Estimation of Non-Gaussian Probability Density by Dynamic Bayesian Networks

Hyun C. Cho*, Sami M. Fadali**, and Kwon S. Lee***

* Electrical Engineering Dept./260, University of Nevada-Reno, NV, 89557, USA
(Tel: 1-775-324-2509, E-mail: hyun@unr.nevada.edu)
** Electrical Engineering Dept./260, University of Nevada-Reno, NV, 89557, USA
(Tel: 1-775-784-6951, E-mail: fadali@ieee.org
*** Division of Electrical, Electronic, and Computer Eng., Dong-A Univ., Busan, 604-714, Korea
(Tel: +82-51-200-7739, E-mail: kslee@dau.ac.kr)

Abstract: A new methodology for discrete non-Gaussian probability density estimation is investigated in this paper based on a dynamic Bayesian network (DBN) and kernel functions. The estimator consists of a DBN in which the transition distribution is represented with kernel functions. The estimator parameters are determined through a recursive learning algorithm according to the maximum likelihood (ML) scheme. A discrete-type Poisson distribution is generated in a simulation experiment to evaluate the proposed method. In addition, an unknown probability density generated by nonlinear transformation of a Poisson random variable is simulated. Computer simulations numerically demonstrate that the method successfully estimates the unknown probability distribution function (PDF).

Keywords: Probability density estimate, Discrete non-Gaussian distribution, Dynamic Bayesian networks, Kernel function

1. INTRODUCTION

Probability estimation is an important issue for many engineering applications, especially, for pattern recognition, signal detection, artificial intelligence, etc. Several parametric and nonparametric techniques are available for the estimation [1]. Parametric methods are the simplest to estimate the parameters of a distribution of known form. A Gaussian distribution is widely used as a parametric density model due to its convenient statistical analysis. However, non-Gaussian distributions are often encountered in practical applications. A nonparametric method is more important to this case for which the form of the distribution is unknown. The histogram is the simplest nonparametric approach available. However, the method is less attractive because of its discontinuous nature and because it requires a large amount of data. Another popular nonparametric method is the kernel-based approach in which an estimator is constructed using a set of kernel functions. The best known kernel-based approach is the Parzen-window estimate [2] which uses Gausssian kernels. For the best performance, an appropriate kernel function and its parameter values are chosen for specific data samples. However, the method is very sensitive to parameter values and it is often difficult to determine the optimal kernel. Additional computation, such as smoothing, is sometimes needed.

In recent years, several approaches to PDF estimation have been proposed, such as soft computation, machine learning, information theory, etc. In [3], Fiori and Bucciarelli proposed estimating the distribution of a quasi-stationary random process by using an adaptive neural network which was trained to maximize the differential entropy. M. Srikanth *et al*. utilized a MinMax measure to estimate a distribution, which is a quantitative measure of information contained in a given set of moment constraints [4]. A conditional distribution function (CDF) estimate was developed by A. Sarajedini *et al*. in [5] based on sigmoidal network learning and compared to a kernel CDF estimator for a higher dimensional problem. Miller and Horn investigated estimating both a PDF and CDF by using an information-based learning in which encoding and decoding steps were proposed [6]. A generalized Gaussian distribution was involved in [7] where an estimate of a PDF shape parameter was addressed. In [8], Modha and Fainman obtained a PDF estimate using an exponential family based on multilayer feedforward networks, and derived an unsupervised learning algorithm for ML function. Baram and Roth developed an estimation method that maximized the output entropy of a sigmoidal neural network and named the proposed approach density shaping [9].

We propose a new probability model estimation approach for discrete problems using a DBN and a set of kernel functions. A Bayesian network (BN) is a graphical reasoning system that uses a probability distribution to describe stochastic relationships between its nodes. Each node in a BN represents a random variable and edges indicate dependencies between nodes. A DBN is a dynamic extension of a BN based on an evolving probabilistic model. BNs and DBNs are typically used to solve problems with significant uncertainty.

