Training Deep Neural Networks on Imbalanced Data Sets

Shoujin Wang [1], Wei Liu [1], Jia Wu [2], Longbing Cao [1], Qinxue Meng [2], Paul J. Kennedy [2]

[1] Advanced Analytics Institute, University of Technology Sydney, Sydney, Australia

[2] Centre for Quantum Computation & Intelligent Systems, University of Technology Sydney, Sydney, Australia

Email: Shoujin.Wang@student.uts.edu.au,{Wei.Liu, Longbing.Cao}@uts.edu.au, {Jia.Wu, Qinxue.Meng, Paul.Kennedy}@uts.edu.au

***Abstract*—Deep learning has become increasingly popular in both academic and industrial areas in the past years. Various domains including pattern recognition, computer vision, and natural language processing have witnessed the great power of deep networks. However, current studies on deep learning mainly focus on data sets with balanced class labels, while its performance on imbalanced data is not well examined. Imbalanced data sets exist widely in real world and they have been providing great challenges for classification tasks. In this paper, we focus on the problem of classification using deep network on imbalanced data sets. Specifically, a novel loss function called *mean false error* together with its improved version *mean squared false error* are proposed for the training of deep networks on imbalanced data sets. The proposed method can effectively capture classification errors from both majority class and minority class equally. Experiments and comparisons demonstrate the superiority of the proposed approach compared with conventional methods in classifying imbalanced data sets on deep neural networks.**

Keywords—deep neural network; loss function; data imbalance

I. INTRODUCTION

Recently, rapid developments in science and technology have promoted the growth and availability of data at an explosive rate in various domains. The ever-increasingly large amount of data and more and more complex data structure lead us to the so called "big data era". This brings a great opportunity for data mining and knowledge discovery and many challenges as well. A noteworthy challenge is data imbalance. Although more and more raw data is getting easy to be accessed, much of which has imbalanced distributions, namely a few object classes are abundant while others only have limited representations. This is termed as the "Class-imbalance" problem in data mining community and it is inherently in almost all the collected data sets [1]. For instance, in the clinical diagnostic data, most of the people are healthy while only a quite low proportion of them are unhealthy. In classification tasks, data sets are usually classified into binary-class data sets and multi-class data sets according to the number of classes. Accordingly, classification can be categories as binary classification and multi-class classification [2], [3]. This paper mainly focuses on the binary-classification problem and the experimental data sets are binary-class ones (A multi-class problem can generally be transformed into a binary-class one by binarization). For a binary-class data set, we call the data set imbalanced if the minority class is under-represented compared to the majority class, e.g., the majority class severely out represents the minority class [4].

Data imbalance can lead to unexpected mistakes and even serious consequences in data analysis especially in classification tasks. This is because the skewed distribution of class instances forces the classification algorithms to be biased to majority class. Therefore, the concepts of the minority class are not learned adequately. As a result, the standard classifiers (classifiers don't consider data imbalance) tend to misclassify the minority samples into majority samples when the data is imbalanced, which results in quite poor classification performance. This may lead to a heavy price in real life. Considering the above diagnose data example, it is obvious that the patient constitute the minority class while the healthy persons constitute the majority one. If a patient was misdiagnosed as a healthy person, it would delay the best treatment time and cause significant consequences.

Though data imbalance has been proved to be a serious problem, it is not addressed well in the standard classification algorithms. Most of classifiers were designed under the assumption that the data is balanced and evenly distributed on each class. Many efforts have been made in some well-studied classification algorithms to solve this problem professionally. For example, sampling techniques and cost sensitive methods are broadly applied in SVM, neural network and other classifiers to solve the problem of class imbalance from different perspectives. Sampling aims to transfer the imbalanced data into balanced one by various sampling techniques while cost sensitive methods try to make the standard classifiers more sensitive to the minority class by adding different cost factors into the algorithms.

However, in the field of deep learning, very limited work has been done on this issue to the best of our knowledge. Most of the existing deep learning algorithms do not take the data imbalance problem into consideration. As a result, these algorithms can perform well on the balanced data sets while their performance cannot be guaranteed on imbalanced data sets.

In this work, we aim to address the problem of class imbalance in deep learning. Specifically, different forms of loss functions are proposed to make the learning algorithms more sensitive to the minority class and then achieve higher classification accuracy. Besides that, we also illustrate why we

978-1-5090-0620-5/16/$31.00 ©2016 IEEE

propose this kind of loss function and how it can outperform the commonly used loss function in deep learning algorithms.

Currently, mean squared error (MSE) is the most commonly used loss function in the standard deep learning algorithms. It works well on balanced data sets while it fails to deal with imbalanced ones. The reason is that MSE captures the errors from an overall perspective, which means it calculates the loss by firstly sum up all the errors from the whole data set and then calculates the average value. This can capture the errors from the majority and minority classes equally when the binary-classes data sets are balanced. However, when the data set is imbalanced, the error from the majority class contributes much more to the loss value than the error from the minority class. In this way, this loss function is biased towards majority class and fails to capture the errors from two classes equally. Further, the algorithms are very likely to learn biased representative features from the majority class and then achieve biased classification results.

