RSCTC'2010 Discovery Challenge: Mining DNA Microarray Data for Medical Diagnosis and Treatment

Marcin Wojnarski[1,2], Andrzej Janusz[2], Hung Son Nguyen[2], Jan Bazan[3], ChuanJiang Luo[4], Ze Chen[4], Feng Hu[4], Guoyin Wang[4], Lihe Guan[4], Huan Luo[4], Juan Gao[4], Yuanxia Shen[4], Vladimir Nikulin[5], Tian-Hsiang Huang[5,6], Geoffrey J. McLachlan[5], Matko Bošnjak[7], and Dragan Gamberger[7]

[1] TunedIT Solutions,
Zwirki i Wigury 93/3049, 02-089 Warszawa, Poland
marcin.wojnarski@tunedit.org

[2] Faculty of Mathematics, Informatics and Mechanics, University of Warsaw
Banacha 2, 02-097 Warszawa, Poland
andrzejanusz@gmail.com, son@mimuw.edu.pl

[3] Institute of Mathematics, University of Rzeszow
Rejtana 16A, 35-959 Rzeszow, Poland
bazan@univ.rzeszow.pl

[4] Institute of Computer Science and Technology,
Chongqing University of Posts and Telecommunications,
Chongqing 400065, P.R.China
{hufeng,wanggy}@cqupt.edu.cn

[5] Department of Mathematics, University of Queensland
v.nikulin@uq.edu.au, gjm@maths.uq.edu.au

[6] Institute of Information Management, National Cheng Kung University, Taiwan
huangtx@gmail.com

[7] Laboratory for Information Systems, Rudjer Boskovic Institute,
Bijenicka 54, 10000 Zagreb, Croatia,
matko.bosnjak@irb.hr, dragan.gamberger@irb.hr

Abstract. RSCTC'2010 Discovery Challenge was a special event of Rough Sets and Current Trends in Computing conference. The challenge was organized in the form of an interactive on-line competition, at TunedIT.org platform, in days between Dec 1, 2009 and Feb 28, 2010. The task was related to feature selection in analysis of DNA microarray data and classification of samples for the purpose of medical diagnosis or treatment. Prizes were awarded to the best solutions. This paper describes organization of the competition and the winning solutions.

1 Introduction

In recent years, a lot of attention of researchers from many fields has been put into investigation of DNA microarray data. This growing interest is largely motivated by numerous practical applications of knowledge acquired from such data in medical diagnostics, treatment planning, drugs development and many more. When analyzing microarray data, researchers have to face the few-objects-many-attributes problem, as the usual ratio between the number of examined genes and the number of available samples exceeds 100. Many standard classification algorithms have difficulties in handling such highly dimensional data and due to low number of training samples tend to overfit. Moreover, usually only a small subset of examined genes is relevant in the context of a given task. For these reasons, feature extraction methods – in particular the ones based on rough-set theory and reducts - are an inevitable part of any successful microarray data classification algorithm. With RSCTC'2010 Discovery Challenge, the organizers wanted to stimulate investigation in these important fields of research.

The challenge was organized in the form of an interactive on-line competition, at TunedIT (http://tunedit.org) platform, in days between December 1, 2009 and February 28, 2010. The task was to design a machine-learning algorithm that would classify patients for the purpose of medical diagnosis and treatment. Patients were characterized by gene transcription data from DNA microarrays. The data contained between 20,000 and 65,000 features, depending on the type of microarrays used in a given experiment.

Organizing Committee of the challenge had four members: Marcin Wojnarski, Andrzej Janusz, Hung Son Nguyen and Jan Bazan.

2 Organization of the Challenge

Challenge comprised two independent tracks, namely *Basic* and *Advanced*, differing in the form of solutions. In Basic Track, the participant had to submit a text file with predicted decisions for test samples, which was later compared with the ground truth decisions – a typical setup used in other data mining challenges. In Advanced Track, the participant had to submit Java source code of a classification algorithm. The code was compiled on server, the classifier was trained on a subset of data and evaluated on another subset.

On one hand, Advanced Track was more challenging for participants than Basic Track because there were restrictions on the way how the algorithm was implemented – it must have been written in Java, according to API defined by one of three data mining environments: Weka, Debellor or Rseslib. On the other hand, every algorithm have been trained and tested a number of times on the same datasets, using different splits into train/test parts which allowed much more accurate evaluation of solutions. This is particularly important for the problems like DNA microarray data analysis, where datasets are small and evaluation with single train/test split is not fully objective.

