UNN: A Neural Network for Uncertain Data Classification

Jiaqi Ge, Yuni Xia, and Chandima Nadungodage

Department of Computer and Information Science, Indiana University – Purdue University, Indianapolis, USA
{jiaqge,yxia,chewanad}@cs.iupui.edu

Abstract. This paper proposes a new neural network method for classifying uncertain data (UNN). Uncertainty is widely spread in real-world data. Numerous factors lead to data uncertainty including data acquisition device error, approximate measurement, sampling fault, transmission latency, data integration error and so on. The performance and quality of data mining results are largely dependent on whether data uncertainty are properly modeled and processed. In this paper, we focus on one commonly encountered type of data uncertainty - the exact data value is unavailable and we only know the probability distribution of the data. An intuitive method of handling this type of uncertainty is to represent the uncertain range by its expectation value, and then process it as certain data. This method, although simple and straightforward, may cause valuable information loss. In this paper, we extend the conventional neural networks classifier so that it can take not only certain data but also uncertain probability distribution as the input. We start with designing uncertain perceptron in linear classification, and analyze how neurons use the new activation function to process data distribution as inputs. We then illustrate how perceptron generates classification principles upon the knowledge learned from uncertain training data. We also construct a multilayer neural network as a general classifier, and propose an optimization technique to accelerate the training process. Experiment shows that UNN performs well even for highly uncertain data and it significantly outperformed the naïve neural network algorithm. Furthermore, the optimization approach we proposed can greatly improve the training efficiency.

Keywords: Uncertainty, classification, neural network.

1 Introduction

Data tends to be uncertain in many applications [1], [2], [3], [4], [5]. Uncertainty can originate from diverse sources such as data collection error, measurement precision limitation, data sampling error, obsolete source, network latency and transmission error. The error or uncertainty in data is commonly treated as a random variable with probability distribution. Thus, uncertain attribute value is often represented by an interval with a probability distribution function over the interval [6], [7]. It is important to cautiously handle the uncertainty in various data mining applications, as the

M. J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 449–460, 2010.
© Springer-Verlag Berlin Heidelberg 2010

data uncertainty is useful information which can be leveraged in order to improve the quality of the underlying results [17]. However, many traditional data mining problems become particularly challenging for the uncertain case. For example, in a classification application, the class which a data point belongs to may be changing as a result of the vibration of its uncertain attributes' values. Furthermore, the uncertainty over the whole dataset may blur the boundaries among different classes, which brings extra difficulties to classify uncertain datasets. Thus, data mining algorithms for classification, clustering and frequent pattern mining may need to integrate data uncertainty models to achieve satisfactory performance.

Classification is one of the key processes in machine learning and data mining. Classification is the process of building a model that can describe and predict the class label of data based on the feature vector [8]. An intuitive way of handling uncertainty in classification is to represent the uncertain value by its expectation value and treat it as a certain data. Thus, conventional classification algorithms can be directly applied. However, this approach does not effectively utilize important information such as probability function or distribution intervals. We extend data mining techniques so that they can take uncertain data such as data interval and probability distribution as the input. In this paper, we design and develop a new classifier named uncertain neural network (UNN), which employs new activation function in neurons to handle uncertain values. We also propose a new approach to improve the training efficiency of UNN. We prove through experiments that the new algorithm has satisfactory classification performance even when the training data is highly uncertain. Comparing with the traditional algorithm, the classification accuracy of UNN is significantly higher. Furthermore, with the new optimization method, the training efficiency can be largely improved.

The paper is organized as follows. In section 2, we discuss related work. Section 3 defines the classification problem for uncertain data. In section 4, we first analyze the principle of uncertain perceptron in linear classification, and then construct the multilayer uncertain neural network, and discuss the training approach. Section 5 introduces an optimized activation function to improve the efficiency. The experiments results are shown in section 6, and section 7 makes a conclusion for the paper.

