Monograph

Title: Gene Expression Tree Classifier System for MAIGES

Tutor: Prof. Dr. Eduardo Jordão Neves

Author: Xieli Zhaofu

Table of Contents

PART I

1. Introduction

DNA microarray as a newly emerged technology is playing a very crucial role in the modern molecular biology and medicine science researches. It consists of thousands and millions microscopic spots which can be used to measure gene expression. In a microarray involved experiment, one of the challenges to researchers normally is to find out new tools and ways to process the plenty of gene related information.

This article discusses and represents some fundamental mathematical tools for decision-making processes in a pattern recognition system, and a gene data tree classifier system which can be integrated into MAIGES[1].

This monograph, as a part of my T.C.C[2], is divided into several chapters and sections:

In Chapter 1, there is a brief introduction for pattern recognition system and its related concepts which bring a background for this monograph. And also, at the second part of the chapter, there is an introduction about the T.C.C's objective.

In Chapter 2 and Chapter 3, the realized studies and research topics, such as CART algorithm, decision tree etc., are summarized, as well as all the classifier system.

In Chapter 4, I will summarize my T.C.C as whole, which includes the result and my conclusion about it.

In Chapter 5, all the study and research resources are listed.

In the second part of this monograph, there are my opinions about how my B.C.C. course study related with my T.C.C. project.

1.1 Pattern Recognition System

Pattern classification, also called pattern recognition, is the task to classify data basing on a priori known information or information extracted from the data. It is being used in the diverse research and application areas, such as: image processing, bioinformatics, clinical medicine, etc.

[1] MAIGES - Mathematical Analysis of Interacting Gene Expression Systems, URL: http://www.maiges.org/
[2] T.C.C – Trabalho de Conclusão de Curso

A pattern recognition system could be run in two models: supervised learning or unsupervised learning. The more details about these two models will be listed in the chapter 2.

Normally, a pattern recognition system has the following steps:

- Collection: corresponds to gather raw data. Sometimes, it is also being called "Sensing" when the raw data is gathered by sensors.
- Normalization: corresponds to normalize the raw data to be processed.
- Feature Extraction: corresponds to extract numeric or symbolic information from the normalized data.
- Classification: categorizes the data according to the extracted features and generates output data.
- Comparing: compares the output data with the desired output.
- Adjusting: uses the comparing result to adjust classifier parameters.
- Complete classifier design.

In this project, the gene expression tree classifier system acts like a modified supervised learning pattern recognition system. The system uses a priori known gene expression data as train, and generates classifiers according to the features of data. The generated classifiers can be used in the future classification experiments.

1.2 Objective

The objective of this T.C.C. can be divided into the following steps:

- Study and research the related topics.
- Create a gene expression tree classifier system by using the study and research result from the step 1.

At the step 1, As a completely newcomer to bioinformatics, my main goal is to have some feelings about the Bioconductor and Microarray related topics, and also study related bioinformatics and statistical knowledge, as most as possible, to turn the starting of the step 2 more suave and less pungent.

At the step2, the goal is to create a robust and portable classifier system which can be integrated into MAIGES and its related R statistic computation environment. The classifier system needs to provide the system users, most likely gene related researchers, sufficient information about the generated classifiers and the classifiers' selection process information as well as the classifiers' performance information. Also, the system needs to be able to receive personalized parameters from the users and generates classifiers according to their specified requirements. The more information about the classifier system[3] is stated in Chapter 3.

[3] *Classys* is the name of the created classifier system

2. Concepts

Before touching the classifier system, some basic concepts were needed to be studied. In this chapter, I will try to summarize some more important concepts of all the stuffs that I have studied at step 1.

