Balanced Sampling Method for Imbalanced Big Data Using AdaBoost

Hong Gu and Tao Song
School of Control Science and Engineering, Dalian University of Technology, Dalian, China

Keywords: Big data, Class Imbalance Learning, Sampling, Boosting, Bagging, Parallel Algorithms.

Abstract: With the arrival of the era of big data, processing large volumes of data at much faster rates has become more urgent and attracted more and more attentions. Furthermore, many real-world data applications present severe class distribution skews and the underrepresented classes are usually of concern to researchers. Variants of boosting algorithm have been developed to cope with the class imbalance problem. However, due to the inherent sequential nature of boosting, these methods can not be directly applied to efficiently handle large-scale data. In this paper, we propose a new parallelized version of boosting, AdaBoost.Balance, to deal with the imbalanced big data. It adopts a new balanced sampling method which combines undersampling methods with oversampling methods and can be simultaneously calculated by multiple computing nodes to construct a final ensemble classifier. Consequently, it is easily implemented by the parallel processing platform of big data such as the *MapReduce* framework.

1 INTRODUCTION

Big data is a dynamic definition whose large quantity and complexity makes it difficult to capture, store, manage, search, share, transfer, mine, analyze and visualize using existing data management tools, data processing applications and data mining approaches (Manyika et al., 2011). It is generated in many scientific and real-world areas such as genomics, proteomics, bioinformatics (Greene et al., 2014), telecommunications, health care, pharmaceutical or financial businesses. Big data is characterized with volume, velocity, variety, veracity, validity, volatility, variability or value (Laney, 2001; del Río et al., 2014). Consequently, it is vital to develop more suitable and faster algorithms to efficiently deal with such massive amounts of data.

What makes this challenge even more difficult is that most of these big data also exhibit imbalanced class distributions, in other words, the number of data that represent one class is much lower than the ones of the other classes. Moreover, the minority class is usually the main interest and more important, i.e., there is always an implicit assumption in class imbalance learning that the minority class has higher cost than the majority class. Imbalanced learning problem has become an important issue in the field of machine learning and a large amount of techniques have been developed trying to address this problem (Chawla et al., 2004; He and Garcia, 2009). Depending on how they deal with class imbalance, these methods can be categorized into three groups: algorithm level approaches, data level techniques and cost-sensitive methods (del Río et al., 2014; Galar et al., 2012). The algorithm level approaches create all new algorithms or modify current existing methods to pay more attention to positive samples (minority class) than negative samples (majority class). Data level techniques consist of a preprocessing step to modify an imbalanced data set in order to provide a balanced distribution, usually by oversampling methods, undersampling methods and hybrid methods (combine oversampling with undersampling) (Estabrooks et al., 2004). Cost-sensitive methods combine the ideas from both data and algorithm level approaches by using different cost matrices to consider costs of misclassifications, and its usual practice is to adopt higher misclassification costs for instances of the minority class and minimize the overall cost (Zadrozny et al., 2003; Sun et al., 2007).

In addition to these methods, a large number of techniques based on ensemble learning have been developed to solve the problem of class imbalance (Galar et al., 2012). Ensemble methods train multiple weak learners and then combine them to construct a strong learner by combination methods such as averaging, voting and learning, etc., and it is well known that an ensemble is usually significantly more accu-

Gu H. and Song T..
Balanced Sampling Method for Imbalanced Big Data Using AdaBoost.
DOI: 10.5220/0005254601890194
In *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms* (BIOINFORMATICS-2015), pages 189-194
ISBN: 978-989-758-070-3

rate than a single learner. Ensemble diversity (the difference among the individual learners) is necessary and plays an important role in ensemble learning, since there would be no performance improvement if identical individual learners were combined. Boosting (Schapire, 1990; Freund and Schapire, 1995) and Bagging (Breiman, 1996) are the most representative among the ensemble learning algorithms and have already achieved great success in many real-world applications. The series of boosting algorithms began to develop from the proof of weak learnable theorem (Schapire, 1990), i.e., the equivalence conjecture of strong and weak learnability (Kearns and Valiant, 1994). Bagging contains two key elements: bootstrap and aggregation. It adopts bootstrap sampling (Efron and Tibshirani, 1994) to generate the data subsets for training different base learners and applies voting or averaging for aggregating the outputs of the base learners.

