Gene Co-AdaBoost: A Semi-Supervised Approach for Classifying Gene Expression Data

Nan Du, Kang Li, Supriya D Mahajan, Stanley A. Schwartz,
Bindukumar B Nair, Chiu Bin Hsiao, Aidong Zhang
State University of New York at Buffalo
{nandu, kli22, smahajan, sasimmun, bnair, chsiao, azhang}@buffalo.edu

ABSTRACT

Co-training has been proved successful in classifying many different kinds of data, such as text data and web data, which have naturally split views. Using these views as feature sets respectively, classifiers could make less generalization errors by maximizing their agreement over the unlabeled data. However, this method has limited performance in gene expression data. The first reason is that most gene expression data lacks of naturally split views. The second reason is that there are usually some noisy samples in the gene expression dataset. Furthermore, some semi-supervised algorithms prefer to add these misclassified samples to the training set, which will mislead the classification. In this paper, a Co-training based algorithm named Gene Co-Adaboost is proposed to utilize limitedly labeled gene expression samples to predict the class variables. This method splits the gene features into relatively independent views and keeps the performance stable by refusing to add unlabeled examples that may be wrongly labeled to the training set with a Cascade Judgment technique. Experiments on four public microarray datasets indicate that Gene Co-Adaboost effectively uses the unlabeled samples to improve the classification accuracy.

Keywords

Gene Co-Adaboost, Co-training, Cascade Judgment, Gene Features Split, Multi-Views

1. INTRODUCTION

Co-training is one of the prominent achievement in classification which was first proposed by Blum and Mitchell [1]. They have proved both theoretically and experimentally that Co-training is able to boost the accuracy of a weak initial classifier trained on a small set of labeled data, while only using sufficient amount of previously unlabeled data [9]. The Co-training is based on the assumption that: the instance space can be naturally split into two separate views and each view is sufficient for perfect classification if there were enough labeled data.

However, in some applications this strict assumption is hard to satisfy, especially the compatibility. Since in some data, such as gene expression data, there may even not exist the naturally split independent feature sets. In this case, Co-training's performance would be limited. Zhou and Li [13] generated three classifiers from the original labeled example set using an algorithm named *Tri-training* with only a single view. Zhou and Goldman [12] proposed an algorithm called *Democratic Co-Learning* in which multiple algorithms instead of multiple views enable learners to label data for each other. Although these methods have relaxed the original strict assumption to some extent, they have not fundamentally solved the assumption problem.

After analyzing the assumption of the Co-training, we realize that the main reason of asking for independent and compatible view is to generate some feature subsets which have large diversity between each other and the combinative predictions made from them could improve the accuracy. Since reasonable diversity could decrease the influence from the noisy samples and the sufficient redundancy guarantees the classifiers could make the prediction based on most confidence choice, therefore, we propose to develop a method of splitting the gene feature set into multiple views which maintain diversity and certain compatibility at the same time.

In this paper we propose a new Co-training algorithm named Gene Co-Adaboost which can be applied to perform classification for gene expression data. Gene Co-Adaboost does not require a naturally split of the gene features set, since it could spit the gene features into multiple relatively independent views which contain the confidence and diversity at the same time. In addition, we assign these views to different weak classifiers respectively in an Adaboost framework which could improve the accuracy and is robust to noise. Thus the process of Adaboost can be viewed as the combinative label propagation over multiple classifiers with multiple views. Besides that, such prediction should also be *reliable* which means that in a multiple rounds prediction judgment process the vast majority of classifiers of each round's classifiers agree on this prediction. This judgment process of whether a prediction should be trusted is made by our Cascade Judgment Module which will be introduced below. Finally, we present experiments on four public gene expression datasets which qualitatively evaluate our method.

The rest of this paper is organized as follows. In Section 2 we describe the principal of our Gene Co-Adaboost algorithm. Section 3 shows and discusses the experimental results. Finally, Section 4 briefly presents our conclusions.