2 Related Works

There has been a growing interest in uncertain data mining. A number of data mining algorithms have been extended to process uncertain dataset. For example, UK-Means [9], uncertain support vector machine [10], and uncertain decision tree [11]. The key idea in [10] is to provide a geometric algorithm which optimizes the probabilistic separation between the two classes on both sides of the boundary [7]. And [11] extends the decision tree to handle interval inputs and takes probability cardinality to select the best splitting attribute. However, both these two uncertain classifiers use a simple bounded uncertain model, and in our work, we use Gaussian noise instead to model the uncertainty, which is more common in realistic world. Artificial neural network has been used in model-based clustering with a probability gained from expectation-maximization algorithm for classification-likelihood learning [12]. We adopt the concept to estimate the probability of membership when the uncertain data

are covered by multiples classes. However, probability estimation presented here is unprecedented.

In fuzzy neural network models for classification, either attributes or class labels can be fuzzy and are presented in fuzzy terms [13]. Given a fuzzy attribute of a data tuple, a degree (called membership) is assigned to each possible class, showing the extent to which the tuple belongs to a particular class. Some other fuzzy systems [18] build reasoning mechanisms based on rules, and try to simulate the fuzzy cases inside the network. But they do not take the detailed uncertainty information as probability distribution into account in neuron level. Our work differs from previous work in that we revise the activation functions to compute the membership based on uncertain data distribution information, instead of using Fuzzy logic for tuning neural network training parameters. Our approach can work on both certain and uncertain data.

3 Problem Definition

In our model, a dataset D consists of d training tuples, $\{t_1,t_2,\ldots,t_d\}$, and k numerical attributes, $A_1,\ldots, A_k$. Each tuple t_i is associated with a feature vector $Vi = (f_{i,1}, f_{i,2}, \ldots, f_{i,k})$, and a class label $c_i \in C$. Here, each $f_{i,j}$ is a pdf modeling the uncertain value of attribute A_j in tuple t_i. Table. 1 shows an example of an uncertain dataset. The first attribute is uncertain. The exact value of this attribute is unavailable, and we only know the expectation and variance of each data tuple. This type of data uncertainty widely exists in practice [1], [2], [5], [6], [7].

Table. 1. An example of uncertain dataset

ID	Class Type	Attribute #1 (expectation, standard variance)
1	Yes	(105, 5)
2	NO	(110,10)
3	No	(70,10)
4	Yes	(120,18)
5	No	(105,10)
6	No	(60,20)
7	Yes	(210,20)
8	No	(90,10)
9	No	(85,5)
10	No	(120,15)

The classification problem is to construct a relationship M that maps each feature vector $(f_{x,1}, f_{x,2}, \ldots, f_{x,k})$ to the membership P_x on class label C, so that given a test tuple $t_0=(f_{0,1}, f_{0,2}, \ldots, f_{0,k})$, $M(f_{0,1}, f_{0,2}, \ldots, f_{0,k})$ predict the membership to each class. If the test instance has positive probability to be in different classes, then it will be predicted to be in the class which has the highest probability. The work in this paper is to build a neural network when only uncertain training data tuples are available, and the goal is to find the model with the highest accuracy despite of the uncertainty.

4 Algorithm

4.1 Uncertain Perceptron

We start with perceptron, which is a simple type of artificial neural network. Perceptron is a classical model which constructs linear classifier as:

$$y = F(\sum_{i=1}^{n} x_i \omega_i + \theta) \ , \tag{4.1}$$

$$F(s) = \begin{cases} 1, s \geq 0 \\ -1, s < 0 \end{cases} .$$

Where $x = (x_1,\cdots,x_n)$ is the input vector, $\omega = (\omega_1,\ldots,\omega_n)$ is the weight vector, F is the activation function, and y is the perceptron's output.

For data sets with uncertain attributes, we need revise the functions and develop an uncertain perceptron for linear classification. We will illustrate our approach through a simple 2-dimensional dataset. Assume dataset has two attributes $X = (x_1, x_2)$ and one class type y, and assume each uncertain attribute has a distribution as $x_i \sim N(\mu_i, \sigma_i)$, and the class type can be +1 or -1, Fig. 1 is a geometric representation of linear classification for a 2-dimensional uncertain dataset. In this figure, each data instance is represented by an area instead of a single point because each dimension/attribute is an uncertain distribution, not an accurate value.

