Lab 10. Microarray data analysis 2

1. Objective

- To implement, train and validate disease classifier using machine learning method.
- To learn how to use C++.
- To create disease classifier using decision tree through C++.
- To validate disease classifier using cross-validation through C++.

2. Background

2.1. Classification of conditions using selected genes with machine learning algorithm

We can develop a classifier that identifies the conditions (e.g. normal vs. cancer) using the selected genes that show significantly differential expression in two conditions. We can estimate the possible conditions of an unknown sample by measuring and testing the expression levels of the selected genes using this classifier. To develop a classifier, we begin with gene expression values of the genes from known conditions (e.g., normal vs. cancer) and "train" an algorithm to learn a rule to determine the conditions. Positive and negative examples are used to train the algorithm. The algorithm is then applied to test samples that are not used in training, and its accuracy as a predictor or classifier is assessed. One can use genes randomly and test the performance iteratively to build the best classifier by chance. However, if start with gene sets discriminating two conditions significantly, you can build classifier more efficiently.

2.2. Machine learning

To find a rule classifying the condition out of differential data, we can use machine learning methods. Machine learning is a branch of research area where the algorithm changes to better algorithm automatically through learning experience. Algorithms that implement machine learning are such as Artificial Neural Network (ANN), Decision Tree (DT), Genetic Algorithm, and Support Vector Machine. Here we introduce the details of DT for our lab experiment.

2.2.1. Decision Tree

Decision tree is a graphical representation of procedure through accumulated data to predict the result. Decision tree has the root node at the top and it branches down to the leaf node. The record is inputted at the root node then it is decided to which branch it will go down to. The rule for the selection of branch is decided with algorithm. One of the most popular algorithms for decision tree in data mining area is called ID3. ID3 algorithm uses "information gain" of attribute to find the attribute to use for decision. Until it reaches the leaf node this procedure is repeated to find the rule of selection of branch.

Since the data analysis through decision tree can be graphically shown as a tree, the analyst can understand and explain the mode. (Figure 1)

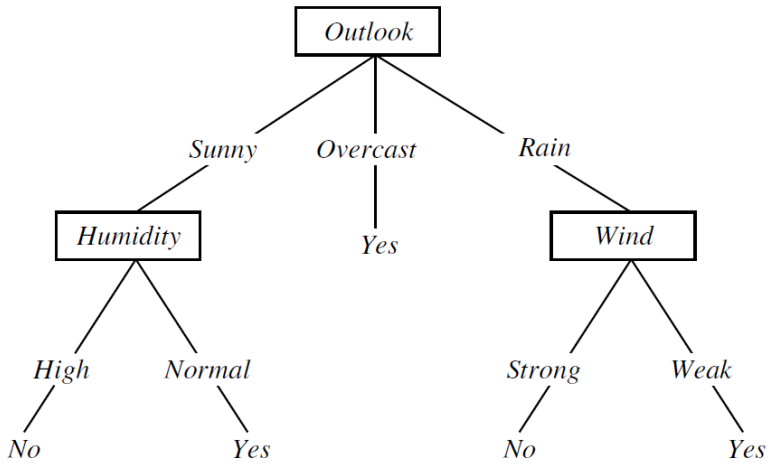

Figure 2. An example of decision trees to decide whether to play or not with weather changes. All decision tree starts from the root and attribute to use for the branching down is selected with information gain result.

2.3. The ID3 Heuristic Algorithm

The ID3 heuristic uses this concept to come up with the "next best" attribute in the data set to use as decision criteria, a node in the decision tree. In this lab, each gene can be an attribute of sample to decide to separate samples, which imply characteristics of a sample.

The entropy is a measure of uncertainty of a state of samples. In other words, it is related with the amount of information needed to specify the exact states of data (or system) given its specifications. When a state of samples is heterogenous the entropy is high, however, if the state is homogenous the entropy is low. When imagine a heterogenous state of samples, if the samples are grouped with similar samples together and different samples apart, then the entropy of state of each group of samples will be lower.

