# EEE 511 - Assignment 2
# Pattern Classification Using $k$-NN and $k$-Means

Xiao Huang, Suhas Lohit, Rakshit Raghavan
1206424709, 1206451918, 1207751060

***Abstract*—In this assignment, we implement two simple non-linear classifiers and test their classification performance on two 2D synthetic datasets - a non-separable dataset and a non-linearly separable dataset. We implement the $k$-Nearest Neighbor and the $k$-Means algorithms. Even though $k$-Means is a clustering algorithm used in unsupervised learning, we demonstrate how it can be used for classification. In each case, we compute the confusion matrix and compare their performance with each other. Through this, we gain the understanding of theory underlying each of these algorithms, their advantages and shortcomings and thus, their scope in terms of applying them in real world problems.**

***Index Terms*—$k$-Nearest Neighbor, $k$-Means.**

## I. Introduction

THERE are many simple as well as sophisticated algorithms for linear classification i.e., classifiers that learn linear separating hyperplanes, such as perceptron, LMS, Wiener filter etc. However, there is no reason to believe that the data we encounter in the real world are linearly separable. In such a case, linear classifiers are no longer guaranteed to perform well. The perceptron algorithm does not even achieve convergence if the data are not linearly separable. This necessitates the development of non-linear classification algorithms.

In this assignment, we discuss and implement two non-linear classification algorithms - $k$ Nearest Neighbor ($k$-NN) and $k$-Means [1]. We see that even these algorithms are conceptually very simple and at the same time are powerful enough to tackle non-linearly separable data. Even though $k$-Means is usually employed in an unsupervised setting, with minor additions, can be used for classification.

We generate two 2D synthetic datasets which are described later. We train both the classifiers for each of these datasets. In order to compare the performance, for each classifier, we compute the confusion matrix using a test set containing points drawn from the same distribution as the training set.

Section II gives a more elaborate description of the two datasets. Section ... gives the outline of the $k$-NN algorithm and discusses its mathematical properties including complexity. Section .. details the working of the $k$-Means algorithm employed in a classification problem and discusses the various properties of the algorithm. The last section explains the training and testing protocol used and demonstrates the classification performance of the two algorithms on the two generated datasets.

## II. Generating Training and Test Data

According to the project requirement, we generate two 2D synthetic datasets to test and compare the performance of different classifiers. The first data set is from two Gaussian clusters of 2D data, so this dataset consists of points belonging to two classes. In our project, both clusters are 2D Gaussian distributions with unit variances but different means. The first cluster has mean $\mu_1 = (0, 0)$ while the second cluster has mean $\mu_2 = (2.5, 0)$. In the mean time, we set the covariance of two features in the distribution to 0, so that the two features of the data are uncorrelated. We generate 1000 data points from each cluster as the training data and label the data points from the first cluster as class 1 while the data points from the second cluster as class 2. Then from the same distribution, we generate another 1000 data points from each cluster as the testing data. In the experiment, we use "mvnrnd" command which is a built-in function to create multivariate normal random numbers to generate the dataset. In Figure 1, the first two plots show the pdfs of two 2D Gaussian clusters and the last plot shows the training data points we have generated for the experiment.

The second dataset also consists of points belonging to two classes. Points inside the unit circle belong to class 1 while points outside belong to class 2 and the unit circle is centered at $(0, 0)$. We generate 1000 data points respectively from inside and outside unit circle as the training data and also generate another 2000 data points in the same way for the testing data. This is done by randomly generating the angle of the data points between 0 to $2\pi$ and the radius of the data points between 0 to 1 (for the outside unit circle data points, we add a unit length offset). Then we use the $sin$ and $cos$ functions to get the $x$ and $y$ values of the data points. Figure 2, shows the plot of the training data points we have generated for the experiment.

## III. Classification Algorithms

### A. k-NN

$k$ - Nearest Neighbor (kNN) algorithm is a non-parametric method of density estimation and is one of the simplest classification methods in pattern recognition [2]. The main idea of $k$-NN is that a sample belongs to a category if most of the $k$ nearest neighbors of this sample belong to the same category. Thus, $k$-NN algorithm determines the category only based on the nearest one or more samples of the entire dataset. Since kNN is mainly based on the limited surrounding neighbor samples, this algorithm is well suited for samples which have many overlaps. The algorithm is outlined in Algorithm 1.