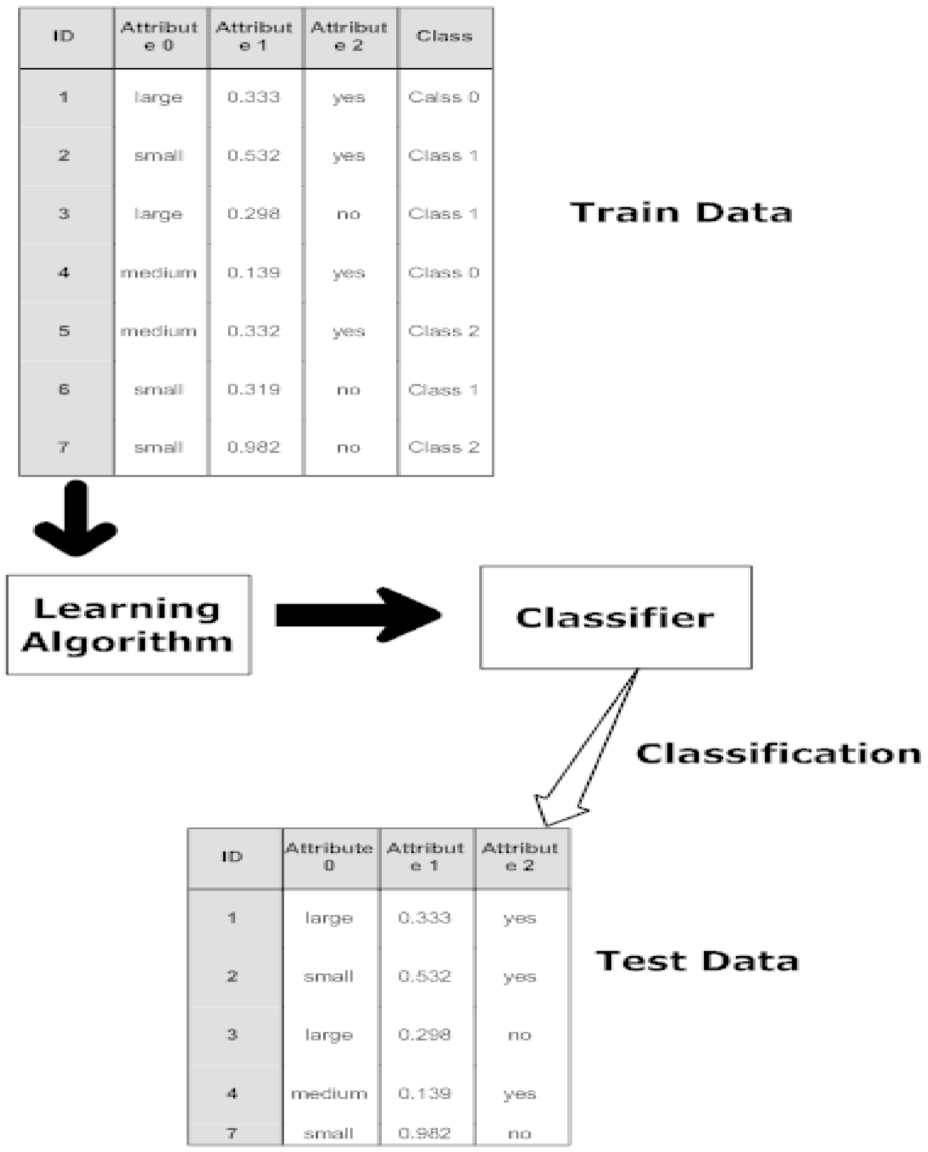
2.1 Classification

In the context of this project, classification is a way to label a given data to a known group according to the corresponding data features.

In other words, if we have a set of K known classes L = {*c1, c2, c3, …., ck*}, and a set of data X = {x1,x2, …., xm}. Classification should be a function F, by which F(X) =ci Є L, 1 ≤ i ≤ k.

The following diagram illustrates an imagined classification task: first, we have a given set of samples with IDs {1, 2 … 7}, named *"Train Data"*. For each sample, there are 3 attributes, named features, and one class label. And also, we have a set of samples with unknown class labels, named *"Test Data"*. The task is to "learn" the relations between the known *Train Data* features and their corresponding class labels, and uses the learning results to identify the unknown class labels of *Test Data*. But before classifying *Train Data,* we must sure that *Test Data* and *Train Data* belong to the same set of classes (i.e. they have at least some relations with each other).

In this diagram example, we can see that we can apply one machine learning algorithm on Train Data to generate a classifier. And later, we use the generated classifier and the unknown Test Data features to find out the class labels.



