Effective data generation for imbalanced learning using conditional generative adversarial networks

Georgios Douzas[1], Fernando Bacao[1]

[1]NOVA Information Management School, Universidade Nova de Lisboa

*Corresponding Author

Postal Address: NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal

Telephone: +351 21 382 8610

Learning from imbalanced datasets is a frequent but challenging task for standard classification algorithms. Although there are different strategies to address this problem, methods that generate artificial data for the minority class constitute a more general approach compared to algorithmic modifications. Standard oversampling methods are variations of the SMOTE algorithm, which generates synthetic samples along the line segment that joins minority class samples. Therefore, these approaches are based on local information, rather on the overall minority class distribution. Contrary to these algorithms, in this paper the conditional version of Generative Adversarial Networks (cGAN) is used to approximate the true data distribution and generate data for the minority class of various imbalanced datasets. The performance of cGAN is compared against multiple standard oversampling algorithms. We present empirical results that show a significant improvement in the quality of the generated data when cGAN is used as an oversampling algorithm.

1 Introduction

Learning from imbalanced data is an important problem for the research community as well as the industry practitioners (N.V. Chawla, N. Japkowicz 2003). An imbalanced learning problem can be defined as a learning problem from a binary or multiple-class dataset where the number of instances for one of the classes, called the majority class, is significantly higher than the number of instances for the rest of the classes, called the minority classes (Chawla et al. 2002). The Imbalance Ratio (IR), defined as the ratio between the majority class and each of the minority classes, varies for different applications and for binary problems values between 100 and 100.000 have been observed (Chawla et al. 2002), (Barua et al. 2014).

Imbalanced data are a characteristic of multiple real-world applications such as medical diagnosis, information retrieval systems, fraud detection, detection of oil spills in radar images, direct marketing, automatic classification of land use and land cover in remote sensing images, detection of rare particles in experimental high-energy physics, telecommunications management and bioinformatics (Akbani, Kwek, and Japkowicz 2004), (He and Garcia 2009), (Clearwater and Stern 1991), (Graves et al. 2016), (Verbeke et al. 2012), (Zhao et al. 2008). Standard learning methods perform poorly in imbalanced data sets as they induce a bias in favor of the majority class. Specifically, during the training of a standard classification method the minority classes contribute less to the minimization of the objective function. Also the distinction between noisy and minority class instances is often difficult. An important observation is that in many of these applications the misclassification cost of the minority classes is often higher than the misclassification cost of the majority class (Domingos 1999), (Ting 2002). Therefore the methods that address the class imbalance problem aim to increase the classification accuracy for the minority classes.

There are three main approaches to deal with the class imbalanced problem (Fernández et al. 2013). The first is the modification/creation of algorithms that reinforce the learning towards the minority class. The second approach is the application of cost-sensitive methods at the data or algorithmic level in order to minimize higher cost errors. The third and more general approach is the modification at the data level by rebalancing the class distribution through under-sampling, over-sampling or hybrid methods.

Our focus in this paper is oversampling techniques, which result in the generation of artificial data for the minority class. Standard oversampling methods are inspired by Synthetic Minority Oversampling Technique (SMOTE) algorithm (Chawla et al. 2002), generating synthetic samples along the line segment that joins minority class samples. A direct approach to the data generation process would be the use of a generative model that captures the actual data distribution. Generative Adversarial Networks (GAN) is a recent method that uses neural networks to create generative models (Goodfellow et al. 2014). A conditional Generative Adversarial Network (cGAN) extends the GAN model by conditioning the training procedure on external information (Mirza and Osindero 2014). In this paper we apply a cGAN on binary class imbalanced datasets, where the cGAN conditioning on external information are the class labels of the imbalanced datasets. The final generative model is used to create artificial data for the minority class i.e. the generator corresponds to an oversampling algorithm.

For the evaluation of cGAN as an oversampling method an experimental analysis is performed, based on 12 publicly available datasets from Machine Learning Repository. In order to test it on a wide range of IRs, additional datasets are created by undersampling the minority classof these 12 datasets as well as by adding simulated datasets with appropriate characteristics. Then the proposed method is compared to Random Oversampling, SMOTE algorithm, Borderline SMOTE (Han,Wang, Mao,2005),andADASYN (He,Bai,Garcia, Li, 2008) andCluster-SMOTE (Cieslak, Chawla, Striegel, 2006). For the classification of the binary class datafiveclassifiers and three evaluation metrics are applied.

