Chapter 14
Integration of Semantics Information and Clustering in Binary-Class Classification for Handling Imbalanced Multimedia Data

Chao Chen and Mei-Ling Shyu

Abstract It is well-acknowledged that the data imbalance issue is one of the major challenges in classification, i.e., when the ratio of the positive data instances to the negative data instances is very small, especially for multimedia data. One solution is to utilize the clustering technique in binary-class classification to partition the majority class (also called negative class) into several subsets, each of which merges with the minority class (also called positive class) to form a much more balanced subset of the original data set. However, one major drawback of clustering is its time-consuming process to construct each cluster. Due to the fact that there are rich semantics in multimedia data (such as video and image data), the utilization of video semantics (i.e., semantic concepts as class labels) to form negative subsets can (i) effectively construct several groups whose data instances are semantically related, and (ii) significantly reduce the number of data instances participating in the clustering step. Therefore, in this chapter, a novel binary-class classification framework that integrates the video semantics information and the clustering technique is proposed to address the data imbalance issue. Experiments are conducted to compare our proposed framework with other techniques that are commonly used to learn from imbalanced data sets. The experimental results on some highly imbalanced video data sets demonstrate that our proposed classification framework outperforms these comparative classification approaches about 3–16 %.

C. Chen (✉) · M.-L. Shyu
Department of Electrical and Computer Engineering, University of Miami, 1251 Memorial Drive, Coral Gables, FL, USA
e-mail: c.chen15@umiami.edu; shyu@miami.edu

Introduction

The success and affluence of social media networks in the Internet have enabled the sharing and distribution of a huge amount of multimedia data in the forms of videos, images, etc. The abundance of multimedia data poses an urgent need on effectively and efficiently searching, indexing and retrieving the data which are of interest to the end users. For example, effective goal detection [6, 7, 20] in a large collection of soccer videos is helpful to the fans who are very interested in the fantastic goals made by the soccer players. Under this circumstance, the goal shots are regarded as positive data instances and the rest of the shots are regarded as negative data instances. TRECVID [21] semantic indexing task is another example in which each video shot is required to be indexed for different semantic concepts. It is a common strategy to build a binary-class data set for each semantic concept where the positive class is composed of the video shots that contain the specified semantic concept and the remaining video shots are categorized into the negative class. Both of the aforementioned examples can be addressed by adopting binary-class classification approaches to detect and rank the video shots related to the positive class.

Binary-class classification aims to separate one target class from a mixture of other non-target classes based on some separation rules or properties. By default, the data instances belonging to the target class are called the positive data instances, and all data instances belonging to the non-target classes as a whole are considered as the negative data instances. However, many popular learning algorithms encounter difficulties when they are directly deployed into these detection tasks because of the so-called data imbalanced issue [14]. In a binary-class data set, the data imbalance issue is usually represented by the negative class dominating the positive class. In details, the size of the positive class is much smaller than that of the negative class. In this case, the majority class is the negative class and the minority class is the positive class. It is not uncommon to see the ratio between the minority class and the majority class on the order of 100:1, 1,000:1, or even 10,000:1. Most of the popular learning algorithms such as Support Vector Machines (SVM) [24] and Neural Networks [19] assume that their models are built on a balanced data set. However, this assumption is often violated in real-world applications. Without any strategy to handle the data imbalance problem, these classification algorithms tend to predict all data instances as the members of the majority class since the learning algorithm is biased towards the majority class. For example, within a data set where 0.1 % of the data instances are the positive data instances and 99.9 % are the negative data instances, all data instances are probably predicted to be negative. This is because that such a misclassification of positive instances only produces a tiny prediction error (probably the minimum prediction error) by the adopted classification algorithm. However, usually these positive data instances are more important than the negative ones, as can be seen by the previous two examples. Therefore, these important positive data instances will most likely be incorrectly predicted in an imbalanced data set.

The data imbalance issue has attracted attentions from many research communities. In Year 2000, the Association for the Advancement of Artificial Intelligence (AAAI) [13] called for a workshop about learning from imbalanced data sets. The International Conference on Machine Learning as well as Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining Explorations (ACM KDD Explorations) also held some similar workshops on the topic of learning from imbalanced data sets [2, 4]. Since then, many algorithms and frameworks have been proposed to handle the data imbalance issue. He and Garcia [11] provided a comprehensive survey of the state-of-the-art solutions to this issue as well as the standardized evaluation metrics to measure the effectiveness for imbalanced data learning.

To address such a data imbalance issue, we propose a novel binary-class classification framework that integrates the semantics information in the multimedia data and the clustering technique in this book chapter. For the binary-class data set built for a target concept, the original training data set is first divided into a positive class subset and a negative class subset, in which the positive class subset is composed of all the training data instances containing the target concept and the negative class subset consists of the remaining data instances. On account of the original imbalanced data set, where the negative class dominates the positive class, the negative class subset is further divided into many negative groups by either clustering or holding out some other non-target concept classes within the original negative class subset so that the ratio of the data size between each negative group and the positive class subset is not large. Therefore, the data groups generated by combining each negative group and positive class subset does not suffer from the data imbalance issue. For each balanced group, a subspace model is trained and optimized. Finally, the subspace models trained on all data groups are integrated with the subspace model built on the original imbalanced data set to form an integrated model which is able to render a better classification performance than the subspace model trained on the original data set alone.