Another advantage of Advanced track was the possibility to evaluate not only the accuracy of decisions made by algorithms but also their time and memory complexity. Limits were set for execution of the evaluation procedure, so if the algorithm was too slow or required too much memory, the computation was interrupted with an error. Moreover, after the end of the competition it was possible to disclose the source codes of the solutions at TunedIT server, exactly in the same form that underwent evaluation, to be used by all researchers as a benchmark or starting point for new research. Other details of the challenge setup can be found at http://tunedit.org/challenge/RSCTC-2010-A.

3 Datasets

Twelve microarray datasets from a wide range of medical domains were used in the competition. All of them were acquired from a public microarray repository ArrayExpress[1] (to find out more about the repository see [1]). All microarray experiment results in this repository are stored in MIAME standard and their detailed description as well as previous usage is available on-line. The datasets chosen for the basic track of the challenge are related to diverse research problems: recognition of acute lymphoblastic leukemia genetic subtypes (experiment accession number E-GEOD-13425), diagnostic of human gliomas (accession number E-GEOD-4290), transcription profiling of human healthy and diseased gingival tissues (accession number E-GEOD-10334), transcription profiling of human heart samples with different failure reasons (accession number E-GEOD-5406), recognition of genomic alterations that underlie brain cancer (accession number E-GEOD-9635) and profiling of human systemic inflammatory response syndrome (SIRS), sepsis, and septic shock spectrum (accession number E-GEOD-13904). For the advanced track, selected datasets concerned prediction of response to anthracycline/taxane chemotherapy (accession number E-GEOD-6861), diagnostic of human Burkitts lymphomas (accession number E-GEOD-4475), investigation of a role of chronic hepatitis C virus in the pathogenesis of HCV-associated hepatocellular carcinoma (accession number E-GEOD-14323), profiling of several murine genotypes on subjects stimulated with purified Toll-like receptor agonists (accession number E-TABM-310), recognition of ovarian tumour genetic subtypes (accession number E-GEOD-9891) and recognition of multiple human cancer types (accession number E-MTAB-37).

For the purpose of the competition only the processed versions of the datasets were utilized and no additional microarray normalization was performed. Data preparation was done in *R System*[2] (see [2]). During preprocessing, decision classes of samples were assigned based on the available "*Sample and Data Relationship*" files. Those decision classes, which were supported only by few samples, were removed from data or they were merged with similar classes (e.g. some subtypes of a specific medical condition could have been merged together to form a decision class which is better-represented in data). Any additional information about samples (such as gender, age, smoking habits) was disregarded.

Several precautions were taken to avoid identification of the datasets by contestants. For each decision set, sample and gene identifiers were removed. After that, the samples as well as genes were randomly shuffled and a few samples were taken out with some probability. Finally, gene expression values were divided by the standard deviation of all expression levels in the corresponding sets and the datasets, for which at least one gene fulfilled a criterion that its range was more

[1] www.ebi.ac.uk/arrayexpress

[2] http://www.R-project.org

than 100 times greater than the distance between its first and the third quantile, were logarithmically scaled using the formula:

$$x' = sign(x) * \log(|x| + 1)$$

Brief characteristics of the prepared datasets are given in Table 1.

Table 1. A brief summary of the microarray datasets used in the challenge

Accession number:	no. samples	no. genes	no. classes
E-GEOD-13425	190	22276	5
E-GEOD-4290	180	54612	4
E-GEOD-10334	247	54674	2
E-GEOD-5406	210	22282	3
E-GEOD-9635	186	59003	5
E-GEOD-13904	227	54674	5
E-GEOD-6861	160	61358	2
E-GEOD-4475	221	22282	3
E-GEOD-14323	124	22276	4
E-TABM-310	216	45100	7
E-GEOD-9891	284	54620	3
E-MTAB-37	773	54674	10

In order to provide reasonable baseline scores for participants, three classic features selection methods were combined with 1-Nearest-Neighbor algorithm and used to perform a classification of the test samples from Basic track. The first method was based on the *relief* algorithm (for more details see [3]). This multivariate filter approach measures usefulness of attributes in k-NN classification and can efficiently identify irrelevant features. The gene selection threshold was estimated on the training data using the random probes technique with a probability of selecting an individually irrelevant gene set to 0.05. The irrelevant genes were removed form data and the elimination process was repeated until all the genes that left in a dataset were marked as relevant. The second and the third method were utilizing univariate statistical tests (Pearson's correlation test and the t-test) to filter out unimportant genes (see [4]). For each dataset, the number of selected genes was also estimated using random probes but this time, a desired probability of choosing an irrelevant gene was tuned by *leave-one-out cross-validation* on training examples. The results achieved by the baseline methods were published on the leaderboard during the competition and are summarized in Table 3.