The proposed approach utilizes a DBN in which a transition distribution is replaced with a set of kernel functions and the probability of feasible variables is expressed in terms of prior probabilities. The optimal parameter values in the estimator are determined via a learning algorithm that maximizes the likelihood function using gradient descent optimization. We test the proposed approach using computer simulations that estimate a discrete-type Poisson density then a nonlinear transformation of the discrete Poisson data.

We next review kernel-based PDF estimation and the

DBN. Then we describe our proposed estimation approach and derive its learning algorithm. Finally, we present and discuss our simulation results.

2. KERNEL-BASED PROBABILTY ESTIMATE

A kernel-based estimate is typically composed of a mixture of kernel (basis) functions

$$P(x) = \sum_{i=1}^{N} \alpha_i \phi(x - x_i, \beta_i) \tag{1}$$

where x is a data vector, $P(x)$ is the probability of x, ϕ is a kernel function, and α_i and β_i are parameters. Based on probability axioms, the kernel function must satisfy $\phi \geq 0$ and the parameters α_I must be positive.

$$\int \phi(u) du = 1 \tag{2}$$

The parameters α_i and β_i affect estimation performance. Thus, choosing the parameter values is one of crucial issues in probability estimation. A natural method for choosing the parameters is to plot several curves with different values and visually examine them to search for the best fit. However, numerous applications require an automated or an adaptive optimization approach.

ML estimation [10] is used to determine the optimal parameter values in (1). Assuming independent data points, the objective function is defined as

$$L_o(x \mid \alpha, \beta) = \prod_{i=1}^{N} P(x_i) \tag{3}$$

Applying the natural logarithm to (3), we get

$$\begin{aligned} L(x \mid \alpha, \beta) &= \ln(L_o) \\ &= \sum_{i=1}^{N} \ln P(x_i) \\ &= \sum_{i=1}^{N} \ln \left\{ \sum_{j=1}^{N} \alpha_j \phi(x_i - x_j, \beta_j) \right\} \end{aligned} \tag{4}$$

We maximize (4) to obtain two parameter vectors $\alpha = [\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_N]^T$ and $\beta = [\beta_1 \quad \beta_2 \quad \cdots \quad \beta_N]^T$

$$J = \max_{\alpha, \beta} \sum_{i=1}^{N} \ln \left\{ \sum_{j=1}^{N} \alpha_j \phi(x_i - x_j, \beta_j) \right\} \tag{5}$$

By gradient descent method, the update rules for the parameters are expressed as

$$\alpha_j(t_k + 1) = a_j(t_k) + \eta \frac{\partial J}{\partial \alpha_j} \tag{6}$$

and

$$\beta_j(t_k + 1) = \beta_j(t_k) + \eta \frac{\partial J}{\partial \beta_j} \tag{7}$$

where t_k is discrete time, η is the learning rate. We expand the partial differential equations using the chain rule as

$$\begin{aligned} \frac{\partial J}{\partial \alpha_j} &= \frac{\partial J}{\partial P(x_i)} \frac{\partial P(x_i)}{\partial \alpha_j} \\ &= \sum_{i \neq j}^{N} \frac{1}{P(x_i)} \phi(x_i - x_j, \beta_j) \end{aligned} \tag{8}$$

and

$$\begin{aligned} \frac{\partial J}{\partial \beta_j} &= \frac{\partial J}{\partial P(x_i)} \frac{\partial P(x_i)}{\partial \beta_j} \\ &= \sum_{i \neq j}^{n} \frac{1}{P(x_i)} \left\{ \alpha_j \frac{\partial \phi(x_i - x_j, \beta_j)}{\partial \beta_j} \right\} \end{aligned} \tag{9}$$

The parameter values are determined using the recursions (6) and (7) with the derivatives expressed as (8) and (9).

3. DYNAMIC BAYESIAN NETWORKS

A BN is a graphic model for stochastic relationships. It represents random variables as nodes and their dependencies as edges between the nodes. The random variables are assumed to have a finite number of states. The BN is described by a directed acyclic graph and a table of conditional probabilities. Let $U = (A, B, C, D)$ be a universe of random variables, then a corresponding BN is depicted in Fig. 1.