In general, the members of boosting family are sequential ensemble methods where the base learners are generated sequentially. The basic motivation of sequential methods is to exploit the dependence between the base learners, where the later learners focus more on the mistakes made by the former. While the base learners of parallel ensemble methods represented by bagging are generated in parallel. The basic motivation of parallel ensemble methods is to exploit the independence between the base learners, since the error can be reduced dramatically by combining independent base learners (Zhou, 2012). The parallel architecture of bagging make it easier to achieve fast, scalable and parallel implementations by using *MapReduce* (Dean and Ghemawat, 2008) framework. *MapReduce* is a programming model and an associated implementation for processing and generating large data sets. It abstracts the calculation process in two phases: *Map* divides the original set into independent subsets and distributes them to worker nodes; *Reduce* combines the partial solutions in a way to form the final output. However, there still are class imbalance problems when directly using the *MapReduce* to process the imbalanced big data (because the subsets obtained in the *Map* phase still present severe class distribution skews).

In this work, we present an new technique which combines both bagging and boosting to deal with two-class imbalanced big data, where there is a positive (minority) class,with the lowest number of instances, and a negative (majority) class, with the highest number of instances. Specifically, a balanced resampling method is applied in the data preprocessing phase. It combines random oversampling with random undersampling or SMOTE (Chawla et al., 2011) to divide the original imbalanced set into independent balanced subsets, as well as takes the relationship between the imbalance ratio (IR, the size of majority class divided by that of minority class) and the computing resource (number of computing nodes) into account. The purpose of considering this relationship is to fully exploit useful information of the majority class and increase the ensemble diversity by SMOTE under some computing resource. Then, boosting is running on these independent balanced subsets in parallel simultaneously. Finally, a single ensemble is obtained which is an ensemble of ensembles. To be brief, we use bagging as the main ensemble learning method and train each bag using boosting. Consequently, our method combines the advantages of both boosting and bagging, that boosting mainly reduces bias while bagging mainly reduces variance (Maclin and Opitz, 2011). The experimental results indicate that our method can effectively deal with the class imbalance problem. In addition, due to the parallel nature of our method, it is easier to be implemented on the *MapReduce* platform.

2 MATERIALS AND METHODS

Our method is described below in the following aspects: the AdaBoost algorithm, the Balance-Sampling algorithm, the AdaBoost.Balance algorithm (includes AdaBoost.BSR and AdaBoost.BSS), database and performance evaluation criteria.

2.1 AdaBoost

The AdaBoost (Freund and Schapire, 1995) algorithm is a representative of boosting family which resolves many practical issues of the early boosting algorithm. The input of AdaBoost is a training set of n labeled samples $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in X$ (X represents the instance space), $y_i \in Y$ ($Y = \{-1, +1\}$ is the set of labels). AdaBoost repeatedly calls the given weak learner (base classifier), whose main idea is to maintain a weight distribution on the training set. The weak learners have to be slightly better than a random guess. The weight of sample (x_1, y_1) is denoted as p_i^t in the tth iteration ($t = 1, \ldots, T$, T is the number of iterations). Initially, the weights of all the examples are set to be equal ($1/n$). But the weights of the misclassified instances will increase in each iteration, in order to force weak learner to focus on the difficulties in training set. The task of the weak learner is to find a suitable weak hypothesis: h_t: $X \rightarrow [0,1]$ based on the distribution $\vec{P}^t$. When coping with the two-class classification problem, the range of h_t are two values: $\{-1, +1\}$. Therefore, the learner's task is to minimize the error $\varepsilon_t = \Pr_{i\sim\vec{P}^t}\left[h_t(x_i) \neq y_i\right]$. Once the h_t is obtained, AdaBoost selects a parameter $\alpha_t \subseteq R$, this parameter intuitively measure the correctness or the degree of importance of h_t, i.e., a learner with lower error rate gets higher weight. The final hypothesis H is obtained by combining T weak hypothesis together using weighted majority vote after T cycles. Typically, AdaBoost sets $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ for the two-class classification problem. In general, AdaBoost is an ensemble learning method which iteratively induces a strong classifier from a pool of weak hypotheses.

Algorithm 1: AdaBoost.

```
Input: Training set of n labeled samples
     $D_n = \{(x_1,y_1),\ldots,(x_n,y_n)\}$
     Number of iterations $T$
     Weak learning algorithm WeakLearn
Output: The final classifier $H$
 1: Initialize $w_i^1 = \frac{1}{n}$, $i = 1,\ldots,n$
 2: for $t = 1,\ldots,T$ do
 3:     Set the distribution $\vec{P}^t = \frac{\vec{w}^t}{\sum_{i=1}^n w_i^t}$
 4:     Call WeakLearn, providing it with distribution $\vec{P}^t$, get back a hypothesis $h_t\colon X \to [0,1]$
 5:     Calculate the error of $h_t$:
        $\varepsilon_t = \sum_{i=1}^n p_i^t \left|h_t(x_i) - y_i\right|$
 6:     Set $\beta_t = (1-\varepsilon_t)/\varepsilon_t$
 7:     Set the weight of $h_t$: $\alpha_t = \frac{1}{2}\log(\beta_t)$
 8:     Set the new weights vector to be
        $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$
 9: end for
10: Set the ensemble's threshold: $\theta = \frac{1}{2}\sum_{t=1}^T \alpha_t$
11: Return $H(x) = \mathrm{sgn}\left(\sum_{t=1}^T \alpha_t h_t(x) - \theta\right)$
```