This chapter is organized as follows. Section "Related Work" introduces the related work. The details of the proposed classification framework is illustrated in section "The Proposed Framework". Section "Experiment" demonstrates the setup and results of the comparative experiment, and finally section "Conclusion and Future Work" concludes this chapter and explores some future directions.

Related Work

A number of techniques can be applied to address the data imbalance issue. Data sampling is a common technique to learn from an imbalanced data set. The idea of sampling is to adjust the ratio between the positive data instances and the negative data instances that are used for training the classification models by reducing the number of negative data instances and/or by increasing the number of positive data instances. Therefore, data sampling can be further divided into oversampling and undersampling [1].

Oversampling aims to add more positive data instances to the original imbalanced data set so that the number of the positive data instances is comparable with the number of the negative data instances. New positive data instances can be generated either by simply replicating existing positive data instances (called random oversampling) or by syntactical sampling of the positive (minority) data instances (called Synthetic Minority Oversampling TEchnique (SMOTE)). The idea of oversampling is quite straightforward, that is, to balance the ratio between the number of the positive data instances and the number of the negative data instances without losing any information related to the data instances of both positive and negative classes. However, random oversampling by replicating the positive data instances from the original data set could lead to the overfitting problem, as indicated by Mease et al. [18]. The other oversampling method like SMOTE generates each syntactical positive data instance $X^{(new)}$ between two existing data instances, as shown in Eq. (14.1).

$$X^{(new)} = (1 - \varepsilon) \cdot X^{(i)} + \varepsilon \cdot \hat{X}^{(i)}, \tag{14.1}$$

where $X^{(i)}$ is an arbitrary positive data instance and $\hat{X}^{(i)}$ is randomly picked from the K-nearest neighbors of $X^{(i)} \cdot \varepsilon$ is a random variable between 0 and 1. Similar to random interpolation, it is reasonable to assume that there is a data instance that lies between two existing data instances if they are close to each other. However, over-generalization seems to be a major issue for SMOTE. Therefore, some adaptive synthetic sampling algorithms [10, 12] were proposed to consider the information about neighboring data instances, such as their class labels.

Different from oversampling, undersampling balances the ratio between the positive class and the negative class by removing the data instances belonging to the negative (majority) class. The way that undersampling tries to balance the data set is quite simple and sometimes it is effective. However, some important negative data instances that represent the characteristics of the negative class could be discarded during the undersampling process and the training model could thus be compromised. Data sampling methods directly manipulate the data instances by either increasing or reducing the size of a specified class.

On the other hand, boosting methods handle the data imbalance issue in a different way. The boosting methods acknowledge that the training models built from an imbalanced data set might be not good, but consider the use of an appropriate "re-weighting" of these weak training models to lead to a good classification result. Boosting methods combine weak learning models to reduce the negative influence caused by the data imbalance problem. Among them, AdaBoost [8] is a representative boosting algorithm, which reweighs the training data instances and models iteratively during the training phase by minimizing the prediction error produced by an ensemble of the training models. In the classification phase, the class label of each testing data instance is determined by the voting of these weighted ensemble models. AdaBoost is proved to be effective in many real-world applications. However, the major drawback of AdaBoost as well as other boosting methods is that they usually require a time-consuming iteration process to find the optimal weights for the ensemble models.

There are also algorithms that integrate both boosting methods and sampling methods. For example, SMOTEBoost [3] is built by combining SMOTE and Adaboost.M2 algorithm. SMOTEBoost interactively uses SMOTE at each boosting step and the learning of the positive (minority) class is gradually strengthened and emphasized during the iterative steps. Another approach is the DataBoost-IM [9] method that aims to utilize the boosting procedure to ensure the predictive accuracy values of the positive class and the negative class are both satisfactory. To prevent the training models from overfitting, JOUS-Boost [18] uses the Adaboost algorithm together with over/under-sampling and jittering of the training data.

Another category of approaches that address the data imbalance problem are the cost sensitive learning methods [15, 16, 25, 27]. In these methods, the cost is associated with the misclassification of the positive data instances and the negative data instances are different. In the case where the positive class is dominated by the negative class, the misclassification of the positive data instances should be given a larger cost than that of misclassifying the negative data instances. Studies from [17, 27] showed that cost sensitive learning is able to render better performance than the sampling methods. Algorithms like Cost Sensitive Decision Tree and Cost Sensitive Neural Networks are well studied. However, cost sensitive learning can also be integrated with the other classifiers. One of the problems of cost sensitive learning is the configuration of the cost matrix. Although it is obviously that misclassifying a data instance of the minority class should be given a larger cost, one question arises when it comes to determine how larger the cost value should be. Therefore, it is still a challenging task to find a suitable cost matrix for the cost sensitive learning methods when they are used in an imbalanced data set.