The straight line L in Fig 1 represents the equation:

$$\omega_1 x_1 + \omega_2 x_2 + \theta = 0 \ . \tag{4.2}$$

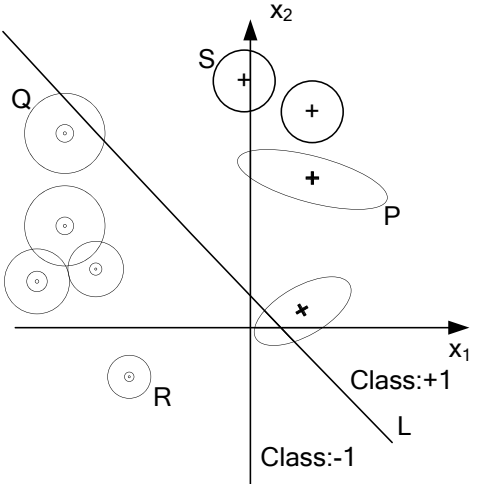


Fig. 1. Geometric representation of uncertain Perceptron

where x_1, x_2 are uncertain attributes. We define a parameter t as

$$t = \omega_1 x_1 + \omega_2 x_2 + \theta \ . \tag{4.3}$$

As mentioned earlier, attributes (x_1, x_2) follow the distribution $x_i \sim N(\mu_i, \sigma_i^2)$. Since these attributes are independent, t will have a distribution as:

$$f(t) \sim N(\omega_1\mu_1 + \omega_2\mu_2 + \theta,\ \omega_1^2\sigma_1^2 + \omega_2^2\sigma_2^2) \ . \tag{4.4}$$

Let $s = P(t>0)$ represent the probability of t larger than 0. If $P(t>0) = 1$, t is definitely larger than 0, which means this tuple is in class +1, and locates above the line L in Fig. 1, for example, like Point P. If $P(t>0) = 0$, t is less than or equal to 0, which means this tuple is in class -1, and it is below line L such as Point R. For uncertain data, it is possible that the uncertain range of a data instance may cover the linear classification line L, for example, Point Q is one such instance. In this case, Q has positive probability to belong to both classes, and the membership of class will be determined by which class has a high probability. Therefore, we construct an activation function as equation (4.5).

$$F(s) = \begin{cases} 1, s \geq 0.5 \\ -1, s < 0.5 \end{cases}. \tag{4.5}$$

Where, $s = P(t>0)$. Fig. 2 is structure of the uncertain perceptron model. In Fig.2, (μ_i, σ_i) is the expectation and standard deviation of uncertain attributes, as inputs. When the distribution is Gaussian, s can be calculated as:

$$s = \int_0^{\infty} \frac{1}{\sqrt{2\pi}\sigma_t} \exp(-\frac{(t-u_t)^2}{2\sigma_t^2}) dt \ . \tag{4.6}$$

Based on the single uncertain neurons, we can develop a multilayer neural network.

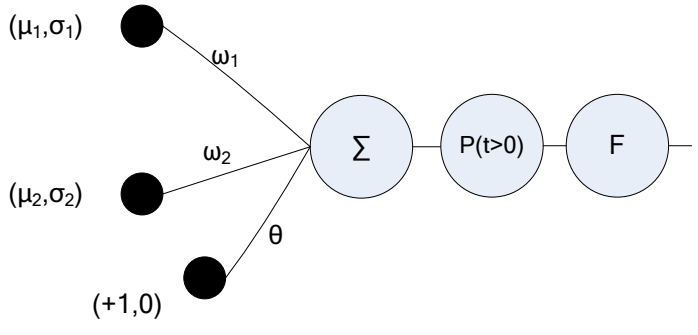


Fig. 2. Uncertain perceptron structure

4.2 Uncertain Neural Network

An uncertain multilayer feed-forward neural network is constructed by adding a hidden layer which contains the uncertain neurons between input and output layers. We call this algorithm as UNN (for uncertain neural network). Fig. 3 is an instance of the layer structure of neural network. Here, the hidden layer has a transfer function as

$$F(\mu,\sigma)=P(t>0)\,. \tag{4.7}$$

Where,

$$t=\sum_{i=1}^{2}\omega_i x_i+\theta\,.$$

$$x_i \sim N\ (\mu_i,\ \omega_i)\,.$$

P(t>0) will be computed based on uncertain data distribution function, For example, if the data follows Gaussian distribution, then

$$P(t>0)=\int_0^{\infty}\frac{1}{\sqrt{2\pi}\sigma_t}\exp(-\frac{(t-u_t)^2}{2\sigma_t^2})dt\ .$$

The output layer can have an activation function as Sigmoid, since the output values fall in the range (0,1), to represent the membership of every class.

