Methods’ details for the **bnclassify** package

Bojan Mihaljevic, Concha Bielza, Pedro Larranaga

2020-01-20

Abstract

This vignette provides details on the underlying methods and documents implementation specifics, especially where they differ from or are undocumented in the original paper. It complements the “overview” vignette.

Contents

1 Introduction 1

2 Structure learning 1
   2.1 Chow-Liu for one-dependence estimators ................................. 1
   2.2 TAN HC and TAN HC SP ................................................. 2

3 Parameter learning 2
   3.1 Bayesian parameter estimation ............................................. 2
   3.2 Exact model averaging for naive Bayes .................................. 2
   3.3 Weighting to Alleviate the Naive Bayes Independence Assumption .... 3
   3.4 Attribute-weighted naive Bayes .......................................... 4

4 Prediction 4

1 Introduction

All notation and acronyms used here are introduced in vignette("overview", package="bnclassify").

See the remaining vignettes:

- vignette("overview", package="bnclassify") provides details on the implemented methods.
- ?bnclassify provides a concise overview of the package, listing main functionalities and functions.
- vignette("introduction", package="bnclassify") provides details on the implemented methods.

2 Structure learning

2.1 Chow-Liu for one-dependence estimators

The CL-ODE algorithm by [Friedman et al., 1997] adapts the Chow-Liu [Chow and Liu, 1968] algorithm in order to find the maximum likelihood TAN model in time quadratic in \( n \). Since the same
method can be used to find ODE models which maximize decomposable penalized log-likelihood scores, \texttt{bnclassify} uses it to maximize Akaike’s information criterion (AIC) \cite{Akaike1974} and BIC \cite{Schwarz1978}. While maximizing likelihood will always render a TAN, i.e., a network with \(n-1\) augmenting arcs, maximizing penalized log-likelihood may render a FAN, since the inclusion of some arcs might degrade the penalized log-likelihood score.

Note that when data is incomplete \texttt{bnclassify} does not necessarily return the optimal (with respect to penalized log-likelihood) ODE. Namely, that requires the computationally expensive calculation of the sufficient statistics \(N_{ijk}\) which maximize parameter likelihood; instead, \texttt{bnclassify} approximates these statistics with the \textit{available case analysis} heuristic (see Section 3).

2.2 TAN HC and TAN HC SP

TAN HC and TAN HC SP may evaluate equivalent structures at each step. Adding valid arcs \(X_i \rightarrow X_j\) and \(X_j \rightarrow X_i\) results in identical structures because of tree structure of the features subgraph. Namely, \(|Pa(X) \setminus C| \leq 1\) for each \(X\) and thus we can only add the arc \(X_i \rightarrow X_j\) if \(Pa(X) = \{C\}\). Thus, adding an arc \(X_i \rightarrow X_j\) introduces no v-structures into the network, and both \(X_i \rightarrow X_j\) and \(X_j \rightarrow X_i\) only remove the independence between \(X_i\) and \(X_j\). The two obtained networks thus correspond to identical factorizations of the joint distribution.

To avoid scoring equivalent structures, at each step we selected the \(X_i \rightarrow X_j\) such that \(X_i\) (that is, its column name in the data set) is alphabetically before \(X_j\). A preferable implementation would be to select the arc randomly.

3 Parameter learning

3.1 Bayesian parameter estimation

\texttt{bnclassify} only handles discrete features. With fully observed data, it estimates the parameters with maximum likelihood or Bayesian estimation, according to Equation 2 in the “overview” vignette, with a single \(\alpha\) for all local distributions. With incomplete data it uses \textit{available case analysis} \cite{Pigott2001} and substitutes \(N_{ij}\) in Equation 2 in the “overview” vignette with \(N_{ij} = \sum_{k=1}^{r_i} N_{ijk}\), i.e., with the count of instances in which \(Pa(X_i) = j\) and \(X_i\) is observed:

\[
\theta_{ijk} = \frac{N_{ijk} + \alpha}{N_{ij} + r_i\alpha}.
\]

3.2 Exact model averaging for naive Bayes

The MANB parameter estimation method corresponds to exact Bayesian model averaging over the naive Bayes models obtained from all \(2^n\) subsets of the \(n\) features, yet it is computed in time linear in \(n\). The implementation in \texttt{bnclassify} follows the online appendix of Wei et al. [2011], extending it to the cases where \(\alpha \neq 1\) in Equation~(3.1).

The estimate for a particular parameter \(\theta_{ijk}^{MANB}\) is:

\[
\theta_{ijk}^{MANB} = \theta_{ijk}P(G \perp \perp X_i \mid D) + \theta_{ik}P(G \perp \perp X_i),
\]
where \( P(G_{C \perp X_i} \mid D) \) is the local posterior probability of an arc from \( C \) to \( X_i \), whereas \( P(G_{C \perp X_i}) = 1 - P(G_{C \perp X_i} \mid D) \) is that of the absence of such an arc (which is equivalent to omitting \( X_i \) from the model), while \( \theta_{ijk} \) and \( \theta_{ik} \) are the Bayesian parameter estimates obtained with Equation~(3.1) given the corresponding structures (i.e., with and without the arc from \( C \) to \( X_i \)).