The Proposed Framework

In our previous work [5], a clustering-based subspace modeling method called CLU-SUMO was proposed. CLU-SUMO utilizes K-Means clustering to build K negative data groups from the original negative training subset. Each negative data group is combined with the original positive training subset to generate K training data groups. Subspace modeling method (SUMO) is used to build models on each training data group as well as the original imbalanced data set to predict the ranking scores (soft label) for each testing data instance. Next, a combination of these ranking scores are compared with a decision threshold (the threshold is 0 in CLU-SUMO) to predict the final label of the testing data instance. The CLU-SUMO framework has shown to improve the classification performance with the help of clustering the negative data instances [5].

In this chapter, we further enhance the CLU-SUMO classification framework by integrating semantics information and clustering in the construction of a set of balanced data groups to address the data imbalance issue for multimedia data.

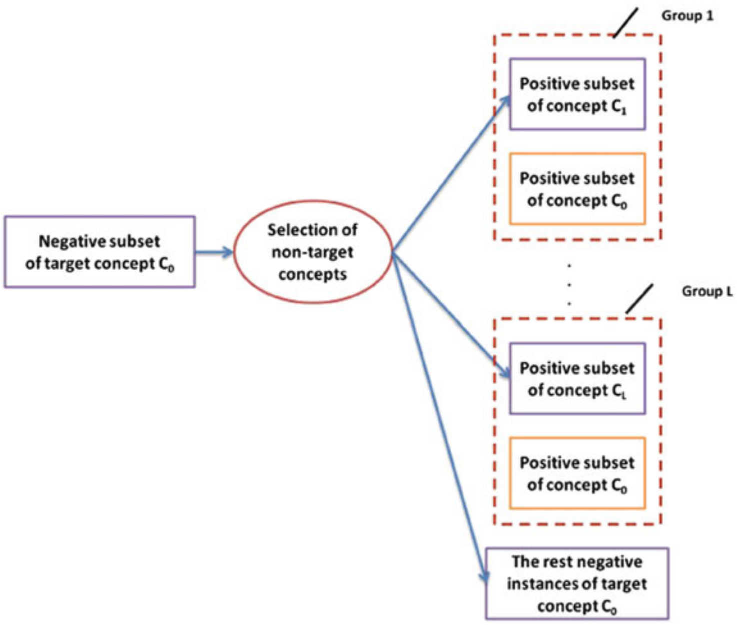
Group 1
Positive subset of concept C_1
Positive subset of concept C_0
Negative subset of target concept C_0
Selection of non-target concepts
Group L
Positive subset of concept C_L
Positive subset of concept C_0
The rest negative instances of target concept C_0

Fig. 14.1 Generation of balanced training subsets by class selection with the utilization of semantics information

In our proposed Class Selection and Clustering based SUbspace MOdeling method (CSC-SUMO), the following enhancements are achieved.

- Speed up the model training procedure. In our proposed framework, the clustering step is applied after some non-target concept classes are held out as negative data groups. The idea behind such a hold-out strategy is that those data instances of the non-target class usually share some common data characteristics and semantics. Since the purpose of applying a clustering method is to find data groups whose data instances share similar data characteristics, from the view of semantics, it is reasonable to regard each non-target concept class as one negative data group though the intra-group similarity cannot be guaranteed to be as small as the one generated by a clustering method.
- Some of the generated data groups hold semantic meanings. Each non-target class corresponds to a particular concept. Therefore, the generated rules that rely on these concepts can help to interpret their meanings. Furthermore, the semantic relationship between concepts can potentially be utilized to help improve the detection results of the target concept.

The proposed CSC-SUMO classification framework consists of three procedures: the generation of balanced training subsets by class selection (as shown in Fig. 14.1), the generation of balanced training subsets by clustering

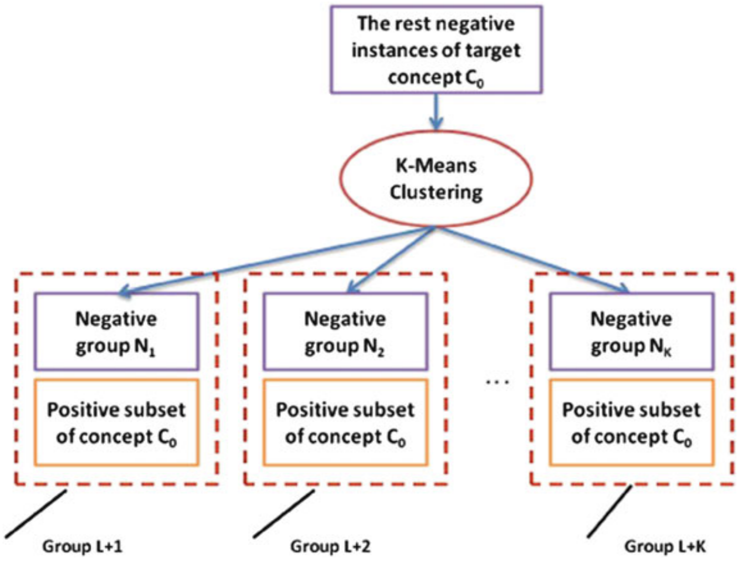
The rest negative instances of target concept C_0
K-Means Clustering
Negative group N_1
Positive subset of concept C_0
Negative group N_2
Positive subset of concept C_0
...
Negative group N_K
Positive subset of concept C_0
Group L+1
Group L+2
Group L+K

Fig. 14.2 Generation of balanced training subsets by clustering

(as shown in Fig. 14.2), and integrated subspace modeling and classification (as shown in Fig. 14.3).

