Towards Effective Classification of Imbalanced Data with Convolutional Neural Networks

Vidwath Raj(✉), Sven Magg, and Stefan Wermter

Knowledge Technology, Department of Informatics, University of Hamburg, Hamburg, Germany
vidwath0011@gmail.com

Abstract. Class imbalance in machine learning is a problem often found with real-world data, where data from one class clearly dominates the dataset. Most neural network classifiers fail to learn to classify such datasets correctly if class-to-class separability is poor due to a strong bias towards the majority class. In this paper we present an algorithmic solution, integrating different methods into a novel approach using a class-to-class separability score, to increase performance on poorly separable, imbalanced datasets using Cost Sensitive Neural Networks. We compare different cost functions and methods that can be used for training Convolutional Neural Networks on a highly imbalanced dataset of multi-channel time series data. Results show that, despite being imbalanced and poorly separable, performance metrics such as G-Mean as high as 92.8 % could be reached by using cost sensitive Convolutional Neural Networks to detect patterns and correctly classify time series from 3 different datasets.

1 Introduction

In supervised classification tasks, effective learning happens when there are sufficient examples for all the classes and class-to-class (C2C) separability is sufficiently large. However, real world datasets are often imbalanced and have poor C2C separability. A dataset is said to be imbalanced when a certain class is overrepresented compared to other classes in that dataset. In binary classification tasks, the class with too many examples is often referred to as the *majority class*, the other as the *minority class* respectively. Machine Learning algorithms performing classification on such datasets face the so-called *'class imbalance problem'*, where learning is not as effective as it is with a balanced dataset [6,10,13], since it poses a bias in learning towards the majority class.

On the one hand, many of the real world datasets are imbalanced and on the other hand, most existing classification approaches assume that the underlying training set is evenly distributed. Furthermore, in many scenarios it is undesirable or dangerous to misclassify an example from a minority class. For example, in a continuous surveillance task, suspicious activity may occur as a rare event which is undesirable to go unnoticed by the monitoring system. In medical applications, the cost of erroneously classifying a sick person as healthy

F. Schwenker et al. (Eds.): ANNPR 2016, LNAI 9896, pp. 150–162, 2016.
DOI: 10.1007/978-3-319-46182-3_13

can have larger risk (cost) than wrongly classifying a healthy person as sick. In these cases it is crucial for classification algorithms to have a higher identification rate for rare events, that means it is critical to not misclassify any minority examples while it is acceptable to misclassify few majority examples.

An extreme example for the imbalance problem would be a dataset where the area of the majority class overlaps that of the minority class completely and the overlapping region contains as many (or more) majority examples. Since the goal of learning is to minimise the overall cost of the cost function, such minimization can be obtained in this case by classifying all points to the majority class. Any other separation would result in misclassifying more data points of the majority class than correctly classifying data points of the minority class. This happens because a standard cost function, i.e. the cost due to erroneous classification, treats all individual errors as equally important.

Different methods such as Backpropagation Neural Networks (BPNN), Radial Basis Function (RBF) and Fuzzy ARTMAP when exposed to unbalanced datasets with different noise levels have this type of problem and it was shown that performance on imbalanced data greatly depends on how well the classes are separated [14]. When the classes are separated well enough, BPNN and Fuzzy ARTMAP could learn features well compared to RBF. Lately, Convolutional Neural Networks have gained a lot of interest in the pattern recognition community but are also subject to this problem as other neural network approaches using supervised learning [9].

Several possible solutions have been suggested already in the literature to tackle the class imbalance problem. There are two main approaches: Changing the training data or adjusting the algorithms. Training data can be changed by over-sampling the minority class (e.g. SMOTE [7]) or under-sampling the majority class (e.g. WWE [20]). Algorithmic approaches change the algorithm so that it favours the minority class. Examples are Threshold Moving [22], Snowball Learning for neural networks [19], ensemble methods [18], or other cost-sensitive learning methods for neural networks where different costs are associated with each class (e.g. [2]). For a comprehensive overview on different methods and an empirical study on cost-sensitive neural networks comparing resampling techniques to threshold-moving and hybrid ensembles, please see the work of Zhou et al. [22] where they report that Threshold Moving outperformed resampling. They also show results and discuss the effectiveness of the approaches for multi-class problems. Also hybrid methods have been used lately for neural networks and were shown to be effective [1,4]. Therefore, there is still a lack of addressing the imbalance problem, which we examine in this paper based on Convolutional Neural Networks.

