Machine Learning using k Nearest Neighbours in Customer System and Sensor Applications

Bashirat Omobolaji Abdullahi (bashirat@aims.ac.za)

African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr Simukai W. Utete
African Institute for Mathematical Sciences, South Africa

18 May 2017

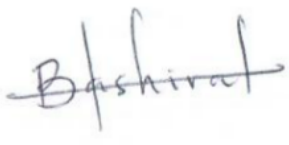
Submitted in partial fulfilment of a structured masters degree at AIMS South Africa

Abstract

k nearest neighbours (kNN) has been widely used in many applications for different purposes due to its simple execution and high effectiveness. Systems that make use of customer data and sensor-acquired data are becoming increasingly common in the range of commercial applications. In this project work, we applied kNN to a customer system and, by extension, to a sensor application. kNN was applied to customer service data to classify target customers into different classes and also to recommend highly rated products to target customers. We have shown that if the customer data can be represented in terms of features, we can apply the same method to the sensor case of reporting features detected in an environment. Performance of kNN depends on a number of its design decisions and one of these decisions is the choice of similarity measure. We compared different similarity measures and show the effect of choice of similarity measures in customer service data. We worked out the similarities based on the customer preferences and we were able to extend this to the sensor case. Jaccard similarity is one of the similarity measures we considered that works well for customer service data and which can actually be adapted to large-scale systems. kNN is one of the types of machine learning algorithm which is flexible in nature in terms of being a lazy learner and also easy to understand.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.



Bashirat Omobolaji Abdullahi, 18 May 2017

Contents

1. Introduction

In this chapter, we will study machine learning using k nearest neighbours in two different applications. Systems that make use of both customer data and sensor-acquired data are becoming increasingly common in a range of commercial applications. We shall consider how k nearest neighbours can be useful and important for classification task in customer system and sensor applications.

1.1 Aim and Objectives

The purpose of this project is to use k nearest neighbours in customer system and sensor applications under the following objectives:

- investigate the properties of k nearest neighbours algorithm, its advantages and drawbacks.
- analyze the design decisions that should be made for k nearest neighbours classification.
- outline how k nearest neighbour learning can be applied in the classification task of customer records and sensor case.
- look at different metrics that can be used as measures in customer system and sensor applications.
- design and implement customer records and show the effect of different design decisions of the classifier on the results.

1.2 Background and Literature Review

In this section, brief overview of related work to this project work shall be reviewed. We shall discuss the background of k nearest neighbours and review what has been done in this project.

1.2.1 Machine Learning.

Machine Learning is about designing algorithms which gives the computer the ability to learn. It comprises three main components which include: the task, which is the problem we want to solve; the model, which is the algorithm we will use to solve the problem in order to give a desired output; and the features, which aids the model in making good prediction. The main idea of machine learning is making use of the right features to build the right model that will output the right result. The main learning algorithms in Machine Learning are supervised learning and unsupervised learning (Flach, 2012).

Supervised learning is a learning algorithm which generates a function that maps the input data (or training data) to the desired output data. In supervised learning, dataset is given and the output of the result is known. Supervised learning can be classified into two parts which are the classification and regression. In classification learning, prediction of results are in discrete output while the prediction of results are in continuous output in regression learning.

In unsupervised learning, the training data are given as an unlabelled data. Some of the approaches to unsupervised learning include: clustering, neural networks, support vector machine and many others. However, performance of a learning algorithm can be evaluated by considering the accuracy, false positives and false negatives of the algorithm which we shall explain in detail in Chapter 4.

1.2.2 k Nearest Neighbours (kNN).

We shall be using a supervised learning algorithm in this project work which is the k nearest neighbours algorithm. kNN is a non-parametric, instance-based and lazy learning algorithm which is usually known for its simple execution, highly effective and efficient techniques.

kNN is a supervised learning which can predict the class of a new observation based on the known classes of its nearest neighbours. It is referred to as non-parametric algorithm because it does not make any assumptions on the basic. kNN is an instance-based learner because it does not learn from the model but rather memorize all the entire training examples and make use of it as its basic knowledge when about to make predictions. Moreover, kNN is known as a lazy learner due to the fact that it stores all the training examples and delay the classification of the model till test instance is available for classification.

The basic idea of kNN is to find the nearest neighbours among all the training examples that are similar or very close to the query point and then label the query point as the class of one of its k nearest neighbours. In kNN classification, the query point can be classified as the class of the majority vote of its nearest neighbours.