During the first procedure, a number of non-target concepts are selected based on the following pre-defined criteria which are chosen via domain knowledge and empirical studies.

- The ratio of the selected non-target concept class to the target concept class should fall within the interval of [0.5, 2];
- The overlapping of non-target concept class and target concept class should be below 1 %;
- The overlapping between the selected non-target concept classes must below 50 %.

The first criterion ensures that each group in Fig. 14.1 is balanced. The second criterion requires the non-target concept class to overlap with the target concept as small as possible, considering that too much overlapping could make it hard to learn separation rules from the generated balanced groups. The third criterion aims to reduce the number of groups generated by the first procedure. If the overlapping between two non-target concept classes is large, then it is not necessary to generate a data group for each of them since one non-target concept class is already enough to describe the majority of the data instances belonging to the other concept class. Since the selected non-target concept classes may not cover the whole negative subset of the target concept, the size of the remaining negative data instances should be small, which will be the input to the second procedure to cluster them into several data groups.

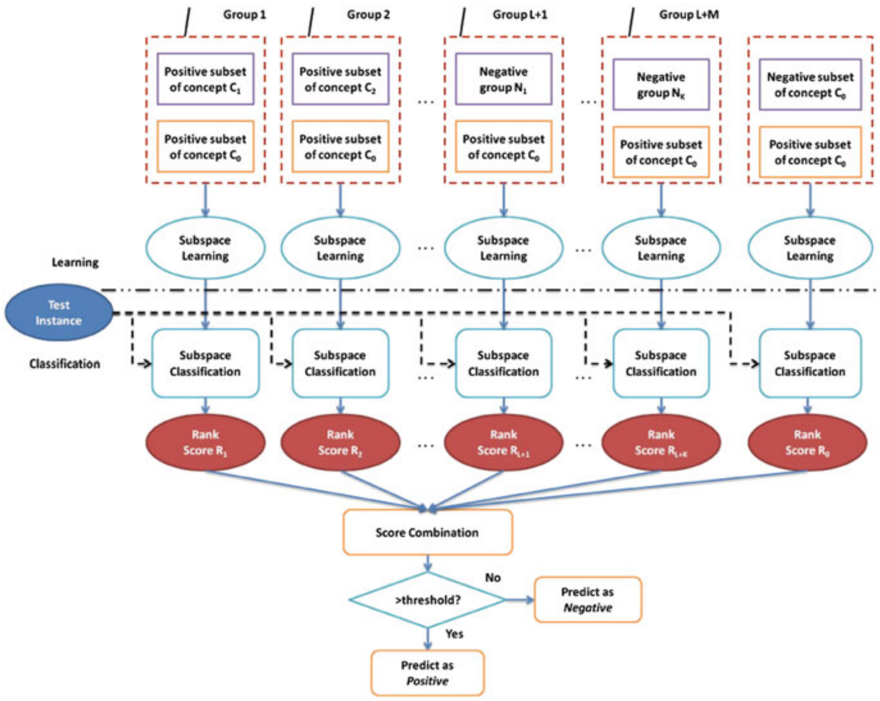

Fig. 14.3 Integrated subspace modeling and classification

The advantages of utilization class label information lie in two folds. First, the efficiency of the clustering-based binary classification framework is enhanced. Second, some negative subsets have semantic meanings, which provides a way to facilitate to interpret the generated rules. On account of this, we propose a new clustering-based binary-class classification framework that integrates both the clustering technique as well as semantic partitioning of negative class to handle the imbalanced data sets.

In the second procedure, K-Means clustering method is used to cluster the remaining negative data instances after the first procedure to form more data groups. This procedure is the same as the one proposed by Chen and Shyu [5]. Till procedure 2, each negative data instance is assigned to one or more data groups since the data groups generated from the first procedure may have some overlapping negative data instances (i.e., those data instances belonging to two or more selected non-target concept classes).

All the balanced data groups and the original imbalanced training data set are trained and optimized by the Subspace Modeling (SUMO) method, as shown in Fig. 14.3. The learning and classification (with the ranking scores) of SUMO is briefly introduced in Code 1 and Code 2, which corresponds to the "Subspace Learning" part and "Subspace Classification" part in Fig. 14.3. Please note that in order to reduce the iterative loops, the parameter β is not used in the learning phase of SUMO as in [5]. The definitions of the functions used in Code 1 and Code 2 are shown as follows.

Definition 1 (function Z). For an $m \times n$ matrix $A = a(i, j)$ and a $\rho \times n$ matrix B,

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

where $\mu(A)$=$[\mu_1(A), \ldots, \mu_n(A)]$ and $s(A)$=$[s_1(A), \ldots, s_n(A)]$ are calculated by Eqs. (14.2) and (14.3), respectively.

$$\mu_j(A) = \frac{1}{m} \sum_{i=1}^{m} a(i, j), j = 1, 2, \ldots, n \tag{14.2}$$

$$s_j(A) = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (a(i, j) - \mu_j(A))^2}, j = 1, 2, \ldots, n \tag{14.3}$$

It can be observed that $Z(A, A)$ is the z-score normalization of A.