2 Methods

The dataset mainly used for this work is highly imbalanced, which makes the use of resampling difficult since too many samples would be lost for training or have to be created artificially, leading to potential overfitting. We therefore

investigated a cost-sensitive algorithmic approach with adaptable costs which is inspired by the recent works of Kahn et al. [9] and Castro et al. [5] for a binary class problem. As performance measure we use accuracy and the geometric mean (G-Mean)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}; \quad \text{G-Mean} = \sqrt{TPR * FPR} \tag{1}$$

with TPR the true positive rate (FPR the false positive rate, respectively). Accuracy is the standard performance measure for classification, but leads to the class imbalance problem if optimised for, since all errors are treated equally. We therefore also use the G-Mean, which takes the performance of each class individually into account and thus leads to performance balancing between the classes.

2.1 Global Mean Square Error Separation

In the standard backpropagation algorithm, the weights are updated to minimise the overall error of the training data. However, if a certain class extensively dominates the training dataset and if C2C separability is low, the errors are contributed mainly from the majority class. Global Mean Square Error (GMSE) Separation [5] is an algorithmic method to differentiate the errors for different classes. The Mean Square Error (MSE) cost function minimises the error between predicted output from the network and the actual output. With GMSE, the cost to be minimised is the weighted sum of the cost of individual classes.

For a binary classification problem, let T be the training set of size n and T^+ and T^- be the set of positive and negative examples respectively. Let us assume T^+ represents the minority class and w represents the weight parameters of the entire network. The error of the p^{th} training example $\varepsilon_p(w)$ is given by

$$\varepsilon_p(w) = \frac{1}{2}(d_p - y_p)^2 \tag{2}$$

where y_p is the predicted output and d_p is the desired output of the p^{th} training example. The global mean squared error is given by

$$E(w) = \frac{1}{n}\sum_{p=1}^{n} \varepsilon_p(w). \tag{3}$$

By separating the global mean squared error into a weighted sum of errors of each class, we have

$$\varepsilon_p^{(k)}(w) = \frac{1}{2}\left(d_p^{(k)} - y_p^{(k)}\right)^2 \text{ for } k = \{+, -\}. \tag{4}$$

Therefore the mean squared error of each class is given by

$$E^{(k)}(w) = \frac{1}{n^{(k)}}\sum_{p=1}^{n^{(k)}} \varepsilon_p^{(k)}(w). \tag{5}$$

Optimization is achieved by minimizing the sum of errors for all k.

Weight Update: In standard backpropagation the weights are updated as follows

$$w^{(m+1)} = w^{(m)} - \eta \nabla E(w^{(m)}) \tag{6}$$

where m is the epoch, η is the learning rate, and $\nabla E(w^{(m)})$ is the gradient from the m^{th} epoch. To separate the GMSE, gradients are calculated separately and a weighted sum is calculated as follows

$$\nabla E(w) = \frac{1}{\lambda^+} \nabla E^+(w) + \frac{1}{\lambda^-} \nabla E^-(w). \tag{7}$$

λ^+ and λ^- can be any positive constant to penalise the classes for misclassification [5]. This also gives flexibility to achieve desired specificity and sensitivity. In standard backpropagation, both λ^+ and λ^- are equal to 1. Values generally used for λ^+ and λ^- are the sizes of the respective classes n^+ and n^-, or a heuristically determined value suitable for the problem. However, using n^+ and n^- with a very large dataset reduces the learning rate significantly. Since the XING dataset used for training in our experiments had 1 million examples and the ratio between the classes was 114:1, using the class size was not suitable and the ratio of 114 was used for λ^+ (with $\lambda^- = 1$). This means the cost for misclassifying a minority example was 114 times higher than the cost of misclassifying a majority example.

