Bayesian Network that Learns Conditional Probabilities by Neural Networks

Yoichi Motomura, Isao Hara, Hideki Asoh, and Toshihiro Matsui

Real World Computing Team, Electrotechnical Laboratory, 1-1-4 Umezono, Tskuba, Ibaraki 305, Japan
motomura@etl.go.jp, hara@etl.go.jp, asoh@etl.go.jp, and matsui@etl.go.jp

Abstract

In this paper, we discuss a Bayesian network that learns conditional probabilities by neural networks (BNNN). The real world domain is filled with many kinds of variables, which often have uncertainty factors. For the reasoning about such variables, we propose a Bayesian network using feed forward neural networks.

Advantages of this model are learning and universal representation capability for the domain mixed with continuous, discrete and multi-dimensional variables. We evaluate the potential of the BNNN on a real world application. Two experiments on learning conditional probabilities with data, which are sampled from the sensors of our autonomous mobile robot and Japanese meteorological statistical records, are introduced.

1 Introduction

The authors are trying to realize flexible intelligence in the autonomous office conversant robot, called Jijo-2[Asoh *et al.*,1996; Matsui *et al.*,1997]. This robot acts and learns in the real world, which is filled with many kinds of uncertainties. Coping with uncertainty caused by sensor noises, unpredictable motions, and dynamic changes in a real environment is an important issue in this project. Probabilistic reasoning by Bayesian networks [Pearl,1988] is one promising approach. In this model, appropriate conditional probabilities are necessary. However, providing them 'a priori' is difficult . Therefore, the robot should obtain them from statistical data sampled in the real world environment. Neural networks can be applied to this problem. Recently, the importance of learning in Bayesian networks was also discussed in *Uncertainty in AI* community. Thus, practical learning methods like neural networks for a real world domain are also significant in the community. Moreover, it is necessary to handle many kinds of variables in the real world domain. The neural network approach has advantages for the requirement, because it can provide a general framework for representing and learning non-linear conditional probabilities mixed with continuous, discrete, multi-valued, and multi-dimensional variables.

In this paper, we construct a Bayesian network that learns conditional probabilities by feed forward neural networks for a real world application. This model was implemented with the parallel processing mechanism in Java. Finally, learning conditional probabilities from real world samples was evaluated with this program.

2 Model

2.1 Feed forward neural network

We denote the input vector of the neural network as x_i, and the output vector of a neural network as $\boldsymbol{f}(\boldsymbol{x};\theta) = (f_1(\boldsymbol{x};\theta), \cdots)$. Each output is defined as,

$$f_k(\boldsymbol{x};\theta) = \sum_{i=1}^{n} \frac{v_{ki}}{1+\exp(-(\sum_{j=1}^{m} w_{ij}x_j + b_i))} + a_k,$$

where v_i is the weight of the connection from the i-th hidden neuron to the output neuron, w_{ij} is a weight of the connection from the j-th input neuron to the i-th hidden unit, a is a threshold parameter of the output neuron, b_i is a threshold parameter of the i-th hidden neuron, m is the number of input neurons (it means the input is m dimensional variable) and n is the number of hidden neurons, and θ represents all of the variable parameters simply.

This neural network can represent probability distributions. If X and Y are random variables, we denote $P(Y|X)$ as a conditional probability distribution of Y, given X. For discrete random variables, normalized output neurons can be regarded as probabilistic distribution (the sum of them equals 1.0). For continuous random variables Y according to a single Gaussian distribution, the conditional probability of Y, given instance x is denoted as,

$$P(Y|x) = G(\mu(x), \sigma(x)),$$

$\mu(x)$ and $\sigma(x)$ are approximated by two outputs of a neural network as,

$$\mu(x) \approx f_1(x;\theta),\ \sigma(x) \approx f_2(x;\theta).$$

In the case of the sum of weighted M Gaussians (Gaussian mixture), the distribution is approximated as,

$$P(Y|x) = \sum_{k=1}^{M} \alpha_k G(f_{2k-1}(x), f_{2k}(x)).$$

by a neural network with $2M$ output neurons.

Neural networks can treat multi-valued, multi-dimensional, discrete, continuous variable, or some mixture of them in the same style. If a variable is k-dimensional, the variable is represented with k neurons in the layer. If it is a l-valued discrete variable, the variable can be represented with l neurons in the layer.

2.2 BNNN

Bayesian networks (also called belief networks) are probabilistic reasoning models that represent the dependency between variables and give a concise specification of the joint probability distribution. It is defined as a directed acyclic graph (DAG), which has conditional probabilities in each node. Nodes mean random variables in a domain. Directed links in the graph mean dependency between nodes.