ID	Attribut e 0	Attribut e 1	Attribut e 2	Class
1	large	0.333	yes	Calss 0
2	small	0.532	yes	Class 1
3	large	0.298	no	Class 1
4	medium	0.139	yes	Class 0
5	medium	0.332	yes	Class 2
6	small	0.319	no	Class 1
7	small	0.982	no	Class 2

ID	Attribute 0	Attribut e 1	Attribut e 2
1	large	0.333	yes
2	small	0.532	yes
3	large	0.298	no
4	medium	0.139	yes
7	small	0.982	no

In the microarray research areas, classification has a significant challenge which is how to choose a learning algorithm which can learn efficiently from thousands and millions of gene features. In machine learning, there are two principal classification techniques: one is unsupervised learning and the other one is supervised learning.

2.2 Unsupervised Learning

Unsupervised learning is a technique to seek to summarize and explore the features of a data set. The difference from supervised learning is that unsupervised learning involves statistical process analysis from only unlabeled train data. One of the classification tasks that use unsupervised learning technique is "Clustering".

In a clustering problem, the task is to partition a raw data into subsets, named clusters. The data in the same cluster should have some common traits to different from the data of other clusters.

Some notable application fields of clustering analysis are data mining, image processing, artificial intelligence etc.

2.3 Supervised Learning

Supervised learning is a technique for learning the relations between data features and data labels. The diagram showed in the section *2.1 Classification* is actually a supervised learning classification case. In order to solve a supervised learning problem, there are at least three requisites: train data, the known data class labels and learning algorithm.

One of possible use example of supervised learning is weather forecast. A weather forecast system normally applies learning algorithm on the history weather observations to find out the relations between observation data and weather. And then, the obtained relations are used in the future weather forecast.

As unsupervised learning, supervised learning also has a lot of approaches and algorithms, such as: decision tree learning, case-based reasoning, Gaussian process regression, learning automata etc. In the context of our classifier system, decision tree learning will be stated with details in Section 2.5 Classifier and Classification Algorithm.

2.4 Gene Expression Data

Gene is the basic unit of an organism. It contains all heredity information which could pass the living things' traits to their offspring. Gene expression data contains those gene information measured by DNA microarray and is used to in the diverse bioinformatics related researches, such as: cancer research.

One of my T.C.C. objectives is to construct a gene expression tree classifier system. The system treats a gene expression data as a matrix with N rows and M columns. Each row represents a sample data with M gene features. We will use this convention in the chapter 3 when the classifier system is introduced.

2.5 Classifier and Classification Algorithm

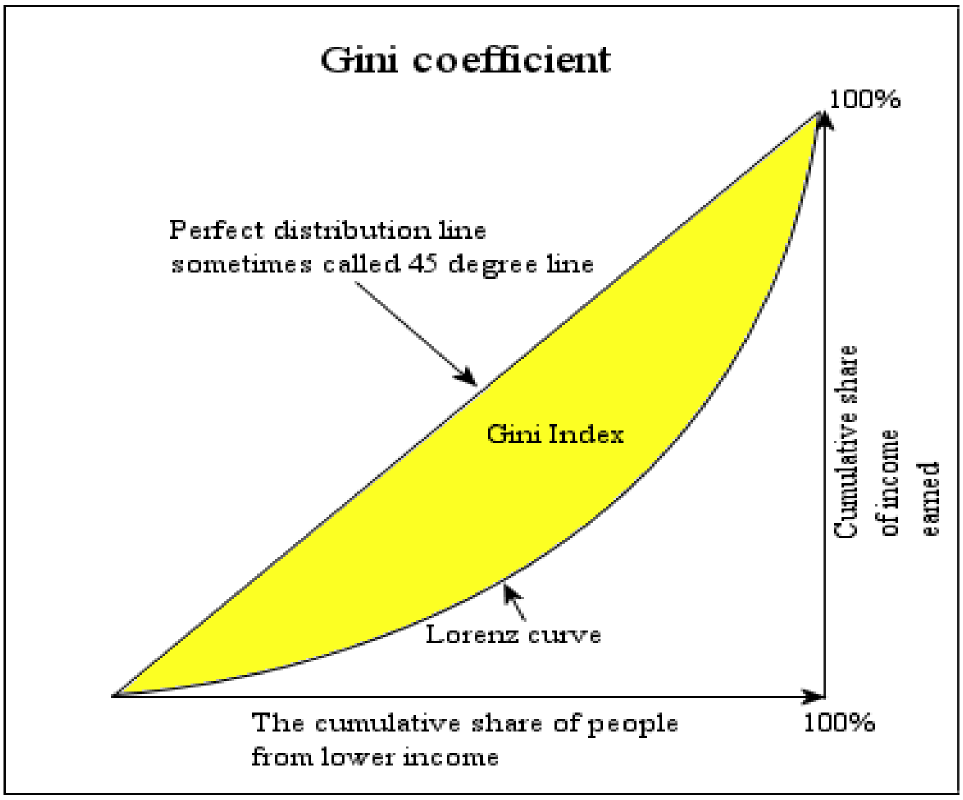
In the context of a supervised learning process, a classifier is a function which is used to classify an object into one of the known classes basing on some related criteria. And a classification algorithm is used to generate classifiers from a set of know train set.

