ÍI. Executive Summary

II. Problem Description

It’s very difficult to make a judgment that a classification algorithm is better than another because it may work well in a certain data environment, but worse in others. Evaluation on performance of a classification algorithm is usually on its accuracy. However, other factors, such as computational time or space are also considered to have a full picture of each algorithm. This project will evaluate performance of various classification techniques on three different data environment, which are noisy, un-noisy, and computational-intensive.

Classification technique can be classified into five categories, which are based on different mathematical concepts. These categories are statistical-based, distance-based, decision tree-based, neural network-based, and rule-based. Each category consists of several algorithms, but the most popular from each category will be used and compared on 3 different classification tasks. The algorithms are C4.5, Naïve Bayes, K-Nearest Neighbors, and Backpropagation Neural Network.

There are three data environments that are being studied in this project. One of them is the Iris dataset, which consists of relatively clear distinction of the three Iris classes. The second dataset is Diabetes, which has a very noisy data environment or unclear distinction of the classes. The last data environment is Spambase, which is a real-world dataset used by many company to classify an email as either Spam or Not Spam. This dataset is used to compare the practical usage of each algorithms being studied.

III. Analysis Technique

Classification is one of several major tasks in data mining. It is a predictive process where prediction about values of data is made using known results found from different data. In a typical classification task, there are two main steps: generating a data model, and predicting data values using that model. A formal description of a classification task is as below:

> *Given a training data* $\{(t_1, c), (t_2, c), \ldots, (t_n, c)\}$*, a classifier* $h: T \rightarrow C$ *is produced and used to map an instance* $t \in T$ *to its classification label* $c \in C$.

Although the second step actually does the classification, most research has been on step 1, where a data model is being generated using a mathematical-base algorithm. Decision tree and Naïve Bayes are examples of such. Classification algorithms can be categorized into 5 groups: statistical-based, distance-based, decision tree-based, neural network-based, and rule-based. A brief description of each category is below:

> ***Statistical-based algorithms*** create statistical descriptions about data to make predictions. Statistical descriptions include mean, standard deviation, regression, probability, etc…
>
> ***Distance-based algorithms*** consider the data model as a vector space and map the input data to the class of the data instance in the model that is “similar” to it. An example of “similarity” is the Euclidian distance between the input vector and a vector in the population space.

Decision tree-based algorithms create a classification tree to model the classification process. The process starts at the root of the tree, goes down to the appropriate branch according to the data's value, and stops at the leaf of three, where its class can be determined.

Neural network-based algorithms model the classification network similar to that of a human brain. An input data is run through the network, and output(s) is produced that can determined the input's class.

Rules-based algorithms typically derive from a decision tree or a neural network. The idea is to interpret the graphical model into rules that can be used more easily to classify data. In this project, rule-based approach is considered to be as a decision-tree approach, thus their performance is similar.

Each of these five categories consists of many algorithms. However, only four of the most popular and useful algorithms are being studied. They are Naïve Bayes, K-Nearest Neighbors, C4.5, and Backpropagation Neural Network. These algorithms and their concepts are discussed next.

1. Algorithms

a) *Naïve Bayes Classifier:*

The idea behind Naïve Bayes algorithm is the posterior probability of a data instance t_i in a class c_j of the data model. The posterior probability $P(t_i \mid c_j)$ is the possibility of that t_i can be labeled c_j. $P(t_i \mid c_j)$ can be calculated by multiply all probabilities of all attributes of the data instance in the data model:

$$P(t_i \mid c_j) = \prod_{k=1}^{p} P(x_{ik} \mid c_j)$$

with p denoted as the number of attributes in each data instance. The posterior probability is calculated for all classes, and the class with the highest probability will be the instance's label.