4 Evaluation of Solutions

Solutions were evaluated using a total of 12 datasets from a variety of microarray experiments, each one related to a different medical problem, with different number of attributes and decision classes. Thus, participants had to design algorithms which can be successfully applied to *many* problems of DNA microarrays analysis, not only to one. Evaluation was performed automatically on TunedIT servers using TunedTester application. Every solution underwent two distinct evaluations: *preliminary* and *final*. The results of preliminary evaluation were published on the leaderboard (after they were calculated), while the final results were disclosed after completion of the challenge. Only the final results were taken into account when deciding the winners.

Each of the 12 datasets was assigned to one of the two tracks, thus solutions on every track were evaluated using six datasets, different for each track. The data used on Basic Track were divided into separate training and test sets and were published on the challenge web page. The decisions for samples from the test sets were kept secret and the task was to submit their predictions. On the server, the solutions were compared with expected decisions and their quality was calculated. To avoid bias in the final results caused by overfitting, half of the predictions were used for calculation of the preliminary results, and another half for the final results.

In Advanced Track, all datasets were kept secret, so participants could not access them. Instead, participants could have used public data from Basic Track to test their solutions before submission to the challenge. After submission, the algorithms were evaluated on each dataset with Train+Test procedure applied a number of times. Each Train+Test trial consisted of randomly splitting the data into two equal disjoint parts, training and test subsets, training the algorithm on the first part and testing it on the second. Quality measurements from all trials on a given dataset were averaged. Randomization of data splits was the same for every submitted solution so every algorithm was evaluated on the same splits. The number of repetitions of Train+Test procedure on each dataset was set to 5 for the preliminary evaluation and 20 for the final.

The datasets that were employed on Basic Track in the preliminary and the final evaluation, included: E-GEOD-13425, E-GEOD-4290, E-GEOD-10334, E-GEOD-5406, E-GEOD-9635 and E-GEOD-13904.

The preliminary evaluation on Advanced Track employed 5 datasets: E-GEOD-4475, E-GEOD-14323, E-TABM-310, E-GEOD-9891 and a half of E-MTAB-37 (part A). The final evaluation on Advanced Track employed a total of 6 datasets: 4 datasets from preliminary evaluation, another half of E-MTAB-37 and a new dataset, not used in preliminary evaluation: E-GEOD-6861, E-GEOD-4475, E-GEOD-14323, E-TABM-310, E-GEOD-9891 and E-MTAB-37 (part B).

Datasets from medical domains usually have skewed class distributions, with one dominant class represented by majority of samples and a few minority classes represented by small number of objects. This was also the case in this challenge. Typically, minority classes are more important than the dominant one and this fact should be reflected by the quality measure used to assess the performance of algorithms. For this reason, solutions were evaluated using *balanced accuracy* quality measure. This is a modification of standard classification accuracy that is insensitive to imbalanced frequencies of decision classes. It is calculated by computing standard classification accuracies (acc_k) for every decision class and then averaging the result over all classes ($k = 1, 2, \ldots, K$). In this way, every class has the same contribution to the final result, no matter how frequent it is:

$$S_k = \#\{i : class(sample_i) = k\}$$

$$acc_k = \#\{i : prediction(sample_i) = class(sample_i) = k\}/S_k$$

$$BalancedAcc = (acc_1 + acc_2 + \ldots + acc_K)/K$$

In the case of 2-class problems with no adjustable decision threshold, balanced accuracy is equivalent to *Area Under the ROC Curve* (AUC). Thus, it may be viewed as a generalization of AUC to multi-class problems.

In the competition, the balanced accuracy of algorithms was calculated separately for each dataset used on a given track and then the results were averaged.

In the evaluation of the Advanced Track, not only accuracy, but also time-and-memory-efficiency of algorithms were considered. A time limit was set for the whole evaluation: 5 hours in the preliminary tests and 20 hours in the final tests. Therefore, a single Train+Test trial of the algorithm lasted, on average, no longer than 60 minutes. The memory limit was set to 1,500 MB, both in preliminary and final evaluation. Up to 450 MB was used by evaluation procedure to load a dataset into memory, so 1 GB was left for the algorithms. Tests were performed on a station with 1.9 GHz dual-core CPU, 32-bit Linux and 2 GB memory, running Sun Java HotSpot Server 14.2 as a JVM.

5 Results

There were 226 participants registered to the challenge. The number of *active participants* – the ones who submitted at least one solution – was 93 for Basic Track and 29 for Advanced Track. If a participant made more than one submission, the last solution was considered as the final one. The first 3 winners on each track, together with baseline results, are presented in Tables 2 and 3.

After the challenge, we calculated the results that would be obtained by an ensemble made of a number of top solutions from Basic Track, through a simple voting. These results are presented in Table 3. Combining 7 top solutions gave significantly higher accuracy than that of the best individual algorithm. We also constructed an ensemble of 54 solutions whose individual performances were better than the best baseline and an ensemble of 92 solutions which achieved higher rank than a "majority vote" classifier.

