Clustering-based Binary-class Classification for Imbalanced Data Sets

Chao Chen and Mei-Ling Shyu
Department of Electrical and Computer Engineering
University of Miami
Coral Gables, FL 33124, USA
Email: c.chen15@umiami.edu, shyu@miami.edu

Abstract

In this paper, we propose a new clustering-based binary-class classification framework that integrates the clustering technique into a binary-class classification approach to handle the imbalanced data sets. A binary-class classifier is designed to classify a set of data instances into two classes; while the clustering technique partitions the data instances into groups according to their similarity to each other. After applying a clustering algorithm, the data instances within the same group usually have a higher similarity, and the differences among the data instances between different groups should be larger. In our proposed framework, all negative data instances are first clustered into a set of negative groups. Next, the negative data instances in each negative group are combined with all positive data instances to construct a balanced binary-class data set. Finally, subspace models trained on these balanced binary-class data sets are integrated with the subspace model trained on the original imbalanced data set to form the proposed classification model. Experimental results demonstrate that our proposed classification framework performs better than the comparative classification approaches as well as the subspace modeling method trained on the original data set alone.

Keywords: Binary classification, Subspace Modeling, Imbalanced data sets, Clustering.

1 Introduction

Recently, with the prosperity of social networks and the advances of the Internet techniques, a huge amount of multimedia sources, such as videos and images, has posed a great challenge on searching, indexing, and retrieving the data that the users are interested in from the multimedia sources. For example, soccer fans are very interested in fantastic goals made by the soccer players. Therefore, effective goal detection in a large collection of soccer videos is vital to meet the request of soccer fans [4][5][12]. Another example would be the semantic indexing within large video collections. TRECVID [13] semantic indexing task attracts attention from research institutions all over the world. The objective of semantic indexing task is to detect and rank video shots that contain a target concept from a given video collection. Since different concepts are mixed together in the video shots, a popular preprocessing step is to generate a number of binary-class data sets where the target concept is regarded as the positive class and the rest of the concepts are regarded as the negative class.

However, one of the problems in the aforementioned detection task is that the size of positive (concept) class within a binary-class data set is typically much smaller than that of the negative (non-concept) class, so that the positive class becomes the minority class and the negative class is the majority class. This is so-called data imbalance issue [9]. As most of the popular learning algorithms develop their learning models based on the assumption that the classes are balanced, the performance of their learning models is usually not satisfactory when the data set is imbalanced [8].

In this paper, a novel binary-class classification framework is proposed to address such data imbalance issue. First, the original (training) data set is divided into two subsets, one for positive class and one for negative class, based on the class labels. Then, K-means clustering algorithm is applied to cluster the data instances in the negative class subset into K negative groups. Each of the K negative groups is combined with the positive class subset so that new K data groups are formed. For each data group, one subspace model is trained and optimized. Furthermore, these K subspace models are then integrated with the subspace model trained on the original data set to build an integrated model that is expected to render better performance than the subspace model trained on the original data set alone.

The paper is organized as follows. Section 2 introduces the related work. The details of the proposed framework is illustrated in Section 3. The setup and results of the comparative experiment are shown in Section 4. Finally, Section 5

concludes this paper and explores some future directions.

2 Related Work

Broadly, there are two major categories of techniques developed to address the data imbalanced issue. One is data sampling and the other one is boosting [11]. Data sampling can be further divided into oversampling and undersampling [2]. The idea of oversampling is to add more new data instances to the minority class to balance a data set. These new data instances can either be generated by replicating the data instances of the minority class or by applying synthetic methods like Synthetic Minority Over-sampling Technique (SMOTE) [3]. Undersampling differs from oversampling in that it removes data instances of the majority class to balance a data set. There are also approaches that combine both undersampling and oversampling together [10]. The problem with data sampling is that: on one hand, the removal of the data instances of the majority class causes information loss, although the time of model learning is reduced. On the other hand, duplicating or creating data instances of the minority class requires more time for model learning, but the improvement on performance may be negligible [6].

