Discrete Bayesian network classifiers in R and an application to Neuroscience

Bojan Mihaljevic

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

February 7th, 2013
Outline

1 Introduction
   - Bayesian Network Classifiers

2 Bayesian Network Classifiers in R
   - The R Environment for Statistical Computing
   - Discrete Bayesian Network Classifiers

3 bayesClass R Package
   - Overview
   - Learning Algorithms
   - Cross-validation for Bayesian Network Classifiers
   - Summary and Pending Tasks

4 Application to Neuroscience
   - Nomenclature of GABAergic Interneurons
   - A Practical Approach
   - Further Insights into Classification of Interneurons
   - Conclusions and Future Work
Bayesian Network Classifiers

- Using a Bayesian network to perform supervised classification
- One node is the class, the rest are predictor variables
- We assign an incoming observation to the most probable a posteriori class

\[ \arg \max_c p(c|x) = \arg \max_c p(x, c) \]

- We factorize \( p(x, c) \) according to a Bayesian network
Bayesian Network Classifiers

Learning in Restricted Structure Space

- Unconstrained Bayesian networks are generally not good classifiers
- The special 'status' of the class node should be considered
- Fixing the class as root node and parent of all predictors gives rise to augmented Naive Bayes models

\[ p(x, c) = p(c)p(x|c) \]

Figure: Naive Bayes
Learning Bayesian Network Classifiers

**Structure**
- Wrapper: Evaluate a network according to its **predictive performance**
- Filter: Evaluate an edge according to **statistics** computed from data

**Parameters**
- Parameters estimated from data using **MLE** or **Bayesian estimation**
- Some proposals extend Naive Bayes with **additional parameters**

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

- \(w_c\) are class weights and \(w_i\) attribute weights
- Weights are learned with **filter** or alertwrapper techniques
Outline

1 Introduction
   - Bayesian Network Classifiers

2 Bayesian Network Classifiers in R
   - The R Environment for Statistical Computing
   - Discrete Bayesian Network Classifiers

3 bayesClass R Package
   - Overview
   - Learning Algorithms
   - Cross-validation for Bayesian Network Classifiers
   - Summary and Pending Tasks

4 Application to Neuroscience
   - Nomenclature of GABAergic Interneurons
   - A Practical Approach
   - Further Insights into Classification of Interneurons
   - Conclusions and Future Work
Open-source programming environment
Considered the *lingua franca* of data analysis and statistical computing
Increasingly popular with the *machine learning* community
User contributed packages through a platform called CRAN

*Figure:* Logo of the R project
Discrete Bayesian Network Classifiers in R

### Packages

- **bnlearn**
  - Learning general Bayesian networks
  - Naive and Tree Augmented Bayes
  - Conditional independence tests, prediction,...but **only for complete data**

- **e1071**
  - Various statistical functions
  - Naive Bayes

- **gRain**
  - Inference
  - Can be used to predict with networks learned with **bnlearn**

### RWeka

- An interface to the Weka Machine Learning environment
- Naive and Tree Augmented Bayes, Lazy Bayesian Rules, NBTree
1 Introduction
   - Bayesian Network Classifiers

2 Bayesian Network Classifiers in R
   - The R Environment for Statistical Computing
   - Discrete Bayesian Network Classifiers

3 bayesClass R Package
   - Overview
   - Learning Algorithms
   - Cross-validation for Bayesian Network Classifiers
   - Summary and Pending Tasks

4 Application to Neuroscience
   - Nomenclature of GABAergic Interneurons
   - A Practical Approach
   - Further Insights into Classification of Interneurons
   - Conclusions and Future Work
Availability of Bayesian network classifiers comes short of the thriving development in the area.

We are building a package that will include many state-of-the-art Bayesian network classifiers, covering various aspects:

- Learning
- Prediction
- And other related functionalities, such as accuracy evaluation

We plan to share this package with the community under the name *bayesClass*.
The degree of influence of each attribute on the joint is controlled by parameters $w_i$

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

$w_i$ are learned in a filter fashion
- Inversely proportional to degree of dependence of $X_i$ on other attributes in a set of decision trees
- Dependence of an attribute in a tree is estimated as

$$d_{ti} = 1/\sqrt{d_i}$$

where $d_i$ is the minimum depth at which $X_i$ is tested in tree $t$. $d_{root} = 1$

$w_i =$ dependence averaged across bagged trees

$$w_i = \overline{d}_{ti}$$
Adjusted Probability Naive Bayes Classifier (Webb et al., 1998)

- **Linear adjustments** are applied to Naive Bayes’ probability estimates

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

- The adjustments are found using a hill-climbing search for the maximization of **resubstitution accuracy**

- **Overfitting** managed by only making steps that improve accuracy significantly

- Improvement significance evaluated with a **binomial sign test**
Selective Naive Bayes

