Neural Learning from Unbalanced Data: Special Issue: Engineering Intelligent Systems (Guest Editor: László Monostori)

Article *in* Applied Intelligence · September 2004

DOI: 10.1023/B:APIN.0000033632.42843.17 · Source: DBLP

Yi Murphey
University of Michigan-Dearborn
109 PUBLICATIONS **1,586** CITATIONS

Lee Feldkamp
Ford Research and Engineering,Dearborn, USA
125 PUBLICATIONS **9,722** CITATIONS

Applied Intelligence 21, 117–128, 2004

Neural Learning from Unbalanced Data

YI L. MURPHEY AND HONG GUO
Department of Electrical and Computer Engineering, The University of Michigan-Dearborn, Dearborn Michigan 48128-1491, USA
yilu@umich.edu

LEE A. FELDKAMP
Research and Advanced Engineering, Ford Motor Company, Dearborn, MI 48121, USA

Abstract. This paper describes the result of our study on neural learning to solve the classification problems in which data is unbalanced and noisy. We conducted the study on three different neural network architectures, multi-layered Back Propagation, Radial Basis Function, and Fuzzy ARTMAP using three different training methods, duplicating minority class examples, Snowball technique and multidimensional Gaussian modeling of data noise. Three major issues are addressed: neural learning from unbalanced data examples, neural learning from noisy data, and making intentional biased decisions. We argue that by properly generated extra training data examples around the noise densities, we can train a neural network that has a stronger capability of generalization and better control of the classification error of the trained neural network. In particular, we focus on problems that require a neural network to make favorable classification to a particular class such as classifying normal(pass)/abnormal(fail) vehicles in an assembly plant. In addition, we present three methods that quantitatively measure the noise level of a given data set. All experiments were conducted using data examples downloaded directly from test sites of an automobile assembly plant. The experimental results showed that the proposed multidimensional Gaussian noise modeling algorithm was very effective in generating extra data examples that can be used to train a neural network to make favorable decisions for the minority class and to have increased generalization capability.

Keywords: machine learning, neural networks, unbalanced data, data noise

1. Introduction

Neural networks have been applied to various problems including engineering diagnosis, pattern classification, intelligent manufacturing and control problems [1–3]. There has been much progress in developing methods for training complex configurations of these networks, but little has been known about the general learning properties of neural networks [4]. Our research is focused on the following three major issues within the problem scope of pattern classification: neural learning from unbalanced data examples, neural learning from noisy data, and making intentional biased decisions.

In many application problems, the training data for each class is extremely unbalanced. One example is the classification of defective products at the end of manufacturing lines such as automobile assembly plants. One thing in common in a manufacturing environment is that most products are good and only a few are defects. If we further divide the defect products into classes of different defect types, we will have far more data examples from the "normal" class than those in any one of the defective classes for neural learning [1].

This problem has been referred to as classification under unbalanced training data. If the training methods are not proper, the features representing the classes that have small number of examples in the training set may likely be ignored by the neural networks. This problem is caused by the overwhelming number of learning examples in one class input to the learning system that partially undo the training effect on the small number of learning examples of a different class. As we will show, this problem is more serious when the data set has a high level of noise. Data noise in classification problems can be generally described as data examples of different classes inseparable in the feature space. In other words, if a data set is considered noisy, the class boundary to separate different class examples in the feature space is almost impossible to draw. Noise in training and test data rises from a number of sources, the set of features used for classification is not sufficient to draw class boundaries, data examples are mislabeled, poor data acquisition processes, etc. These problems are inevitable in many engineering applications.

Another important characteristic in many applications is the control of misclassification. In other words, if misclassification is inevitable, we need to train the neural network to make favorable classification decisions towards a particular class. For example, in the End-of-Line test at automotive assembly plants, we would rather have a classification system misclassify a good car as bad instead of a bad car being identified as a good car. The reason is that cars being identified as bad will be checked by mechanics for repair, so a truly good car can be returned to the pool.

In this paper, we present our study on the behavior of three neural network architectures, multi-layered Back Propagation (BP) network, Radial Basis Function (RBF), and Fuzzy Adaptive Resonance Theory (ART) network, in response to unbalanced data with different levels. We present three different methods to quantitatively describe noise levels in a training data set, addressing three major issues: neural learning from unbalanced data examples, neural learning from noise data, and making intentional biased decisions. We propose a noise modeling algorithm that uses the multidimensional Gaussian distribution to analyze the separation of difference class examples in a training data set and generate extra training data examples with an aim of training a more robust neural network that is less susceptible to noise, has a capability of making more favorable classification decision to a particular class and more generalization capabilities. Our approach is based on the following hypothesis: in supervised learning, the noise model or distribution in the unknown test data set is not grossly different from the training data and the ability of generalization of neural networks is very much dependent on the data noise along the class boundaries.

2. Behavior of Neural Networks Trained on Unbalanced Data

This section presents our research on the behavior of neural networks on unbalanced data. We will first introduce three different neural network architectures used throughout this paper, and then present methods to measure data noise levels. At the end we will show that the three neural networks behave quite differently in response to the unbalanced training data sets with different noise levels.

2.1. Three Neural Network Architectures under Study

We study the behavior of three neural network architectures, multi-layered BP network, RBF and Fuzzy ARTMAP, on noisy data with unbalanced class distributions. BP network is the most popular neural network in pattern recognition and classification. It has been shown that a BP network with one hidden layer can approximate any L_2 function $f : [0, 1]^n \rightarrow R^m$ in any square error ε, and a multiple hidden layer network can approximate any function [5]. In order to make sure that the complexity of the input space and output space is well represented, we chose to use a network of three layers, with two hidden layers, and the well-known back propagation (BP) learning algorithm [6].

