Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood

Abstract

Bayesian networks are a powerful probabilistic representation, and their use for classification has received considerable attention. However, they tend to perform poorly when learned in the standard way. This is attributable to a mismatch between the objective function used (likelihood or a function thereof) and the goal of classification (maximizing accuracy or conditional likelihood). Unfortunately, the computational cost of optimizing structure and parameters for conditional likelihood is prohibitive. In this paper we show that a simple approximation – choosing structures by maximizing conditional likelihood while setting parameters by maximum likelihood – yields good results. On a large suite of benchmark datasets, this approach produces better class probability estimates than naïve Bayes, TAN, and generatively-trained Bayesian networks.

1. Introduction

The simplicity and surprisingly high accuracy of the naïve Bayes classifier have led to its wide use, and to many attempts to extend it. In particular, naïve Bayes is a special case of a Bayesian network, and learning the structure and parameters of an unrestricted Bayesian network would appear to be a logical means of improvement. However, Friedman et al. found that naive Bayes easily outperforms such unrestricted Bayesian network classifiers on a large sample of benchmark datasets. This explanation was that the scoring functions used in standard Bayesian network learning attempt to optimize the likelihood of the entire data, rather than just the conditional likelihood of the class given the attributes. Such scoring results in suboptimal choices during the search process whenever the two functions favor differing changes to the network. The natural solution would then be to use conditional likelihood as the objective function. Unfortunately, Friedman et al. observed that, while maximum likelihood parameters can be efficiently computed in closed form, this is not true of conditional likelihood. The latter must be optimized using numerical methods, and doing so at each search step would be prohibitively expensive. Friedman et al. thus abandoned this avenue, leaving the investigation of possible heuristic alternatives to it as an important direction for future research. In this paper, we show that the simple heuristic of setting the

parameters by maximum likelihood while choosing the structure by conditional likelihood is accurate and efficient.

Friedman et al. chose instead to extend naive Bayes by allowing a slightly less restricted structure (one parent per variable in addition to the class) while still optimizing likelihood. They showed that TAN, the resulting algorithm, was indeed more accurate than naive Bayes on benchmark datasets. We compare our algorithm to TAN and naive Bayes on the same datasets, and show that it outperforms both in the accuracy of class probability estimates, while outperforming naive Bayes and tying TAN in classification error.

2. Bayesian Networks

A Bayesian network encodes the joint probability distribution of a set of v variables, $\{x_1, ..., x_v\}$, as a directed acyclic graph and a set of conditional probability tables (CPTs). Each node corresponds to a variable, and the CPT associated with it contains the probability of each state of the variable given every possible combination of states of its parents. The set of parents of x_i, denoted π_i, is the set of nodes with an arc to x_i in the graph. The structure of the network encodes the assertion that each node is conditionally independent of its non-descendants given its parents. Thus the probability of an arbitrary event $X = (x_1, ..., x_v)$ can be computed as $P(X) = \prod_{i=1}^{v} P(x_i|\pi_i)$. In general, encoding the joint distribution of a set of v discrete variables requires space exponential in v; Bayesian networks reduce this to space exponential in $max_{i\in\{1,...,v\}}|\pi_i|$.

2.1 Learning Bayesian Networks

Given an i.i.d. training set $D = \{X_1, ..., X_d, ..., X_n\}$, where $X_d = (x_{d,1}, ..., x_{d,v})$, the goal of learning is to find the Bayesian network that best represents the joint distribution $P(x_{d,1}, ..., x_{d,v})$. One approach is to find the network B that maximizes the likelihood of the data of (more conveniently) its logarithm:

$$LL(B|D) = \sum_{d=1}^{n} logP_B(X_d) = \sum_{d=1}^{n}\sum_{i=1}^{v} logP_B(x_{d,i}|\pi_{d,i}) \quad (1)$$

When the structure of the network is known, this reduces to estimating p_{ijk}, the probability that variable i is in state k given that its parents are in state j, for all i, j, k.

When there are no examples with missing values in the training set and we assume parameter independence, the maximum likelihood estimates are simply the observed frequency estimates $\hat{p_{ijk}} = n_{ijk}/n_{ij}$, where n_{ijk} is the number of occurrences in the training set of the kth state of x_i with the jth state of its parents, and n_{ij} is the sum of n_{ijk} over all k.

Since on average adding an arc never decreases likelihood on the training data, using the log likelihood as the scoring function can lead to severe overfitting. This problem can be overcome in a number of ways. The simplest one, which is often surprisingly effective, is to limit the number of parents a variable can have. Another alternative is to add a complexity penalty to the log-likelihood. For example, the MDL method minimizes $MDL(B|D) = \frac{1}{2} m \log n - LL(S|D)$, where m is the number of parameters in the network. In both these approaches, the parameters of each candidate network are set by maximum likelihood, as in the known-structure case. Finally, the full Bayesian approach maximizes the *Bayesian Dirichlet (BD)* score:

$$\begin{aligned} P(B_S, D) =& P(B_S) P(D|B_S) \\ =& P(B_S) \prod_{i=1}^{v} \prod_{j=1}^{q_i} \frac{\Gamma(n'_{ij})}{\Gamma(n'_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(n'_{ijk} + n_{ijk})}{\Gamma(n'_{ijk})} (1) \end{aligned} \quad (2)$$

where B_S is the structure of network B, $\Gamma()$ is the gamma function, q_i is the number of states of the Cartesian product of x_i's parents, and r_i is the number of states of x_i. $P(B_S)$ is the prior probability of the structure, which Heckerman et al. set to an exponentially decreasing function of the number of differing arcs between B_S and the initial (prior) network. Each multinomial distribution for x_i given a state of its parents has an associated Dirichlet prior distribution with parameters n'_{ijk}, with $n'_{ij} = \sum_{k=1}^{r_i} n'_{ijk}$. These parameters can be thought of as equivalent to seeing n'_{ijk} occurrences of the corresponding states in advance of the training examples. In this approach, the network parameters are not set to specific values; rather, their entire posterior distribution is implicitly maintained and used. The BD score is the result of integrating over this distribution.

2.2 Bayesian Network Classifiers

The goal of classification is to correctly predict the value of a designated discrete *class* variable $y = x_v$ given a vector of *predictors* or *attributes* $(x_1, ..., x_{v-1})$. If the performance measure is accuracy (i.e., the fraction of correct predictions made on a test sample), the optimal prediction for $(x_1, ..., x_{v-1})$ is the class that maximizes $P(y|x_1, ..., x_{v-1})$. If we have a Bayesian network for $(x_1, ..., x_v)$, these probabilities can be computed by inference over it. In particular, the *naïve Bayes classifier* is a Bayesian network where the class has no parents and each attribute has the class as its sole parent. Friedman et al.'s TAN algorithm uses a variant of the Chow and Liu method to produce a network where each variable has one other parent in addition to the class. More generally, a Bayesian network learned using any of the methods described above can be used as a classifier. All of these are *generative* models in the sense that they are learned by maximizing the log likelihood of the entire data being generated by the model, $LL(B|D)$, or a related function. However, for classification purposes only the *conditional log likelihood* $CLL(B|D)$ of the class given the attribute is relevant, where

$$CLL(B|D) = \sum_{d=1}^{n} \log P_B(y_d|x_{d,1}, ..., x_{d,v-1}) \quad (3)$$

Notice $LL(B|D) = CLL(B|D) + \sum_{d=1}^{n} \log P_B(x_{d,1}, ..., x_{d,v-1})$. Maximizing $LL(B|D)$ can lead to underperforming classifiers, particularly since in practice the contribution of $CLL(B|D)$ is likely to be swamped by the generally much larger (in absolute value) $\log P_B(x_{d,1}, ..., x_{d,v-1})$ term. A better approach would presumably be to use $CLL(B|D)$ by itself as the objective function. This would be a form of *discriminative* learning, because it would focus on correctly discriminating between classes. The problem with this approach is that, unlike $LL(B|D)$ (Equation 1), $CLL(B|D) = \sum_{d=1}^{n} \log[P_B(x_{d,1}, ..., x_{d,v-1}, y)/P_B(x_{d,1}, ..., x_{d,v-1})]$ does not decompose into a separate term for each variable, and as a result there is no known closed form for the optimal parameter estimates. When the structure is known, locally optimal estimates can be found by a numeric method such as conjugate gradient with line search, and this is what Greiner and Zhou's ELR algorithm does. When the structure is unknown, a new

gradient descent is required for each candidate network at each search step. The computational cost of this is presumably prohibitive.

3. The BNC Algorithm

We now introduce BNC, an algorithm for learning the structure of a Bayesian network classifier by maximizing conditional likelihood. BNC is similar to the hill climbing algorithm of Heckerman et al. except that it uses the conditional log likelihood of the class as the primary objective function. BNC starts from an empty network, and at each step considers adding each possible new arc (i.e., all those that do not create cycles) and deleting or reversing each current arc. BNC pre-discretizes continuous values and ignores missing values in the same way that TAN does.

We consider two versions of BNC. The first, $BNC-nP$, avoids overfitting by limiting its networks to a maximum of n parents per variable. Parameters in each network are set to their maximum likelihood values. The network is then scored using the conditional log likelihood $CLL(B|D)$ (Equation 3). The rationale for this approach is that computing maximum likelihood parameter estimates is extremely fast, and, for an optimal structure, they are asymptotically equivalent to the maximum conditional likelihood ones.

The second version, $BNC-MDL$, is the same as $BNC-nP$, except that instead of limiting the number of parents, the scoring function

$CMDL(B|D) = \frac{1}{2} m \log n - CLL(S|D)$ is used, where m is the number of parameters in the network and n is the training set size.

The goal of BNC is to provide accurate class probability estimates. If correct class predictions are all that is required, a Bayesian network classifier could in principle be learned simply by using the training-set accuracy as the objective function (together with some overfitting avoidance scheme). While trying this is an interesting item for future work, we note that even in this case the conditional likelihood may be preferable, because it is a more informative and more smoothly varying measure, potentially leading to an easier optimization problem.