TunedIT awarded the first winners on both tracks with money prizes: 2,000 USD on Advanced track and 1,000 USD on Basic track. Additionally, conference registration fees for two participants, one from each track, were covered.

All employed datasets and the source code of evaluation procedures are available at TunedIT Repository[3] so new algorithms can be tested against challenge data, using the same experimental setup. In this way, the challenge contributed to creation of benchmark datasets that can be reused in the future by the whole scientific community.

In the following sections, the winners briefly describe their approaches.

[3] Links can be found at http://tunedit.org/challenge/RSCTC-2010-A

Table 2. Final results of the Advanced Track

Rank	Participant or Team	Username	Final Result
1	ChuanJiang Luo, Ze Chen, Feng Hu, Guoyin Wang, Lihe Guan, Inst of Computer Science and Technology, Chongqing Univ of Posts & Telecomm, China	RoughBoy	0.75661
2	Huan Luo, Juan Gao, Feng Hu, Guoyin Wang, Yuanxia Shen, Inst of Computer Science and Technology, Chongqing Univ of Posts & Telecomm, China	ChenZe	0.75180
3	wulala	wulala	0.75168

Table 3. Final results of the Basic Track

Rank	Participant or Team	Username	Final Result
1	Vladimir Nikulin, Dept. of Mathematics, University of Queensland, Australia	UniQ	0.73870
2	Matko Bošnjak, Dragan Gamberger, Rudjer Boskovic Institute, Croatia	RandomGuy	0.73485
3	Ryoji Yanashima, Keio University, Japan	yanashi	0.73108
–	Baseline_relief-1NN	–	0.65122
–	Baseline_corTest-1NN	–	0.64087
–	Baseline_tTest-1NN	–	0.63464
–	Baseline: majority classifier	–	0.28056
–	Ensemble of top 7 final solutions	–	0.7469
–	Ensemble of top 54 final solutions	–	0.7113
–	Ensemble of top 92 final solutions	–	0.6870

6 The Best Solution from Basic Track: Feature Selection with Multi-class Wilcoxon Criterion Applied to Classification of High-Dimensional Microarray Data

6.1 Initial Model and Evaluation Scheme

Initially, we decided to conduct some experiments with the Nearest Shrunken Centroids (NSC) method as it is described in [5]. The NSC method may be viewed as a sequence of two steps FS+Model, (i) feature selection (FS) and (ii) classification. The FS step depends on a very important parameter Δ, which should be selected specially for the particular dataset. As far as we were dealing with the case of a small sample size, LOO (leave-one-out) was the most appropriate evaluation scheme:

$$FS + LOO(Model). \tag{1}$$

With this scheme, we can consider several parameter settings, and the system will select the most appropriate setting depending on the LOO evaluations.

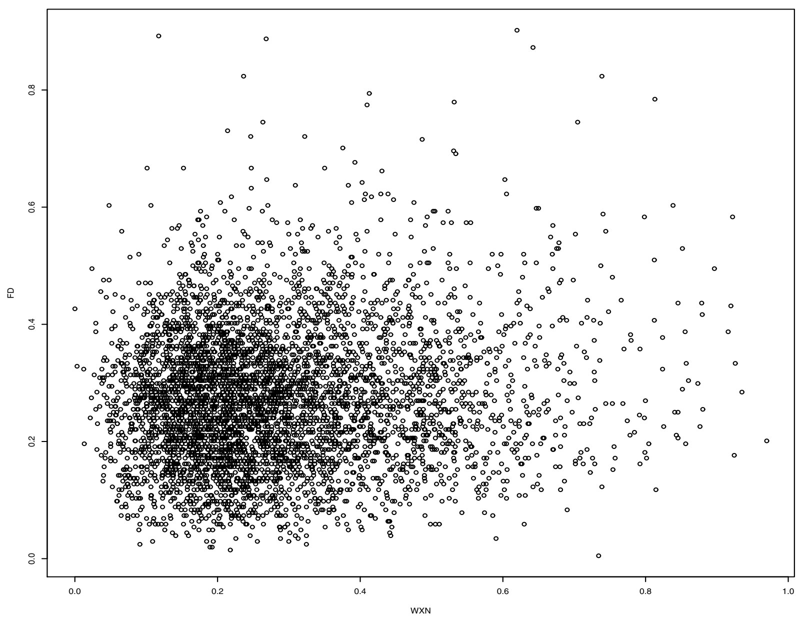

Fig. 1. Relation between WXN and FD scoring functions, where the most simplest Set3 was used as a benchmark

6.2 Wilcoxon and Fisher Discriminant Criterions for Feature Selection

Let us denote by $\mathcal{N}_a$ a set of all samples/tissues within the class a. The following criterion (named Wilcoxon) was used for the selection of the most relevant features

$$WXN(feature) = \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \max(q_{ab}(feature), q_{ba}(feature)), \tag{2}$$

where

$$q_{ab}(feature) = \sum_{i \in \mathcal{N}_a} \sum_{j \in \mathcal{N}_b} I(feature_i - feature_j \leq 0),$$

where I is an indicator function.