To simplify, let us consider about a simplest network $X \rightarrow Y$. X is called the *parent node* and Y is called the *child node*. The child node has a $P(Y|X)$. When information about X is given, *Belief* of Y (or posterior probability distribution of Y given $P(X)$) is computed by marginalizing operations,

$$BEL(Y = y) \quad = \quad \int_{x \in \Omega_X} P(y|x)\, dP(x). \qquad (1)$$

The other hand, when we have $P(Y)$ at first, *Belief* of X is updated by,

$$\begin{aligned} BEL(X = x) \quad &= \quad \int_{y \in \Omega_Y} P(x|y)\, dP(y) \\ &= \quad \alpha \int_{y \in \Omega Y} P(y|x)\, dP(y) \cdot P(x), \quad (2) \end{aligned}$$

where α is a normalizing constant. For more complex Bayesian networks, such computations are propagated through all links in the network.

However, for many kinds of variables, two points need to be considered. First, how can we represent conditional probabilities naturally? Second, how can we obtain for real world problems? In order to give a solution for the questions, we apply neural networks that represent conditional probabilities in the Bayesian network. As seen in the previous section, we can deal with many variety of random variables and conditional probabilities can be approximated by learning from samples. We call the model Bayesian network that learns conditional probabilities by neural networks (BNNN). For example (Fig. 1), the child node of the Bayesian network is represented by the output neuron, and the parent nodes A and B correspond to the input neurons. Thus, each child node of the Bayesian network has its own neural network. For discrete variables, integral operation, equation (1) and (2) can be computed exactly, but not for continuous variables. In the latter case, we use Monte Carlo methods.

3 Learning conditional probabilities

In this paper, we assume that a DAG structure of BNNN can be given, thus only learning for conditional

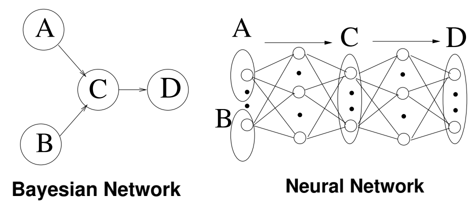

Fig. 1: Structure of BNNN

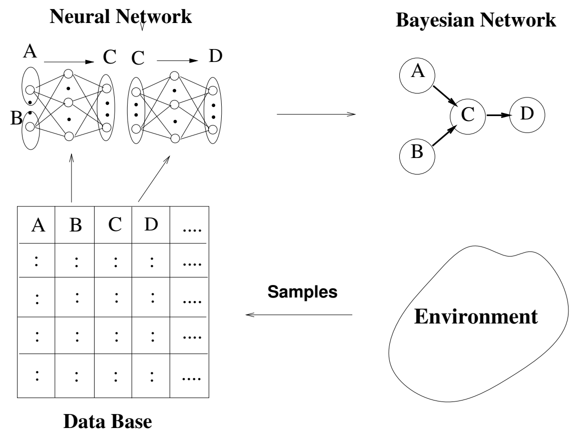

Fig. 2: Learning of BNNN from database

probabilities are focused on. Learning conditional probabilities are achieved by gradient descent methods (called Back Propagation). For discrete conditional probability, a target probability vector is represented by output neurons, and then, they are normalized to make the sum of outputs equals to 1.0. Learning continuous conditional probabilities, $P(Y|X)$ is regarded as approximation for nonlinear functions, $\mu_Y(X)$ and $\sigma_Y(X)$. Given training set of X and Y, $\mu_Y(X)$ is acquired. Training set of $\sigma_Y(X)$ can be given by final error of $\mu_Y(X)$.

After training the neural network successfully, it can represent the conditional dependency between the variables of the training set. This property means that this training scheme can be applied to detect conditional dependency between the variables.

4 Implementation

Bayesian networks are essentially parallel processing models. Monte Carlo method in each node can be executed independently. As these reasons, a multi-thread mechanism and object-oriented features are suitable for implementing BNNN. Therefore, our system, called 'BAYONNET', is implemented in Java that has multi-thread capability. It works also as a probabilistic reasoning server for our Jijo2 robot introduced in next section. It receives requests from other program modules to control the robot and send results through TCP/IP.

The Java applet that we developed is also available at [http://www.etl.go.jp/˜motomura/BN/] (Fig. 3).