To make up for this shortcoming of mean squared error loss function used in deep learning, a new loss function called ***mean false error*** (MFE) together with its improved version ***mean squared false error*** (MSFE) are proposed. Being different from MSE loss function, our proposed loss functions can capture the errors both from majority class and minority class equally. Specifically, our proposed loss functions firstly calculate the average error in each class separately and then add them together, which is demonstrated in part III in detail. In this way, each class can contribute to the final loss value equally.

TABLE I. AN EXAMPLE OF CONFUSION MATRIX

		Predicted Class		
		P	***N***	
True Class	***P'***	86	4	90
	N'	5	5	10
		91	9	

Let's take the binary classification problem shown in Table I as an example. For the classification problem in Table I, we compute the loss value using MSE, MFE and MSFE respectively as follows. Please note that here is just an example for the calculation of three different loss values, the formal definitions of these loss functions are given in part III. Please note that in this binary classification problem, the error of a certain sample is 0 if the sample is predicted correctly, otherwise the error is 1.

$$l_{\mathrm{MSE}} = \frac{4+5}{90+10} = 0.09 \tag{1.1}$$

$$l_{\mathrm{MFE}} = \frac{5}{10} + \frac{4}{90} = 0.54 \tag{1.2}$$

$$l_{\mathrm{MSFE}} = (\frac{5}{10})^2 + (\frac{4}{90})^2 = 0.25 \tag{1.3}$$

From table I, it is quite clear that the overall classification accuracy is (86+5)/(90+10)=91%. However, different loss values can be achieved when different kinds of loss functions are used as showed in Eq. (1.1) to Eq. (1.3). In addition, the loss values computed using our proposed MFE and MSFE loss functions are much larger than that of MSE. This means a higher loss values can be achieved when MFE (MSFE) is used as the loss function instead of MSE under the same classification accuracy. In other words, under the condition of the same loss values, a higher classification accuracy can be achieved on imbalanced data sets when MFE (MSFE) is used as the loss function rather than MSE. This empirically demonstrates that our proposed loss functions can outperform the commonly used MSE in imbalanced data sets. It should be noted that only the advantages of MFE and MSFE over MSE are illustrated here, and the reason why MSFE is introduced as an improved version of MFE will be given in part III.

The contributions of this paper are summarized as:

(1). Two novel loss functions are proposed to solve the data imbalance problem in deep network.

(2). The advantages of these proposed loss functions over the commonly used MSE are analyzed theoretically.

(3). The effect of these proposed loss functions on the back-propagation process of deep learning is analyzed by examining relations for propagated gradients.

(4). Empirical study on real world data sets is conducted to validate the effectiveness of our proposed loss functions.

The left parts of this paper are organized as follows. In part II, we review the previous studies addressing data imbalance. Followed by problem formulation and statement in part III, a brief introduction of DNN is given in part IV. Part V describes the experiments of applying our proposed loss functions on real world data sets. Finally, the paper concludes in part VI.

II. RELATED WORK

How to deal with imbalanced data sets is a key issue in classification and it is well explored during the past decades. Until now, this issue is solved mainly in three ways, sampling techniques, cost sensitive methods and the hybrid methods combining these two. This section reviews these three mainstream methods and then gives a special review on the imbalance problem in neural network field, which is generally thought to be the ancestor of deep learning and deep neural network.

A.Sampling techinique

Sampling is thought to be a pre-processing technique as it deals with the data imbalance problem from data itself perspective. Specifically, it tries to provide a balanced distribution by transferring the imbalanced data into balanced one and then works with classification algorithms to get results. Various sampling techniques have been proposed from different perspectives. Random oversampling [5], [6] is one of the simplest sampling methods. It randomly duplicates a certain number of samples from the minority class and then augment them into the original data set. On the contrary, under-sampling randomly remove a certain number of instances from the majority class to achieve a balanced data set. Although these sampling techniques are easy to implement and effective, they may bring some problems. For example, random oversampling may lead to overfitting while random under-sampling may lose some important information. To

avoid these potential issues, a more complex and reasonable sampling method is proposed. Specifically. The synthetic minority oversampling technique (SMOTE) has proven to be quite powerful which has achieved a great deal of success in various applications [7], [8]**.** SMOTE creates artificial data based on the similarities between existing minority samples. Although many promising benefits have been shown by SMOTE, some drawbacks still exist like over generalization and variance [9], [10].

B.Cost sensitive learning

In addition to sampling technique, another way to deal with data imbalance problem is cost sensitive learning. It targets at the data imbalance problem from the algorithm perspective. Be contrasted with sampling methods, cost sensitive learning methods solve data imbalance problem based on the consideration of the cost associated with misclassifying samples [11]. In particular, it assigns different cost values for the misclassification of the samples. For instance, the cost of misclassifying a patient into a healthy man would much higher than the opposite. This is because the former may lose the chance of the best treatment and even lose one's life while the latter just leads to more examinations. Typically, in a binary classification, the cost is zero for correct classification for either class and the cost of misclassifying minority is higher than misclassifying majority. An objective function for the cost sensitive learning can be constructed based on the aggregation of the overall cost on the whole training set. An optimal classifier can be learned by minimizing the objective function [12,13,23]. Though cost sensitive algorithms can significantly improve the classification performance, they can be only applicable when the specific cost values of misclassification are known. Unfortunately, in many cases, an explicit description of the cost is hard to define, instead, only an informal assertion is known like "the cost of misclassification of minority samples is higher than the contrary situation" [14]. In addition, it would be quite challenging and even impossible to determine the cost of misclassification in some particular domains [15].