CART, the abbreviation of "*Classification And Regression Tree*", is a popular tree-based classification and regression algorithm.

Normally, a CART algorithm runs under 3 principal steps:

The first step is to choose splitting attributes. At each node of tree, attributes of train data are evaluated in order to find out the basis for separating the classes of the train samples. One of the typical evaluating functions is: Gini index.

Gini index is usually used to measure inequality of distribution, for example: inequality of income distribution or inequality of wealth distribution etc. Below is a widely used Gini index diagram to measure inequality of income distribution.


Gini coefficient
100%
Perfect distribution line sometimes called 45 degree line
Gini Index
Cumulative share of income earned
Lorenz curve
The cumulative share of people from lower income
100%

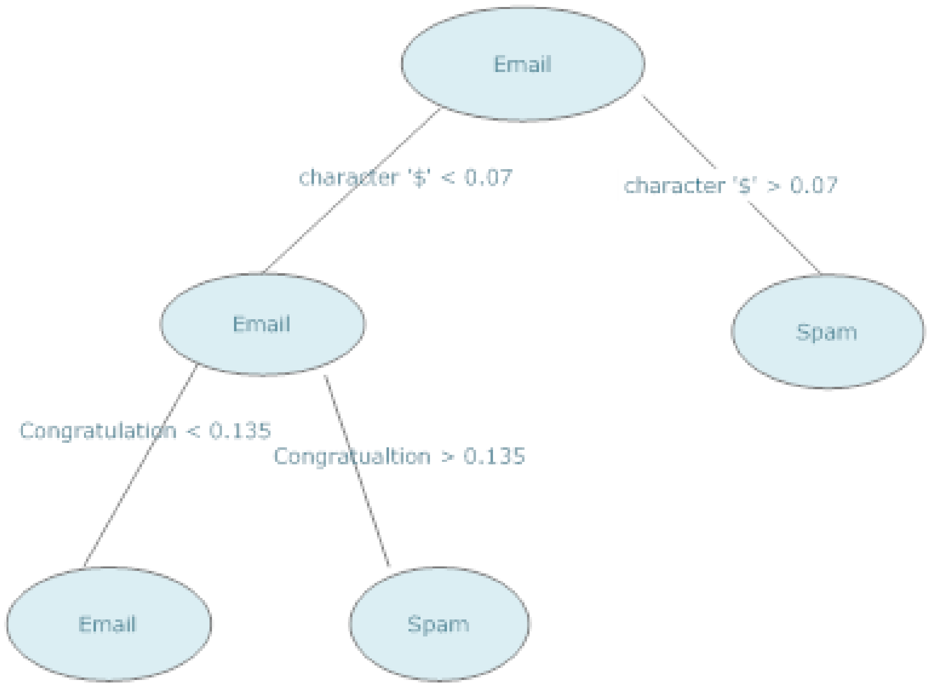
The calculation of Gini index is:

$$G = 1 - 2\int_0^1 L(X)dX.$$

Where L(X) represents the Lorenz curve.

The second step is to determine splitting criteria. Generally, a good splitting criteria is to produce “the purest” nodes in the smallest decision tree. But for different research experiment requirements, the notion of “the purest node” could be varied. In our classifier system, the purest code means a node with the lowest misclassification ratios (i.e. the ratio between the returned label results and the nodes’ actual labels.)

And the third step is to determine when to stop split. The stopping criterion varies a lot according to the different experiments. Sometimes, researchers want the splitting stop after all tree nodes are pure (i.e. perfectly classified). But sometimes, it is not viable to classify all tree nodes because of that the tree size could increase quickly. So some adapted criteria would be used instead of perfection.



