Proceedings of the 2nd International Conference on Computer and Information Science and Technology (CIST'16)
Ottawa, Canada – May 11 – 12, 2016
Paper No. 111

A Computational Verb Rules Based Approaching for Classification of Microarray Data

KunHong Liu, Vincent Ng
Department of Computing, The Hong Kong Polytechnic University
Kowloon, Hong Kong, China
lkhqz@163.com; cstyng@comp.polyu.edu.hk

Abstract - This paper proposes a novel method to extract interpretable rules from microarray data based on computation verb theory. Different from the existing rule based methods, the computational verb technique can produce template-based verbs and adverbs to describe the relationships between gene pairs, making the results to be better understood. We use the verb stay based rules to deal with the analysis task, and carry out experiments based on binary class and multiclass microarray datasets. And the experimental results show that only five rules can produce more robust results compared with the tree-based methods in classifying the microarray data in most cases.

***Keywords*:** Computational verb rules; microarray data; interpretability

1. Introduction

Up to now, many supervised and unsupervised machine learning techniques have been proposed to tackle the microarray data analysis task, leading us to a more complete understanding of the molecular variations among tumors [1]. One difficulty of the microarray data analysis is that the number of samples collected tends to be much smaller than the number of genes, resulting with a large number of genes making no contribution to cancer diagnosis [2]. A widely deployed solution for this problem is the feature selection approach, with which a set of biologically significant genes will be discovered and used to improve the cancer diagnosis accuracy [3]. In order to get a deep insight in the gene functions and regulation relationships, when analyzing microarray data, experts hope that the mining results do not only produce accurate results, but also generate comprehensive knowledge. There are many different methods for extracting rules from microarray. For example, fuzzy-rule based models have been successfully applied in microarray classification by modeling noun and adjective, such as "high", "low", etc. Some researchers presented a framework for learning fuzzy rules by extracting fuzzy rules using genetic algorithms[4,5], while others presented rule-induction and filtering strategies to obtain a small fuzzy classifier using a grid partition of feature space, to obtain rules like "if (H62098,HIGH) then BI" [6]. Note that it is possible to incorporate biological knowledge into this kind of algorithm by specifying the forms and parameters of fuzzy membership functions. However, the coverage of such rules cannot be naturally guaranteed because the rules are only based on local information, leading to a large proportion of redundant rules.

On the contrary, there are some algorithms producing rules based on the whole dataset, so the generalization of these rules is better than the previously described ones. A typical method is k–Top Scoring Pairs (TSP), proposed in [7]. It compares gene pairs, exploiting discriminating information contained in the data focusing on 'marker gene pairs', for which there is a significant difference in the expression level across the N samples between two classes. These rules take the form of: "IF SPTAN1>CD33 THEN ALL; ELSE AML". Furthermore, there are some evolutionary based rule generation systems. One typical system has been implemented with genetic programming (GP) by employing multiple logical and mathematical operations in individual's structure. The final GP individuals can be expressed as rules like: "if max(Gene1722,Gene2165)>1.9190 then small-cell lung carcinomas"[8]. With such implementations, the final decision can be made by fewer rules because each rule can cover all samples.

The development of computational science technique provides us more different method for investigating microarray data [9-10]. In this paper, we propose a Computational verb (CV) based method to generate rule based methods for microarray data analysis. CV theory was invented by Tao Yang, and then it has been applied to a lot of scientific fields, such as linguistics, biology, psychology, physics and computer sciences [11]. CV can be applied to build a complete

artificial language into machines, and the CV based rule (CVR) is a further step of CV, which can be expressed as some simple rules using verbs and adverbs to describe the changes or status by summarizing interaction terms and constants into linguistically interpretable forms. The CVR can act as a special role in investigating the relationships among different classes in microarray data, because of the verb-based descriptions in the rules, which can provide a new point of view for biologists and biomedical scientists to investigate microarray data. Experiments have been carried out in both binary class and multiclass microarray datasets, and compared the CVR performance with different tree based methods. The results confirm the effectiveness of our CVR methods. And we also list some typical rules to demonstrate the interpretability of our algorithm.