The core idea of this method is to have a weighted sum of the gradients of each class so as to achieve the desired specificity and sensitivity. In other words, each class has a different learning rate since the rate at which the weights are updated is different. One problem is that many libraries for GPU programming do not give easy access to the error gradients, but similar results can be achieved by having different costs of misclassification for each class, thus rewriting the error function as

$$E = \frac{1}{n} \sum_{p=1}^{n} (d_p - \kappa * y_p)^2 \quad \text{where} \quad \kappa = \begin{cases} 1, & \text{if } p \in \text{majority} \\ \dfrac{max(n^+, n^-)}{min(n^+, n^-)}, & \text{otherwise} \end{cases} \tag{8}$$

where $\kappa = 1$ if p is from the majority class and the ratio of class sizes if not. n^+ and n^- are again the size of T^+ and T^- respectively. Our implementation (see Algorithm 1) is similar to Kukar et al. [11] and simplifies the implementation when using the python GPU library Theano[1] [3]. It can be shown mathematically that the implementation has the same effect of GMSE separation [16].

One of the drawbacks of this method is to find the optimum value for κ. There could exist a dataset dependent value κ^* with which better performance could be achieved. We now make κ an adaptable parameter and show empirically that it can achieve better results compared to a static value.

2.2 Learnable κ in Cost Function

In the previous approach we used a fixed κ to penalise more for misclassifying minority examples. Since the overall goal is to obtain a maximum G-Mean, κ can

[1] http://deeplearning.net/software/theano/.

Algorithm 1. Optimised Implementation of Global Mean Square Error Separation

```
Input: Training data
Output: Learned weights
set maxepoch;
initialise weights of the network randomly;
while epoch not equal to maxepoch do
    for minibatch do
        Forward pass;
        Calculate error as per Eq. 8;
        Calculate gradients for the error;
        Update weights;
    end
end
```

be optimised or learned to maximise G-Mean for a specific dataset. The ideas and methods are originally motivated by the work of Khan et al. [9].

$$E = \frac{1}{n}\sum_{p=1}^{n}(d_p - \kappa * y_p)^2 \text{ where } \kappa = \begin{cases} 1, & \text{if } p \in \text{majority} \\ \kappa^*, & \text{otherwise} \end{cases} \tag{9}$$

κ is initialised to 1 and is updated to optimise G-Mean or the combination of both G-Mean and accuracy. The effect of optimising different metrics are discussed in the subsequent sections. In stochastic gradient learning, the weight parameters are updated for each minibatch. But κ is updated at the end of each epoch and has its own learning rate. The implementation can be seen in Algorithm 2.

***κ* Optimization:** The goal of the learning is to jointly learn weight parameters as well as κ alternatively, keeping one of these parameters constant and minimizing the cost with respect to the other [9]. Some modifications on optimising κ were made compared to the original approach. In our work, κ is optimised as follows

$$\kappa^* = argmin F(\kappa); \quad F(\kappa) = ||T - \kappa||^2 \tag{10}$$

and using a gradient decent algorithm

$$\nabla F(\kappa) = \nabla ||T - \kappa||^2 = -(T - \kappa). \tag{11}$$

For T we have used different methods to gain more insights into the effect of optimising for different metrics. H is the maximum cost applicable to the minority class which in this case is the ratio of imbalance:

– Optimising for G-Mean and Accuracy

$$T_1 = H * exp\left(-\frac{GMean}{2}\right) * exp\left(-\frac{Accuracy}{2}\right) \tag{12}$$

Algorithm 2. Learning Optimal Parameters

Input: Training data, Validation Data, maxepoch, Learning Rate for w, learning rate for κ
Output: Learned parameters, w^* and κ^*

```
Initialise Network;
Initialise Network weights randomly;
Initialise κ to 1;
while epoch not equal to maxepoch do
    for minibatch do
        Forward pass;
        Calculate error as per Eq. 9;
        Calculate gradients for the error;
        Update weights;
    end
    Compute gradients for κ using Eq. 10 with T either T1, T2, or T3;
    Update κ;
end
```

– Optimizing only for G-Mean

$$T_2 = H * exp\left(- \frac{GMean}{2} \right) \tag{13}$$

– Optimizing for G-Mean and validation errors (1 - accuracy). The motivation behind this equation is to see if bringing down accuracy would help improve G-Mean.

$$T_3 = H * exp\left(- \frac{GMean}{2} \right) * exp\left(- \frac{(1 - Accuracy)}{2} \right) \tag{14}$$

This explores a κ between 1 and H, which can be set to the ratio from minority to majority class. For brevity, let us call the adaptable κ using Eqs. 12, 13 and 14 as κ_1, κ_2 and κ_3 respectively.

