A Weighted Mean Square Error Technique to Train Deep Belief Networks for Imbalanced Data

* Laxmi Sree B R
Dr. G. R. Damodaran College of Science
Avinashi Road, Coimbatore, Tamilnadu
India.
* viporala@yahoo.com

Vijaya M S
PSGR Krishnammal College for Women
Avinashi Road, Coimbatore, Tamilnadu
India
msvijaya@psgrkc.ac.in

***Abstract* - In spite of the popularity and success rates of the Deep learning algorithms in solving complex non-linear problems, it can be observed that the imbalanced dataset contributes to the misclassification rate of the models. Studies at present merely focus on the problem with imbalanced dataset. In this paper, we propose Weighted Mean Square Error (WMSE) to handle the imbalanced dataset problem while training the Deep Belief Networks. This error metrics help in reducing the dominance of the majority classes' influence on building the classification model. The measure is evaluated against imbalanced subset of benchmark datasets MNIST (Appendix-I) and CIFAR-100 (Appendix-II); and with a Tamil phoneme dataset 'Kazhangiyam' built in our earlier work and found to build better classification models for Tamil phoneme recognition problem.**

Keywords - Imbalanced dataset; Deep Belief Networks; Tamil Phoneme Recognition; Mean Square Error; Multi-class problem.

I. INTRODUCTION

In the era of big data, it can be noticed that the world is filled with a huge set of problems with imbalanced dataset. Imbalanced dataset is a dataset where, the number of instances belonging to each class is highly skewed. The variability of the samples in each class – the class imbalance, strongly influences the accuracy of the model built to represent any problem. When a model of imbalanced dataset is observed, it can be noticed that the accuracy of the model is heavily influenced by the rare classes (classes with lesser number of samples), contributing more to misclassification error rates. This is because of the fact that most standard measures available in the literature treat all classes equally.

Machine learning is a powerful learning mechanism which learns to self-build the model from the dataset available. One of the main challenges of these learning mechanisms is learning from rarities. For example, in a clinical diagnostic data, we can find only a few autism samples. Performing data mining needs a huge set of data, which will be lacking in rare cases.

The data imbalance is a serious problem which needs more focus to improve the model accuracy. This problem can be dealt either through sampling techniques or cost sensitive learning. Sampling is a pre-processing phase that transfers the imbalanced dataset to a balance set before the training phase. A simplest sampling method is Random oversampling which randomly duplicates the samples in the minority class to increase the samples (Candy et al. 1992; Hui et al., 2013) of the actual dataset. Under-sampling, in contrast to oversampling eliminates samples from the majority class to help balance the class size. These easy implementable sampling techniques lead to model overfitting in case of random oversampling and loss of some useful information in case of under-sampling procedure. To overcome the loss of information, repetitive under-sampling techniques are proposed in Van Hulse, (2009). Ensemble of models is generated where each model is trained on different under-sampled subset of the training data. This helps reduce the loss of information and improves the performance of the model with better accuracy. Synthetic minority over-sampling technique (SMOTE) is a popular method to handle the problem of data imbalance, Chawla Nitesh V., et al. (2002). This approach uses both minority over-sampling and majority under-sampling to improve the sensitivity of the classifier. SMOTE have proven its efficiency by improving the classification accuracy for minority classes for various machine learning algorithms, imbalance ratio and datasets, but still faces problem of over generalization and variance, Mathew Josey, et al. (2015).

Cost sensitive learning method, on the other hand deal imbalanced dataset in algorithm perspective, based on the cost associated with the misclassified sample, Thai-Nghe (2010). These methods treat the correct classifications of both majority samples and minority samples equally, but in case of misclassifications, wrong classification of minority samples take up most cost compared to the one for majority samples. Thus, the objective function for these problems is to minimize the overall cost incurred by the misclassification of minority samples, Domingos, Pedro, (1999). Learning Classifier Systems, a reinforcement learning scheme proves its ability to build rule sets using smaller disjuncts, Orriols-Puig (2009). It estimates the imbalance ratio and self adapts to evolve itself for infrequent cases.

Generally, the classification problems in the real-world fall under either binary classification or multi-class classification. Most studies on imbalanced dataset concentrate on binary classification (Orriols-Puig, 2008, Wang, Shoujin, et al., 2016). Here, we deal with multi-class classification problem to classify phonemes in continuous Tamil speech which is highly imbalanced. In our earlier work on phoneme recognition, we have built various DBN models trained using back-propagation to classify the phonemes of Tamil continuous speech. In all those training process the standard Mean Square Error (MSE) was used as loss function in the back-propagation training process. The F1-scores of the models is listed out in Table I.

