Journal of
Computer Science & Systems Biology

Jain et al., J Comput Sci Syst Biol 2013, 6:5
DOI; 10.4172/jcsb.1000124

Research Article Open Access

Data Adaptive Rule-based Classification System for Alzheimer Classification

Mohit Jain[1]*, Prerna Dua[2], Sumeet Dua[1] and Walter J Lukiw[3]

[1]Department of Computer Science, Louisiana Tech University, Ruston, LA 71270, USA
[2]Department of Health Informatics and Information Management, Louisiana Tech University, Ruston, LA 71270, USA
[3]Neuroscience Centre of Excellence, Louisiana State University Health Sciences Centre, New Orleans, LA 70112, USA

Abstract

Microarrays have already produced huge amounts of valuable genetic data that is challenging to analyse due to its high dimensionality and complexity. An inherent problem with the microarray data which is characteristic of diseases such as Alzheimer's is that they face computational complexity due to the sparseness of the points within the data, which affect both the accuracy and the efficiency of supervised learning methods. This paper proposes a data-adaptive rule-based classification system for Alzheimer's disease classification that generates relevant rules by finding adaptive partitions using gradient-based partitioning of the data. The adaptive partitions are generated from the histogram by analyzing Tuple Tests following which efficient and relevant rules are discovered that assist in classifying the new data correctly. The proposed approach has been compared with other rule-based and machine learning classifiers, and detailed results and discussion of the experiments are presented to demonstrate comparative analysis and the efficacy of the results.

Keywords: Alzheimer's; Partitioning; Classification; Rule tuple test

Introduction

Alzheimer's disease (AD), the most common cause of dementia, is a neurodegenerative disorder which leads to loss of intellectual ability and eventually resulting in death. Early detection of AD is seen as important because treatment may be most efficacious if introduced at its nascent stage. In practice, a diagnosis is largely based on clinical history and examination supported by neuropsychological evidence of the pattern of cognitive impairment [1]. However, the reality is that only about half of those with probable dementia are actually recognized in the primary care setting [2]. Microarrays are at the core of a biotechnology, which assists researchers to analyze the expressions of thousands of genes under different samples (conditions) at the same time. However, researchers face challenge in analyzing the microarray data due to its high dimensionality, noise and complexity in the gene expression dataset, which are characteristic of diseases such as Alzheimer's.

Therefore, it is important to find salient features from the Alzheimer's disease gene expression dataset, which can assist in providing additional information in differentiating sub types of the disease along with the underlying biological phenomenon. Mining such data poses a critical problem with an aim to find out patterns and knowledge from these huge amounts of gene expression data. Therefore there is a need to develop an automatic system, which has the ability to classify the Alzheimer's disease and improve the precision and accuracy of the diagnosis [3,4]. This paper presents a data adaptive partitioning schema, which finds efficient partitions in every gene using gradient-based histogram partitioning approach. These partitions assist in finding the relationship and patterns among different genes in gene expression dataset in the form of rules, which helps in classifying new samples into their respective sub types of Alzheimer disease. Below are the outlines of various related research in the Alzheimer disease classification.

Related Research

Joshi et al. [5] have proposed an attribute evaluation classification approach for the classification of Alzheimer's disease by selecting the most important attributes by using various feature selection methods such as chi-squared, gain ratio etc. and then compared various classification techniques to find best classification technique for the given data such as Neural Networks (NN) and Machine Learning (ML) methods. The limitation of feature selection method is that each feature is considered separately and feature dependencies are ignored, which can adversely affect the accuracy and efficiency of classification techniques. Lee et al. [6] classifies the Alzheimer's data by employing a rough-fuzzy hybrid approach called ARFIS (a framework for Adaptive TS-type Rough Fuzzy Inference Systems). In this approach, the entropy-based discretization technique is employed to find the boundary points on the training data in order to find best partitions in the data which assists in producing maximum information gain. The rough set-based feature reduction method is employed to find the relevant attributes which helps in classifying the future samples into their respective classes. However, they have not used any validation measure to validate their partitions as to whether they are efficient or not. Lopez et al. [7] presented a framework for the classification of Alzheimer's disease by employing kernel based Principal Component Analysis (PCA) to reduce the feature space by transforming the data into non-linear mapping. Linear Discriminant Analysis (LDA) is then applied on the reduced data that groups the data according to their class labels. Finally, this reduced feature space is used to train a kernel-based Support Vector Machine (SVM) classifier, which assists in classifying Alzheimer's disease. Kloppel et al. [8] combined datasets from multiple

***Corresponding author:** Mohit Jain, Department of Computer Science, Louisiana Tech University, Ruston, LA 71270, USA, E-mail: mja025@latech.edu

Received June 28, 2013; **Accepted** August 28, 2013; **Published** September 07, 2013

Citation: Jain M, Dua P, Dua S, Lukiw WJ (2013) Data Adaptive Rule-based Classification System for Alzheimer Classification. J Comput Sci Syst Biol 6: 291-297. doi:10.4172/jcsb.1000124

sources and used SVMs to form a classification system by classifying the grey matter segment of T1-weighted MR scans from Alzheimer's disease patients from two centres with different scanning equipment.

