An Unconventional Classifier for Imbalanced Data

Karthikessh P, Raghul Prashath K A, Rahul S, Sivaram T
Department of Applied Mathematics and Computational Sciences
PSG College of Technology

***Abstract*—Real life data is often imbalanced. We have developed a unconventional classifier to tackle the imbalance without using any resampling technique. We have ensembled a Support Vector classifier with a Random Forest classifier to improve the performance of the classifier which performs better than traditional machine learning algorithms.**

I. Introduction

Classification learning becomes complicated if the class distribution of the data is imbalanced. The class imbalance problem occurs when the number of representative instances is much less than that of other instances. Recently, the classification problem of imbalanced data appears frequently and has been widely concerned.

Imbalanced data sets are grouped into two, the majority class and the minority class, respectively denoted as positive and negative class. Traditional techniques are divided into few categories. Resampling technique is to increase the minority class of instances (oversampling) or decrease the majority class of instances (undersampling). Existing algorithms are improved by increasing the weight of positive instances. Classifier ensemble method is widely adopted to deal with the imbalance problem in the last decade [1].

II. Random Forest

A *decision tree* is a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits in a smaller number of steps. A decision tree is composed of internal decision nodes and terminal leaves. Each decision node m implements a test function $f_m(\boldsymbol{x})$ with discrete outcomes labeling the branches. Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome. This process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output [2].

Random forests form a family of methods that consist in building an ensemble (or forest) of decision trees [3]. Decision trees are indeed ideal candidates for ensemble methods since they usually have low bias and high variance, making them very likely to overfit the data and hence works poor for unseen data [4]. This algorithm works by drawing a *bootstrap sample* for each tree. Each tree is uncorrelated with other trees which helps generalize for unseen data. For classification, the random forest predicts the class with majority vote. If we built B decision trees, $\{T_b\}_1^B$ for C classes, then the predicted class would be $\hat{C}_{\text{rf}}^B(\boldsymbol{x}) = \arg\max\{\hat{C}_b(\boldsymbol{x})\}_1^B$ where $\hat{C}_b(\boldsymbol{x})$ is class prediction of the bth random-forest tree.

III. Support Vector Machine

Improving classifier effectiveness has been an area of intensive machine learning research over the last two decades, and this work has led to a new generation of state-of-the-art classifiers, one such classifier is SVM (Support Vector Machine) [5]. An SVM is a kind of large-margin classifier: it is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data.

The SVM in particular defines the criterion to be looking for a decision surface that is maximally far away from any data point. This distance from the decision surface to the closest data point determines the **margin** of the classifier.This method of construction necessarily means that the decision function for an SVM is fully specified by a (usually small) subset of the data which defines the position of the separator. These points are **support vectors** referred to as the support vectors [6].

The linearly separable case: Let us formalize an SVM with algebra. A decision hyperplane can be defined by an intercept term b and a decision hyperplane normal vector $\boldsymbol{w}$ which is perpendicular to the hyperplane. This vector is commonly referred to in the machine learning literature as the weight vector. Now suppose that we have a set of training data points $D = (\boldsymbol{x}_i, y_i)$, where each member is a pair of a point $\boldsymbol{x}_i$ and a class label y_i corresponding to it. For SVMs, the two data classes are always named $+1$ and -1 (rather than 1 and 0), and the intercept term is always explicitly represented as b (rather than being folded into the weight vector $\boldsymbol{w}$ by adding an extra always-on feature). The math works out much more cleanly if you do things this way, as we will see almost immediately in the definition of functional margin. The linear classifier is then:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{1}$$
$$\text{subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b) + \xi_i \geq 1,$$
$$\xi_i \geq 0 \text{ for all } i = 1, 2, \ldots N.$$

where slack variables ξ_i, measure the error instances that violate the constraint $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b) \geq 1$, the user defined parameter C determines the trade-off maxmimizing the margin

and minimizing the error. In dual form, this problem can be solved as

$$\max_{\alpha_i \in \mathbb{R}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,k=1}^{N} \alpha_i y_i \alpha_k y_k \boldsymbol{x}_i^\top \boldsymbol{x}_k \tag{2}$$

subject to $0 \leq \alpha_i \leq C, i = 1, 2, \ldots N$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

To construct non-linear decision boundaries, kernel methods are used [7].

IV. Our Ensemble Model

Ensembling models is area of research in the upcoming years. Let's say that model 1 has lower training accuracy, than model 2 on the training data. There may however be data points where model 1 performs better, but for some reason it performs terribly on others. Instead, model 2 may have a better overall performance on all the data points, but it has worse performance on the very set of points where model 1 is better. The idea is to combine these two models where they perform the best. This is why creating out-of-sample predictions have a higher chance of capturing distinct regions where each model performs the best. This type of models works well with imbalanced datasets, because it creates some probability estimates for training set, from that estimate we can conclude that which points are classified with less confidence. We can pay more attention to those points, thus try to classify those points correctly as well, which improves overall performance of the model.