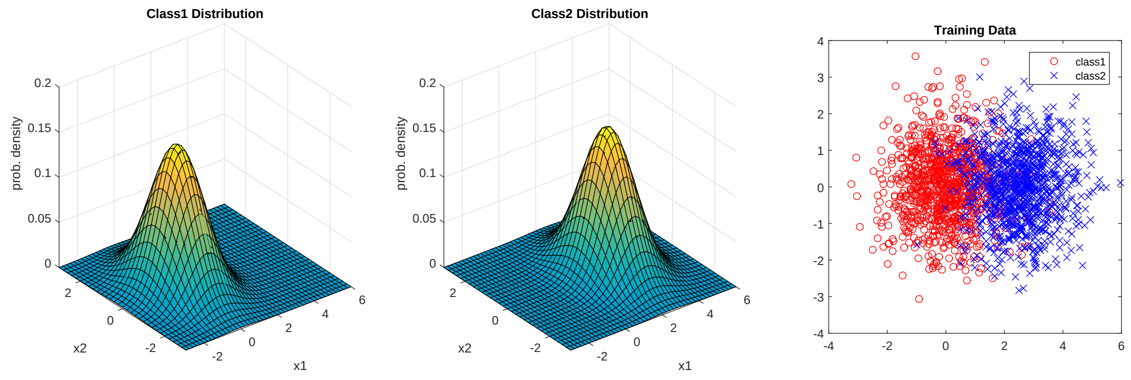

Fig. 1. (Left) pdf of Gaussian cluster with $\mu_1 = (0, 0)$ - Class 1. (Center) pdf of Gaussian cluster with $\mu_2 = (2.5, 0)$ - Class 2. (Right) Training dataset 1 generated from the two pdfs in (a) and(b).

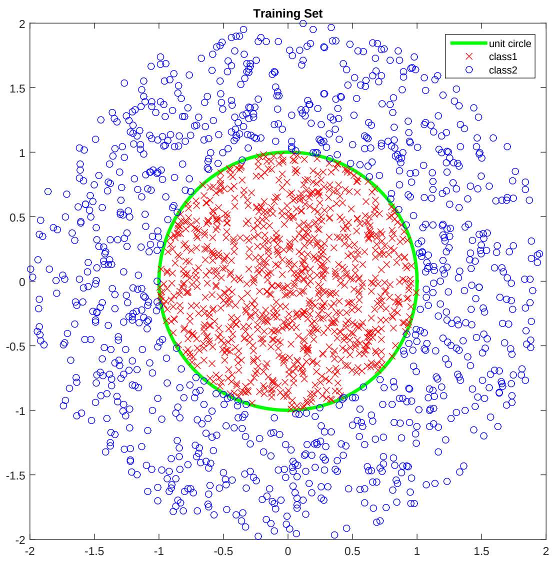

Fig. 2. Training dataset 2 with the datapoints belonging to two classes depending on whether they are inside (Class 1) or outside (Class 2) the unit circle.

**Algorithm 1** $k$-NN

```
Input: Training Data: $X = \{\mathbf{x}_i\}$, Labels: $\{y_i\}, i = 1, 2, ..n, y_i \in \{-1, 1\}\}$, Test Data: $\hat{X} = \{\hat{\mathbf{x}}_j\}, j = 1, 2, ..m$, No. of nearest neighbors = $k$.
Output: Predictions $\hat{y}_i \in \{-1, 1\}\}, i = 1, 2, ..m$
  for $\hat{\mathbf{x}}_\mathbf{j} \in \hat{X}$ do
    for $\mathbf{x}_\mathbf{i} \in X$ do
      Compute $d_i = ||\mathbf{x}_\mathbf{i} - \hat{\mathbf{x}}_\mathbf{j}||_2^2$
    end for
    $[\mathbf{d}_{sort}, I] = sort(\{d_i\}, ascend)$, #$I$ contains the indices of the sorted elements in $\mathbf{d}$.
    $V = X(I(1 : k))$
    Count $n_1 = \#V_i$ with label $1$
    Count $n_{-1} = \#V_i$ with label $-1$
    if $n_1 > n_{-1}$ then
      $\hat{y}_j = 1$
    else
      $\hat{y}_j = -1$
    end if
  end for
```

The algorithm shown here is a naïve implementation with complexity $O(kdn)$, where $d$ is the dimension of the data. This is expensive for large datasets. Faster variants exist such as dimensionality reduction before $k$-NN [3], which produce approximate results, space partitioning [4] etc. Although we do not it do it explicitly, $k$-NN estimates the distribution $p(\mathbf{x})$ non-parametrically and as $k \to \infty$, the estimates converge to the true density. The optimal value of $k$ is chosen through cross validation, usually as the smallest value that provides the desired accuracy.