As we can see in the calculation formula of posterior probability, the algorithm is designed for categorical data. This is one disadvantage in Naïve Bayes. However, there is a way to overcome this problem where continuous data is divided into ranges. Then the probability of a value is considered the range's probability. The program used in this project divides continuous data into range with a predefined number of values. For example, in the Iris dataset, each attribute is divided into 3 ranges: the first range with lower bound and upper bound consists of the first 41 instances of Iris, the second range with lower bound and upper bound consists of the next 40 instances of Iris, and the third range with lower bound and upper bound consists of the last 39 instances of Iris.

Naïve Bayes is chosen as a representative of statistical-based category because it works very well in a non-linear problem domain. A non-linear problem occurs when the model's classes can not be divided linearly, just as demonstrated in Figure 1.

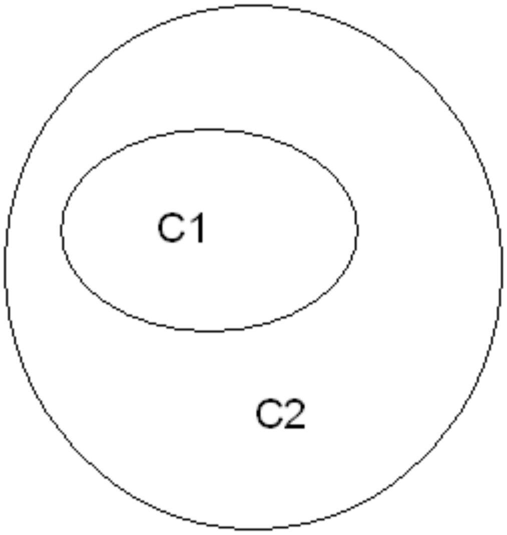

Figure 1: *Nonlinear data space*

b) ***K-Nearest Neighbors***

How K-Nearest Neighbors works is very simple: *finding the K nearest neighbors to an input instance in the population space and assign the instance to the class the majority of these neighbors belong to*. The "nearest" measurement refers to the Euclidean distance between two instances. For example, the Euclidean distance between t_i and t_j is

$$D(t_i, t_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$

with p denoted as the number of attributes in each data instance. Since the class assignment is based on the majority, K must be odd to avoid situation when there is an equal number of each class. Besides, the value of K must be greater than the number of classes in the problem.

c) ***C4.5***

C4.5 is the most popular and the most efficient algorithm in Decision tree-based approach. A Decision tree algorithm creates a tree model by using values of only one attribute at a time. At first, the algorithm sorts the dataset on the attribute's value. Then it look for regions in the dataset that clearly contain only one class and mark those regions as leaves. For the remaining regions that have more than one classes, the algorithm choose another attribute and continue the branching process with only the number of instances in those regions until it produces all leaves or there is no attribute that can be used to produce one or more leaves in the conflicted regions.

Some effective decision tree algorithms choose which attribute to make the partitions by calculating the information gain for each attribute. The gain is calculated by subtracting the entropy of the attribute from the entire dataset entropy. The entropy measure the amount of disorder in a set of values. One disadvantage of calculating an attribute's entropy is that it relates to categorical data. However, just like in Naïve Bayes, this can be overcome by dividing continuous data into ranges. The entropy of an attribute is then determined by firstly calculating the entropy in each range

$$H(\text{range, attribute}) = \sum p(\text{class}_j \mid \text{range}) \text{ x } \log [p(\text{class}_j \mid \text{range})]$$

(Where $p(\text{class}_i \mid \text{range})$ is the probability of each class appearing in the calculated range)

The entropy of the entire attribute is determined as following:

$$\mathrm{H(attribute)} = \sum \mathrm{p(range_i)} \times \mathrm{H(range_i, attribute)}$$

(p(range$_i$) is the probability of an attribute's range in the entire dataset)

The information gain of that attribute is then:

$$\mathrm{Gain(attribute, dataset) = H(dataset) - H(attribute)}$$

$$\text{with } \mathrm{H(dataset)} = \sum \mathrm{p(class_j \mid dataset)} \times \log [\mathrm{p(class_j \mid dataset)}]$$

(p(class$_j$) is the probability of class j in the entire dataset)

