MAC6916 PROBABILISTIC GRAPHICAL MODELS LECTURE 10: BAYESIAN NETWORK CLASSIFIERS

DENIS D. MAUÁ

1. Introduction

A *classifier* is a function $f(a_1, \ldots, a_n) \mapsto y$ mapping an object represented by its attribute values $a_1, \ldots, a_n$ into a class or category c. An example of a classification application is the detection of spam messages, which can be posed as a classification of messages (represented as word counts) into $c \in \{0, 1\}$ (not spam/spam). A *probabilistic classifier* classifies an object based on a probabilistic model $p(C|a_1, \ldots, a_n)$. For instance, a classification rule might be to select the maximum a posteriori class:

classifier

probabilistic classifier

$$f(a_1, \ldots, a_n) = \arg\max_c \mathbb{P}(C = c|A_1 = a_1, \ldots, A_n = a_n)\,.$$

In this lecture we examine classes of Bayesian networks used to build probabilistic classifiers. These are called *Bayesian network classifiers* (BNCs). BNCs have several advantages over other approaches, including the ability to represent a model learned from data in a intuitive and concise form, a sound and clear semantics for assessing the uncertainty over its predictions, an easy and principled way to perform cost-sensitive classification (i.e., when different misclassifications are assessed differently), a unified treatment of feature selection and model selection, the ability to accommodate missing data and structured variables of various sorts (continuous, ordered, categorical), the ability to perform both supervised, semi-supervised and unsupervised learning in a single framework, and the abundance of different techniques for learning, evaluating and querying models.

Bayesian network classifiers

We start the discussion with the simplest Bayesian network classifier, called *Naive Bayes Classifier* (NBC), which assumes that all attributes are conditionally independent given the class. We then move forward to the *Tree-Augmented Naive Bayes classifier* (TAN), which relaxes the independence assumption and assume the class-conditional distribution $p(A_1, \ldots, A_n|C = c)$ is modeled by a tree-shaped Bayesian network. We discuss the issue of model uncertainty, and examine an ensemble model obtained by a collection of simple TANs that do not require learning.

In standard classification, the class variable Y takes on a finite number of integer values, e.g. $\{1, \ldots, K\}$. In many applications, the class variable is structured, and is better presented by a multidimensional vector $C_1, \ldots, C_m$. This is the case, for instance, of multilabel classification, where an object can be assigned many labels and each $C_i \in \{0, 1\}$ indicates a label (e.g., tagging images with occurring objects), and of aspect-based sentiment analysis, where an object is classified according to posive/negative opinions on several aspects (e.g., classifier whether a restaurant user review is positive towards the service, food, ambiance, etc). A simple approach is to

Date: May 3, 2016.

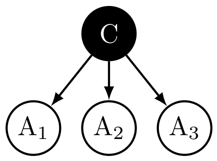

FIGURE 1. Naive Bayes Classifier for a 3-attribute problem.

decompose this problem into multiple unidimensional classification problems, where each learns a classifier for a class dimension Y_i. This however ignores correlation among the dimensions and can hurt performance (e.g., overall opinion about a restaurant is strongly dependent on the opinion about the food quality). A better approach is to build a single Bayesian network modeling the class correlations. We discuss the Average Multidimensional One-Dependence classifier, an extension of the Naive Bayes classifier for structured classification.

For simplicity, we limit the discussion to classifications problems with categorical attributes, although some of the classifiers discussed can also cope with continuous variables (e.g., by representing them as soft evidence). Numerical attributes can be turned into categorical attributes by several discretization techniques (e.g., [5, 1, 6]).