C.Imbalance problem in neural network

In the area of neural network, many efforts have been made to address data imbalance problem. Nearly all of the work falls into the three main streams of the solutions to imbalance problem mentioned above. In particular, it's the specific implementations of either sampling or cost sensitive methods or their combinations on neural networks though the details may differ. Kukar presented a few different approaches for cost-sensitive modifications of the back-propagation learning algorithm for multilayered feed forward neural networks. He described four approaches to learn cost sensitive neural networks by adding cost factors to different parts of the original back propagation algorithms. As a result, cost-sensitive classification, adapting the output of the network, adapting the learning rate and minimization the misclassification costs are proposed [16]. Zhou empirically studied the effect of sampling and threshold-moving in training cost-sensitive neural networks. Both oversampling and under sampling techniques are used to modify the distribution of the training data set. Threshold-moving tries to move the output threshold toward inexpensive classes such that examples with higher costs become harder to be misclassified [17]. Other similar work on this issue includes [18,19,20,24]. Although some work has been done to solve the data imbalance problem in neural network, quite few literatures related to the imbalance problem of deep network can be seen so far. How to tackle the data imbalance problem in the learning process of deep neural network is of great value to explore. In particular, it can broaden the application situations of the powerful deep neural network, making it not only work well on balanced data but also on imbalanced data.

III. PROBLEM FORMULATION

We address the data imbalance problem during the training of deep neural network (DNN) . Specifically, we mainly focus on the loss function.

Generally, an error function expressed as the loss over the whole training set is introduced in the training process of the DNN. A set of optimal parameters for a DNN is achieved by minimizing errors in training the DNN iteratively. A general form of error function is given in equation (3.1):

$$E(\theta) = l\left(\boldsymbol{d}^{(i)}, \boldsymbol{y}_{\theta}^{(i)}\right), \tag{3.1}$$

where the predicted output of i^{th} object $\boldsymbol{y}_{\theta}^{(i)}$ is parameterized by the weights and biases θ of the network. For simplicity, we will just denote $\boldsymbol{y}_{\theta}^{(i)}$ as $\boldsymbol{y}^{(i)}$ or $\boldsymbol{y}$ in the following discussions. l denotes a kind of loss function. $\boldsymbol{d}^{(i)} \in \{0,1\}^{1\times n}$ is the desired output with the constraint $\sum_{\mathrm{n}} \boldsymbol{d}_{\mathrm{n}} := 1$ and n is the total number of neurons in the output layer , which is equal to the number of classes. In this work, we only consider the binary classification problem, so n=2. Note that, the value of $E(\theta)$ is higher when the model performs poorly on the training data set. The learning algorithm aims to find the optimal parameter (θ^*) which brings the minimum possible error value $E^*(\theta)$. Therefore, the optimization objective is expressed as:

$$\theta^* = \arg\min_{\theta} E(\theta). \tag{3.2}$$

The loss function $l(\cdot)$ in Eq. (3.1) can be in many different forms, such as the Mean Squared Error (MSE) or Cross Entropy (CE) loss. Out of the various forms of loss function, MSE is the most widely used in the literature. Next, we will first give a brief introduction of the commonly used loss function and then propose two kinds of novel loss functions which target at imbalanced data sets.

A.MSE loss:

This kind of loss function minimizes the squared error between the predicted output and the ground-truth and can be expressed as follows:

$$l = \frac{1}{M}\sum_i \sum_n \frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \tag{3.3}$$

where M is the total number of samples and $d_n^{(i)}$ represents the desired value of i^{th} sample on n^{th} neuron while $y_n^{(i)}$ is the corresponding predicted value. For instance, in the scenario of binary classification, if the 4th sample actually belonged to the second class while it is predicted as the first class incorrectly, then the label vector and prediction vector for this sample is $\boldsymbol{d}^{(4)} = [0,1]^T$ and $\boldsymbol{y}^{(4)} = [1,0]^T$ respectively. Further, we have

$d_1^{(4)} = 0$ and $d_2^{(4)} = 1$ while $y_1^{(4)} = 1$ and $y_2^{(4)} = 0$. So the error of this sample is 1/2*((0-1)^2+(1-0)^2)=1, further all the error of a collection of samples predicted by a classifier is the number of incorrectly predicted samples in binary classification problem, which can be seen from Eq. (1.1).