Thus, the idea behind the ID3 heuristic is to find the attribute that most lowers the entropy for the data set, thereby reducing the amount of information needed to completely describe each piece of data separated by the attribute. By following this heuristic you will essentially find the best attributes to classify the records (according to a reduction in the amount of information needed to describe the remaining data division) in the data set.

2.4. Entropy and Information Gain

In ID3 heuristic algorithm, the information gain of separation of samples are used to choosing "next best" attribute. The information gain is the amount of information gained about a state of samples from knowing an attribute, component of state of sample.

When the entropy of state of sample having binary class can be calculated with:

$$H(\mathrm{S}) = -p\ \ log(p) - (1-p)\ \log(1-p)$$

Where S denotes a state of samples, p is proportion of sample with a certain class, and 1-p is the proportion of sample with the other class.

And the information gain can be calculated by change in entropy of state of sample from a prior state to a state some information given an attribute. Let D denote a set of samples, each of the form $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{k})$ where x_a is the value of the a^{th} attribute, then the state of sample can be denoted by $\mathrm{S}_{\mathrm{a(v)}} = \{\mathbf{x} \in \mathrm{D} | \mathrm{x}_\mathrm{a} = v\}$. Information gain, IG (S, A) = expected reduction in entropy due to sorting on A is given:

$$\mathrm{IG}(\mathrm{S}, \mathrm{A}) = H(\mathrm{S}) - \sum_{v \in Values(A)} \frac{|S_{A(v)}|}{|S|} H(S_{A(v)})$$

Where, A is an attribute, S is a prior state of samples, $S_{A(v)}$ is state of samples in S given a certain value v of attribute A. $|\cdot|$ denote the size of state, in other words, the number of samples

For example, when we decide rule for separate cancer and normal samples into two group, if chosen gene is 'Gene1' and the chosen cutoff is 2.0 in the case of (figure 3). Then,

Entropy of samples in input: $H(\{norm: 4, cancer: 4\}) = -\frac{4}{4+4}\log_2\frac{4}{4+4} - \frac{4}{4+4}\log_2\frac{4}{4+4}$

Entropy of samples with higher than cutoff: $H(\{norm: 0, cancer: 2\}) = -\frac{0}{0+2}\log_2\frac{0}{0+2} - \frac{2}{0+2}\log_2\frac{2}{0+2}$

Entropy of samples with lower than cutoff: $H(\{norm: 4, cancer: 2\}) = -\frac{4}{4+2}\log_2\frac{4}{4+2} - \frac{2}{4+2}\log_2\frac{2}{4+2}$

And information gain: $\mathrm{IG}(\mathrm{S}, \mathrm{Gene1}) = H(\{norm: 4, cancer: 4\}) - \frac{2}{8}H(\{norm: 0, cancer: 2\}) - \frac{6}{8}H(\{norm: 4, cancer: 2\})$

