PUBLICATION OF Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st May 2011. Vol. 27 No.2
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A NOVEL BASED APPROACH FOR UNCERTAIN DATA CLASSIFICATION USING PROBABILISTIC NEURAL NETWORKS

[1]G.V.SURESH, [2]K.SURESH BABU [3]K.KARUNAKAR,[4]N.VIJAYA KUMAR
[1]Assoc Prof., Department of Computer Sciences & Engineering, UCET, Guntur, India-522438
[2]Assoc Prof., Department of Computer Sciences & Engineering, VVIT, Guntur, India-522438
[3]Assoc Prof., Department of Computer Sciences & Engineering, UCET, Guntur, India-522438
[4]Assoc Prof., Department of Computer Sciences & Engineering, UCET, Guntur, India-522438
E-mail: vijaysuresh.g@gmail.com , kollurusuresh@gmail.com, pltkaruna@gmail.com

ABSTRACT

Data uncertainty is common in real-world applications due to various causes, including imprecise measurement, network latency, out-dated sources and sampling errors. These kinds of uncertainty have to be handled cautiously, or else the mining results could be unreliable or even wrong. We propose that when data mining is performed on uncertain data, data uncertainty has to be considered in order to obtain high quality data mining results. We present a Probabilistic Neural Network model which is suitable for classification problems. This model constitutes an adaptation of the classical RBF network where the outputs represent the class conditional distributions. Since the network outputs correspond to probability densities functions, training process is treated as maximum likelihood problem and an Expectation-Maximization (EM) algorithm is proposed for adjusting the network parameters. Experimental results show that proposed model exhibits superior classification performance on uncertain data.

Keywords: *Uncertain data, RBF Network, Maximum likelihood Problem Data Mining, Expectation-Maximization (EM) algorithm*

1. INTRODUCTION

Data is often associated with uncertainty because of measurement inaccuracy, sampling discrepancy, outdated data sources, or other errors. This is especially true for applications that require interaction with the physical world, such as location-based services [1] and sensor monitoring [3]. For example, in the scenario of moving objects (such as vehicles or people), it is impossible for the database to track the exact locations of all objects at all-time instants. Therefore, the location of each object is associated with uncertainty between updates [4]. These various sources of uncertainty have to be considered in order to produce accurate query and mining results. We note that with uncertainty, data values are no longer atomic. To apply traditional data mining techniques, uncertain data has to be summarized into atomic values. Taking moving-object applications as an example again, the location of an object can be summarized either by its last recorded location or by an expected location. Unfortunately, discrepancy in the summarized recorded value and the actual values could seriously affect the quality of the mining results. In recent years, there is significant research interest in data uncertainty management. Data uncertainty can be categorized into two types, namely existential uncertainty and value uncertainty. In the first type it is uncertain whether the object or data tuple exists or not. For example, a tuple in a relational database could be associated with a probability value that indicates the confidence of its presence. In value uncertainty, a data item is modelled as a closed region which bounds its possible values, together with a probability density function of its value. This model can be used to quantify the imprecision of location and sensor data in a constantly-evolving environment.

1.1. Uncertain Data Mining

There has been a growing interest in uncertain data mining[1], including clustering[2], [3], [4], [5], classification[6], [7], [8], outlier detection [9], frequent pattern mining [10], [11], streams mining[12] and skyline analysis[13] on uncertain data, etc. An important branch of mining uncertain data is to build classification models on uncertain data. While [6], [7] study the classification of uncertain data using the support vector model, [8] performs classification using decision trees. The focus is on the use of a probabilistic neural network (PNN) for representing the distribution of feature vectors of each class in order to generate a feature-label interaction constraint. The classical PNN is similar to an "intelligent memory" since each training pattern is stored as one unit of the layer of Gaussians. Algorithms that train PNNs are therefore infeasible for large datasets because the resulting network contains as many neurons as there are patterns in the training dataset. PNN is often an excellent pattern classifier, outperforming other classifiers including backpropagation. This paper unprecedentedly explores yet another model, using Probabilistic Neural Networks, and extends them to handle uncertain data. For uncertain classification problems, however, we should learn the class conditional density from uncertain data objects represented by probability distributions.

