



Bayesian networks for classification

Priyantha Wijayatunga

*Department of Mathematical and Computing Sciences, Tokyo Institute of Technology,
Oo-okayama, 2-12-1, W-28, Meguro-ku, Tokyo 152-8552, Japan.*
✉ spwijay@is.titech.ac.jp or wijayatunga@gmail.com

Abstract

A Bayesian network models probabilistic relationships among a set of random events through a directed acyclic graph. Due to availability of efficient inference algorithms, they are used as probabilistic expert systems. Classification is fundamental task in data analysis and the application of Bayesian networks for this task has become popular. Though there are many learning methods of Bayesian networks from data and/or expert knowledge they are not optimal for learning Bayesian networks to be used as classifiers. Here we present and discuss a simple method to learn parameters of Bayesian networks so that they have better classification accuracy. We also discuss the background of the topic somewhat in detail but with less mathematical descriptions.

Key words: *Bayesian networks, classification accuracy, parameter estimation.*

Introduction

Often we find situations or phenomena that have many random events associated with it. When we model them, probabilistic models can be very useful as they model probabilistic relationships among the set of random events (variables) concerned. Graphical models are such models which have become very popular among statistical and artificial intelligence communities due to their facility of easing the communication between domain expert and the modeler. Bayesian networks are a class of graphical models that models probabilistic relationships through a directed acyclic graph of which each node corresponds to a random variable and each arrow corresponds to a probabilistic dependence relation between two variables concerned. Absence of an arrow between two variables may indicate that their dependency is mediated through some of the other variables. Reading independence relations can be somewhat detailed. For comprehensive discussion on graphical models and Bayesian networks the reader is referred to Pearl (1988), Lauritzen (1996), and Cowell, et al. (1999).

First we present a simple description on Bayesian networks and their use as classifiers. Assume that we are interested in modeling five random discrete variable vector, say,

$\underline{X} = (X_1, \dots, X_5)$. By the chain rule of probability, the joint density of \underline{X} can be written as

$$p(\underline{x}) = p(x_1) \prod_{i=2}^5 p(x_i | x_1, \dots, x_{i-1}) \quad (1)$$

for $p(\underline{x}) > 0$, where $p(x_i | x_1, \dots, x_{i-1})$ is the conditional probability density of X_i given $X_1 = x_1, \dots, X_{i-1} = x_{i-1}$. There can be some independence and conditional independent relations among variables. For example, if we assume that X_3 is conditional independent of X_2 given X_1 written as $X_3 \perp X_2 | X_1$ and furthermore $X_4 \perp \{X_2, X_3\} | X_1$ and $X_5 \perp \{X_1, X_2\} | \{X_3, X_4\}$ then the above model (1) will be simplified to

$$p(\underline{x}) = p(x_1)p(x_2 | x_1)p(x_3 | x_1)p(x_4 | x_1)p(x_5 | x_3, x_4) \quad (2)$$

Thus, we have arrived at an efficient factorization of the joint density of \underline{X} which can be represented by a directed acyclic graph (DAG) as follows (see Fig. 1).

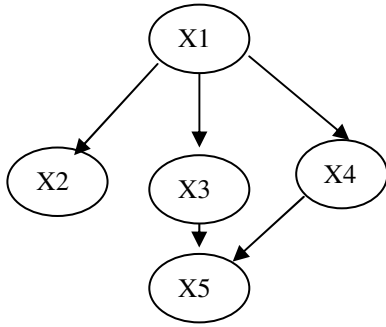


Figure 1. A simple Bayesian network

Note how the arrows are drawn, for example, to represent the factor $p(x_5 | x_3, x_4)$ in (2) two arrows are drawn each from X_3 and X_4 to X_5 and so on. Therefore the set $\{X_3, X_4\}$ is called the parents of the node X_5 and it is denoted by $pa(X_5)$. The opposite relation is the children, for example, children of X_1 are the set $\{X_2, X_3, X_4\}$ which is denoted by $ch(X_1)$. Similarly parents and children sets are defined for all the variables in the network. Note that $pa(X_1) = \Phi$. So we can write

$$p(\underline{x}) = \prod_{i=1}^5 p(x_i | pa(x_i)) \quad (3)$$

This is our Bayesian network model but still it is incomplete. Note that we derived it assuming some (conditional) independence relations. In fact, these independence relations define the structure of the network. Note that a conditional independence relation expresses an “information irrelevance”, for example, the relation $X_4 \perp \{X_2, X_3\} | X_1$ expresses that once we know the value of X_1 the knowledge about X_2 and X_3 is irrelevant to determine the density of X_4 therefore the factor $p(x_4 | x_1, x_2, x_3)$ in (1) reduces to $p(x_4 | x_1)$ in (2).