The other category of techniques to handle data imbalance issue is Boosting. Unlike data sampling which directly copes with data imbalance issue on the data level, Boosting methods are designed to reduce the influence of data imbalance on the model level by improving the performance of weak and poor models. The most famous boosting method is AdaBoost [7]. In the training phase, Adaboost reweighs the training data instances and training models iteratively by minimizing the error of prediction produced by an ensemble of models. In the classification phase, a vote of these weighted ensemble models determines the label of each testing data instance. The boosting methods are proved to be effective, but their major drawback is that they usually require a time-consuming iteration process to find the optimal weights.

3 Proposed Framework

Our proposed framework is based on a few subspace models built on several data sets. Here, the framework of subspace modeling is first introduced.

Suppose that the training data set Tr contains R data instances and C attributes. Thus, it can be regarded as a 2-D matrix with R rows and C columns. Each row stands for a data instance which is denoted as $Tr(x,:)$[1], where $x \in [1,R]$. Each column stands for one attribute, which is denoted as $Tr(:,y)$, where $y \in [1,C]$. An element of the 2-D matrix $Tr(x,y)$ means the value of attribute y for data instance x. The testing data set is denoted as Ts, which has the same attributes as the training data set. $Ts[i]$ means the i-th data instance in the testing data set. The training and testing phases of the subspace modeling are shown in Code 1 and Code 2, respectively. Section 3.1 illustrates the definitions of the operators and functions used in the Codes.

[1] we use A(i,:) to denote the i-th row of matrix A, as will be seen later.

3.1 Definitions

Dot operators are defined in Definition 1 to facilitate later expressions. These operators are commonly seen in Matlab.

Definition 1 (Dot operator)**.** ***Dot division*** *(./) between two vectors* $a = \{a_1, \ldots, a_d\}$ *and* $b = \{b_1, \ldots, b_d\}$ *is defined as* $a./b = \{a_1/b_1, \ldots, a_d/b_d\}$. ***Dot multiplication*** *(.*) between two vectors* $a = \{a_1, \ldots, a_d\}$ *and* $b = \{b_1, \ldots, b_d\}$ *is defined as* $a.*b = \{a_1 \times b_1, \ldots, a_d \times b_d\}$.

Definition 2. *The mean vector of a* $m \times n$ *matrix* $A = a(i,j)$ *is defined as* $\mu(A)$, $\mu(A)$=$[\mu_1(A), \ldots, \mu_n(A)]$. *Each element* $\mu_j(A)$ *is calculated by:*

$$\mu_j(A) = \frac{1}{m}\sum_{i=1}^{m} a(i,j), j = 1,2,\ldots,n \quad (1)$$

Definition 3. *The standard deviation vector of a* $m \times n$ *matrix* $A = a(i,j)$ *is defined as s(A),* $s(A)$=$[s_1(A), \ldots, s_n(A)]$. *Each element* $s_j(A)$ *is calculated by:*

$$s_j(A) = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(a(i,j) - \mu_j(A))^2}, j = 1,2,\ldots,n \quad (2)$$

Definition 4 (function Z)**.** *For a* $m \times n$ *matrix* $A = a(i,j)$ *and a* $\rho \times n$ *matrix B,*

$$Z(B,A) = \begin{bmatrix} (B(1,:) - \mu(A))./s(A) \\ \cdot \\ \cdot \\ \cdot \\ (B(\rho,:) - \mu(A))./s(A) \end{bmatrix}$$

Here, $Z(A,A)$ is the z-score normalization result of A.

Definition 5 (SVD)**.** *The standard SVD (Singular Value Decomposition) is shown in Equation (3).*

$$A = U\Sigma V^T. \quad (3)$$

The SVD function of a $m \times n$ *matrix A defined here will further produce two important items,* $\lambda(A)$ *and* $PC(A)$. $\lambda(A)$ *is the positive diagonal elements of* $\Sigma^T\Sigma$ *sorted in a descending manner. In other words,* $\lambda(A)$=$\{\lambda_1(A), \ldots, \lambda_\theta(A) | \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_\theta(A) > 0\}$. *The other item* $PC(A)$ *is the eigenvectors from V that correspond to the sorted* $\lambda(A)$.

