.3 \pm 1.2\%$	$95.6 \pm 1.1\%$
	3	$93.3 \pm 0.9\%$	$95.5 \pm 1.8\%$
Vote	0	$90.2 \pm 3.8\%$	$90.2 \pm 3.8\%$
	1	$92.6 \pm 3.6\%$	$92.1 \pm 5.3\%$
	2	$92.3 \pm 3.5\%$	$93.5 \pm 4.1\%$
	3	$93.0 \pm 2.2\%$	$94.0 \pm 3.2\%$
Text	0	87.0%	87.0%
	1	87.9%	87.4%
	2	87.0%	88.3%
	3	86.8%	86.8%

Table 2: Classification accuracies for KDB.

Results

We tested KDB on five datasets from the UCI repository (Murphy & Aha 1995) as well as a text classification domain with many features (a small subset of the Reuters Text database (Reuters 1995)). These datasets are described in Table 1. Specifically, we wanted to measure if increasing the value of k above 0 would help the predictive accuracy of the induced models (i.e. compare the dependence modeling capabilities of KDB with Naive Bayes). Moreover, we wanted to see if we could uncover various levels of dependencies that we know exist in a few artificial domains by seeing how classification accuracy varied with the value of k. We also tested the modified KDB algorithm which employs a mutual information threshold, θ. In these trials we set $\theta = 0.03$, which was a heuristically determined value. The results of our experiments are given

in Table 2 with KDB-orig refering to the original algorithm and KDB-θ refering to the variant using the mutual information threshold.

The two artificial domains, Corral and LED7, were selected because of known dependence properties. The Corral dataset represents the concept $(A \wedge B) \vee (C \wedge D)$ and thus is best modeled when a few feature dependencies are allowed, as is borne out in our experimental results (higher accuracies when $k > 0$). The LED7 dataset, on the other hand, contains no feature dependencies when conditioned on the class. Our algorithm helps discover these independencies, as reflected by the similar accuracy rates when $k = 0$ and $k > 0$.

In the real-world domains we find that modeling feature dependencies very often improves classification results. This is especially true for the KDB-θ algorithm, where classification accuracies when $k > 0$ are almost always greater than or equal the $k = 0$ (Naive Bayes) case. In the Chess ($k = 1, 2, 3$), Vote ($k = 2, 3$) and DNA ($k = 2, 3$) domains, these improvements are statistically significant (t-test with $p < 0.10$). Moreover, by noting how the classification accuracy changes with the value of k we get a notion of the degree of feature dependence in each domain. For example, in both Chess and DNA, we see large jumps in accuracy when going from $k = 0$ to $k = 2$ and that there is no gain when $k = 3$, thus indicating many low-order interactions in these domains.

It is important to note that the Boolean encoding of the Vote and DNA domains has introduced some feature dependencies into the data, but such represetational issues (which are often unknown to the end user of a data mining system) also argue in favor of methods that can model such dependencies when they happen to exist. Also worth noting is the fact that as k grows, we must estimate a larger probability space (more conditioning variables) with the same amount of data. This can cause our probability estimates to become more inaccurate and lead to an overall decrease in predictive accuracy, as is seen when going from $k = 2$ to $k = 3$ in many of the domains. The KDB-θ algorithm is less prone to this effect, but it is still not impervious. In many data mining domains, however, we may have the luxury of have a great deal of data, in which case this degradation effect will not be as severe. Nevertheless, these results indicate that we can model domains better in terms of classification accuracy and get an idea for the underlying dependencies in the domain, two critical applications of data mining.

In future work, we seek to automatically identify good values for k for a given domain (possibly through employing cross-validation) and better motivate the value of the θ threshold. Also, comparisons with other Bayesian network learning methods are planned.

Acknowledgements We thank Moises Goldszmidt for his comments on an earlier version of this paper. This work was supported by ARPA/NASA/NSF under a grant to the Stanford Digital Libraries Project.

References

Buntine, W. 1994. Operations for learning with graphical models. *JAIR* 2:159–225.

Cooper, G. F., and Herskovits, E. 1992. A bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9:309–347.

Cooper, G. F. 1987. Probabilistic inference using belief networks is NP-Hard. Technical Report KSL-87-27, Stanford Knowledge Systems Laboratory.

Ezawa, K. J., and Schuermann, T. 1995. Fraud/uncollectible debt detection using a bayesian network learning system. In *UAI-95*, 157–166.

Friedman, N., and Goldszmidt, M. 1996. Building classifiers using bayesian networks. In *AAAI-96*.

Geiger, D. 1992. An entropy-based learning algorithm of bayesian conditional trees. In *UAI-92*, 92–97.

Good, I. J. 1965. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. M.I.T. Press.

Heckerman, D.; Geiger, D.; and Chickering, D. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20:197–243.

Koller, D., and Sahami, M. 1996. Toward optimal feature selection. In *Proceedings of the Thirteenth Int. Conference on Machine Learning*.

Kononenko, I. 1991. Semi-naive bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning*, 206–219. Pitman.

Murphy, P. M., and Aha, D. W. 1995. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Pazzani, M. J. 1995. Searching for dependencies in bayesian classifiers. In *Proceedings of the Fifth Int. Workshop on AI and Statistics*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann.

Reuters. 1995. Reuters document collection. ftp://ciir-ftp.cs.umass.edu/pub/reuters1.

Shachter, R. D. 1986. Evaluating influence diagrams. *Operations Research* 34(6):871–882.