2. The Framework of Computational Verb Rules

CV can solve engineering problems by transforming different types of natural words to mathematical formulas. The CV theory can be traced back to the 1997, and it has undergone a rapid growth [11-14]. However, there are still few explorations in the application of CV to the bioinformatics field. Up to now, only Tong proposed a CVR based method to analyze microarray data [15]. In his work, a computational verb rule is used to compare the change of expression levels between them to deal with a binary-class problem by using the rules presented as: If Gene i increases/decreases relative to Gene j, then class 1/2. In the rule, class 1/2 refers to normal/cancer type. With this type of rules, a sample can be classified based on the expression levels of two genes. However, the gene expression data is not time-varying data, and increase/decrease may not be perfect words to describe the comparison of gene expression level. Such rules may be affected by the input sequence of a training set, making the performance of CVR unstable. Hence, the changes in the sequence may lead to different decisions. An extreme example is the case when the input sequence is reverse. In such circumstance, the sample set satisfies the first rule would no longer fit the second rule; vice versa, it becomes to satisfy the second rule, producing completely different results.

In this paper, we further the investigation of the use of CVR, and provide a deeper insight of CVR in the bioinformatics field. Instead of using an action related verb, we suggest a verb describing states to model data: stay. Although our rules are also based on a gene pair, they are different from the previously proposed rule set. Furthermore, we also try to use computational adverbs to make our rules more comprehensible.

2.1. The Concept of Computational Verb

A Computational Verb is formalized as a 4-tuple $(v, T, \Psi, \varepsilon)$ [14], where v is a verb (e.g. increase). (T, Ψ, ε) is a dynamical system, in which T is the life span. Ψ is the state space of the system, and ε is the evolution function of the dynamical system.

$$\varepsilon : T \times \Psi \rightarrow \Psi \tag{1}$$

$$\varepsilon(0, x) = x,\ \varepsilon(t_2, \varepsilon(t_1, x)) = \varepsilon(t_1 + t_2, x) \tag{2}$$

where (T,+) is a monoid, x is the observation.

Typical verbs in CV are: become, stay. They can describe the trend of changes in a time serial. Some adverbs, such as fast, slow, can also be applied to facilitate the model building process. The consequents and antecedents of CVR can be constructed with both dynamic and static forms. A CV based rule may take the form of:

$$\text{If } X_1 \text{ } Ad_{x1} \circ V_{x1} \text{ AND } X_2 \text{ } Ad_{x2} \circ V_{x2} \text{ THEN } Y \text{ } Ad_Y \circ V_Y \tag{3}$$

where Adx○Vx represents the adverb (Adx) and verb (Vx) used to feature Xi. A rule can contain as many verbs as needed, but typical rules would not contain more than two verbs for alleviating the complexity. Most of verbs require one or two features, the number of which is determined by the formula of a verb.

2.2. The CV Template and Evolving Functions

In CV, there are many different template functions for verbs and adverbs. A typical computational verb template is

$$V = w_2 t^2 + w_1 t + w_0 \tag{4}$$

In (4), $t_0 = 0$ is set as the starting point, and $\Delta t = 1$ as the time step. w_0 ,w_1 and w_2 are parameters requiring to be set by some learning algorithms. Such verbs can be used to form a set of rules. For example, a CV rule can express as:

$$\text{If } X_1 \textit{ fastly become } V_{x1}, \text{ and } X_2 \textit{ stay } V_{x2} \text{ then } Y \textit{ slowly become } V_Y \tag{5}$$

where both X_1 and X_2 are the observations for this rule, and Y is the expected outputs. Words like fastly/slowly are computational adverbs, which also requires to be defined on the context. There are also some other suggested templates for adverbs in [14]. Then CV can be formulated as:

$$Y_{new} = f(s(X, V_x), V_y, Y_{current}) \tag{6}$$

where s(X,Vx) is used to evaluate the similarity between X and Vx. f is a preset function used to determine the way to update Ynew according to the similarity between Vy and $Y_{current}$. The evolving function of the CV stands for an orbit of its dynamical system, and it describes the change of an action with respect to time. The evolving function of the inputs is generated with given samples. For example, the verb increase (expressed as Vincrease) can be expressed as a 4-tuple (increase; R+; R; ε), where ε(t; x) = t + x, and $t \in R+$. Here R represents the real number field. There are many possible orbits for this dynamical system with different initial states. By setting x to 1, we can get a simple form of evolving function for the verb increase as suggested in [14]:

$$\varepsilon_{increase} = 1 + t;\ t \in R^+ \tag{7}$$

For different verbs, the evolving functions are different. For the verb *stay*, a typical evolving function is:

$$\varepsilon_{stay} = |1 + t| < \delta;\ t \in R^+ \tag{8}$$

where δ is the expected deviation.