2. GENE CO-ADABOOST METHOD

In this section, we describe the main classification method that we will use in this paper. We start by formally defining the classification problem. Assume that each gene expression dataset can be represented as a matrix $X = [V_{ij}]$. Each V_{ij} describes the expression level of gene j in ith instance. And we assume there are n instances and m genes, therefore A is represented as a matrix of $n * m$. And we also assume in this dataset, the first l instances are labeled $x_i \in X(i \leq l)$ which together is named L, and the rest of $n - l$ instances are unlabeled $x_u \in X(l + 1 \leq u \leq n)$ and together is named U, where $L \cup U = X$. Every instance x_i in L owns a label $y_i \in Y$, where Y is the label set. In our work, we only consider binary cases, thus $Y = \{-1, 1\}$, where 1 represents positive class while -1 represents negative class, respectively.

2.1 Gene Split

The process of Gene Split includes three steps: (i) Rank the genes with Signal to Noise ratio, (ii) Select the informative genes from the gene features, and (iii) Separate the informative genes into multiple views.

Informative genes are the genes that manifest the phenotype structure of the samples [3], which means that if we find the informative genes we then can use them to distinguish the samples effectively. We describe our Gene Split as the following three steps: Firstly, we rank the genes by the signal to noise ratio by Golub [10], which has been proved effectively and widely used in informative gene selection [2]. The formula of the signal to noise ratio for each gene i is shown below:

$$S_i = \frac{\mu_1^i - \mu_{-1}^i}{\sigma_1^i + \sigma_{-1}^i}, \tag{1}$$

where μ_1^i and μ_{-1}^i denote the mean of the positive class samples and negative class samples at gene i, respectively, and σ_1^i and σ_{-1}^i denote the standard deviations of each class. Secondly, we select the K_1 top ranked genes (highly expressed in the positive group) and the K_{-1} bottom ranked genes (highly expressed in the negative group). Now, we have selected $K(K = K_1 + K_{-1})$ informative genes from the original gene set. It is worth notice that vast majority of genes have a low ratio, which means they contribute little to distinguish the samples. On the contrary, the genes with a high ratio are the informative genes playing an important role in distinguishing the samples. Let's see an example in the *Leukemia* [10] gene expression dataset which has 7129 genes. The distribution of each gene's absolute signal to noise ratio is shown in Table 1. It can be easily drawn from this table that few genes are the informative genes.

Table 1: Distribution of Each Gene's absolute ratio

Signal to Noise ratio	# Gene	Percent
0 – 0.3	5610	78.7%
0.3 – 0.6	1327	18.6%
0.6 – 1	179	2.55%
1 – 1.3	12	0.15%

Thirdly, we circularly assign these K genes to j bins, then the genes in each bin is regarded as a view. By this method, on one hand the gene set in each bin has relatively the same capacity to distinguish samples, on the other hand each gene set is relatively independent to the others. The comparison of this algorithm to the other gene split methods is in Section 3.

2.2 Adaboost Process

After the Gene Split given above, we have separated the genes into j views which are denoted as $(V_1, V_2, \ldots V_j)$. Then we assign them to j classifiers $(A_1, A_2, \ldots A_j)$ which would be used as a weak classifier in a Adaboost algorithm which trains j weak classifiers respectively from j different views and compute the confidence of the final classifier. After judging by the Cascade Judgment Module which will be mentioned later, we would decide whether to add the predicted samples to the training set. Such a process would be repeated until all the unlabeled samples have been predicted. In this subsection, we will show how this process performs.