TABLE I. PERFORMANCE COMPARISON OF THE DBN MODELS WITH MSE BACK-PROPAGATION FOR VEHICLE, IMNIST AND KAZHANGIYAM PROBLEMS IN TERMS OF F1-SCORE AND AVERAGE AUC

Method	**F1-score (For Imbalanced Set)**			**F1-score (For Balanced Set)**		
	Vehicle	**IMNIST**	**Kazhangiyam**	**Vehicle**	**IMNIST**	**Kazhangiyam**
DBN (MSE)	0.6147	0.7312	0.2743	0.8582	0.8751	0.6686
PSO-DBN (MSE)	0.6582	0.8138	0.6473	0.8999	0.8419	0.8617
TPSO-DBN (MSE)	0.7605	0.8682	0.7903	0.8851	0.9342	0.8892

It can be observed that greater F1-scores are produced for models trained with balanced datasets, which shows MSE works fine for the problems with balanced dataset as it treats all the classes equally. But in case of imbalanced data the contribution of majority class highly influences on the loss function than the minority class leading to model accuracy driving towards the majority class, failing to capture the details of the minority class. In this paper, we introduce Weighted Mean Square Error (WMSE), analyze its importance on training the Deep Belief Networks during back-propagation training phase and during the process of pre-training DBNs using variants of Particle Swarm Optimization (PSO). Tamil phoneme dataset “Kazhangiyam”, built in our earlier work, Laxmi Sree et al. (2016) is used for the study in addition to the subsets of dataset from CIFAR-100 and MNIST databases to verify the influence of the proposed error to prove WMSE.

II. DEALING WITH DATA IMBALANCE

The problem is formulated to handle imbalanced dataset using a suitable loss function while pre-training and training the Deep Belief Network (DBN). A popular means to initialize the connection weights and bias parameters of the DBN is through contrastive divergence. The iterative training then proceeds with the back-propagation of loss function to fine-tune the parameters to build more accurate classification model. In our earlier works, we have studied the advantages of replacing the contrastive divergence initialization phase with PSO based pre-training phase to optimally initialize the DBN parameters in a faster and consistent way. We propose a loss function to handle imbalanced multi-class dataset, following the standard MSE loss function.

A. Mean Squared Error (MSE) Loss

This loss function is expressed as a squared difference between the predicted output and the expected output and is given as follows:

$$L = \frac{1}{M}\sum_{i=1}^{M}\sum_{n}\frac{(y_n^{(i)}-o_n^{(i)})^2}{N} \qquad (1)$$

where, M is the number samples in the training dataset, y_n^((i)) and o_n^((i)), denotes the predicted and expected output, and N, denotes the number of classes.

B. Weighted Mean Square Error (WMSE) Loss

The MSE loss function handles equally the error raised by the samples of all the classes, this happens to be true for problems with balanced dataset. But when it comes to the imbalanced dataset, the model being built is already molded towards the majority classes; the error caused by misclassification of a sample in majority classes is evaluated more accurately in MSE than for the one in minority class. But in the proposed loss function, the error raised by the members of different class is handled differently by a class influence term, β_n which is multiplied with the error of each member in n^th class. The WMSE loss is given as follows,

$$L = \frac{1}{M}\sum_{i=1}^{M}\sum_{n}\frac{(y_n^{(i)}-o_n^{(i)})^2}{N}\,\beta_n \qquad (2)$$

where β_n is the ratio between the number of samples in the n^th class to the number of samples in the training dataset, and it given by:

$$\beta_n = \frac{C_n}{M} \qquad (3)$$

where C_n is the number of instances in class n of the training dataset. This class relevance is greater for classes with more number of samples in the training dataset than classes with mere samples, thus the classes with greater influence value contributes more to error than the classes with lesser influence.