2. RESEARCH BACKGROUND

2.1 Neural Networks for Data Mining

Neural networks are suitable in data-rich environments and are typically used for extracting embedded knowledge in the form of rules, quantitative evaluation of these rules, clustering, self-organization, classification and regression, feature evaluation and dimensionality reduction. The following are the major stages in solving a DM problem. The entire DM process is iterative, and the result in each step can be feed back to any of the previous steps for improvement. The loop will continue until a satisfactory result has been obtained. A lot of work in current DM has been done in developing integrated systems to support all 7 stages not only stages 5 and 6 that are typical for NN and machine learning. There are many nice features of NN, which make them attractive for DM. These features include learning and generalization ability, adaptivity, content addressability, fault tolerance, self-organization, robustness, and simplicity of basic computations. NN are useful especially when there is no a priori knowledge about the analyzed data. They offer a powerful and distributed computing architecture, with significant learning abilities and they are able to represent highly nonlinear and multivariable relationships

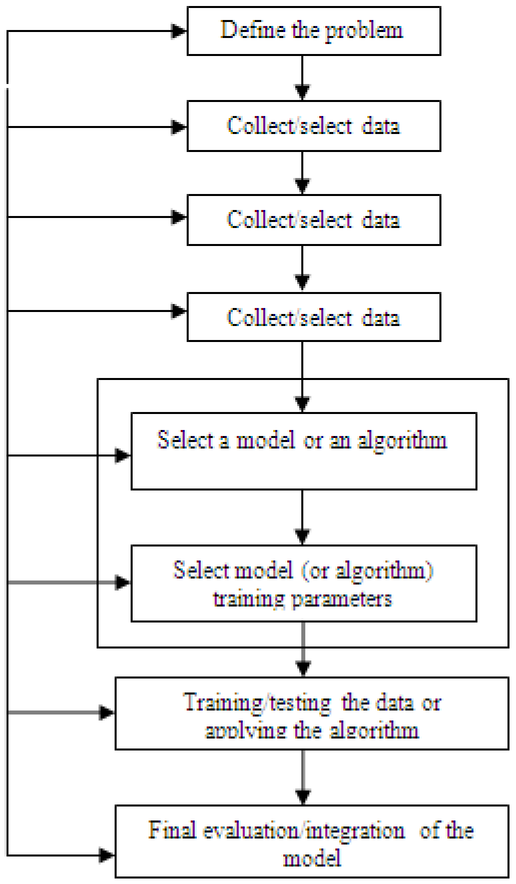
Define the problem
Collect/select data
Collect/select data
Collect/select data
Select a model or an algorithm
Select model (or algorithm) training parameters
Training/testing the data or applying the algorithm
Final evaluation/integration of the model

Figure 1: Data modeling process

2.2 Probabilistic Neural Networks

In pattern recognition it is well-known that a convenient way to consider a classifier is on the basis of inferring the posterior probabilities of each class. From the statistical point of view this inference can be achieved by first evaluating the class conditional densities $P(x \mid k)$ and the corresponding prior probabilities $P(k)$ and then making optimal decisions for new data points by combining these quantities through Bayes theorem

$$P(k \mid x) = \frac{p(x \mid k)P(k)}{\sum_{k'} p(x \mid k')P(k')} \qquad (1)$$

and selecting the class with maximum $P(k \mid x)$. Consequently, the above approach is based on the evaluation of each class conditional density $P(x \mid k)$ which is estimated separately by considering only the data points of the corresponding class k. In the proposed probabilistic network we combine characteristics of the statistical and neural approaches. In particular, the developed RBF neural network which provides

PUBLICATION OF Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st May 2011. Vol. 27 No.2
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

output values corresponding to the class conditional densities $P(x \mid k)$. Since the network is RBF, the kernels are shared among classes and each class conditional density is evaluated using not only the corresponding class data points (as in the traditional statistical approach), but using all available a points. In order to train the PRBF network, an Expectation-Maximization (EM) algorithm been derived, which provides a fast iterative procedure for adjusting the network parameters.