### *B. k-Means*

Since $k$-Means was originally developed as a clustering algorithm, we will describe it in this fashion. k-means on its own is not designed to employ labels and perform classification and as such, the performance of this algorithm on the datasets that we have in this assignment, especially on the circular dataset as we will discuss later, is bound to be poor when compared to classification algorithms, even a simple one such as $k$-NN.

For a given dataset, $k$-Means returns $k$ vectors, called the cluster centers, such that the dataset is divided into $k$ clusters. Each point in the dataset is assigned to a single cluster center that is "closest" to it in some sense.

More formally, given a set of $n$ datapoints $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\}$, we want to obtain $k$ disjoint clusters $C = (C_1, C_2, ...C_k)$ with cluster centers, $\{\mu_1, \mu_2, ...\mu_k\}$ where each $\mu_i$ is the mean of the points in cluster $i$, such that the total sum of intra-cluster distances over the entire dataset is minimized. Mathematically, we would like to find

$$argmin_C \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \tag{1}$$

It is known the problem in Equation 1 is NP-hard [5]. A heuristic algorithm called Lloyd's algorithm, widely known as the k-means algorithm, is generally used to solve the problem approximately. The solution produced by this algorithm depends heavily on the initialization.

The algorithm consists of two steps which are repeated one after the other until convergence, that is when the cluster centers, which are initialized randomly, no longer change. The first step is assigns all the datapoints to their closest cluster centers. The second step is used to calculate the updated cluster centers which are simply the means of the various clusters. The algorithm is outlined in Algorithm 2.

**Algorithm 2** $k$-Means for clustering

```
Input: Training Data: $X = \{\mathbf{x}_i\} i = 1, 2, ..n,$,
    No. of Clusters = $k$
Output: Cluster Centers $\{\mu_i\}, i = 1, 2, ...k$
    Initialize $\{\mu_k\}$ randomly by choosing $k$ points from the dataset
    while (Cluster Assignments Change) do
        for $i = 1 : n$ do
            Cluster($\mathbf{x}_i$) = $argmin_i ||\mathbf{x} - \mu_i||^2$
        end for
        for $i = 1 : k$ do
            $\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_i \in C_i} \mathbf{x}_i$
        end for
    end while
```

One of the inputs to the algorithm is $k$, the number of clusters. In practice, the optimal value of $k$ is chosen through cross validation.

In this assignment, the problem that we are tackling is classification. However, $k$-Means, which groups datapoints based on a metric, is not guaranteed to group datapoints belonging to the same class into the same cluster [6]. This is especially true for the circular dataset, where $k$-Means is expected to return clusters which almost definitely contain points belonging to both classes. Thus, we design an algorithm, based on $k$-Means that can perform reliable classification.

Consider a binary classfication problem. Instead of once for the entire dataset, we call the $k$-Means algorithm twice, once for each class. This ensures that the labels provided in the training data are utilized. This is outlined in Algorithm 3.

**Algorithm 3** $k$-Means for binary classification

```
Input: Class 1 Data: $X_1 = \{\mathbf{x}_i\}, i = 1, 2, ..n_1$, Class 2 Data:
    $X_2 = \{\mathbf{x}_j\}, j = 1, 2, ..n_2$, No. of clusters for class 1, $k_1$
    and class 2, $k_2$
Output: Cluster centers for the 2 classes $\{\mu_i\}, i = 1, 2, ...k_1$
    and $\{\mu_j\}, j = 1, 2, ...k_2$.
    $\{\mu_i\}$ = $k$-Means($X_1, k_1$)
    $\{\mu_j\}$ = $k$-Means($X_2, k_2$)
```

For testing, if the testing point is closest to a cluster center in $\{\mu_i\}$, it is assigned to class 1. Otherwise, it is assigned to class 2. Note that there are two inputs $k_1$ and $k_2$ required for this algorithm, the optimal values of which are obtained through cross validation.