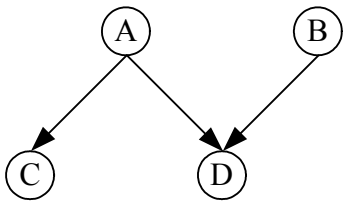

Fig. 1. An example of a BN.

In this network, the probabilities $P(A)$, $P(B)$, $P(C|A)$, and $P(D|A,B)$ must be known. A joint probability $P(U)$ is calculated as

$$\begin{aligned} P(U) &= P(A, B, C, D) \\ &= P(A)P(B)P(C \mid A)P(D \mid A, B) \end{aligned} \tag{10}$$

Inference is the task of calculating the probability of each node by probability calculus from other probabilities. For instance, in Fig. 1, conditional probability of the variable A given B, C, and D is expressed using Bayes rule as

$$P(A \mid B,C,D) = P(A \mid D,C) = \frac{P(D \mid A,C)P(A \mid C)}{P(D \mid C)} \quad (11)$$

because each of the pairs (A, B) and (C, D) is statistically independent. We also have

$$P(A \mid C) = \frac{P(C \mid A)P(A)}{P(C)} \quad (12)$$

Substituting (12) in (11), we obtain

$$P(A \mid B,C,D) = \frac{P(D \mid A)P(C \mid A)P(A)}{P(D)P(C)} \quad (13)$$

Next, we substitute the identity

$$P(D \mid A) = \sum_B P(D \mid A,B) \quad (14)$$

in (13) to get

$$P(B \mid A,C,D) = \frac{\sum_A P(D \mid A,B)P(D)}{P(B)} \quad (15)$$

This BN illustrates a quit simple computation of probability for a particular set of random variables. However, in case of complicated or large Bayesian networks, an efficient inference algorithm is required for a probability updating [11].

DBNs are used to model dynamic events and represents temporally probabilistic relationships among random variables. This temporal (dynamic) aspect of modeling is more realistic in practice such as artificial intelligence, diagnosis, economic analysis, etc. Several DBN models have been proposed for a variety of applications. The simplest DBN example is the first-order Markov model shown in Fig. 2.

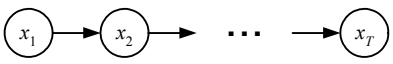

Fig. 2. A first-order Markov model.

In Fig. 2, a state vector $x \in \{x_1, x_2, \ldots, x_T\}$ is modeled at a certain discrete time $k \in \{1,\ldots, T\}$. Each variable is assumed to be first order Markov, i.e. it is dependent only upon the variable at the preceding time. A joint probability for this data sequence is expressed as

$$P(x_1, x_2, ..., x_T) = P(x_1)\prod_{k=2}^{T} P(x_k \mid x_{k-1}) \quad (16)$$

where the conditional probability $P(x_k|x_{k-1})$ is called the transition probability. A higher-order Markov model provides a more sophisticated solution but at an increasingly higher computational cost. A hidden Markov model (HMM) is a popular structure in which an observation variable depends on a hidden state and a sequence of hidden states evolves according to a Markov process. Fig. 3 illustrates a typical example of an HMM.

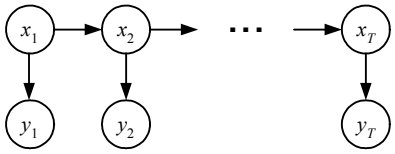

Fig. 3. An example of HMM.

The joint probability of the hidden states and observation variables is expressed similarly to the first Markov model as

$$P(\{x_k, y_k\}) = P(y_1)P(y_1 \mid x_1)\prod_{k=2}^{T} P(x_k \mid x_{k-1})P(y_k \mid x_k) \quad (17)$$

where the state transition probability $P(x_k|x_{k-1})$ is formed by a single transition matrix for a time-invariant HMM. The observation probability $P(y_k|x_k)$ is usually modeled as Gaussian, a combination of Gaussians, or a neural network. In (17), the transition state x_k and observation state y_k are decomposed into deterministic and stochastic terms as

$$\begin{aligned} x_k &= f_k(x_{k-1}) + w_k \\ y_k &= g_k(x_k) + v_k \end{aligned} \quad (18)$$