In addition, $y_n^{(i)}$ can be expressed as a function of the output of the previous layer $o_n^{(i)}$ using the logistic function [16]:

$$y_n^{(i)} = \frac{1}{1+\exp(-o_n^{(i)})} \tag{3.4}$$

B.MFE loss:

Now we introduce the ***Mean False Error*** loss function proposed by our research. The concept "false error" is inspired by the concepts "false positive rate" and "false negative rate" in the confusion matrix and it is defined with the consideration of both false positive error and false negative error. This kind of loss function is designed to improve the classification performance on the imbalanced data sets. Specifically, it makes the loss more sensitive to the errors from the minority class compared with the commonly used MSE loss by computing the errors on different classes separately. Formally,

$$FPE = \frac{1}{N}\sum_{i=1}^{N}\sum_n \frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \tag{3.5}$$

$$FNE = \frac{1}{P}\sum_{i=1}^{P}\sum_n \frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2 \tag{3.6}$$

$$l' = FPE + FNE \tag{3.7}$$

where FPE and FNE are mean false positive error and mean false negative error respectively and they capture the error on the negative class and positive class correspondingly. The loss l' is defined as the sum of the mean error from the two different classes, which is illustrated in Eq. (3.7). *N* and *P* are the numbers of samples in negative class and positive class respectively. A specific example to calculate l' is illustrated in Eq. (1.2) where the part $\sum_n \frac{1}{2}(d_n^{(i)} - y_n^{(i)})^2$ is simplified to 1 based on the computation result in part Ⅲ.A. In imbalanced classification issues, researchers usually care more about the classification accuracy of the minority class. Thereby the minority class is treated as the positive class in most works. Without loss of generality, we also let the minority class to be the positive class in this work.

Note that only the form of loss function is redefined in our work compared to the traditional deep network .Therefore, $d_n^{(i)}$ and $y_n^{(i)}$ are associated with the same meanings as they are in the MSE scenario and $y_n^{(i)}$ here is still computed using Eq. (3.4) .

C.MSFE loss:

The ***Mean Squared False Error*** loss function is designed to improve the performance of MFE loss defined before. Firstly, it calculates the FPE and FNE values using Eq. (3.5) and Eq. (3.6). Then another function rather than Eq. (3.7) will be used to integrate FPE and FNE. Formally,

$$l'' = FPE^2 + FNE^2 \tag{3.8}$$

A specific example to calculate l'' is illustrated in Eq. (1.3). The reason why MSFE can improve the performance of MSE is explained here. In the MFE scenario, when we minimize the loss, it can only guarantee the minimization of the sum of FPE and FNE, which is not enough to guarantee high classification accuracy on the positive class. To achieve high accuracy on positive class, the false negative error should be quite low. While in the imbalanced data sets, the case is that FPE tends to contribute much more than FNE to their sum (the MFE loss) due to the much more samples in negative class than the positive class. As a result, the MFE loss is not sensitive to the error of positive class and the minimization of MFE cannot guarantee a perfect accuracy on positive class. The MSFE can solve this problem effectively. Importantly, the loss function in MSFE can be expressed as follows:

$$\begin{aligned} l'' &= FPE^2 + FNE^2 \\ &= \frac{1}{2}((FPE + FNE)^2 + (FPE - FNE)^2) \end{aligned} \tag{3.9}$$

So the minimization of MSFE is actually to minimize $(FPE + FNE)^2$ and $(FPE - FNE)^2$ at the same time. In this way, the minimization operation in the algorithm is able to find a minimal sum of FPE and FNE and minimal the difference between them. In other words, both the errors on positive class and negative class will be minimized at the same time, which can balance the accuracy on the two classes while keeping high accuracy on the positive class.

Different loss functions lead to different gradient computations in the back-propagation algorithms. Next, we discuss the gradient computations when using the above different loss functions.

D.MSE loss back-propagation:

During the supervised training process, the loss function minimizes the difference between the predicted outputs $\boldsymbol{y}^{(i)}$ and the ground-truth labels $\boldsymbol{d}^{(i)}$ across the entire training data set (Eq. (3.5)). For the MSE loss, the gradient at each neuron in the output layer can be derived from Eq. (3.3) as follows:

$$\frac{\partial l(\boldsymbol{d}^{(i)}, \boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -(d_n^{(i)} - y_n^{(i)})\frac{\partial y_n^{(i)}}{\partial o_n^{(i)}} \tag{3.10}$$

Based on Eq. (3.4), the derivative of $y_n^{(i)}$ with respect to $o_n^{(i)}$ is:

$$\frac{\partial y_n^{(i)}}{\partial o_n^{(i)}} = y_n^{(i)}(1 - y_n^{(i)}) \tag{3.11}$$

The derivative of the loss function in the output layer with respect to the output of the previous layer is therefore given by the Eq. (3.12):

$$\frac{\partial l(\boldsymbol{d}^{(i)}, \boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -(d_n^{(i)} - y_n^{(i)})y_n^{(i)}(1 - y_n^{(i)}) \tag{3.12}$$

E.MFE loss back-propagation:

For the MFE loss given in Eq. (3.5) to Eq. (3.7), the derivative can be calculated at each neuron in the output layer as follows:

$$\frac{\partial l(\boldsymbol{d}^{(i)}, \boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = \frac{\partial FPE}{\partial o_n^{(i)}} + \frac{\partial FNE}{\partial o_n^{(i)}}$$

$$= -\frac{1}{N}\left(d_n^{(i)} - y_n^{(i)}\right)\frac{\partial y_n^{(i)}}{\partial o_n^{(i)}} - \frac{1}{P}\left(d_n^{(i)} - y_n^{(i)}\right)\frac{\partial y_n^{(i)}}{\partial o_n^{(i)}} \quad (3.13)$$