The sections in the paper are organized as follows. In section 2, an overview of related previous works and existing sampling methods is given. In Section 3, the theory behind GANs is described. Section 4 presents the proposed method in detail. Section 5 presents the research methodology. In section 6 the experimental results are presented while conclusions are provided in section 7.

2 Related work

Considering that our focus is the modification on the data level, and particularly the generation of artificial data, we provide a short review of the oversampling methods. A review of the other methods can be found in (Galar et al. 2012), (N.V. Chawla n.d.). Oversampling methods generate synthetic examples for the minority class and add them to the training set. A simple approach, known as Random Oversampling, creates new data by copying random minority class examples. The drawback of this approach is that the exact replication of training examples can lead to overfitting since the classifier is exposed to the same information.

An alternative approach that aims to avoid this problem is SMOTE. Synthetic data are generated along the line segment that joins minority class samples. SMOTE has the disadvantage that, since the separation between majority and minority class clusters is not often clear, noisy samples may be generated (He and Garcia 2009). To avoid this scenario various modifications of SMOTE have been proposed. SMOTE + Edited Nearest Neighbor (Batista, Prati, and Monard 2004) combination applies the edited nearest neighbor rule (Wilson 1972) after the generation of artificial examples through SMOTE to remove any misclassified instances, based on the classification by its three nearest neighbors. Safe-Level SMOTE (Bunkhumpornpat, Sinapiromsaran, and Lursinsap 2009) modifies the SMOTE algorithm by applying a weight degree, the safe level, in the data generation process. Borderline-SMOTE (Han, Wang, and Mao 2005), MWMOTE (Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning) (Barua et al. 2014), ADASYN (He, H., Bai, Y., Garcia, E., Li 2008) and its variation KernelADASYN (Tang, B. and He, H. 2015) aim to avoid the generation of noisy samples by identifying the borderline instances of the majority and minority classes that in turn are used to identify the informative minority class samples.

The methods above address the problem of between-class imbalance (Nekooeimehr and Lai-Yuen 2016). Another type of problem is the within-class imbalance (Nekooeimehr and Lai-Yuen 2016), (Bunkhumpornpat, Sinapiromsaran, and Lursinsap 2012), (Cieslak and Chawla 2008), (Jo T. and Japkowicz N. 2004) i.e. when sparse or dense subclusters of minority or majority instances exist. Clustering based oversampling methods that deal with the between-class imbalance problem have recently been proposed. These methods are initially partitioning the input space and then apply sampling methods in order to adjust the size of the various clusters. Cluster-SMOTE applies the k-means algorithm and then generates artificial data by applying SMOTE in the clusters. Similarly DBSMOTE (Bunkhumpornpat, Sinapiromsaran, and Lursinsap 2012) uses the DB-SCAN algorithm to discover arbitrarily shaped clusters and generates synthetic instances along a shortest path from each minority class instance to a pseudo-centroid of the cluster. A-SUWO (Nekooeimehr and Lai-Yuen 2016) creates clusters of the minority class instances with a size, which is determined using cross validation and generates synthetic instances based on a proposed weighting system. SOMO (Douzas and Bacao 2017) creates a two dimensional representation of the input space and based on it, applies the SMOTE procedure to generate intracluster and intercluster synthetic data that preserve the underlying manifold structure. Other types of oversampling approaches are based on ensemble methods (Wang, Minku, Yao, 2015), (Sun et al., 2015) such as SMOTEBoost (Chawla, Lazarevic, Hall, and Bowyer 2003) and DataBoost-IM (Guo and Viktor 2004).

3 GAN and cGAN algorithms

In this section, we provide a summary of the GAN and cGAN frameworks following closely the notation in (Goodfellow et al. 2014) and (Gauthier 2015). The GAN is based on the idea of competition, in which a generator G and a discriminator D are trying to outsmart each other. The objective of the generator is to confuse the discriminator. The objective of the discriminator is to distinguish the instances coming from the generator and the instances coming from the original dataset. If the discriminator is able to identify easily the instances coming from the generator then, relative to its discrimination ability, the generator is producing low quality data. We can look at the GAN setup as a training environment for the generator where the discriminator, while also improving, is providing feedback about the quality of the generated instances, forcing the generator to increase its performance.