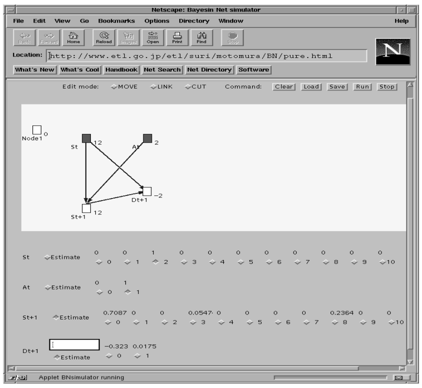

Fig. 3: BAYONNET

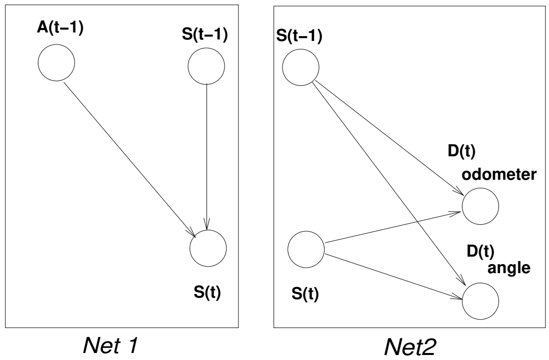

Fig. 4: Bayesian network for robot navigation

5 Experiments

We introduce two case studies using our system.

5.1 Autonomous mobile robot (Jijo2 robot)

The robot's sonar provides distance measures that can distinguish the differences of the situation in the environment. We make the realistic assumption that the robot does not use a 2D geometric map . We don't assume that the robot can know exact position in the map. The information about an environment is only represented as a relation between locations and the robot's actions in its experience. Labels for the locations are given by speech recognition from a human trainer. The robot's locomotive experience is accumulated in a table as shown in table1.

S_{t-1}	A_t	S_t	$D1_t$	$D2_t$
E-232	go forward	E-231	50.0	10.0
E-232	go backward	E-233-1	175.0	5.0
:	:	:	:	

table 1.

Each row of the table represents the starting location S_{t-1}, an action of the robot A_t, the destination location S_t, the value of odometry of this motion $D1_t$, and the value from angle-sensor $D2_t$. S_{t-1}, S_t, A_t are discrete values. $D1_t, D2_t$ are continuous variables.

Due to obstacle avoidance behavior of the robot, and fluctuation caused by slipping, the reading value of the odometry varies at each trial. In order to cope with such uncertainty, each sensor value of the robot is considered as a random variable conditioned by S_{t-1} and S_t. The next location S_t is regarded as a discrete random variable conditioned by S_{t-1} and A_t. This dependency is obtained from the statistical example of the robot's experience in the environment. Thus, the state transition probabilities $P(S_{t-1}|S_t, A_t)$ and observation probabilities $P(Di_t|S_t, S_{t-1}), (i = 1, 2)$ are modeled as a Bayesian network (Fig. 4).

Belief about the current location is evaluated by $P(S_t|S_{t-1}, A_t)$, and belief about the sensor value, given $S_{t-1} and S_t$ is evaluated by $P(D|S_{t-1}, S_t)$. They are learned from the actual data sampled by the robot. For $P(S_{t-1}|S_t, A_t)$, the neural network learns the probabilities' table of them. For $P(D|S_{t-1}, S_t)$, the neural network learns the mean and standard deviation of the Gaussian.

In this experiment, the robot is moving around twelve locations and actions are 'go forward' and 'go backward'. The neural network used here has twenty-four input neurons, ten hidden neurons, and three output neurons for angle data, and two output neurons for odometry data. Evidence variables (input neurons) for Net1 take a start location and an action. Evidence variables for Net2 take the start location (12-dimensional vector), and a destination location (12-dimensional vector) for Net2. In order to get samples for learning, the robot moved with a human trainer in the environment hundred times. In the learning stage, the robot collected destinations for Net1 and sensor observations for Net2 as a training set. The number of learning iterations for each neural network was ten thousand.

As the result of learning, knowledge about the environment is stored in the conditional probabilities of the BNNN. Thus, before the start, the robot can predict possible locations to reach by $P(S_t|S_{t-1}, At-1)$. In a locomotion, the robot can detect the error of sensor value by $P(D_t|S_{t-1}, St)$.

Moreover, the learning results can give flexible function to the robot. The total belief about variable X in the situation S is estimated by the entropy,

$$\int_{x \in X} -P(X|S) log P(X|S) dx.$$

The robot can make appropriate questions by watching the entropy. When belief is high, the robot can select a plausible candidate without questions. These flexible behaviors can decrease the load on a human operator.

5.2 Meteorological data

For an example of learning from a large database, we use Japanese meteorological records in 1995, which was taken every hour for the year, provided by the Japan

Meteorological Agency on CD-ROM. In order to handle the data easily, we loaded it into a commercial database software package, and accessed it through JDBC (JAVA DataBase Connectivity) API from our system.