Substitute Eq. (3.11) into Eq. (3.13), we can get the derivative of the MFE loss with respect to the output of the previous layer:

$$\begin{cases} \frac{\partial l(\boldsymbol{d}^{(i)},\boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -\frac{1}{N}\left(d_n^{(i)} - y_n^{(i)}\right)y_n^{(i)}\left(1 - y_n^{(i)}\right), (i \in \boldsymbol{N}) & (3.14) \\ \frac{\partial l(\boldsymbol{d}^{(i)},\boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -\frac{1}{P}\left(d_n^{(i)} - y_n^{(i)}\right)y_n^{(i)}\left(1 - y_n^{(i)}\right), (i \in \boldsymbol{P}) & (3.15) \end{cases}$$

where N and P are the numbers of samples in negative class and positive class respectively. $\boldsymbol{N}$ and $\boldsymbol{P}$ are the negative sample set and positive sample set respectively. Specifically, we use different derivatives for samples from each class. Eq. (3.14) is used when the sample is from the negative class while Eq. (3.15) is used when it belongs to the positive class.

F.MSFE loss back-propagation:

For the MSFE loss given in Eq. (3.8), the derivative can be calculated at each neuron in the output layer as follows:

$$\frac{\partial l(\boldsymbol{d}^{(i)},\boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = 2FPE \cdot \frac{\partial FPE}{\partial o_n^{(i)}} + 2FNE \cdot \frac{\partial FNE}{\partial o_n^{(i)}} \quad (3.16)$$

where $\frac{\partial FPE}{\partial o_n^{(i)}}$ and $\frac{\partial FNE}{\partial o_n^{(i)}}$ have been computed in Eq. (3.13), substitute it into Eq. (3.16), the derivatives at each neuron for different classes can be given as:

$$\begin{cases} \frac{\partial l(\boldsymbol{d}^{(i)},\boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -\frac{2FPE}{N}\left(d_n^{(i)} - y_n^{(i)}\right)y_n^{(i)}\left(1 - y_n^{(i)}\right), (i \in \boldsymbol{N}) & (3.17) \\ \frac{\partial l(\boldsymbol{d}^{(i)},\boldsymbol{y}^{(i)})}{\partial o_n^{(i)}} = -\frac{2FNE}{P}\left(d_n^{(i)} - y_n^{(i)}\right)y_n^{(i)}\left(1 - y_n^{(i)}\right), (i \in \boldsymbol{P}) & (3.18) \end{cases}$$

where N and P together with $\boldsymbol{N}$ and $\boldsymbol{P}$ have the same meanings as that used in MFE loss. Similarly, for the samples from different classes, different derivatives are used in the training process.

IV. DEEP NEURAL NETWORK

We use deep neural network (DNN) to learn the feature representation from the imbalanced and high dimensional data sets for classification tasks. Specifically, here DNN refers to neural networks with multiple hidden layers. With multiple layers, DNN owns a strong generalization and extraction ability for data especially for those high dimensional data sets. The structure of the network used in this work is similar to the classical deep neural network illustrated in [21] except that the proposed loss layer is more sensitive to imbalanced data sets using our proposed loss functions. Note that the DNN in our work is trained with MFE loss and MSFE loss proposed by us in Eq. (3.5) to Eq. (3.8) while DNN trained with MSE loss will be used as a baseline in our experiment.

How to determine network structure parameters like the number of layers and the number of neurons in each layer is a difficult problem in the training of deep networks and it's out of the scope of this work. In our work, different numbers of layers and neurons for DNN (use MSE loss function) are tried on each data set. Those parameters which make the network achieve the best classification performance are chosen to build the network in our work. For example, for the Household data set used in our experiment, a DNN with MSE as the loss function is built to decide the network structure. Specifically, we first use one hidden layer to test the classification performance of the DNN on that data set and then add to two hidden layers, three hidden layers or more. Similarly, when the number of hidden layers is chosen, different numbers of neurons on those hidden layers are examined on the same data set until to gain the best classification performance. Using this heuristic approach, the structure of DNN with the best performance is chosen for each specific data set in our experiment. It should be noted that, when the number of layers increases, the classification performance firstly increases to a peak point and then decrease. Things are the same for the number of neurons. The specific settings are shown in Table Ⅱ.

TABLE II. DNN PARAMETER SETTING

Data set	Number of Hidden Layers	Number of Neurons on Hidden Layers (from bottom to up)
Household	3	1000, 300, 100
Tree 1	3	1000, 100, 10
Tree 2	3	1000, 100, 10
Doc. 1	6	3000, 1000, 300, 100, 30, 10
Doc. 2	6	3000, 1000, 300, 100, 30, 10
Doc. 3	6	3000, 1000, 300, 100, 30, 10
Doc. 4	6	3000, 1500, 800, 400, 200, 50
Doc. 5	6	3000, 1500, 800, 400, 200, 50

V. EXPERIMENTS AND RESULTS

In this section, we evaluate the effectiveness of our proposed loss functions on 8 imbalanced data sets, out of which, three ones are images extracted from the CIFAR-100 data set and five ones are documents extracted from the 20 Newsgroup data set. All of them are of high dimensions, specifically, the image data sets have 3072 dimensions while the documents own 11669 dimensions extracted from the original 66861 dimensions. Each data set contains various numbers of samples and they are splatted into the training set and testing set. The deep neural networks (DNNs) are firstly trained on the training set and then tested on the testing set in terms of their classification performance. To test the classification performances of our proposed methods under different imbalance degrees, the DNNs are trained and tested when each data set is with different levels of imbalance. The details of the data sets and experimental settings will be explained in the next section.

