A Neural Network Cost Function for Highly Class-Imbalanced Data Sets

David Twomey and Denise Gorse

University College London - Computer Science
Gower Street, London, WC1E 6BT - UK

Abstract. We introduce a new cost function for the training of a neural network classifier in conditions of high class imbalance. This function, based on an approximate confusion matrix, represents a balance of sensitivity and specificity and is thus well suited to problems where cost functions such as the mean squared error and cross entropy are prone to overpredicting the majority class. The benefit of the new measure is shown on a set of common class-imbalanced datasets using the Matthews Correlation Coefficient as an independent scoring measure.

1 Introduction

Machine learning classification algorithms perform optimally on data that is balanced and evenly class-distributed. High levels of class imbalance are however found in many real-world datasets, including domains such as medical diagnosis where underprediction of the minority could lead to serious consequences [1]. This paper proposes a cost function for neural networks designed to better handle binary class-imbalanced problems, even in cases in which the majority class outnumber the minority by ratios exceeding 100:1. The geometric mean (G-mean) is a composite measure that captures the trade-off between the avoidance of both false negatives (sensitivity) and false positives (specificity). The G-mean is, however, a counting measure and as such is unsuited as a cost function for learning models trained using gradient based methods. We introduce an approximate G-mean based on an approximated confusion matrix and derive its derivative for use in a backprop-trained feedforward neural network. We demonstrate the outperformance of this new cost function over standard ones on common class-imbalanced UCI datasets.

2 Background and Motivation

The majority of methods to handle class-imbalanced data sets either attempt to re-sample the data in a way that leaves the dataset more balanced [2-4], or associate different costs to minority vs. majority misclassification errors so as to make the classifier more sensitive to the minority class [5-8].

The best known of the former methods are oversampling and undersampling. Simple oversampling duplicates samples of the minority class, while undersampling randomly removes instances of the majority. Both methods are easy to implement, but the former can lead to overfitting while the latter can lead to a loss of important information. Undersampling also introduces unwanted nondeterminism into what is otherwise a deterministic learning process [2]. SMOTE [3] is a variant of

oversampling that introduces synthetic examples of the minority class along line segments joining them to their nearest (minority class) neighbours [3]. Despite its benefits, some drawbacks still exist such as over-generalisation and variance [4].

The major difficulties in cost sensitive learning are that it may be hard to decide how much greater should be the cost of misclassifying a minority example [5], and exactly where in the learning process the cost should be applied. Nevertheless the method has been applied successfully in a wide range of applications. For example [6] presents several different approaches to achieving cost-sensitive neural networks by adding cost factors to either the learning rate or the output of the network, while [7] shows that dividing network outputs by the 'class probability' substantially improves identification success. However [8], which used a number of the 30 imbalanced UCI datasets used in this current work, found that cost sensitive learning was less successful than the manipulation of the decision threshold for minority cases.

It is unlikely any one method will prove to be a panacea for the problem of class imbalance. However there is one approach that has been so far little investigated: the change—as opposed to the modification by class-dependent costs—of the cost function used in training to one more appropriate to class-imbalanced data sets. [9] explored the use of an approximated F-score for backpropagation training, comparing this to the use of mean squared error (MSE), but the derivative of the approximated measure was however somewhat complex and possibly liable to local minima, and as such its advantage (displayed in [9] for an image classification dataset with imbalance ratio (IR) of around 14) might not extend to cases of more severe imbalance.

3 Methodology

3.1 Why Use the G-mean?

Many performance measures can be derived from a confusion matrix, including the F-score used in [9] and the Matthews Correlation Coefficient (MCC) [10] (to be used as an independent scoring measure in the Results section below). We choose to use as our cost function a differential approximation to the G-mean

$$G\text{-}mean = \text{sqrt}(\frac{tp}{(tp+fp)} \times \frac{tn}{(tn+fp)})$$

(where *tp*, *tn*, *fp*, *fn* are the number of true positives, true negatives, false positives, and false negatives respectively). We pick the G-mean for two reasons: (1) this measure is considered a good one for imbalanced datasets, as highlighted in [11]; and (2) unlike the MCC (and F-score, as used in [9]) the G-mean allows for an approximated form with a relatively simple derivative.

3.2 Computation of Weight Changes

The computation of weight changes in our method starts with a pattern-by-pattern accumulation of an *approximate confusion matrix*, CM_{apx}, which for pattern q, with network output $y_q \in [-1,+1]$ and true class label $t_q \in \{-1,+1\}$, is incremented by

$$\Delta CM_{apx}(q) = \frac{1}{4}\begin{bmatrix}(1-y_q)(1-t_q) & (1+y_q)(1-t_q)\\ (1-y_q)(1+t_q) & (1+y_q)(1+t_q)\end{bmatrix}.$$

