Methods for Handling Highly Skewed Datasets

Sanjeev Suman   Kamlesh Laddhad   Unmesh Deshmukh

Part I - October 3, 2005
Part II - October 6, 2005

1 Literature Survey

1.1 DataBoost-IM approach [7]

This is a boosting based method that combines with synthetic data generation and is used to increase the predictive accuracies of both majority and minority class. In the first stage of the method the hard-to-classify examples are detected for each class and the synthetic examples are generated for each class separately (Seed selection and data generation). More synthetic examples are generated for minority class. In the second stage the frequency of classes in the combined (original + synthetic) dataset are re-balanced to alleviate the learning algorithm's bias towards majority class (Class frequency balancing). In the last stage the total weights of instances of different classes are re-balanced so that the next classifier in the sequence will focus on hard as well as minority class examples (Training weight balancing).

The method iterates for user specified no. of iterations unless the error term exceeds some threshold value. The method performs comparably well for highly imbalanced datasets and importantly does not sacrifice one class for the other but attempts to improve accuracies of major as well as minority class.

1.2 ONE SIDED SELECTION (OSS) [2]

One-sided selection (OSS) is an under-sampling method resulting from the application of Tomek links followed by the application of (closest nearest neighbor) CNN. Tomek links are used as an under-sampling method and removes noisy and borderline majority class examples. Borderline examples can be considered unsafe since a small amount of noise can make them fall on the wrong side of the decision border. CNN aims to remove examples from the majority class that are distant from the decision border. The remainder examples, i.e. safe majority class examples and all minority class examples are used for learning. This is a algorithm specific to KNN (K-nearest neighbor) classification method. This is a very good algorithm which is slow because this uses Tomek Link but very efficient in tackling noise because here we reduce the possibility of noise affecting the decision taken by data.

1.3 SMOTE [5]

Synthetic Minority Over-sampling Technique (Smote) is an over-sampling method. Its main idea is to form new minority class examples by interpolating between several minority-class examples that lie together. Thus, the over-fitting problem is avoided and this causes the decision boundaries for the minority class to spread further into the majority class space.

If the amount of over-sampling needed is 200%, only two neighbors from the k nearest neighbor are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

Over-sampling minority class examples can balance class distributions, but some other problems usually present in data sets with skewed class distributions are not solved. Frequently, class clusters are not well defined since some majority class examples might be invading the minority class space. The opposite can also be true, since interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply in the majority class space. Inducing a classifier under such a situation can lead to over fitting.

Thus, instead of removing only the majority class examples that form Tomek links, examples from both classes are removed. First, the original data set is over-sampled with Smote and then Tomek links are identified and removed, producing a balanced data set with well-defined class clusters.

1.4 TOMEK LINKS [1]

Tomek links can be defined as follows: Given two examples x and y belonging to different classes, and d(x, y) is the distance between x and y. A (x, y) pair is called a Tomek link if there is not an example z, such that $d(x, z) < d(x, y)$ or $d(y, z) < d(x, y)$. If two examples form a Tomek link, then either one of these examples is noise or both examples are border-line. Tomek links can be used as an under-sampling method. As an under-sampling method, only examples belonging to the majority class are eliminated.

The major drawback of Tomek Link under-sampling is that this method can discard potentially useful data that could be important for the induction process. This method uses an algorithm which is very time consuming and having higher order. So this method will run slower then other algorithms.

This is a generic algorithm which can be applied in variety of classification methods. This method is particularly very useful for noisy dataset as this algorithm potentially removes those data which if noise; can lead to error in judgment.

1.5 NEIGHBOURHOOD CLEANING RULE (NCL)[3]

Neighborhood Cleaning Rule (NCL) uses the Wilson’s Edited Nearest Neighbor Rule (ENN)[4] to remove majority class examples.ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. NCL modifies the ENN in order to increase the data cleaning. For a two-class problem the algorithm can be described in the following way: For each example Ei in the training set, its three nearest neighbors are found. If Ei belongs to the majority class and the classification given by its three nearest neighbors contradicts the original class of Ei, then Ei is removed. If Ei belongs to the minority class and its three nearest neighbors misclassify Ei, then the nearest neighbors that belong to the majority class are removed.

This algorithm is specific to KNN (K-nearest neighbor) classification method. This is often appropriate and can provide high generalization accuracy for real - world application, but the storage and computational requirements can be restrictive when the size of the training set is large. Results depend upon the order in which the data comes. This algorithm can be modified so that it will use the full data set and so as to remain unaffected by the order in which the data is coming.

This algorithm removes noisy instances, as well as close border points which in turn smoothes decision boundary slightly. This helps avoiding over-fitting.

2 Part II - Implementation

As second part of this assignment, we have implemented two out of the five techniques (submitted in part 1) for handling highly skewed datasets, namely Synthetic Minority Over-sampling Technique (SMOTE) and Neighborhood Cleaning Rule (NCL). For implementing the methods, we have extended the code of WEKA tool and have written the needed classes.