2.3. The Similarity Function

The similarity measurement function between two verbs is based on their evolving functions. Given the evolving functions of the two verbs, $\varepsilon_1, \varepsilon_2 \in E$, where E stands for the set of evolving functions. Their similarity is calculated according to a similarity function, $s : E^2 \to R^+$, which satisfies the following conditions:

$$s(\varepsilon_1, \varepsilon_2) = s(\varepsilon_2, \varepsilon_1),\ \forall\ \varepsilon_1, \varepsilon_2 \in E \tag{9}$$

$$s(\varepsilon_1, \varepsilon_1) = 1,\ \forall\ \varepsilon_1 \in E \tag{10}$$

$$\text{If } \varepsilon_1(t)\ \varepsilon_2(t) \equiv 0,\ \varepsilon_1(t) + \varepsilon_2(t) \equiv 1, \forall\ t \in T, \text{ then } s(\varepsilon_1, \varepsilon_2) \equiv 0 \tag{11}$$

$$\forall\ \varepsilon_1, \varepsilon_2, \varepsilon_3 \in E, \text{ if } \forall\ \varepsilon_1 \le \varepsilon_2 \le \varepsilon_3, \text{ then } s(\varepsilon_1, \varepsilon_2) \le s(\varepsilon_1, \varepsilon_3) \text{ and } s(\varepsilon_2, \varepsilon_3) \le s(\varepsilon_1, \varepsilon_3) \tag{12}$$

A similarity function is used to compare two states of a verb (V_x and X) or two verbs (V_1 and V_2). The design of verb similarity function is problem-dependent. Up until now, there are mainly three kinds of verb similarity functions that are widely used in various applications: distance-based, trend-based and frequency-based. In this study, we evaluate the distance between two verbs according to the change of distance (s_d) within the time interval [0, T], as shown in (13). s_d is the sum of the square of point-by-point amplitude diversities between two evolving functions within the time interval. Function g() in (13) takes the form in formula (14) to map the outputs within the range of [0, 1].

$$s_d(\varepsilon_1, \varepsilon_2) = g\left(\sum_{t=0}^{T} |\varepsilon_1(t) - \varepsilon_2(t)|^2\right) \tag{13}$$

$$g(x) = \frac{2}{1 + e^x} \tag{14}$$

2.4. The Application to Cancer Classification Problem

By using the canonical form, the template function can be parameterized and become learnable. Both the inputs and the template functions are modeled by using a computational verb (v, R^+, R, ε). Two genes are regarded as two tuples in timeline, and a computational verb rule is used to compare the change of expression levels between them to deal with a binary-class problem. Hence, a sample can be classified based on the expression levels of two genes. The similarity function evaluates the probability of a sample belonging to a class.

The computational verb *stay* is used to describe the states to model gene data, and the rules are presented as:

If Gene *i stay relatively active to* Gene *j*, then class 1; (15)

If Gene *i stay relatively inactive to* Gene *j*, then class 2; (16)

Although our rules are also based on a gene pair, just like [15], they are different from the previously proposed rule set in that the verb *stay* can describe the accumulate effect among a dataset and our method become independent to the input sequence. The words *active* and *inactive* are used to report the gene expression status so that the rules are more illustrative for researchers to understand the results.

- Step 1. For a gene *i*, calculate the mean expression value of each class, and assign the larger one to G_{i1}, the smaller to G_{i2}. Then calculate the differentiation value (DV_i) of the *active* and *inactive*, $DV_i=|G_{i1} - G_{i2}|$. Repeat the calculation until all genes are checked.
- Step 2. Sort genes according to *DV*, and select the top 100 genes to form a candidate pool.
- Step 3. Form a gene pair by picking the top gene *i* in the pool, and match it with a gene *j* that can produce the largest value $D_{i,j}$ according to formula (19). Remove *i* and *j* from the pool.
  $$D_{i,j} = |G_{i1} - G_{j2}| + |G_{i2} - G_{j1}| \quad (19)$$
- Step 4. Repeat Step 3 until the pool is empty.
- Step 5. For each gene pair: Train the gene pairs to build the computational verb rules according to (17) and (18); Optimize the parameters (k_1/k_2 and W_1/W_2) using the gradient descent algorithm.
- Step 6. Select the top *N* trained CV*R* classifiers to predict the unseen samples with majority vote method.