2.2 The Balance-sampling Algorithm

The purpose of using sampling methods in class imbalance problems is to modify an imbalanced data set by some mechanisms to provide a balanced distribution for training a classifier. Among the sampling methods, random undersampling methods randomly remove the majority class examples or randomly select a subset of examples from the original majority class set to make the size of this subset the same as the minority class. Random oversampling methods randomly select examples from the original minority class set and augment original set by replicating the selected examples and adding them to it until the size is equal to that of majority class set. However, randomly removing the majority class examples may lose some useful information, while randomly duplicating the minority class examples may increase the risk of overfitting. To relax those problems, many advanced resampling methods have been developed. For example, the one-sided sampling method (Kubat et al., 1997) selectively remove the majority class examples in order to keep more informative examples, SMOTE generates synthetic examples by randomly interpolating between a minority class example and one of its neighbors from the same class instead of exact copies to reduce the risk of overfitting. In addition, it has been empirically evaluated that neither the oversampling nor the undersampling alone is always the best one to use, and a combining strategy could be useful and effective (Estabrooks et al., 2004). Consequently, we introduce a way to combine the two resampling methods.

The proposed Balance-Sampling algorithm combines random oversampling with random undersampling or SMOTE to divides the original imbalanced set into independent balanced subsets. It determines the size of majority class examples for each subset by the ratio of imbalance ratio R and the number of computing workers M. When $\frac{R}{M} \geq 1$, the size of majority class examples for each computing worker is selected as $|N_k| = \frac{|N|}{M}$, and the size of minority class example subset is determined as $|P_k| = |N_k|$. In other cases of $\frac{R}{M}$, the size of majority class example subset is selected as $|N_k| = \frac{|N|}{R}$, and the size of minority class example subset is the same as that of the original minority class example set. In all cases, union of all the majority class example subsets is able to contains as many majority class examples as possible, i.e., the Balance-Sampling algorithm can fully exploit useful information of the majority class.

In line 5 and 11, we used random under-sampling method to randomly select majority class examples from the original dataset N for N_k until the number $|N_k|$ is achieved. In line 6, according to different under-sampling methods, the Balance-Sampling algorithm can be classified as BSR (Balance-Sampling by Random under-sampling) and BSS (Balance-Sampling by the SMOTE algorithm).

2.3 The AdaBoost.Balance Algorithm

The AdaBoost.Balance algorithm is proposed based on the AdaBoost algorithm and the Balance-Sampling algorithm. It is an ensemble of ensembles, i.e., the Balance-Sampling algorithm is firstly taken as the bootstrap sampling, then AdaBoost is applied to train a base learner on each obtained balanced subset and finally the weighted majority voting is adopted for aggregating the outputs of the base learners. Consequently, our method combines the advantages of both boosting and bagging, that boosting mainly reduces bias while bagging mainly reduces variance. Moreover, because AdaBoost.Balance takes bagging as the main ensemble learning method, it has the same parallel nature as bagging and that makes AdaBoost.Balance easier to be implemented on the *MapReduce* platform.

In line 1, when using the BSR method the AdaBoost.Balance algorithm is denoted as AdaBoost.BSR and it is represented as AdaBoost.BSS when adopting the BSS method.

Algorithm 2: The Balance-Sampling algorithm.

```
Input: The training sets $D = \{P, N\}$
    $P$ is a set of minority class examples
    $N$ is a set of majority class examples
    Number of computing workers $M$
Output: Balanced training sets $\left(D^1_{n^1}, \ldots, D^M_{n^M}\right)$
 1: Calculate the Imbalance Ratio (IR): $R = \frac{|N|}{|P|}$
 2: if $\frac{R}{M} \geq 1$ then
 3:     for $k = 1, \ldots, M$ do
 4:         $|N_k| = \frac{|N|}{M}$, $|P_k| = |N_k|$
 5:         Undersampling $N_k$ from $N$
 6:         Oversampling $P_k$ from $P$
 7:     end for
 8: else
 9:     for $k = 1, \ldots, M$ do
10:         $|N_k| = \frac{|N|}{R}$, $|P_k| = |P|$
11:         Undersampling $N_k$ from $N$
12:         $P_k = P$
13:     end for
14: end if
15: Return $\left(D^1_{n^1}, \ldots, D^M_{n^M}\right)$, where
    $D^k_{n^k} = \{P_k, N_k\}$, $n^k = |P_k| + |N_k|$
```