A Radial Basis Function (RBF) network has a feedforward architecture that has an internal representation of hidden processing elements that are radially symmetric [7]. The response of the hidden layer units in a RBF network is localized and decreases as a function of the distance of inputs from the unit's receptive field center. The RBF network used in this study is illustrated in Fig. 1. The RBF network has three major components:

- The weight vector from input layer to the pattern unit is defined as the center of the radially symmetric transfer function.
- A Euclidean distance measure is defined to determine how far an input vector is from the center.
- The Gaussian function is used as a transfer function that maps the output of the distance function to the output of a process element.

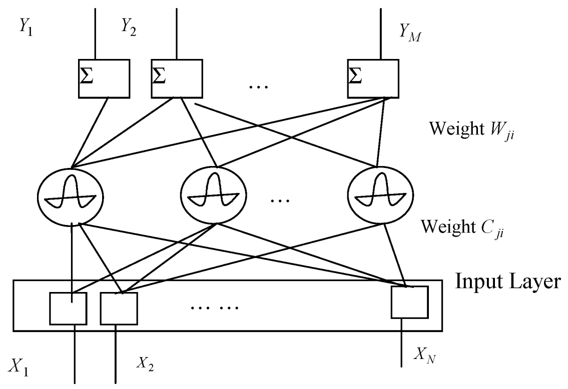

Figure 1. An RBF network.

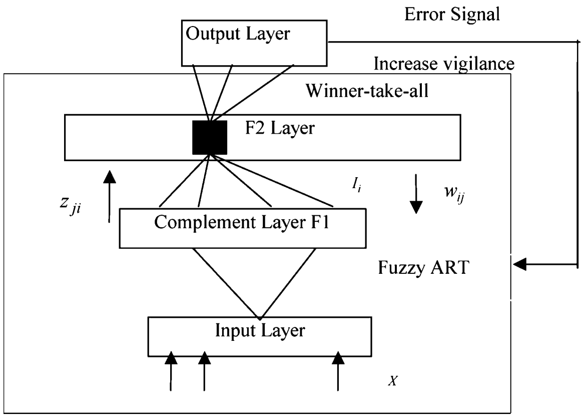

Figure 2. A Fuzzy ARTMAP network.

Formally, the transfer function for ith hidden unit is:

$$V_i = \exp\left[-\frac{(X - C_i)^T (X - C_i)}{2\sigma_i^2}\right]$$

where X and C_i is the input and the weight vector to the ith hidden unit respectively, $i = 1, \ldots, K$. The weight vector C_i and the width σ_i of the Gaussian functions are determined during the neural learning. The learning algorithm employed in this study uses the standard K-means clustering to determine the centers C_i, and then uses the nearest neighbor heuristic to determine σ_i. The K-means clustering algorithm finds a set of K cluster centers $C_1, C_2, \ldots, C_K$ such that the sum of the squares of the distances between each training vector X and its closest cluster center is a local minimum. A cluster center $C_i = (C_{1i}, \ldots, C_{\mathrm{Ni}})$ is used as a weight vector from the input layer to the ith hidden node which is shown in Fig. 1.

We used a P nearest neighbors heuristic to determine the σ_i of the transfer function. Given a cluster center C_i, let $i_1, \ldots, i_p$ be the indices of the P nearest neighboring cluster centers, the width of the transfer function:

$$\sigma_i = \sqrt{\frac{1}{P}\sum_{j=1}^{P} \|C_i - C_{i_j}\|^2}.$$

When the self-organization (with no desired output is used) is complete, the output layer is trained using the standard delta rule learning which adaptively minimizes the mean square error between desired output vectors and actual output vectors to obtain the weights W_{ji} between ith node in the hidden layer and the jth node in the output layer.

Fuzzy ARTMAP is a supervised version of Fuzzy ART [3] which in turn is the Fuzzified network of Adaptive Resonance Theory (ART) [8–10]. Figure 2 shows the Fuzzy ARTMAP network used in this study, which is a supervised version of Fuzzy ART network with an additional single output layer. The Fuzzy ARTMAP network consists of two interconnected layers of neurons, F1 and F2. The input leads to the activity in the feature detector neurons in F1, which is also called short-term memory activity. The short-term activity passes through the connections to the neurons in F2. Each F2 neuron adds together its input from all the F1 neurons and generates an output. A measure is taken over all the neurons in F2 and the output of one neuron will be selected as the system output, i.e. Winner-take-all. The network allows a top-down feedback from F2 to reinforce the activity in F1 to represent a template or set of critical features in a classification category. During recall and categorization, the exchange of bottom-up and top-down information leads to a resonance in neural activity.

The learning algorithm used in this study is described as follows. The input nodes of a Fuzzy ART network are considered as a set of features. If a node is active, the corresponding feature is present; otherwise the feature is absent. Let $\bar{I} = (I_1, I_2, \ldots I_M)$ be the input vector that can take on any values between 0 and 1 to represent the membership function value in the fuzzy set theory. Let w_{ij} be an element in matrix W, which represents the top-down weight from winning node j in the F2 layer to a node i in the F1 layer, and z_{ji} be the corresponding bottom-up weight in matrix Z, n be the dimension of input vector, and M be the number of nodes in F2. For every input $\bar{I} = (I_1, I_2, \ldots, I_M)$ presented to the input nodes, we compute:

$$y_j = \sum_i z_{ji} I_i$$

for $j = 1, 2, \ldots, M$. Assuming the node m has the maximum value of y, i.e. $y_m = \max_{j=1,2,\ldots M} y_j$, the similarity test is:

$$\frac{1}{\|\bar{I}\|} \sum_{i=1}^{M} w_{im} I_i > \rho$$

where ρ is a vigilance parameter and the norm $\|\bar{I}\|$ is defined as $\|\bar{I}\| = \sum_{i=1}^{M} |I_i|$. If the similarity test is true, the weights w_{im} and z_{mi}, $(i = 1, 2, \ldots M)$ are updated as follows:

$$z_{mi}(t+1) = \frac{w_{im}(t) I_i}{0.5 + \sum_{i=1}^{M} w_{im}(t) I_i}$$
$$w_{im}(t+1) = \beta w_{im}(t) I_i + (1-\beta) w_{im}(t),$$

where β controls the learning rate. The weights can be initially set as $w_{im} = I_i$ for $i = 1, \ldots, M$. After every weight updating, the learning algorithms takes a new input vector to the network.