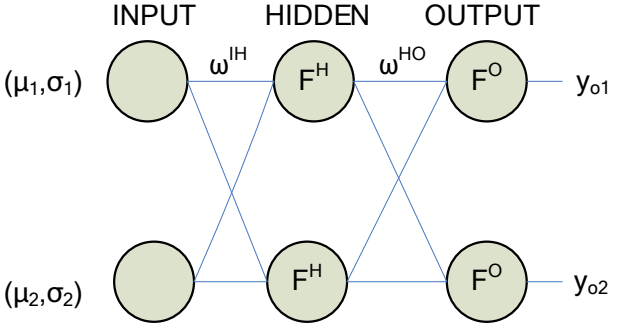


Fig. 3. Multilayer neural network structure

4.3 Algorithm Analysis

A straight-forward way to deal with the uncertain information is to replace the probability distribution function with its expected value. Then the uncertain data can be treated as certain data and the traditional neural network can be used for classification. We call this approach AVG (for Averaging). This approach, as mentioned earlier, does not utilize valuable uncertain information and may result in loss of accuracy. We illustrate the reason with the following example. Fig. 4 is an example of

classifying an uncertain dataset. Line *L*1 and *L*2 reflect the training result of the hidden layers of a neural network. Suppose *P* is a test data instance and we need predict the class type of P. Because the expectation of P locates in area II, it will be assigned to class II if using AVG algorithm. However, from Fig. 4, it is obvious that if we consider the distribution of *P*, it has a larger probability to be in area I than in area II. Therefore, it should be classified to class I. UNN will perform the classification correctly since it computes the probability of P belonging to both classes I and II according to the probability distribution information and predicts it to be in the class which has a larger probability. In this sense, the uncertain neural network can achieve higher classification accuracy.

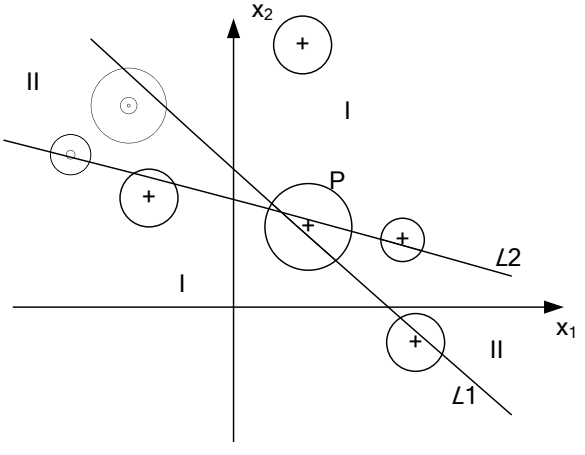


Fig. 4. Classifying a test tuple P

4.4 Network Training

We adopt a Levenberg-Marquardt back propagation algorithm [14], to train this supervised feed-forward neural network. It requires all the activation function has a derivative. Suppose Equation (4.7) is the hidden layer activation function of the uncertain neural network, then its derivative is like:

$$\frac{dF}{d(\mu,\sigma^2)} = (\frac{\partial F}{\partial \mu}, \frac{\partial F}{\partial \sigma^2}) = (\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\mu^2}{2\sigma}}, -\frac{\mu}{2\sqrt{2\pi}\sigma^3} e^{-\frac{\mu^2}{2\sigma}}) . \quad (4.8)$$

And,

$$\frac{d\mu_t}{d\omega_i} = \mu_i, \frac{d\sigma_t^2}{d\omega_i} = 2\sigma_i^2 * \omega_i . \quad (4.9)$$

Therefore, by substituting Equation (4.8) (4.9) into Equation (4.10), we can get the activation function's derivatives.

$$\frac{dF}{d\omega_i} = \frac{\partial F}{\partial \mu}\frac{d\mu}{d\omega_i} + \frac{\partial F}{\partial \sigma}\frac{d\sigma}{d\omega_i} \ . \tag{4.10}$$

When we have the derivatives of these activation functions, it is intuitive to train the network based on traditional method such as gradient decent. After training, we can then use the model for prediction for uncertain data.