C. WMSE Loss Back-Propagation

Supervised learning using back-propagation technique is used for training the DBN. The objective function of this back-propagation training is to minimize the WMSE loss function. Back-propagation technique evaluates the error of the model at the output layer and propagates its deviation from the expected output to the preceding layers in the form of gradients at each neuron and fine-tunes the DBN. The gradients are evaluated as the derivative of the WMSE loss function. The gradients at the output layer L is thus given by:

$$\delta_j^{(L)} = \beta_j\, y_j^{(L)}(1 - y_j^{(L)})(y_j^{(L)} - o_j) \tag{4}$$

Further, the back-propagation of error for the neurons in hidden layers $l = L - 1, L - 2, \ldots .2$ is given as usual by:

$$\delta_j^{(l)} = y_j^{(l)}(1 - y_j^{(l)}) \sum_{q=1}^{m} w_{jq}^{(l+1)}\, \delta_q^{(l+1)} \tag{5}$$

where, $y_j^{(l)}$ denotes the j^{th} neuron in l^{th} layer, $w_{jq}^{(l(+1)}$ is the connection weight between j^{th} neuron in l^{th} layer and q^{th} neuron in $(l+1)^{th}$ layer. The weights and biases of the DBN layers are learnt with a learning rate γ, updated as before, and are given as follows:

$$\Delta w_{jq}^{(l)} = -\gamma y_j^{(l)} \delta_q^{(l+1)} \tag{6}$$

And

$$\Delta b_j^{(l)} = -\gamma \delta_q^{(l+1)} \tag{7}$$

III. LEARNING METHODOLOGY

In our work, we use DBNs for building the classification problems under consideration. DBNs are much dynamic structures that are capable of learning itself from the available non-linear data. These learners are good feature extractors and are capable of handling imbalance and high dimensional data. DBN is a graphical structure which is built as a collection of Restricted Boltzmann Machines (RBMs) layers. Each RBM has two layers, a visible and an invisible layer, the visible layer of the first RBM acts as the input layer of the DBN. Generally, DBNs are trained layer by layer using the contrastive divergence technique, in which the output of each RBM acts as the input to the next RBM. The DBN is then fine-tuned by back-propagating the WMSE cost using back-propagation algorithm. The entire methodology is portrayed in algorithm 1.

Algorithm 1: Training DBN using WMSE back-propagation

```
Input: Training dataset
Output: DBN model
-1. Define the number of layers L, number of neurons in each layer $\eta_l$ of the DBN
-2. Initialize the DBN parameters biases, b and connection weights, w with random initial values
-3. Train DBN using contrastive divergence
-4. For i = 1 to maxIter
-5. Calculate the error at the output layer using equation 4
-6. For each layer l=L-1 to 1
-6.1 Back-propagate the error in the previous layer l+1 to l using equation 5
-6.2 Calculate the change in biases and weight using equations 6 and 7 and update the biases and weights of the layer
-7. End for
-8. End for
```

The second model is built using PSO-DBN, proposed in our earlier work which replaces the contrastive divergence based pre-training with PSO. The proposed algorithm for building PSO-DBN that uses WMSE loss function while pre-training and training is portrayed in algorithm 2. The pre-training phase starts by initializing the position and velocity of the individuals in the population. In each generation of PSO, once the new position and velocity of the individuals are evaluated, the cost of the individuals in the population is estimated using WMSE to keep track of the individual's local best and the global best of the entire population. Following the pre-training phase, the training phase starts by transforming the global best of PSO to DBN and further fine-tuning itself using WMSE back-propagation. The third model is built using TPSO-DBN with WMSE cost in both pre-training and training phases. TPSO (Temperature controlled PSO), introduced in our earlier work uses a temperature term to control the velocity of the individuals. The entire flow is listed in algorithm 3. The methodology is similar to PSO-DBN except, a temperature term controls the velocity the individuals.