Class Separability: Learning in imbalance depends on how well the classes are separated [14]. However, κ is never affected by C2C separability so far. Thus it is meaningful, to introduce an effect on κ with respect to class-to-class separability. The idea is that when C2C separability is good, i.e. when the classification is easier, errors should cost more and, respectively, errors on hard classification problems should be punished less.

For this, the Silhouette score [17] was used as a C2C separability measure. This technique quantifies how well each data point lies within its own cluster and the value ranges from -1 to $+1$. For individual points, $+1$ means that the point lies perfectly within the own cluster, 0 means the point is on the boundary between both clusters, and -1 means the point is perfectly within the opposite

cluster. The sum over all points gives a measure on how tightly formed and separated the clusters are. Given classes A and B,

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{15}$$

where, $b(i)$ = minimum $d(i, B)$ and $d(i, B)$ is the average dissimilarity of i from Class A to all other objects of Class B ($a(i)$ respectively) [17]. The average of $s(i)$ over all data points is the measure of how well the classes form clusters or how well they can be separated.

Let us denote the Silhouette score as S and with the imbalance ratio $IR = max(n^+, n^-)/min(n^+, n^-)$, H is now given by

$$H_{adjusted} = IR(1 + |S|). \tag{16}$$

This adjusts the maximum reachable cost for the minority class based on its separability. If two classes are well separable, the maximum cost is twice IR. Notice that for perfectly balanced classes, if the classes are clearly separable ($|S| = 1$), we incur twice the cost on one of the classes. This may seem wrong, but we will show this case has no practical effects on learning.

3 Experiments and Results

In this section experiments are reported to show the classification performance with the different methods we have introduced above. We have used three real datasets (XING, Earthquake, ECG) and some specifically selected subsets to enforce specific IR or C2C scores. The *XING dataset* was the main target and includes activity timelines of one million users of the XING social network, tracking 14 different activities. The likelihood of a job change had to be predicted on the basis of the recent past. Since job changes are very rare compared to normal activity, the IR was 114:1. The Silhouette separability score S for XING was calculated on a small random sample of 10,000 examples, since it is a computationally expensive function and a good estimate was sufficient. The mean S for five random samples was -0.184 ± 0.037. The negative sign signifies that many majority examples are within the area of the minority class. This was expected, because a lot of users' activities is similar to the activities of a job changer, except that they did not change their job in the end.

The other datasets are taken from [8], also include timeline data, and have been used mainly for comparison at the end: The *Earthquake dataset* contains hourly averages of sensory readings. The positive instances are major events greater than 5.0 on the Richter scale and not preceded by another major event for at least 512 hours. Negative examples are readings with Richter scale less than 4.0. This dataset has a very low class separability score S of 0.0005 and has an imbalance ratio of 3.95.

The *ECG dataset* contains electrocardiography recordings from two different days on the same patient. The positive instances had an anomalous heartbeat.

The data exhibits linear drift, which in medical terms is called a wandering baseline. Globally, both positive and negative instances are very similar with separability score S being only 0.09. The raw dataset was perfectly balanced, therefore imbalance was created artificially by removing positive examples to reach an imbalance ratio of 20.

The primary set of experiments include a comparison of methods in terms of performance on the XING dataset as well as other datasets. The secondary goal was to analyse the behaviour of different adaptable κ and the effects of adjusting the maximum applicable cost H by incorporating the silhouette score of the corresponding datasets.

A multi-channel CNN with one-dimensional kernels for each feature was used similar to [21]. 90 % of the datasets was used for training and 10 % for validation in each epoch. Since training CNNs takes a long time, all experiments on the XING data were stopped after the 100th epoch since little improvement was found beyond that point. The size of each minibatch was set to 5000 and for training Stochastic Gradient Descent was used due to its faster convergence [12]. Smaller minibatches were not used to make sure a minority example was contained with high chance. Also, $L2$ regularization was used to avoid overfitting [15]. All reported values are means ($\pm$ standard deviation) of five runs for each experiment to account for variability in the random training/validation splits of the datasets.