A.Data sets and experimental settings

Image Classification: CIFAR-100 contains 60,000 images belonging to 100 classes (600 images/class) which are further divided into 20 superclasses. The standard train/test split for each class is 500/100 images. To evaluate our algorithm on various scales data sets, three data sets of different sizes are extracted from this data set. The first one is relatively large and it is the mixture of two superclasses *household furniture* and *household electrical devices*, which is denoted as Household in the experiment. The other two small ones have approximate sizes, each of which is the combination of two classes randomly selected from the superclass *trees*. Specifically, one is the mixture of *maple tree* and *oak tree* and the other is the blending of *maple tree* and *palm tree*. These two data sets are

denoted as Tree 1 and Tree 2 respectively in the experiment. To imbalance the data distribution to different degrees, we reduce the representation of one of the two classes in each extracted data set to 20%, 10% and 5% images respectively.

Document classification: 20 Newsgroups is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups with around 600 documents contained in each newsgroup. We extract five data sets from this data set with two randomly selected newsgroups contained in each one. To be more specific, the five extracted data sets are the mixture of *alt.atheism* and *rec.sport.baseball*, *alt.atheism* and *rec.sport.hockey*, *talk.politics.misc* and *rec.sport.baseball*, *talk.politics.misc* and *rec.sport.hockey*, *talk.religion.misc* and *soc.religion.christian* respectively, which are denoted as Doc.1 to Doc.5 correspondingly in the experiment. To transform the data distribution into different imbalance levels, we reduce the representation of one of the two classes in each data set to 20%, 10% and 5% of documents respectively.

B.Experimental results

To evaluate our proposed algorithm, we compare the classification performances of the DNN trained using our proposed MFE and MSFE loss functions with that of the DNN trained using conventional MSE loss function respectively. To be more specific, DNNs are trained with one of the three loss functions each time on one data set in the training procedure. As a result, three DNNs with different parameters (weights and bias) are achieved to make prediction in the following testing procedure. To characterize the classification performance on the imbalanced data sets more effectively, two metrics F-measure and AUC [22] which are commonly used in the imbalanced data sets are chosen as the evaluation metrics in our experiments. In general, people focus more on the classification accuracy of the minority class rather than the majority class when the data is imbalanced. Without loss of generality, we mainly focus on the classification performance of minority class which is treated as the positive class in the experiments. We conduct experiments on the three image data sets and five document data sets mentioned before and the corresponding results are shown in Table Ⅲ and Table Ⅳ respectively. In the two tables, the Imb. level means the imbalance level of the data sets. For instance, the value 20% of Imb. level in the Household data set means the number of samples in the minority class equals to twenty precents of the majority one.

TABLE III. EXPERIMENTAL RESULTS ON THREE IMAGE DATA SETS

Data set	Imb. level	*F-measure*			*AUC*		
		DNN (MSE)	*DNN (MFE)*	*DNN (MSFE)*	*DNN (MSE)*	*DNN (MFE)*	*DNN (MSFE)*
Household	20%	0.3913	**0.4138**	**0.4271**	0.7142	**0.7397**	**0.7354**
	10%	0.2778	**0.2797**	**0.3151**	0.7125	**0.7179**	**0.7193**
	5%	0.1143	**0.1905**	**0.2353**	0.6714	**0.695**	**0.697**
Tree 1	20%	0.55	0.55	0.5366	0.81	**0.814**	**0.8185**
	10%	0.4211	0.4211	0.4211	0.796	**0.799**	**0.799**
	5%	0.1667	**0.2353**	**0.2353**	0.792	**0.8**	**0.8**
Tree 2	20%	0.4348	0.4255	0.4255	0.848	0.845	0.844
	10%	0.1818	**0.2609**	**0.25**	0.805	0.805	**0.806**
	5%	0	**0.1071**	**0.1481**	0.548	**0.652**	**0.7**