2. Naive Bayes Classifier

Let $A = (A_1, \ldots, A_n)$ be a list of (categorical) attribute variables (used to represent objects), and let C be a class variable taking on finitely many values. We assume we have available a complete data set of observations $(a^1, c^1, \ldots, a^N, c^N)$ (although our methods could handle incomplete data as well), from which a classifier is to be built. The data are assumed to be generated independently and identically distributed (i.i.d.) from a Bayesian network $p(A_1, \ldots, A_n, C)$. Note that by the chain rule, such a distribution can always be written as

$$p(A_1, \ldots, A_n, C) = p(A_1, \ldots, A_n | C) p(C) \,.$$

The Naive Bayes Classifier (NBC) is based around the assumption that the *attribute variables are conditionally independent given a class value* (see Figure 1):

$$p(A_1, \ldots, A_n | C) = \prod_{i=1}^{n} p(A_i | C) \,.$$

Even though this assumption may seem unrealistic, it is often a reasonable assumption for high dimensional problems ($n/N \sim 1$), when data is insufficient to learn more complex models. Also, when the focus is on *classification accuracy* (i.e., the relative frequency of predictions correct), NBC can be fairly accurate despite being a poor probability estimator since it has low variance (over data sets) [3, 4]. classification accuracy

The parameters of the network are learned using either MLE or Bayesian estimates. NBC is typically used to classify an object $a_1, \ldots, a_n$ by selecting the class value c which maximizes the class posterior probability $\mathbb{P}(C = c | a_1, \ldots, a_n)$. By taking the logarithm, and dispensing with terms that are constant w.r.t. the choice of class label, one can see that the maximum a posteriori class label is (when MLE parameters are used):

$$f_{\text{NBC}}(a_1, \dots, a_n) = \arg\max_c (1-n)\ln N[C=c] + \sum_{i=1}^{n} \ln N[A_i = a_i, C = c] \,.$$

Hence, classification with NBC takes linear time in the number of attributes, and is extremely efficient.

On some occasions, one wishes to compute the posterior class-conditional probability (for instance, to issue a cost-sensitive classification). This, of course, can be achieved by normalizing $p(a,c)$ over C, so that $p(c|a) = p(a,c)/p(a)$. The issue with this approach is that each $p(a|c)$ is usually the product of very small probabilities, which leads to numerical errors. The solution is to apply the so-called *log-sum-exp trick*, and to obtain $\ln p(a)$ by

log-sum-exp trick

$$\ln p(a) = \ln\left(\sum_c e^{\ln p(c,a)}\right) = \ln\left(e^u \sum_c e^{[\ln p(a,c)]-u}\right) = u + \ln \sum_c e^{\ln[p(a,c)]-u} \,,$$

where $u = \max_c \ln p(a,c)$.

3. Tree-Augmented Naive Bayes Classifier

The NBC is represented by Bayesian network with edges pointing from the class variable C to every attribute variable A_i. A natural extension to this classifier is to consider a Bayesian network which, in addition to the class-to-attribute arcs, also admits arcs between attributes, thus relaxing the conditional independence assumption. We call any such family of classifiers an *augmented naive Bayes classifier* [7]. An example is depicted in Figure 2. In order to limit the computational difficulty of both learning and inference, the augmented structure is often restricted to be a chordal graph. One particularly efficient and simple form of chordal graphs are naive Bayes structures augmented with a tree over the attributes. This is the sort of classifier we discuss next.

augmented naive Bayes classifier

Recall that any tree-shaped Bayesian network with skeleton $G = (\mathcal{V}, \mathcal{E})$ satisfies

$$p(\mathcal{V}) = \prod_{X \in \mathcal{V}} p(X) \prod_{X-Y \in \mathcal{E}} \frac{p(X,Y)}{p(X)p(Y)} \,.$$

The MLE log-likelihood is [8]

$$LL^{\text{MLE}}(G) = \max_\theta LL(\theta, G) =$$

$$\underbrace{\sum_{X \in \mathcal{V}} \sum_{x \sim X} N[X=x] \ln \frac{N[X=x]}{N}}_{\text{constant}} + \sum_{X-Y \in \mathcal{E}} \underbrace{\sum_{x,y \sim X,Y} N[X=x,Y=y] \ln \frac{N[X=x,Y=y]}{N[X=x]N[Y=y]}}_{=-\omega(X,Y)} \,.$$