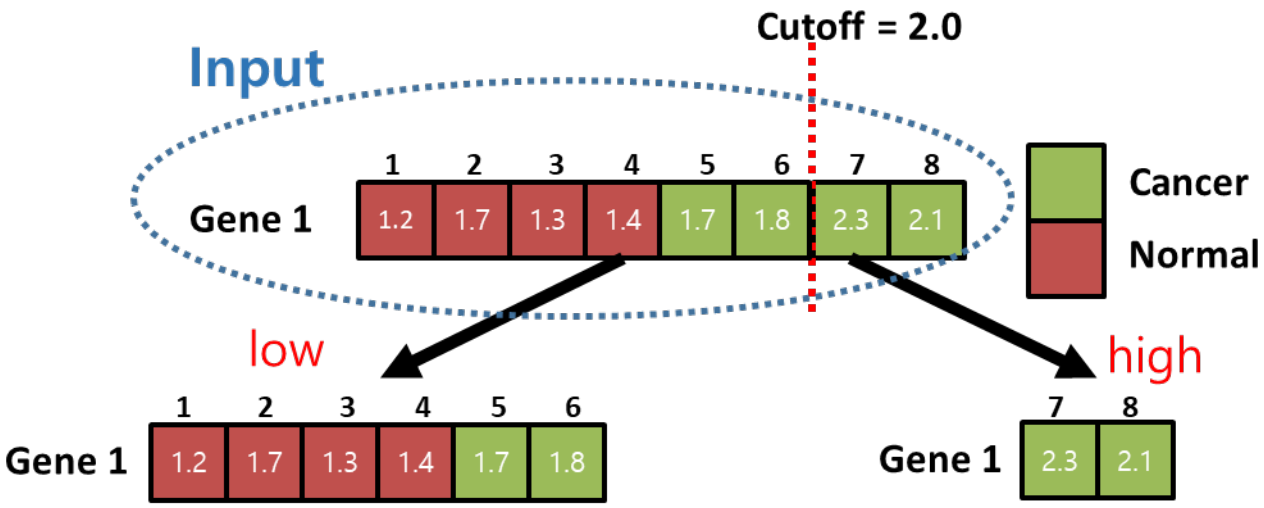

Figure 4 Separating samples with given gene and cutoff value. There are total 8 samples (cancer: 4, normal:4) and the samples are separating into two groups by cutoff of 2.0 of 'Gene1' expression value.

2.5. The decision tree learning algorithm

With most of the preliminary information out of the way, you can now look at the actual decision tree algorithm. This part is the main function used to create your decision tree.

First you will load datasets from files. Before decision tree learning algorithm starts, the attributes and their values for each sample should be stored in proper datatype, e.g. list and map using the C++ standard library, to be searched easily by gene index or sample index, because they will be frequently recalled to calculate entropy and information gain. In this lab, the expression of genes for each sample and the differential expressed genes (DEGs) will be loaded.

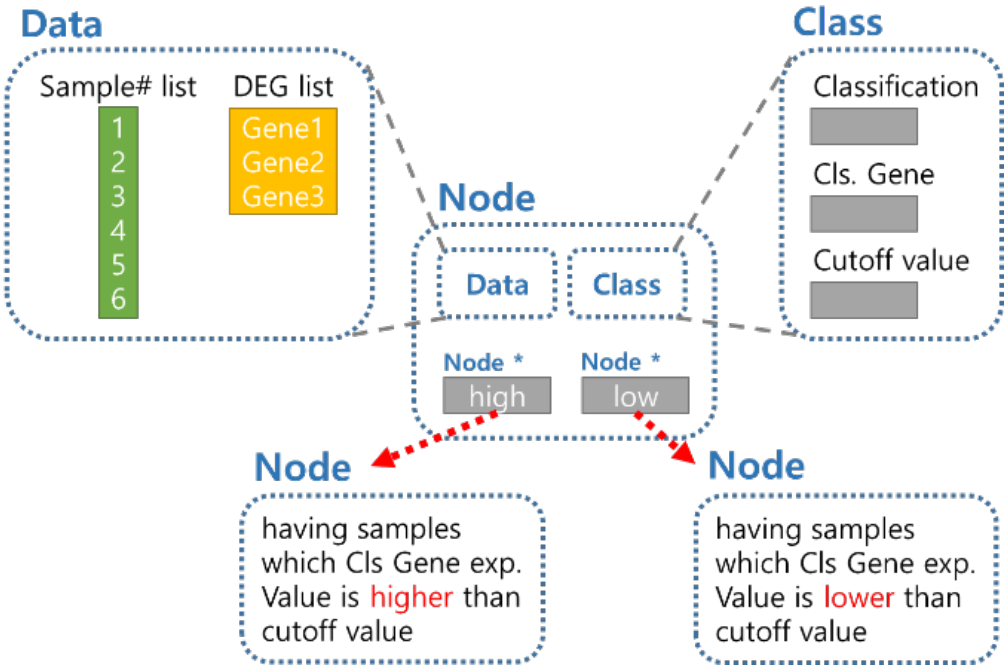

Figure 5. An example of Data Structure for a Node in the Decision Tree in this Lab.