Algorithm 2: Training PSO-DBN using WMSE back-propagation

```
Input: Training dataset
Output: PSO-DBN model
-1. Define the number of layers L, number of neurons in each layer $\eta_l$ of the DBN
-2. Define the population P of size s for PSO based pre-training phase
-3. Initialize the position of individuals $x_i(1)$ in the population to represent DBN parameters biases, b and connection weights, w with random initial values
-4. Define the velocity of individuals $v_i(1)$ as zero
-5. For t = 1 to maxGen
-5.1 For each individual i in the population P
-5.1.1 Calculate the new velocity $v_i(t+1) = \omega v_i(t) + c_1 r_1(p_i(t) - x_i(t)) + c_2 r_2 (p_g(t) - x_i(t))$
-5.1.2 Calculate the new position $x_i(t+1) = x_i(t) + v_i(t+1)$
-5.1.3 Calculate WMSE cost using equation 2
-5.1.4 Update personal best cost and position $(p_i)$
-5.1.5 Update global best cost and position $(p_g)$
-5.1.6 Update inertia $(\omega)$
```

```
-5.2 End For
-6. End For
-7. Build PSO-DBN using the global best individual $(p_g)$ of PSO
-8. For i = 1 to maxIter
-9. Calculate the error at the output layer using equation 4
-10. For each layer l=L-1 to 1
-10.1 Back-propagate the error in the previous layer l+1 to l using equation 5
-10.2 Calculate the change in biases and weight using equations 6 and 7 and update the biases and weights of the layer
-11. End for
-12. End for
```

```
Algorithm 3: Training TPSO-DBN using WMSE back-propagation
Input: Training dataset
Output: TPSO-DBN model
-1. Define the number of layers L, number of neurons in each layer $\eta_l$ of the DBN
-2. Define the population P of size s for TPSO based pre-training phase
-3. Define the temperature function h
-4. Initialize the position of individuals $x_i(1)$ in the population to represent DBN parameters biases, b and connection weights, w with random initial values
-5. Define the velocity of individuals $v_i(1)$ as zero
-6. For t = 1 to maxGen
-6.1 For each individual i in the population P
-6.1.1 Calculate the new velocity $v_i(t+1) = \omega v_i(t) + h(t)v_i(t) + c_1r_1(p_i(t) - x_i(t)) + c_2r_2\left(p_g(t) - x_i(t)\right)$
-6.1.2 Calculate the new position $x_i(t+1) = x_i(t) + v_i(t+1)$
-6.1.3 Calculate WMSE cost using equation 2
-6.1.4 Update personal best cost and position $(p_i)$
-6.1.5 Update global best cost and position $(p_g)$
-6.1.6 Update inertia $(\omega)$
-6.2 End For
-7. End For
-8. Build PSO-DBN using the global best individual $(p_g)$ of PSO
-9. For i = 1 to maxIter
-10. Calculate the error at the output layer using equation 4
-11. For each layer l=L-1 to 1
-11.1 Back-propagate the error in the previous layer l+1 to l using equation 5
-11.2 Calculate the change in biases and weight using equations 6 and 7 and update the biases and weights of the layer
-12. End for
-13. End for
```

IV. EXPERIMENTS AND RESULTS

Three different datasets have been used in our experiments to validate the performance of the proposed loss function. The first dataset is built as a subset of MNIST database of handwritten digits comprising about 60,000 samples in train set, with 10,000 examples per class and 10,000 samples in test set, with about 1000 samples per class Deng, Li. (2012). The subset of MNIST used here, coined as Imbalanced MNIST (IMNIST) is complied as a dataset with 25% data imbalance between majority (classes 1 to 5) and minority classes (classes 6 to 10).

The second dataset is a subset of CIFAR-100 database of coloured images of 100 classes, Krizhevsky (2009). The subset hereby, named Vehicles dataset comprises of images of 10 classes namely, bicycle, bus, motorcycle, pickup truck, train, lawn-mower, rocket, street car, tank and tractor. The first five classes in the list are majority classes with 500 training images in each class and the next five are considered as minority classes with 125 training images in each class. Similarly, the test dataset is built with 100 images in majority classes and 25 images in minority classes summing up to 625 images in test dataset.

The third dataset Kazhangiyam is a labelled dataset of Tamil phonemes extracted from continuous Tamil speech of 39 speakers. The dataset used here, built in our earlier work is an imbalanced dataset of 39 classes with an aggregated set of 187452 instances. The segmentation of phonetic segments from the continuous speech was done using a graph-cut based segmentation technique, Laxmi Sree (2016).

The proposed loss function is experimented with the three different models DBN with WMSE, PSO-DBN with WMSE and TPSO-DBN with WMSE explain in algorithm 1, 2 and 3 respectively. The results of our experiments are compared with the results of the most competent learning algorithms C4.5 and IBk. C4.5 is an algorithm that builds decision tree that has its root in ID3 algorithm, Quinlan (1995). IBk is an instance based learning algorithm based on the nearest neighbor algorithm, David (1991).

Deciding the parameters like number of layers in DBN and number of neurons in each layer is a tedious task. The parameters are chosen here by trial and error method. Table II shows the configuration of DBN used for various datasets under consideration.