where w_k and v_k are random noise vectors, and f and g are transition and observation functions, respectively. If both functions are linear and time-invariant, we have the dynamic linear time-invariant model

$$\begin{aligned} x_k &= \mathrm{A}x_{k-1} + w_k \\ y_k &= \mathrm{C}x_k + v_k \end{aligned} \quad (19)$$

where A is the state transition matrix and C is the observation matrix. If the two noise vectors in (19) are modeled as zero-mean Gaussian with covariance matrices Q and R, respectively, the transition and observation probabilities are

$$\begin{aligned} P(x_k \mid x_{k-1}) &\sim N(\mathrm{A}x_{k-1}, \mathrm{Q}) \\ P(y_k \mid x_k) &\sim N(\mathrm{C}x_k, \mathrm{R}) \end{aligned} \quad (20)$$

where $N(\cdot)$ denotes Gaussian.

After constructing the Bayesian network, a learning procedure is developed, based on given data samples

and prior knowledge. Bayesian network learning is concerned with determining the parameter values and the structure of the network: the former (parameter learning) determines the conditional probabilities for each state, and the latter (structure learning) selects the network model under given causal constraints. See [12] for a detailed description of BN learning.

4. DENSITY ESTIMATE BY A BN

In this Section, we propose a new probability estimation method using a kernel-based approach (Section II) and a BN (Section III). In Fig. 2, the posterior probability of a state vector x at k+1 is given by

$$P(x(k+1)) = \sum_{i=1}^{N} P(x(k+1) \mid x_i(k)) P(x_i(k)) \tag{21}$$

where $P(x_i)$ is prior probability of x_i. For specific values z_i, $z_i \in \Re^N$, of x_i in (21) we have

$$P(x(k+1)) = \sum_{i=1}^{N} P(x(k+1) \mid z_i(k)) P(z_i(k)) \tag{22}$$

In (22), the transition distribution is obtained by a combination of kernel functions as in (1). Thus, the proposed estimate is formed as

$$P(x(k+1)) = \sum_{i=1}^{N} \left\{ \sum_{j=1}^{N} \alpha_{ij} \phi(x - z_j, \beta_j) \right\} P(z_i(k)) \tag{23}$$

This estimate is sequentially updated for a random vector x_k at time k. The structure of this estimate is depicted in Fig. 4.

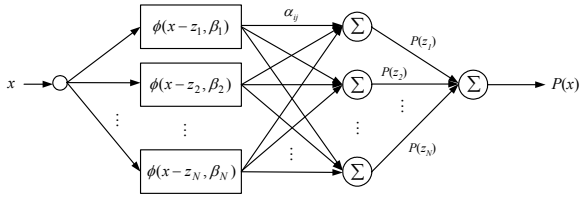

Fig. 4. The proposed estimate structure.

As stated in Section II, the parameter vectors α and β in Fig. 4 are recursively adjusted via sequential learning. The objective function is similarly given by

$$\begin{aligned} J &= \max_{\alpha,\beta} \sum_{n=1}^{T} \ln\{P(x_n)\} \\ &= \max_{\alpha,\beta} \sum_{n=1}^{T} \ln\left\{ \sum_{i=1}^{N} \left[\sum_{j=1}^{N} \alpha_{ij} \phi(x_n - z_j, \beta_j) \right] P(z_i) \right\} \end{aligned} \tag{24}$$

Using (6), (7), (8) and (9) of Section II, the adjustment rules of the parameters are obtained as

$$\alpha_{ij}(t_k + 1) = \alpha_{ij}(t_k) + \eta \sum_{n=1}^{N} \left\{ \frac{1}{P(x_n)} \phi(x_n - z_j, \beta_j) P(z_i) \right\} \tag{25}$$

and

$$\beta_j(t_k + 1) = \beta_j(t_k) + \eta \sum_{n=1}^{N} \frac{1}{P(x_n)} \left\{ \sum_{i=1}^{N} \alpha_{ij} P(z_i) \frac{\partial \phi(x_n - z_j, \beta_j)}{\partial \beta_j} \right\} \tag{26}$$

5. EXAMPLES AND RESULTS

We simulate the proposed method to estimate the discrete probability vector of a non-Gaussian signal. In (25) and (26), a kernel function is used with the Gaussian model

$$\phi(x_n - z_j, \beta_j) = \frac{1}{\sqrt{2\pi}\beta_j} \exp\left\{ -\frac{(x_n - z_j)^2}{\beta_j^2} \right\} \tag{27}$$

For simplicity, the prior probabilities of the values z_i i = 1, …, N, in (23) are assumed equal, i.e. $P(z_i) = 1/N$.