Fig. 1: CVRStay algorithm.

The verb used in the CVRStay algorithm in Figure 1 is based on the typical form recommended in [11]. For each rule, the similarity is measured between the input and the antecedents of these rules. The final decisions are based on the similarity of the rules' output.

Assume that V_{1i}/V'_{1j} represents *stay active*, and V_{2i}/V'_{2j} represents *stay inactive* for gene *i/j*. x_i/x_j represents the expression level of a sample for gene *i/j*, and there are *n* samples in a dataset. Let S_1 and S_2 represent the results of the antecedents of rule (15) and (16), then they can be calculated by combining the distance based similarity function, as shown in (17), (18).

$$S_1 = s_{d1}(V_{1i}, V_{1j}) = g[\sum_n (k_1(|x_i - G_{i1}| - \sigma_1) - (|x_j - G_{j1}| - \sigma_2))^2]$$
$$= g[\sum_n (k_1(|x_i - G_{i1}| - |x_j - G_{j1}| - w_1))^2] \quad (17)$$

$$S_2 = s_{d2}(V_{2i}, V_{2j}) = g[\sum_n (k_2(|x_i - G_{i2}| - \sigma_3) - (|x_j - G_{j2}| - \sigma_4))^2]$$
$$= g[\sum_n (k_2|x_i - G_{i2}| - |x_j - G_{j2}| - w_2)^2] \quad (18)$$

In (17) and (18), all σ are used as parameters for adjusting the rules to fit the data distribution better. To simplify the calculation process, two parameters, w_1 and w_2 are introduced instead. Here, $w_1=k_1\times\sigma_1+\sigma_2$, $w_2=k_2\times\sigma_3+\sigma_4$.

The verb template in (5) is applied to generate the rules along with the evolving function *stay*. In order to evaluate the status of a gene, G_{i1}/G_{j1} and G_{i2}/G_{j2} are parameters used to determine whether gene *i*/*j*'s expression level is relatively high (*active*) or low (*inactive*). They are determined entirely by using the data distribution. Here, a gene *i*'s average expression values of both classes in a dataset are calculated first. The relatively larger/smaller value is assigned to G_{i1}/G_{i2} to represent the level of *active*/*inactive* status. The difference between these values affects the discriminative power of the final rules. It is obvious that the performance of final rules relies on the differentiation of the *active/inactive* comparison levels in the selected gene pair. The more significant difference in the level of two statuses in a gene pair, the higher generalizability of the produced CVR could be obtained.

To better illustrate the difference in the comparison of statuses in a rule, k_1 and k_2 are used to implement the adverb *relatively to* in the rules. In the rules, r*elatively to* is deployed to measure the degree of *active/inactive*. Such a r*elatively to* relationship can be *more* if k_1 or k_2 is larger than 1, and *less* otherwise. In this way, a rule can give a more accurate comparison for a gene pair, taking the form as: If Gene *i stay more active compared with* Gene *j*, then class 1. With these four parameters, k_1/k_2 and W_1/W_2, the CVRStay algorithm can be optimized to make rules matching with the training data better. The gradient descent algorithm is employed to optimize these parameters. To limit the final outputs within [0, 1], the final results are normalized by adjusting the output of (17) and (18) as:

$$S_1^{'} = S_1/(S_1+S_2)$$
$$S_2^{'} = S_2/(S_1+S_2) \tag{20}$$

In this way, S_1' and S_2' can reflect the probabilities of a sample belonging to a class 1/2. The final decision for a sample is made by assigning it to the class obtaining a higher probability. In short, the workflow of algorithm CVRStay is shown as Figure 1. In our framework, it is obvious that the larger difference in *active* or *inactive* status within a gene pair, the more powerful the generalizability the CVR can be. Step 2-4 try to filter out unremarkable genes, and match informative genes to form gene pairs quickly using a greedy selection scheme. After these steps, only top *N* gene pairs left for building computational rules. This approach can produce gene pairs without overlaps. In our experiments, *N* is set to 5. So in each CVR based experiment, only 10 genes are used to produce results.