CODE 1: SUBSPACE MODELING: LEARNING PHASE

Input:
(1) A set of training data instances Tr
(2) Training labels
Output: $pl^{(opt)}$, $\beta^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$, $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$, $PC(TrN)$

Divide Training data set Tr into positive class TrP and negative class TrN according to training labels.
Calculate $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$ and $s(TrN)$
Calculate $Z(TrP, TrP)$ and $Z(TrN, TrN)$
Perform SVD function on $Z(TrP, TrP)$ to derive the corresponding positive eigenvalues $\lambda(TrP)$ and $PC(TrP)$, and the eigenvectors of $Z(TrP, TrP)^T * Z(TrP, TrP)$. Likewise, derive $\lambda(TrN)$ and $PC(TrN)$
Perform $PCP(Tr, TrP)$ and $PCP(Tr, TrP)$ to get the projected training data on the PC subspaces of positive class and negative class, respectively.
for $pl \leftarrow 1$ to θ
  Calculate $Score(Tr, TrP, pl)$ and $Score(Tr, TrN, pl)$.
  for $\beta \leftarrow init\_value$ to end_value with step s
    Get $FinalScore(Tr, TrP, TrN, \beta, pl)$ by subtracting $\beta \times Score(Tr, TrN, pl)$ from $Score(Tr, TrP, pl)$.
    The data instances with positive $FinalScore$ are predicted as negative. Otherwise, predicted as positive.
    Compute F1-score
  end
end
Output $pl^{(opt)}$ and $\beta^{(opt)}$ corresponding to the best F1-score.
Output $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$ and $s(TrN)$.
Output $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$ and $PC(TrN)$.

Definition 6 (function PCP)**.** *Suppose there is a $m \times n$ matrix $B = b(i, j)$, and eigenvectors $PC(A)=\{PC_1(A), \ldots, PC_\theta(A)\}$, where $PC_i(A)$ is a $n \times 1$ vector, i=1,..., θ, as defined in Definition 5. The Principal Component Projection (PCP) of B on PC(A) is defined as:*

$$PCP(B,A) = \{B * PC_1(A), \ldots, B * PC_\theta(A)\} \quad (4)$$

Definition 7. *Based on Definitions 5 and 6, we can further define the score function $Score(B,A,pl)=[Score_1(B,A,pl), \ldots, Score_x(B,A,pl), \ldots, Score_m(B,A,pl)]^T$, where $Score_x(B,A,pl)$ is defined as:*

$$Score_x(B,A,pl) = \sum_{y=1}^{pl} \frac{PCP_{(x,y)}(B,A) \times PCP_{(x,y)}(B,A)}{\lambda_y(A)}, \quad (5)$$

where pl can be any integer between 1 and θ.

Definition 8. *$FinalScore(T,A,B,\alpha,pl)$ for a vector T is defined as follows, where α is a scalar.*

$$FinalScore(T,A,B,\alpha,pl) = Score(T,A,pl) - \alpha * Score(T,B,pl). \quad (6)$$

CODE 2: SUBSPACE MODELING: CLASSIFICATION PHASE

Input:
(1) Testing data instance $Ts[i]$, i=1 to ω (the total number of testing data instances)
(2) Output from the learning phase: $pl^{(opt)}$, $\beta^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$, $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$, $PC(TrN)$
Output: $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)})$ and predicted label of $Ts[i]$

Calculate $Z(Ts[i], TrP)$ and $Z(Ts[i], TrN)$
Perform $PCP(Ts[i], TrP)$ and $PCP(Ts[i], TrN)$ to get the projected testing data on the PC subspaces of positive class and negative class, respectively.
Calculate $Score(Ts[i], TrP, pl^{(opt)})$ and $Score(Ts[i], TrN, pl^{(opt)})$.
Get $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)})$ by subtracting $\beta^{(opt)} \times Score(Ts[i], TrN, pl^{(opt)})$ from $Score(Ts[i], TrP, pl^{(opt)})$.
if $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)}) \leq 0$
    Predict $Ts[i]$ as positive.
else
    Predict $Ts[i]$ as negative.
end
Output $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)})$ and $Ts[i]$'s predicted label.

From Equation (4), it can be seen that $PCP(B,A)$ is a $m \times \theta$ matrix. If we use $PCP_{(x,y)}(B,A)$ to denote the element of $PCP(B,A)$ at x-th row and y-th column, $PCP_{(x,y)}(B, A)$ is the projection of x-th row vector of B on y-th eigenvector of $PC(A)$.