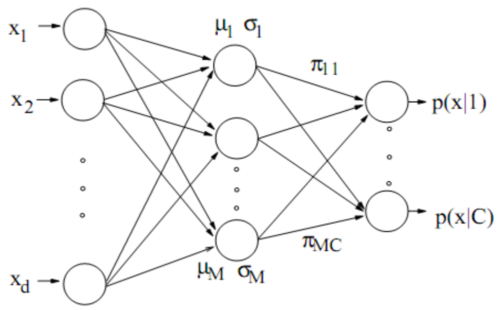

Figure 2: The architecture of the probabilistic network

2.3 Description of the Network

Consider a classification problem with **C** classes and a training set **X** having **N** supervised $(x^n k^n)$ where $x^n \in R^d$ and k^n is an integer indicating the class of the pattern x^n. The original set **X** can easily be partitioned into **C** independent subsets $\mathbf{X_k}$ with $\mathbf{N_k}$ elements, where $k = 1\ldots$, C, so that each subset contains only the data of the corresponding class. As mentioned in the introduction our intention is to model the class conditional densities using an RBF network. The network architecture is displayed in Fig. 2. Typically this probabilistic network has d input units and C output units (one for each class). The main difference with a classical RBF network lies on the specific functional form of the basis functions which are considered to be densities functions as well as on some constraints involving weights from the hidden to the output layer. More specifically, each basis function j (j = 1. . . M) in the hidden layer is Gaussian kernel of the form

$$P(x \mid j) = \frac{1}{\sqrt{(2\pi\sigma_j^2)^d}} \exp\{-\frac{\| x - \mu_j \|^2}{2\sigma_j^2}\} \qquad (2)$$

where μ_j is a vector representing the center of the j kernel and σ_j^2 is the corresponding variance. This specific form of the kernel function assumes that the components of the pattern x are independent and can be represented by a common variance. Each output unit k provides density function values $P(x \mid k)$ for the corresponding class k in the following way:

$$p(x \mid k) = \sum_{j=1}^{M} \pi_{jk} p(x \mid j) \qquad (3)$$

where the weight π_{jk} represents the prior probability that a data point has been generated from kernel, j given that it belongs to class k. These parameters satisfy the constraints

$$\sum_{j=1}^{M} \pi_{jk} = 1 \; and \; \pi_{jk} \geq 0 \text{ for every } k \qquad (4)$$

Using the Bayes theorem we can compute the a posterior probability $P(j \mid k, x)$ that kernel j is responsible for generating pattern x given that it belongs to class k

$$P(j \mid k, x) = \frac{\pi_{jk} p(x \mid j)}{\sum_{j'} \pi_{j'k} p(x \mid j')} \qquad (5)$$

From the above it is clear that the adjustable parameters of the PNN network are the means μ_j and σ_j^2 variances of the M Gaussian kernels and also the priors π_{jk}. We denote the whole parameter vector by θ.

2.4 Maximum Likelihood

Let P(k) where k = 1. . .C denotes the prior probability of the k class. In order to use Bayes rule (1) for unlabeled input data we have to find first appropriate values for both prior probabilities and parameter vector θ (PNN). The whole adjustable parameter vector is θ' = (θ, P (1). . . P(C)). Assuming that all data points have been independently drawn from an underlying process, we can write the log-likelihood function of the dataset X as

$$L(\theta') = \log p(X \mid \theta') = \sum_{n=1}^{N} \log p(x^n, k^n) \qquad (6)$$

Using the relation $p(x, k)$ = P(k) p(x|k) and the fact that the dataset X consists of C independent subsets X_k, the above equation takes the for

$$L(\theta') = \sum_{k=1}^{C} N_k \log P(k) + \sum_{k=1}^{C} \sum_{n=1}^{N_k} \log p(x^n \mid k) \qquad (7)$$