Even though the problem in Equation 1 is NP-hard, it can be solved approximately using Algorithm 2. But, the optimal is no longer guaranteed to be found. The cluster center set heavily depends on the initialization and a good solution can be obtained by using multiple runs. The complexity of $k$-means algorithm is $O(nkd)$, where $n$ is the number of datapoints, $k$ is the number of clusters and $d$ is the number of dimensions.

## IV. Experiments

The two algorithms described above are implemented using MATLAB. We generated two data sets as explained in Section II.

As we are tackling a classification problem, we first train our algorithm using the training test data and compute the accuracy of the model on the test data which contain points drawn from the same distribution as the training data. In order to tune the parameters required for the algorithms, such as the number of nearest neighbors in the case of $k$-NN and the number of clusters in the case of $k$-means, we use cross-validation on the training data. For this purpose, we subdivided both the training datasets into 750 data points as training data and 250 data points as cross-validation data.

### A. k-NN algorithm

This implemented as per Algorithm 1. Cross Validation for the $k$-NN algorithm is performed using the $m$-fold cross validation technique. The training set is divided into $m$ disjoint partitions and a single set is set aside as the cross validation set. The remaining $m - 1$ partitions are now used as training samples and the process is repeated over until each partition is used at least once as the cross validation set. For a given value of $k$, all the test set points are assigned labels. The accuracy is computed for each iteration and the average accuracy is calculated for the given $k$. Cross validation is repeated for multiple values of $k = 1, 2, ..10$ and the optimal value of $k$ is picked in order to proceed with the testing data set. The predictions of the $k$-NN algorithm on the Gaussian cluster dataset are displayed in Figure 3(b) The value of $k$ used here, that provided maximum accuracy during cross-validation is 9. The confusion matrix is shown in Table I. The predictions on the circular dataset are displayed in Figure 4(b). $k = 4$ is used in this case. The confusion matrix is shown in Table III.

### B. k-Means algorithm

This is implemented as per Algorithms 2 and 3. Cross validation is done over $k_1 = 1, 2, 3, 4, 5$ and $k_2 = 1, 2, 3, 4, 5$. The cross validation step returns an accuracy matrix of size $5 \times 5$, the row denoting the value of $k_1$ and the column denoted the value of $k_2$. The indices corresponding to the maximum value of this matrix is then picked as the optimal number of clusters for class 1 and class 2 respectively and the cluster centers corresponding to these values are used for the test set. The predictions of the $k$-means algorithm on the Gaussian cluster dataset are displayed in Figure 3(c) The value of $k_1$ and $k_2$ used here are... . The confusion matrix is shown in Table II. The predictions on the circular dataset are displayed in Figure 4(c). $k_1 = 4$ and $k_2 = 5$ are used in this case. The confusion matrix is shown in Table IV. Comparing these results with

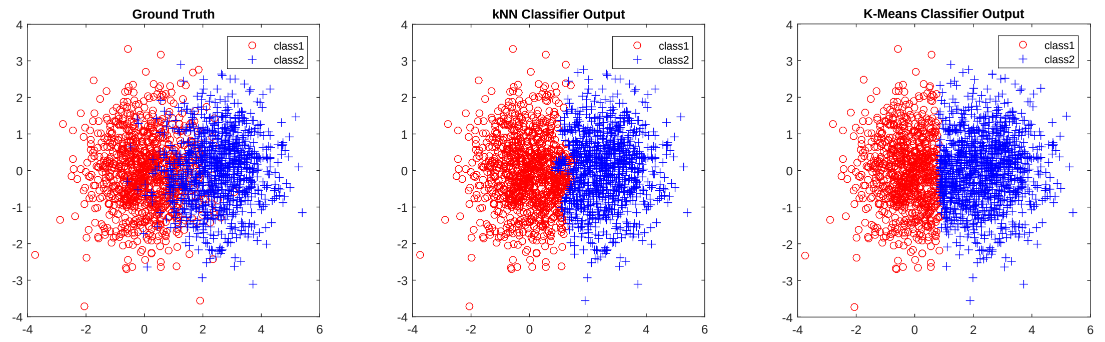

Fig. 3. (Left) Test data for the Gaussian clusters dataset- ground truth. (Center) Predictions from $k$-NN. (Right) Predictions from $k$-Means.