Using Bayes’ theorem,

\[
P(G_{C \perp X_i} \mid D) = \frac{P(G_{C \perp X_i})P(D \mid G_{C \perp X_i})}{P(G_{C \perp X_i})P(D \mid G_{C \perp X_i}) + P(G_{C \perp X_i})P(D \mid G_{C \perp X_i})}.
\]

Assuming a Dirichlet prior with hyperparameter \( \alpha = 1 \) in Equation~3.1, Equation~(6) and Equation~(7) in the online appendix of Wei et al. [2011] give formulas for \( P(D \mid G_{C \perp X_i}) \) and \( P(D \mid G_{C \perp X_i}) \):

\[
P(D \mid G_{C \perp X_i}) = \prod_{j=1}^{rc} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!,
\]

\[
P(D \mid G_{C \perp X_i}) = \frac{(r_i - 1)!}{(N_i + r_i - 1)!} \prod_{k=1}^{r_i} N_{ik}!,
\]

where \( N_{i,k} = \sum_{j=1}^{rc} N_{ijk} \). Noting that the above are special cases of Equation~(8) in Dash and Cooper [2002], we can generalize this for any hyperparameter \( \alpha > 0 \) as follows:

\[
P(D \mid G_{C \perp X_i}) = \prod_{j=1}^{rc} \frac{\Gamma(r_i \alpha)}{\Gamma(N_{ij} + r_i \alpha)} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha)}{\Gamma(\alpha)},
\]

and

\[
P(D \mid G_{C \perp X_i}) = \frac{\Gamma(r_i \alpha)}{\Gamma(N_i + r_i \alpha)} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ik} + \alpha)}{\Gamma(\alpha)}.
\]

Following Wei et al. [2011], \textit{bnclassify} assumes that the local prior probability of an arc from the class to a feature \( X_i \), \( P(G_{C \perp X_i}) \), is given by the user. The prior of a naive Bayes structure \( G \), with arcs from the class to \( a \) out of \( n \), features and no arcs to the remaining \( n - a \) features is, then,

\[
P(G) = P(G_{C \perp X_i})^a (1 - P(G_{C \perp X_i}))^{(n-a)}. \quad (1)
\]

Note that \textit{bnclassify} computes the above in logarithmic space to reduce numerical errors.

### 3.3 Weighting to Alleviate the Naive Bayes Independence Assumption

The WANBIA [Zaidi et al., 2013] method updates naive Bayes’ parameters with a single exponent ‘weight’ per feature. The weights are computed by optimizing either the conditional log-likelihood or the mean root squared error of the predictions. \textit{bnclassify} implements the conditional log-likelihood
optimization as described in the original paper, namely optimizing it with the L-BFGS [Zhu et al., 1997] algorithm, with its gradient $g$ given by

$$ g_i = \sum_{j=1}^{N} \left( \log P(X_i = x_i^{(j)} | c^{(j)}) - \sum_{c \in C} P(c | \mathbf{x}; \mathbf{w}) \log P(X_i = x_i^{(j)} | c) \right), $$

where the probabilities are those estimated with maximum likelihood, i.e., without taking weights into account, whereas $P(c | \mathbf{x}; \mathbf{w})$ takes weights into account. This corresponds to discriminative learning of parameters, as a discriminative, rather than generative, objective function is optimized.

If $X_i$ is unobserved for some instance $j$, that is, $x_i^{(j)} = \text{NA}$, then we replace $P(X_i = x_i^{(j)} | c^{(j)})$ and $P(X_i = x_i^{(j)} | c)$ with 1 in Equation 2 (as a leaf in the Bayesian network, an unobserved $X_i$ does not affect conditional log-likelihood).

### 3.4 Attribute-weighted naive Bayes

The AWNB parameter estimation method is intended for the naive Bayes but in bnclassify it can be applied to any model. It exponentiates the conditional probability of a predictor,

$$ P(\mathbf{X}, C) \propto P(C) \prod_{i=1}^{n} P(X_i | \text{Pa}(X_i))^{w_i}, $$

reducing or maintaining its effect on the class posterior, since $w_i \in [0, 1]$ (note that a weight $w_i = 0$ omits $X_i$ from the model, rendering it independent from the class.). This is equivalent to updating parameters of $\theta_{ijk}$ given by Equation (3.1) as

$$ \theta_{ijk}^{\text{AWNB}} = \frac{\theta_{ijk}^{w_i}}{\sum_{k=1}^{r_i} \theta_{ijk}^{w_i}}, $$

and plugging those estimates into Equation 1 in the “overview” vignette. Weights $w_i$ are computed as

$$ w_i = \frac{1}{M} \sum_{t=1}^{M} \frac{1}{\sqrt{d_{ti}}}, $$

where $M$ is the number of bootstrap [Efron, 1979] subsamples from $\mathcal{D}$ and $d_{ti}$ is the minimum testing depth of $X_i$ in an unpruned classification tree learned from the $t$-th subsample ($d_{ti} = 0$ if $X_i$ is omitted from $t$-th tree).

### 4 Prediction

bnclassify implements prediction for augmented naive Bayes models with complete data. This amounts to multiplying the corresponding entries in the local distributions and is done in logarithmic space, applying the \textit{log-sum-exp} trick before normalizing, in order to reduce the chance of underflow.
With incomplete data this cannot be done and therefore `bnclassify` uses the `gRain` package [Højsgaard, 2012] to perform exact inference. Such inference is time-consuming and, therefore, wrapper algorithms can be very slow when applied on incomplete data sets.

References