Forward Sequential Selection (Langley et al., 1994)

- A subset of predictors used to learn a Naive Bayes

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

\[ F \subseteq \{1, 2, ..., n\} \]

- A forward search through the feature space, evaluating subsets via a wrapper objective function

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

- All predictors \( X_i \) that pass a \( \chi^2 \) independence test based on mutual information with the class, \( I(X_i, C) \), are used to learn a Naive Bayes
Semi-naive Bayes

Forward Sequential Selection and Joining (Pazzani, 1996)

- Predictors are grouped into super-variables by forming Cartesian products between them

\[
p(c|x) \propto p(c) \prod_{j=1}^{K} p(x_{S_j}|c)
\]

\[\bigcup_{j=1}^{K} S_j \subseteq \{1, 2, \ldots, n\}, S_j \cap S_l = \emptyset, j \neq l\]

- A hill-climbing search across the space of structures is guided in a wrapper way

- A candidate predictor is included in the model either as:
  - Conditionally independent of other predictors
  - Part of a Cartesian product with a predictor already in the model

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

- Steps in structure search are evaluated using the \(\chi^2\) independence test between the class and each candidate (super-)variable
Tree Augmented Naive Bayes

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

- A tree structure encodes dependencies among predictors
  \[ p(c|x) \propto p(c)p(x_r|c) \prod_{i=1, i \neq r}^n p(x_i|x_{j(i)}, c) \]
  \{X_{j(i)}\} = Pa(X_i) \setminus C, \ i \neq r, \ r = \text{tree root}

- Edges are selected so that the sum of conditional mutual informations is maximized

Selective TAN (Blanco et al., 2005)

- Filter feature subset selection

- Only edges between variables whose conditional mutual information \( I(X_i; X_j|C) \) passes a \( \chi^2 \) test of independence are considered

- A forest instead of a tree might be formed in the predictor subgraph
A Novel Classifier

Semi-naive Selective Tree Augmented Naive Bayes

- Two steps
  - A Semi-naive Bayes is learned using the wrapper approach
  - Selective TAN applied to resulting (super-)predictors

\[
p(c|x) \propto p(c) \prod_{l=1}^{K} p(x_{S_l}|x_{j(l)}, c)
\]

\[
\{X_{j(l)}\} = \begin{cases} 
  Pa(X_{S_l}) \setminus C, & l \notin R \\
  \emptyset, & \text{otherwise}
\end{cases}
\]

\[
\bigcup_{j=1}^{K} S_j \subseteq \{1, 2, \ldots, n\}, \ S_j \cap S_l = \emptyset, \ j \neq l
\]

- Combination of wrapper and filter approaches
- Might be interesting to use wrapper only approaches
Incomplete Data

Computation of sufficient statistics from incomplete data

- We ignore the cases containing incomplete data for the corresponding variables

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X4</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>?</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X4</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>?</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Not a new practice for Naive Bayes but not generally adopted in implementations of augmented Naive Bayes models (e.g., Weka, bnlearn, etc.)

- On the basis of these statistics we approximate mutual information, maximum likelihood parameters, etc.
Cross-validation for Bayesian Network Classifiers

**Speed-up**

- For each fold, model is not re-learned from entire training set
  - Instead, full data set cross-classification tables are updated by subtracting the counts corresponding to the test set and CPT’s are recomputed
- More performance gain with higher k (e.g, with leave-one-out cv)
- **Maintains structure** across folds so not an unbiased estimator of a structure learning algorithm’s performance
- Useful for comparing candidate structures in a wrapper setting
## Comparison To Other Packages

### Learning Algorithms

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>WNB</th>
<th>TAN</th>
<th>SNB</th>
<th>SN-NB</th>
<th>SN-TAN</th>
<th>LBR</th>
<th>NBT</th>
<th>GBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>bnlearn</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>e1071</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RWeka</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>bayesClass</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **NB**: Naive Bayes
- **WNB**: Weighted NB
- **TAN**: Tree Augmented NB
- **SNB**: Selective NB
- **SN-NB**: Semi-naive NB
- **SN-TAN**: Semi-naive TAN
- **LBR**: Lazy Bayesian Rules
- **NBT**: Naive Bayes Tree
- **GBN**: General Bayesian Network
## Summary

### Pending Tasks and Future Work

- Wrap-up our code as an R package
  - Standardize function names
  - Document functions according to R standards
- Integrate with **gBase** for network visualization

### Contributions to the R community

- When completed, our package will contribute the following
  - 8 **learning algorithms** that, to the best of our knowledge, are not currently available to the R community
  - Bayesian classifiers that can learn from **incomplete data**
  - **Prediction** implementation that is faster than that of **gRain** when dealing with complete data
  - Relatively fast implementation of **cross-validation** for BN classifiers
Outline