If the similarity test does not hold and the F2 layer has more than a single active node, the node m is deactivated by setting y_m to zero to prevent this node from participating in the current cluster search, and then the algorithm finds a new winner among y_j to repeat the similarity test and weight updating procedure. If the similarity test fails and F2 has only one active node, the weight z_{ji} is updated using the formula above and $w_{im} = I_i$. One problem with this weight updating rule is that $|\bar{W}_m| = \sum_{i=1}^{M} |w_{im}|$ decreases each time node m is selected for learning because I_i less than 1. This will lead to a proliferation of categories since eventually the training vector that originally established the category will often no longer match the pattern stored in weights.

The difference between the desired and actual output is used as an error signal being sent to Fuzzy ART to try a different F2 layer node other than the previous one. This is implemented by changing ρ, the current value of vigilance. If the difference between the desired output and the actual output is larger than error tolerance ε, ρ increases until $\rho > |\bar{I} \cdot \bar{W}_m \| \bar{I}|^{-1}$. When this happens, Fuzzy ART searches for another F2 layer node J such that $\rho > |\bar{I} \cdot \bar{W}_j \| \bar{I}|^{-1}$. If the difference between the desired and actual output for this F2 node J is less than ε, the learning of Fuzzy ART is performed as described above.

2.2. Data Noise Analysis

Data noise in classification problems can be generally described as data examples of different classes inseparable in the feature space. If a data set is considered noisy, the class boundary to separate different class examples in the hyper feature space is difficult to draw. In practical application problems noise in training and test data arises from a number of sources; the set of features used for classification is not sufficient to draw class boundaries, data examples are mislabeled, poor data acquisition processes, dirty sensors, etc. These problems are inevitable in many engineering applications. The challenge is that for a given set of noisy data, how can we train a neural network to learn to correctly classify data examples of different classes in the training data and have the capability of generalizing to unknown data? In order to find answers to this question, we need to define measures of data noise levels over data sets.

Data noise level is traditionally being vaguely defined. The objective of our research is to derive quantitative measures of noise level for a given set of data. We introduce three different approaches in measuring noise levels, inter- and intra-class distances, inter-class distances using Mahalanobis function, and mixed examples in hyper bounding boxes.

Measuring Noise Levels Using Intra and Inter Class Distances. For a given data set X, we take two measures for each data example, x_i in X, d_i^s and d_i^o, where d_i^s is the nearest distance from x_i to the examples of the same class, and d_i^o is the nearest distance from x_i to the examples of the opposite class. In another words d_i^s measures the intra-class distance and d_i^o measures the inter-class distance with respect to data example x_i. If $d_i^s > d_i^o$, data example x_i is closer to its opposite classes than to its own class, which implies that x_i is very likely to be misclassified. The quantitative measure of noise level of data set X is formally defined by $\frac{M}{N}$, where $N = \|X\|$, M is the number of all data examples, x_α, in X that has $d_\alpha^s > d_\alpha^o$. Generally, the larger the ratio $\frac{M}{N}$ is, the noisier the data set X is.

Measuring Noise Levels using Linear Separability between two Classes. For a 2-class problem, we first attempt to find if there exists a discriminant function that is a linear composite of the input vector components that can separate the data examples in the feature space. We assume that all of the input components are independent and have a multivariate normal distribution. Let C_1 and C_2 denote the data examples of X belonging in class 1 and 2 respectively.

Suppose $\bar{x}_i$, $i = 1, 2$, are the mean vector of the data examples in class C_i. The pooled example variance-covariance matrix S of the input vectors is

$$S = \frac{1}{n_1 + n_2 - 1}\left(X_1^T X_1 + X_2^T X_2\right)$$

where n_1 and n_2 are the number of examples of C_1 and C_2 respectively, X_i, $i = 1, 2$ is a matrix each row of which is a vector coming from class C_i.

The following function Z measures the statistical between-class distance and is developed based on linear discriminant function [11]:

$$Z = \frac{n_1 n_2}{n_1 + n_2} \frac{n_1 + n_2 - p - 1}{(n_1 + n_2 - 2)p} D^2$$

where p is the dimension of the input vector, and D is the Mahalanobis distance between two class centroid defined as

$$D^2 = (\bar{x}_1 - \bar{x}_2)^T S^{-1} (\bar{x}_1 - \bar{x}_2)$$

The noise level of X can be measured by the value of Z: a larger Z indicates a larger distance between the two classes, therefore X has less noise.