TABLE II. DBN CONFIGURATION FOR VARIOUS DATASETS

Dataset	Number of Hidden Layers	Number of Neurons in Each Hidden Layer
Vehicle	5	2500, 1500, 700, 300, 100
IMNIST	3	400, 200, 100
Kazhangiyam	5	120, 140,120, 100, 80

The performance of various methods are analyzed using the standard Root Mean Square Error (RMSE) and Phoneme Error Rate (PER) for the Kazhangiyam dataset, listed in Table III.

TABLE III. PERFORMANCE ANALYSIS OF VARIOUS DBNS BUILT USING MSE/WMSE COST FUNCTION

Method	Training Phase		Testing Phase	
	RMSE	PER	RMSE	PER
DBN (MSE)	0.01523	14.62	0.01528	14.97
DBN (WMSE)	0.01438	11.72	0.01488	13.33
PSO-DBN (MSE)	0.01549	12.2	0.01620	12.98
PSO-DBN (WMSE)	0.01361	10.83	0.01496	10.45
TPSO-DBN (MSE)	0.0094	10.5	0.0105	10.81
TPSO-DBN (WMSE)	0.00815	7.26	0.0089	8.31

It can be observed that the RMSE of the models DBN, PSO-DBN and TPSO-DBN trained using WMSE shows an improvement over the models trained using the MSE cost function with an average of 2% reduction in the phoneme error rate for all the DBN models. The observed PERs are 11.72%, 10.83% and 7.26% for DBN, PSO-DBN and TPSO-DBN with WMSE cost function and 14.62%, 12.2% and 10.5% for the respective models with MSE cost function for training data. Further, the best PER is observed for TPSO-DBN for the test data and is recorded as 8.31%.

The DBN models are further analyzed against the performance measures F1-score and AUC. The observed readings are recorded in the Table IV. The DBN models are compared with the performance of the well known algorithms C4.5 and IBk, capable of handling imbalanced data and are experimented for the datasets vehicle, IMNIST and Kazhangiyam.

TABLE IV. COMPARISON OF THE DBN MODELS WITH WMSE BACK-PROPAGATION WITH THE STANDARD C4.5 AND IBK ALGORITHM FOR VARIOUS PROBLEMS IN TERMS OF F1-SCORE AND AVERAGE AUC

Method	F1-score			Average AUC		
	Vehicle	IMNIST	Kazhangiyam	Vehicle	IMNIST	Kazhangiyam
C4.5	0.7391	0.8921	0.5504	0.7903	0.8379	0.7086
IBk	0.7523	0.7289	0.5893	0.7634	0.7932	0.6223
DBN (WMSE)	0.7271	0.8243	0.4831	0.7691	0.8121	0.7301
PSO-DBN (WMSE)	0.8045	0.9116	0.7681	0.7947	0.8743	0.8747
TPSO-DBN (WMSE)	0.8629	0.9313	0.8573	0.8216	0.9411	0.9403

The results in terms of F1-score and AUC shows a good improvement for models implemented with WMSE (Table IV) compared to the models implemented with MSE (Table I). It can also be observed that in most cases the scores are comparable or better than the results of C4.5 and IBk algorithms. The best F1-score for vehicle dataset is observed as 0.8629 for TPSO-DBN(WMSE), followed by 0.8045 for PSO-DBN and 0.7605 for TPSO-DBN(MSE). The best three F1-scores for IMNIST dataset is observed as 0.9313, 0.9116 and 0.8921 for TPSO-DBN(WMSE), PSO-DBN(WMSE) and C4.5 respectively. Similarly, the top three F1-scores for Kazhangiyam dataset is observed as 0.8573, 0.7903 and 0.7681 for TPSO-DBN(WMSE), PSO-DBN(WMSE) and DBN(WMSE) respectively.

The average AUC readings in Table IV shows better classification accuracy for WMSE based DBN models to its MSE counterparts. The TPSO-DBN (WMSE) version is observed to outperform the other models under examination. The highest average AUC for the datasets vehicle, IMNIST and Kazhangiyam is observed for TPSO-DBN (WMSE) with readings 0.8216, 0.9411 and 0.9403 respectively. The lowest average AUC observed is 0.6582 (DBN(MSE)) , 0.775 (DBN(MSE)) and 0.6223 (IBk) for datasets vehicle, IMNIST and Kazhangiyam respectively.