The objective of this work is to develop a partitioning approach in the gene expression data to generate data-adaptive partitions for rule-based classification. The data partitioning schema is based on the premise that rapid but infrequent changes in frequency distribution of the gene expression data can be effectively and efficiently employed as the basis of discovering data-adaptive boundaries and partitioning of the data. These partition labels are then employed in a rule-based classification framework to generate rules, whose specificity and sensitivity in classification is evaluated by classifying new samples into their respective classes. These rules can assist in finding important relationship and insightful patterns between gene expressions and helps in predicting patient sample into a diseased or a healthy sample.

Methodology

Many data mining techniques have been developed to extract potentially useful information or knowledge from large databases. A data-mining technique such as rule-based classification is a category of supervised classification that is employed for discovering knowledge in a variety of application domains. Specifically, for gene expression analysis, rules can represent an important relationship or associations between different genes in a gene expression dataset or relationship between the genes and the classes [9,10]. Table 1 shows an example of a gene expression dataset which has two classes disease and healthy. By applying rule based classification algorithm on a dataset, we can form meaningful rules between the genes and the samples. For example - $g_1, g_2, g_3 \xrightarrow{Rule} disease$, this $g_1, g_2, g_3 \xrightarrow{rule} disease$ rule implies that if a new sample express genes g_1, g_2 and g_3 then the sample is likely to be of type *disease*. Therefore this rule can be used to classify new samples of unknown type as disease. Similarly, other rules can be found such as $g_5, g_7, g_9 \xrightarrow{Rule} Healthy$ which implies if a new sample expresses genes g_5, g_7 and g_9, it is likely to be of type healthy. A rule can contain more than three genes, for example, $g_5, g_7, g_8, g_{10} \xrightarrow{Rule} Healthy$ which implies if a new sample expresses genes g_5, g_7, and g_{10} is likely to be type healthy.

Rule-based algorithms rely on discretization of data and are represented in the form of categorical variables. As the number of partitions increases, the number of rules grows exponentially as shown in Equation 1; this growth can result in a number of insignificant or irrelevant rules, which can hamper the classification system.

$$N = p^d \quad (1)$$

Where, '*N*' are the number of generated rules, '*p*' are the number of partitions and'd' are the number of dimensions in the dataset.

In order to generate efficient and accurate rules, the data need to be split or divided into different partitions, resulting in discretization. Once these partitions are found, they could then be subjected to classification rules or models to observe meaningful associations. The partitions were generated in two different phases. In the first phase, partitions are generated using different Tuple Tests; in the second phase, significant rules are generated from the partitions obtained in the first phase. Since the number of partitions is linearly related to the number of discrete variables that can be found in the data, the partitioning or splitting point of the data is critical to find relevant rules. These rules observed from the transformed data make an efficient classifier. Our objective is to find data adaptive partitions by analyzing the data represented as a histogram and discover the efficient partitions using different Tuple Tests, which helps in generating efficient rules and discards all irrelevant rules. Hence, these efficient rules will assist in classifying new samples into the irrespective classes. We have experimented, compared and validated our obtained partitions and rules on a well known hippocampal gene expression dataset. The proposed algorithmic framework is the gradient-based partitioning rule-based classification method, which is described the section below.

Samples	Expressed Genes	Class Labels
S_1	g_1 g_2 g_3 g_4	Disease
S_2	g_2 g_4 g_6	Disease
S_3	g_{10} g_5 g_7	Healthy
S_4	g_8 g_5 g_7 g_9	Healthy

Table 1: An example of a Gene expression dataset.

Gradient-based Partitioning

In gradient-based partitioning approach, a gradient is calculated for each vertical bar of the histogram to validate whether the bar is a peak or valley, and then the gradient is compared with the default threshold range, which is defined by the user or expert. If the gradient of the bar is less than the lower limit of the threshold range, then the bar is called a valley, while if the gradient of the bar is greater than the upper limit of the threshold then the bar is called the peak. These peak and valley are used in finding the partitions in the dataset by performing different Tuple Tests on the histogram data. These obtained partitions will allow in generating the rules in the data, and these rules will help in classifying the new sample into their respective classes. Figure 1 shows the algorithm for the rule-based classification system.

The algorithm has been divided into three phases as shown in Figure 1. Phase 1 comprises of generating partitions using different Tuple Tests. Phase 2 encompasses rule generation where absolute membership is assigned to values that help find the antecedent of the rule following which the consequent class of the rule is found. In Phase 3, classification schema is transcribed to explain how new samples or data are classified into their respective classes. The proposed methodology and the algorithm with the explanation of each phase are described below.