Definition 2 (SVD). The standard SVD (Singular Value Decomposition) is shown in Eq. (14.4).

$$A = U \Sigma V^T, \tag{14.4}$$

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

Definition 3 (function PCP). Suppose there are an $m \times n$ matrix $B = \{b(i, j)\}$ and eigenvectors $PC(A)$=$\{PC_1(A), \ldots, PC_\theta(A)\}$, where $PC_i(A)$ is an $n \times 1$ vector, i=1,..., θ, as defined in Definition 2. The Principal Component Projection (PCP) of B on PC(A) is defined as follows.

$$PCP(B, A) = \{B * PC_1(A), \ldots, B * PC_\theta(A)\}. \tag{14.5}$$

Equation (14.5) shows that $PCP(B, A)$ is an $m \times \theta$ matrix. If $PCP_{(x,y)}(B, A)$ is used to denote the element of $PCP(B, A)$ at the x-th row and y-th column, then $PCP_{(x,y)}(B, A)$ is the projection of the x-th row vector of B on y-th eigenvector of $PC(A)$.

Definition 4. Based on Definitions 2 and 3, 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 in Eq. (14.6).

$$Score_x(B, A, pl) = \sum_{y=1}^{pl} \frac{PCP_{(x,y)}(B, A) \times PCP_{(x,y)}(B, A)}{\lambda_y(A)}, \tag{14.6}$$

where pl can be any integer between 1 and θ.

CODE 1: SUMO: LEARNING PHASE

```
1   Input:
    (1) A set of training data instances $Tr$
    (2) Training labels
2   Output: $pl^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$, $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$,
    $PC(TrP)$, $PC(TrN)$
    ________________________________________
3   Divide Training data set $Tr$ into positive class $TrP$ and negative class $TrN$
    according to the training labels.
4   Apply normalization function Z and SVD to positive and negative classes and
    derive the projected data $PCP(Tr, PC(TrP))$ and $PCP(Tr, PC(TrN))$.
5   Iteratively search $pl$ to optimize the F1-Score of the learning model.
6   Output $pl^{(opt)}$ corresponding to the best F1-score,$\mu(TrP)$, $\mu(TrN)$, $s(TrP)$
    and $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$ and $PC(TrN)$.
```

CODE 2: SUBSPACE MODELING: CLASSIFICATION PHASE

```
1   Input:
    (1) Testing data instance $Ts[i]$, $i=1$ to $\omega$ (the total number of testing
    data instances)
    (2) Output from the learning phase: $pl^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$,
    $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$, $PC(TrN)$
2   Output: ranking score of $Ts[i]$
    ________________________________________
3   Derive the projected testing data instance by applying $Z$ function and
    subspace projection to calculate $Score(Ts[i], TrP, pl^{(opt)})$ and $Score(Ts[i],
    TrN, pl^{(opt)})$.
4   Let $S=Score(Ts[i], TrN, pl^{(opt)})+Score(Ts[i], TrP, pl^{(opt)})$
5   Output $(Score(Ts[i], TrN, pl^{(opt)})-Score(Ts[i], TrP, pl^{(opt)}))/S$ as the
    ranking score of $Ts[i]$
```

For a testing data instance *Ts[i]*, the generated ranking scores from the subspaces are combined by a Score Combination module to produce a final ranking score. The final ranking score $R_{final}[i]$ of *Ts[i]* is calculated by Eq. (14.7).

$$R_{final}[i] = (L + K) \cdot R_0 + \sum_{j=1}^{L+K} e^{(-(1+||Ts[i]-C_j||))} \cdot R_j, \tag{14.7}$$

Table 14.1 An example of training data instances

Instance ID	Attribute 1	Attribute 2	$\cdots$	Attribute 120	Class label
1	0.50861	0.50584	$\cdots$	0.42221	0
2	0.44957	0.46049	$\cdots$	0.39756	0
3	0.4168	0.549	$\cdots$	0.43015	0
4	0.50199	0.48082	$\cdots$	0.39877	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
10000	0.42924	0.56815	$\cdots$	0.017	1
10001	0.4448	0.48015	$\cdots$	0.016	1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table 14.2 The centroid of each negative group

Centroid ID	Attribute 1	Attribute 2	$\cdots$	Attribute 120
C_1	0.45888	0.46478	$\cdots$	0.49454
C_2	0.4407	0.50273	$\cdots$	0.45169
C_3	0.46194	0.49503	$\cdots$	0.47173
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
C_{14}	0.33237	0.22438	$\cdots$	0.49799

where C_j is the centroid of the j-th negative data group generated either from the first or the second procedure, and $||\cdot||$ stands for the norm operation. If $R_{final}[i]$ is larger than a threshold value, then *Ts[i]* is predicted as positive. Otherwise, *Ts[i]* is predicted as negative. In the experiment, this threshold value is set to 0.