TABLE IV. EXPERIMENTAL RESULTS ON FIVE DOCUMENT DATA SETS

Data set	Imb. level	*F-measure*			*AUC*		
		DNN (MSE)	*DNN (MFE)*	*DNN (MSFE)*	*DNN (MSE)*	*DNN (MFE)*	*DNN (MSFE)*
Doc. 1	20%	0.2341	**0.2574**	**0.2549**	0.5948	**0.5995**	**0.5987**
	10%	0.1781	**0.1854**	**0.1961**	0.5349	**0.5462**	**0.5469**
	5%	0.1356	**0.1456**	**0.1456**	0.5336	**0.5436**	**0.5436**
Doc. 2	20%	0.3408	0.3393	0.3393	0.6462	**0.6464**	**0.6464**
	10%	0.2094	0.2	0.2	0.631	**0.6319**	**0.6322**
	5%	0.1256	0.1171	**0.1262**	0.6273	**0.6377**	**0.6431**
Doc. 3	20%	0.2929	**0.2957**	**0.2957**	0.5862	**0.587**	**0.587**
	10%	0.1596	**0.1627**	**0.1698**	0.5577	**0.5756**	**0.5865**
	5%	0.0941	**0.1118**	**0.1084**	0.5314	**0.5399**	**0.5346**
Doc. 4	20%	0.3723	**0.3843**	0.3668	0.6922	**0.7031**	**0.7054**
	10%	0.1159	**0.2537**	**0.2574**	0.5623	**0.6802**	**0.6816**
	5%	0.1287	**0.172**	**0.172**	0.6041	**0.609**	**0.609**
Doc. 5	20%	0.3103	**0.3222**	**0.3222**	0.6011	0.5925	0.5925
	10%	0.1829	0.1808	**0.1839**	0.5777	**0.5836**	**0.5837**
	5%	0.0946	**0.1053**	**0.1053**	0.5682	**0.573**	**0.573**

The classification performance of DNNs trained using different loss functions on different data sets is shown in Table Ⅲ and Table Ⅳ. Specifically, for each data set, the more imbalanced the data is the worse classification performance we achieve, which is illustrated by the general downward trends of both F-measure and AUC with the increase of the imbalance degree (the smaller the Imb. level the more imbalanced the data set is). More importantly, for most of the data sets, the DNNs trained using MFE or MSFE loss functions achieve either equal or better performances than the DNNs trained using MSE loss function on the same data set associated with the same imbalance level (Those results from our algorithms better than that from the conventional algorithms are in bold face in Table Ⅲ and Table Ⅳ). These results empirically verify the theoretical analysis in part Ⅲ. One more interesting thing is that, our proposed methods can lift the F-measure and AUC more obviously in the extremely imbalanced data sets such as those data sets with Imb. level of 5%, which shows the more imbalanced the data is the more effective our methods are. For example, in the Tree 2 data set, when Imb. level is 5%, the boosting values of F-measure and AUC are 0.1071 and 0.104 respectively by replacing MSE with MFE, while the boosting values are only 0.0791 and 0 under the Imb. level of 10%.

In addition to the optimal classification performance of the algorithms on each data set shown in Table Ⅲ and Table Ⅳ, we also test the performances of these algorithms under the same loss values on some data sets. Specifically, the F-measure and AUC values of our proposed MFE and MSFE algorithms and the baseline MSE algorithm are illustrated in Fig.1 and Fig.2 along with the decrease of the loss values on the Household data set. It can be clearly seen that both the F-meas. and AUC resulted from MFE and MSFE are much higher than those resulted from MSE under all the loss values. This empirically verifies the theory analysis illustrated in the introduction part that higher classification accuracy can be achieved on imbalanced data sets when MFE (MSFE) is used as the loss function rather than MSE. Another advantage of our methods is that the performance is more stable compared with the heavily fluctuated performance resulted from MSE methods, which is clearly shown by the relatively smooth curves achieved by our methods together with the jumping

curve obtained from MSE related approach. This can benefit much on the gradient descent optimization during the training of DNN. Specifically, with a relatively stable trend, the optimal point with the best performance can be found more easily.

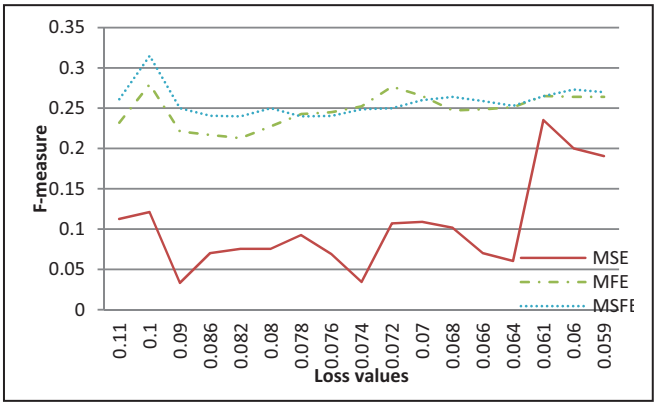

Fig. 1. Our proposed MFE and MSFE methods always achieve higher F-measure values than the conventional MSE method under the same loss values on household data set with Imb.level of 10% (Only the parts of the three curves under the common loss values are shown).

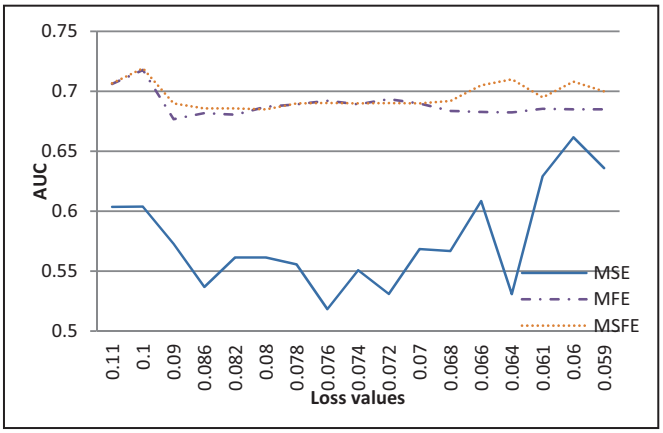

Fig.2. Our proposed MFE and MSFE approches achiveve higher AUC than the MSE approch under the same loss values on household data set with Imb.level of 10% (Only the parts of the three curves under the common loss values are shown).