Phase 1: Generate adaptive partitions

Phase 1 is used to generate adaptive partitions which have been divided into three steps. Step 1 is the data normalization Step 2 is histogram analysis to identify the peak and valley in a histogram and Step 3 is to generate adaptive partitions using different Tuple Tests.

Step 1: Data normalization

Normalization of the data is performed to scale the values and to avoid the effects of extreme values if the data distribution is far from the mean. Data normalization also keeps each dimension of the dataset in a same range, in this case by transforming the raw data into the range of 0 to 1. Due to the large variability in the expression values of the genes, normalization is performed on the dataset by first applying z-score normalization and then min-max normalization which are defined below:

Z-score normalization: In z-score normalization, instances of a variable are normalized based on the mean and standard deviation. It is denoted by z' in Equation 2:

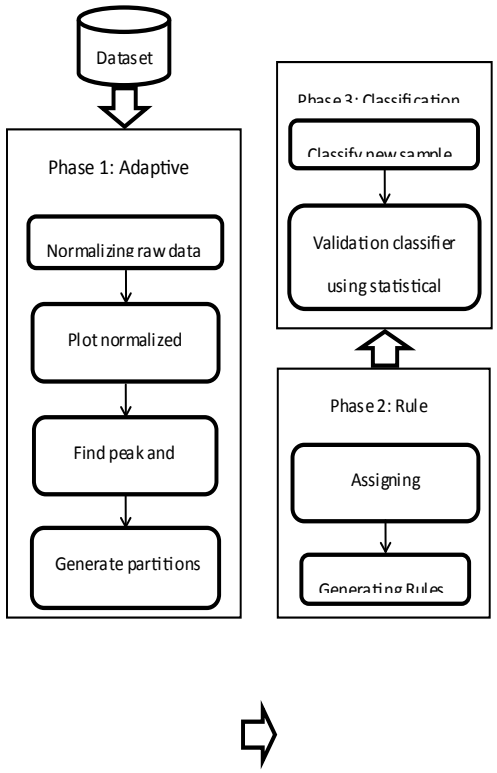


Figure 1: Gradient based partitioning rule-based classification framework.

$$z' = \frac{z - \mu}{\sigma} \quad (2)$$

Where μ = mean of a variable σ = standard deviation of a variable, z is the instance of a variable which is to be mapped in the new range, and z' is the new value of z which is in the new range.

Min-max Normalization: Suppose that $\min_A$ and $\max_A$ are the minimum and maximum values of an attribute A. Min-max normalization maps a value of v_i into v_i' in the range $[new\_\max_A, new\_\min_A]$ by computing the following Equation 3:

$$v_i' = \frac{v_i - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A \quad (3)$$

For example, with $[new\_\max_A = 0, new\_\min_A = 1]$, the range of the data is shifted to [0,1]. In Figure 2, g_1, g_2 *and* g_3, are genes of the dataset where raw data has been normalized into range (0 to 1) by first applying first z-score followed by the min-max normalization. Figure 2 shows the snapshot of the dataset before and after normalization.

Step 2: Identifying peak and valley in a histogram

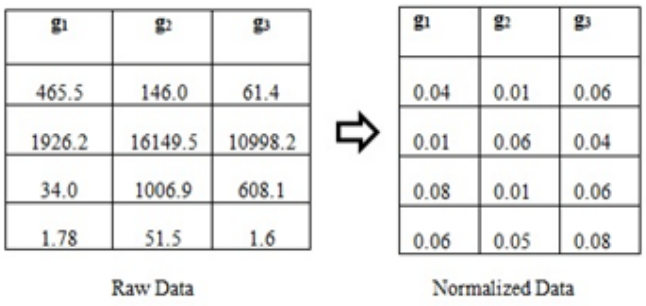
Step 2.1: Histogram is drawn for each gene of the dataset where bins are on the *x*-axis and frequency counts are on the *y*-axis as shown in Figure 3. Frequency count is defined as the total number of data points that lie in that bin range.

Several experiments were conducted to determine the optimum bin width. It was observed that bin width from 0.01 to 0.05 provides better performance. Let us assume gene (g_1) has normalized values [0.01, 0.08, 0.04, 0.09, 0.05, 0.09, 0.07, 0.02, 0.08, 0.03, and 0.085]. These normalized values have been plotted into the histogram in Figure 3, where there are 5 bins each of width of 0.02. In Figure 3, first bin ranges from 0 to 0.02 and frequency count is 2, as there are two normalized values for g_1 (i.e., 0.01 and 0.02) that fall in this range. Similarly, width of the second bin ranges from greater than 0.02 to 0.04 and frequency count is 2 (0.03, 0.04), the third bin ranges from greater than 0.04 to 0.06 and frequency count is 1 (0.05) and so on.

If there are 100 genes in a dataset, then there will be 100 histograms and each histogram is independent of the other i.e., at this point we are not concerned about the dependencies between the genes and consider each gene independently. Our objective is to find efficient partitions for each gene in the gene expression dataset.