5 Improve on Activate Function

The hidden layer's activate function, in Equation (4.7), has an output ranging between 0 and 1. When we consider two different data instances that are absolutely in the same class, their function output will both be 1. This may cause the network training to be time consuming in some scenarios. In order to improve the training efficiency, we can design new hidden layer activate functions. For example, when the uncertainty is represent by Gaussian distribution, we devise a new hidden layer activate function, as Equation (5.1) to accelerate the training process.

$$F_2(\mu,\sigma) = \begin{cases} u_t * P(t>0), & \text{if } u_t > 0; \\ 0, & \text{if } u_t = 0; \\ u_t * P(t<0), & \text{if } u_t < 0; \end{cases} \tag{5.1}$$

$$f(t) \sim N\left(\sum_i^I \omega_i \mu_i + \theta, \ \sum_i^I \omega_i^2 \sigma_i^2\right).$$

Here $F_2(\mu, \sigma)$ is continuous at $u_t = 0$, since

$$\lim_{\mu \to 0+} F_2(\mu,\sigma) = \lim_{\mu \to 0-} F_2(\mu,\sigma) = F_2(0,\sigma) = 0 \ .$$

$F_2(\mu, \sigma)$ also has a derivative:

$$\frac{dF_2}{d(\mu,\sigma^2)} = \left(\frac{\partial F_2}{\partial \mu}, \frac{\partial F_2}{\partial \sigma^2}\right) . \tag{5.2}$$

$$\frac{\partial F_2}{\partial \mu} = \begin{cases} \eta + \dfrac{\mu}{\sqrt{2\pi}\sigma} e^{-\frac{\mu^2}{2\sigma}}, if \ \ \mu_t > 0 \\ 1/2, \qquad\qquad if \ \ \mu_t = 0 \\ 1\text{-}\eta - \dfrac{\mu}{\sqrt{2\pi}\sigma} e^{-\frac{\mu^2}{2\sigma}}, if \ \ \mu_t < 0 \end{cases} . \tag{5.3}$$

$$\frac{\partial F_2}{\partial \sigma^2} = -\frac{\mu^2}{2\sqrt{2\pi}\sigma^3} e^{-\frac{\mu^2}{2\sigma}} \ . \tag{5.4}$$

Thus, substitute Equation (5.2) (5.3) (5.4) into Equation (5.5), we get the derivative of F_2.

$$\frac{dF_2}{d\omega_i} = \frac{\partial F_2}{\partial \mu}\frac{d\mu}{d\omega_i} + \frac{\partial F_2}{\partial \sigma}\frac{d\sigma}{d\omega_i} \tag{5.5}$$

Equation (5.5) then can be used in Levenberg-Marquardt back propagation training algorithm.

6 Experiments

6.1 Experiment on Accuracy

We have implemented the UNN approach using Matlab6.5[15], and applied them to 5 real data sets taken from the UCI Machine Learning Repository [16]. The results are shown in Table. 2. For the datasets except "Japanese Vowel", the data uncertainty is modeled with a Gaussian distribution with a controllable parameter ω, which is a percentage of the standard deviation to the value of expectation. In our experiments, we vary the ω value to be 0.1, 0.3 and 0.5. For "Japanese Vowel" data set, we use the uncertainty given by the original data to estimate its Gaussian distribution.