From the above results, it is clear that treating classes with varied number of samples differently has a greater influence in the model building process. It can be inferred from the results that the proposed weighed mean square error as a loss/cost function during the pre-training and training phases has high influence in building better models than using a measure like mean square error that has been designed for dataset with balanced classes. The ROC curves for various minority classes /uu/, /au/, /sh/, /nj/, /zh/ and /j/ in the Kazhangiyam dataset is visualized for various DBN models in Fig. 1 below.

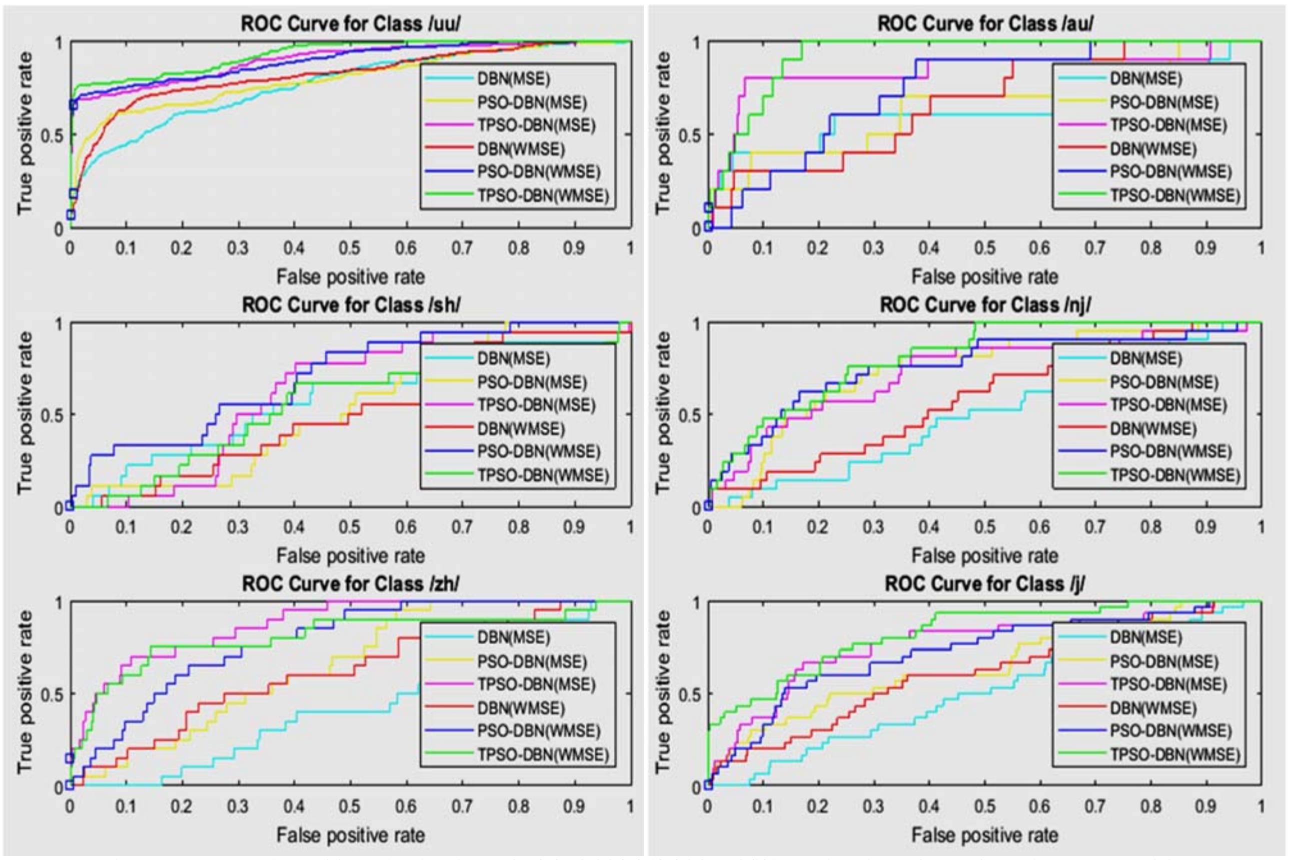

Fig. 1. ROC comparison of few minority classes /uu/, /au/, /sh/, /nj/, /zh/, and /j/ in Kazhangiyam dataset for various DBN models.