Step 2.2: Peaks and valleys are identified once the normalized data is plotted into the histogram. The default threshold range varies from -λ to λ where λ is a number set by a user or expert. We performed various experiments by varying the value of λ from 2 to 10. Our experimental results show that the performance varies with the dataset and classification always perform better when the value of λ ranges from 4 to 7. The highest accuracy in the classification results for this data set was observed when λ was set to5. A gradient is calculated for each vertical bar of the histogram to identify whether the bar is a peak or a valley and then the gradient is compared with the default threshold range (-λ to λ) where λ is a number and it depends upon data which is set by a user or an expert. Peak and valley are defined below in Equations 4 and 5, respectively and gradient (m) is defined in Equation 7.

Peak: A vertical bar in a histogram is called a peak if the gradient



g_1	g_2	g_3
465.5	146.0	61.4
1926.2	16149.5	10998.2
34.0	1006.9	608.1
1.78	51.5	1.6

Raw Data

g_1	g_2	g_3
0.04	0.01	0.06
0.01	0.06	0.04
0.08	0.01	0.06
0.06	0.05	0.08

Normalized Data

Figure 2: Raw data transformed to [0,1] by applying normalization.

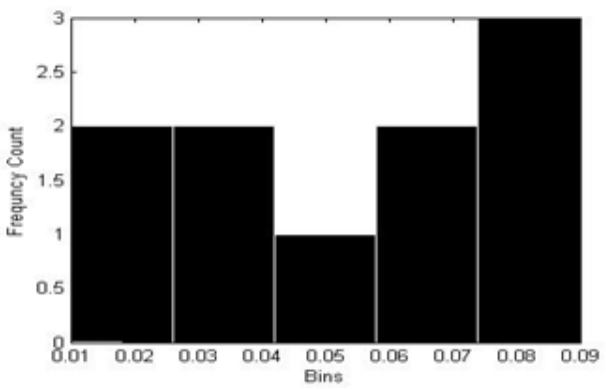


Figure 3: Gene (gi) plotted into a histogram, where i=1,2,3….total number of genes in a dataset.

('*m*') of the bar is greater than the higher limit of the threshold range ('λ') and it is defined by '*p*'.

Valley: A vertical bar in a histogram is called a valley if the gradient ('*m*') of the bar is less than the lower limit of the threshold range ('-λ') and it is defined by '*v*'.

Unassigned histogram: A vertical bar in a histogram is called an unassigned histogram if the bar of the histogram is neither a peak nor a valley and its gradient ('*m*') lies between ('-λ to λ') and it is defined by '*x*' as shown in Equation 6.

$$peak(p_i) = m_i > \lambda \quad (4)$$

$$valley(v_i) = m_i < -\lambda \quad (5)$$

$$\text{Unassigned Histogram } (x_i) = -\lambda < m_i < \lambda \quad (6)$$

$$m_i = \frac{y_i - y_j}{x_i - x_j} \quad (7)$$

Where, i is the i^{th} vertical bar of the histogram, m_i is the gradient between i^{th} vertical bar and last identified peak or valley, y_j is the frequency count of vertical bar, y_j is the frequency count of previous peak or valley is identified, x_i is the bin width of i^{th} vertical bar, and x_j is the bin width of previous peak or valley is identified.

Step 3: Finding adaptive partitions using different tuple test

After identifying peaks and valleys in a histogram, the next step is to find the adaptive partitions using four different Tuple Tests: Peak-Valley-Peak Test (PVP), Valley-Peak-Valley Test (VPV), Peak-Valley-Peak-Valley-Peak Test (PVPVP), and Valley-Peak-Valley-Peak-Valley Test (VPVPV). Each Tuple Test is independent of the other and rules are generated from each of these tests. Following the rule generation, the classification accuracy of each test is compared with the existing rule-based classifiers and other machine learning algorithms. The assumptions made for these Tuple Tests are:

1. If there are two or more consecutive valleys or peaks or any unassigned points in the histogram, then they will be combined as one valley or one peak or one unassigned point respectively. For e.g. if two consecutive valleys i.e. valley v_1 is from 0.05 to 0.1 and valley v_2 is from 0.1 to 0.15, then both valleys (v_1 *and* v_2) are combined into one valley, i.e. $v_{3=} v_{1+} v_2$ *and* v_3 range will be from 0.05-0.15 and will be called a partition range.

2. If there is no peak in the histogram, then the whole histogram will be considered as one adaptive partition and the partition range will be from 0 to 1.

Peak-Valley-peak test (PVP)

Once the distribution of the data points in bins is obtained using a histogram and the peaks and valleys are identified, the next step is to find the combinations of the Peak-Valley-Peak triplets in the histogram. Each Peak-Valley-Peak combination gives one partition. To obtain Peak-Valley-Peak combination, some assumptions have been made. The assumptions are as follows:

1. In between two peaks, (p_i *and* p_j) if there is any valley (v) then peaks p_i *and* p_j and valley (v) will be combined into one adaptive partition.