A running example is given as follows. Assume that the concept to be retrieved is "Car" (i.e., target concept). An example of the input training data instances is shown in Table 14.1, where the instances with class label 1 are positive, which contain the concept "Car"; while those with class label 0 are negative, which are irrelevant to the concept "Car". In the learning phase, suppose Instances 1 and 2 also contain semantic concept "meeting". Assume that the concept "meeting" satisfies the pre-defined criteria to be a negative group (G_1), then Instances 1 and 2 are assigned to the negative group G_1. For convenience, in this example, there is only one negative group formed by the non-target class which is the negative group containing concept "meeting". The remaining negative instances with class label 0 (such as Instances 3 and 4) are clustered into several negative groups (G_2, $\cdots$, and G_{14}) using the K-Means Clustering method, assuming K is 13 here. The centroids of G_1, G_2, $\cdots$, and G_{14} are C_1, C_2, $\cdots$, and C_{14}, respectively (see Table 14.2).

Next, the positive group which consists of all instances whose class labels are 1 is combined with G_1, G_2, $\cdots$, and G_{14} separately to form new binary datasets D_1, D_2, $\cdots$, and D_{14}. Let the original dataset be D_0. In the final step of the learning phase, one subspace model is trained on each binary dataset by following the steps in CODE 1 (e.g., applying normalization in line 4, searching the optimal *pl* value in line 5, and etc.). For example, the subspace model M_0 is built on D_0, M_1 is on D_1, and so on.

Table 14.3 An example of testing data instances

Instance ID	Attribute 1	Attribute 2	⋯	Attribute 120	Class label
1	0.42898	0.54476	⋯	0.40614	N/A
2	0.41305	0.4928	⋯	0.46485	N/A
3	0.47003	0.48959	⋯	0.46499	N/A
4	0.35064	0.54878	⋯	0.49799	N/A
⋯	⋯	⋯	⋯	⋯	⋯

Table 14.4 The ranking scores of testing data instances from different subspace models

Instance ID	M_0	M_1	M_2	⋯	M_{14}
1	−0.0833	0.2732	0.5213	⋯	0.1535
2	−0.0821	0.1738	0.4691	⋯	0.1909
3	−0.1383	0.1686	0.5779	⋯	0.2380
4	−0.0074	0.4092	0.6705	⋯	0.2925
⋯	⋯	⋯	⋯	⋯	⋯

In the testing phase, each testing data instance $Ts[i]$ is input into all subspace models built in the learning phase to get the ranking scores from all subspace models by following the steps in CODE 2 (e.g., applying subspace projection and calculating $Score(Ts[i], TrP, pl^{(opt)})$, and $Score(Ts[i], TrN, pl^{(opt)})$ in line 4, deriving the ranking score of $Ts[i]$ in line 6, etc.). For example, $Score_0[i]$ is $Ts[i]$'s ranking score from model M_0, $Score_1[i]$ is $Ts[i]$'s ranking score from model M_1, and etc. Table 14.3 shows an example of the testing data instances. The ranking score for each testing data instance from all subspace models are listed in Tables 14.4 and 14.5 displays the final ranking scores calculated by Eq. (14.7) and the corresponding predicted labels ("1" for positive and "0" for negative) using 0 as the decision threshold.

Experiment

To show the effectiveness of our proposed framework, experiments are conducted using the public available data sources. The proposed framework is also compared with other popular approaches that are commonly used to handle imbalanced data sets. The setup of the experiment are illustrated in section "Experimental Setup" and the results are shown and discussed in section "Experimental Results".

Experimental Setup

The data sets used for the experiment are from MediaMill Challenge Problem [22], which contains 85 h of news video data [23]. The data set in Experiment 1 of

Table 14.5 The final ranking scores of testing data instances

Instance ID	Final score	Predicted label
1	−0.8785	0
2	−2.1424	0
3	−4.6432	0
4	0.3738	1
...	...	...

Table 14.6 The positive and negative training data instance ratio for the selected concepts

ID	Concept	Positives (P)	Negatives (N)	P-to-N ratio
19	Car	1, 509	29,484	0.051
22	Military	1, 283	29,710	0.043
23	Vegetation	1, 198	19,795	0.040
24	Sports	1, 166	29,827	0.039
26	Graphics	897	30,096	0.030
29	People_marching	597	30,396	0.020
30	Soccer	517	30,476	0.017
34	Screen	475	30,518	0.016

the MediaMill Challenge Problem is used in our experiment, in which the low-level features and class labels are represented by sparse vectors. The training data set and testing data set are divided in advance by the provider. The training data set is composed of a total of 30,993 data instances with 120 attributes; while there are 12,914 data instances in the testing data set with the same number of attributes. A number of concepts corresponding to an imbalanced binary-class data set are selected. The information related to these concepts are shown in Table 14.6. The Positive to Negative (P-to-N) ratio of these concepts varies between 0.016 and 0.051. Therefore, such imbalanced data sets are suitable to evaluate the effectiveness of the proposed framework.

In the experiment, 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 mean of precision and recall, as shown in Eq. (14.8).

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{14.8}$$

For CSC-SUMO, K is carefully chosen so that the ratio of positive data instances to negative data instances is on average 1:2, balancing the positive and negative classes in the generated data groups.

With regard to the classification algorithms used for performance comparison, a list of popular approaches such as Adaboost with C4.5 algorithm (Adaboost), Cost Sensitive Decision Trees (CostDTree) and classic re-sampling method are used, which are available in Weka [26]. The cost matrix CM used by Cost Sensitive Decision Tree is shown below.