In addition, we conducted some experiments with Fisher Discriminant criterion:

$$FD(feature) = \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \frac{|\mu_a(feature) - \mu_b(feature)|}{s_a(feature) + s_b(feature)}, \tag{3}$$

where $\mu_a(feature)$ and $s_a(feature)$ are mean and standard deviation of *feature* within the class a. Note that both criterions WXN and FD were normalised to the range $[0, \dots, 1]$.

Figure 1 illustrates significant structural difference between the WXN and FD criterions. We used an ensemble constructor as it is described in [6] to create an ensemble (named ENS) of WXN and FD criterions, and gained some improvement in application to Set1.

FS was conducted according to the rule:

$$ENS(feature) \geq \Delta > 0.$$

6.3 Classification Models

Very simple and fast classification (FS+FD). The following classification rule may be viewed as a simplification of the NSC:

$$decision = \underset{a}{\operatorname{argmin}} \sum_{feature=1}^{p} \frac{|feature(new.sample) - \mu_a(feature)|}{s_a(feature)}, \quad (4)$$

and may be used immediately after FS step. Note that LOO evaluation and validation results, which we observed with (4), were better compared to the NSC model. It is easy to see a close relation between (3) and (4). Consequently, we use abbreviation FS+FD for the model (4).

Table 4. Statistical characteristics of the solution, which was produced using ENS+FD method, where p_S is the number of selected features; LOO values are given in terms of the balanced accuracy; n1, n2, n3, n4, n5 are the numbers of decisions per class

N	Δ	p_S	LOO	n1	n2	n3	n4	n5
1	0.876	171	0.9088	86	37	-	-	-
2	0.801	44	0.8273	43	54	7	-	-
3	0.55	1542	0.9765	21	7	24	26	16
4	0.663	1031	0.5433	10	32	39	22	9
5	0.845	123	0.7432	17	22	29	21	-
6	0.731	679	0.7127	19	6	14	41	12

Validation experiments. During validation trials we conducted experiments with several classification models, including our own (translated from C to JAVA), CLOP (Matlab)[4] and, also, models from the Weka package[5].

In the case of the basic track the best validation result **0.76** was produced with ENS+FD (Set 1), WXN+FD (Set 5), WXN+MLP(Weka) (Sets 2-4, 6), where MLP stands for multilayer perceptron.

The top validation result **0.8089** for the advanced track was produced with WXN+MLP(Weka), where we used fixed $\Delta = 0.67$ for FS and

$$ops = \{``-L", ``0.25", ``-N", ``1200", ``-H", ``11", ``-M", ``0.1"\}$$

- settings for MLP(Weka).

[4] http://clopinet.com/CLOP/

[5] http://weka.sourceforge.net/doc/overview-tree.html

Also, we can recommend to consider SVM(Weka) with the following setting

$$ops = \{``-C", ``1", ``-R", ``1", ``-G", ``9"\}.$$

Using above model we observed validation result 0.7947 in the advanced track.

In difference to simple and fast model described in Section 6.3, models SVM(Weka) and MLP(Weka) are rather slow. As a consequence, and in order to avoid "time-out" outcome, we did not use the most natural scheme (1) for our final submission (advanced track). Our final JAR-file was prepared with fixed Δ - that means, the power of FS was about the same for all test data sets. However, based on our experiments with LOO-evaluations, we have noticed that the performance of the model is a very sensitive to the selection of the parameter Δ, see Table 4.

7 The Second Best Solution from Basic Track: Random Forest Approach for Distributed and Unbalanced Prediction Tasks

7.1 Introduction

Typical properties of microarray datasets are a large number of attributes, small number of examples and usually unbalanced class distributions. Most importantly, relevant information in these datasets is distributed across many attributes. As such, these datasets are a challenging prediction task.

Good prediction results for such datasets can be expected from systems able to appropriately reduce the attribute space to some reasonable size but retain the distributed information. Also, a classifier constructed from the reduced attribute set should still be able to integrate a relatively large number of attributes into a model. There is a significant danger of overfitting because the model must be complex and the number of available training examples is small. In such situation construction of many diverse classifiers and implementation of an appropriate voting scheme seems as the only possible solution.