3.2 Clustering-based subspace modeling

Figure 1 shows the overall framework of CLUstering-based SUbspace MOdeling (CLU-SUMO). In this framework, the negative training data set TrN is clustered into K groups, namely $TrN^{(1)}$, ..., $TrN^{(K)}$. Each of the K groups is combined with TrP to from Group 1, ..., Group K, as shown in Figure 1. Each Group K is modeled by a SUbspace MOdel (SUMO). In the classification phase, the scores of a testing data instance from SUMOs of all groups and the original data set are then integrated together using different weights. The label of that testing data instance is then predicted by checking if the integrated score is greater than zero or not. The learning and classification of CLU-SUMO is shown in Code 3.

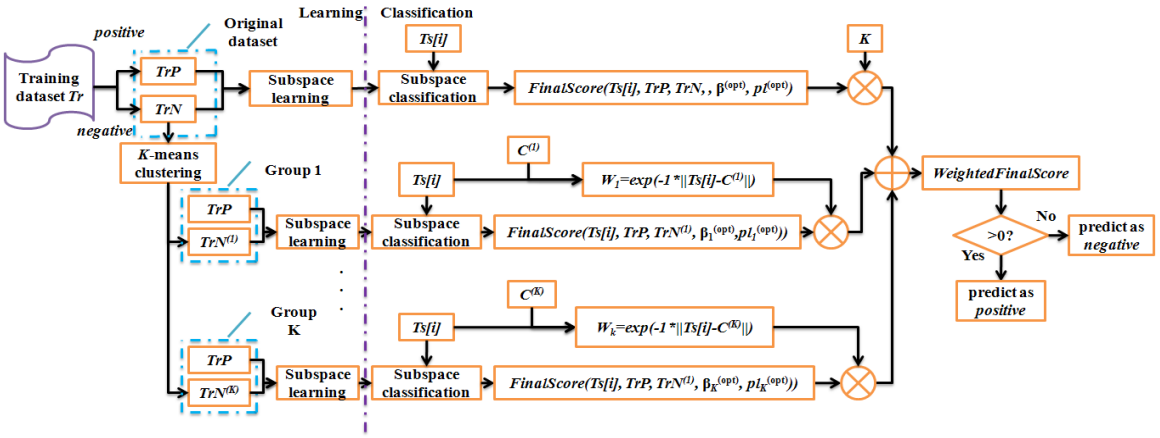

Figure 1. The clustering-based subspace modeling

CODE 3: CLU-SUMO: LEARNING & CLASSIFICATION

Learning Step:
Divide training data set Tr into positive set TrP and negative set TrN.
Apply K-mean clustering method to cluster TrN into K clusters $TrN^{(1)}, \ldots, TrN^{(K)}$. Derive $C^{(1)}, \ldots, C^{(K)}$, which are the centroids of $TrN^{(1)}, \ldots, TrN^{(K)}$.
Build Group j by combining $TrN^{(j)}$ with TrP, $j = 1, \ldots, K$.
Apply subspace learning on original data set as well as on Group 1 to Group K.
Classification Step:
For a testing data instance $Ts[i]$, apply subspace testing using the parameters from subspace learning models and get the $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)})$ and $FinalScore(Ts[i], TrP, TrN^{(j)}, \beta_j^{(opt)}, pl_j^{(opt)})$, $j = 1, \ldots, K$.
Calculate the weight for Group j using W_j=$exp(-1 * ||Ts[i] - C^{(j)}||)$, $j = 1, \ldots, K$.
Calculate $WeightedFinalScore$ = $FinalScore(Ts[i], TrP, TrN, \beta^{(opt)}, pl^{(opt)}) \cdot K$ + $\sum_{j=1}^{K} FinalScore(Ts[i], TrP, TrN^{(j)}, \beta_j^{(opt)}, pl_j^{(opt)}) \cdot W_j$.
if $WeightedFinalScore \leq 0$
    Predict $Ts[i]$ as positive.
else
    Predict $Ts[i]$ as negative.
end

4 Experiments

In order to evaluate the effectiveness of our proposed framework, experiments are conducted using the public available data sources. The proposed framework is also compared with other existing approaches. The experiment setup and results are shown in Section 4.1 and Section 4.2, respectively.