Measuring Noise Level Using Hyper Bounding Boxes in Feature Space. Let X be a class of data examples with dimension n, and $X = \{X_1, X_2, \ldots, X_m\}$, $X_i = \langle a_{i1}, a_{i2}, \ldots, a_{\text{in}} \rangle$, where a_{ij} is the jth feature of X_i. The hyper bounding box HBB_i of class CL_i is an n-tuple, $\text{HBB}^i = \langle x_1^i, x_2^i, \ldots, x_n^i \rangle$, such that

$$\begin{aligned} x_j^i &= \left(\min\_x_j^i, \max\_x_j^i\right) \\ &= \Big(\min_{1 \le k \le m} \{a_{kj} \mid X_k \in X \cap CL_i\}, \\ &\qquad \max_{1 \le k \le m} \{a_{kj} \mid X_k \in X \cap CL_i\}\Big). \end{aligned}$$

Two class examples are well separated if HBB_i contains no data examples of class $1 - i$, for $i = 0, 1$. The noise level of X can be measured by the number of data examples of class $1 - i$ inside HBB_i, for $i = 0, 1$. In another words, the number of examples of one class contained in the hyper bounding box of another class can be used as a measure of class separation. Mathematically, we define

$$\Psi = \frac{\sum_{i=0,1} \left|\left\{(X_j \in CL_{1-i}) \text{ and } \left(a_{j1} \in x_1^i\right) \text{ and} \ldots \text{and}\left(a_{jn} \in x_n^i\right) \mid 1 \le j \le m\right\}\right|}{|X|}.$$

Ψ ranges from 0 to 1. When $\Psi = 0$, the two classes are well separated by their hyper bounding boxes, which indicates the noise level of X is minimum. When $\Psi = 1$, the data examples of two classes in X are completely enclosed in each other's hyper bounding box, which indicates the highest noise level of X. We use Fig. 3 to illustrates this method. The data examples in this figure are in a two dimensional feature space. There are two classes of data examples, class 0 is illustrated in a shape of triangles, and class 1 in circles. We have $\text{HBB}_0 = \{(5, 25), (11, 39)\}$, $\text{HBB}_1 = \{(17, 33), (5, 25)\}$, $|X| = 26$, $\Psi = \frac{7}{26}$.

Summary of Noise Level Measurement Methods. These three methods use different characteristics of data set in measuring noise level. The first method measures the noise level based on the Euclidian distance

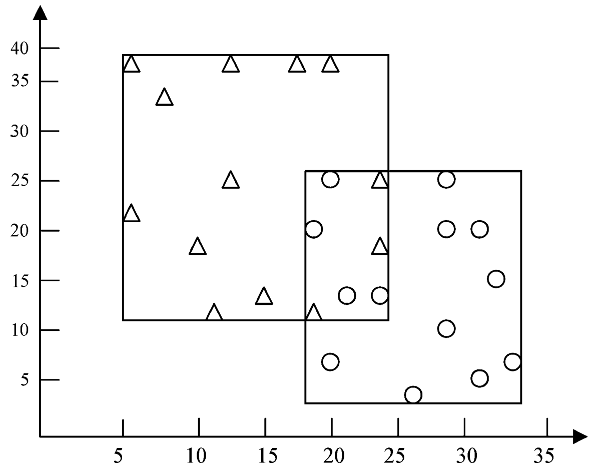

Figure 3. Illustration of noise level measured by HBB.

between the opposite classes and within the each class. The second method measures the linear separability of the data set. The third method measures the number of opposite class data samples falling within the bounding box of each class in the feature space. The third method is the easiest to implement, though it may not be as accurate as the other two.

2.3. Behavior of Neural Networks on Unbalanced Data

This section analyzes the behavior of the three neural networks presented in Section 2.1 through experiments performed on three data collections collected from three different vehicle models at a test site in an automobile assembly plant. The neural networks are trained to classify whether a given vehicle is "good" or "bad" using input vectors in five dimensions. Since new vehicles manufactured by an auto company are mostly in the "good" class, the data examples in both the training set and test set are unevenly distributed. Through out this paper we refer to these data collections as vehicle Model I, II, and III to protect the proprietary information. The distribution of the data collections are illustrated in the following table.

Table 1 shows that all three data collections are extremely unbalanced, the number of normal vehicle data examples is overwhelmingly larger than abnormal vehicles. Table 2 shows the noise level of three data collections measured by the three methods presented in Section 2.2. All three methods show that Model III has a very high level of noise. The HBB method indicates that Model I and Model II have noise level 0. The intra and inter class distance measure indicates that Modle I has a lower noise level than Model II, but the linear separability measure indicates that Model II has lower noise level than Model I. In summary all three methods indicate that Model I and Model II have low noise level and Model III has very high level of noise.

In our experiment, the parameters associated with each of the three neural networks are set as follows:

BP network: the layers are 5-4-4-1, Momentum $= 0.90$, Tanh function, epoch size $= 16$

RBF: the layers are 5-100-10, P $= 2$, Momentum $= 0.90$

Fuzzy ARTMAP: the initial vigilance is set to 0, error tolerance $\varepsilon = 0.01$, $\beta = 0.5$, F2 layer node number is 100, the dimension of input vector is 5.

Table 3 shows the performance of the three neural networks on the three data collections. One common approach in evaluating system behavior in the pattern recognition community is to divide the available data into two sets, a training set and a test set. Normally, a fixed percentage of data instances is used for training and the remainder for testing. According to Weiss and Kulikowski, the usual proportions of training and test data examples are approximately a 2:1 split [12, 13], and a neural network is evaluated based on its performance on the test set. However, the estimate on test data only is a relatively pessimistic estimate of the true error rate when the identical classification method is applied to all the cases. Therefore we also evaluate the system performance over the entire data set after it is trained over the 2/3 of a data collection.

Table 1. Class distribution of three data collections.

Data collections	Number of normal vehicle data	Number of abnormal vehicles data
Model I data	3960	9
Model II data	8015	15
Model III data	1083	201

Table 2. Data noise measure on the data collections used in experiments.

Data collections	*M/N*	*Z*	Ψ	Noise level
Model I data	0.04%	20.46	0	Very low
Model II data	0.1%	44.54	0	Very low
Model III data	23.64%	5.86	0.44	High

Table 3. Classification accuracy of the three neural networks on the three data collections.

