bayesclass
A Package for Learning Bayesian Network Classifiers

Bojan Mihaljevic, Pedro Larrañaga, and Concha Bielza

Computational Intelligence Group
School of Computer Science, Technical University of Madrid

July 11th 2013
Outline

1. Introduction
2. Learning Algorithms in bayesclass
3. Discrete Bayesian Network Classifiers in R
4. bayesclass Features
5. A Sample Session
6. Conclusion
The bayesclass package

- Algorithms for learning Bayesian network classifiers from discrete/categorical data
- Integrates with other packages to provide easy assessment of predictive performance, graph plotting, etc.
- Not on CRAN yet
  - Not all functions well-documented nor consistent (e.g. in argument naming)
Bayesian Network Classifiers (BNC)

- A BNC is a Bayesian network (BN) applied to a supervised classification task
- The predictive features $X$ and class $C$ are random variables
- Factorizes the joint probability $p(c, x)$ according to the BN

$$p(c, x) = p(c | \text{pa}(c)) \prod_{i=1}^{n} p(x_i | \text{pa}(x_i))$$

- And assigns an instance to the most likely class

$$c^* = \arg \max_c p(c | x) = \arg \max_c p(c, x)$$

- Many different models exist (naive Bayes, tree augmented naive Bayes, AODE, etc.)
Naive Bayes (Minsky, 1961)

\[ p(c|x) \propto p(c) \prod_{i=1}^{n} p(x_i|c) \]

Features are independent given the class

Attribute Weighted Naive Bayes (Hall, 2007)

- A weight \( w_i \in [0, 1] \) for each feature
- Can reduce a feature’s relevance (\( w_i = 0 \Rightarrow X_i \) is irrelevant)

\[ p(c|x) \propto p(c) \prod_{i=1}^{n} p(x_i|c)^{w_i} \]

- Relevance inversely proportional to dependence on other features
- Dependency \( \propto \) minimum depth of testing in an unpruned decision tree (0 if not tested)

Bagging is used to stabilize the estimates \( w_i = \frac{\sum_{j=1}^{M} \frac{1}{\sqrt{d_{ij}}}}{M} \), where \( M \) is the number of trees and \( d_{ij} \) the minimum depth of \( X_i \) in \( j \)-th tree
Adjusted Probability Naive Bayes Classifier (Webb et al., 1998)

\[ p(c|x) \propto w_c p(c) \prod_{i=1}^{n} p(x_i|c) \]

- Linear adjustments \((w_c \in [0, \infty))\) applied to Naive Bayes’ estimate of \(p(c|x)\)
- Hill-climbing search which maximizes resubstitution accuracy
- Search proceeds if best improvement is unlikely to have been obtained by chance
Selective Naive Bayes (Langley et al., 1994)

A naive Bayes over a subset of features

\[ p(c|x) \propto p(c)p(x_F|c) = p(c) \prod_{i \in F} p(x_i|c), \]

where the pruned feature set is \( X_F, F \subseteq \{1, \ldots, n\} \)

Algorithms

Forward Sequential Selection (Langley et al., 1994)

- A forward search which maximizes predictive accuracy (estimated by cross-validation)

Filter Forward Sequential Selection (Blanco et al., 2005)

- Keep features that are not independent from the class as deemed with a \( \chi^2 \) test of independence
Semi-naive Bayes (Pazzani, 1996)
Feature dependencies within disjoint subsets

\[ p(c|x) \propto p(c) \prod_{j \in Q} p(x_{S_j}|c) \]

\( S_j \subseteq \{1, \ldots, n\}, \bigcup_{j \in Q} S_j \subseteq \{1, 2, \ldots, n\} \) and \( S_j \cap S_l = \emptyset, j \neq l \)

Algorithms

Backward sequential elimination and joining algorithm (BSEJ) (Pazzani, 1996)
- A variant of the backward search which maximizes predictive accuracy

Filter forward sequential selection and joining (Blanco et al., 2005)
- Based on the forward search heuristic
- At each step several candidate related feature subsets are considered
- The candidate subset \( X_{S_{new}} \) with the lowest p-value of the test of \( \chi^2 \) independence of \( X_{S_{new}} \) and \( C \) is included in the model...
- ...as long the p-value is below a threshold \( \alpha \)
Learning Algorithms

Tree Augmented Naive Bayes (TAN) (Friedman et al., 1997)

\[
p(c|x) \propto p(c)p(x_r|c) \prod_{i \neq r} p(x_i|x_{j(i)}, c)
\]

\(X_r\) is the root of the tree

Selective TAN (STAN) (Blanco et al., 2005)

\[
p(c|x) \propto p(c) \prod_{i \in R} p(x_i|c) \prod_{i \in F \setminus R} p(x_i|x_{j(i)}, c)
\]

\(X_F\) is the pruned feature set, \(R \subseteq F\) are the tree roots, and \(\{X_{j(i)}\} = Pa(X_i) \setminus C, i \notin R\), are the parent features of \(X_i\)

- Omit features independent of class (FSS)
- Black-list edges between class-conditionally independent features (\(\chi^2\) test)
- Find the MWST(s) from the feasible edges
Augmented Semi-naive Bayes (Mihaljevic et al., 2013)

Dependences among semi-naive Bayes’ subsets of related features

\[
p(c|x) \propto p(c) \prod_{i \in R} p(x_{S_i}|c) \prod_{i \in Q \setminus R} p(x_{S_i}|x_{j(i)}, c),
\]

where \( S_j \subseteq \{1, \ldots, n\} \) is the \( j \)-th feature subset, \( Q = \{1, \ldots, K\} \) are the indices of feature subsets, \( \cup_{j \in Q} S_j \subseteq \{1, 2, \ldots, n\} \) and \( S_j \cap S_l = \emptyset, \ j \neq l \), \( R \subseteq Q \) are indices of feature subsets that are root(s) of the augmenting trees(s), and \( \{X_{j(i)}\} = Pa(X_{S_i}) \setminus C \).