Algorithm 3: The AdaBoost.Balance.

```
Input: Training set $D = \{P, N\}$
    Number of boosting iterations $T$
    Number of computing workers $M$
    Weak learning algorithm WeakLearn
Output: The final classifier $H$
 1: $\left(D^1_{n^1}, \ldots, D^M_{n^M}\right)$ ←Balance-Sampling$(D)$
 2: for $k = 1, \ldots, M$ do
 3:     $H^k(x)$ ←AdaBoost$\left(D^k_{N^k}, T\right)$
 4: end for
 5: Return
    $H(x) = \text{sgn}\left(\sum_{k=1}^{M} \sum_{t=1}^{T} \alpha_t^k h_t^k(x) - \sum_{k=1}^{M} \theta^k\right)$
```

3 EXPERIMENTS

In this section, we demonstrate the performance of our proposed algorithms in term of AUC (Area Under the receiver operating characteristic Curve) (Bradley, 1997). Eleven UCI data sets (Blake and Merz, 1998) have been used to characterize the efficiency of our methods compared with other methods.

3.1 Performance Evaluation Criteria

It is now well-known that the traditional evaluation criteria such as the sensitivity, precision, recall, specificity and accuracy are not longer appropriate for evaluating the performance of algorithms when there are class imbalance problems, because the minority class would be dominated by the majority class. For example, in credit card fraud detection it is meaningless to achieve high accuracy when a data set whose imbalance ratio is 100, i.e., the number of majority class examples (legitimate transactions) is 100 times that of minority class examples (fraudulent transactions). An algorithm that tries to maximize the accuracy may obtain an accuracy of 99% just by predicting all the examples as majority class, though the accuracy seems high, the predictor is useless since no fraudulent transaction will be detected.

Consequently, we adopt the AUC as the performance evaluation measure, which has been proved to be a reliable performance measure for class imbalance problems (Fawcett, 2004). AUC is the area under the receiver operating characteristic (ROC) curve, where the ROC curve depicts the trade-off between the benefits (tpr, true positive rate) and costs (fpr, false positive rate). AUC provides a single measure of a method's performance to evaluate which one is better (the higher the value of AUC the better the performance of the method). The AUC measure is achieved by calculated the area under the ROC curve.

3.2 Implements and Parameters

Information about 11 UCI data sets is summarized in Table 1: number of attributes (#Attribute), number of examples (Size), number of minority class examples (#Min), number of majority class examples (#Maj), and imbalance ratio (IR). The data sets in Table 1 are ordered according to the ascending order of imbalance ratio. The original Multi-class data sets of UCI were revised to two-class imbalanced problems by combining one or more classes into the minority class and combining the remaining classes into the majority class.

Table 1: Summary Description of the Imbalanced Databases. #Attribute is the Number of Attributes, Size is the Total Number of Examples, #Min and #Maj are the Number of Minority Class Examples and Majority Class Examples Respectively, and IR is the Imbalance Ratio.

Database	#Attribute	Size	#Min	#Maj	IR
wdbc	30	569	212	357	1.7
ionosphere	33	351	126	225	1.8
pima	8	768	268	500	1.9
car	6	1728	518	1210	2.3
haberman	3	306	81	225	2.8
wpbc	33	198	47	151	3.2
mf-zernike	47	2000	200	1800	9.0
mf-morph	6	2000	200	1800	9.0
mf-kar	64	2000	200	1800	9.0
balance	5	625	49	576	11.8
abalone	8	4177	261	3916	15.0

We compared the performance of our proposed algorithms with standard AdaBoost and EasyEnsemble (Liu et al., 2009) in term of AUC. The EasyEnsemble algorithm is also a an ensemble of ensembles. It uses bagging as the main ensemble learning method and trains each bag using AdaBoost. However, it only applies random undersampling method to deal with the class imbalance problem. The number of boosting iterations T was set as 10 and the number of computing workers (or the number of bags) M was selected as 4. We adopted the decision tree (Mitchell, 1997) as the weak learning algorithm. For each data set, we perform a 10-fold cross-validation for all methods. That is, each data set was split into 10 folds, each fold containing 10% of examples of the data set. For each fold, algorithms were trained with the examples contained in the remaining nine folds and then tested with the current fold. The mean and standard deviation of AUC of each method was calculated from the 10 results of this cross validation process.