Algorithm 1: The Process of Co-Adaboost

```
Input: $L$:$n$ instances $\langle (x_1, y_1), \ldots (x_n, y_n) \rangle\rangle$ with labels
$y_i \in Y = \{1, -1\}$
$U$: Unlabeled example set
$V$: Set of views, $V = V_1 \cup V_2 \cup \cdots \cup V_j$
$A$: Set of weak classifier, $A = A_1 \cup A_2 \cup \cdots \cup A_j$
Initialize: Set the weight vector $D(0)$ uniform.
  For $i = 1, \ldots n$, where $w_i^1 = 1/n$ for all $i$
Repeat until: $U = \emptyset$
  For $t = 1, \ldots, j$
    Set $P_t \leftarrow \frac{w^t}{\sum_{i=1}^n w_i^t}$
    1.Call Weak classifier $A_t$ providing $P_t, V_t, L$,get
    back a hypothesis $H_t$.
    2.Calculate the error of $H_t : \epsilon_t = \sum_{i=1}^n P_i^t |H_t(x_i) \neq y_i|$
    3.If $\epsilon_t \geq \frac{1}{2}$, then set $t = t - 1$ and abort loop.
    4.Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$
    5.Set the new weights vector to be $D(t)$, where each
    weight of sample $i$
                $w_i^{t+1} = w_i^t \beta_t^{1-|H_t(x_i) \neq y_i|}$
    6.Call Weak classifier $A_t$ providing $P_t, V_t, U$,get
    back a hypothesis $\bar{H}_t$.
  For each instance x ,compute its confidence as
    If $\sum_{t=1}^T \left(log\frac{1}{\beta_t}\right) |H_t(x) = 1| \geq \sum_{t=1}^T \left(log\frac{1}{\beta_t}\right) |H_t(x) = -1|$
      $Con_i = \sum_{t=1}^T \left(log\frac{1}{\beta_t}\right) |H_t(x) = 1|$
    else
      $Con_i = \sum_{t=1}^T \left(log\frac{1}{\beta_t}\right) |H_t(x) = -1|$
  Select $m$ instances with the top $Con_i$,named $\bar{U}$
    $\bar{L} = \text{CascadeJudge}\left(L, \bar{U}, V, A, Q, H\right)$
  Remove $\bar{L}$ from $U$, $L = L \cup \bar{L}$, $\bar{L} \leftarrow \emptyset$
End repeat
```

Like Adaboost, our goal is to find a highly accurate classification rule by combining many weak classifiers, each of which may be only moderately accurate. Let L be a sequence of n instances $\langle (x_1, y_1), \ldots (x_n, y_n) \rangle\rangle$ with labels $y_i \in Y = \{1, -1\}$, where 1 represents positive class while -1 represents negative class respectively. Initially, the weights distribution D is uniform. In each round t, the distribution $D(t)$ and the training set L is given to each weak classifier which computes a weak hypothesis H_t that interprets as a prediction as to what label -1 or 1 is assigned to each unlabeled samples in

U. The measurement of the final combinative prediction is interpreted as a measure of "confidence" in the prediction, which is described below.

Let $Con_i (1 \le i \le n)$ denote the confidence of the sample $i (1 \le i \le n)$. The support from the weak classifiers on the hypothesis that the sample belongs to the negative class is:

$$\sum_{t=1}^{T} \left(log \frac{1}{\beta_t} \right) \left| h_t \left(x \right) = -1 \right|, \tag{2}$$

where T is the number of iterations which can also be considered as the number of the weak classifier, $h_t(x)$ is the hypothesis of sample x by the tth weak classifier, and β_t is chosen as a function of ϵ_t which denotes the error of the hypothesis $h_t(x)$ and is used for updating the weight vector. This equation denotes that we compute the sum of the $log \frac{1}{\beta_t}$ when the weak classifier's hypothesis is negative class. Similarly, the support from the weak classifiers on the hypothesis that the sample belongs to the positive class is:

$$\sum_{t=1}^{T} \left(log \frac{1}{\beta_t} \right) \left| h_t \left(x \right) = 1 \right|. \tag{3}$$

Finally, the confidence would be set as the larger one.

Obviously, the confidence meets its maximum value when the classifiers make a consensus. In each round, we utilize the current labeled dataset L to train the j weak classifiers, and then select the m_k top highest confidence samples. Among these m_k samples, if one is predicted by a consensus from all classifiers, then it would be directly labeled and added to the training set. Otherwise, we would provide this sample to a Cascade Judgment Module which judges weather this sample should be added to the training set. This process is presented in Algorithm 1.