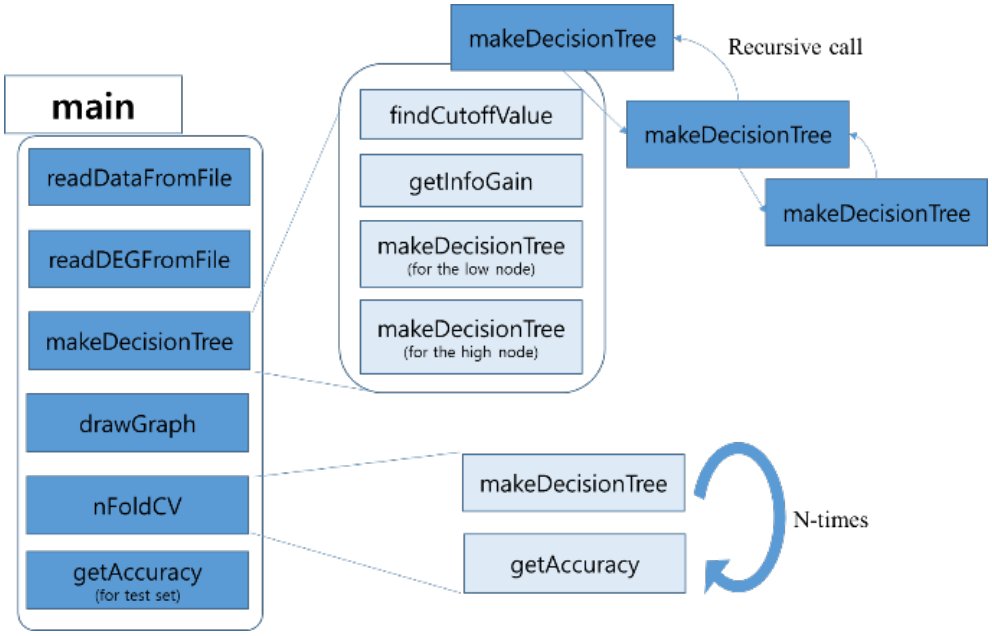

Figure 6. Brief schema of implementation of the algorithm in this Lab.

Now we need to implement the real essential of the algorithm. Let's say this algorithm will be implemented in *makeDecisionTree* function. The algorithm makes use of recursion to create the decision tree through tree branches. You can use proper datatype like `struct` to store local datasets like remaining sample and attributes list, specified to each node of tree.

And as such it needs a base case (or, in this case, two base cases) to prevent it from entering an infinite

recursive loop. For starters, if either the remaining sample or attributes list is empty, then the algorithm has reached a stopping point. If either list is empty, then the algorithm returns a default value. It returns the value with the highest frequency in the data set for the labels of samples. The only other case to worry about is when the remaining samples in the data list all have the same value for the labels of samples, in which case the algorithm returns that value.

When everything else is normal (that is, the remaining sample or attributes lists are not empty and the samples in the data still have multiple values for the labels of samples), the algorithm needs to choose the "next best" attribute for classifying the test data and add it to the decision tree.

This process is responsible for picking the "next best" attribute for classifying the samples in the data set. After this, the process creates a new node containing attributes list except the newly selected "best" attribute. Then the recursion takes place. In other words, each of the subtrees is created by making a recursive call to the *makeDecisionTree* function and adding the tree to the newly created node in the last step.