	BP network majority class/ minority class	RBF network majority class/ minority class	Fuzzy ARTMAP majority class/ minority class
Model I data			
Entire set	99.87%/100%	99.95%/0%	100%/100%
Test set	99.97%/88.89%	99.71%/0%	99.81%/100%
Model II data			
Entire set	100%/100%	100%/100%	100%/100%
Test set	99.98%/86.67%	99.97%/6.67%	100%/100%
Model III data			
Entire set	99.63%/9.86%	100%/9.86%	74.24%/71.83%
Test set	98.89%/0%	91.97%/1.35%	76.45%/20.27%

The first finding from the study is that if the data between the two classes are well separated such as Model I and Model II, both the BP network and FUZZY ARTMAP give excellent performance on unbalanced data. The RBF network does not learn the features of the minority class well.

The second finding is that when data is noisy, none of the three networks performed well if a simple training method is used. All three classifiers are poor in performance when the data set has noise, e.g. Model III data (see Table 3). In particular, BP and RBF are weak in learning features of the minority class. Both networks perform better on the majority class, which is the good example class, than the minority class, which is the bad example class. Although Fuzzy ARTMAP was a little better than the other two, it still ignored the minority class on the test set. This experiment suggests that in the case of unbalanced, noisy training data examples, multi-layered backpropagation network (BP), Radial-Basis Function (RBF) and FUZZY ARTMAP all ignore the minority class, namely minority data examples are often misclassified. This motivated us to find a different neural training method that can control the classification decision over unbalanced and noise data.

3. Neural Learning from Unbalanced Data Using Gaussian Noise Estimation and Modeling

We developed an algorithm with the aim of training a neural network that can learn minority class features and have better generalization. The algorithm was based on the estimation of local densities and distance between two different classes in a given training data set. For every data example in the minority class, we attempt to construct the constant potential surface (CPF) of the Gaussian function which is an ellipsoid in a multi-dimension space.

3.1. Noise Modeling Using Gaussian CPF

Mathematically the problem of data noise around a given data s_i in a given class can be described as follows. If we assume data noise is in a Gaussian distribution, for data example s_i the data examples of its opposite class distributed around s_i can be modeled as random vectors Z that have the density function:

$$f_z(Z \mid s_i) = (2\pi)^{\frac{-M}{2}} |\Sigma_i|^{-\frac{1}{2}} \exp\left(-\frac{(Z - s_i)^T (Z - s_i)}{2|\Sigma_i|}\right)$$

where M is the dimension of each data example and Σ is the covariance matrix of s^i and its M nearest neighbors. We can further decompose Σ as $\Sigma = Q\Lambda Q^T$, where the diagonal entries of Λ and the column vectors of Q are the eigenvalues and eigenvectors of Σ respectively. Since Σ is symmetric, all eigenvalues and eigenvectors are real.

According to [14], if covariance matrix Σ is positive definite and symmetric, there exists a unique lower triangular matrix C such that $\Sigma = CC^T$. The random vector Z can be represented as:

$$Z = CY + s_i$$

where $Y = (y_1, y_2, \ldots, y_M)$ is a random vector generated by a Gaussian function with zero mean and identity covariance matrix.

The jth column vector of Q, q_j, $j = 1, 2, \ldots, M$, is computed recursively using the following formula:

$$q_j = \frac{b_j}{\|b_j\|}, \quad \text{and} \quad b_j = (a_j - s_i) - \sum_{k=1}^{j-1} b_k^T (a_j - s_i) b_k.$$

The initial vector is computed using $b_1 = a_1 - s_i$, where a_1 is the nearest neighbor of the opposite class of s_i. The successive column vectors can be computed recursively using the formula above. The eigenvalues of Λ can be obtained by $\lambda_j = \frac{1}{4r^2}\|b_j\|^2$, for $j = 1, 2, \ldots, M$, where r is the radius of a hypersphere that the probability of the hypersphere enclose the local density is v. Musavi et al. showed that for a given v, there is a fixed function relation between r and M, and the values of r for $M = 1, \ldots, 10$ can be found in [15].

The eigenvectors of Q are the principal axes of the ellipsoid of the constant potential surface (CPS) of the Gaussian distribution function, and the square roots of the eigenvalues define the lengths of the principal axes of the ellipsoid. The M nearest neighbors of the opposite class are all on the surface of the hyper-rectangle around s_i. We use Fig. 4 to illustrate the CPS and its relationship to class boundary in the 2D space. In the figure, A is a example data from one class, B and C are its two nearest neighbors of the opposite class. B and C are on the boundary of larger rectangular bounded by $2\|b_j\|$, for $j = 1, 2$. The classification boundary between the two classes can be drawn by the smaller rectangle, which is bounded by $\|b_j\|$ and encloses the CPS ellipse. The new random vectors Z_j shown in green "x"'s were generated using the formula above.

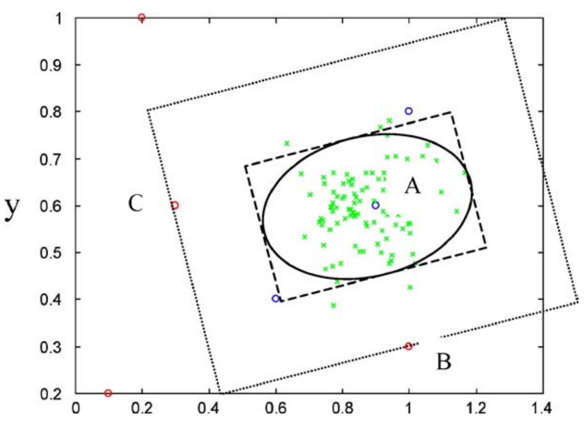

Figure 4. Illustration of Gaussian local density. A is a data example from one class, B and C are its two nearest neighbors of the opposite class. The generated random vectors are green "x"'s.

It is clear that the new vectors Z_j were mostly located within an ellipse. In our implementation, we chose $v = 0.9545$, with $M = 2$, $r^2 = 6.18$ according to [15].