Maximization of the first term in the above equation yields P (k) = N_k/N (k = 1. . . C), while the maximization of the second term is equivalent to training the PNN network. Consequently, the appropriate log-likelihood function for PRBF training is given by

$$L(\theta) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \log p(x^n \mid k) \qquad (8)$$

In order to maximize *L(θ)* it is possible to employ computationally intensive nonlinear optimization techniques. Nevertheless, since we seek maximum likelihood estimates, it is also possible to employ the iterative EM algorithm

2.4 The EM algorithm

The Expectation-Maximization (EM) algorithm is a general technique for maximum likelihood estimation. The algorithm assumes the existence of two data sets; the incomplete data set that consists of the actual observations and the hypothetical complete data set which contains some additional values called unobservable or hidden variables. The notion of hidden variables suggests that the problem to be solved would be straightforward if these variables were known. One iteration of the EM algorithm consists of two steps: i) the expectation step (E-step) where the expected value of the log-likelihood of the complete data set is evaluated, given the current parameter vector and the incomplete data set and ii) the maximization step (M-step) where this expected value is maximized with respect to the parameters of the model. In order to apply the EM for maximizing we have to express the complete data set; the corresponding uncertain data set is X. We express this uncertain information by introducing for each data point x^n a variable z^n which is an M-dimensional vector of one-zero values specifying the kernel that generated x^n. If x^n was generated from kernel j, then, $z_j^n = 1$ otherwise. $z_j^n = 0$.

Using these hidden variables the complete data set Y is defined as follows:

$$Y = \{y^1, \ldots, y^n\}, \text{Where } y^n = (x^n, k^n, z^n) \qquad (9)$$

and the corresponding log-likelihood function is written in the form

$$L_C(\theta) = \sum_{n=1}^{N} \sum_{j=1}^{M} z_j^n \log\{\pi_{jk^n} p(x^n \mid j)\} \qquad (10)$$

At the t + 1 iteration the current expected value of the z_j^n, given the data point x_n is equal to the posterior probability $P^{(t)}(j \mid k^n, x^n)$ where t reminds us that this probability have been evaluated using the current parameters $\theta^{(t)}$. Eventually, the quantity to be maximized in the M-step is given by

$$Q(\theta;\theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j \mid k, x^n)\{\log \pi_{jk} + \log p(x^n \mid j)\} \qquad (11)$$

It can be shown that M-step is analytically implemented. The above equation can be written as Q = $Q_1 + Q_2$ where

$$Q_1(\theta;\theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j \mid k, x^n) \log \pi_{jk} \qquad (12)$$

and

$$Q_2(\theta;\theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j \mid k, x^n) \log(x^n \mid j) \qquad (13)$$

The quantity Q_1 depends solely on the parameters π_{jk}, while the quantity Q_2 depends solely on the parameters of the kernels. In order to maximize Q_1 we must take into account the constrain $\sum_{j=1}^{M} \pi_{jk} = 1$ $\pi_{jk} = 1$ which holds for every class k. Therefore we introduce C multipliers λ_k and after performing some algebra we finally find that the update equation of the prior π_{jk} at the M-step is

$$\pi_{jk}^{(t+1)} = \frac{1}{N_k} \sum_{k=1}^{N_k} P^{(t)}(j \mid k, x^n) \qquad (14)$$

for k = 1, . . . ,C and j = 1, . . . ,M . Taking the derivatives of Q_2 with the respect to μ_j and σ_j^2 respectively and setting them to zero we obtain the following equations for j = 1. . . M

$$\mu_j^{t+1} = \frac{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j \mid k, x^n) x^n}{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j \mid k, x^n)} \qquad (15)$$

$$(\sigma_j^2)^{(t+1)} = \frac{1}{d} \frac{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j \mid k, x^n) \| x^n - \mu_j^{t+1} \|^2}{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j \mid k, x^n)} \qquad (16)$$

Starting from some initial parameter values, we perform alternatively the E-step and M-step until we reach convergence.