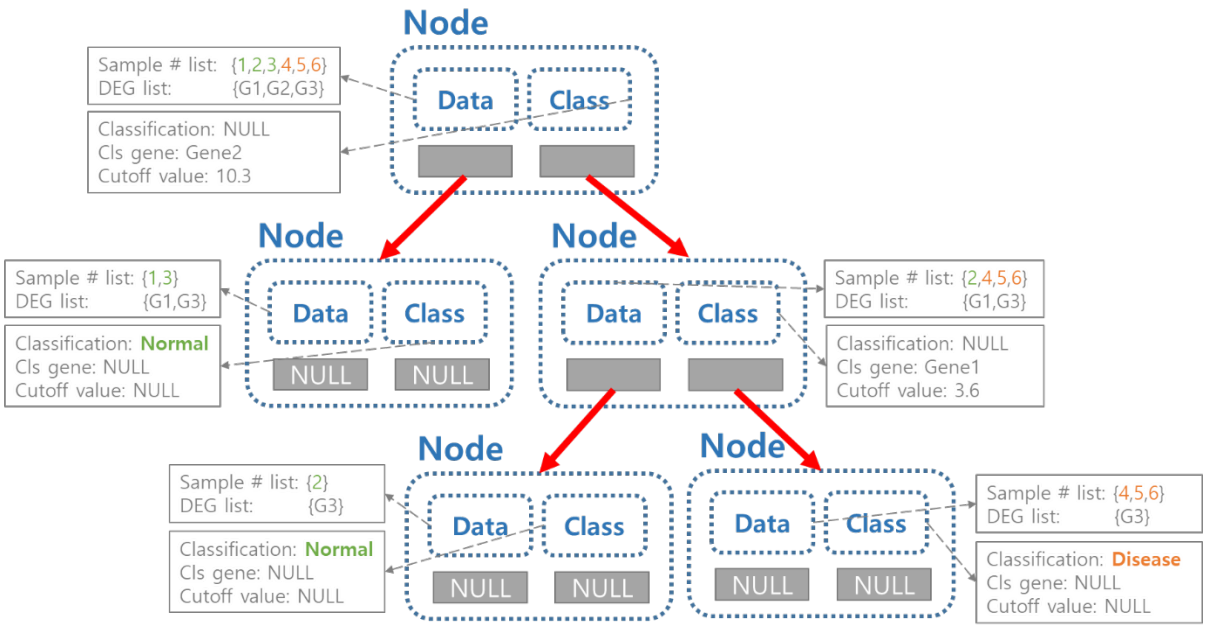

Figure 7 An Example of Decision Tree. This figure describes the completely grown decision tree. From root to leaf node.

2.6. Validation of classifier

The accuracy measurement of machine learning algorithm is shown below.

		Gold standard		
		True	False	
Predicted condition	Positive	True positive	False positive	Positive predictive value
	Negative	False negative	True negative	Negative predictive value

	Sensitivity	Specificity	Accuracy

$$\text{Accuracy} = \frac{number\ of\ true\ positive + number\ of\ ture\ negatives}{numbers\ of\ true\ positives + false\ positives + false\ negatives + true\ negatives}$$

2.6.1. N-fold Cross-validation method

In N-fold cross-validation, the data set is divided into N subsets, and the holdout method is repeated N times. Each time, one of the N subsets is used as the test data and the other N-1 subsets are used as training data. Then the average error across all N trials is computed. If you use a 3-fold cross-validation method for calculating accuracy: the data set is randomly divided into thirds (set1 1, 2, and 3) and each set has the same size. Set 1 and 2 are used to train the machine learning algorithm; then the algorithm is tested on set 3 as the "unknowns." Next, sets 1 and 3 are used for training and set 2 is tested as the unknowns. Finally, sets 2 and 3 are used for training, and set 1 is tested. The result of three estimated errors from the folds can be averaged to generate a single estimated error.

3. Prelab activities

3.1. Notice: Prepare Datasets and Guideline Code

- Implement your own codes using given prelab guideline code (C++) from Lab: "Microarray Data Analysis 1" with given input data by TA.
- You can refer to the C++ references from online (e.g. cplusplus.com). But you must describe correct citation and write down the description of the part you get hint on your report in your own words.

3.2. Implement "readDataFromFile()" function, for reading expression data and sample class from file.

① open the data file with `filename`. read first line (sample classes). read following lines (expression values for a gene of each sample)

② insert data from file to proper lists. The sample index and class of each sample should be saved at `sampleCls`. The expression values for genes of sample is saved at `exp`.

`sampleCls`

Sample id	Sample cls
...	...

`exp`

Sample id, GeneName	Expression Value
...	...

3.3. Implement "readDEGFromFile()" function, for reading differentially expressed gene list from file.

① open the data file with filename. read lines (the name of DEGs)

② insert data from file to list of string. e.g. `deg`

③ return the list of name of DEG (string)

3.4. Implement "findCutoffValue()", for finding a best cutoff of expression of a gene, that can separate samples into normal and cancer.