The following are the steps on how kNN works for a given query point with unlabelled class and training data with known classes. We need to

- find the distances between the query point and each of the training data.
- take the k smallest distances among all the distances measured to the query point as the k nearest neighbours of the query point.
- assign the class of the majority vote among the corresponding classes of k nearest neighbours as the class of the query point.

Some of the advantages of kNN are as follows: robust to noisy data; easy to implement and works effectively for classification problem; results are reliable if there is large training data; and simple and easy to understand the classification techniques (Flach, 2012).

Despite its advantages, kNN has some drawbacks which include: to determine the optimal value for k might be difficult; it requires large space of memory when implementing with large data; it is computationally expensive when computing the distances from query point to each training samples; high dimensional space of dataset leads to low accuracy of the results; and it does not learn from the training data but rather uses it for classification (Flach, 2012).

1.2.3 Similarity measures.

kNN requires similarity or distance measure before it can be used for classification. We looked into different distance measures such as Euclidean distance, Minkowski distance, Hamming distance, Manhattan distance, Chebychev distance, Mahalanobis distance (Flach, 2012). We also considered similarity measures such as Jaccard similarity and Cosine similarity (Leskovec et al., 2014) which we shall discuss later in the next chapter.

Properties of distance and similarity measures are discussed in this thesis (Deza and Deza, 2009). Distance measures are the measures used to find the distance between two points in a space while similarity measures are used to find how similar or dissimilar two sets are to each other. Problem encountered when trying to do learning in high-dimensional space is referred to as curse of dimensionality. This occurs when the data we work with is of very high dimensional space in which the space grows exponentially as the number of dimensions increases (McMullen, 2015).

Euclidean distance is one of the distance measure we considered for sensor application. It measures the distance between two points in an euclidean space (Leskovec et al., 2014). Euclidean distance is useful in sensor case because it measures how close or extreme the query point and each sensor nodes are to each other.

Jaccard similarity is one of the similarity measure we considered which compute the similarity between two finite sets. It is given as the ratio of the size of the intersection between two sets to the size of their union (Leskovec et al., 2014). It works efficiently well for binary data, sets or multisets and real values. For large scale system, Jaccard similarity can be estimated using minhashing techniques (Leskovec et al., 2014).

Cosine similarity is an important similarity measure for points in the plane and in higher dimensions (Widdows, 2004). It measures the similarity between two non-zero vectors by computing the angle between them (Leskovec et al., 2014). Cosine measure gives high similarity to the points that are of the same direction but gives no similarity to points that are perpendicular to each other (Widdows, 2004). It works efficiently well for sparse vectors but takes more time when computing for real values compared to Jaccard similarity.

1.2.4 Applications of kNN in Customer and Sensor Systems.

We applied kNN to the well-known Fisher iris dataset, obtained from (Lichman, 2013) using Euclidean distance as the distance measure and we evaluated the performance of kNN on the data using cross-validation in Waikato Environment for Knowledge Analysis (WEKA) open source (Markov and Russell, 2006). Cross-validation is a way of estimating the accuracy of classification, i.e., it is used to predict the fitness of a model. kNN classification is used in different applications such as medical treatment, image processing, recommendation system, sensors, network graphs, web pages and many others. The two applications that kNN will be applied to in this thesis are the customer system and sensor applications.

In customer system, services are rendered to customers in order to satisfy their needs. Knowing the needs of customers, we can recommend products that may be of interest to them by comparing their similarities. One of the approach of recommending products to customers which is based on similarity is referred to as collaborative filtering. It will be familiar to users of online systems as recommendations based on customers who liked similar items.

Collaborative filtering is one of the recommendation approach where we recommend products or items to target customers due to the interest or similar taste of other customers on the products (Leskovec et al., 2014). One of the applications kNN can be applied to in customer system is the collaborative filtering system which is based on the similarities of customer preferences (Liu and Shih, 2005).

kNN can be used for classification in customer system for classifying customers and also recommending highly rated products to target customers. kNN is also applicable in information retrieval system for classifying relevant documents in a large corpus of documents. We considered Jaccard similarity as the similarity measure used for customer service records and which can also be estimated for large-scale systems (Leskovec et al., 2014).