3. RELATED WORKS

3.1 How the PNN Works

PUBLICATION OF Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st May 2011. Vol. 27 No.2
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A probabilistic neural network (PNN) has 3 layers of nodes. The Figure.2 below displays the architecture for a PNN that recognizes K = 2 classes, but it can be extended to any number K of classes. The input layer (on the left) contains N nodes: one for each of the N input features of a feature vector. These are fan-out nodes that branch at each feature input node to all nodes in the hidden (or middle) layer so that each hidden node receives the complete input feature vector x. The hidden nodes are collected into groups: one group for each of the K classes as shown in the Figure.3

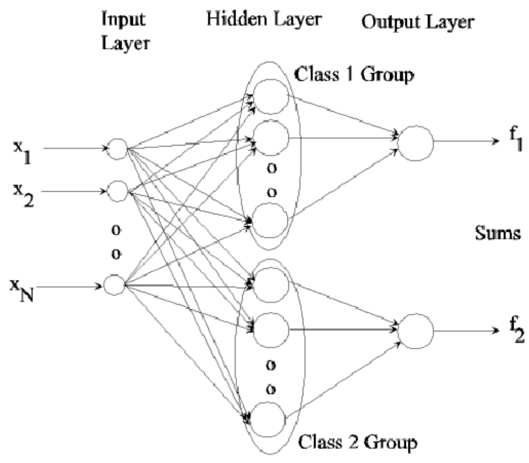

Figure 3: Probabilistic Neural Network

At the output node for Class k (k = 1 or 2 here), all of the Gaussian values for Class k are summed and the sum is scaled to so the probability volume under the sum function is unity so that the sum forms a probability density function. Here we temporarily use special notation for clarity. Let there be **P** exemplar feature vectors $\{\mathbf{x}^{(p)}: = 1,..., \mathbf{P}\}$ labeled as Class 1 and let there be **Q** exemplar feature vectors $\{ \mathbf{y}^{(r)} : = 1,...,R \}$ labeled as Class 2. In the hidden layer there are P nodes in the group for Class 1 and R nodes in the group for Class 2. The equations for each Gaussian centered on the respective Class 1 and Class 2 points $\mathbf{x}^{(p)}$ and $\mathbf{y}^{(q)}$ (feature vectors) are (where **N** is the dimension of the vectors) are, for any input vector **x**.

$$g_1(\mathbf{x}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{x}-\mathbf{x}^{(p)}\|^2/(2\sigma^2)\} \quad (17)$$

$$g_2(\mathbf{y}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{y}-\mathbf{y}^{(q)}\|^2/(2\sigma^2)\} \quad (18)$$

The **F** values can be taken to be one-half the average distance between the feature vectors in the same group or at each exemplar it can be one-half the distance from the exemplar to its nearest other exemplar vector. The kth output node sums the values received from the hidden nodes in the kth group, called mixed Gaussians or Parzen windows. The sums are defined by

$$f_1(\mathbf{x}) = [1/\sqrt{(2\pi\sigma^2)^N}](1/P)\Sigma_{(p=1,P)}\exp\{-\|\mathbf{x}-\mathbf{x}^{(p)}\|^2/(2\sigma^2)\} \quad (19)$$

$$f_2(\mathbf{y}) = [1/\sqrt{(2\pi\sigma^2)^N}](1/Q)\Sigma_{(q=1,Q)}\exp\{-\|\mathbf{y}-\mathbf{y}^{(q)}\|^2/(2\sigma^2)\} \quad (20)$$

where **x** is any input feature vector, σ_1 and σ_2 are the spread parameters (standard deviations) for Gaussians in Classes 1 and 2 , respectively, **N** is the dimension of the input vectors, **P** is the number of center vectors in Class 1 and **R** is the number of centers in Class 2, $\mathbf{x}^{(p)}$ and $\mathbf{y}^{(q)}$ are centers in the respective Classes 1 and 2, and $|\mathbf{x}-\mathbf{x}^{(p)}\|$ is the Euclidean distance (square root of the sum of squared (p)differences) between **x** and, $\mathbf{x}^{(p)}$. Any input vector **x** is put through both sum functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$and the maximum value (maximum a posteriori, or MAP value) of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$decides the class. For K > 2 classes the process is analogous. There is neither iteration nor computation of weights. For a large number of Gaussians in a sum, the error buildup can be significant. Thus the feature vectors in each class may be reduced by thinning those that are too close to another one and making σ larger.