The Random Forest (RF) [7] approach for supervised inductive learning enables a relatively simple and straightforward framework for building a set of diverse classifiers. Some of its advantages are the ability to cope with extremely large number of attributes even when there is a small number of instances, and an option to balance datasets through user defined weights.

In the next Section we present some basic concepts of the RF approach, describe its current parallel implementation prepared at the Rudjer Boskovic Institute, and demonstrate RF attribute importance feature. In Sections 7.3 and 7.4 we describe details of our solution which concentrated mainly on the task of reducing the size of the original attribute space, search for the optimal weights of classes thus balancing class distribution, and an approach to refine RF results by outlier detection and correction.

7.2 Random Forest

Random Forest is a general purpose classification and regression meta-learning algorithm which works by combining bagging [8] and random subspace method [9] approaches in constructing an ensemble of random decision trees.

RF is computationally efficient as it is able to learn a forest faster than bagging or boosting, it is capable of handling large datasets with thousands of categorical and continuous attributes without deletion which is very suitable for microarray analysis, and is also capable of balancing out class-wise error rates through a user-defined set of weights used in the process of tree growing. Besides this, it is capable of producing many useful data for result interpretation such as attribute importance, a feature we used for our attribute selection process.

Attribute importance is calculated by comparing the misclassification rate of the original vs. per-attribute randomly permuted data in the process of error estimation for the single tree. By subtracting the number of correct votes for the attribute-permuted data from the number of correct votes of the original data and averaging them over all trees in the forest, raw importance and later, significance levels for the attributes are obtained. Attribute importance is especially helpful when using data with a large number of attributes like microarrays.

In this work we used a parallel implementation of the RF algorithm developed by G. Topić and T. Šmuc at the Rudjer Boskovic Institute. This implementation is written in Fortran 90 and the parallelization of the algorithm has been accomplished using MPI (Message Passing Interface). PARF is licensed under GNU GPL 2.0 license. More information about the implementation, usage, help and source code can be found on PARF's homepage: `http://www.parf.irb.hr/`.

7.3 Finding Optimal Random Forest Parameters

The main problems of RF approach practical application for the RSCTC'2010 datasets has been confronted with are a) selection of the appropriate subset of attributes and b) selection of appropriate weights for classes with small number of examples in the training set. The problems have been solved with a series of experiments performed with different levels of attribute reduction and different weights for rare example classes. The optimal combination has been identified by minimal out-of-bag error estimation [7] which has been remodeled to fit the evaluation criteria of the balanced accuracy defined for the Challenge. All of the experiments were executed with a number of trees in the forest equal to 1000 to ensure stable prediction quality.

In order to implement a relatively systematic search through the space of all possibly interesting combinations, a double nested loop has been implemented. In the outer loop the threshold for attribute significance was varied from $0.1 - 0.01$ in steps of 0.01. With this parameter we have varied the size of the attribute set entering the second RF learning phase. The actual numbers of the selected attributes varied between datasets from as low as 7 to 603.

In the inner loop we had a parameter for class weight optimization. The weights for all classes with high number of instances were set to 1 while for rare classes they have been preset inverse proportionally to the size of the class. The latter have been multiplied by the parameter for weight class optimization which varied from $1 - 4$ in increments of 0.2.

In the described setting a total of 160 experiments have been performed for each dataset. Typically optimal attribute significance was in the range $0.06-0.09$ with class weight parameter typically in the range $1.5 - 3.1$.

7.4 Outlier Detection from Integrated Training and Test Sets

The methodology described in the previous section enabled us to select optimal RF parameters for each dataset. By using these parameters we have constructed one final model for each dataset which we used to classify test set examples. Afterwards, we additionally applied a methodology for outlier detection in order to improve the solution.

For this task we have integrated each dataset with classified examples from its corresponding test set. By their construction we have been able to test if there are strong outliers in these large datasets and in cases when the outliers are from test sets, try to correct them. For this task we have applied the saturation based filtering methodology for explicit outlier detection described in [10].

The saturation based outlier detection methodology tries to estimate minimal complexity of the hypothesis that is able to correctly classify all examples in the available dataset. After that it tries to identify if there exist examples by whose elimination this complexity could be significantly reduced. If one or more such examples can be found, they are declared as potential outliers. The methodology is appropriate for domains in which useful classification information is concentrated in a small set of very important attributes. Having distributed information in microarray datasets in this Challenge we had to use it with special care. Additionally, the currently available version of the methodology can handle only two-class domains. Because of these problems we have used it in a semi-automatic mode, carefully evaluating each detected outlier and a potential change of its originally determined class.