Table 2. Accuracy experiment results

Japanese Vowel	Uncertainty	Train	Test
UNN	Distribution based raw data	98.50%	94.95%
AVG		99.17%	94.31%

Iris	Uncertainty	Train	Test
UNN	ω=0.1	98.05%	99.93%
	ω=0.2	98.33%	99.93%
	ω=0.5	97.78%	99.38%
AVG		99.17%	98.89%

Ionosphere	Uncertainty	Train	Test
UNN	ω=0.1	92.75%	93.71%
	ω=0.2	94.50%	90.73%
	ω=0.5	99.13%	92.05%
AVG		97.17%	87.86%

Magic Telescope	Uncertainty	Train	Test
UNN	ω=0.1	96.93%	80.01%
	ω=0.2	97.50%	76.58%
	ω=0.5	97.50%	80.56%
AVG		99.67%	73.17%

Glass	Uncertainty	Train	Test
UNN	ω=0.1	77.05%	65.75%
	ω=0.2	76.00%	69.59%
	ω=0.5	79.02%	65.57%
AVG		74.02%	65.22%

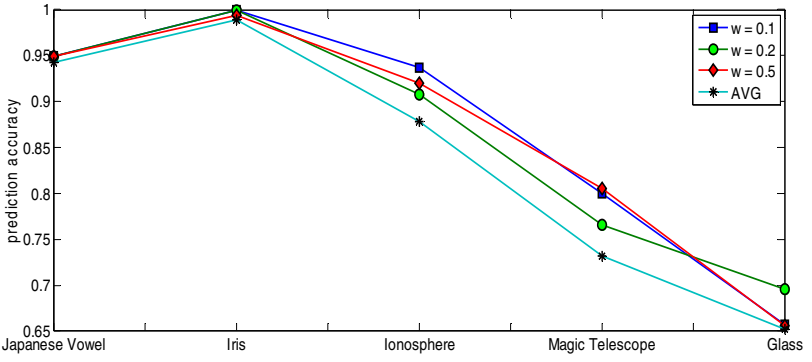
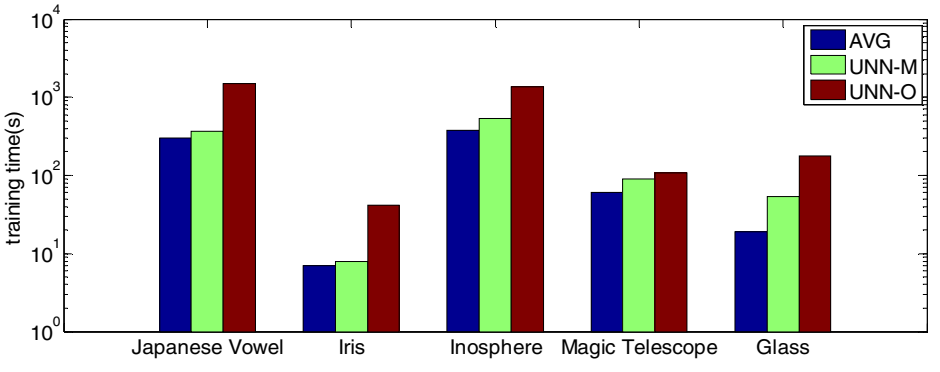
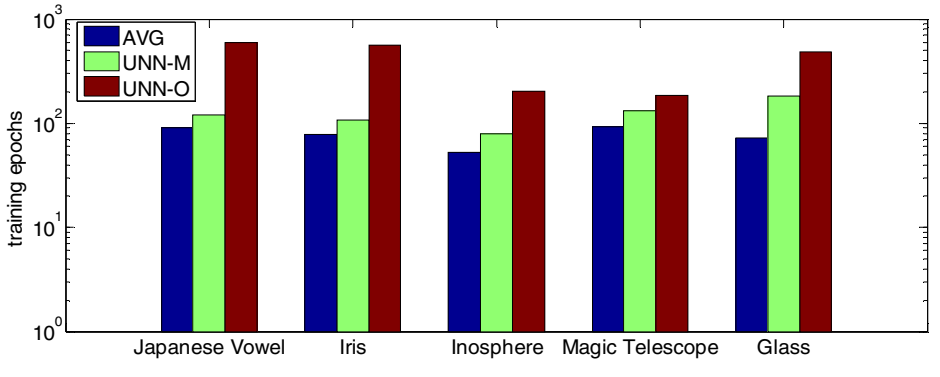