By extension, we considered how kNN can be applied in sensor case. kNN can be used in sensor application to detect the features of the query location by using the features of the k nearest sensor nodes to the query point (Han et al., 2015). We shall give a detailed explanation of the design decisions of kNN in the sensor application in Chapter 3.

1.3 Organization of the Thesis

This project consists of five chapters which are organized as follows. This chapter presents the introductory part which consists of the aim and objectives of the project work, background and literature review. Chapter 2 presents the analysis of the design decisions for kNN and the different similarity and distance measures used in kNN. Chapter 3 presents the applications of kNN, i.e., how kNN are applicable to customer system and sensor case. We give results and discussed them in Chapter 4. Finally, we present the conclusions in Chapter 5.

2. Similarity and Distance Measures

In this chapter, we shall focus on design decisions that must be made for kNN such as how to choose the value of k, choosing the similarity measure and how to combine the nearest neighbours decisions in classification with the kNN. We shall also take into consideration different similarity and distance measures and explain how some of these measures are useful in customer system and sensor application.

2.1 Design Decisions for kNN

kNN is one of the learning algorithm used for classification in sorting items or objects into different categories. In this section, we are concerned with how the design decisions must be made for kNN in classification problem. The purpose of kNN is to assign a class to the test instance by means of finding the nearest neighbours in the training samples which are very close to the test instance and then base its classification on the class of the majority vote of its nearest neighbours.

Formally, this fact can be described as follows: Let a function F be defined as

$$F : X \to \mathcal{C}, \tag{2.1.1}$$

where X is the training set of the observed data and $\mathcal{C} = \{c_1, \ldots, c_m\}$ is the set of classes corresponding to each training data.

2.1.1 Properties.

Given the main idea of kNN, let us now focus on the major design decisions that must be made for kNN classification. The properties listed below are the desired properties which are needed in kNN. The specifications required in kNN are given as follows: Let

- $X = (x_i, c_i) \in \mathbb{R}^{n \times m}$, $\mathcal{C}$ and X_q be the set of training data, set of classes for each training data and set of test instances respectively;
- $x_i \in X,\ c_i \in \mathcal{C}$ and $x_q \in X_q$ be the corresponding i^{th} training data, i^{th} class and the query point respectively;
- $k \in \mathbb{N}^+$ be the number of nearest neighbours in the training data to the query point;
- $N \in \mathbb{N}$ be the number of instances in the dataset;
- $\Omega_i(x_i)$ and $\Omega_q(x_q)$ be the features set for each training data and the query point respectively.

Property 1. (Distance) Let $d_i(x_q, x_i)$ be the corresponding distance from each i^{th} training data $x_i \in X$ to the query point $x_q \in X_q$ corresponding to i^{th} class of x_i. We define the distance $d_i(x_q, x_i)$ as

$$d_i(x_q, x_i) : X_q \times X \to \mathbb{R}^+, \quad \forall\, x_q \in X_q \text{ and } x_i \in X.$$

Thus, the distance used in k nearest neighbours depends on the type of features $\Omega_i(x_i)$ given in the datasets.

Property 2. (Weighting) Let $w_i(x_i)$ be the weight on each i^{th} training data. Then, we define the weight on k nearest neighbours $w_k(x_k)$ among the weight $w_i(x_i)$ to the query point x_q as

$$w_k(x_k) = \begin{cases} 1, & \forall\ \min\{d_i(x_q, x_i)\} \\ 0, & \forall\ \max\{d_i(x_q, x_i)\} \end{cases}$$

Thus, the classification of the query point made by the majority weighted voting is given as

$$c_q = \text{arg max} \sum_{(x_k,\, c_k) \in X} w_k(x_k) \times c_k.$$

Property 3. (Choice of k) Determining the optimal value for k is difficult but however the general rule thumb for determining k is given as $k = \sqrt{N}$ (Duda et al., 2012) where N is the number of instances in a dataset.

For a two class problem, the best value to choose for k is odd numbers, i.e.,

$$\forall\ c_i \in \mathcal{C},\ \text{we choose } k = n + 1 \text{ such that } \ \ i = 1, 2 \text{ and } n = 0, 1, 2, \ldots$$

Property 4. (Feature Selection) Some features are useful in making good decisions. Features which are not useful are referred to as redundant features which may either cause misclassification of the test instance or does not have effect on the prediction of the test instance.

Redundant features might be a problem to kNN however it can be solved by preprocessing the data before applying kNN but this will not be covered in this essay.