3. Experiments and Results

In our experiments, we use three binary class and four multiclass microarray datasets. The details about these microarray datasets are listed in Table 1 and 2, respectively. In these datasets, all data samples have already been assigned to training set or test set. Preprocessing of the datasets is done exactly as described in [7]: transforming the raw data to natural logarithmic values, and then standardizing each sample to zero mean and unit variance. Besides the original division, for the binary class problem, each dataset are reshuffled with 9 randomizations. For all datasets, each randomized training and test set contains the same amount of samples of each class compared to the original training and test set.

In all our experiments, the classifiers are built using the training samples, and the classification results are estimated using the independent test set. The standardization and the feature selection steps are applied for all the classifiers. In other words, all the classifiers receive the same reduced subset of features for a given training and test dataset.

For comparison of the performance of CVRStay, some tree-based learners are also used in all experiments, including decision tree, Random Forest and Rotation Forest. For decision tree and Random Forest, the implementations in scikit-learn library are used [23]. An improved Rotation Forest algorithm, hybrid extreme rotation forest (HERF) [24], is deployed, and its python implementation is available at [25]. All classifiers are used with the default settings. For each two-class problem, five CV rules are used to produce outputs, working as an ensemble classifier. However, since there are only ten genes used, the scale of the ensemble CV rules is of a smaller scale even when compared with using decision trees.

There are no duplicate genes used in building the rules, so that the diversity of the final ensemble can be guaranteed.

To evaluate the results, we use some different measures based on the suggestion of [16]. After obtaining the values of true positive (tp), true negative (tn), false positive (fp) and false negative (fn), we use accuracy and Fscores to compare the results. Classification accuracy indicates the percentage of correctly classified samples, and Fscores reveals the balance degree of results between two classes.

Table 1: Binary Class Datasets used in Experiments.

Datasets	No. of Genes	No. of samples of two classes	Reference
Ovarian	15154	162/91	[17]
Colon	20000	40/22	[18]
Lung	12533	150/31	[19]

Table 2: Multiclass Datasets used in Experiments.

Dataset	No. of classes	No. of genes	No. of training samples	No. of test samples	Reference
Leukemia1	3	7129	38	34	[1]
Leukemia2	3	12,582	57	15	[20]
Lung1	3	7129	64	32	[21]
Lung2	5	12,600	136	67	[22]

For a *m*-class problem, different class labels are represented as 1, 2, *m*. Since CVRStay can only directly solve binary-class problems by producing Yes/No answer, in our experiments, a multiclass problem is decomposed to a set of binary class problems. Two commonly used decomposition methods are employed: One vs. One (OVO) and One vs. Rest (OVR). For fair comparisons, decision tree, Random Forest and Rotation Forest methods are also used as binary classifiers, fused with OVO and OVR methods. It should be noted that decision trees are also ensemble systems after they are fused with OVO/OVR methods.

In tackling the multiclass problem, experiments are only carried out based on the original splits. Because random forest and HERF are based on random division of sample sets, they ran ten times with different random seeds.We use two different measures, $Fscore_{\mu}$ and *Average Accuracy* (*AAc* for short) for results comparisons. Assume that there are *c* classes in a dataset. Different from accuracy, *AAc* indicates the average per-class performance of a classifier. If a classifier fails to recognize samples in a 'hard' class, it cannot achieve high scores in *AAc*. $Fscore_{\mu}$ is a measure combining the scores of both *precision* and *recall* among all classes.

Table 3: Experimental results for binary datasets.

		colon	Ovarian	Lung
CVRStay	Fscore	**0.824±0.024**	**0.968±0.000**	**0.947±0.000**
	Accuracy	0.843±0.010	**0.976±0.000**	**0.983±0.000**
DT	Fscore	0.743±0.053	0.967±0.009	0.891±0.022
	Accuracy	0.799±0.040	0.975±0.007	0.961±0.008
RF	Fscore	0.743±0.053	0.966±0.009	0.891±0.022
	Accuracy	0.835±0.004	0.969±0.003	0.918±0.040
HARF	Fscore	0.816±0.027	0.960±0.005	0.936±0.027
	Accuracy	**0.862±0.045**	0.956±0.048	0.974±0.009

To get a balance between precision and recall, β is set to 1 in calculating the Fscoreμ. From Table 3, it is found that CVRStay can achieve the best Fscores and accuracy in two datasets. Since higher scores on Fscore represents the better balanced results, it is obvious that our CVRStay algorithm can overcome the sample-imbalanced problem with providing more accurate results.