The shape of the hyper-ellipsoid is controlled by the distribution of the M nearest neighbors of the opposite class of the example data s_i. We use Fig. 4 and the three examples in Fig. 5 to analyze the relationship between the distribution of the data examples in the feature space and the Gaussian CPS. In each example in Fig. 5, we marked one example data in the "blue" class, and its two nearest neighbors of opposite "red" class marked as neighbor 1 and neighbor 2. One hundred new data examples were generated by the procedure described above and are represented by green "x"'s in all examples.

It is clear that the newly generated data examples form an ellipse centered at the selected data example. The principal axes are proportional to the two eigenvalues. In both Figs. 5(a) and (b), the two nearest neighbors of the s_i are located in the same direction, and the resulting ellipse is long and narrow. In both Figs. 4 and 5(c), the two nearest neighbors of each data example are located in different directions in the feature space, more than 90° apart, the resulting ellipses are more like a circle, which means that the two eigenvalues in each example have very close values. Table 4 illustrates the eigenvalues, eigenvectors and the Euclidean distances between every example data to its nearest neighbors.

The characteristics of the eigenvalues and eigenvectors of the CPS are summarized as follows:

1. According to its definition, the first eigenvalue λ_0 is a monotonically increasing function of the distance between example data and its nearest neighbor of the opposite class. This is verified by our experiment result shown in Table 4.
2. If the nearest neighbors are located in the same orientation with respect to the example point, the CPS ellipsoid is long in the principal axis but narrow at the others, and the new data examples have high density in the ellipse. In Fig. 5(a), the two nearest

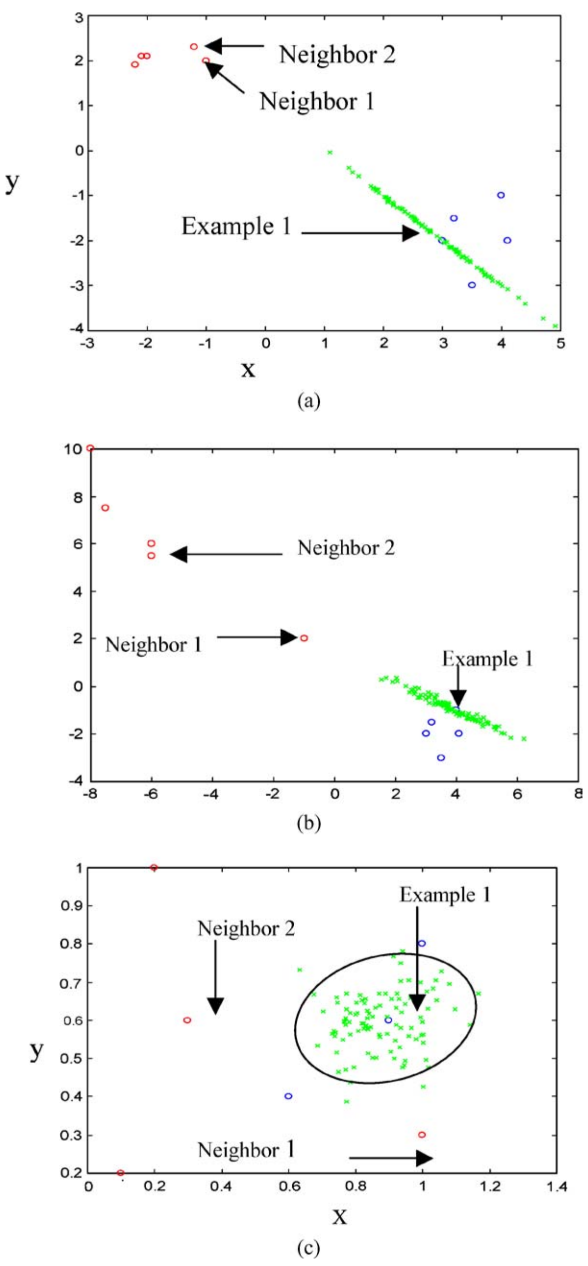

Figure 5. Three examples of different Gaussian CPS generated by different distributions of example.

Table 4. The Euclidean distances d_i, eigenvectors q_i, eigenvalues λ_i for $i = 0, 1$ of the examples shown in Figs. 1 and 2.

	Figure 1	Figure 2(a)	Figure 2(b)	Figure 2(c)
d_0	0.1211	0.9727	0.224	0.26
λ_0	0.0049	0.0393	0.00906	0.0105
E_0	$[0.2873, -0.9578]^T$	$[-0.6438, 0.7652]^T$	$[-0.873, 0.488]^T$	$[-0.1961, -0.9806]^T$
d_1	0.3600	1.10269	0.933	0.6678
λ_1	0.01336	0.000059	0.000045	0.0228
E_1	$[-0.9578, -0.2873]^T$	$[0.7652, 0.6438]^T$	$[0.488, 0.873]^T$	$[-0.9806, 0.1961]^T$

neighbors are close and at the same direction with respect to the example data, and in (b), the two nearest neighbors are in the same direction but quite apart. However the two CPS's in Figs. 5(a) and (b) are similar in shape, and the new data examples have high density in the both ellipses.

3. If the neighboring examples are far away from one another in terms of direction with respect to the example data, the corresponding principal axes in the Gaussian CPS should be similar in lengths and the CPS is more like a circle. Both Figs. 4 and 5(c) along with Table 4 show this property of CPS. Quantitatively, this property can be measured by the ratio of λ_i $i = 0, 1$. The difference between eigenvalues can help us to understand the distribution of the examples of the opposite class. For example, if λ_0 is much larger than all of the other eigenvalues $\lambda_i (I = 1, 2, \ldots M - 1)$, most of the data examples of the opposite class concentrate in the direction of the eigenvector corresponding to λ_0. If all of the M eigenvalues are very close to each other, the examples of the opposite class distribute evenly (to some extent) in a hyper-sphere around the example.