The first term is constant w.r.t. to the skeleton G, while the second is a sum of edge weights $\omega(X,Y) \geq 0$. Thus, the problem of selecting the tree-shaped Bayesian network that maximizes the log-likelihood is equivalent to the problem of selecting the spanning tree of the complete graph with node set $\mathcal{V}$ that minimizes

$$\omega(G) = \sum_{X-Y} \omega(X,Y) \,.$$

This problem, called *minimum-cost spanning tree* can be solved in time $O(n^2 \log n)$ by e.g. Kruskal's minimum spanning tree algorithm [9]. Note that since the weights

minimum-cost spanning tree

are non-negative, for each forest (non-connected graph) there is always a tree which is at least as good (so that we loose nothing by considering only trees).

The *tree-augmented Naive Bayes classifier* (TAN) relaxes the independence assumption of NBC, and assumes that the parents of each attribute variable are the class variable and at most one parent attribute variable, that is, that $p(X_1, \ldots, X_n | Y)$ is a tree-shaped Bayesian network (in principle, the subgraph can be disconnected, but the learned networks will always be connected for the reasons discussed). The TAN classifier is learned by selecting the tree-structure and the parameters that maximize the log-likelihood:

$$(G^{\text{TAN}}, \theta^{\text{TAN}}) = \arg \max_{G:\text{pa}(A_i)=\{A_j, C\}} LL^{\text{MLE}}(G) \, .$$

This is achieved by computing the minimum-cost spanning tree G^* for the weighted complete graph over $A_1, \ldots, A_n$ with weights

$$\omega(A_i, A_j) = -\sum_{c, a_i, a_j} N[a_i, a_j, c] \ln \frac{N[a_i, a_j, c]}{N[a_i, c] N[a_j, c]} \, ,$$

and then learning the parameters with that fixed structure. We can use any estimators for learning the parameters such as Bayesian estimates, MAP estimates and MLE. It is also possible to define weights based on Bayesian estimates; however, to retain the equivalence with a minimum spanning tree problem, we need to select the hyperparameters so as to preserve *likelihood equivalence*: two Markov equivalent structures should have the same posterior distribution. This is achieved by setting

likelihood equivalence

$$\alpha_{A_i | a_j, c}(x) = \alpha_{x_i x_j | c} = \frac{\alpha_c}{r_{X_i} \cdot r_{X_j}} \, ,$$

class-conditional equivalent sample size

where α_c is a parameter known as the *class-conditional equivalent sample size* and $r_X = |\Omega_X|$. Typical values for α are values in $[1, 5]$. Note that the time complexity of TAN is most often dominated by the construction of the complete weighted graph, which takes $O(n^2 N)$ time.

Classification with TAN is performed by maximizing the class posterior probability, which is equivalent to maximizing

$$f_{\text{TAN}}(a_1, \ldots, a_n) = \ln \frac{N[C = c]}{N} + \sum_{i=1}^{n} \ln \frac{N[A_i = a_i, A_{\sigma(i)} = a_{\sigma(i)}, C = c]}{N[A_{\sigma(i)} = a_{\sigma(i)}, C = c]} \, ,$$

where $\sigma(i)$ is the index of the parent attribute of A_i.

TAN makes less unrealistic assumptions about the data, and often outperforms NBC, especially, when $n/N \ll 1$ [7].

4. Average One-Dependence Classifier

The TAN classifier selects a single Bayesian network among several possible candidate structures. Often, there are many structures with very similar scores (log-likelihood), which makes the selection highly sensitive to small perturbations of the data. This increases the variance of the classifier and reduces its accuracy. A more principled approach is to retain an ensemble of high scoring structures, and to perform classification by aggregation. While this approach is quite general and applies to any method which performs model selection, it takes a very interesting form in the case of TANs.

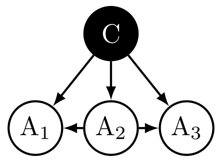

FIGURE 2. One-Dependence Classifier with 3 attributes and superparent A$_2$.