Let $\gamma_i \in \Omega_i(x_i)$ and $\gamma_q \in \Omega_q(x_q)$ be the i^{th} features in each feature sets of the training data x_i and in the query point x_q respectively. Then, we say x_i has a useful feature to x_q if $\gamma_i = \gamma_q$ and it is a redundant feature if otherwise, i.e., $\gamma_i \neq \gamma_q$.

Property 5. (Computational Cost) Let n be the number of training data consisting of $\Omega_i(x_i)$ features and k be defined just as above. Let the run time to compute the distance of each training data of $\Omega_i(x_i)$ features to the query point be $O(d)$. Thus, the run time to compute all distances of n training data to the query point is given as $O(nd)$.

To determine the k smallest distances, we sort all the distances and then iterate from 1 to k. Therefore, the run time to sort all the n distances and then iterate is $O(nk)$. Hence, the overall computational cost to run the kNN is given as $O(nd + nk) = O(\rho)$ where $\rho = nd + nk$.

Property 6. (Surjectivity) The kNN classifier is surjective in nature, i.e., $\forall\ c_i \in \mathcal{C}\ \ \exists$ at least one training data $x_i \in X$ such that

$$F(x_i) = c_i.$$

Property 7. (Robustness) We would like the kNN classifier to be robust to noise. Let $x_i^* = x_i + \eta(x_i)$ where $\eta(x_i)$ denotes the function of noisy data added to the training data such that

$$F(x_i^*) = F(x_i).$$

Property 8. (Classification) To assign the class of the query point, we need to sort all the distances of the training data to the query point in ascending order, i.e.,

$$d_{\text{sort}} = \underset{>}{\text{sort}}\ \{d_i(x_q, x_i)\},$$

where d_{sort} are the sorted distances. So, we choose the first k smallest distances as the nearest neighbours of the query point x_q, i.e.,

$$d_k = \min_{i=1}^{k} \{\ d_{\text{sort}}\},$$

where d_k is the set of k smallest distances.

Then, classification is based on assigning the predominant class among the corresponding classes c_k of d_k to the query point x_q, i.e.,

$$F(x_q) = c_q \text{ such that } c_q = \arg\max_{c_k \in \mathcal{C}} \{c_k \mid i = 1, \ldots, k\}.$$

The above desired properties of kNN are useful and important. The mathematical analysis of the design decisions of kNN used for classification will be explained in the next section.

2.2 Mathematical Analysis

In this section, we shall explain the mathematical analysis of the design decision of kNN which can be used for classification. The design decisions of kNN are as follows:

- **Choosing value for k**: One of the most important decision when building up the kNN algorithm is the choice of k due to its strong influence when predicting the class of the test instance. There are several ways of choosing good values of k so as to predict the class of the query point. The general rule of thumb on the choice of k is equal to the square root of the number of instances (Duda et al., 2012). However, we can not determine the value of k that can be appropriate but nevertheless, the value of k usually depends on the number of instances in a given dataset (Flach, 2012). Ways of selecting a good value of k are given as follows:

  1. For two-class problem, we need to choose k as odd values so as to avoid falling into ties of classes otherwise we choose randomly or apply weight. In this thesis, we shall only consider two classes for customer service problem which are either high-spending customers or low-spending customers. Suppose we fall into "ties", say "High" or "Low", we choose randomly by considering the effect of false positives and false negative on the results which we shall discuss later in section 4.3.

  2. Cross-validation is another strategy to choose a good value of k. It helps to predict the fitness of a model by using validation set to test the training data but it also consumes time with large dataset. There are different types of cross-validation but in this essay, we considered k-fold cross validation. In section 4.1, we split the iris data into 10-fold cross-validation and applied kNN in WEKA to predict the results.

  3. Choosing small values for k can cause overfitting when predicting which may not provide correct output (Flach, 2012). Overfitting in this case implies having a good fit for the available data but does not generalize well to new observations.

  4. Choosing large values for k can cause over-generalization which may select too many irrelevant data as the nearest neighbours (Flach, 2012). The extreme of this would be using all of the instances in the dataset.

  Due to the difficulties in determining the optimal value for k, many researchers make use of algorithm tuning to experiment their results and look for which works best. Nevertheless, we need to choose the value of k to be large enough so as to minimize the rate of misclassification and small enough so that the nearest neighbours in the training examples will move closer to the query point (Flach, 2012).