3.1 Comparison of Methods

Figure 1 shows a typical example of performance over epochs of CNNs with different cost methods. With no cost sensitivity, G-Mean remains 0 due to the class imbalance problem. Further experiments revealed, no matter the number of iterations, there was no further improvement in the G-Mean score. Figure 2 shows how accuracy and G-Mean vary with cost. With just a little loss in overall accuracy, the classifier's performance on both classes together can be increased significantly.

Table 1 summarises the performance of different methods on a smaller random subset of the XING data with 200,000 examples. Using the imbalance ratio as κ shows significant improvement in comparison with having no cost at all. It also shows that learning κ gave better performance in comparison to constant κ (t-test: $p = 0.0192$). The adaptable κ starts by incrementally increasing the cost, and these increments get smaller as the G-Mean increases. Therefore a gradual increase in G-Mean can be noticed and at the end a slight improvement in comparison to a constant κ, thus an improvement by tuning κ to the dataset was possible as hypothesised.

3.2 Closer Look at Adaptable κ

We have proposed three different optimisation approaches, κ_1, κ_2, and κ_3. Figure 3 shows G-Mean and accuracy scores of all three methods over iterations. As can be seen, κ_1 improves the G-Mean score over time while trying to

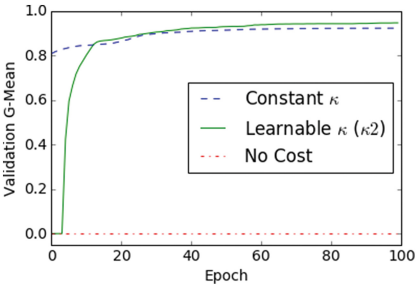

Fig. 1. Results of example runs of different costs functions on G-Mean metric over learning time.

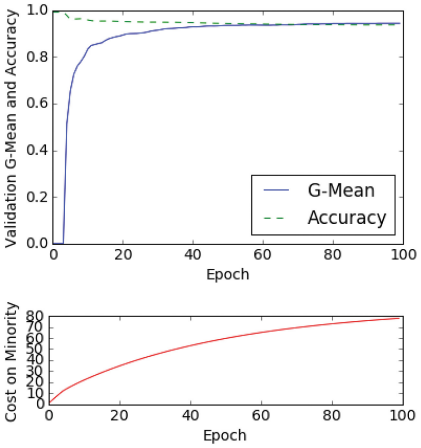

Fig. 2. Example of how Accuracy and G-Mean (a) are affected by cost (κ_2) over epochs (b)

Table 1. Performance comparison of different cost functions.

Method	G-Mean Validation	Accuracy Validation
Const. κ	86.4% ± 2.8%	77.2% ± 5.6%
$\kappa 1$	90.1% ± 1.5%	92.5% ± 1.3%
$\kappa 2$	90.3% ± 1.0%	87.7% ± 1.3%
$\kappa 3$	90.8% ± 1.1%	88.5% ± 1.0%
No Cost	0% ± 0%	99.1% ± 0%

keep a high accuracy. The other two focus on optimisation of G-Mean with κ_3 being more aggressive than κ_2. Nevertheless, in the end, κ_3 achieves a slightly higher score for both performance measures (compare Table 1).

3.3 Combined Effect of Separability and Imbalance

We also introduced the usage of the class separability score S to adjust the maximum applicable cost for minorities. Table 2 compares the performance outcome of using a maximum cost on minority to be H and $H_{adjusted}$. Results clearly show an advantage of having adjusted cost (t-test: p = 0.00018). However, when classes are balanced and fairly separable, imposing greater cost on one of the classes appears improper both theoretically and intuitively. Nevertheless, experiments were conducted with κ_2 to evaluate practical outcomes of such scenario. Table 3 shows the performance of regular learning and cost-sensitive learning on an artificial subset of *XING* with different S and imbalance ratios.

Table 2. Effect of adjusting maximum cost H to impose on minority using C2C separability score compared to using H as imbalance ratio.