A One-Dependence Bayesian network Classifier (ODC) is a tree-augmented Naive Bayes classifier where the class conditional distribution $p(A_1, \dots, A_n|C)$ follows a Naive Bayes structure (see Figure 2 for an example) [10, 11]. The attribute A_i with a single parent Y is called the *superparent* of the model. The classifier can be learned by selecting the ODC which maximizes the log-likelihood or some other convenient scoring function. Alternatively, we can dispense with the model selection and consider all possible ODC classifiers. A classification is made by averaging over all classifiers. The result is called *Average One-Dependence Classifier* (AODC). The joint distribution of AODC is given by

superparent

Average One-Dependence Classifier

$$p(A_1, \dots, A_n, C) = \frac{1}{n} \sum_{i=1}^{n} p(A_i, C) \prod_{j \neq i} p(A_j|A_i, C) .$$

A classification is usually performed considering only the ODCs whose superparent has is reliably estimated:

$$f_{\text{AOD}}(a_1, \dots, a_n) = \arg\max_y \frac{1}{|S|} \sum_{i \in S} N[a_i, c] \prod_{j \neq i} \frac{N[a_j, a_i, c]}{N[a_i, c]} ,$$

where S is the set of indices such that the tally $N[A_i = a_i, C = c]$ exceeds a certain threshold (say, 30).

5. Multidimensional Bayesian Network Classifiers

In many tasks, the class variable is multidimensional: $C = (C_1, \dots, C_m)$. When the number of dimensions is small, we can treat the multidimensional classes as unidimensional classes and use any standard (Bayesian network) classifier. This approach is however unfit for high dimensional classes due to computational and statistical inefficiency. A simple alternative is to treat the classification of each dimension as a separate classification problem, that is, to build a classifier for the domain $(A_1, \dots, A_n, C_j)$ for each $j = 1, \dots, m$. While this approach often performs reasonably well, an explicit modeling of class dimension correlations can often improve accuracy (and particularly so if the accuracy measure takes class correlations into account) [14, 13, 15].

Recall that the conditional joint distribution factorizes as $p(C_1, \dots, C_m|A) = \prod_{j=1}^{m} p(C_j|C_1, \dots, C_{j-1}, A)$. A more sophisticated approach is to consider a sequence of increasingly more complex unidimensional classifiers that incorporate the previous class dimensions as attributes. That is, we obtain a sequence of unidimensional probabilistic classifiers $p(C_j|C_1, \dots, C_{j-1}, A)$, and we classify an instance $a_1, \dots, a_n$ by feeding the classification of classifier C_{j-1} as an attribute to classifier

C_j:

$$f^j_{\text{PCC}}(a_1, \ldots, a_n, c_1, \ldots, c_{j-1}) = \arg\max_{c_j} \mathbb{P}(C_j = c_j | a_1, \ldots, a_n, c_1, \ldots, c_{j-1}) \,.$$

probabilistic chain classifiers

This approach, termed *probabilistic chain classifiers* [14], often improves on ensembles of uncorrelated class variable classifiers and with low computational and statistical overhead. Yet, this approach can misrepresent class correlations (especially, if strong assumptions over the attributes are made in each classifier, such as a Naive Bayes).

multidimensional Bayesian network classifier

A more powerful approach is to consider a *multidimensional Bayesian network classifier* (MDBNC) which specifies a Bayesian network over all the variables such that there is no arc from an attribute into a class variable (resembling an augmented naive Bayes classifier) [12, 13]. The arcs in a MDBNC are categorized into

- class-to-class arcs, which connect two arcs and represent direct dependencies between class dimensions;
- class-to-feature arcs, which connect a class dimension to a feature, and perform feature selection;
- feature-to-feature arcs, which connect attributes, and represent attribute correlation.