2.3 Cascade Judgment

The cascade structure classifier which was proposed by Paul Viola and Michael Jones in 2001 [8] can rapidly determine where an image or an object might occur in a large image. Although this strategy has been widely used in the face recognition and object detection, it has rarely been used in other fields, especially in the bioinformatics. This is mainly because the situation in bioinformatics is different. In the case of face recognition or object detection, within any single image, an overwhelming majority of sub-windows are negative, which should be rejected. However, in the field of gene expression data, the case is the opposite, where vast majority of prediction are reliable and only few of them may be unreliable. So conditions are different and the methods should also be different. Our intuition is that this cascade judgment strategy can effectively reject the untrustedly labeled samples while adding the trusted labeled samples to the training set.

Each stage is constructed by weak classifiers using AdaBoost. At the beginning, the first stage is constructed by multiple weak classifiers and the hypothesis from the previous AdaBoost which was mentioned in the last section. Intuition tells us that previous prediction also has a reference value. The weight parameter α is designed to give a weight to the hypothesis from the last stage (If we are in the first stage, we should consider the Adaboost's prediction). Then the confidence of each stage's prediction is defined as:

$Conf =$:

$$argmax(\sum_{|h_t(x)=y|}^{T} \left((1-\alpha) log \frac{1}{\beta_t} \right) + \alpha * \bar{h}_t \left| \bar{h}_t \left(x \right) = y \right|), \tag{4}$$

where T is the number of iterations, α is the weight for the previous hypothesis, $h_t(x)$ is the hypothesis of sample x by the tth weak classifier, β_t is chosen as a function of ϵ_t which denotes the error of the hypothesis $h_t(x)$ and is used for updating the weight vector, and $y (\in (-1, 1))$ is the possible label. The confidence of each stage describes the degree of agreement between classifiers and the previous hypothesis. Obviously, the confidence meets its maximum value when all the classifiers and the previous hypothesis make a consensus.

In a certain stage i, if all the r weak classifiers and the prediction from the previous stage make a consensus, then we consider this sample trusted and add it to the training set directly which will be utilized in the future training. If they do not make a consensus, we compare the maximal confidence of this stage with a threshold τ. If the confidence $\le \tau$, we do not add this sample to the training set and stop the cascade judgment process. If confidence $\ge \tau$, we go to the next stage.

Algorithm 2: The Process of Cascade Judgment

```
Input: L:n instances $\langle (x_1, y_1), \ldots (x_n, y_n) \rangle \rangle$ with labels
$y_i \in Y = \{1, -1\}$
$\bar{U}$: $n$ instances Unlabeled example set provided by Adaboost
process mentioned in last section
$V$: Set of views, $V = V_1 \cup V_2 \cup \cdots \cup V_j$
$A$: Set of weak classifier, $A = A_1 \cup A_2 \cup \cdots \cup A_j$
$Q$: Threshold of the current cascade judgment
$H$: The prediction from the last layer or Co-Adaboost
layer: The layer of the current Cascade
Initialize: $\bar{L} \leftarrow \emptyset$
    For each instance in $\bar{U}$,$i = 1, \ldots m$,
      If $H_1 = H_2 = \cdots = H_j$
        $\bar{L} \leftarrow \{\bar{L} \cap (\bar{U}_i, H_1)\}$
      else
        $(\bar{H}, w) \leftarrow Adaboost(L, V, A, \bar{H}_i)$
        $\bar{H}$ is the hypothesis of j weak classifier,
        $w$ is the weight vector of these j weak classifier.
        If $\bar{H}_1 = \bar{H}_2 = \cdots = \bar{H}_j$
          $\bar{L} \leftarrow \{\bar{L} \cap (\bar{U}_i, \bar{H}_1)\}$
        else
          If $Max(Conf) \le Q$
            Remark $Conf$
          else
            $Q \leftarrow AdjustThreshold(Q, layer)$
            $l$=CascadeJudge$(L, \bar{U}_i, V, A, Q, layer + 1)$
              If $l \ne \emptyset$
                $\bar{L} \leftarrow \{\bar{L} \cap (\bar{U}_i, l)\}$
              esle
                Remark $argmax(Conf)$
          End If, End IF, End If, End If
    end of loop
Output: $\bar{L}$
```