A simple idea:

- Learn a semi-naive Bayes with BSEJ and apply STAN on it
- Promising empirical results
The *bnlearn* Package

- Naive Bayes and tree augmented naive Bayes
- Maximum likelihood and Bayesian estimation of parameters
- Nice features:
  - Prediction, cross-validation of a learning algorithm and of a network structure
  - Graph plotting, arc black/white listing, etc.
  - Only handles complete data (for both learning and prediction)

Other Implementations of Naive Bayes

- The *e1071*, *CORElearn* packages
Features:

- 10 algorithms for learning Bayesian network classifiers from discrete/categorical data
- Maximum likelihood and Bayesian estimation of parameters (through gRain package)
  - A unique hyperparameter \(\alpha\) for all parameters
- Predictive performance assessment (through caret)
- Does handle incomplete data (although in a naive way)
- Other features: graph plotting (Rgraphviz), cross-validation assessment of a network structure
Load the bayesclass package and the car data set:

```r
library(bayesclass)
data(car)
str(car)
```

'data.frame': 1728 obs. of 7 variables:

```
$ buying : Factor w/ 4 levels "low","med","high",...:
4 4 4 4 4 4 4 4 4 4
$ maint : Factor w/ 4 levels "high","low","med",...:
4 4 4 4 4 4 4 4 4 4
$ doors : Factor w/ 4 levels "2","3","4","5more":
1 1 1 1 1 1 1 1 1 1
$ persons : Factor w/ 3 levels "2","4","more":
1 1 1 1 1 1 1 1 1 1 2 ...
$ lug_boot: Factor w/ 3 levels "big","med","small":
3 3 3 2 2 2 1 1 1 3
$ safety : Factor w/ 3 levels "high","low","med":
2 3 1 2 3 1 2 3 1 2 ...
$ class : Factor w/ 4 levels "unacc","acc",...:
1 1 1 1 1 1 1 1 1 1 ...
```

The car data frame contains seven discrete variables. We predict the class variable and the remaining are predictive features.
Learn a Bayesian network classifier calling the `bnc` function:

- First argument: the training data. `bnc` assumes that the last column is the class
- The `learner` argument tells which learning algorithm to use

```
tan.car <- bnc(car, learner = "tan", smooth = 0.01)
```

`bnc` returns a `bayesclass` object. View its network structure with `plot`:

```
plot(tan.car)
```
Learning algorithm-specific parameters are specified with the `lrn_args` argument:

```r
stan.car <- bnc(car, learner = "stan",
               lrn_args = list(alpha = 0.1),
               smooth = 0.01)
```

List the features in a model with the `features` function:

```r
features(stan.car)
```

```
[1] "buying"  "maint"  "persons"  "safety"  "lug_boot"
```
Use a `bayesclass` object to get class or class posterior probability predictions for new data. Set the result argument of `predict` to "prob" or "class". This returns the class posterior for the first six rows in `car`:

```r
pred.tan <- predict(tan.car, car, result = "prob")
head(pred.tan)
```

```
  unacc     acc  good  vgood
 [1,]  1 1.639e-08 9.852e-06 1.722e-05
 [2,]  1 2.377e-09 3.337e-12 9.870e-06
 [3,]  1 7.375e-09 1.335e-08 2.367e-12
 [4,]  1 2.591e-08 1.750e-05 2.795e-12
 [5,]  1 9.777e-09 1.247e-08 2.246e-08
 [6,]  1 1.338e-08 1.167e-08 1.103e-11
```
Get a resampling estimate of predictive performance with the `assess` function. Two repetitions of stratified 5-fold cross-validation for the TAN:

```r
pred_performance <- assess(tan.car, car, k = 5, repeats = 2)
pred_performance
```

```
  learner smooth Accuracy  Kappa AccuracySD KappaSD
  1    tanbc  0.01   0.9427   0.8764   0.008508  0.01837
```

Leftmost columns specify the learning algorithm. Also use `assess` for a paired comparison of learning algorithms. Supply a list of `bayesclass` objects as the first argument:

```r
compare <- assess(list(stan.car, tan.car), car, k = 5, seed = 0)
```

```
  alpha learner smooth Accuracy  Kappa AccuracySD KappaSD
  1   0.1    stanbc  0.01    0.9184    0.8268   0.02284   0.04490
  2    NA     tanbc  0.01    0.9433    0.8776   0.01264   0.02703
```
Future Work

- Real-valued features
- More flexible priors
- Incomplete data: structural EM
- Of course, publish on CRAN
[Blanco et al., 2005] Blanco, R., Inza, I., Merino, M., Quiroga, J., and Larrañaga, P.
Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M.
Bayesian network classifiers.
*Machine Learning, 29:131-163.*

[Hall, 2007] Hall, M.
A decision tree-based attribute weighting filter for naive Bayes.
Further Reading II

[Langley et al., 1994] Langley, P. and Sage, S.
Induction of selective Bayesian classifiers.

[Mihaljevic et al., 2013] Mihaljevic, B., Larrañaga, P., and Bielza, C.
Augmented Semi-naive Bayes Classifier.

[Pazzani, 1996] Pazzani, M.
Constructive induction of cartesian product attributes.
Adjusted probability naive Bayesian induction