The constructed CM_{apx} can be used to calculate an approximated G-mean, *GMN*. We in practice choose to optimise the square of this quantity,

$$\mathrm{GMN}^2 = \frac{1}{4}\times\frac{\left(\sum_p\left(1-y_p\right)(1-t_p)\right)\left(\sum_p\left(1+y_p\right)(1+t_p)\right)}{\left(\sum_p(1-t_p)\right)\left(\sum_p(1+t_p)\right)},$$

on the basis that this choice will achieve the same goals as optimising GMN itself while resulting in a simpler derivative, given below,

$$\frac{\partial}{\partial y_q}\mathrm{GMN}^2 = \frac{\left[\left(1+t_q\right)\left(\sum_{\mathrm{p}}\left(1-y_p\right)(1-t_p)\right)-\left(1-t_q\right)\left(\sum_{\mathrm{p}}\left(1+y_p\right)(1+t_p)\right)\right]}{16n_1n_2}$$

(where n_1 and n_2 denote the numbers of examples in classes 1 and 2, respectively). This derivative can be used to calculate derivatives with respect to weights, and hence weight changes, via the chain rule of differentiation. We note that while the form of the derivative above is relatively simple, it does still imply a precomputation of the y_p values in order to know the values of the summations, so that as in [9] each iteration of backpropagation involves two passes through the training set.

3.3 **Datasets**

The 30 chosen datasets were downloaded from the UCI repository [12], with class imbalance ratios ranging from 5.55 (*dermatology-6*) to 129.44 (*abalone19*), feature dimensions ranging from 6 (seven instances, e.g. *car-good*) to 41 (five instances, e.g. *kddcup-guess_passwd_vs_satan*), and numbers of examples ranging from 148 (*lymphography-normal-fibrosis*) to 2935 (*abalone19*). Space precludes a full listing here of the datasets and their attributes, available from the authors on request.

3.4 **Experiment Design**

3.4.1 Data Preprocessing

1) **Initial training-testing *k*-fold split**: Each dataset was shuffled and divided into five cross-folds, each allocating 20% of the examples to testing.
2) **Training-validation split**: For each fold, the remaining 80% of the data was subdivided into 75% training and 25% validation, using stratified-sampling to ensure similar class distributions for training and validation. (Note that the separations for constructing the test sets were *not* stratified.)
3) **Standardisation**: All inputs were standardised by subtracting the mean and dividing by the standard deviation for that input in the training dataset.

3.4.2 Training and Testing

Both the hidden and output activations were set as *tanh* and the output binarised with a decision threshold of 0. The nets used a single hidden layer with the number of nodes equal to the average of the number of inputs and outputs, trained using bold-driver backpropagation. At each epoch the cost function was evaluated on the validation set and weights at the best performing epoch were stored for use in testing.

3.4.3 Performance Comparison

Competitor cost functions were evaluated using the MCC [10]. An MCC score of +1 represents a perfect prediction, 0 either no better than random or all assigned to one class, and -1 an inverse prediction. In a binary setting the MCC is defined as

$$\mathrm{MCC} = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}.$$

More specifically, the methods were compared using the *relative underperformance to the GMN-trained net* (in relation to MCC), calculated as given below,

$$\xi_{\mathrm{ALT}} = \frac{\mathrm{MCC}_{\mathrm{ALT}} - \mathrm{MCC}_{\mathrm{GMN}}}{|\mathrm{MCC}_{\mathrm{GMN}}|},$$

where the subscript refers to the network's cost function and ALT $\in$ {RMSE, CE}. Underperformance was chosen instead of outperformance in order to avoid divide-by-zero errors, as for several datasets the ALT cost functions gave zero MCC scores as all instances were assigned to the same class. In our observation the working range of ξ_{ALT} was from -1 (ALT score zero, GMN score > 0) to slightly above zero (ALT score slightly exceeding GMN score). An underperformance value of -1 will represent the strongest evidence in our test set for the superiority of GMN.

4 Results

Figure 1 shows the average competitor underperformances ξ_{ALT} across five different (dataset shuffling and weight) initialisations as a function of IR. (An MCC score for a run is based on the combined confusion matrix of all five folds, not on an average of each fold.) For IR < 20 all three cost functions behave similarly, but as the IR becomes more extreme there are more examples of clear superiority of GMN. In the most extreme example (*abalone19*), with an IR of 129.44, both the MSE and CE trained networks make no attempt to predict the minority class and thus achieve zero MCC scores, while in contrast GMN attains a positive MCC of 0.09±0.04.

Though increasing IR generally leads to better relative performance of GMN, there are exceptions. One such is *abalone-20_vs_8-9-10* (IR=72.69), for which both ALTs marginally outperform GMN. We investigated this case with an additional 55 runs, discovering that even in such an apparently unfavourable case there may be benefits to the use of GMN, as evidenced in Figure 2. It can be seen there that while

the two ALT MCC scores have a higher average there is much more variance, and in particular more instances for which both ALTs have an MCC of zero (all cases assigned to the majority class), and thus no discretionary power.