Generally, we know that weather at a particular point depends on the previous weather in more western areas. The BNNN is constructed according to the dependency (Fig.5). We consider predicting the time when it begins to rain in city X after the news that it starts to rain in city B_i. When we have evidence $\{T_{B_i} | i = 1 \cdots 3\}$, the belief about the rainfall in city X is computed in the BNNN as,

$$P(T_X | T_{B_1}, T_{B_2}, T_{B_3}) = \int_{A_1, A_2} P(T_X | T_{A_1}, T_{A_2}) \prod_{i=1}^{2} P(T_{A_i} | T_{B_i}, T_{B_{i+1}}) dP(T_{A_1}) dP(T_{A_2}). \quad (3)$$

where T_{city} denotes the time of the beginning of rain in the city.

In this experiment, observations at {*Tokyo, Matsumoto, Iida, Fukui, Gifu, Nagoya*} are used as $\{X, A_1, A_2, B_1, B_2, B_3\}$ respectively. The number of data at every hour through the year was 8760, and the number of data about a start to rain was around 150. All conditional probabilities in the BNNN ware trained from the data with at most 10,000 iterations and converged with the mean square error (m.s.e.) less than 0.02. Training data is taken for cases where precipitation exceeds 1.0mm, and the time difference between rainfall for neighbor cities is less than 24 hours. The start of rainfall is taken to be the hour prior to which precipitation was first measured to be more than 1.0mm.

As compared with an ordinary neural network, which has output X and input B_1, B_2, B_3, the BNNN can give slightly better prediction (m.s.e. are 0.018 vs. 0.023), and the BNNN can also give the current accuracy of the prediction by σ_{T_X}.

6 Related works

The relation between Bayesian networks and neural networks is also discussed in a general discussion about graphical networks[Buntine,1994], parameter learning like neural networks[Russell, Binder, and Koller,1994], applying feed forward neural networks to discrete domain[Musick,1994], and sigmoidal belief networks for binary domain[Neal,1991; Jordan,1995]. The study in this paper extends the idea that a neural network is effective for learning in a Bayesian network for a more realistic domain, which includes continuous and discrete variables. It shows a success of a learning conditional dependency for the real world application.

7 Concluding remarks

In this paper, Bayesian network using neural networks was discussed. We have also presented an implementation of the model and the results of learning conditional probability from statistical data sampled in real world domains. The BNNN as a whole is regarded as one kind of neural network: it is a large network constituted of many feed forward networks. As remarkable difference of the BNNN, it can represent the relation between their variables explicitly by the graph of a Bayesian network.

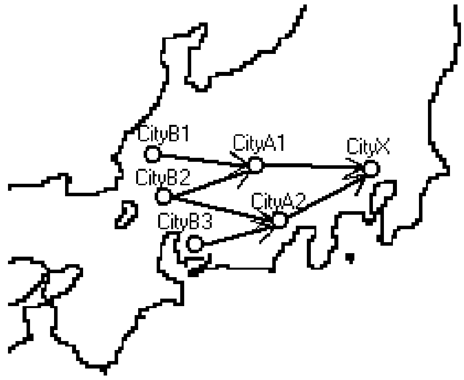

Fig. 5: Bayesian network for rainfall prediction

Acknowledgements

This work is supported by the Real World Computing Program organized by the Ministry of International Trade and Industry of Japan. The authors thank Dr.Steven Phillips for helpful comments, and Prof. Stuart Russell of the Univ. of California at Berkeley for information about R.Musick's dissertation.

References

[Asoh *et al.*,1996] H.Asoh, Y.Motomura, I.Hara, S.Akaho, S.Hayamizu, and T.Matsui, Combining probabilistic map and dialog for robust life-long office navigation. In *Proc. of IROS'96.* 1996.

[Buntine,1994] W.Buntine, Operations for learning with graphical models. *Journal of Artificial Intelligence Research, vol.2.* 1994.

[Jordan,1995] M.Jordan, Why the logistic function? a tutorial discussion on probabilities and neural networks. *MIT Computational Cognitive Science Technical Report 9503.* 1995.

[Matsui *et al.*,1997] T.Matsui, H.Asoh, S.Hayamizu, I.Hara, and Y.Motomura, An office conversant mobile robot that learns by navigation and conversation. *Proc. of 1997 Real World Computing Symposium, Real World Computing Partnership Technical Report TR-96001.* 1997.

[Musick,1994] R.Musick, *Belief Network Induction. PhD thesis, University of California at Berkeley.* 1994.

[Neal,1991] R.Neal, Connectionist learning of belief networks. *Artificial Intelligence, 56.* 1991.

[Pearl,1988] J.Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers. 1988.

[Russell, Binder, and Koller,1994] S.Russell, J.Binder, and D.Koller, Adaptive probabilistic networks. *Technical report UCB//CSD-94-824.* 1994.