Table 14.7 Performance of classification on concept "Car"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	20.28	45.56	28.07
CLU-SUMO	25.42	31.59	28.17
Adaboost	48.20	12.00	19.20
CostDTree	18.30	21.70	19.80
ResampleLG	28.39	26.76	27.55

Table 14.8 Performance of classification on concept "Military"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	23.98	42.24	30.59
CLU-SUMO	26.10	34.94	29.88
Adaboost	32.60	5.30	9.10
CostDTree	17.90	17.10	17.50
ResampleLG	16.32	68.59	26.37

Table 14.9 Performance of classification on concept "Vegetation"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	10.50	67.11	18.16
CLU-SUMO	7.75	83.97	14.20
Adaboost	39.10	7.20	12.10
CostDTree	14.00	16.40	15.10
ResampleLG	37.08	11.02	16.99

$$CM = \begin{bmatrix} 0 & 1 \\ \omega & 0 \end{bmatrix}$$

where ω is set to the negative to positive ratio for each target concept. For a re-sampling method, a logistic regression model is trained on the re-sampled training data set and later is used for predicting the class labels of testing data set. This method is denoted as Re-sampling with Logistic Regression Model (ResampleLG). The re-sampling percentage is tuned according to different data sets. We also compare the proposed method with our previous work (clustering-based subspace modeling method (CLU-SUMO) [5]). The clustering number of CLU-SUMO is chosen based on empirical studies and varies for different concepts.

Experimental Results

The experimental results on the selected eight concepts are shown from Tables 14.7–14.14. The average F1 scores for all classifiers including CSC-SUMO are shown in Table 14.15. The results show that the proposed CSC-SUMO framework outperforms all the comparative approaches in terms of F1 measure (about 3–16 % improvement on average).

Table 14.10 Performance of classification on concept "Sports"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	24.62	33.23	28.28
CLU-SUMO	22.57	34.42	27.26
Adaboost	58.50	11.30	18.90
CostDTree	11.50	20.80	14.80
ResampleLG	18.50	31.45	23.30

Table 14.11 Performance of classification on concept "Graphics"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	45.69	60.13	51.92
CLU-SUMO	30.47	61.69	40.80
Adaboost	75.60	28.30	41.20
CostDTree	34.00	37.40	35.60
ResampleLG	35.35	50.78	41.86

Table 14.12 Performance of classification on concept "People_marching"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	30.79	24.95	27.57
CLU-SUMO	14.24	67.92	23.55
Adaboost	36.70	3.40	6.20
CostDTree	18.20	18.90	18.50
ResampleLG	9.89	75.23	17.48

Table 14.13 Performance of classification on concept "Soccer"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	71.88	60.53	65.71
CLU-SUMO	80.00	52.63	63.49
Adaboost	75.00	55.30	63.60
CostDTree	9.10	65.80	15.90
ResampleLG	12.80	42.11	19.63

Table 14.14 Performance of classification on concept "Screen"

Classifier	Precision (%)	Recall (%)	F1 (%)
CSC-SUMO	64.15	13.88	22.82
CLU-SUMO	88.00	8.98	16.30
Adaboost	82.40	5.70	10.70
CostDTree	6.00	11.40	7.90
ResampleLG	9.62	23.67	13.68

Table 14.15 Average F1 on all concepts

Classifier	Mean F1 (%)
CSC-SUMO	34.14
CLU-SUMO	30.78
Adaboost	22.63
CostDTree	18.14
ResampleLG	23.34

The performance of ResampleLG method seems to be unstable. For example, for some concepts like “Car”, the F1 value of ResampleLG is slightly worse (about 0.5 %) than CSC-SUMO. However, for concept “Soccer”, the F1 value of ResampleLG is much smaller than CSC-SUMO. This is due to the fact that Resampling methods often require an appropriate selection of sampling percentage, which has a large impact on the prediction quality of the classifiers. However, it is often hard to determine such an appropriate sampling percentage.

With regard to CostDTree, there is an inherent problem related to the configuration of the cost matrix. Similar to the re-sampling method, it is also difficult to build a cost matrix that can always render satisfactory classification results. Adaboost method is able to provide better results than CostDTree. However, it requires a time-consuming model training process in order to achieve better results. In the situation when the training time is a major concern, Adaboost may have its limitations.

As shown in our previous work [5] that the weighted voting of all subspace models can improve the classification results. This is mainly because that these K subspace models were built on balanced data sets, and each balanced learning model could learn the patterns belonging to the positive class accompanied by a proportion of data instances of the original negative subset. The selection of K definitely will have an influence on the final classification results. On one hand, a small K value does little help to improve the classification results in terms of F1-measure. On the other hand, a large K value may increase the number of involved training models and could cause the overfitting problem, since too few negative data instances are trained in each subspace model. Besides, compared the results of CSC-SUMO with those of CLU-SUMO, it is interesting to observe that the generation of negative groups in a combination manner between semantic information and clustering seems better than in a purely clustering manner, which implies the semantic information can be utilized to build balanced subsets from the original dataset to improve the effectiveness of semantic concept detection on an imbalanced dataset.