More formally, the generative model G, defined as G: Z $\rightarrow$X where Z is the noise space of arbitrary dimension dZ that corresponds to a hyperparameter and X is the data space, aims to capture the data distribution. The discriminative model, defined as D: X $\rightarrow$ [0, 1], estimates the probability that a sample came from the data distribution rather than G. These two models, which are both multilayer perceptrons, compete in a two-player minmax game with value function:

minGmaxDV(D, G)= Expdata(x)[logD(x)]+ Ezpz(z)[log(1 D(G(z)))] (1)

The xX values are sampled from the data distribution pdata(x) and the zZ values are sampled from the noise distribution pz(z) . The training procedure consists of alternating between k optimizing steps for D and one optimizing step for G by applying SGD. Therefore during training, D is optimized to correctly classify training data and samples generated from G, assigning 1 and 0 respectively. On the other hand the generator is optimized to confuse the discriminator by assigning the label 1 to samples generated from G. The unique solution of this adversarial game corresponds to G recovering the data distribution and D equal to $\frac{1}{2}$ for any input (Goodfellow et al. 2014).

The cGAN is an extension of the above GAN framework. An additional space Y is introduced, which represents the external information coming from the training data. The cGAN framework modifies the generative model G, to include the additional space Y, as follows:

G: Z $\times$ Y $\to$X (2)

Similarly to G the discriminator D is modified as follows:

D: X $\times$ Y $\to$ [0, 1] (3)

Equation (1) is also modified reflecting the redefinition of G and D:

minGmaxDV(D, G)= Ex, ypdata(x, y)[logD(x, y)]+ Ezpz(z), yp(y) [log(1 D(G(z, y), y))] (4)

The (x, y)X $\times$ Y values are sampled from the data distribution pdata(x, y), the zZ values are sampled from the noise distribution pz(z) and yY values are sampled from conditional data vectors found in the training data and represented by the density function py(y) . The training procedure for cGANs is identical to the training of the GAN model. Rephrasing equation (4), the cost functions for the gradient update of the discriminator and generator on a minibatch of m training examples (xi, yi)mi=1 and m noise samples zimi=1 are the following logistic cost expressions:

JD= 12m(i=1mlogD(xi, yi)+ i=1mlog(1 − D(G(zi, yi),yi))) (5)

JG= 1m i=1mlogD(G(zi, yi),yi) (6)

Equation (6) is the modified version of the generator's loss function based on equation (4) in order to avoid saturation of the discriminator (Goodfellow et al. 2014). The cGAN model is trained by alternating gradient-based updates of equations (5) and (6), similarly to the GAN framework, where D parameters are updated k times followed by a single update of G parameters.

4 The cGAN application to the imbalanced learning problem

The aim of the paper is to evaluate the effectiveness of a cGAN's generator G as an oversampling method. Specifically, the cGAN framework is trained on binary class imbalanced data (xi, yi)ni=1, where (xi, yi)X $\times$ 0, 1 with y = 1 corresponding to the minority class. The external information of the CGAN process is represented by the class variable y. As it was mentioned above Z is a noise space of dimensionality dZ while the dimensionalities of X and Y are defined as dX and dY, respectively. Therefore the generator receives as input a vector that belongs to the Z $\times$ Y space and outputs a vector that belongs to the input space X. On the other hand the discriminator receives as input a vector that belongs to the X $\times$ Y space and classifies it as real or as, generated by G, artificial data. After the end of cGAN training the generator can be used to create artificial data for the minority class by receiving input vectors of the form (z, y=1)Z $\times$ Y, where z is a sample from the noise space Z.

The hyperparameters of the above process are the dimension dZ of the noise space, the hyperparameters related to the G and D networks architecture as well as their training options. According to equations (2) and (3), the networks architectures of the two models are constrained by the dimensionality of the input space X, the output space Y and the fact that D is a binary classifier. Specifically, the input and output layers of G have dZ + dY and dX number of units, respectively. Also, the input and output layers of D have dX + dY and one unit respectively. For a binary classification problem the dimensionality dY of the class y is equal to one. Therefore, using a single hidden layer for G and D, the non-constrained hyperparameters of the cGAN are the dimension dZ of the noise space and the number of units for the hidden layers of G and D.