- **Choosing similarity measure**: Another essential part in kNN is the way of choosing the distance or similarity measure. Before a decision can be made, we need to find the distance between the query point and the training data. Then, we choose the k nearest neighbours to the query point and take the class of the majority vote to decide the class of the query point. In determining the similarity measure, we take feature selection into consideration (Flach, 2012). Different types of similarity measures can be used in k nearest neighbours depending on selected features we take into account. Selected features can either be strings (e.g text), numerics (e.g real values) or categorical (e.g binary).
- **Distance Weighting**: The idea of using distance weighting occur when we fall into ties of different classes. "Ties" in this case simply means having the same number of votes for each class. Way to break such ties is to apply random breaking or weighting scheme. Random breaking simply implies choosing the class at random and assigned it to be the class of the query point.

  Weights are applied to the training data so as to weigh the contribution of each of the k neighbours according to their distances to the new observation which will give greater weight to the nearest neighbours. Neighbours with smaller distances are weighted heavily than neighbours with greater distances in the sense that nearest neighbours are weighted as 1 while the farthest neighbours are weighted as 0 and other neighbours are scaled linearly in between the interval [0, 1] (Gou et al., 2012).
- **Classification**: To classify a new observation or query point, a distance has to be measured from each training data to the new observation. We choose the k nearest neighbours in the training data having smallest distances to the new observation. Classification is then based on the classes of the k nearest neighbours that were chosen (Flach, 2012). We assigned the class of the majority vote as the class of the new observation. If we fall into ties of classes, then we based the classification using majority weighted vote of the classes or we choose the class randomly and assign it as the class of the new observation.

2.3 Measures to find Similarity or Distance

Most of the learning algorithms used in Machine learning are generally based on exploiting similarity from new observation data to existing training data. One of the relevant properties of kNN is the notion of distance. kNN requires distance or similarity measure to find the distance from the query point to each of the training data. In this section, we shall focus on the properties of distance and similarity measures, consider their different types and explain how some of these measures are useful to customer system and sensor applications.

Distance measures are measures used to find the distance between two set of points while similarity measures are measures used to find how similar or different set of objects are to each other. We say instances are similar in terms of their feature values if their distances are relatively small. Nevertheless, different distance and similarity measures are used in kNN depending on the type of features given in the dataset, i.e features can either be strings, categorical (variables having two or more classes) or numerics (real values).

2.3.1 Distance Metrics.

Let X be a set of points called Space. We define the distance metric on the space as a function

$$d(x, y) : X \times X \to \mathbb{R}^+ \quad \text{such that} \quad x,\ y \in X$$

satisfies the following axioms (Deza and Deza, 2009) :

(i) $d(x,y) \geq 0$ if and only if $x \neq y$, (non-negativity)

(ii) $d(x,y) = 0$ if and only if $x = y$, (identity)

(iii) $d(x,y) = d(y,x)$, (symmetric)

(iv) $d(x,y) \leq d(x,z) + d(z,y)$, where $z \in X$ (triangle inequality)

The above conditions implies that (i) distance is always positive when they differ; (ii) distance from point to itself is zero when they are the same; (iii) distance from point x to point y is the same as distance from point y to point x; and (iv) the distance between two points is shorter than the sum of any other two distances (Flach, 2012).

Distance measures are useful and important in customer applications which satisfy the above properties of distance metrics given as follows: distance measured between two different customers can not be negative; distance measured between two customers is zero if they are the same; two customers are symmetric in nature since the distance from customer c_i to customer c_j is the same as the distance from customer c_j to customer c_i; distance measured from customer c_i to customer c_j is small compared to the sum of the distances measured from customer c_i to customer c_k and customer c_k to customer c_j.

2.3.2 Similarity Measure.

Let X be set of points. We define the similarity measure on X as a function

$$s(x,y) : X \times X \to \mathbb{R}^+ \text{ such that } x,\ y \in X$$

satisfies the following properties (Deza and Deza, 2009) :

(i) $s(x,y) = 1$ if and only if $x = y$ (identical)

(ii) $s(x,y) = s(y,x)$ (symmetric)

(iii) $s(x,y) \leq s(x,x)$ (inequality)

The above properties implies that (i) there is maximum similarity between two points if they are the same; (ii) the similarity between x and y is the same as the similarity between y and x; and (iii) the similarity between two points can not be greater than the similarity between a point and itself.