2. In between two peaks (p_i *and* p_j), if there is any unassigned data point (x) then peaks (p_i *and* p_j) and unassigned data point (x) will be combined into one adaptive partition.

3. In between two peaks (p_i *and* p_j), if there is both valley (v) and unassigned data point (x) then peaks (p_i *and* p_j), valley (v), and unassigned data point (x) will be combined into one adaptive partition.

4. If there is only one peak (p_1) in the histogram, then the partition range will be from 0 to p_1 and p_1 to 1, i.e. two adaptive partitions.

5. If there is a combination of Peak-Valley-Peak (p_i-v-p_j) triplet in the histogram, then the partition range will be from half of the partition from the left of p_i and half of the partition from the left of p_j because both the peaks (p_i *and* p_j) are being shared with other partitions as well.

Figure 4 shows the plot of the histogram, which is used to show the distribution of the data for a gene (g_i). The histogram has 10 bars where each bar is a bin of width 0.1. Several experiments were conducted to determine the width of a bin ranging from 0.01 to 1. It was observed that for our dataset a bin width of 0.1 gave the best classification accuracy. The x-axis in the histogram represents the bin number and y-axis the frequency count. According to step 2.2 we can identify peak (p), valley (v) and unassigned data points (x) in this plot. Figure 5 shows the identification of the peak, valley and unassigned data points. In Figure 5, we can observe three partitions based upon the above assumptions made in Peak-Valley-Peak Test (PVP). First partition is from bin number 1 to bin number 5 its partition range is from 0.5 to 0.45. Second partition is from bin number 5 to bin number 7 and its partition range is from 0.45 to 0.65 and third partition is from bin number 7 to bin number 10 and its partition range is from 0.65 to 1. Similarly other Tuple Tests can be performed by using same assumptions for Valley-Peak-Valley (VPV) Tuple Test, Peak-Valley-Peak-Valley-Peak (PVPVP) Tuple Test and Valley-Peak-Valley-Peak-Valley (VPVPV) Tuple Test.

Phase 2: Rule generation

The adaptive partitions obtained in Phase 1 will support in generating rules, which eventually assist in the classification process. Phase 2 has been divided into four steps. Step 1 is assigning membership values to data points based on their presence or absence in the partition. Step 2 is to find classes for the partitions and generate rules. Step 3 is to assign classes to the empty partitions based on the neighboring partitions, and Step 4 is to represent rules. Below is the explanation of these four steps:

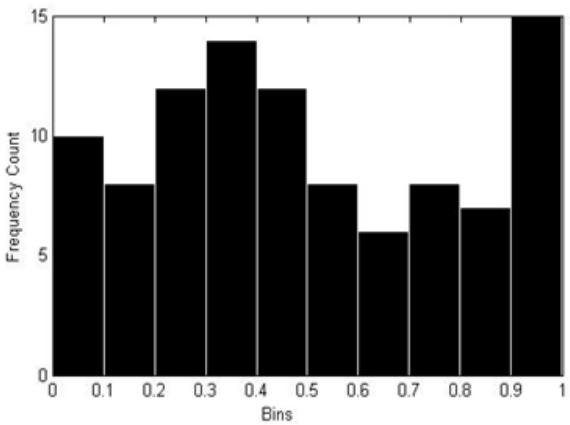


Figure 4: Plot of the histogram data for a gene g_i showing10 bins.

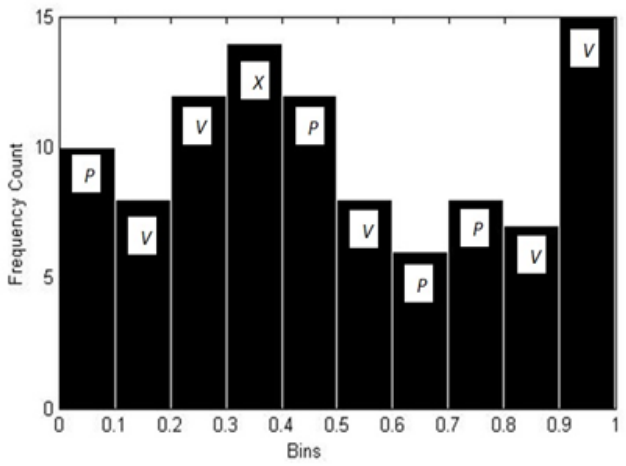


Figure 5: Identification of peak (*p*), valley (*v*) and unassigned point (x) on a histogram.

Step 1 Assigning absolute membership values to data points

Absolute membership, i.e., 0 or 1 is employed to the data point where '0' means the data point is absent in the partition, and '1' means the data point is present in the partition. It is denoted by μ and it is defined in Equation 8. For e.g., let us assume there are two partitions P_1 and P_2 in a gene. Partition P_1 is 0 to 0.5 and P_2 is 0.5 to 1. If a data point has a value of 0.8, then 0 will be assigned to P_1 and 1 will be assigned to P_2.