3.2 PNN Algorithm to handle Uncertain Data

Step 1. Read in the file of exemplar vectors and class numbers
Step 2. Sort these into the K sets where each set contains one class of vectors
Step 3. Determine the number of kernels M and the initial parameter vector $\theta^{(0)}$

Once the PNN is defined, then we can feed vectors into it and classify them as follows
Step 4. Read input vector and feed it to each Gaussian function in each class
Step 5. For each group of hidden nodes, compute all Gaussian functional values at the hidden nodes
Step 6. Set t: = 0 and compute the initial log-likelihood L (0)
Repeat
(a) *E*-step: Compute $P^{(t)}(j\,|\,k,x^n)$ for each $k=1,....,C, n=1,....N_k$ and $j=1,....,M$
(b) M-step: Compute the new parameter values $\pi_{jk}^{(t+1)}$, $\mu_j^{(t+1)}$ and $(\sigma_j^2)^{(t+1)}$ using respectively.

(c) Set t: = t + 1 and compute the new log-likelihood $L^{(t)}$

until

$| L^{(t)} - L^{(t-1)} |<$ ϵ where ϵ determines the strictness of the convergence criterion.

Step 7. Find maximum value of all summed functional values at the output nodes

4. EXPERIMENTS

4.1. Results

We have implemented the this approach using Matlab 6.5, and test on 2 real data sets taken from the UCI Machine Learning Repository i.e. Glass data, Iris,. We compare the classification performance of this model on this UCI datasets

Table 1. Statistics of the datasets are listed

	Glass	Iris
Number of Data	214	150
Number of features	10	4
Number of Classes	6	3

We Plot the model size and Error rate for the terminal nodes and Misclassification cost for Glass data set by using this model and inferred that the PNN exhibits good classification performance

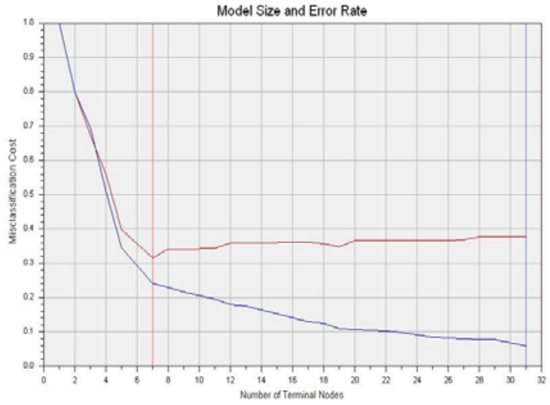

Figure 4: Model size and Error rate for Glass data set

We plot the lift and gain for Virginica spices in Iris data set using this model and found that it performs well when compared with RBF network. This will give PRBF network trained using the EM algorithm provides superior performance in classification of the data.