In the next stage *i+1*, we not only update τ to a higher value, but also add one more different weak classifier to the Adaboost from the previous stage, which means now we have *r+1* weak classifiers. Since better performance can be anticipated with more classifiers, therefore the classification capacity of stage *i+1* is stronger than stage *i*'s. Similar to the previous stage, if the $r+1$ weak classifiers and the previous prediction from stage i do not make a consensus, we calculate the confidence of stage $i+1$ and compare it with the updated τ, and so on. Obviously, as the stage goes higher, the difficulty to make a consensus and pass the threshold τ is higher. As such, the cascade attempts to reject each sample which could not get enough confidence in a certain stage. If a sample could be made a consensus on a certain stage or pass through the cascade judgment, we have sufficient reasons to believe that the prediction on this sample is reliable. This process is presented in Algorithm 2. Note that "Remark" defined in Algorithm 3 means that the prediction would only be made a record but would not be added to the the training set.

A trusted result from the first stage triggers the evaluation of a second stage whose confidence threshold would be adjusted to a higher value and the prediction from whom would have more confidence since the quantity of weak classifiers increase. A trusted result from the second stage triggers a third stage, and so on.

3. EXPERIMENT RESULTS

Four public gene expression datasets are used in this experiment. Information on these datasets is shown in Table 2. In this table, column "# Gene" shows the number of genes, column "# Instances" shows the number of instances, column "# Class 1" shows the number of positive samples and "# Class -1" shows the number of negative samples. The sum of the "# Class 1" and "#Class -1" equals to "# Instances". Note that all these gene expression datasets' feature sets do not exist naturally split.

Table 2: Experiment Data Description

Data	# Gene	# Instances	# Class 1	# Class -1
Leukemia	7129	72	47	25
Colon	2000	62	40	22
Lung	12533	181	31	150
Ovarian	15154	253	91	162

3.1 Comparing with Other Algorithms

We compared our Gene Co-Adaboost approach with three Semi-Supervised learning algorithms: Tri-training [13], GF (Semi-Gaussian Fields Approach) [11], and Self-training [6]. The purpose of this experiment is to evaluate the performance of our method when it is affected by varying sizes of labeled data. Tri-training is a famous Co-training approach which uses a single-view on three weak classifiers and labels a sample to a weak classifier depending on the other two classifiers. GF uses the graph-based approach: Gaussian random field to do semi-supervised classification on gene expression data. Self-training is a commonly used technique for semi-supervised learning whose classifier trains with the labeled samples repeatedly and add the most confidently predicted sample to the training set in each round [13]. Therefore, each compared algorithm has certain representation.

For the sake of consistency and simplicity, in each dataset's experiment, we begin with selecting 200 informative genes and split them into 3 views, which are randomly assigned to a weak classifier (The weak classifier is chosen from 3-NN, Naive Bayes, ID3 [7] and J4.8 [4]). Henceforth, the range of choosing weak classifiers would be the same. The self-training algorithm we used is similar to [5] (using a Naive Bayes as classifier), while the classifiers label the samples by majority voting. The Tri-training in our experiment is defined the same way as in [13] using a J4.8 as a classifier. Note that all these algorithms use the original feature sets without gene selection. Since GF (Semi-Gaussian Fields Approach) used different gene selection strategies on these four gene expression data in their paper, we keep the same strategies as the paper.