Let us assume that gene 'g_i' is divided into K partitions $\{A_1^k, A_2^k, ..., A_k^k\}$ where A_1^k the i^{th} partition of gene is 'g_i' *and* '*k*' indicates the total number of partitions in gene 'g_i'. We have defined the membership function as follows;

$$\mu_i^k(x) = \{1, p_1 \le p_2, else, and \mathbf{0} \quad (8)$$

Where, μ_i^k is an absolute membership value of data point '*x*' in i^{th} partition of a gene and P_1 and P_2 are lowest point and highest point respectively of a partition range.

Step 2 Finding classes and generating rules

Rule formation has two parts, consequent part and antecedent part. Consequent part is the left side of the rule, which defines partition ranges of all the genes while antecedent part is the right side of the rule, which defines classes. Ishibuchi et al. [11] has explained how to assign classes to the generated rules. The classes can be determined by the following procedure. Let us assume that there are *m* samples $x_p = (x_{p1}, x_{p2.....}x_{pm})$ where p=1, 2,….*m* are training samples from M classes: C_1, C_2, …..C_M and μ is an absolute membership value of a data point in a partition. The consequent part of a rule can be obtained by the following procedure:

Rules are generated by finding the dependency between the partitions of different genes. A partition can have samples of different classes however each partition has one rule; therefore, we can assign only one class to a rule. Equation 9 finds the weightage of each class in a partition by finding an absolute membership value of each data point, which is denoted by β_{CT} where CT= {classC1, classC2,.....classCM. Equation 10 shows the class, which has the maximum weightage or domination in a partition; hence, the class with a maximum weightage will be assigned to the rule and is denoted by β_{CX}.

Step1: Calculate β_{CT} for *T*=1,2,….*M* as

$$\beta_{CT} = \sum_{p \in CT} \mu_i^k(x_{p1}) . \mu_j^k(x_{p2}) \qquad G_{gi}^k \quad (9)$$

Step 2: Find class *X*(*CX*) by

$$\beta_{CX} = \max\{\beta_{C1}, \beta_{C2}, ..., \beta_{CM}\} \quad (10)$$

By this procedure the consequent part or class is determined for the rule. Therefore, if a new sample matches any of the antecedent part of the rule, then its corresponding class will be assigned to a new sample. Moreover, there can be empty partitions because of an absence of data points in it. To assign classes to the empty partitions, the nearest neighborhood approach has been employed which is explained in step 3.

Step 3 Empty partitions

In some cases, partitions can be empty. Classes are assigned to the empty partitions based on neighboring partitions. If a test sample belongs to empty partition, then the class is assigned to the sample depending on the neighboring classes around the empty partition. Table 2 shows the example for empty partitions, P_5 and P_{12}. For example, if a test sample belongs to empty partition P_5, then it is assigned to class '*C1*' since '*C1*' dominates around empty partition P_5.

Step 4 Rule representations

Let R_i^n be the label of the rules and $G_{gi}^{k\square}$ be represent the partition of the genes, where *n*=number of rules, *i*=1, 2,…,*n*, G represents the gene expression dataset, g is the i^{th} gene in the dataset, *k* are the total number of partitions for gene 'g_i'. Below are examples of a rule:

***Rule R*$_1$:** g_1(0-0.5) ∧ g_2(0-0.4) ∧ g_3(0-0.3) ∧ g_4(0-0.2) → C1,

***Rule R*$_2$:** g_1(0-0.5) ∧ g_2(0.4-0.7) ∧ g_3(0-0.3) ∧ g_4(0.2-0.5) → C2,

***Rule R*$_3$:** g_1(0.5-0.1) ∧ g_2(0.4-0.7) ∧ g_3(0.7-1) ∧ g_4(0.2-0.5) → C3.

Rules, 1, 2 and 3 are the representative rules and the values in the brackets represent the partitioning ranges. For e.g. g_1 (0-0.5) implies g_1 is a gene and its one of the partition has a range from 0 to 0.4. The part of the rule before the arrow is called antecedents of the rule, whereas *'C1', 'C2'* and *'C3'*denotes the class name called the consequent of the rule. Figure 6 shows the pseudocode for generating rules. The input to the algorithm is number of partitions (*n*), dataset (*y*), and partition points (*pp*), and output are rules (*ci*). To generate rules, first assign absolute membership 0 and 1 to the data, which is explained from Lines 01 to 07; then find all possible combinations of the partitions, which are explained from Lines 08 to 12. Finally, classes are assigned to the partitions or the consequent of the rule that is explained from Lines 13 to 27.

Phase 3: Classification and validating experiments

We performed several experiments to validate the results. The results were evaluated and compared by using statistical measures, such as *10*-fold cross validation, sensitivity, specificity, and F-measure [12-14] and the summary of the comparison results have been shown in Table 3.