1 Introduction
   - Bayesian Network Classifiers

2 Bayesian Network Classifiers in R
   - The R Environment for Statistical Computing
   - Discrete Bayesian Network Classifiers

3 bayesClass R Package
   - Overview
   - Learning Algorithms
   - Cross-validation for Bayesian Network Classifiers
   - Summary and Pending Tasks

4 Application to Neuroscience
   - Nomenclature of GABAergic Interneurons
   - A Practical Approach
   - Further Insights into Classification of Interneurons
   - Conclusions and Future Work
Nomenclature of GABAergic interneurons

GABAergic interneurons

- GABAergic interneurons are aspiny short-axon neurons from the cortex
- Different **classes** have been identified by scientists...
- ...but no clear **rules** for distinguishing between classes exist...
- ...making it **hard to share knowledge**

Two examples of GABAergic cortical interneurons
A Practical Approach

Insights into nomenclature of interneurons (DeFelipe et al., 2013)

- A practical (rather than systematic) classification based on five anatomical features was proposed in a recent study.
- A community-based strategy was adopted for validating the proposal.
  - Experts from labs across the world classified neurons according to the features.
  - Agreement between experts and usefulness of classes was analysed using statistical and machine learning techniques.
A Practical Approach

Anatomical Features

- Four **axonal features** were considered
  - **F1** Laminar projection
  - **F2** Columnar projection
  - **F3** Location w.r.t. location of dendritic arbor
  - **F4** Vertical orientation
Fifth feature: neuronal class

Choice limited to nine classes
Data Collection

- A Web-based platform was built for experts to classify neurons according to the features.
A Practical Approach

Analyses

- **Inter-expert agreement analysis**
  - Bayesian networks, agreement indices, clustering
- **Predicting features given morphological data**
  - 10 supervised classifiers from Weka were used
  - Problem: instead of a single label there are 42 votes
  - Solution: most voted class was used as label

Results

- Two neuronal classes seem **clearly defined**, others more or less **vague**
- Low expert consensus on F5, **high** on F1-F4
- **Machine learning** methods equally or more accurate than humans when discriminating neurons
  - High accuracy on F1-F3
  - Low accuracy on F4 and F5
Further insights into classification of interneurons

Less naive label approximation

- Most voted feature can have a low number of votes, yielding unreliable labels
- We used vote thresholds
  - Only neurons whose most voted feature has $\geq$ threshold votes are considered
  - Trade-off: more reliable labels $\Rightarrow$ less data...
  - ...therefore, we learned across different thresholds

New classifiers and learning tasks

- 11 Bayesian network classifiers (implemented in bayesClass) were used
- Joint prediction of $(F1, F2, F3, F4)$ (multi-dimensional classification)
Further insights into classification of interneurons

Data

- Up to 240 variables for learning an anatomical feature
- Different degrees of difficulty across learning tasks
  - F5, F4, (F1,F2,F3,F4) Class imbalance, classes with few samples
  - F1,F2, and F3 Balanced, binary class
- Continuous values were discretized with the equal-frequency method

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Variables</th>
<th>Classes</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>57</td>
<td>2</td>
<td>220</td>
</tr>
<tr>
<td>F2</td>
<td>57</td>
<td>2</td>
<td>221</td>
</tr>
<tr>
<td>F3</td>
<td>100</td>
<td>2</td>
<td>223</td>
</tr>
<tr>
<td>F4</td>
<td>100</td>
<td>3</td>
<td>97</td>
</tr>
<tr>
<td>F5</td>
<td>240</td>
<td>7</td>
<td>97</td>
</tr>
<tr>
<td>(F1,F2,F3,F4)</td>
<td>100</td>
<td>12</td>
<td>193</td>
</tr>
</tbody>
</table>

Table: Databases at threshold = 50%
Further insights into classification of interneurons

Predictive Accuracy

- Accuracy estimated with 5-fold CV (affects small data sets)

- 77% accuracy with 60% votes threshold (≈ 25% of data)

- Accuracy for F1-F3 > 90% at 90% threshold
Further insights into classification of interneurons

Classifiers Comparison: Accuracy on F5

- **TAN**, **AWNB**, and **Naive Bayes** generally have best accuracies.
- **Filter FSS** and **Filter FSSJ** have worst and identical results.
- **Novel classifier**’s results identical to **FSSJ**.
Conclusions

- With reliable labels, features can be predicted with high accuracy.
- For all features, predictive accuracy grows with label reliability.
  - Confusing for experts = confusing for machines? Some features might not be clear enough in all cases.
- F4 might be more useful than previously thought, as it is learned with higher precision.
- Some neuronal classes might be more useful than previously though as they can be predicted well when labels are reliable.

Future Work

- Hierarchical classification of F5.
- Used predicted F1-F4 to predict F5.
Discrete Bayesian network classifiers in R and an application to Neuroscience

Bojan Mihaljevic

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

February 7\textsuperscript{th}, 2013