	Max Cost on Minority	**G-Mean using Cost Sensitive CNN ($\kappa 2$)**
H	114	$92.8\% \pm 0.24\%$
$H_{adjusted}$	142	$\mathbf{94.1\% \pm 0.25\%}$

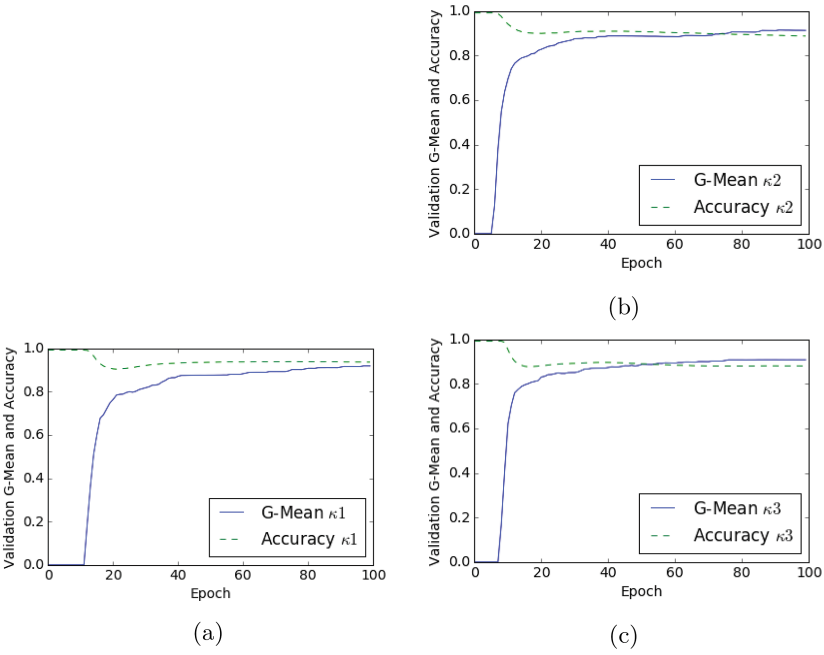

Fig. 3. Rate of change of G-Mean and Accuracy with (a) Learnable $\kappa 1$, (b) Learnable $\kappa 2$ and (c) Learnable $\kappa 3$ in example runs on XING data

These results show again a clear benefit when using κ_2 with adjusting H to the respective class separability when there exists an imbalance in class distribution. Punishing one class in the balanced case has a small but negligible effect compared to regular learning. For the case of imbalance with $|S| = 0.02$ the learning failed to distinguish the classes. Table 4 shows results of running experiments 5 times on each of the other real datasets. In both cases cost sensitive learning has clearly outperformed regular learning. In the case of ECG, only the method with adaptable κ and adjusted H was successful.

Table 3. Effect of S and IR on learning of a small artificially adjusted dataset.

$\|S\|$ (Class separability)	IR (Imbalance ratio)	G-mean regular learning	G-mean cost sensitive learning ($\kappa 2$)
0.0005	1	**50.42 %**	49.80 %
0.17	1	**77.46 %**	72.43 %
0.42	1	**91.49 %**	91.39 %
0.69	1	**100 %**	98.99 %
0.02	5	0 %	0 %
0.16	5	40.82 %	**73.95 %**
0.42	5	86.60 %	**89.56 %**
0.69	5	95.74 %	**100 %**

Table 4. Performance comparison of regular and cost sensitive learning on other time series datasets.

Dataset	$\|S\|$ (Class separability)	IR (Imbalance ratio)	G-mean regular learning	G-mean cost sensitive learning ($\kappa 2$)
Earthquake	0.0003	3.95	27.9 % ± 4.72 %	**45.4 % ± 10.1 %**
ECG	0.09	20	0.00 % ± 0 %	**98.6 % ± 1.03 %**

4 Conclusion

We have seen that real world datasets can be highly imbalanced and are different from most competitive datasets. The impact of imbalance and class separability would make most machine learning algorithms biased towards one single majority class, leading to the class imbalance problem.