2.1 SMOTE

SMOTE is an over-sampling approach to handle skewed datasets. In this technique minority classes are oversampled by generating synthetic examples of minority class and adding them to the dataset. That way, the class distribution in the dataset changes and probability of correctly classifying minority class increases.
As it is a pre-processing technique (filter) that is applied on the data, we have created SmoteSample.java class and placed it in the weka.filters.unsupervised.instance package. SmoteSample implements createSubsample() method that carries out the filtering as per the SMOTE technique. In this method we need to find the indices of k nearest neighbors and for this we have coded a method find-KNeighborsIndex (Instance) in MyIBk.java. For generating synthetic samples we have used Random-number based approach wherein the attribute values of the new example are obtained by considering the attribute values of current instance and it's neighbor under consideration scaled by some random number. WEKA Classes/Interfaces used: weka.classifiers.lazy.IBk.java, weka.filters.Filter.java, UnsupervisedFilter and OptionHandler interface.

New Classes Added: MyIBk.java in weka.classifiers.lazy package that extends IBk.java and SmoteSample.java in weka.filters.unsupervised.instance package that extends Filter.java and implements UnsupervisedFilter interface.

Expected benefits: This oversampling technique will increase presence of minority-class examples. This in turn will increase the accuracy of minority-class getting correctly classified.

Actual Findings: We tested our SmoteSample filter on all the three supplied training datasets and obtained predictions for the datasets.
For testing purpose we divided the cora.data dataset into two parts of approximately same size (cora1.data and cora2.data) and used the first half for training purpose and other half for testing purpose. First we ran the J48 classifier using the cora1.data as training set and cora2.data for testing. The confusion matrix that we got is as follows.

	a	b
a	4428	7
b	10	544

Then we filtered the cora1.data using our SmoteSample filter and again ran the J48 classifier using the cora1.data as training set and cora2.data for testing. The confusion matrix that we got this time is

	a	b

a	4423	12
b	5	549

Inferences/Comparisons: From the confusion matrices, we can say that the number of correctly classified minority class instances after using SMOTE on the dataset has increased, but number of correctly classified majority class examples has decreased slightly in this process. Thus SMOTE increases predictive accuracy for minority classes but at the expense of majority class accuracy.

2.2 NEIGHBOURHOOD CLEANING RULE (NCL)

Contrary to SMOTE, Neighborhood Cleaning Rule (NCL) is an under-sampling method for handling the highly skewed datasets. Here approach is to under-sample the majority class examples so that those majority class examples that invade minority class space and lead to misclassification of minority class examples in that region are removed. Then such a filtered dataset is used that is expected to give better minority classification. This can be viewed as data-cleaning as it can also remove noisy examples while performing under-sampling.

This again is a pre-processing (Filtering) technique and so we have placed our MyFilter.java file (This implements the NCL functionality) in the weka.filters.unsupervised.instance package. MyFilter implements the createSubsample() method that carries out the filtering as per the NCL technique. This method also uses the findKNeighborsIndex (Instance) method to find the indices of k nearest neighbors which is implemented in MyIBk.java. WEKA

Classes/Interfaces used: weka.classifiers.lazy.IBk.java, weka.filters.Filter.java,UnsupervisedFilter interface,OptionHandler Interface

New Classes Added: MyIBk.java in weka.classifiers.lazy package that extends IBk.java and MyFilter.java in weka.filters.unsupervised.instance package that extends Filter.java and implements UnsupervisedFilter and OptionHandler interface.

Expected benefits: This under-sampling technique is expected to remove those majority class examples that tend to misclassify the minority class examples by invading into the minority space.

Actual Findings: When we ran our MyFilter filter on the datasets, the results we observed that with parameter N=3 (No. of nearest neighbors to be considered) the amount of filtering that was taking place is negligible. Then we ran the algorithm by increasing 'N' and the no. of majority class samples deleted increased (amount of filtering increased) but by increasing 'N' we are also taking into consideration those instances that are not that close to the current instance. So pruning such instances can mean that our algorithm will be minority-biased.

The possible reason that we could think of for the negligible amount of filtering at small values of 'N' is that in the dataset the minority class examples are very near to each other and very few majority class examples invade that space. We obtained predictions for all the three datasets using MyFilter filter.

Inferences/Comparisons: The NCL technique can indeed help in increasing the prediction accuracy of minority class examples. The performance of NCL technique depends upon how close the minority class examples are to each other in the dataset. The NCL technique can be expected to give good filtering (improved accuracy) if the dataset has majority class instances invading the minority instance space and vice-versa.

3 References

[1] Tomek, I. Two Modifications of CNN. IEEE Transactions on Systems Man and Communications SMC-6 (1976), 769 772.

[2] Kubat, M., and Matwin, S. Addressing the Course of Imbalanced Training Sets: One-sided Selection. In ICML (1997), pp. 179 186.

[3] Laurikkala, J. Improving Identification of Di cult Small Classes by Balancing Class Distribution. Tech. Rep. A-2001-2, University of Tampere, 2001.

[4] Domingos, P. MetaCost: A General Method for [24] Wilson, D. L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Communications 2, 3 (1972), 408 421.

[5] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. JAIR 16 (2002).

[6] A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data 20 Gustavo E. A. P. A. Batista, R. C. Prati, M. C. Monard

[7] Hongyu Guo, Herna L. Viktor : Learning from Imbalanced Datasets with Boosting and Data Generation: The DataBoost-IM Approach