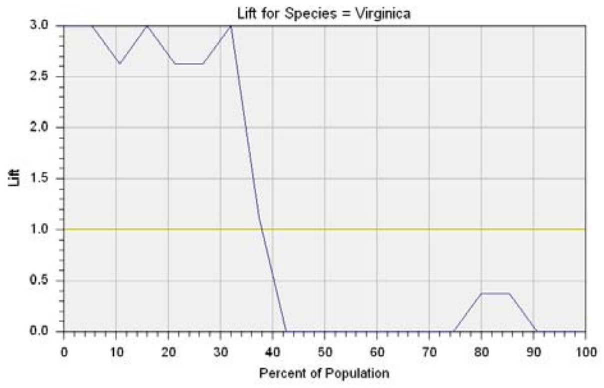

Figure 5: Lift Virginica spices in Iris data set

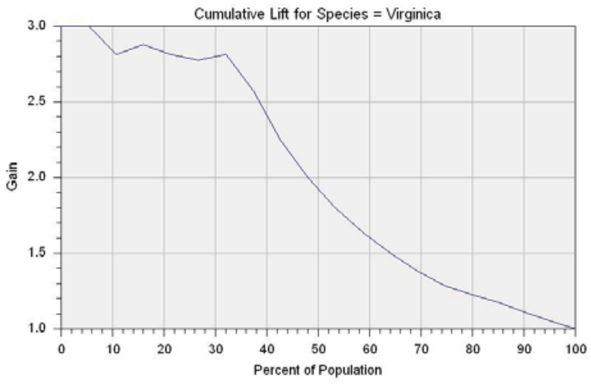

Figure 6: Gain Virginica spices in Iris data set

We started plotting the information for Pima Indian-Diabetes data set to predict the given sample is having Diabetes with the Threshold Probability for having Diabetes is 1 and trained with the model and deduced the results

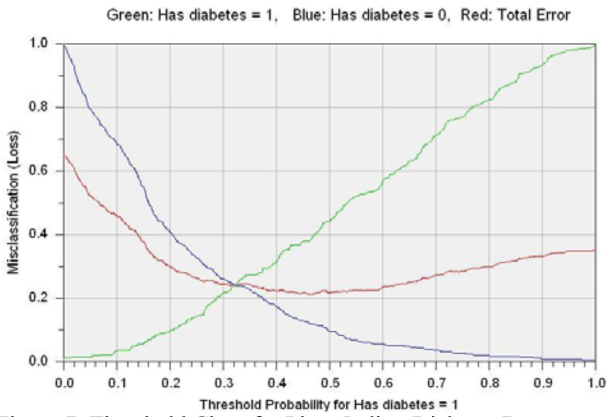

Figure 7: Threshold Chart for Pima-Indian-Diabetes Data

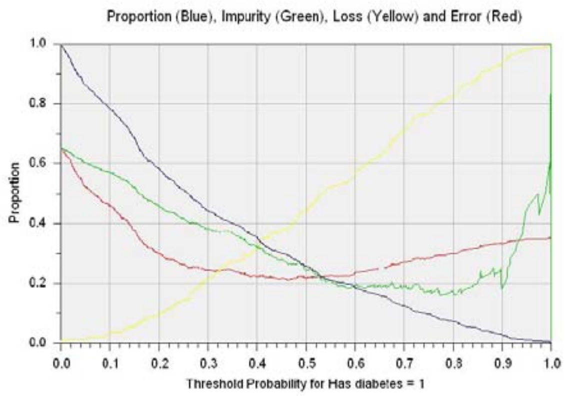

Figure 8: Balanced Chart for Pima-Indian-Diabetes Data

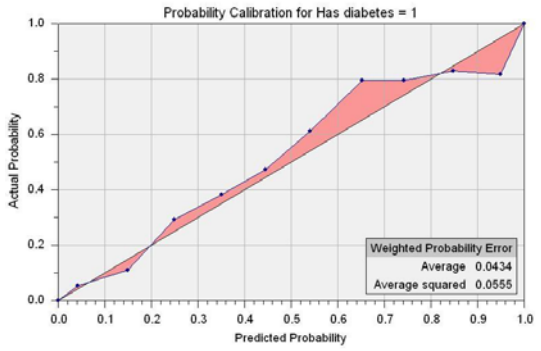

Figure 9: Probability calibration Pima-Indian-Diabetes Data

Table 2. Generalization error for the clouds data set

	Number of kernels		
Algorithm	6	8	12
PNN	11.13	10.46	10.3
RBF	25.46	23.5	22.94

We have chosen one artificial dataset (Clouds), in order to obtain an estimate of the generalization error, we have employed the K-fold cross-validation method with K = 5. Tables 2 provide the obtained results for the PRBF and RBF networks, for several values of the number of kernel functions M. These results indicate that the proposed PRBF network trained using the EM algorithm provides superior performance compared to the classical RBF network. Therefore, the classification and prediction process is more sophisticated and comprehensive and has the potential to achieve higher accuracy