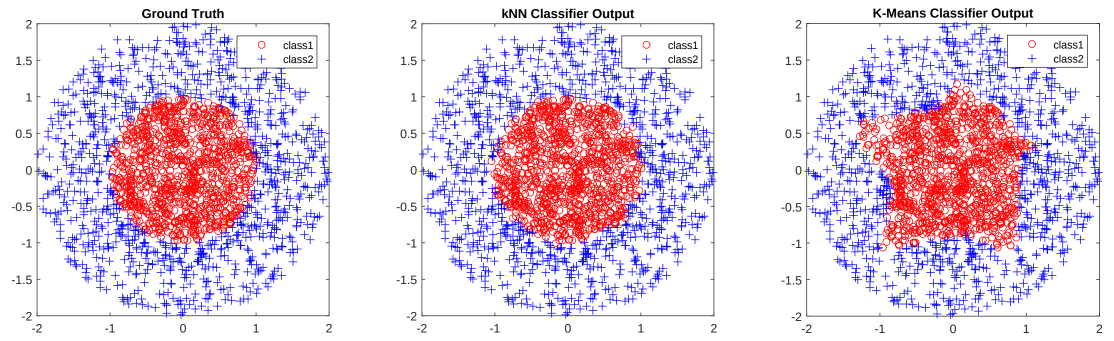

Fig. 4. (Left) Test data for the circular dataset- ground truth. (Center) Predictions from $k$-NN. (Right) Predictions from $k$-Means.

those of $k$-NN, we see that $k$-NN performs better for these datasets. This is expected because $k$-means is designed for clustering.

| | Class 1 | Class 2 |
|---|---|---|
| Class 1 | 874 | 126 |
| Class 2 | 110 | 890 |

TABLE I
CONFUSION MATRIX FOR GAUSSIAN CLUSTERS DATA WITH $k$-NN

| | Class 1 | Class 2 |
|---|---|---|
| Class 1 | 792 | 208 |
| Class 2 | 38 | 962 |

TABLE II
CONFUSION MATRIX FOR GAUSSIAN CLUSTERS DATA WITH $k$-MEANS

| | Class 1 | Class 2 |
|---|---|---|
| Class 1 | 988 | 12 |
| Class 2 | 21 | 979 |

TABLE III
CONFUSION MATRIX FOR CIRCULAR DATA WITH $k$-NN

| | Class 1 | Class 2 |
|---|---|---|
| Class 1 | 945 | 55 |
| Class 2 | 112 | 888 |

TABLE IV
CONFUSION MATRIX FOR CIRCULAR DATA WITH $k$-MEANS

## V. CONCLUSION

In this assignment, we studied two popular and powerful algorithms in pattern recognition namely $k$-Means and $k$-NN. The mathematical principles underlying these techniques were understood and their applications were explored. These methods were then applied to two distinct datasets and the confusion matrices were created. Optimal values of hyperparameters were computed using cross validation data. From the output figures shown above, it can be seen that both algorithms are able to find non-linear decision boundaries and classify the data well. Both algorithms are easy of implement. However, for large datasets, these algorithms are prohibitively expensive.

## REFERENCES

[1] Ethem Alpaydin, *Introduction to Machine Learning*, MIT Press, 2014.
[2] Duda, R.O., Hart P.E., and Stork D.G., *Pattern classification*, John Wiley & Sons, 2012.
[3] Bingham E., and Mannila H., *Random Projection in Dimensionality Reduction: Applications to Image and Text Data*, Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2001.
[4] Yu C., Ooi B. C., Tan K., Jagadish H.V., "Indexing the distance: An Efficient Method to KNN processing." VLDB. Vol. 1. 2001.
[5] Aloise, D., Deshpande, A., Hansen, P., Popat, P., *NP-hardness of Euclidean Sum-of-squares Clustering*, Machine Learning , 2009.
[6] Al-Harbi, Sami H., and Victor J. Rayward-Smith. *Adapting $k$-Means For Supervised Clustering*, Applied Intelligence 24.3, p 219-226, 2009.