class subgraph

bridge subgraph

feature subgraph

The MDBNC are classified in terms of the types of arcs they contain. The *class subgraph* is the subgraph of a MDBNC containing only class variables and class-to-class arcs. The *bridge subgraph* contains all variables and only class-to-feature arcs. Finally, the *feature subgraph* contains only attributes and feature-to-feature arcs. The simplest family of MDBNC is defined by empty class and feature subgraphs. This types of bipartite graphs are the natural extensions of naive Bayes classifiers to multidimensional classes, as they assume unconditional independence among classes and conditional dependence among attributes. This *multidimensional naive Bayes classifier* is still strictly more expressive than an ensemble of unidimensional NBCs, are the classes in the multidimensional NBC are d-connected by means of common (attribute variables) children. We can obtain more expressive families of MDBNC by allowing increasingly more complex subgraphs. For instance, a *multidimensional tree-augmented naive Bayes classifier* is obtained by allowing tree-shaped class subgraph and feature subgraph.

multidimensional naive Bayes classifier

multidimensional tree-augmented naive Bayes classifier

6. Average Multidimensional One-Dependence Classifiers

While more complex structures in MDBNCS increase the representational power, it often worsens its practical performance, as the number of parameters grows to large and become unreliably assessed. There is also the variance introduced by selecting one particular structure withing the family of models considered.

A possible workaround to both problems is given by a multidimensional version of the Average One-Dependence Classifier, which consists of an ensemble of MDBCS where the class subgraph is a naive Bayes (each with a different superparent class variable) and the feature subgraph is empty [15]. Each feature subgraph is learned maximizing a common scoring function for Bayesian networks such as the BIC or the BDeu scores. This leads to models which share the feature subgraph, thus making both learning and inference more tractable. In fact, it can be shown that learning is performed in linear time if a bound on the number of parents of each attribute variable is imposed. The classification is carried out by selecting the

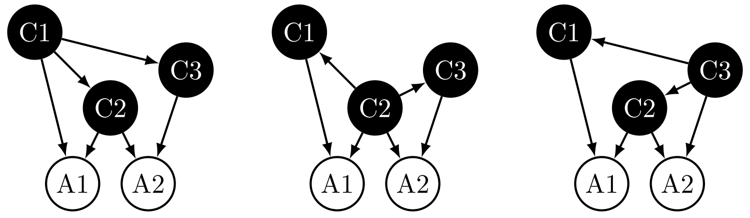

FIGURE 3. Multidimensional Average One-Dependence Classifier.

marginal maximum a posteriori class for each dimension:

$$\arg\max_{c_j} \mathbb{P}(C_j = c_j, a_1, \dots, a_n) = \prod \mathbb{P}(a_i|\text{pa}(C_j)) \sum_{\{C_k \neq C_j\}} \sum_{k=1}^{m} \frac{1}{m} \mathbb{P}(C_k) \prod_{\ell \neq k} p(C_\ell|C_k) .$$

Since the overall graph usually has large treewidth, the above inference is performed by approximate algorithms such as sum-product.

Dataset	n	m	BR-NB	ECC-NB	ECC-J48	ML-NB
emotions	6	72	77.6%	78.1%	78.0%	78.0%
scene	6	294	82.6%	83.5%	88.3%	88.0%
yeast	14	103	70.3%	70.3%	77.1%	77.3%
slashdot	22	1079	95.9%	95.9%	95.9%	95.9%
genbase	27	1186	99.6%	99.6%	99.8%	99.8%
cal500	174	68	61.5%	57.0%	61.4%	85.9%

7. Evaluation

In a classification setting, we are mostly interested in the classification performance on unseen data, that is, on instances not in the the data set used to learn the classifier. This however can only be estimated from the available data. Using the same data set for learning and evaluating leads to an optimistic performance measure. A better approach is to split the data set into a *training set* and a *test set*; the classifier is then learned from the training set and evaluated using the test set. The split introduces a trade-off on learning versus evaluation accuracy. A big training set allows better classifiers to be learned but produces poor performance estimates; conversely, a small training set induces poor classifiers and more reliable estimates. A common rule-of-thumb is to use a 70%/30% training/test split.

training set
test set

8. Cross-Validation

A more refined approach (especially when the data set is small) is cross-validation. It consists in partitioning the data set into k folds of approximately equal size. To make the estimates more reliable, this partitioning is usually *stratified*, meaning that the class label proportion in the whole data set is approximated in each part (fold). This can be achieved e.g. by sampling without replacement with probability proportional to the marginal class distribution.