For each enlarged dataset with C classes we have constructed C different concept learning (two-class) problems so that each class is once a positive class and examples from all other classes are treated as negative class examples. In this way one original multiclass example has once been a positive example and $C-1$ times a negative example. Outlier detection process has been repeated for all concept learning tasks independently. Finally, we have searched for examples coming from the test set that have been detected as potential outliers when they have been among positive examples and exactly once when they have been in some negative class. Only in such cases we accepted to change the original classification of the example. The new classification of the example corresponded to the positive class of the concept learning task when the example has been detected as a negative outlier. This methodology enabled correction of the classification for up to 3 examples per dataset.

8 The Best Solution from Advanced Track: A Feature Selection Algorithm Based on Cut Point Importance and Dynamic Clustering[6]

In RSCTC'2010 Discovery Challenge, the DNA data arrays [11] with large number of features (attributes) and small number of records (objects) were provided. Suppose $|U|$ be the number of objects, and $|C|$ be the number of features. According to the provided data, it is obvious that $|C| \gg |U|$. Therefore, it is urgent to select smaller subset of features from the DNA data array. In this section, an efficient solution for feature selection method, based on importance of cut points and dynamic clustering, is introduced. It is combined with SVM.

Firstly, according to [12], the importance of cut points can be computed. After that, the feature selection algorithm based on importance of cut points and dynamic clustering will be presented as follows.

Algorithm 1. Feature Selection Algorithm Based on Cut Point Importance and Dynamic Clustering:

```
Input: Decision table $S =< U, A = C \cup D, V, f >$ and feature number $K$.
Output: Selected feature set $SelectFeature$.
Step1: (Computing the importance of cut points on all features).
    FOR i = 1 TO $|C|$ DO
        Computing the importance value of cut points on feature $c_i$, according to [12];
        Normalization the importance value of cut points on feature $c_i$.
    END FOR
Step2: (Dynamic clustering)(**Due the limitation of page size, we can not present the complex algorithm in detail**)
    FOR each $c_i (1 \leq i \leq |C|)$ DO
    Step2.1: Sorting cut points.
    Step2.2: Connecting the importance value of all cut points. It can be
            found that there will be only a summit on the curve of the
            importance value of cut points. According to the summit,
            dividing the cut points into two parts: $Left$ and $Right$.
    Step2.3: Dynamic clustering the importance of cut points. Then, the
            importance of cut points in $Left$ or $Right$ can be dynamic
            clustered respectively.
    Step2.4: Suppose the clustered classifications on feature $c_i$ be $k_i$.
    END FOR
```

[6] This work is supported by the National Natural Science Foundation of China (NSFC) under grant No.60573068 and No.60773113, Natural Science Foundation of Chongqing under grant No.2008BA2017 and No.2008BA2041, and Science & Technology Research Program of Chongqing Education Commission under grant No.KJ090512.

```
Step3: Sort $k_1, k_2, ..., k_{|C|}$ by increasing order. Suppose the order result be
    $c_{r_1}, c_{r_2}, ..., c_{r_{|c|}}$;
    $SelectFeature = \{c_{r_1}, c_{r_2}, ..., c_{r_k}\}$.
Step4: RETURN $SelectFeature$.
```

Secondly, a good library for Support Vector Machines is adopted. The configuration of LIBSVM [13] is introduced (see Table 5). In Table 5, the changed parameters are showed. Besides, the rest parameters are default by LIBSVM.

During experiments, parameter K was set to 5000. Performance of the presented "Feature Selection + SVM" method were tested on the DNA datasets of RSCTC'2010 Discovery Challenge. The final experimental results for Advanced Track was 0.75661.

Table 5. The parameter configuration of LIBSVM

Configuration Parameter	Description	Configuration Value
svm type	set type of SVM	C-SVC
kernel type	set type of kernel function	LINEAR
gamma	set gamma in kernel function	1/num_features
cost	set the parameter C of C-SVC, epsilon-SVR, and nu-SVR	100

9 The Second Best Solution from Advanced Track: A Feature Selection Algorithm Based on Attribute Relevance[7]

When analyzing microarray data [11], we have to face the few-objects-many-attributes problem. However, there exists dependence between condition attributes and decision attribute, and only a small subset of attributes is relevant in the context of a given task. In this section, an effective feature extraction method, the feature selection algorithm based on attribute relevance, is introduced. Furthermore, combining with the proposed algorithm, the LIBSVM [13] is adopted to solve the given task in RSCTC'2010 Discovery Challenge' Advanced Track. According to [14], the ratio of condition attribute's between-groups to within-groups sum of squares can represent the relevance between condition attribute and decision attribute. We modified the ratio expression and proposed a new expression to represent attribute relevance.