We have combined several methods and for the first time used them for learning time series with multi-channel CNNs. Using Global Mean Error Separation from Castro et al. [5] to achieve cost-sensitive learning and combining them with a method to learn the parameter κ for the specific dataset introduced by Kahn et al. [9], we were able to show successful learning on a highly imbalanced dataset from XING with low C2C separability. The chosen approach also permits a simpler implementation with GPU libraries like Theano. Further improvements could be achieved with the novel use of the Silhouette score to adjust the maximum applicable cost to the C2C separability of the given dataset, and thus defining a better range for κ. Although the used adjustment function can be further improved to not punish a class when using balanced data, we have shown that this seemed to lead to no significant negative effect on learning. These results have also been confirmed for two other imbalanced time series datasets (Earthquake and ECG data) with different C2C separability, showing significantly improved results to regular learning without cost sensitivity.

Further work to generalise and evaluate the approach on multi-class problems is necessary. Although the single methods and metrics can be extended to several classes, it was already shown that this does not automatically mean a solution for multi-class imbalances [22]. Also a more comprehensive comparison with other approaches in the literature that could be used for CNNs is needed. Overall, the proposed method was shown to be an effective algorithmic approach to solve the class imbalance problem for binary classes when using convolutional deep neural networks that can easily be integrated into different neural network architectures.

References

1. Alejo, R., Valdovinos, R.M., García, V., Pacheco-Sanchez, J.: A hybrid method to face class overlap and class imbalance on neural networks and multi-class scenarios. Pattern Recogn. Lett. **34**(4), 380–388 (2013)
2. Berardi, V.L., Zhang, G.P.: The effect of misclassification costs on neural network classifiers. Decis. Sci. **30**(3), 659–682 (1999)
3. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy), vol. 4, p. 3, Austin, TX (2010)
4. Cao, P., Zhao, D., Zaïane, O.R.: A PSO-based cost-sensitive neural network for imbalanced data classification. In: Li, J., Cao, L., Wang, C., Tan, K.C., Liu, B., Pei, J., Tseng, V.S. (eds.) PAKDD 2013. LNCS, vol. 7867, pp. 452–463. Springer, Heidelberg (2013)
5. Castro, C.L., de Pádua Braga, A.: Artificial neural networks learning in ROC space. In: IJCCI, pp. 484–489 (2009)
6. Chan, P.K., Stolfo, S.J.: Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In: KDD, vol. 1998, pp. 164–168 (1998)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
8. Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/
9. Khan, S.H., Bennamoun, M., Sohel, F., Togneri, R.: Cost sensitive learning of deep feature representations from imbalanced data (2015). arXiv preprint arXiv:1508.03422
10. Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Supervised neural network modeling: an empirical investigation into learning from imbalanced data with labeling errors. IEEE Trans. Neural Netw. **21**(5), 813–830 (2010)
11. Kukar, M., Kononenko, I., et al.: Cost-sensitive learning with neural networks. In: ECAI, pp. 445–449. Citeseer (1998)
12. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backprop. In: Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 1524, pp. 9–48. Springer, Heidelberg (2012)
13. Liu, X.Y., Zhou, Z.H.: The influence of class imbalance on cost-sensitive learning: an empirical study. In: Sixth International Conference on Data Mining, 2006, ICDM 2006, pp. 970–974. IEEE (2006)

14. Murphey, Y.L., Guo, H., Feldkamp, L.A.: Neural learning from unbalanced data. Appl. Intell. **21**(2), 117–128 (2004)
15. Ng, A.Y.: Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 78. ACM (2004)
16. Raj, V.: Towards effective classification of imbalanced data with convolutional neural networks. Master's thesis, Department of Informatics, University of Hamburg, Vogt-Koelln-Str. 22527 Hamburg, Germany, April 2016
17. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
18. Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn. **40**(12), 3358–3378 (2007)
19. Wang, J., Jean, J.: Resolving multifont character confusion with neural networks. Pattern Recogn. **26**(1), 175–187 (1993)
20. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **3**, 408–421 (1972)
21. Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: Li, F., Li, G., Hwang, S., Yao, B., Zhang, Z. (eds.) WAIM 2014. LNCS, vol. 8485, pp. 298–310. Springer, Heidelberg (2014)
22. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans. Knowl. Data Eng. **18**(1), 63–77 (2006)