Table 1. Performance of classification on Concept Building

Classifier	Precision	Recall	F1
CLU-SUMO	0.28	0.56	0.37
SUMO	0.33	0.34	0.33
SVM	0.45	0.19	0.27
NB	0.20	0.54	0.30
NN	0.29	0.19	0.23
3-NN	0.24	0.38	0.29
Ada	0.33	0.17	0.22
DTree	0.29	0.28	0.28
MP	0.37	0.31	0.34

4.1 Experiment Setup

The data sets used in the experiments are from the MediaMill Challenge Problem [14], which uses 85 hours of video data from the 2005 NIST TRECVID training and testing sets [1]. There are 5 experiments in the challenge problem and the training and testing data sets in Experiment 1 are used in our experiments. The training data set consists of 30993 data instances and 120 attributes; while the testing data set has 12914 data instances. Five concepts are selected with positive to negative ratio between 0.043 to 0.074. Therefore, these data sets are very imbalanced and thus suitable to prove the effectiveness of our proposed framework.

In the experiments, all classifiers take the same training and testing data sets and the performance from all classifiers is evaluated in terms of F1-score which is the harmonic

Table 2. Performance of classification on Concept Car

Classifier	Precision	Recall	F1
CLU-SUMO	0.25	0.32	0.28
SUMO	0.43	0.19	0.26
SVM	0.34	0.24	0.28
NB	0.08	0.56	0.14
NN	0.28	0.25	0.27
3-NN	0.18	0.36	0.24
Ada	0.41	0.18	0.25
DTree	0.23	0.24	0.24
MP	0.28	0.20	0.23

Table 3. Performance of classification on Concept Meeting

Classifier	Precision	Recall	F1
CLU-SUMO	0.28	0.29	0.29
SUMO	0.39	0.19	0.25
SVM	0.34	0.25	0.28
NB	0.07	0.84	0.12
NN	0.16	0.16	0.16
3-NN	0.13	0.33	0.19
Ada	0.25	0.17	0.20
DTree	0.22	0.16	0.19
MP	0.22	0.35	0.27

Table 4. Performance of classification on Concept Female

Classifier	Precision	Recall	F1
CLU-SUMO	0.15	0.22	0.18
SUMO	0.17	0.15	0.16
SVM	0.18	0.11	0.14
NB	0.03	0.68	0.06
NN	0.08	0.15	0.10
K-NN	0.06	0.32	0.11
Ada	0.18	0.08	0.11
DTree	0.08	0.14	0.11
MP	0.11	0.21	0.14

Table 5. Performance of classification on Concept Military

Classifier	Precision	Recall	F1
CLU-SUMO	0.26	0.35	0.30
SUMO	0.29	0.20	0.24
SVM	0.35	0.17	0.23
NB	0.11	0.70	0.20
NN	0.21	0.16	0.18
K-NN	0.17	0.30	0.22
Ada	0.28	0.08	0.13
DTree	0.18	0.25	0.21
MP	0.28	0.26	0.27

mean of precision and recall.

For SUMO and CLU-SUMO, the *init_value*, *end_value*, and step s of β for our framework are selected as -3, 3, and 0.02, respectively in the learning phase of subspace modeling. In order to balance the positive and negative classes in the generated data groups, K is chosen to be 5 in our experiments so that the positive to negative ratio is, on average, a little more than 1/5. With regard to the classification algorithms used for performance comparison, a list of popular approaches such as Support Vector Machine (SVM), Naive Bayes (NB), Nearest Neighbor (NN), K-Nearest Neighbor (K-NN), Adaboost with C4.5 algorithm (Ada), C4.5 algorithm (DTree), and Multilayer Perceptron (MP) available in Weka [15] are used. These classifiers produce the probability that a testing data instance belongs to the positive class, which is defined as probability of positiveness (PoP) in this paper. The classification rule based on the PoP is shown as follows:

$$\begin{cases} \text{IF PoP} \geq 0.5, \text{THEN assign positive label} \\ \text{IF PoP} < 0.5, \text{THEN assign negative label} \end{cases}$$

On account of the data imbalance issue, we utilize an adaptive threshold τ instead of 0.5 to achieve an equivalent effect as the "reweighting" method. Therefore, the classification rule is modified as follows.

$$\begin{cases} \text{IF PoP} \geq \tau, \text{THEN assign positive label} \\ \text{IF PoP} < \tau, \text{THEN assign negative label} \end{cases}$$

We search τ from 0.1 to 1 with a small step size 0.02 for all the aforementioned comparative algorithms to get their best F1-score on the testing data. In this way, we believe it is more reasonable and fair to compare our proposed framework with these comparative methods using an adaptive threshold.