Conclusion and Future Work

This book chapter introduces a novel binary-class subspace modeling classification framework. Our proposed framework utilizes both the video semantics information of the non-target concept class (for class selection) and the K-means clustering method to divide the negative training subset into $L + K$ different negative data groups. Each negative group is combined with the positive training subset to construct $L + K$ new balanced data groups, each of which is trained using the subspace modeling methods. From the experimental results, our proposed framework shows its effectiveness by producing competitive results against the other comparative learning methods for handling the imbalanced data sets.

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, an appropriate classification threshold of CSC-SUMO should be determined dynamically to bring robustness to the proposed framework. The number of selected non-target concepts of each fold in the first procedure could also differ for the target concept class. Therefore, an effective strategy to adaptively determine such an important threshold with respect to each target concept class will be developed.

References

1. Batista GE, Batista RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newsl 6(1):20–29
2. Chawla NV, Japkowicz N, Kolcz A (2003) Workshop learning from imbalanced data sets ii. ACM SIGKDD Explorations Newsletter. In: Proceedings of the ICML'2003 workshop on learning from imbalanced data sets, Washington DC, Aug 2003
3. Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) Smoteboost: improving prediction of the minority class in boosting. In: Proceedings of the seventh European conference on principles and practice of knowledge discovery in databases, Cavtat-Dubrovnik, Sept 2003, pp 107–119
4. Chawla NV, Japkowicz N, Kolcz A (2004) Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explor Newsl 6(1):1–6
5. Chen C, Shyu M-L (2011) Clustering-based binary-class classification for imbalanced data sets. In: The 12th IEEE international conference on information reuse and integration (IRI 2011), Las Vegas, Aug 2011, pp 384–389
6. Chen S-C, Shyu M-L, Zhang C, Luo L, Chen M (2003) Detection of soccer goal shots using joint multimedia features and classification rules. In: Proceedings of the fourth international workshop on multimedia data mining, Washington, DC, Aug 2003, pp 36–44
7. Chen S-C, Shyu M-L, Zhang C, Chen M (2006) A multimodal data mining framework for soccer goal detection based on decision tree logic. Int J Comput Appl Technol, Special Issue on Data Mining Applications 27(4):312–323
8. Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: Proceedings of the 13th international conference on machine learning, Bari, July 1996, pp 148–156
9. Guo H, Viktor HL (2004) Learning from imbalanced data sets with boosting and data generation: the databoost im approach. ACM SIGKDD Explor Newsl 6(1):30–39
10. Han H, Wang WY, Mao BH (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: Advances in intelligent computing, Hefei, pp 878–887
11. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
12. He H, Bai Y, Garcia EA, Li S (2008) Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: IEEE international joint conference on neural networks, Hong Kong, June 2008, pp 1322–1328
13. Japkowicz N (2000) Learning from imbalanced data sets. In: Proceedings of association for the advancement of artificial intelligence, Austin, July–Aug 2000, pp 10–15
14. Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6(5):429–450
15. Liu XY, Zhou ZH (2006) The influence of class imbalance on cost-sensitive learning: an empirical study. In: Sixth international conference on data mining (ICDM'06), Hong Kong, Dec 2006, pp 970–974

16. Maloof MA (2003) Learning when data sets are imbalanced and when costs are unequal and unknown. In: Proceedings of the ICML'2003 workshop on learning from imbalanced data sets, workshop learning from imbalanced data sets II, Washington, DC, Aug 2003
17. McCarthy K, Zabar K, Weiss GM (2005) Does cost-sensitive learning beat sampling for classifying rare classes? In: Proceedings of the 1st international workshop on utility-based data mining, Chicago, Aug 2005, pp 69–77
18. Mease D, Wyner AJ, Buja A (2007) Boosted classification trees and class probability/quantile estimation. J Mach Learn Res 8:18–36
19. Moya M, Hush D (1996) Network constraints and multi-objective optimization for one-class classification. Neural Netw 9(3):463–474
20. Shyu M-L, Xie Z, Chen M, Chen S-C (2008) Video semantic event/concept detection using a subspace-based multimedia data mining framework. IEEE Trans Multimed 10(2):252–259
21. Smeaton AF, Over P, Kraaij W (2006) Evaluation campaigns and TRECVid. In: ACM international workshop on multimedia information retrieval (MIR06), Santa Barbara, Oct 2006, pp 321–330
22. Sneok C, Worring M, Gemert J, Geusebroek J, Smeulders A (2006) The challenge problem for automated detection of 101 semantic concepts in multimedia. In: ACM multimedia, Santa Barbara, Oct 2006, pp 421–430
23. The mediamill challenge problem (2005). Available at http://www.science.uva.nl/research/mediamill/challenge/data.php
24. Vapnik V (1998) Statistical learning theory. Wiley, New York
25. Weiss GM (2004) Mining with rarity: a unifying framework. ACM SIGKDD Explor Newsl 6(1):7–19
26. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco
27. Zadrozny B, Langford J, Abe N (2003) Cost-sensitive learning by cost-proportionate example weighting. In: Third international conference on data mining (ICDM'03), Melbourne, FL, Nov 2003, pp 435–442