In order to define (3) fully, we need to

define each factor in it numerically. In the discrete case conditional density $p(x_i | pa(x_i))$ is modeled by multinomial distribution with parameter vector, say, $\theta_{i|pa(x_i)} = (\theta_{x_i|pa(x_i)} : \forall x_i \in sp(X_i))$ where $sp(X_i)$ denotes the state space of X_i . It is schematically written as $X_i | pa(x_i) \sim Mn(\theta_{i|pa(x_i)})$. In the non-Bayesian setting $\theta_{x_i|pa(x_i)} = P(X_i = x_i | pa(x_i))$ and in the Bayesian setting $E\{\theta_{x_i|pa(x_i)}\} = P(X_i = x_i | pa(x_i))$ where E denotes the expectation with respect to density of $\theta_{i|pa(x_i)}$.

Therefore, for node X_i we have the total parameter vector $\theta_i = (\theta_{i|pa(x_i)} : \forall pa(x_i) \in sp(pa(X_i)))$.

And for the total network the parameter vector is $\theta = (\theta_1, \dots, \theta_5)$ which is called the parameter of the Bayesian network and in order to define it fully, we need to define each component of it numerically.

Therefore to build a Bayesian network, we need to define the (conditional) independence relations among the set of variables and corresponding multinomial parameters numerically. Researchers build Bayesian network models using data on the set of variables concerned and perhaps along with expert knowledge. Here the expert knowledge refers to the knowledge of experts in the field from which data are taken. They use such methods as statistical (conditional) independence tests to identify the structure of the network and parameter estimation techniques such as method of maximum likelihood estimation. Further there are many sophisticated methods for this purpose but we avoid describing them here.

Once we have built a Bayesian network it can be used to calculate efficiently the probabilities such as $P(X_3 = a | X_2 = b, X_5 = c)$, etc. where a, b and c are possible realizations of the respective variables. This efficiency is rooted

in the efficient factorization of the joint density and obtained through inference algorithms such as “junction tree algorithm” developed in the literature. Reader is referred elsewhere for details on inference algorithms.

In a classification task, we are interested in estimation of the probability density of the “class” variable given the values on some or all of the “attribute” variables that are probabilistically related to the class. For example, a doctor may be interested in estimating probability of a patient having a certain severe disease by observing some symptoms before he performs an expensive medical test. Here the class is “disease” which has two values “yes” or “no” and symptoms such as “fever”, “appetite”, “X-ray-result” are the attributes, each has its own state space. Doctor may do the expensive test on the patient if he is convinced that probability of having disease is high.

Here we discuss a very popular Bayesian network classifier called naïve Bayes classifier for appraisal of companies –to predict whether the company is “bad”, “moderate” or “good” based on values on attributes that are financial variables.

Naïve Bayes classifier

Naïve Bayes network (NBN) has a very simple structure but its classification accuracy is surprisingly high. Suppose C is the class variable and $\underline{X} = (X_1, \dots, X_n)$ be the n -attribute vector. In the NBN we assume that $X_i \perp X_j | C$ for any $i, j \in \{1, \dots, n\}$ where $i \neq j$. Therefore we have the model

$$p(\underline{x}, c) = p(c) \prod_{i=1}^n p(x_i | c) \quad (4)$$

Therefore network structure looks like what is shown in the Fig. 2.

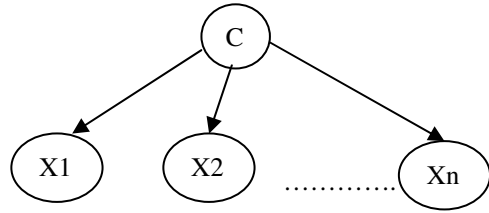


Figure 2. Naïve Bayesian network (n attributes)

The parameter vector of the network is $\theta = (\theta_c, \theta_1, \dots, \theta_n)$. Recall that the components of the vector θ are themselves vectors. In order to use NBN we need to learn its parameter θ using training data on (C, \underline{X}) . This type of training is called “supervised learning” in machine learning literature. Further there are two methods of learning: “generative learning” and “discriminative learning”.

In generative learning we try to train the parameters of the model so that we get the model for the data generating process. Therefore we maximize the likelihood of data with respect to the assumed model, thus we obtain the corresponding values for parameters (maximum likelihood estimates (MLE)). Note that here we do not take into account the fact that we are training the model for a classification task therefore, the learnt model may not suit the task.

In a classifier we are interested in the collection of conditional densities $\{p(c | \underline{x}), \forall \underline{x}\}$. Explicitly, we need to find the densities,

$$p(c | \underline{x}) = \frac{p(c) \prod_{i=1}^n p(x_i | c)}{\sum_d p(d) \prod_{i=1}^n p(x_i | d)} \quad (5)$$

for all $\underline{x} \in sp(\underline{X})$. Ideally these densities should be learnt so that they will have a better classification accuracy.