The ROC curves of most of the models with WMSE as loss function is observed to contribute greater AUC when compared to their counterparts with MSE loss function confirming the importance of the class influence term in the WMSE measure.

V. CONCLUSIONS

Although a wide range of machine learning algorithms are in existence that are capable of self-learning from the datasets, comparably least work can be seen to address the problem of data imbalance. Most of the loss function that is used in the machine learning algorithms treat all classes in the dataset equally. But in most real-world datasets, the classes are imbalanced. To handle the imbalanced data, weighted mean square error loss function is proposed and used in the pre-training and training phases of various DBN learning process. It is found that the models built using WMSE loss function outperforms the models built using MSE for the imbalanced datasets vehicle, IMNIST and Kazhangiyam. The performance of the models is found to classify better to the models built using the standard algorithms C4.5 and IBk for our imbalanced datasets. In future work, the effect of the proposed loss function with other learning algorithms will be experimented and explored.

REFERENCES

[1] Aha, David W., Dennis Kibler, & Marc K. Albert. (1991). Instance-based Learning Algorithms. Machine learning 6(1): 37-66.

[2] Candy J. C. & Temes G. C (1992). Oversampling Delta-Sigma Data Converters: Theory, and Simulation. IEEE Press.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research, 16: 321-357.

[4] Deng, Li. (2012). The MNIST Database of Handwritten Digit Images For Machine Learning Research [best of the web]. IEEE Signal Processing Magazine 29(6): 141-142.

[5] Domingos, P. (1999, August). Metacost: A General Method for Making Classifiers Cost-Sensitive. In Proceedings of the fifth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining, Pp. 155-164.

[6] J.R. Quinlan. (1995). C.4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California.

[7] Krizhevsky, Alex, & Geoffrey Hinton, (2009). Learning multiple layers of features from tiny images. 1(4). Technical report, University of Toronto.

[8] Laxmi Sree, B.R., & Suguna, M. (2016 June). AAYUDHA: A Tool for Automatic Segmentation and Labelling of Continuous Tamil Speech. International Journal of Computer Applications, 143(1), ISSN: 0975 – 8887.

[9] Laxmi Sree, B.R., & Vijaya, M.S. (2016). Graph Cut Based Segmentation Method for Tamil Continuous Speech. Annual Convention of the Computer Society of India. Springer, Singapore.

[10] Li, H., Li, J., Chang, P. C., & Sun, J. (2013). Parametric prediction on Default Risk of Chinese Listed Tourism Companies by Using Random Oversampling, Isomap, and Locally Linear Embeddings on Imbalanced Samples. International Journal of Hospitality Management, 35: 141-151.

[11] Mathew, J., Luo, M., Pang, C. K., & Chan, H. L. (2015, November). Kernel-based SMOTE for SVM Classification of Imbalanced Datasets. In Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE, Pp. 001127-001132.

[12] Orriols-Puig, A., & Bernadó-Mansilla, E. (2009). Evolutionary Rule-Based Systems for Imbalanced Datasets. Soft Computing, 13(3): 213.

[13] Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010, July). Cost-Sensitive Learning Methods for Imbalanced Data. In Neural Networks (IJCNN), The 2010 International Joint Conference on, Pp. 1-8.

[14] Van Hulse, J., Khoshgoftaar, T. M., & Napolitano, A. (2009, August). An empirical Comparison of Repetitive Under-Sampling Techniques. In Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on, Pp. 29-34.

[15] Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J. (2016, July). Training Deep Neural Networks on Imbalanced Datasets. In Neural Networks (IJCNN), 2016 International Joint Conference on, Pp. 4368-4374.

Appendix-I: MNIST Database

See: https://en.wikipedia.org/wiki/MNIST_database

The **MNIST database** (**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

Sample images from MNIST test dataset.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. There have been a number of scientific papers on attempts to achieve the lowest error rate; one paper, using a hierarchical system of convolutional neural networks, manages to get an error rate on the MNIST database of 0.23%. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support vector machine to get an error rate of 0.8%. An **E**xtended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters.

Appendix-II: CIFAR-10

See: https://en.wikipedia.org/wiki/CIFAR-10

The **CIFAR-10** dataset (**C**anadian **I**nstitute **F**or **A**dvanced **R**esearch) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.

Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works. Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10.

CIFAR-10 is a labeled subset of the 80 million tiny images dataset. When the dataset was created, students were paid to label all of the images.