In Table 4, it can be found that CVRStay based classification results also take advantages in most cases for the multiclass problem. In considering both Fscores and AAC, OVR based CVRStay beats other methods in two datasets, and OVO based CVRStay wins in dealing with the Leukemia1 dataset. For other datasets, only random forest wins in case of the DLBCL dataset. As a result, CVRStay can take advantage with a small ensemble size compared with other classifiers.

Table 4: Experimental results for multiclass datasets.

Table 4. Experimental results for multiclass datasetsDatasets		OVO			
		CVRStay	DT	Random Forest	HERF
Leukemia1	$Fscore_{\mu}$	**0.941**	**0.941**	0.881 ± 0.061	0.923 ± 0.074
	AAc	**0.961**	0.960	0.918 ± 0.040	0.949 ± 0.049
Leukemia2	$Fscore_{\mu}$	0.933	0.733	0.934 ± 0.066	0.956 ± 0.011
	AAc	0.956	0.778	0.953 ± 0.043	0.997 ± 0.008
Lung1	$Fscore_{\mu}$	0.813	0.668	0.791 ± 0.017	0.817 ± 0.013
	AAc	**0.875**	0.792	0.861 ± 0.011	0.869 ± 0.009
Lung2	$Fscore_{\mu}$	0.8358	0.955	0.950 ± 0.008	0.933 ± 0.013
	AAc	0.9343	0.982	0.980 ± 0.031	0.965 ± 0.009
Datasets		OVR			
		CVRStay	DT	Random Forest	HERF
Leukemia1	$Fscore_{\mu}$	0.794	0.882	0.917 ± 0.012	0.941 ± 0.019
	AAc	0.863	0.922	0.944 ± 0.008	0.951 ± 0.012
Leukemia2	$Fscore_{\mu}$	**1.000**	0.773	0.921 ± 0.050	0.925 ± 0.026
	AAc	**1.000**	0.882	0.947 ± 0.033	0.940 ± 0.017
Lung1	$Fscore_{\mu}$	0.813	0.781	0.771 ± 0.025	**0.818 ± 0.036**
	AAc	**0.875**	0.854	0.847 ± 0.017	0.868 ± 0.026
Lung2	$Fscore_{\mu}$	**0.970**	0.851	0.908 ± 0.014	0.946 ± 0.017
	AAc	**0.988**	0.940	0.963 ± 0.006	0.964 ± 0.013

4. Conclusion

In this paper, the computational verb rule (CVR) is introduced to analyze microarray datasets by providing the verb *stay* based rules. Such rules can describe the status of data by summarizing them into linguistically interpretable expressions. So the rules can provide biologists and biomedical scientists with a deep insight of microarray data. The principle and learning method for CVRStay is described in this study. To demonstrate the effectiveness of CVRStay, some experiments are carried our based on binary class and multiclass datasets, and the decision tree, random forest and rotation forest are also used in these experiments for comparisons. Although only top five rules are fused to form the final CVRStay classifiers for each two-class problem with ten genes engaged, the final results prove that CVRStay can achieve the best performance in most cases.

Acknowledgements

The authors would like to thank Mr. Tong Muchenxuan for his help in the design of algorithm. The work is partly supported by National Science Foundation of China (No. 61100106).

References

[1] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, et al., "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531-537, 1999.