These properties of eigenvalues and the corresponding CPS ellipsoid are used to develop the following algorithm to solve the unbalanced data problem for neural network learning.

3.2. *Generating Data Examples Using Gaussian CPS*

The objective of this algorithm is to train a neural network to learn the classification features from the data examples of a minority class in the training set and to make more favorable decisions to the minority class. The algorithm we investigated was to generate new minority data examples near the classification boundary using the Gaussian CPS and add these new data examples to the training data. The neural networks trained on this set should make more favorable decision to the minority class with the minimization of misclassification of the majority class and have increased generalization capability.

The algorithm generates p new data examples around every minority data example s subject to its local Gaussian distribution of the opposite class through the following computational steps.

Let us assume the input vector is M dimensional. The noise modeling algorithm first finds the M data examples of the majority class that are closest to $s, t_1, t_2, \ldots, t_M$, from which we construct the $M \times M$ covariance matrix of the Gaussian probability density function described in the last section, and obtain M eigenvalues of the covariance matrix $\lambda_0, \lambda_1, \ldots \lambda_{M-1}$.

We need to be cautious about generating extra data examples. If we generate unnecessary ones, we may weaken the classification capability of the neural network on the majority class. For example, if λ_0 is large, it implies that s is quite apart from the majority class, and a neural network may easily learn the classification boundary around s. If we artificially generate more minority examples, we may force the trained neural network to make more classification errors on the majority class than necessary. The noise modeling algorithm generates new data examples only at the locations where it is difficult to differentiate minority data examples from the majority examples, and adding noisy random vectors does not affect too many majority examples.

Based on the properties of Gaussian CPS, we developed the following rules. Let ρ_1 and ρ_2, the number of majority and minority examples falling within the hyper bounding box R of example data s and $s, t_1, t_2, \ldots, t_M$ respectively. Specifically, R is equal to $\|s - t_1\| x \|s - t_2\| x \ldots \|s - t_M\|$.

Rule 1: If ρ_1, the density of majority class examples around s, is large, do not generate noisy data around s.

Rule 2: If ρ_2, the density of minority class examples around s, is small, do not generate noisy data around s.

Rule 3: If λ_0, the 1st eigenvalue of Gaussian covariance matrix, is large, do not generate noisy data around s.

For a minority data example s, only if s does not satisfy any of the three rules will the noise modeling algorithm generate p new data examples around s. The value p can be determined based on the ratio of the number of data examples in the majority and the minority class.

Another important issue is to limit the noise random vectors in the hyper bounding box R. As we discussed in the last section, the random vectors fall within the CPS ellipsoid with probability v. However with probability $1 - v$, the new random vectors may fall outside the Gaussian CPS. Furthermore, the CPS ellipsoid may exceed the hyper bounding box R due to the symmetry of Gaussian distribution. Since we have no knowledge of what is beyond these data examples, the noise modeling algorithm discards the new extra data examples generated beyond the hyper bounding box (see Fig. 6). Therefore, the noise modeling algorithm accepts a random vector as a noise data example only it belongs to the conjuncture of the rectangular and the ellipse (see Fig. 6, where the rectangle is the bounding box equal to $\|s - t_1\| x \|s - t_2\|$.)

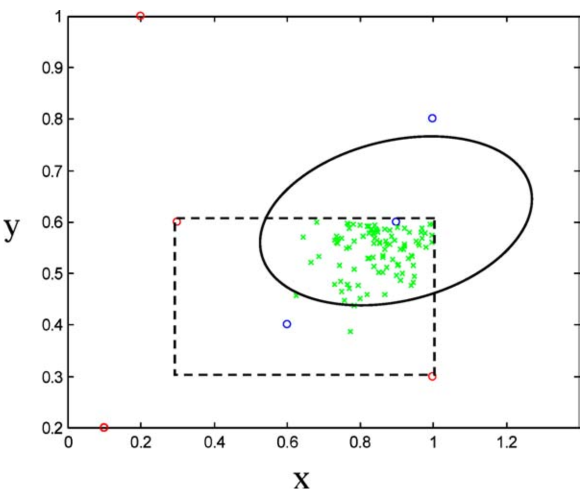

Figure 6. Only the data examples that are within the conjuncture of the bounding box and the ellipse are accepted as new training data examples.

3.3. *Experiment Results*

The ultimate goal of this study is to find a neural network architecture along with a training method that gives the best and most robust performance. In this section we will compare the experimental results generated by the Gaussian CPS method with the two relatively simple methods that also attempt to make a neural network to learn minority class features.

Duplicating Data Examples of Minority Class. The first method is called duplicating-x that duplicates data examples of the minority class by a fixed number of times in a training data set before a training process. After the duplication, the new training data set has about the same number of data examples in both classes.

Snowball Training. The snowball training method was used to train neural networks for multi-font character recognition by Wang and Jean [16]. The basic idea of the snowball method is to first train a neural network purely with the examples of the minority class so that a set of connection weights favorable to these examples are established. Then we increase the neural network's capability to recognize the examples of the majority class by using a dynamic training set which includes all the examples of minority class and an increasing number of examples of the majority class. In this way, the undo effect of the presentation of minority class examples can be greatly reduced. The algorithm was implemented as follows. Assume the trained network is R, the set of the examples in the minority class is denoted as MN. MJ, the set of all the examples in the majority class, is randomly divided into q disjoint subsets of roughly equal size, $MJ_1, MJ_2, \ldots MJ_q$, where $MJ_1 \cup MJ_2 \cup \ldots \cup MJ_q = MJ$. There are three parameters t_1, t_2, t_3 to be used during the training for the number of sweeps. First we train a neural network on MN until it converges or t_1 sweeps have been tried. The we train the neural network on a subset MJ_j of the majority class. If it converges within t_2, train the neural network on the next majority subset. If it does not converge within t_2 sweeps, train the network on $MN \cup MJ_{j-1}$ until it converges or t_1 sweeps have been tried. This process repeats until the neural network is trained on all of the majority subsets except the last

Table 5. System performance using more complicated training methods.