3.3 Performances of Different Methods

We implemented four methods (AdaBoost, EasyEnsemble, AdaBoost.BSR and AdaBoost.BSS) on a single machine to compare their performance in class imbalance problem. The mean and standard deviation of AUC of all these methods are summarized in Table 2. The mean and standard deviation are presented above and below respectively. The highest average AUC with its standard deviation of each data set is marked bold. Table 2 shows that EasyEnsemble, AdaBoost.BSR and AdaBoost.BSS which are specially designed to deal with class imbalance problem achieve better performance than the standard AdaBoost. The proposed AdaBoost.BSR and AdaBoost.BSS obtain the highest average AUC on all the imbalanced data sets except *pima*. Moreover, AdaBoost.BSS, which combines the random undersampling method with the SMOTE algorithm to cope with the class imbalance problem, shows the superior performance with the increase of imbalance ratio. This is probably due to that the division mechanism of the the Balance-Sampling algorithm can make the AdaBoost.BSR algorithm to fully exploit useful information of the majority class and SMOTE increases the ensemble diversity of it.

Table 2: AUC of the Compared Methods. The Mean and Standard Deviation are Presented Above and Below Respectively. The Highest Average AUC with Its Standard Deviation of Each Data Set is Marked Bold.

	AdaBoost	EasyEnsemble	AdaBoost.BSR	AdaBoost.BSS
wdbc	0.9892	0.9914	**0.9954**	0.9870
	0.0077	0.0083	**0.0066**	0.0134
ionosphere	0.9453	**0.9739**	0.9716	0.9709
	0.0359	**0.0240**	0.0235	0.0398
pima	0.7514	0.8181	**0.8339**	0.8156
	0.0293	0.0268	**0.0362**	0.0499
car	0.9999	0.9999	0.9999	**0.9999**
	0.0002	0.0002	0.0002	**0.0002**
haberman	0.6617	0.6571	**0.6812**	0.6614
	0.0901	0.1484	**0.0842**	0.0975
wpbc	0.6247	0.7387	0.7480	**0.7573**
	0.1338	0.1739	0.0861	**0.1388**
mf-zernike	0.7957	0.9086	0.9011	**0.9113**
	0.0389	0.0174	0.0213	**0.0206**
mf-morph	0.8824	0.9143	0.9216	**0.9258**
	0.0350	0.0170	0.0159	**0.0202**
mf-kar	0.9915	0.9949	0.9956	**0.9957**
	0.0119	0.0046	0.0039	**0.0054**
balance	0.6162	0.5779	0.6417	**0.6707**
	0.1161	0.1399	0.0875	**0.0873**
abalone	0.8134	0.8728	0.8769	**0.8815**
	0.0432	0.0314	0.0239	**0.0259**

4 CONCLUSION

We have developed two parallel ensemble methods (AdaBoost.BSR and AdaBoost.BSS) with the purpose of trying to deal with the imbalanced big data. We use bagging as the main ensemble learning method and train each bag using AdaBoost which combines the advantages of both boosting and bagging. In addition, the proposed Balance-Sampling algorithm, which combines random undersampling with random oversampling or the SMOTE algorithm, is able to fully exploit useful information of the majority class and increases the ensemble diversity. The experimental results indicate that our method can effectively deal with the class imbalance problems, especially the data sets with higher imbalance ratio. Due to the algorithms' parallel structure, our methods are easier to be implemented in parallel. In the future, we plan to implement our proposed methods on the imbalanced big data sets in a parallel distributed *MapReduce* framework to test their efficiency.

REFERENCES

Blake, C. and Merz, C. J. (1998). Uci repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of california. *Department of Information and Computer Science*, 55.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2011). Smote: synthetic minority oversampling technique. *arXiv preprint arXiv:1106.1813*.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

del Río, S., López, V., Benítez, J. M., and Herrera, F. (2014). On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*, volume 57. CRC press.

Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36.

Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31:1–38.

Freund, Y. and Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484.

Greene, C. S., Tan, J., Ung, M., Moore, J. H., and Cheng, C. (2014). Big data bioinformatics. *Journal of cellular physiology*.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

Laney, D. (2001). 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6.

Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(2):539–550.

Maclin, R. and Opitz, D. (2011). Popular ensemble methods: An empirical study. *arXiv preprint arXiv:1106.0257*.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute.

Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Zadrozny, B., Langford, J., and Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 435–442. IEEE.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC Press.