It is also important to notice that there are different formulations of the optimization problem as well as choices of the loss function of the cGAN framework. Following (Goodfellow et al. 2014) and (Gauthier 2015), we choose the vanilla cGAN formulation and the logistic cost functions given in equations (5) and (6). The choice of the hyperparameters of the networks is described in the next section.

5 Research methodology

In order to test the performance of the proposed application of cGANs, 12 imbalanced datasets from the Machine Learning Repository UCI were used. For each one of them, additional datasets were generated that resulted from undersampling the minority class of the initial datasets such that their final IR was increased approximately by a multiplication factor of 2, 4, 6, 10, 15 and 20. This procedure was applied for a given multiplication factor only when the final number of minority class instances was not less than 8. Additionally 10 artificial datasets were generated using the Python library Scikit-Learn (Pedregosa et al. 2011) that adapts an algorithm from (Guyon 2003). Specifically, each class is composed of a number of gaussian clusters, in the range of 1 to 5, each located around the vertices of a hypercube. For each cluster, informative features are drawn independently from N(0, 1) and then randomly linearly combined within each cluster in order to add covariance. The clusters are then placed on the vertices of the hypercube. Therefore a subcluster structure is created for the minority class. Table 1 shows a summary of the 71 data sets:

The performance of the cGAN as an oversampling method was evaluated and compared against Random Oversampling, SMOTE, Borderline SMOTE, ADASYN and Cluster-SMOTE. Since total accuracy is not appropriate for imbalanced datasets (balanced) F-measure, G-mean and Area Under the ROC Curve (AUC) are used (He and Garcia 2009), (Galar et al. 2012).

A ranking score was applied to each oversampling method for every combination of the 71 data sets, 3 evaluation metrics and 5 classifiers. Additionally to the 6 oversampling algorithms we also included the performance of the classifiers when no oversampling is used. Therefore the ranking score for the best performing method is 1 and for the worst performing method is 7. The Friedman test was applied to the ranking results, followed by the Holm's test where cGAN oversampler was considered as the control method (Guyon 2003). Generally the Friedman test is used to detect differences in the results across multiple experimental attempts, when the normality assumption may not hold. The null hypothesis is whether the classifiers have a similar performance across the oversampling methods and evaluation metrics when they are compared to their mean rankings. Holm's test is a non-parametric version of the t-test and the null hypothesis is whether the proposed application of cGAN outperforms the other methods as the control algorithm.

For the evaluation of the oversampling methods, Logistic Regression (LR) (McCullagh P. 1984), Support Vector Machine with radial basis function (SVM) (Chang and Lin 2011), Nearest Neighbors (KNN) (Cover and Hart 1967), Decision Trees (DT) (Quinlan 1993) and Gradient Boosting Machine (GBM) (Friedman 2001) were used. In order to evaluate the performance of the algorithms k-fold cross validation was applied with k = 5. Before training, in each stage i 1, 2, ... , k of the k-fold cross validation procedure, synthetic data Tg, i were generated based on the training data Ti of the k - 1 folds such that the resulting Tg, i [] Ti training set becomes perfectly balanced. This enhanced training set in turn was used to train the classifier. The performance evaluation of the classifiers was done on the validation data Vi of the remaining fold.

The hyperparameter tuning of the classifiers and the various oversampling algorithms was done using only the training data Ti of each cross validation stage i 1, 2, ... , k . Specifically, an additional training/validation splitting on Ti was applied as Ti = Tt, i [] Tv, i. The classifiers were trained on Tt, I and the optimal hyperparameters were selected in order to maximize the AUC of the validation set Tv, i. For SMOTE, Borderline SMOTE, Cluster-SMOTE and ADASYN algorithms the tuning parameters, for any combination of datasets, metrics and classifiers, were selected such that the performance of the classifier on the validation set Tv, i is maximized when synthetic data were added to Tt, i and formed the training set for the classifier.