The similarity measure is useful in customer system application which satisfies the above axioms as follows: there is a maximum similarity between two customers if they have the same taste or buying habits; the similarity between customer A and customer B is the same as the similarity between customer B and customer A which satisfies the symmetric property; and the similarity between two customers can not be greater than the similarity between customer to himself.

Similarity measure can either range from 0 to 1 or -1 to 1 depending on the type of similarity. However, dissimilarity (or distance) d can be calculated from similarity but which depends on the type of the similarity measure we are given. Similarity sometimes are measured as $1 - d$ or $\frac{1-d}{2}$ respectively.

Different types of distance and similarity measures that are taken into consideration are given as follows:

Minkowski Distance - It is a metric in a normed vector space which is referred to as l_p-norm (Flach, 2012). It is denoted by $d_p(x,y)$ which is given as

$$d_p(x,y) = \left(\sum_{j=1}^{n} (x_i - y_i)^p \right)^{\frac{1}{p}} = ||x_i - y_i||_p \ , \qquad p > 0.$$

Euclidean Distance - It measures the distance between two points in an euclidean space. It is a special case of Minkowski distance when $p = 2$ and referred to as l_2-norm and also a special case of Mahalanobis distance (Flach, 2012). It is denoted by $d_2(x, y)$ which is given as

$$d_2(x,y) = \left(\sum_{j=1}^{n}(x_i - y_i)^2\right)^{\frac{1}{2}} = ||x_i - y_i||_2 = \sqrt{(x-y)^T(x-y)}.$$

Euclidean distance is a distance measure which is simple and easy to calculate but does not work well with data of high dimensions. It is useful in sensor application for computing the distance between each sensor node and the query point in order to make it easy to detect the features of the query location.

Manhattan Distance - This distance is referred to as City block or l_1-norm. It measures two points along the coordinates axes at right angle (Flach, 2012). It is denoted by $d_1(x, y)$ which is given as

$$d_1(x,y) = \sum_{j=1}^{n}|x_i - y_i|.$$

Hamming Distance - It measures the distance between two strings of equal length by counting the number of characters for which the strings differ (Leskovec et al., 2014). It is denoted by $d_o(x, y)$ which is given as

$$d_o(x,y) = \sum_{i=1}^{n}(x_i - y_i)^o = \sum_{i=1}^{n} I\,[x_i = y_i],$$

where I is the identity (Flach, 2012).

For instance, the Hamming distance between "mark" and "cart" is 2.

Chebyshev Distance - It measures the distance between two vectors or points having standard coordinates (Flach, 2012). It is also called l_∞-norm and denoted by $d_\infty(x, y)$ which is given as

$$d_\infty(x,y) = \max\left(|x_i - y_i|\right).$$

Mahalanobis Distance - This measures the distance between a point and distribution of data (Flach, 2012). The data distribution is expressed by the mean and covariance matrix which is given by

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)},$$

where S is the covariance matrix.

Jaccard Similarity - It measures the similarity between two finite sets and defined as the ratio of the size of the intersection between two sets to the size of their union (Leskovec et al., 2014). It is denoted as $J(A, B)$ which is given as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Jaccard similarity of two finite sets satisfies the following properties:

- it ranges from $[0,\ 1]$,

- it results into 1 if they are of the same elements,
- it results into 0 if the two sets are disjoint, i.e., the numerator becomes zero,
- it becomes large if the two sets are closer to each other,
- it becomes relatively small if they are far, i.e., if one of the set is larger than the other.

Hence, the higher the result of the Jaccard similarity, the more the sets becomes similar to each other.

For instance, let $A = \{a, b, c, d\}$ and $B = \{a, b, d, e, f, c\}$ be two finite sets. Then, Jaccard similarity is calculated as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|\{a, b, c, d\}|}{|\{a, b, c, d, e, f\}|} = \frac{4}{6} = 0.6667,$$

which implies that the two sets are 67% similar to each other.

Jaccard similarity is one of the similarity measures used to compare two objects to see how similar the objects are to each other. We make use of Jaccard similarity in customer application since it helps in comparing the similarity between customers and classifying them into a different segment. It is also useful in recommendation system for comparing products purchased by different customers so as to recommend the highly rated products to target customers. Jaccard similarity can also be estimated for large scale system.