The diagram shows a typical CART tree of a simplified Email/Spam verification system

In order to explicate how a CART tree can be used, we will see a very simplified Email/Spam verification system example. The CART tree of this system is looked like the diagram showed above.

In this decision tree, the splitting attributes are the character ‘$’ and the word “congratulation”. The splitting criteria are: i) appearance of (‘$”) < 0.07 * number of email message words and ii) appearance of (“congratulation”) < 0.135 * number of message words. Since there are only two questions (i.e. splitting criteria), the stop splitting criteria is to get all questions asked and answered.

So when a new message arrives, the verification system will first check if the percentage of appearance of the character ‘$’ is less than 7% or not, if not the email is a spam, if so, the second question is asked, whether the percentage of appearance of the word “congratulation” is less than 13.5%, if so the message is an email, otherwise it is a spam.

3. Classifier System – Classys

After the studies of the related topics, my TCC progresses to its second step. At this step, my main concern is how to build a robust and portable classifier system by using all the studies result of the step 1 and my computation knowledge.

3.1 Picking Classification Algorithms

There are several approaches and algorithms used to perform predictive analysis, such as parametric and non-parametric techniques, regression and machine learning techniques etc.

Each algorithm has its own different technique and complexity, and also for different testing data, a same algorithm could represent completely different performances. Instead of picking a fixed classification algorithm for the classifier system, I decide to make a flexible system which could accept a set of candidate classification algorithms from the users. At the same time the system also provide some default supported and tested classification algorithms.

The tested classification algorithms by the classifier system, Classys, are:

- Recursive Partitioning and regression trees, provided by the R package *rpart.*
- Classification and Regression trees, provided by the R package *tree*.
- Linear Discriminant analysis, provided by the R package *MASS*.
- Quadratic Discriminant analysis, also provided by the R package *MASS*.

There are two ways to a user to provide a specified algorithm: by providing an algorithm by an xml file or by calling system provided functions.

In the case of an algorithm is provided by an xml file: the xml file must follow a pre-defined schema as showing in the below example.

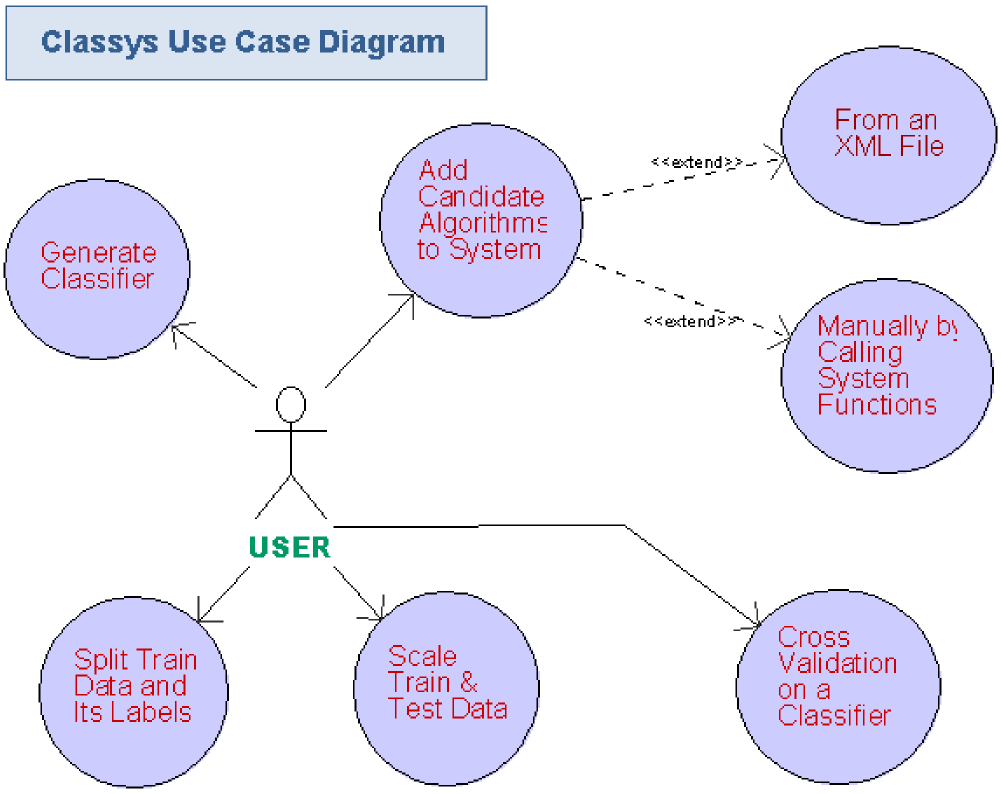
```
<?xml version="1.0" encoding="UTF-8"?>
<list>
      <algorithm>
            <is.ignored>FALSE</is.ignored>
            <name>my.tree</name>
            <package>tree</package>
            <function>
                  <name>tree</name>
                  <pass.arguments>TRUE</pass.arguments>
            </function>
            <train>
                  <parameter.name>data</parameter.name>
                  <combine.with.labels>
                        TRUE
                  </combine.with.labels>
                  <make.data.frame>TRUE</make.data.frame>
            </train>
            <labels>
                  <parameter.name>formula</parameter.name>
                  <formula>
                        <left.hand>labels</left.hand>
                        <right.hand>.</right.hand>
                  </formula>
            </labels>
            </algorithm>

      </list>
```