Example I: First, we simulate the estimation of the Poisson distribution

$$P(x = k) = \exp(-\lambda) \frac{\lambda^k}{k!} \tag{28}$$

where λ is a parameter and k = 0, 1, …, ∞. Fig. 5 shows N=100 Poisson distributed data samples for λ=10 which are generated using the MATLAB© command *poissrnd*.

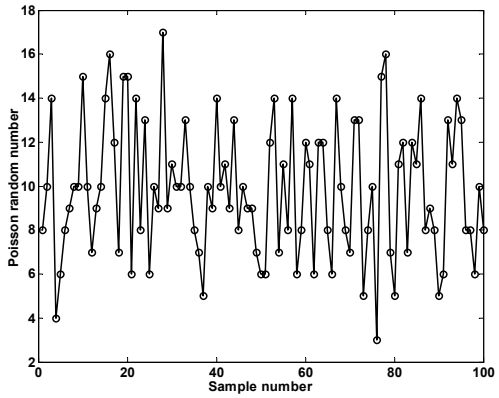

Fig. 5. Poisson random number (*Example* I).

Initial values of α and β are randomly selected as uniformly distributed in [0,1] with the learning rate η = 0.5. Recursive learning continues until a specified error tolerance is reached. The error function is the average absolute difference between a reference probability p^* and the estimated probability P i.e.

$$e = \frac{1}{N}\sum_{n=1}^{N}\left|P^*(x) - P(x)\right| \qquad (29)$$

In the training simulation, 100 training data samples at each time k were temporally generated. Fig. 6 illustrates simulation result of the estimated probabilities along with the reference values. The plot shows that the estimation errors are less than 0.01 at all data point with an average error value of 0.0089.

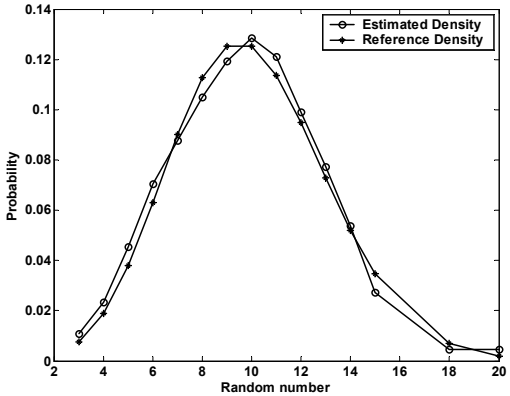

Fig. 6. Estimated probability (*Example* I).

Example II: We nonlinearly transform Poisson random data in this *Example*. To simulate this scenario, a random input is fed to a nonlinear system and the probabilities of the system output are estimated. A block diagram for this process is depicted in Fig. 7.

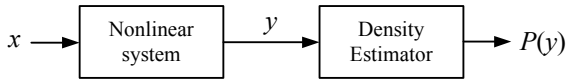

Fig. 7. Transformation of a random variable.

The random input is Poisson distributed with the same parameters as *Example* I. Because the data is nonlinearly transformed, the output probabilities are not Poisson and the output distribution is of an unknown form. We define the nonlinear function as $y = cx^3(k)$ where c is constant and generate 500 random input values. A time history of the output is shown in Fig. 8.