Jaccard Distance - It measures how two finite sets are different from each other. It is defined as the ratio of the size of symmetric difference between two sets to the size of their union (Leskovec et al., 2014). It is the complement of Jaccard similarity which is denoted by $d_J(A, B)$ and is given as

$$d_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} = \frac{|A \triangle B|}{|A \cup B|}.$$

Consider the example of the Jaccard similarity, then we calculate the Jaccard distance as

$$d_J(A, B) = 1 - J(A, B) = 1 - \frac{4}{6} = \frac{2}{6} = 0.3333,$$

which implies that the two sets are 33% different from each other.

Edit Distance - It is the minimum number of insertions or deletions to convert one string into another. It can be referred to as Levenshtein Distance if it takes substitution, insertion or deletion into consideration. It can also be computed as a Longest Common Subsequence (LCS) of two strings which is defined as the sum of the length of the two strings minus twice the length of their LCS (Leskovec et al., 2014).

For instance, let $x = abcde$ and $y = bcda$ be the two given strings. We calculate the edit distance as

$$d(x, y) = |x| + |y| - 2|LCS(x, y)| = 5 + 4 - 2(4) = 1.$$

Thus, the higher the edit distance, the more the difference between the strings.

Edit distance can be useful in customer application which could be used to convert customer data records into strings. Suppose we record each item purchased by each customer as 1 and 0 if otherwise, we can actually convert this binary data into set of items purchased by each customer as strings.

Cosine Similarity - It measures the cosine angle between two non-zero vectors of an inner product space (Leskovec et al., 2014). Cosine similarity can be calculated as

$$\cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{||x||_2||y||_2} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}.$$

where x_i and y_i are the components of vectors x and y respectively.

The cosine similarity ranges from 0 to 1. Thus, if cosine angle is 0 implies that there is maximum similarity and if it is 1 implies no similarity (Widdows, 2004). To compute the similarity between two vectors, the vectors must be of the same equal length.

For instance, consider the two vectors $x = (0, 1, 1, 0, 1, 1)$ and $y = (0, 0, 1, 0, 0, 0)$. We compute the cosine angle between x and y as

$$\cos(\theta) = \frac{\sum_{i=1}^{6} x_i y_i}{\sqrt{\sum_{i=1}^{6} x_i^2}\sqrt{\sum_{i=1}^{6} y_i^2}} = \frac{0+0+1+0+0+0}{\sqrt{1^2+1^2+1^2+1^2}\cdot\sqrt{1^2}} = \frac{0+0+1+0+0+0}{\sqrt{4}\cdot\sqrt{1}} = \frac{1}{2},$$

$$\Rightarrow \theta = \cos^{-1}\left(\frac{1}{2}\right) = 60^o,$$

which implies that the angle between the two vectors are 60^o from each other. Thus, the two vectors are 0.5 similar to each other.

Cosine similarity is useful in customer data records which can be used to find the similarity between customers to know how similar they are to each other by calculating the angle between them. In the above example, it works efficiently for sparse vectors and does not take the magnitude of the vectors into consideration.

Suppose we need to find the similarity between the items purchased by each pair of customers which can be represented as vectors, the cosine similarity does not take the quantity of each item into consideration but rather looks at the direction of the similarity, i.e., it maintains the angle between the customers but does not consider the quantity of items purchased by each customer. Another drawback of cosine similarity is that it requires lot of time when calculating vectors of high dimensions (McMullen, 2015).

Hausdorff Distance - It measures how close or extreme two sets are from each other, i.e., it measures from one point in a set to the nearest point in the other set. Let X and Y be two non-empty subsets of a metric space. We define the Hausdorff distance $d_H(X, Y)$ of X and Y as

$$d_H(X, Y) = \max\{\sup_{x \in X} d(x, Y),\ \sup_{y \in Y} d(y, X)\},$$

where

$$d(x, Y) = \inf\{d(x, y) \mid y \in Y\} \text{ and } d(y, X) = \inf\{d(x, y) \mid x \in X\}.$$

Two sets are said to be close using Hausdorff distance if every point of either of the set is close to some of the points in the other set (Huttenlocher et al., 1993).