Model III data	BP Major/ minority class	RBF Major/ minority class	Fuzzy ARTMAP Major/minority class
Train-Test	100%/0%	91.97%/1.35%	76.45%/20.27 %
Duplicate	48.31%/32.47%	57.06%/12.16%	80.06%/27.03%
Snowball	60.54%/45.54%	88.09%/5.35%	67.31%/93.24%
Gaussian CPS algorithm	70.56%/45.57%	80.03%/8.54%	71.91%/94.25%

one MJ_q. The last step is to train the neural network on $MN \cup MJ$. If the neural network does not converge within t_3 sweeps, we repeat the above training steps starting from MN, then $MJ_1, MJ_2, \ldots MJ_q$.

Table 5 shows the experiments conducted on the Model III collection, which has been shown to have a high level of noise. The training data contains 722 normal vehicles and 134 problematic vehicles, the test data contains 361 normal vehicles and 67 problematic vehicles. For duplicating-x method, we duplicated 5 copies of every abnormal vehicle. For the snowballing method, we set $t_1 = 40$, $t_2 = 20$ and $t_3 = 30$. The $MN =$ all abnormal vehicle cases, and $MJ =$ all normal vehicle cases. The NM was divided into 10 equal size subsets, MJ_j, $j = 1, \ldots, 10$. A neural network was trained continuously with the data sets, $MN \cup MJ_1, \ldots, MN \cup MJ_{10}$.

Our experiments showed that the proposed Gaussian CPS algorithm showed better control over the error rates of both the majority and minority classes. When all three neural networks trained using the training data generated by the Gaussian CPS algorithm, the classification accuracy over the minority class improved significantly with a minimal increase of error over the majority class. Using the Duplicate-x and the Snowball training method, both the BP and RBF networks increased the classification accuracy over the minority class with the price of dramatically decreasing classification accuracy over the majority class. In general Fuzzy ARTMAP responded very well to the unbalanced noise data when snow ball or Gaussian CPS algorithm is applied to the training data. RBF did not respond well to unbalanced noise data with either training method.

4. Conclusions

This paper presents our study on solving the classification problem in which data population is unbalanced among different classes and has a high level of noise. Our study used three neural networks, BP, RBF and Fuzzy ARTMAP. We presented three methods for measuring noise levels for a given data set. Our study showed that the neural network performance on unbalanced data very much depends on how well the two classes are separated. The BP and Fuzzy ARTMAP can learn classification features over minority class when data noise level is low; the RBF network does not learn the features of the minority class well. When data is noisy, none of the three networks performs well without any additional processes over unbalanced data. When we introduced two training methods, duplicating-x and Snowball technique, into neural training, our experiments showed that they are not effective on BP and RBF networks. For the Fuzzy ARTMAP network, the Snowball technique has made great improvement on the minority class.

We presented an algorithm, noise modeling algorithm, for training a neural network to learn the classification features from unbalanced data examples. The algorithm was developed based on the Gaussian CPS theory to generate random noise over the classification boundary for a given training set with the aim of increasing classification features of a given class and the capability of generalization for the neural networks. We showed through experimental results that the noise modeling algorithm is effective in the training of both BP and FUZZY ARTMAP neural networks. We speculate that the algorithm can be extrapolated to the general classification problem of P classes within which p classes are to be emphasized, where $p < P$. By generating noisy data examples along the classification boundaries for these p classes using the noise modeling algorithm, the trained neural network would have increased classification capability and generalization ability over the p classes.

Acknowledgments

This work was supported in part by a Grant from NSF DMII and a research contract from the Ford Motor Company.

References

1. Y. Lu, H. Guo, and L. Feldkamp, "Robust neural learning from unbalanced data examples," *IEEE IJCNN*, 1998.
2. C.H. Dagli (Eds.), *Artificial Neural Networks for Intelligent Manufacturing*, 1992.
3. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, 1992.
4. D. Mackay, "Bayesian methods for adaptive models," Ph.D thesis, CIT, 1991.
5. B. Irie and S. Miyake, "Capabilities of three-layered perceptrons," in *Proc. of the International Conference on Neural Networks*, pp. 641–648, 1988.
6. D.E. Rummelhart and J.L. McClelland. *Parallel distributed processing: Explorations in the microstructure of cognition*, vol 1: *Foundations*. MIT Press: Cambridge, MA, 1986.
7. J. Moody and C.J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.
8. G.A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, pp. 4919–4930, 1987.
9. G.A. Carpenter, S. Grossberg, N. Markuzon et al., "Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps", *IJCNN*, June 1992, pp. 309–314.
10. G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, vol. 3, pp. 698–713, 1992.
11. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press Inc., 1972.
12. S.M. Weiss and C.A. Kulikowski, *Computer Systems that Learn*, Morgan Kaufmann Publishers, Inc., 1991.
13. S. Amari, N. Murata, K.-R. Muller, M. Finke, and H. Yang, "Statistical theory of overtraining—Is cross-validation asymptotically effective?," Advances in Neural Information Processing Systems 8, *Proceedings of the 1995 Conference*, David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo (Eds.), 1996, pp. 176–182.
14. R.Y. Rubinstein, *Simulation and the Monte Carlo method*, John Wiley & Sons, 1981.
15. M.T. Musavi, K.H. Chan, D.M. Hummels, and K. Kalantri, "On the generalization ability of neural network classifiers," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 659–663, 1994.
16. J. Wang and J. Jean, "Resolve multifont character confusion with neural network," *Pattern Recognition*, vol. 26, no. 1, pp. 173–187, 1993.