In this example, a tree function from the “tree” package is provided to the system. The system will read the package name which is “tree”, and load the package if necessary, and then read the function name which is coincidently “tree” and the function parameters. With these information, the system knows how to pass the train data and the labels to the algorithm function and generate the corresponding classifier.

The other way is to call system functions: csys.make.algorithm(…) and csys.add.algorithm(…). The details are in the section *3.2 Classifier System Architecture*

3.2 Classifier System Architecture


Classys Use Case Diagram
Generate Classifier
Add Candidate Algorithms to System
From an XML File
<<extend>>
<<extend>>
Manually by Calling System Functions
USER
Split Train Data and Its Labels
Scale Train & Test Data
Cross Validation on a Classifier

As shown in the system use case diagram, the classifier system, named “Classys”, can be considered as a join of 5 use cases:

a. Add candidate algorithms to system:
   (a) From an XML file: user can add a set of algorithms from a local or web based XML file by calling: csys.load.algorithm(file.name). The XML file format must be the same showed in the section 3.1.
   (b) Manually: by calling two functions csys.make.algorithm(…) and csys.add.algorithm(…). When a user calls csys.make.algorithm(…), the following parameters are needed:
      (i) name – the algorithm name, a user defined name to identify algorithm.
      (ii) package – the package name.
      (iii) func.name – the algorithm function name.
      (iv) train.name – the train data parameter name, which will be used when algorithm function is being invoked.
      (v) labels.name – the labels parameter name, which will be used when algorithm function is being invoked.
      (vi) pass.argument – if system function parameters will be passed to algorithm function. Its value must be either “TRUE” or “FALSE”.
      (vii) cbind – if the train data is combined with its labels. This option sometimes is required by some classification algorithm. Its value must be either “TRUE” or “FALSE”.
      (viii) data.frame – if the train data is in a data.frame[4] format. Its value must be either “TRUE” or “FALSE”.
      (ix) labels.formula.left – if the labels need to be passed to algorithm function in a formula[5] format. The symbol which locates at left part.
      (x) labels.formula.right - if the labels need to be passed to algorithm function in a formula format. The symbol which locates at right part.

b. Generate classifier from a set of candidate algorithms: by calling csys.get.classifier(train, labels, algorithm.name, supported.algorithms, …)
   (a) train – train data. It could be a matrix or a data.frame
   (b) labels – train data labels, It should be a vector of string values.
   (c) algorithm.name – the candidate algorithms’ names. It should be a vector of string values
   (d) supported.algorithms – all supported algorithms in the data.frame format. Normally it is obtained by calling csys.add.algorithm(…) or csys.algorithm.load(…)

[4] data.frame is an object format in R environment. For more details about data.frame. please see R related manuals which can be found here: http://cran.r-project.org/manuals.html

[5] A formula is a string format with two parts divided by “~”, for example: “labels ~ .” is an acceptable formula. Please see corresponding algorithm function manuals to know which are acceptable formulas.

c. Cross-Validation on a classifier: by calling csys.cv(train, labels, classifier, iterance, …)
    (a) train – train data. Matrix or data.frame
    (b) labels – train data labels.
    (c) classifier – classifier generated by csys.get.classifier(…)
    (d) iterance – the number of iterance of cross validation on the given classifier.

d. Split data: by calling csys.split(data, labels, ratio, mask)
    (a) data – data to be split
    (b) labels – data labels to be split
    (c) ratio – splitting ratio
    (d) mask – user specified splitting index

e. Scale data: by calling csys.scale(train, test, scale.method)
    (a) train – train data
    (b) test – test data
    (c) scale.method – the scale method must be one of three supported methods: “max-min”, which scales max value to 1 and min value to 0; “mean-sd”, which scales mean to 0 and mean+sd to 1; “median-mad”, which scales median to 0 and median+mad to 1.

