# Project Proposal:

# Naïve Bayes, KNN, and Neural Networks

Team: LicSel

Gerardo Licon, Luis Selvera

Department of Computer Science, The University of Texas at San Antonio

CS-5473: Data Mining

Dr. Zhang

October 11, 2020

## Introduction

Our 2-member team will engage in an in-depth exploration of the theory, techniques, and implementation of the data mining algorithms known as Naïve Bayes', K-Nearest-Neighbor (KNN), and Neural Networks. Along with an implementation of each algorithm, our work will also include the validation of our proposed implementations, as well as a comparison of our implementation with similar algorithms from the Python SciKit Learn package. Finally, our team will put together a simple user interface which will allow a user to run each of our algorithm implementations and specify the data that is to be used.

Our team aims to accomplish our objectives by completing tasks in modules, each algorithm being its own dedicated module, and the final module will be the programming of a user interface. The team intends to maintain and improve the effectiveness of our implementations with the use of periodic performance evaluations of the proposed algorithm implementations. Our team expects to develop implementations that rival other popularly used implementations when compared in terms of a number of metrics used to evaluate the quality of data mining algorithm implementations.

## Algorithms

### Naïve Bayes

Bayesian classifiers are statistical classifiers that can predict the probability that a given tuple belongs to a particular class. Naïve Bayesian classification is based on Bayes' theorem of posterior probability. It assumes that the effect of an attribute value on a given class is independent of the values of the other attributes.

**Pseudocode**

Input: Class-labeled training set D

Output: The class of testing dataset T

Steps:

1. Read training set $D$, with tuple $X = (x_1, x_2, \dots, x_n)$, $n$ attributes $A_1, A_2, \dots A_k$, and $m$ classes $C_1, C_2, \dots C_m$.
2. For each class $c_i$, attribute $A_k$, and value $x$ in the attribute, estimate distribution of $P(A_k = x|C = c_i)$
   a. If $A_k$ categorical:
   $$P(A_k = x|C = c_i) = \frac{\#tuples\ in\ c_1 with\ A_k = x}{\#tuples\ in\ c_i}$$
   b. If $A$ is a continuous-valued, assume Gaussian distribution
   $$P(A_k = x|C = c_i) = g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
3. For each class $c_i$, compute
   $$P(C = c_i) \prod_{k=1}^{n} P(A_k = x_k|C = c_i)$$
4. Assign an unseen testing set T to the class that has the highest probability
   $$class(T) = argmax\{P(C = c_i) \prod_{k=1}^{n} P(A_k = x_k|C = c_i)\}$$

## Nearest Neighbors

The k-nearest-neighbor (KNN) clustering algorithm is an algorithm that uses a minimum distance to measure the distance between two clusters. In a nearest-neighbor algorithm, an object is assigned to the class most common among its nearest neighbors. It searches the pattern space for k training tuples closest to the object.

**Pseudocode**

Input: K: number of clusters, D: a class-labeled dataset containing n objects

Output: The class of testing dataset T

Steps:

1. Calculate distances between points using Euclidean
2. Arrange calculated $n$ Euclidean distances in non-decreasing order
3. Let $k$ be a positive integer, take first $k$ distance from sorted list
4. Find $k$-points corresponding to these $k$-distance
5. Let $k_i$ denotes the number of points belonging to the $ith$ class among k points
6. Assign *x* in class I if $k_i > k_j \forall i \neq j$

## Neural Networks

A neural network is a set of connected input/output units in which each connection has a weight associated with it. The network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples. Advantages of neural networks include their high tolerance of noisy data as well as their ability to classify patterns on which they have not been trained. They can be used when there is little knowledge of the relationships between attributes and classes. We will use the most popular neural network algorithm backpropagation.

**Pseudocode**

Input: $D$, a data set consisting of the training tuples and their associated target values;
$l$, the learning rate; $network$, a multilayer feed-forward network
Output: A trained neural network
Steps:

1. Initialize all weights and biases innetwork;
2. While terminating condition is not satisfied{
   a) For each training tuple $X$ in $D${
      // Propagate the inputs forward:
      i) For each input layer unit $j${
         (1) $O_j = I_j$; // output of an input unit is its actual input value
      ii) For each hidden or output layer unit j{
         (1) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit $j$ with respect to the previous layer, $i$
         (2) $O_j = \frac{1}{1+e^{-Ij}}$;}// compute the output of each unit $j$
      // Backpropagate the errors:
      iii) For each unit $j$ in the output layer
         (1) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
      iv) For each unit $j$ in the hidden layers, from the last to the first hidden layer
         (1) $Err_j = O_j\left(1 - O_j\right) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, $k$
      v) For each weight $w_{ij}$ in network{
         (1) $\Delta w_{ij} = (l) ErrjOi$; // weight increment
         (2) $w_{ij} = w_{ij} + \Delta w_{ij}$;}// weight update
      vi) For each bias $\theta_j$ in network{
         (1) $\Delta \theta_j = (l) Err_j$; // bias increment
         (2) $\theta_j = \theta_j + \Delta \theta_j$;}// bias update
   }}

## Methods

The project will consist of individual development of each algorithm, followed by performance evaluation, and final development of an interface. We will develop the Naïve Bayes, KNN, and Neural Networks algorithms using only python basic libraries. We will execute our algorithms with the following datasets: Iris Flower Species Dataset, Fruits with colors dataset, and Mall Customers, which can easily be found online. To compare our algorithms with those from SciKit learn, we will evaluate them by obtaining accuracy, error rate, precision, F, sensitivity, and specificity for each algorithm with each dataset. We will then report this results into our final project report. To ease the use of our program, we will develop an interface to allow the user to choose an algorithm and dataset to train and test our program.

## Tools

For the development of this project, we will be using python 3.7 programming language. Since we aim to develop our own implementation of the algorithms, we will use only basic Python libraries. However, we will also use SciKit Learn libraries for comparison purposes only at evaluation stage. We will use Anaconda 3's Jupyter Lab and Spyder for developing our algorithms due to being easier to test and debug by executing individual cells. For final testing and interface development, we will use Visual Studio (VS) Code to test our programs as python scripts by using command-line. To maintain collaboration of the project, we will use GitLab and GitHub.

## Information to be Derived/Deliverables

The first of the items to be delivered during the course of this project is a progress report to be submitted in four weeks which will give a description of project updates and issues that may arise at the time. The final submission will include our completed application, as well as a

written report that will not only provide a detailed account of our methods, but also provide instruction on how to carry out the execution of our program.

## Timeline & Milestones

| Date | Milestones |
|---|---|
| October 11, 2020 | Submit proposal |
| October 18, 2020 | Implement naïve bayes classifier |
| October 25, 2020 | Implement nearest neighbor |
| November 1, 2020 | Submit progress report |
| November 8, 2020 | Implement neural networks |
| November 15, 2020 | Performance evaluation |
| November 20, 2020 | Finalize User interface |
| November 22, 2020 | Performance evaluation |
| November 29, 2020 | Submit project final report and presentation |