C1	C1	C2	C3
C1	Empty partition (P_5)	C1	C2
C2	C3	C2	Empty partition (P12)

Table 2: Assigning classes to empty partitions.

```
Algorithm: Generating Rules
Input: y, p, np, pp
Output: ci //rules
01 For d1 = 1 to y(no.  of columns)
02 For d2 = 1 to y(no.  of rows)
03 For i = 1:1:p
04 Finding membership values for the data
05 End For
06 End For
07 End For
08 For i = 1 to d
09 For j = 1 to p(i)
10 t(i,j) = Generate all possible combinations (i,j)
11 End For
12 End For
13 if (t(j,i)>P(i)
14 t(j,:) = 0
15 End if
16 For j = 1 to length(y)
17   ym = ym*y(:,j);
18  End For
//Assigning class to the consequent part of the rule
19 For j1 = 1:1:no.of classes
20 if (y(:,n) == j1)
21   mt= mt*mem(i1,k2,k1(k2))
22   kk = mt*ymul(i1,1);
23  cl(j1) = cl(j1)+ kk;
24   End if
25 End For
26 [C,Ind] = max(cl)
27 ci(i)=Ind
```

Figure 6: Pseudo code for Generating Rules.

Results

In this section we describe the experiments that have been performed to evaluate the accuracy and efficiency of the data-adaptive rule-based classification system. Results obtained through these experiments have been compared to rule-based and non-rule-based classification methods. Obtained results have been validated by using statistical measures such as K-fold cross validation, sensitivity, specificity, F-measure, precision, etc. A the algorithms have been executed on Intel® Core™ i7 CPU 930@ 2.8 GHz 2.79 GHz with 12 GB RAM using MATLAB software.

Dataset

In this paper we used hippocampal gene expression data set provided by Blalock et al. [15]. This dataset is readily available from Gene Expression Omnibus (GEO) repository on NCBI website. This dataset originally consists of 22,283 genes and 31 samples. In order to avoid false negatives and false positives, the dataset was pre-processed by removing the genes that were associated with absent or missing tag. The data was further pre-processed by only considering the genes that had the p-value <=.05. This resulted in a reduced dataset consisting of 4961 genes. The distribution of expression values of the genes across different samples was normalized using z-scores standardization.

In gradient-based partitioning approach, different threshold range of λ (2 to 10) and bin width range (0.01 to 1) have been set to evaluate the performance of four Tuple Tests such as Peak-Valley-Peak (PVP) Tuple Test, Valley-Peak-Valley (VPV) Tuple Test, Peak-Valley-Peak-Valley-Peak (PVPVP) Tuple Test, and Valley-Peak-Valley-Peak-Valley (VPVPV) Tuple Test. Table 3 compares the performance of the proposed data- adaptive rule-based classification method, rule-based classification method and non-rule-based classifiers. Table 3 shows that proposed method out performs all the rule-based classifiers and non-rule-based classifiers except Decision Table and JRIP, which have competitive accuracy with the proposed method. The proposed method performs better because the partitions increased the separation between the classes in Alzheimer's data due to the adaptive nature of the partitions. Further, these partitions are neither too fine nor coarse that generated efficient rules to classify the future sample into their respective classes. We have compared the results by taking different subset of the genes. The gene subset selection has been done by using Chi-Squared feature selection that form the reduced feature set. We have used four different subset of genes i.e., top 100 genes, top 150 genes, top 200 genes and top 250 genes to compare and analyze the results. Proposed gradient-based partitioning has a best classification accuracy of 74% for all the subset of genes i.e., 100, 150, 200 and 250 genes. The proposed methodology generates same accuracy of 74% for all the subset of genes as there are few samples in the dataset (31 samples), that leads to same number of partitions and partition ranges for all the subsets. Based upon the results of Table 3, proposed methodology outperforms all the non-rule-based classifiers because rules promote understanding and provide insightful information of the relationship and patterns in the gene expression dataset. Proposed partitioning rule-based classifier has either superior or comparable accuracy as compared to other rule-based classifiers such as Decision Table, JRIP, PART and NNGE.

Conclusions

A significant percentage of rule-based classification algorithms offer diminished performance when encountered by a large number of rules due to inefficient partitions and resulting in ineffective classification system. To generate efficient rules, partitioning of the data is important as the number of rules depends on the number of partitions. As the number of partitions increases, the number of rules grows exponentially; this growth can result in a number of insignificant or irrelevant rules which can hamper the classification system.