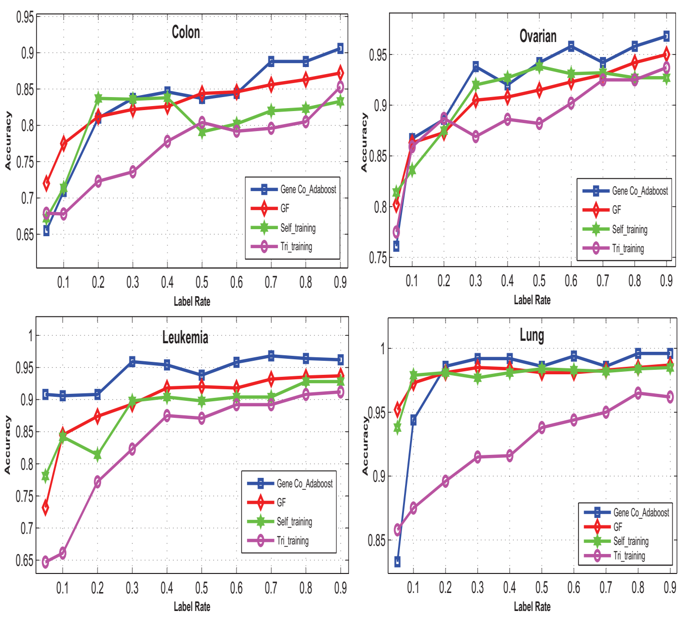

Figure 1: The Result of four Semi-Supervised Algorithms.

As the measurement of performance in this experiment we use *accuracy*. The *accuracy* is defined as $accuracy = (tp + tn)/(tp + tn + fp + fn)$, where tp and tn represent the number of samples which are correctly predicted as positive class and negative class, respectively, and fp and fn represent the number of samples which are falsely predicted as positive class and negative class, respectively. In all the experiments, the entire process is repeated 10 times and the *accuracy* is the average result of these ten experiments. In the experiment of each dataset, we consider various cases when there exists very few labeled examples for training. For each dataset, respectively 5% to 90% data per class are randomly kept as the training set while the rest are used as the test set. Note that our gene split is only based on the training set.

All the results are summarized in Figure 1, which shows the comparison result of Gene Co-Adaboost with GF, Self-training and Tri-training in the four gene expression datasets. Through observing Figure 1 it can be found that, although in two datasets "Lung" and "Colon" Gene Co-Adaboost has a relative low accuracy from the label rate 5% to 20%, after that the accuracy of our algorithm is higher than the others. This suggests that the Cascade Judgment used may play an important role to refuse adding the misclassified samples to the training set, while other algorithms may do. Thus, it can be drawn from these experiments that the Gene Co-

Adaboost approach utilizes the unlabeled samples effectively to improve the accuracy.

3.2 Comparing with Other Gene Split Strategies

As discussed in Section 2.1, we propose the Signal to Noise ratio based feature split method. However, the same Co-adaboost schema can be also applied to other feature split algorithms. In this experiment, we perform further analysis concerning the proposed feature split method by comparing it with three other feature split algorithms: Single-View, Random Split, and Gene Clustering under the same Co-Adaboost framework. By this experiment, we want to test the performance of our Gene Split approach.

The single-view's idea has been proved effective and has been widely used [12, 13], though it is not really a "split" of the feature set. We used Signal to Noise ratio to select 200 genes as we mentioned in Section 2, then treat these 200 genes as a view which are equally assigned to three classifiers in the Co-Adaboost. Some researches have shown that random feature split can be beneficial [1, 6]. Thus we choose it to test our Gene Split's performance. We also select 200 informative genes using Signal to Noise ratio, and then randomly split them into 3 views which are randomly assigned to three classifiers. In our experiment, we define the Gene Clustering strategy as: Firstly, we compute the similarity correlation between each pair of genes using a Pearson correlation metric. Secondly, generating the similarity matrix based on the first step. Finally, three clusters are extracted by a hierarchical clustering algorithm. This method allow us to hypothesize the relationships between genes and classes. Also, this method has been widely used to make gene clustering. All the results are summarized in Figure 2 which shows that our Gene Split approach outperforms the other three approaches while using our Co-Adaboost.