4.2 Experimental Results and Analyses

The experimental results are shown from Table 1 to Table 5. The results reveal that our proposed framework CLU-SUMO is better than all comparative approaches with regard to all concepts used in the experiments. Table 6 shows that on average CLU-SUMO is at least 3% better than the other comparative methods. On account of low F1-scores in the experiments, the 3% improvement is quite valuable.

Table 6. Average F1 on all 5 Concepts

Classifier	mean F1
CLU-SUMO	0.28
SUMO	0.25
SVM	0.24
NB	0.16
NN	0.19
K-NN	0.21
Ada	0.18
DTree	0.20
MP	0.25

Another contribution of CLU-SUMO is shown in Tables 1, 3, and 5. If the subspace model is trained on the original data set alone, the performance in terms of F1-score may be inferior to Multilayer Perceptron for some data sets. However, if clustering-based subspace modeling is applied, the performance is the best among all the compared classification algorithms. This improvement is obviously from the weighted voting of these $K+1$ subspace models. K subspace models are created on K new data groups, which are more balanced than the original data set. These learning models may capture better positive class patterns than the model trained by only the original data set. However, this statement is true only if K is appropriately selected. If K is too small, then the improvement is not obvious. On the other hand, if K is too large, then within some new data groups, the minority class could now be the negative class and the subspace models may be overfitting to the positive class.

5 Conclusion and Future Work

This paper introduces a new clustering-based binary-class subspace modeling classification framework. Our proposed framework first utilizes the K-means clustering method to cluster the negative training data set into K different negative groups. Then each negative group is combined with the positive training data to construct K new data groups which are more balanced, and each data group trains a subspace model. Our proposed framework is demonstrated to be effective according to comparative experiments with other well-known classification algorithms. For the future work, several directions will be investigated to increase the robustness of the framework. First, experiments pertain to the influence of the cluster size K on the performance of the proposed framework should be conducted. Second, boosting technology can be explored in the learning step of CLU-SUMO to further improve the performance.

References

[1] The mediamill challenge problem (2005).

[2] E. Batista, G, R. C. Batista, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, June 2004.

[3] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, January 2002.

[4] S.-C. Chen, M.-L. Shyu, C. Zhang, and M. Chen. A multimodal data mining framework for soccer goal detection based on decision tree logic. *International Journal of Computer Applications in Technology, Special Issue on Data Mining Applications*, 27(4):312–323, October 2006.

[5] S.-C. Chen, M.-L. Shyu, C. Zhang, L. Luo, and M. Chen. Detection of soccer goal shots using joint multimedia features and classification rules. In *Proceedings of the Fourth International Workshop on Multimedia Data Mining*, pages 36–44, August 2003.

[6] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity:why under-sampling beats over-sampling. In *Proceedings of the International Conference on Machine Learning (ICML 2003) Workshop on Learning from Imbalanced Data Sets II*, pages 1–8, July 2003.

[7] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, July 1996.

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

[9] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–450, November 2002.

[10] Y. Peng and J. Yao. Adaouboost: Adaptive over-sampling and under-sampling to boost the concept learning in large scale imbalanced data sets. In *Proceedings of the international conference on Multimedia information retrieval (MIR '10)*, pages 111–118, March 2010.

[11] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1):185–197, January 2010.

[12] M.-L. Shyu, Z. Xie, M. Chen, and S.-C. Chen. Video semantic event/concept detection using a subspace-based multimedia data mining framework. *IEEE Transactions on Multimedia*, 10(2):252–259, February 2008.

[13] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In *ACM International Workshop on Multimedia Information Retrieval (MIR06)*, pages 321–330, October 2006.

[14] C. Sneok, M. Worring, J. Gemert, J. Geusebroek, and A. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM Multimedia*, pages 421–430, October 2006.

[15] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, second edition, June 2005.