5. CONCLUSION

In this paper, we propose a Probabilistic Neural Network model for classifying and predicting uncertain data. The new process is based on an approximation of the random function around the input mean. This process also highlights the correlation between all the parameters, indicating the nature of the likelihood function, and the potential problems for maximum likelihood optimization. Moreover, the structure of the network is modified to implement another principle of the small sample theory, in that it is able to add a percentage of regularization to the estimation of the probabilities. At the same time the method takes into account the limited availability of resources for the practical realization of a PNN device. We plan to explore more classification approaches for various uncertainty models and find more efficient training algorithms in the future

REFERENCES:

[1] Aggarwal, C.C.: A Survey of Uncertain Data Algorithms and Applications. IEEE Transactions on Knowledge and Data Engineering 21(5) (2009)

[2] Cormode, G., McGregor, A.: Approximation algorithms for clustering uncertain data. In: Principle of Data base System, PODS (2008)

[3] Aggarwal, C.C., Yu, P.: A framework for clustering uncertain data streams. In: IEEE International Conference on Data Engineering, ICDE (2008).

[4] Singh, S., Mayfield, C., Prabhakar, S., Shah, R., Hambrusch, S.: Indexing categorical data with uncertainty. In: IEEE International Conference on Data Engineering (ICDE), pp. 616–625 (2007)

[5] Kriegel, H., Pfeifle, M.: Density-based clustering of uncertain data. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 672–677 (2005). .

[6] Bi, J., Zhang, T.: Support Vector Machines with Input Data Uncertainty. In: Proceeding of Advances in Neural Information Processing Systems (2004)

[7] Aggrawal, R., Imielinski, T., Swami, A.N.: Database mining: A performance perspective.IEEE Transactions on Knowledge& Data Engineering (1993)

[8] Aggarwal, C.C.: Managing and Mining Uncertain Data. Springer, Heidelberg (2009)

[9] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data," in *SDM*. SIAM, 2008, pp. 483–493

[10] Hamdan, H. and Govaert, G. "Mixture Model Clustering of Uncertain Data," *IEEE International Conference on Fuzzy Systems* (2005) 879-884

PUBLICATION OF Little Lion Scientific R&D, Islamabad PAKISTAN
Journal of Theoretical and Applied Information Technology
31st May 2011. Vol. 27 No.2
© 2005 - 2011 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AUTHOR PROFILES:

G.V.Suresh received his B.Tech and M.Tech from JNT University Hyderabad, India in 2004 and 2008 respectively and currently working toward PhD degree from JNT University. Currently, he is an Associate Professor for Computer Science and Engineering, at Universal college of Engineering &Technology. His research interests are in Data Mining Neural Networks, and Networks. He is a life member for Computer Society of India (CSI). He has authored a good number of research papers published in International Journals and Conference Proceedings.

K.Suresh received his B.Tech and M.Tech from JNT University Hyderabad, India currently; he is an Associate Professor for Computer Science and Engineering, at VVIT. His research interests are in Data Mining Neural Networks, and Networks. He is a life member for Computer Society of India (CSI).

K.Karunakar received his M.Tech from Andhra University, Vizag, India in 2005 and 2009 respectively he is an Associate Professor for Computer Science and Engineering, at Universal college of Engineering &Technology. His research interests are in Data Mining Neural Networks, and Networks. He is a life member for Computer Society of India (CSI).

N.Vijaya Kumar M.E (C.S.E) Sathyabama University with 9 years experience ,working as a Assoc.Prof & Head of the department of Master of computer Applications in Loyola Institute of Technology and Management ,Sattenapalli. Persuing Ph.D in Acharya Nagarjuna University