The workflow of our ensemble model is as follows,

1) Split the dataset $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$ into D_{train} and D_{test} having train set proportion k_1.
2) Further divide the D_{train} dataset as $D_{\text{train}}^{(1)}$ and $D_{\text{train}}^{(2)}$ such that $D_{\text{train}}^{(1)}$ having k_2 proportion of D_{train}.
3) Fit $D_{\text{train}}^{(1)}$ with a Random Forest classifier having B trees and find probability estimates $\boldsymbol{p}_{\text{train}}^{(2)}$ for the set $D_{\text{train}}^{(2)}$.
4) Identify the points in $D_{\text{train}}^{(2)}$ having probability estimates lie in the interval (p_1, p_2) and denote that set as D_{svm}.
5) Fit a simple Linear SVM over the set D_{svm} and this concludes the training phase of the algorithm.
6) While testing the model for the set D_{test}, find the probability estimate $\boldsymbol{p}_{\text{test}}$. If the estimate does not lie in the interval (p_1, p_2) predict with the random forest classifier else classify the point with the SVM classifier.

V. Experimentation and Analysis

We took the *wine quality* dataset from UCI repository for analysing our presented ensemble model [8]. The dataset inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent). Quality less than 7 is considered bad and 7-10 as good for our classification problem. The classes are not balanced (there are much more bad wines than good ones). It has 4898 data points with 11 attributes (like fixed acidity, volatile acidity etc.) and an output attribute which is the wine quality (good or bad).

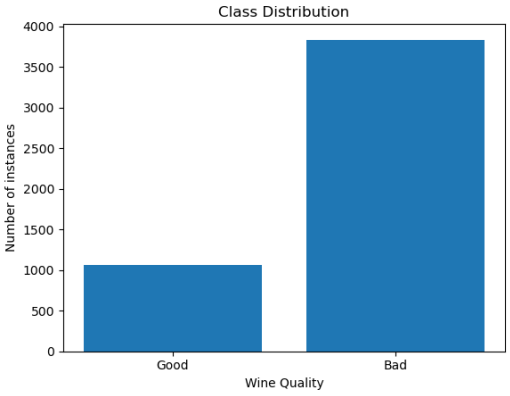

Fig. 1. Class Ditribution

All the models presented here were built from `sklearn` module under Python environment. First, we fit a model with RF classifier which uses 100 trees for classification and gini index as its impurity measure. The results were presented in the Figure 2, showing that the model doesn't seem to work well for this task. We also tried with Linear SVM with regularization parameter $C = 1$, which also yields poor results as shown in the Figure 3.

Class	Precision	Recall	F1-Score	Avg. F1 Score
0	0.33	0.43	0.37	0.60
1	0.84	0.81	0.82	

Fig. 2. RF Model

Class	Precision	Recall	F1-Score	Avg. F1 Score
0	0.02	0.02	0.02	0.44
1	0.87	0.84	0.85	

Fig. 3. Linear SVM Model

Since the data is unbalanced, we can also try with some over sampling methods like Synthetic Minority Over-sampling Technique (SMOTE) [9] which increases the performance of the SVM model significantly (see Figure 4).

Class	Precision	Recall	F1-Score	Avg. F1 Score
0	0.73	0.78	0.75	0.74
1	0.75	0.70	0.73	

Fig. 4. SVM with SMOTE Model

Finally we have provided our ensemble model. For this, first we split the dataset with proportions $k_1 = 0.75, k_2 = 0.70$

where k_1, k_2 being the proportions defined in the above workflow. We then fit the split data with a RF classifier with number of decision trees, $B = 80$ and calculate probability estimates $\boldsymbol{p}^{(2)}_{\text{train}}$. We are then finding the points which are not classified confidently by the RF classifier taken the under confidence interval as $(0.30, 0.70)$ and classify those points with a Linear SVM. The results of our model are shown in the Figure 5.

Class	Precision	Recall	F1-Score	Avg. F1 Score
0	0.94	0.30	0.45	0.69
1	0.85	0.99	0.92	

Fig. 5. Ensembled Model

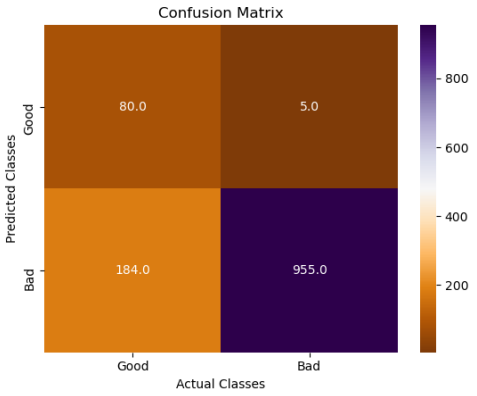

Fig. 6. Confusion Matrix

VI. Conclusion

Our model shows significantly better results with imbalanced data when compared with Linear SVM without SMOTE and Random Forest algorithm. SVM with SMOTE performs better than our model but requires more time and memory which may be a large overhead for large datasets. There is a trade-off between performance and memory when we compare our model with SVM with SMOTE.

References

[1] Jianhong Yan and Suqing Han, *Classifying Imbalanced Data Sets by a Novel RE-Sample and Cost-Sensitive Stacked Generalization Method*, Hindawi, Mathematical problems in Engineering, 2017.

[2] Ethem Alpaydin, *Introduction to Machine Learning*, 3rd edition, MIT Press.

[3] Breiman L, Random forests, *Machine Learning* 45, 5–32 (2001).

[4] Gilles Louppe, *Understanding Random Forests from Theory to Practice*, University of Liege, 2015.

[5] Vapnik V, *Statistical Learning Theory*, Wiley, 1998.

[6] Christopher D, Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[7] MA Aizerman, E Braverman, LI Rozonoer, Avtomat and Telemekh, *Probability problem of Pattern recognition learning and potential functions method*, Moscow, 1964.

[8] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. *Modeling wine preferences by data mining from physicochemical properties* in Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[9] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, Journal of Artificial Intelligence, 2002.