This works presents a data-adaptive partitioning approach in which each gene is partitioned independently, and these partitions assist in finding rules for classification purposes. Using different statistical

Non-rule-based Classifiers				
	Top 100 Genes	**Top 150 Genes**	**Top 200 Genes**	**Top 250 Genes**
Naïve Bayes	67%	61%	58%	58%
Logistics	54%	64%	64%	67%
Multi-Layer Perceptron	64%	64%	61%	64%
RBF Network	64%	61%	67%	58%
Simple Logistic	67%	70%	70%	70%
SMO	61%	67%	67%	64%
Random Tree	71%	71%	70%	69%
Rule-based Classifiers				
	Top 100 Genes	**Top 150 Genes**	**Top 200 Genes**	**Top 250 Genes**
Decision Table	73%	74%	70%	74%
JRIP	74%	68%	77%	74%
PART	73%	73%	73%	73%
NNGE	68%	67%	70%	70%
Data-adaptive partition classifiers				
	Top 100 Genes	**Top 150 Genes**	**Top 200 Genes**	**Top 250 Genes**
Gradient-based partitioning	***74%***	***74%***	***74%***	***74%***

Table 3: Pre-processed Dataset (After Feature Selection, 31 samples, and 2 classes (Normal and Disease).

measures such as TP rate, FP rate, Precision and F-measure, has validated results obtained. Comparison has been done for the overall accuracy of different classification methods such as the proposed data-adaptive partitioning rule-based classification system, rule-based classification (RBC), and non-rule-based classification system by employing Blalock dataset. Results demonstrate that partitions generated from the histogram by using different gradient-based partitioning give adaptive partitions which assist in generating efficient rules, because these adaptive partitions increases the separation between the classes and makes an effective classification system. Moreover, the obtained partitions are efficient because comparing every peak and valley with the threshold range in order to remove outlier or bad partitions validates them. The generated rules are easy to interpret and more accurate for gene expression analysis than other methods and gives concise and biologically meaningful rules. The proposed algorithm is computationally inexpensive and its space and run time costs are only polynomial. Moreover it's scalable to large datasets on which other rule mining classifiers are computationally challenged.

Based on the outcome of classification accuracies, the methods studied can be of enormous use by producing optimally actualized data, which can be applied in the medical decision-making. The automated machine learning methods can produce more reliable classifications that can aid medical professionals in the early diagnosis and treatment of AD.

Acknowledgements

This project was supported by grants from the National Center for Research Resources (NCRR) P20RR016456), the National Institute of General Medical Sciences (NIGMS) (P20GM103424) from the National Institutes of Health (NIH) and LEQSF (2011-14)-RD-A-16. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

1. Blennow K, DeLeon M, Zetterberg H (2006) Alzheimer's disease. Lancet 368: 387-403.
2. Solomon PR, Murphy CA (2005) Should we screen for Alzheimer's disease? A review of the evidence for and against screening Alzheimer's disease in primary care practice. Geriatrics 60: 26-31.
3. Cuingnet R, Gerardin E, Tessieras J, Auzias G, Lehericy S, et al. (2011) Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database. Neuroimage 56: 766-781.
4. Lerch JP, Worsley K, Shaw WP, Greenstein DK, Lenroot RK, et al. (2006) Mapping anatomical correlations across cerebral cortex (MACACC) using cortical thickness from MRI. Neuroimage 31: 993-1003.
5. Joshi S, Shenoy D, VibhudendraSimha GG, Rrashmi PL, Venugopal KR, et al. (2010) Classification of Alzheimer's disease and Parkinson's Disease by using machine learning and neural network methods. ICMLC 2010 Second International Conference.
6. Lee C, Lam CP, Masek M (2011) Rough-Fuzzy hybrid approach for identification of bio-markers and classification on Alzheimer's disease data. BIBE '11 Proceedings of the 2011 IEEE 11th International Conference on Bioinformatics and Bioengineering.
7. Lopez M, Ramirez J, Salas-Gonzalez D, Alvarez I, Segovia F, et al. (2009) Neuro image classification for the Alzheimer's Disease Diagnosis using Kernal PCA and Support Vector Machines. Nuclear Science Symposium Conference Record (NSS/MIC).
8. Kloppel S, Stonnington CM, Chu C, Draganski B, Scahill RI, et al. (2008) Automatic classification of MR scans in Alzheimer's disease. Brain 131: 681-689.
9. Dua S, Du X (2011) Data mining and machine learning in cyber security. (1st edn), Auerbach Publications, Florida.
10. Han J, Kamber M (2006) Data mining: Concepts and techniques. (2nd edn), Morgan and Kaufmann Publishers, San Francisco.
11. Ishibuchi H, Nozaki K, Yamamoto N, Tanaka H (1995) Selecting fuzzy if-then rules for classification problems using genetic algorithms. IEEE transactions on fuzzy systems 3: 260-270.
12. Bezdek JC, Chauh SK, Leep D (1986) Generalized k- nearest neighbor rules. Fuzzy sets and Systems 18: 237-256.
13. Davis DL, Bouldin DW (1979) A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1: 224-227.
14. Yidong Li, Hong Shen (2009) Equi-Width Data Swapping for Private Data Publication. International Conference on Parallel and Distributed Computing, Applications and Technologies 231-238
15. Blalock EM, Geddes JW, Chen KC, Porter NM, Markesbery WR, et al. (2004) Incipient Alzheimer's disease: Microarray correlation analyses reveal major transcriptional and tumor suppressor responses. Proc Natl Acad Sci USA 101: 2173-2178.