C4.5 makes a step further over other algorithms by taking into account the cardinality of each range to avoid overfitting of training data. Overfitting happens when the data model adapts so well to the training dataset so that the test set suffers high error rate. C4.5 uses the GainRatio instead of information Gain, as below:

$$\text{GainRatio(attribute, dataset)} = \frac{Gain(attribute, dataset)}{H\left(p(range_1),..., p(range_n)\right)}$$

$$\text{with}\, H(p(range_1),..., p(range_n)) = \sum_{i=1}^{n} p(range_i)\log(\frac{1}{p(range_i)})$$

Another advantage of C4.5 is tree pruning. There are two pruning strategies:

- Replace a subtree by a leaf node if the replacement results in an error rate close to that of the original tree
- Replace the subtree by its most used subtree within a certain increase in error rate.

d) *Backpropagation Neural Network:*

Neural networks are created from the idea of the human brain. A typical neural network consists of, though much smaller, a number of processing units called *nodes*. A node is connected with other nodes via *links*, each of which is associated with a *weight*. Nodes are arranged into layers (one input layer, one or more hidden layers, and one output layer). The input layer typically has the same number of nodes as the number of attribute each data instance and the output layer has the same number of nodes as the number of classes.

Each node consists of 2 elements: an adder and an activation function. The adder calculate the net input to the node

$$\mathrm{net_k = x_1w_{k1} + x_2w_{k2} + \ldots + x_nw_{kn}} = \sum_{i=1}^{i=m} x_i w_{ki}$$

and the activation function produces the node output

$$\mathrm{y_k = f(net_{k)}}$$

There are many forms of activation function. The activation used in this project it the Sigmoid Function:

$$f(x) = \frac{1}{1 + e^{-cx}}$$

The neural network used in this project is a feed-forward network, meaning that it accepts the inputs at the input layer, forwards them to all other later layers, and finally produces outputs at the output layer. Then the node with the highest value will be the instance's class.

Just like a human neural network, a neural network learns from experience or through training. The back-propagation network used in this project suggest that after the network produces an output in the training process, the learning is running backward by adjusting the weights for all links from outputs layers back to the input layers sequentially. The new weight of each links is calculated as below:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t)$$

with

$w_{ji}(t)$, $w_{ji}(t+1)$: the weights from node i to node j at time step t and time t + 1.

$\Delta w_{ji}(t)$: the weight adjustment from time step t to time t + 1.

The weight adjusting uses the *delta rule* or Widrow-Hoff rule. The rule suggests that the adjustment $\Delta w_{ji}(t)$ is defined by:

$$\Delta w_{ji}(t) = \eta \,.\, e_j(t) \,.\, f'(\text{net}).\, x_i$$ (Kantardzic, 2003)

at the output nodes with:

η: the learning rate (determined experimentally)

$e_j(t) = d_j(t) - y_j(t)$

$d_j(t)$: the desired output of node k at step t

$y_j(t)$: the actual output of node k at step t

f': the derivative of the activation function

x_i : the input from node i in the previous layer.

Let $\delta_j(t) = e_j(t) \,.\, f'(\text{net})$, then $\Delta w_{ji}(t)$ can be written as:

$$\Delta w_{ji}(t) = \eta \,.\, \delta_j(t) \,.\, x_i$$ (Kantardzic, 2003)

At the hidden layers, the weights are adjusted by substituting $\delta_j(t)$ with:

$$\delta_j(t) = f'(\text{net}) \,.\, \sum \delta_k(t) \,.\, w_{kj}(t),\ k \in D$$ (Kantardzic, 2003)

(D denotes the set of all nodes in the next layer)

2. Performance evaluation:

Evaluating a classification technique is a fuzzy problem. However, in this project, the algorithms' performance is measured by the accuracy of the classification. This is considered in three different data environment. In addition, the computational overhead of each algorithm is also evaluated.