[7] This work is supported by the National Natural Science Foundation of China (NSFC) under grant No.60573068 and No.60773113, Natural Science Foundation of Chongqing under grant No.2008BA2017 and No.2008BA2041, and Science & Technology Research Program of Chongqing Education Commission under grant No.KJ090512.

Firstly,feature selection algorithm based on attribute relevance can be described as following in detail.

Algorithm 2. Feature Selection Algorithm Based on Attribute Relevance:

```
Input: Decision table $S =< U, A = C \cup D, V, f >$ and feature number $K$.
Output: Selected feature set $SelectFeature$.
Step1: (Sorting objects,according to the value of decision attribute $d$.)
Step2: (Compute the relevance of condition attributes and decision attribute)
    FOR each $c_i (1 \leq i \leq |C|)$ DO
    Step2.1: Compute the standard deviation $SD_i$ on attribute $d$. where
        $SD_i = \sqrt{\frac{\sum_j^{|U|} (x_{ji} - \overline{x}_{.i})^2}{|U|-1}}$
    Step2.2: Suppose $|U/\{d\}|$ object sets $U^1, U^2, ..., U^{|U/\{d\}|}$. Compute the
        standard deviation in each object set.
        FOR each object set $U^k (1 \leq k \leq |U/\{d\}|)$ DO
        $SD_i^k = \sqrt{\frac{\sum_j^{|U^k|} (x_{ji} - \overline{x}_{ki})^2 * I(y_j = k)}{|U^k|-1}}$, were $\overline{x}_{ki} = \sqrt{\frac{\sum_j^{|U^k|} x_{ji} * I(y_j = k)}{|U^k|}}$
        and $I(y_j = k) = \begin{cases} 1, y_j = k. \\ 0, y_j \neq k. \end{cases}$
    Step2.3: Compute the relevance of condition attributes and decision
        attribute: $R_i = \frac{\sum_k^{|U/\{d\}|} SD_i^k}{SD_i}$
    END FOR
Step3: Sort $R_1, R_2, \cdots, R_{|C|}$ by increasing order. Suppose the order result
    be $c_{r_1}, c_{r_2}, ..., c_{r_{|c|}}$;
    $SelectFeature = \{c_{r_1}, c_{r_2}, ..., c_{r_k}\}$.
Step4: RETURN $SelectFeature$.
```

Secondly, a good library for Support Vector Machines is adopted. The configuration of LIBSVM [13] is introduced (see Table 5). In Table 5, the changed parameters are showed. Besides, the rest parameters are default by LIBSVM.

During experiments, parameter K was set to 5000. Performance of the presented "Feature Selection + SVM" method were tested on the DNA datasets of RSCTC'2010 Discovery Challenge. The final experimental results for Advanced Track was 0.75180.

Acknowledgements

The research has been supported by grants N N516 077837 and N N516 368334 from Ministry of Science and Higher Education of the Republic of Poland.

References

1. Parkinson, H.E., et al.: Arrayexpress update - from an archive of functional genomics experiments to the atlas of gene expression. Nucleic Acids Research 37(Database issue), 868–872 (2009)
2. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008)
3. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: ML92: Proceedings of the ninth international workshop on Machine learning, pp. 249–256. Morgan Kaufmann Publishers Inc., San Francisco (1992)
4. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
5. Tibshirani, R., Hastie, T., Narasimhan, B., Chu, G.: Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proceedings of the National Academy of Sciences USA 99(10), 6567–6572 (2002)
6. Nikulin, V., McLachlan, G.J.: Classification of imbalanced marketing data with balanced random sets. In: JMLR: Workshop and Conference Proceedings, vol. 7, pp. 89–100 (2009)
7. Breiman, L.: Random forests. Machine Learning, 5–32 (2001)
8. Breiman, L.: Bagging predictors. Machine Learning, 123–140 (1996)
9. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20(8), 832–844 (1998)
10. Gamberger, D., Lavrac, N.: Conditions for occam's razor applicability and noise elimination. In: van Someren, M., Widmer, G. (eds.) ECML 1997. LNCS, vol. 1224, pp. 108–123. Springer, Heidelberg (1997)
11. Wikipedia: Dna microarray – wikipedia, the free encyclopedia (2010), `http://en.wikipedia.org/w/index.php?title=DNA_microarray`
12. Nguyen, H.: Approximate boolean reasoning: Foundations and applications in data mining. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets V. LNCS, vol. 4100, pp. 334–506. Springer, Heidelberg (2006)
13. Chang, C., Lin, C.: Libsvm – a library for support vector machines. [EB/OL], `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (2008-11-17/2010-01-3)
14. Dudoit, S., Fridlyand, J., Speed, T.: Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association 97, 77–87 (2002)