① For samples in `sampleList`, get expression value of a gene.

② Within the samples in `sampleList`, find the cutoff value of expression of a gene.

③ For a certain gene, in example, if normal samples show (1, 2, 3, 5) as expression value and cancer samples show (4, 6, 7, 8), then the mean of all sample, in this case 4.5, can be used to classify two different condition. Any of your own idea for finding classifying value is okay (additional score could be given).

④ Describe the way you find cutoff values in your source code, why you choose it among various ways to find cutoff.

3.5. Implement "getInfoGain()" and "entropycal()", for calculating information gain for an input gene and sample list.

① For samples in `sampleList`, get expression value of a gene.

② Within the samples in `sampleList`, calculate information gain of separation of sample set with a certain cutoff value.

③ Calculate entropy of state of samples with the proportion of samples in each class.

4. Mainlab activities

- Classification of samples with Decision tree: our objective is to classify disease state of a new sample with only gene expression but no disease state by using decision tree trained with given datasets.
- There are three steps to achieve: making decision tree, predicting with trained tree, validating performance with cross validation.

4.1. Notice: Prepare Datasets and Guideline Code

① You'll get total three txt files (tab-deliminated). Two of them are **training set** and **test set data files**, of similar form to last lab (lab 9)'s output file. But they are independent to lab9's output file. Another txt file is **DEG list**. You will train decision tree (DT) classifier using training set and DEG list. And you will test the classifier using test set.

② Use your prelab code on proper positions of **mainlab guideline code**. Locations are indicated with "//[Problem] copy and paste from your prelab code"

③ Implement decision tree by filling the blanks on mainlab guideline code file following instructions. Locations are indicated with "//[MainLabProblem]." Hint: you can get insights to implementation by looking at (Figure 8 and 9)

4.2. Making Decision Tree

① Implement "makeDecisionTree" function - This function will find a gene with ID3 algorithm and make child node.

② Start with 'Main function' part. Read the comments and code carefully, focusing on class and datatype.

③ Look at preparing data of root node for a parameter of makeDecisionTree function that generates decision tree.

④ find the gene with maximum information gain in remaining genes by iteratively compare each gene.

A. find the cut off value for the gene using `findCutoffValue` function in prelab.
B. calculate the information gain for the gene and optimal cutoff value for that gene and compare information gain of genes.

⑤ make child node, assign and return proper object to call recursively 'makeDecisionTree'.

Final Report Question 1. What is Definition of 'information gain' with fully explained equations? How is it applied in your ID3 algorithm?

Final Report Question 2. Which gene having maximum information gain?

4.3. Predicting with Trained Decision Tree

① Implement "predSample" and "getAccuracy" function – these function will predict disease state for a new sample and calculate accuracy of the prediction.

② Check node's classification value(clsResult) is not 0, then return true for 1(normal) or return false for 2(tumor).

③ Check the `clsGene` and `cutoffValue` and compared with input sample's expression to determine whether move to high or low node.

④ Recursively call the function(predSample()) with determined high/low Node.

Final Report Question 3.

Describe how you implement the *predSample* with recursive call of the function.

4.4. Validating Performance with Cross Validation

① Implement “nFoldCV” function – this function help you to validate your trained model with k-fold cross validation

② please read carefully the pre-written code to randomize samples for validation

③ complete the function 'nFoldCV' by using predSample and getAccuracy functions.

Final Report Question 4. Vary your ID3 heuristic algorithm choosing best gene from DEGs or function `*findCutoffValue*`. When is the validated performance is best?

5. Reference

[1] Bioinformatics: Sequence and Genome Analysis, 2nd edition, David W. Mount, 2004, Cold Spring Harbor Laboratory Press.

[2] <Manual for ‘map’ container in standard template library (STL)>https://msdn.microsoft.com/en-us/library/s44w4h2s.aspx (English)
https://www.tutorialspoint.com/cpp_standard_library/map.htm (English)
https://msdn.microsoft.com/ko-kr/library/s44w4h2s.aspx (Korean)