3. Dataset:

There are three datasets in this project:

The first dataset is the Iris dataset. The training set includes 120 instances, each of which has four attributes (see appendix) and can be classified as either Virginica, or Setosa or Versicolor. The test set includes 30 instances of Iris. The training set contains relatively clear classification of Iris. In order word, there is a very high correlation between each attributes and the classification.

	Sepal Length	**Sepal Width**	**Petal Length**	**Petal Width**
Correlation	0.793135	-0.41549	0.951772	0.964047

The second dataset is the Diabetes dataset. The training set includes 291 people, each of whom has eight attributes (see appendix) and can be classified as either Healthy or Sick. The test set includes information about 45 people. This dataset contains very noisy data. The correlation between each attribute and the classification is very low.

	Pregnancies	**PG Concentration**	**Diastolic BP**	**Tri Fold Thick**	**Serum Ins**	**BMI**	**DP Function**	**Age**
Correlation	-0.26	-0.53	-0.24	-0.29	-0.33	-0.294	-0.18	-0.38

The last dataset contains information about email that can be classified as Spam or not Spam. This experiment models a real world situation to test not only the accuracy of the algorithm, but also their computational overheads. The training set contains 4101 instances, each of which contains 57 attributes and can be classified as Spam or Not Spam. The test set contains 500 instances of email. This dataset is considered very noisy since the highest absolute value of correlation of between the attribute and the class is 0.39.

Because we consider each attribute in each dataset contribute the same usefulness in the classification, these datasets have to be normalized into the same range of values. For the Iris and Diabetes dataset, all attributes are scaled down to a range between 0 and 1 by. In the Spam dataset, all attributes' values are scaled down to the range between 0 and 100 since their values would be so small if they are in the range between 0 and 1 and the C4.5 program in this project only accept values in floating point number that has one digit after the decimal point.

In addition, since some of the four programs only consider numeric data, the classes of each instance are changed to the numeric values accordingly. For example, in the diabetes dataset, Healthy = 1 and Sick = 0.

4. Procedure

The four programs for four algorithms are used on each dataset. In the testing phase, program outputs are compared to the actual ones. The number of wrong classifications is calculated to determine the accuracy rate. Also, the training and testing phase time of each dataset are measured to compare the computational overhead.

IV. Assumptions

V. Results

The results from the testing phase are below

ACCURACY	*Naïve Bayes*	*K-Nearest Neighbor*	*C4.5*	*Backpropagation Network*
Iris	93.3%	93.3%	86.7%	87.0%
Diabetes	84.4%	77.8%	73.3%	73.3%
Spam	81.6%	89.8%	91.4%	92.6%

TRAINING	*Naïve Bayes*	*K-Nearest Neighbor*	*C4.5*	*Backpropagation Network*
Spam	Very fast	Very fast	Very fast	6 hours

TESTING	*Naïve*	*K-Nearest*	*C4.5*	*Backpropagation*

	Bayes	*Neighbor*		*Network*
All	Very fast	Very fast	Very fast	Very fast

a) *Accuracy*

In the un-noisy environment, Naïve Bayes and K-Nearest Neighbor produce a very high accuracy of 93.3%. Neural network produces the least accuracy and this is not unexpected.

In the noisy environment, there is an interesting result. The accuracy in noisy environment with small number of training data is decreasing from the left to right in the table, but with very large number of training data, the accuracy is then increasing from the left to the right. This suggests that measurement of accuracy of a classification algorithm is really tough, and it depends on the characteristics of the dataset.

b) *Training and Testing time*

Neural Network has a very high computational overhead in the training phase. When being timed on the Spam training set, it takes 6 hours to create a data model for the classification. If a dataset contains millions of training data with many attributes, and the number of training loops on the set is high, the network will take a very long time on a typical computer to arrive with a model. The other algorithms run very fast not only in any data environment. However, the more data and the longer time a neural network is train with, the better result it will produce.

VI. Issues

The major issue in this project is to determine what performance factor to take into account to evaluate the algorithms. Since this is a general comparison, accuracy and time overhead are two good factors to consider. For specific comparison, performance factors have to be considered carefully.

VII. Appendix