[2] C-H. Zheng, L. Zhang, T. Yee Ng, S. C. K. Shiu, D-S. Huang, "Metasample-Based Sparse Representation for Tumor Classification," *IEEE/ACM Trans. Comput. Biology Bioinform*, vol. 8, no. 5, pp. 1273-1282, 2011.
[3] S. K. Pati and A. Kumar D., "Gene Selection and Classification Rule Generation for Microarray Dataset," Advances in Intelligent Systems and Computing, vol. 178, pp. 73-83, 2013.
[4] Z. Wang and V. Palade, "Building interpretable fuzzy models for high dimensional data analysis in cancer diagnosis," *BMC Genomics*, vol. 12, pp. S5, 2011.
[5] S. Ho, L. S. Shu, and J. H. Chen, "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 522-541, 2004.
[6] L. Ohno-Machado, S. Vinterbo, and G. Weber, "Classication of gene expression data using fuzzy logic," *Journal of Intelligent and Fuzzy Systems*, vol. 12, pp. 19-24, 2002.
[7] D. N. A. Tan, L. Xu, R. Winslow, and D. Geman, "Simple decision rules for classifying human cancers from gene expression profiles," *Bioinformatics*, vol. 21, pp. 3896-3904, 2005.
[8] K. H. Liu and C. G. Xu, "A genetic programming-based approach to the classification of multiclass microarray datasets," *Bioinformatics*, vol. 25, pp. 331-337, 2009.
[9] J. C. Chai, S. Park, H. Seo, S. Y. Cho, and Y. S. Lee, "Identification of cancer-specific biomarkers by using microarray gene expression profiling", *BioChip*, vol. 7, no. 1, pp. 57-62, 2013.
[10] B. Y. M. Fung, V. T. Y. Ng, "Meta-classification of Multi-type Cancer Gene Expression Data," BIOKDD: 4th Workshop on Data Mining in Bioinformatics, Seattle, WA, USA, August 22nd, pp. 31-39.
[11] Y. Tao, *The Mathematical Principles of Natural Languages: The First Course in Physical Linguistics, volume 6 of YangSky.com Monographs in Information Sciences*, Tucson, AZ, Yang's Scientific Press, 2007.
[12] Y. Tao, *Lingua Naturalis Principia Mathematica(in Chinese)*, First Edition, Xiamen University Press, 2011.
[13] G. C. T. Pham, "Introduction to Fuzzy Systems," in *Chapman & Hall/CRC Mathematical and Computational Biology*, November 2005.
[14] Y. Tao, "Physical Linguistics: A Measuable Linguistics based on Computational Verb Theory," in *Fuzzy Theory and Probability: Monographs in Information Sciences*, vol. 5, Tucson: Yang's Scientific Press, 2004.
[15] M. Tong, "Extracting Dynamic Classification Rules from Microarray Data," Master thesis, Xiamen University, 2012.
[16] G. L. M. Sokolova, "A systematic analysis of performance masures for classification tasks," *Inf Process Manag*, vol. 45, pp. 423-437, 2009.
[17] A. A. Petricoin, B. A. Hitt, P.J. Levine, V.A Fusaro, S.M. Steinberg, G.B. Mills, C. Simone, D. A. Fishman, E. C. Kohn, and L. A. Liotta, "Use of proteomic patterns in serum to identify ovarian cancer," *Lancet*, vol. 359, pp. 572-577, 2002.
[18] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, et al., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," in *Proceedings of the National Academy of Sciences of the United States of America*, Jun 8 1999, vol. 96, pp. 6745-6750.
[19] J. R. Gordon, L. L. Hsiao, S.R. Gullans, J. E. Blumenstock, S. Ramaswamy, W. G. Richards, D. J. Sugarbaker, and R. Bueno, "Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma," Cancer Research, vol. 62, pp. 4963-4967, 2002.
[20] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. X. Mayer, et al., "Gene selection from microarray data for cancer classification - a machine learning approach," *Computational Biology and Chemistry*, vol. 29, pp. 37-46, Feb 2005.
[21] S. L. Beer, C. C. Huang, T. J. Giordano, A. M. Levin, D. E. Misek, L. Lin, G. Chen, T. G. Gharib, D. G. Thomas, M. L. Lizyness, R. Kuick, S. Hayasaka, J. M. G. Taylor, M. D. Iannettoni, M. B. Orringer, and S. Hanash, "Gene-expression profiles predict survival of patients with lung adenocarcinoma," *Nat. Med.*, vol. 8, pp. 816-824, 2002.
[22] J. Khan, J. Wei, M. Ringner, L. Saal, M. Ladanyi, F. West-ermann, et al., "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Medicine*, vol. 7, pp. 673-679, 2001.
[23] Pedregosa et. al., "Scikit-learn: Machine Learning in Python," *JMLR 12*, vol. 12, pp. 2825-2830, 2011.
[24] B. Ayerdi, M. Graña, "Hybrid extreme rotation forest," vol. 52, pp. 33-42, 2014.
[25] B. Ayerdi. (2014, June 6). Adaptative Hybrid Extreme Rotation Forest [Online]. Available: https://github.com/borjaayerdi/AdaHERF