Now assume that we have training data $\underline{D} = \{(c^{(1)}, \underline{x}^{(1)}), \dots, (c^{(N)}, \underline{x}^{(N)})\}$ where the superscript in the brackets refers to the data case number (index). In the maximum

likelihood estimation, the parameters are learnt by maximizing likelihood of data, say, L .

$$L = \prod_{j=1}^N p(\underline{x}^{(j)}, c^{(j)}) = \prod_{j=1}^N p(c^{(j)}) \prod_{i=1}^n p(x_i^{(j)} | c^{(j)}) \quad (6)$$

But it is intuitive to learn parameters in such way that they maximize the collective conditional likelihood (*CCL*) –this is an instance of discriminative parameter learning. These estimates are called maximum *CCL* estimates (*MCCLE*). Unfortunately there is no closed form solution for maximization of *CCL* as in the case of maximization of L therefore numerical methods are used. Explicitly,

$$CCL = \prod_{j=1}^N \frac{p(c^{(j)}) \prod_{i=1}^n p(x_i^{(j)} | c^{(j)})}{\sum_d p(d) \prod_{i=1}^n p(x_i^{(j)} | d)} \quad (7)$$

Further, the maximum likelihood estimates are preferred to other estimates due to their asymptotic properties such as strong consistency, asymptotic normality, etc. However, it has been shown that *MCCLE* are often superior to *MLE* but it is not the case always (see Ng and Jordan (2001)). Recently, Wijayatunga, et al. (2006) have proved the asymptotic properties of *MCCLE*, which is certainly desirable. They followed the famous and elegant “Wald’s Proof” of consistency of *MLE* in theirs.

As mentioned above using *MCCLE* in the naïve Bayes network classifier is a considerable improvement. Earlier there have been many attempts to increase the classification accuracy of naïve Bayes through various parameter learning methods. On such attempt was to build a so-called “recursive Bayesian classifier” (Langley, (1993)) where the usual naïve Bayes training was followed hierarchical model building approach based on misclassifications in the training data. Another approach is by Webb and Pazzani (1998) to adjust class probabilities during the classification time using certain numerical weights and it has been shown it was successful in some cases. In order to increase the classification accuracy some researchers have tried to induce ignored dependencies in

the naïve Bayes classifier by adding further arrows in between the attributes. One instance is “tree augmented naïve Bayes” (TAN) where sub-graph induced by the attributes forms a tree or several trees by Friedman, et al. (1997).

Wijayatunga, et al. (2006) have proposed another discriminative parameter learning method. They use classification accuracy in their objective function, say, “*obj*” when learning parameters. Here we discuss their method. Define

$$obj(\theta) = \sum_{j=1}^N A^{(j)} (c_E^{(j)} - c_T^{(j)})^2 \quad (8)$$

where $A^{(j)}$ which is a weight for the error is dependent on $c_T^{(j)}$ which is the observed value of C for the j^{th} data case and $c_E^{(j)}$ is the estimated value of C for the j^{th} data case by the classifier. Ideally, these two values should be equal for every data case but this is not possible in non-deterministic contexts. Note that $c_E^{(j)}$ is estimated using the current values of θ therefore we need to find θ so that objective function attains its minimum value. Note that the smaller the objective function the better the classifier in terms of its accuracy. We can use the following algorithm for finding the better value of θ .

Algorithm:

- Initialize θ by the method of the method of maximum likelihood estimation
- Iterate an optimization procedure by changing $(\theta_1, \dots, \theta_n)$ to minimize $obj(\theta)$

Note that we propose only to adjust $(\theta_1, \dots, \theta_n)$ but not the whole parameter vector of the network, which is θ . This is certainly desirable as one may need to classify his/her test case with the true distribution of the class variables if he/she possesses no information on any of the attribute variables.

Classification of Japanese electric companies

In this Section we describe how we can apply our method in practice.

Appraisal or credit rating of companies is an important business activity for companies themselves as well as for potential investors. Here we discuss classifying Japanese electrical companies based on financial data of them (see Wijayatunga et al. (2006) for complete discussion). Explicitly, we have a training data set of 523 cases, each has data on a class variable, say, C –superiority of the company, taking values “bad”, “moderate” and “good” and 15 attribute variables (so, $n=15$ in this case) representing various financial measures of the company. For simplicity we assume numerical values $\{1,2,3\}$ for $\{bad, moderate, good\}$ for the variable C .