3.3 Notable Characteristics of *Classys*

Some notable characteristics of Classys are:

- Modular system design brings more use cases to the system users.
- The system users can use specified candidate algorithms by providing a formatted xml file which contains all algorithm related information
- All system process results are disposable to the users
- Test demos and documentations permit new users that learn to use the system quickly

4. Conclusion

As my first experience with bioinformatics, this T.C.C. gave me an opportunity to study bioinformatics and statistical related topics and learn how computer science, such as algorithms and software engineering can be used in other science research areas such as biology and medicine related researches, and improve the research results.

Also, I realized the importance of software engineering in software implementing area, and how a well designed architecture can change completely the performance of software.

Now, seeing the result, I am very glad that the whole my T.C.C. experience is worthy of all my efforts of these months.

5. Bibliography

Duda, R.O., Hart, P.E. and Stork, D.G., Pattern Classification, Wiley, 2000

Esteves, G.H., MaigesPack, R, http://www.bioconductor.org/packages/2.3/bioc/html/maigesPack.html

Freeman, E., Bates, B. and Sierra, K., Head First Design Patterns, O'Reilly, 2004

Friedman, J., Hastie, T. and Tibshirani, R., The Elements of Statistical Learning; Data Mining, Inference, and Prediction, Springer, 2001

Hill, T. and Lewichi, P., Statistics: Methods and Applications, StatSoft, Inc., 2006

Jones, N.C. and Pevzner, P.A., An Introduction to Bioinformatics Algorithms, The MIT Press, 2004

Ripley, B.D. and Venables, W.N., Modern Applied Statiscs with S, Springer, 2002

rpart, Recursive partitioning and regression tree packages, R, http://cran.r-project.org/web/packages/rpart/index.html

The R Manuals, R, http://www.vps.fmvz.usp.br/CRAN/manuals.html

tree, Classification and regression trees, R, http://cran.r-project.org/web/packages/tree/index.html

Wikipedia, http://en.wikipedia.org

Zvelebil, M. and Baum, J., Understanding Bioinformatics, Garland Science, 2007

PART II

1. Challenges & Frustrations

Well, I would like to say that my B.C.C. life at IME-USP is filled full of challenges and also some frustrations.

I think the big challenge that I met during these years is my communication difficulty, which not only caused my difficulty in my study but also caused other things like misunderstanding. These things certainly brought me frustration feelings.

But at the other side, I am also very grad to have luck and the opportunity to be able to study in IME-USP, one of the most respected institute of Brazil and the world and being graduated from there.

2. Related Courses to TCC

Before listing the related courses, it is very important to remember all professors who ministered the courses, and their excellent quality.

MAC0122 – Princípios de Desenvolvimento de Algoritmos introduced me to the computer sciene study.

MAC0328 – Algoritmos em Grafos and MAC0338 – Análise de Algoritmos, these two courses constructed my algorithms' base, and also I learned to understand how to build efficient algorithms.

MAC0332 – Engenharia de Software and MAC0441 – Programação Orientada a Objetos, these two curses are fundamental to help me to design and construct the classifier system.

MAE0121 - Introdução a Probabilidade e a Estatística I , MAE0212 - Introdução a Probabilidade e a Estatística II and MAE0228 - Noções de Probabilidade e Processos Estocásticos. The topics studied from these courses are fundamental to help me to build a statistical classifier system. Probability notions are used to calculate the classifiers accuracy etc.

A lot of other BCC courses do also relate directly or indirectly to my TCC project, but the courses listed above are more significant.

3. Acknowledgement

I would like to use this space to acknowledge the contribution of my tutor Prof. Eduardo Jordão Neves, thank him for the project opportunity and his dedications on my project. Also, I would like to thank all the professors who taught me during these years.