VI. CONCLUSIONS

Although deep neural networks have been widely explored and proven to be effective on a wide variety of balanced data sets, few studies paid attention to the data imbalance problem. In order to resolve this issue, we proposed a novel loss function MFE plus its improved version MSFE used for the training of deep neural network (DNN) to deal with the class-imbalance problem. We demonstrated their advantages over the conventional MSE loss function from the theory perspective and their effects on the back propagation procedures in DNN training. Experimental results on both image and document data sets show that our proposed loss functions outperform the commonly used MSE on imbalanced data sets, especially on extremely imbalanced data sets. In future work, we will explore the effectiveness of our proposed loss functions on different network structures like DBN and CNN.

REFERENCES

[1] N.V. Chawla, N. Japkowicz and A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM SIGKDD Explorations Newsletter vol.6 (1), pp.1-6, 2004.

[2] J. Wu, S.Pan, X. Zhu, Z. Cai, "Boosting For Multi-Graph Classification," IEEE Trans. Cybernetics, vol.45(3), pp.430-443, 2015.

[3] J. Wu, X. Zhu, C. Zhang, P.S. Yu, "Bag Constrained Structure Pattern Mining for Multi-Graph Classification," IEEE Trans. Knowl. Data Eng, vol.26(10), pp.2382-2396, 2014.

[4] H. He and X. Shen, "A Ranked Subspace Learning Method for Gene Expression Data Classification," IJCAI2007, pp.358-364.

[5] J. C. Candy and G. C. Temes, "Oversampling delta-sigma data converters: theory, and simulation," University of Texas Press, 1962.

[6] H. Li, J. Li, P. C. Chang, and J. Sun, "Parametric prediction on default risk of chinese listed tourism companies by using random oversampling, and locally linear embeddings on imbalanced samples," International Journal of Hospitality Management, vol.35, pp.141–151, 2013.

[7] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," J. Artificial Intelligence Research, vol. 16, pp.321-357, 2002.

[8] J. Mathew,M. Luo, C. K. Pang and H. L.Chan , "Kernel-based smote for SVM classification of imbalanced datasets," IECON2015, pp.1127-1132.

[9] B. X. Wang and N. Japkowicz, "Imbalanced Data Set Learning with Synthetic Samples," Proc. IRIS Machine Learning Workshop, 2004.

[10] H. B. He and A. G. Edwardo, "Learning from imbalanced data," IEEE Transactions On Knowledge And Data Engineering, vol.21(9), pp.1263–1284, 2009.

[11] N. Thai-Nghe, Z. Gatner, L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," IJCNN2010, pp.1–8.

[12] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," ICDM1999, pp.155-164.

[13] C. Elkan, "The Foundations of Cost-Sensitive Learning," IJCAI2001, pp.973-978.

[14] M. A. Maloof, "Learning When Data Sets Are Imbalanced and When Costs Are Unequal and Unknown," ICML'03 Workshop on Learning from Imbalanced Data Sets, 2003.

[15] M. Maloof, P. Langley, S. Sage, and T. Binford, "Learning to Detect Rooftops in Aerial Images," Proc. Image Understanding Workshop, pp. 835-845, 1997.

[16] M. Z. Kukar and I. Kononenko, "Cost-Sensitive Learning with Neural Networks,"ECAI 1998, pp.445-449.

[17] Z. H. Zhou and X.Y. Liu, "Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem,"IEEE Trans. Knowledge and Data Eng, vol. 18 (1), pp.63-77, 2006.

[18] C. H. Tsai, L. C. Chang and H. C. Chiang, " Forecasting of ozone episode days by cost-sensitive neural network methods," Science of the Total Environment, vol.407 (6) , pp.2124–2135, 2009.

[19] M. Lin, K. Tang and X. Yao, " Dynamic sampling approach to training neural networks for multiclass imbalance classification," IEEE TNNLS, vol.24 (4) ,pp. 647–660, 2013.

[20] B. krawczyk and M. Wozniak, "Cost-Sensitive Neural Network with ROC-Based Moving Threshold for Imbalanced Classification," IDEAL2015, pp.45-52.

[21] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," Journal of machine learning research, vol 10, pp.1-40, 2009.

[22] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters,vol 27 (8), pp. 861-874, 2006.

[23] Liu, W., Chan, J., Bailey, J., Leckie, C., & Kotagiri, R. "Mining labelled tensors by discovering both their common and discriminative subspaces." In Proc. of the 2013 SIAM International Conference on Data Mining.

[24] Liu, W., Kan, A., Chan, J., Bailey, J., Leckie, C., Pei, J., & Kotagiri, R. "On compressing weighted time-evolving graphs. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management.