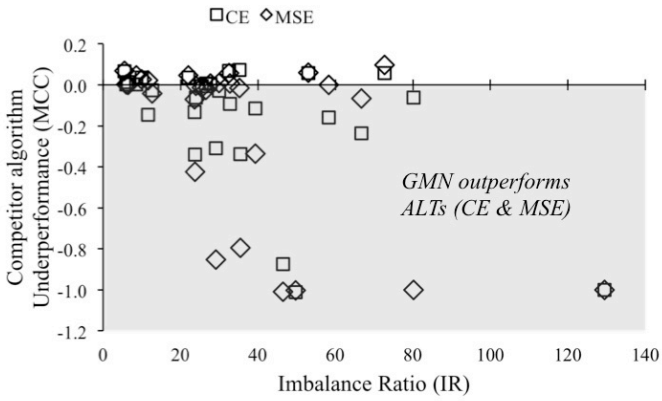

Fig. 1: Underperformance (measured by MCC) of both MSE and CE relative to GMN, plotted against imbalance ratio (IR). GMN outperforms the competitor cost functions everywhere in the shaded region.

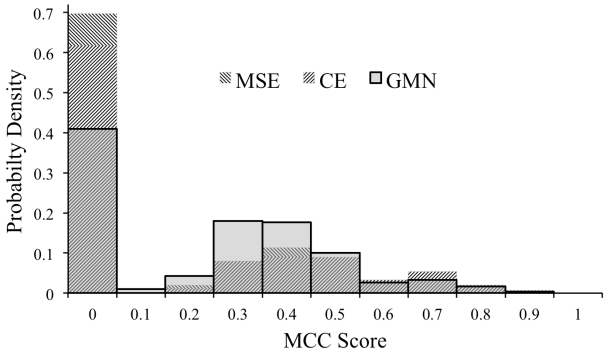

Fig. 2: Abalone-20_vs_8-9-10 dataset out-of-sample scores across all random seeds/*k*-folds (300 total) for GMN, CE, and MSE cost functions.

5 Discussion

We have demonstrated the advantage of GMN, a newly proposed cost function based on an approximated G-mean, over the conventional cross entropy (CE) and MSE on 30 imbalanced UCI datasets (IRs from 5.55 to 129.44). For IRs less than 20 the three cost functions performed similarly, but higher imbalances showed a clear benefit of using GMN. The GMN scores were also found to less sensitive to initial conditions.

A number of avenues will be explored in future work. The approximated F-score of [9], which was tested only on a dataset of relatively low IR (around 14), will be

implemented as a further competitor cost function to GMN, and tested on the same 30 UCI datasets used here, to discover if the approximated F-score does indeed suffer from a problem of convergence to local optima for these more challenging cases.

Given that GMN has here shown its largest benefits in cases of very high IR, further test datasets with high imbalance ratios will be collected or generated (in the latter case either by oversampling a naturally occurring majority class or by the generation of synthetic data where the imbalance ratio can be controlled).

In addition, the extension of GMN from binary classification problems to multi-class could be examined. [11] have shown that existing approaches such as data resampling, believed to be effective in addressing the class imbalance problem, may in fact only be effective for two-class datasets. Lastly, the current GMN-based training method uses batched backpropagation which computes and sums the derivatives of all training patterns and updates once every epoch; however there is no reason why mini-batch training (as in [9]) could not be investigated, which could potentially provide quicker convergence and better accuracy on larger datasets.

References

[1] G. Haixiang, Learning from class-imbalanced data: review of methods and applications, *Expert Systems with Applications,* 73, 220-239, 2017.

[2] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, P. Kennedy, Training Deep Neural Networks on Imbalanced Data Sets, *International Joint Conference on Neural Networks* (IJCNN 2016), pages 4368-4374, July 24-29, Vancouver (Canada), 2016.

[3] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research*, 16:321-357, 2002.

[4] B. Wang, N. Japkowicz, Imbalanced Data Set Learning with Synthetic Learning, in proceedings of the *IRIS Machine Learning Workshop Volume 19*, 2004.

[5] C. Elkan, The foundations of cost-sensitive learning, in proceedings of the *17th International Joint Conference on Artificial Intelligence* (IJCAI 2001), 973-978, 2001.

[6] M. Zukar, I. Kononenko, Cost-Sensitive Learning with Neural Networks, in proceedings of the *13th European Conference on Artificial Intelligence* (ECAI), pages 445-449, 1998.

[7] L. Al-Haddad, C. Morris and L. Boddy, Training radial basis function neural networks: effects of training set size and imbalanced training sets, *Journal of Microbiological Methods*, 43:33-44, 2000.

[8] Z. Zhou and X. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, *IEEE Transactions on Knowledge and Data Engineering*, 18, 63-77, 2006.

[9] J. Pastor-Pellicer, F. Zamora-Martinez, F-measure as the Error Function to Train Neural Networks, 7902:376-384, in proceedings of the *International Work-Conference on Artificial Neural Networks* (IWANN), pages 376-384, 2013.

[10] B. W. Matthews, Comparison of the predicted and observed secondary structure of T4 phage lysozyme, *Biochim. Biophys. Acta*, 405: 442-445, 1975.

[11] S. Wang, L. Minku and X. Yao, A systematic study of online class imbalance learning with concept drift, *IEEE Transactions on Neural Networks and Learning Systems*, arxiv preprint: 1703.06683, 2017.

[12] http://sci2s.ugr.es/keel/imbalanced.php