One the k-partitioning has been made, we evaluate a classifier by repeating the following procedure for each fold j:

(1) train the classifier from the data in folds $1, \dots, j-1, j+1, \dots, k$;

(2) compute the average classification accuracy of the classifier in fold j;

The overall estimate is the average of classification accuracy over all folds. It is also interesting to compute the standard deviation and statistical tests of equal mean (e.g., paired Student's t-test). When comparing multiple classifiers, we need to ensure that each classifier is evaluated in the same partitioning of the data set. The number of folds (parts) is typically chosen to be $k = 5$ or $k = 10$.

9. Reading

- Friedman et al., Bayesian Network Classifiers, 1997 [7]
- Antonucci et al., An Ensemble of Bayesian Networks for Multilabel Classification, 2013 [15]

10. Exercises

No exercises this week.

11. Assignment

- Implement the Naive Bayes Classifier, TAN, AODC, and the multidimensional AODC.
- Evaluate their performance on the provided data sets (available at PACA), using different configurations of parameters (equivalent sample size, minimum sample size for selecting a model in AODC, etc). Use the training set to select parameter configurations (apply cross-validation to compare settings), and evaluate the best configuration of each classifier on the test set. Build unidimensional classifiers for each dimension, and a multidimensional classifier. For data sets with few dimensions, build also unidimensional classifiers with combined class labels (that encodes all configurations of labels). Optionally: build probabilistic chain classifiers using NBC, TAN and AODC.
- Write report discussing their relative performance (accuracy, time complexity, difficult to implement). Try to justify the superiority of a classifier in the given data set. It might be interesting to evaluate the hold-out data log-likelihood of each classifier, that is, the log-likelihood of each classifier on the test set (is there any correlation between likelihood and accuracy).

Due date: 6/6

References

[1] J. Dougherty, R. Kohavi and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 194–202, 1995.

[2] P. Krauss Representation of conditional probability measures on Boolean algebras. Acta Mathematica Academiae Scientiarum Hungarica, volume 19, issues 3–4, pp. 229–241, 1968.

[3] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, volume 29, pp. 103–130, 1997.

[4] J. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery, volume 1, pp. 55–77, 1997.

[5] U.M. Fayyad and K.B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1027, 1993.

[6] N. Friedman and M. Goldszmidt. Discretization of continuous attributes while learning Bayesian networks. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 157–165, 1997.

[7] N. Friedman, D. Geiger and M. Goldszmidt. Bayesian network classifiers. Machine Learning volume 29, pp. 131–163.

[8] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, volume 14, issue 3, pp. 462–467, 1968.

[9] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society 7*, pp. 48–50, 1956.

[10] G.I. Webb, J. Boughton and Z. Wang. Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning, volume 58, pp. 5–24, 2005.

[11] Y. Yang, K.B. Korb, K.M. Ting and G.I. Webb. Ensemble selection for superparent-one-dependence estimators. In *Proceedings of the 18th Australian Conference on Artificial Intelligence*, pp. 102–112, 2005.

[12] L.C. van der Gaag and P.R. de Waal. Multi-dimensional Bayesian network classifiers. In *Proceedings of the Third European Workshop in Probabilistic Graphical Models*, pp. 107–114, 2006.

[13] C. Bielza, G. Li, and P. Larranaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, volume 52, issue 6, pp. 705–727, 2011.

[14] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 279–286, 2010.

[15] A. Antonucci, G. Corani, D.D. Mauá and S. Gabaglio. An Ensemble of Bayesian Networks for Multilabel Classification. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 1220–1225, 2013.