A similar strategy was followed for the cGAN hyperparameter tuning. The validation performance was optimized only for the GBM classifier and the same cGAN hyperparameters were used for the rest of the classifiers. Furthermore instability issues during the cGAN training were observed. This problem was partially solved through the fine tuning of the hyperparameters, which provided a stable state. Both the generator and the discriminator used rectified linear units (Glorot et al. 2011) as activation functions for the single hidden layer and sigmoid activations for the output layer. Each model was trained using the Adam optimizer (Kingma and Ba 2015) with the default settings for both the initial learning rate and the exponential decay rates of the moment estimates. No dropout was applied to either the generator or the discriminator. Also the hyperparameter k, that controls the number of D parameters updates followed by a single G parameters update, was set equal to one since no improvement was observed for higher values. The optimal range for the number of epochs was found to be 2000 - 10000 and the mini-batch size was set to the range from 1/20 up to 1/100 of each training set's sample size. Due to limited computational resources the search for the optimal hyperparameters was not extensive. In fact it should be noted that the time and space complexity of the cGAN follows the standard neural network characteristics. Contrary to the other methods, the cGAN as an oversampler requires a training phase. Taking into consideration the size of the datasets used, standard oversampling methods were faster compared to the cGAN procedure. A mix of random grid search and manual selection was applied, choosing dZ values as well as the number of hidden layer's units for G and D to be a multiple of the dataset's number of features by a factor of 2 up to 10. The resulting optimal values are shown on Table 2:

The experimental procedure was repeated 5 times and the reported results include the average values between the experiments. The implementation of the classifiers and standard oversampling algorithms was based on the Python libraries Scikit-Learn (Pedregosa et al. 2011) and Imbalanced-Learn (Lemaitre, Nogueira and Aridas 2016). The cGAN implementation1 is based on TensorFlow (Abadi et al 2015).

6 Experimental results

The ranking results2 are summarized using 15 plots. Each plot corresponds to a specific classifier and evaluation metric. The x-axis and y-axis of each plot represent the IR of the imbalanced dataset and the ranking score, respectively. Therefore each plot includes 7 polygon lines, corresponding to the 7 oversampling methods, where the bold line represents the cGAN oversampler:

The mean ranking of the oversampling methods across the data sets for each combination of a classifier and evaluation metric is summarized in Table 3:

The cGAN method is the best performing method for all combinations of classification algorithms and evaluation metrics. In order to statistically confirm the conclusion, the Friedman test is applied and the results are shown in Table 4:

Therefore at a significance level of a = 0.05 the null hypothesis is rejected, i.e. the classifiers do not perform similarly in mean rankings for any evaluation metric. The Holm's test is applied with cGAN oversampler as the control method. The adjusted a and the p-values are shown in table 5:

We observe that the cGAN oversampler outperforms all other methods for any evaluation metric when DT is used as a classifier. For the SVM, GBM and KNN classifiers, cGAN outperforms the other methods in 2 out of 3 evaluation metrics. Finally when LR is used as a classifier, cGAN has a higher performance compared to the other oversamplers in the F-measure. Although the simulated data have a subclustering structure for the minority class and vanilla cGAN suffers from the problem of mode collapse (Mehrjou and Saremi 2017), where the generator focuses in parts of the low-dimensional data manifold, we did not observe a decrease in the performance of the classifiers in these cases.

7 Conclusions

In this paper we propose the use of cGAN, as an oversampling approach for binary class imbalanced data. The proposed application of cGAN results to a generative model G that has learned the actual data distribution conditioned on the class labels. G is used to generate synthetic data for the minority class i.e. as an oversampling algorithm. cGAN performance was evaluated on 71 datasets with different imbalance ratios, number of features and subclustering structures and compared to multiple oversampling methods, using Logistic Regression, Support Vector Machine, Nearest Neighbors, Decision Trees and Gradient Boosting Machine as classifiers. The results show that cGAN performs better compared to the other methods for a variety of classifiers, evaluation metrics and datasets with complex structure. The explanation for this improvement in performance relates to the ability of cGAN to recover the training data distribution, if given enough capacity and training time. This ability is in contrast to standard oversampling methods where heuristic approaches are applied in order to generate minority class instances in safe areas of the input space. Although training the cGAN requires more effort and time compared to standard oversampling methods, once the training is finished, the generation of minority class instances is simple and effective. The generator accepts noise and the minority class label as input and outputs the generated data. Future research extensions of this work include the application of cGAN to the multiclass imbalanced problem and the search for more efficient ways to train G and D in the context of the imbalance learning problem.