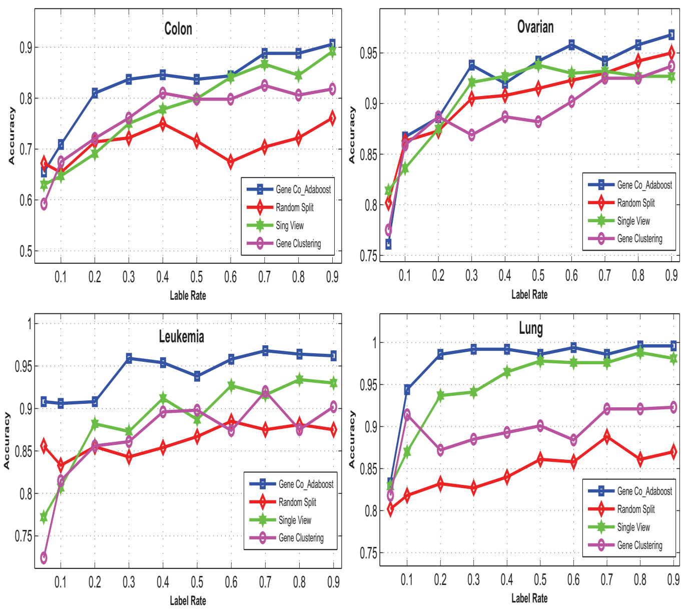

Figure 2: The Result of four Feature Split Approaches.

4. CONCLUSION

In this paper, we have analyzed the challenges of using Co-training approach on gene expression data and we proposed a novel Co-training based classification named Gene Co-Adaboost which not only separates the genes into relatively independent views, but also uses a Cascade Judgment strategy to prevent adding the misclassified samples to the training set. We demonstrated the performance of the proposed approach by experiments on four public gene expression datasets. The experiments result show that our approach is effective and scalable on classifying gene expressing datasets. The classification results consistently show good performance.

5. REFERENCES

[1] A. Blum and T. Mitchell. *Combining labeled and unlabeled data with Co-training* 98. ACM, New York, NY, pp. 92-100, 1998.

[2] Beer DG ,Kardia SL and Huang CC, et al. *Gene-expression profiles predict survival of patients with lung adenocarcinoma* Nat Med 2002;8:816-24.

[3] C. Tang, A. Zhang, and J. Pei. H. *Mining phenotypes and informative genes from gene expression data*, In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[4] I.H. Witten and E. Frank. *Data Mining: Practical Machine LearningTools and Techniques with Java Implementations*, San Francisco, CA:Morgan Kaufmann, 2000

[5] J. Quinlan. *Induction of decision trees*, Machine Learning, 1:81-106, 1986.

[6] K. Nigam and R. Ghani. *Analyzing the Effectiveness and Applicability of CoTraining*. 9th International Conference on Information and Knowledge Management, 2000.

[7] Mitchell, Tom M. *Machine Learning* McGraw-Hill, 1997. pp. 55-58.

[8] P. Viola and M. Jones. *Robust Real Time Object Detection*, IEEEICCV Workshop Statistical and Computational Theories of Vision, July 2001.

[9] Seeger M. *Learning with labeled and unlabeled data*, Technical Report, Umversity of Edinburgh, F_Ainburgh,UK 2001.

[10] T. R. Golub etl. *Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring*, Science, 286:531-537, 1999.

[11] Yun-Chao Gong,et al. *Semisupervised Method for Gene Expression Data Classification with Gaussian Fields and Harmonic Functions*, ICPR 2008. 19th International Conference on Pattern Recognition, 2008.

[12] Zhou and Goldman S. *Democratic colearning* In Procceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence. Washington, DC: IEEE Computer Society Press ,2004.594-602.

[13] Zhou, Z.-H. and Li, M. *Tri-training: Exploiting unlabeled data using three classifiers*, IEEE TKDE, 17(11):1529-1541, 2005.