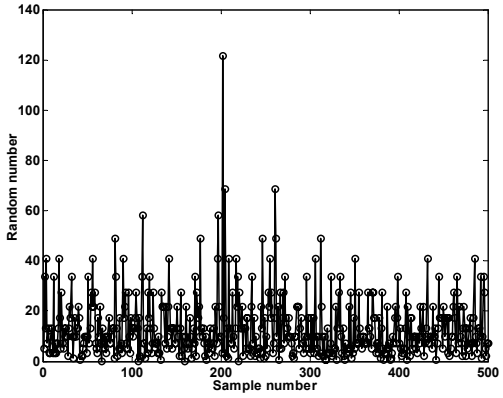

Fig. 8. Poisson random number (*Example* II).

Training is accomplished as in *Example* I. For comparison, the histogram method is additionally performed using the MATLAB command *hist*. The simulation results are given in Fig. 9. The results appear acceptable in comparison with those of the histogram approach.

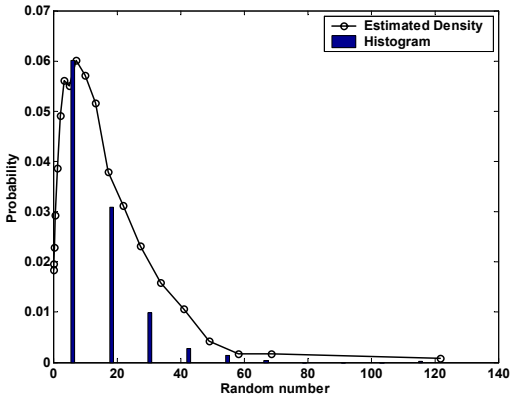

Fig. 9. Estimated probability (*Example* II).

6. DISCUSSION

We propose a probability estimation approach using DBN. The method was successfully used to estimate discrete non-Gaussian signals. From the simulation results, we conclude that the performance of the approach is acceptable.

In future work, we will address the estimation of more complex distributions such as multi-variable non-Gaussian or non-stationary stochastic models. To this end, we will utilize and compare well-known algorithms such as the Viterbi algorithm [13].

ACKOWLEDGEMENTS

The authors are grateful for the support of NRL, National Research Laboratory in Dong-A University, Korea.

REFERENCES

[1] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern classification*, A Wiley-Interscience Publication, NY, 2001.

[2] E. Parzen, "On estimation of a probability density function and mode," *Analysis of Mathematical statistics*, Vol. 33, pp. 1065 - 1076.

[3] S. Fiori and P. Bucciarelli, "Probability density estimation using adaptive activation function neurons," *Neural Processing Letters*, Vol. 12, pp. 31-42, 2001.

[4] M. Strikanth, H. K. Kesavan, and P. H. Roe, "Probability density function estimation using the MinMax measure," *IEEE Trans. on Systems, Man, and Cybernetics-Part C*: *Applications and Reviews*, Vol. 30, No. 1, pp. 77-83, 2000.

[5] A. Sarajedini, R. Hecht-Nielsen, and P. M. Chau, "Conditional probability density function estimation with sigmoidal neural networks," *IEEE Trans. on Neural Networks*, Vol. 10, No. 2, pp. 231-238, 1999.

[6] G. Miller and D. Horn, "Maximum entropy approach to probability density estimation," *International Conference on Knowledge-Based Intelligent Electronic Systems*, Vol. 1, pp. 225-230, 1998.

[7] K. Sharifi and A. Leon-Garcia, "Estimation of shape parameter for generalized Gaussian distributions in subband decompositions of video," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 5, No. 1, pp. 52-56,1995.

[8] D. S. Modha and Y. Fainman, "A learning law for density estimation," *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, pp. 519-523, 1994.

[9] Y. Baram and Z. Roth, "Forecasting by density shaping using neural networks," *IEEE/IAFE Proc. of Computational Intelligence for Financial Engineering*, pp. 57-71, 1995.

[10] J. M. Mendel, *Lessons in estimation theory for signal processing, communications, and control*, Prentice Hall, NJ, 1995.

[11] F. V. Jensen, *Bayesian networks and decision graphs*, Springer, NY, 2001.

[12] D. Heckerman, "A tutorial on learning with Bayesian networks," *Technical report MSR-TR-95-06*, Microsoft Research, Washington, 1996.

[13] G. D. Forney, "The Viterbi algorithm," *Proc. of The IEEE*, Vol. 61, No. 3, pp. 268-278, 1973.