Fig. 5. Accuracy Comparison of UNN and AVG



(a)Training time



(b) Training epochs

Fig. 6. Performance comparison

In our experiments, we compare UNN with the AVG (Averaging) approach, which process uncertain data by simply using the expected value. The results are shown in Fig 5. From the figure, we can see that UNN outperforms AVG in accuracy almost all the time. For some datasets, for example, Ionosphere and Magic Telescope datasets, UNN improves the classification accuracy by over 6% to 7%. The reason is that UNN utilizes the uncertain data distribution information and computes the probability of data being in all different classes. Therefore, the classification and prediction process is more sophisticated and comprehensive than AVG, and has the potential to achieve higher accuracy.

6.2 Experiment on Efficiency

In section 5, we have discussed an alternative activate function for improving the efficiency of network training process. Here, we present an experiment which compares the efficiency of two networks with different hidden layer activate functions. In this experiment, we name the network using the original function (Equation 4.7) as UNN-O, and the network using activate function (5.1) as UNN-M.

The training time of UNN-O and UNN-M is shown in Fig. 6 (a) and the training epochs of UNN-O and UNN-M is shown in Fig 6. (b). Because of the more complex calculations in handling uncertainty, UNNs generally require more training time and epochs than AVG. However, the figures also indicate that efficiency of UNN-M is highly improved, compared with UNN-O. The training of UNN-M requires much fewer epochs than UNN-O, and is significantly faster.

7 Conclusion

In this paper, we propose a new neural network (UNN) model for classifying and predicting uncertain data. We employ the probability distribution which represent the uncertain data attribute, and redesign the neural network functions so that they can directly work on uncertain data distributions. Experiments show that UNN has higher classification accuracy than the traditional approach. The usage of probability distribution can increases the computational complexity, and we propose new activation function for improved efficiency. 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., Yu, P.: A framework for clustering uncertain data streams. In: IEEE International Conference on Data Engineering, ICDE (2008)
2. Cormode, G., McGregor, A.: Approximation algorithms for clustering uncertain data. In: Principle of Data base System, PODS (2008)
3. 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)
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.: Hierarchical density-based clustering of uncertain data. In: IEEE International Conference on Data Mining (ICDM), pp. 689–692 (2005)
6. Aggarwal, C.C.: On Density Based Transforms for uncertain Data Mining. In: IEEE International Conference on Data Engineering, ICDE (2007)
7. Aggarwal, C.C.: A Survey of Uncertain Data Algorithms and Applications. IEEE Transactions on Knowledge and Data Engineering 21(5) (2009)
8. Agrawal, R., Imielinski, T., Swami, A.N.: Database mining: A performance perspective. IEEE Transactions on Knowledge& Data Engineering (1993)
9. Chau, M., Cheng, R., Kao, B., Ng, J.: Uncertain data mining: An example in clustering location data. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 199–204. Springer, Heidelberg (2006)
10. Bi, J., Zhang, T.: Support Vector Machines with Input Data Uncertainty. In: Proceeding of Advances in Neural Information Processing Systems (2004)
11. Qin, B., Xia, Y., Li, F.: DTU: A Decision Tree for Classifying Uncertain Data. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 4–15. Springer, Heidelberg (2009)
12. Cheng, S.-S., Fu, H.-C., Wang, H.-M.: Model-Based Clustering by Probabilistic Self-Organizing Maps. IEEE Transactions on Neural Networks 20(5) (2009)
13. Kulkarni, A.D., Muniganti, V.K.: Fuzzy Neural Network Models For Clustering. In: ACM Symposium on Applied Computing (1996)
14. Stan, O., Kamen, E.W.: New block recursive MLP training algorithms using the Levenberg-Marquardt algorithm. Neural Networks 3 (1999)
15. Matlab: `http://www.mathworks.com/`
16. Asuncion, A., Newman, D.: UCI machine learning repository (2007), `http://www.ics.uci.edu/mlearn/MLRepository.html`
17. Aggarwal, C.C.: Managing and Mining Uncertain Data. Springer, Heidelberg (2009)
18. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE transactions on systems, man, and cybernetics 23, 665–685 (1993)