We use the simulated annealing optimization procedure to minimize our objective function

$$obj(\theta) = \sum_{j=1}^{523} (|c_T^{(j)} - 2| + 1)(c_E^{(j)} - c_T^{(j)})^2 \quad (9)$$

For this optimization we use “optim” function in “Statistical Language and Environment R” which is freely available over Internet (R Development Core Team (2004)). Initially maximum likelihood estimate for θ is used to find the vector $(c_E^{(1)}, \dots, c_E^{(523)})^{(0)}$. Here superscript “(0)” refers that the vector contains initial values. Then the objective function is minimized –that is, we find r such that $(c_E^{(1)}, \dots, c_E^{(523)})^{(r)}$ is close as much as possible to $(c_T^{(1)}, \dots, c_T^{(523)})$ which is the vector of true values of the class variable C for 523 number of data cases (companies).

In the following Tab. 1 to Tab. 4 the classification performance (percentage of success) of both the naïve Bayes classifier with usual parameter estimates (NB) and that with adjusted parameters by our proposed methods (NB*) is given. Here the $Error = c_E - c_T$.

Table 1. Overall classification performance (523 cases).

Error	0	1	-1	2	-2
NB	67.5%	18.0%	14.3%	0.0%	0.2%
NB*	82.0%	9.0%	9.0%	0.0%	0.0%

Table 2. Classification performance when true value of C is “bad” [$C=1$](59 cases)

Error	0	-1	-2
NB	81%	17%	2%
NB*	92%	8%	0%

Table 3. Classification performance when true value of C is “moderate” [$C=2$](342 cases)

Error	0	1	-1
NB	61%	20%	19%
NB*	77%	11%	12%

Table 4. Classification performance when true value of C is “Good” [$C=3$](122 cases)

Error	0	1	2
NB	78%	22%	0%
NB*	93%	7%	0%

One can see easily that parameter adjusted naïve Bayes classifier (NB*) has a very good classification accuracy compared with the usual naïve Bayes classifier (NB). Considerable difference happens in the cases with true value of C is either “bad” or “good”. In fact, we can alter these classification accuracy figures for NB* by altering our objective function, particularly the weights in it.

Discussion

The reader may note that our proposed method is very flexible, as one can select his/her own objective function with desired weights. Further our method is not worse than most of the other method as it adjust the parameters to increase the classification accuracy (from an initial value). That is, derived parameters will be better than (or at least similar in the sense of accuracy) the initially used parameters. One important aspect of our methods is that the applied user can use any sophisticated software for optimization to implement it. We see that this is a very big advantage. Further our method may be applied to any general Bayesian network.

Here, we have not discussed the problem of “over-fitting” when we estimate the parameters. One should take care if the learnt parameters are over fitting the training data. A future research direction is to obtain optimal parameters in terms of classification accuracy with appropriate level of regularization (not over-fitted). Simply one can use appropriate prior distributions on the parameters to regulate them but more sophisticated methods are desirable.

Acknowledgements

The author would like to thank his research supervisor, namely, Professor Shigeru Mase for his guidance and the Ministry of Education, Culture, Sports, Science and Technology of Japan for the financial assistance for his research.

References

- Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J., Probabilistic Expert Systems, Springer, NY, USA, 1999.
- Friedman, N., Geiger, D. and Goldszmidt, M., Bayesian Network Classifiers, Machine Learning, Vol. 29, 1997, pp. 131–163.
- Langley, P., Induction of Recursive Bayesian Classifiers, Proceedings of European Conference on Machine Learning, Lecture Notes in Artificial Intelligence, Vol. 667, Springer, Berlin, Germany, 1993, pp. 153–164.
- Lauritzen, S. L., Graphical Models, Oxford University Press, UK, 1996.
- Ng, A. Y. and Jordan, M. I., On Discriminative Vs Generative Classifiers: A Comparison of Logistic Regression with Naïve Bayes, Advances in Neural Information Processing Systems, 14, MIT Press, Cambridge, MA, USA, 2001, pp. 605–610.
- Pearl, J., Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference, Morgan and Kaufmann, San Francisco, CA, USA, 1988.
- R Development Core Team, R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3–900051–07–0, URL <http://www.R-project.org>, 2004
- Webb, G. I. and Pazzani, M. J., Adjusted Probability Naïve Bayes Induction, 11th Australian Joint Conference on Artificial Intelligence, Lecture Notes in Computer Science, Vol. 1502, Springer-Verlag, Heidelberg, Germany, 1998, pp. 285–295.
- Wijayatunga, P., Mase, S. and Nakamura, M., Appraisal of Companies with Bayesian Networks, International Journal of Business Intelligence and Data Mining, Vol. 1, No. 3, 2006, pp. 329–346.
- Wijayatunga, P. and Mase, S., Asymptotic Properties of Maximum Collective Conditional Likelihood Estimators for Naïve Bayes Classifiers, Submitted for publication in International Journal of Statistics and Systems, 2006.