Example : Consider the metric space of the real numbers $\mathbb{R}$ with a metric d given by

$$d(x, y) := |x - y|, \qquad \forall\, x, y \in \mathbb{R}.$$

Suppose the two subsets of the metric space is given as

$$X := [2, 9] \quad \text{and} \quad Y := [3, 7]$$

Then, we calculate $d(x, Y)$ and $d(y, X)$ as

$$\begin{aligned} d(x, Y) &= \inf\{d(x, y) \mid y \in Y\} = \inf\{d(2, [3, 7]),\ d(9, [3, 7])\} \\ &= \inf\{\{|2 - 3|, |2 - 7|\},\ \{|9 - 3|, |9 - 7|\}\} \\ &= \inf\{\{1, 5\},\ \{6, 2\}\} = \{1, 2\}, \\ d(y, X) &= \inf\{d(x, y) \mid x \in X\} = \inf\{d(3, [2, 9]),\ d(7, [2, 9])\} \\ &= \inf\{\{|3 - 2|, |3 - 9|\},\ \{|7 - 2|, |7 - 9|\}\} \\ &= \inf\{\{1, 6\},\ \{5, 2\}\} = \{1, 2\}. \end{aligned}$$

Therefore, the Hausdorff distance is given as

$$\begin{aligned} d_H(X, Y) = d_H([2, 9], [3, 7]) &= \max\{\sup_{x \in X}\ \{1, 2\},\ \sup_{y \in Y}\ \{1, 2\}\} \\ &= \max\{2, 2\} = 2. \end{aligned}$$

Jaro distance - It measures the distance between two given strings, say x and y which is defined as the minimum number of transpositions of single-character required to change one strings into the other. It is denoted as $d_j(x,\ y)$ and can be calculated as

$$d_{j(x,\ y)} = \begin{cases} 0, & \text{if } m = 0, \\ \dfrac{1}{3}\left(\dfrac{m}{|x|} + \dfrac{m}{|y|} + \dfrac{m - t}{|m|}\right), & \text{if otherwise,} \end{cases}$$

where

- $|x|$ and $|y|$ is the length of the strings;
- m is the number of matching characters in both strings;
- t is half the number of transpositions of strings.

We say two characters are $matched$ from strings x and y if they are the same and not farther than $\left[\frac{\max(|x|,\ |y|)}{2}\right] - 1$ (Wikipedia). Thus, the higher the Jaro distance for two strings, the more similar the strings are. Jaro distance ranges from [0, 1] such that 0 implies no similarity between the strings and 1 implies there is an exact match between the strings (Wikipedia).

Example : Find the Jaro distance between the two strings

$$x = \text{“thick”} \text{ and } y = \text{“trick”}.$$

In this example, $m = 4,\ |x| = 5,\ |y| = 5$ and $t = \frac{1}{2}(0) = 0$. Therefore, the Jaro distance is given as

$$d_{j(x,\,y)} = \frac{1}{3}\left(\frac{4}{5} + \frac{4}{5} + \frac{4-0}{4}\right) = \frac{1}{3}\left(\frac{8}{5} + 1\right) = 0.8667,$$

which implies that the strings are 87% similar to each other.

Jaro distance is relevant to matching of gene sequences but however not useful in this customer service data since we are not considering sequences but sets of data in this case.

Jaccard bag similarity - It can be defined as the ratio of minimum number of times the element appears in both sets to the sum of the number of times the element appear in each of the sets. The maximum value that Jaccard similarity for bags can attain is 0.5 (Leskovec et al., 2014). Similarity measure used to compare the similarity between multisets is referred to as Jaccard bag similarity (Leskovec et al., 2014). Multiset or bag is the generalization of a set consisting of elements occurring more than once in which its order is irrelevant. Sets and multisets make use of single brace $\{\}$ as their notations.

The difference between Jaccard bag and Jaccard similarities is that Jaccard bag similarity find the similarity between multisets which takes the number of each element that appears in the set into consideration while Jaccard similarity compare the similarity between two finite sets which its elements occur only once. For instance, consider $A = \{a, c, e\}$ and $B = \{a, a, a, b, d, d, e\}$. In this case, A is a set since its elements occur once while B is a multiset since some of its elements occur more than once. One of the questions that arises from Jaccard similarity is how to compare or find the similarity between set and multiset using Jaccard bag similarity? To compare set and multiset using Jaccard bag similarity, we consider the following example.

Let us compute the Jaccard bag similarity of each pair of the following multisets (or bags):

$$A = \{1, 1, 1, 2\}, \quad B = \{1, 1, 2, 2, 3\}, \quad \text{and} \quad C = \{1, 2, 3, 4\}$$

The Jaccard bag of similarity for each pair is given as

$$J(A,\ B) = \frac{|\{1,1,1,2\} \cap \{1,1,2,2,3\}\}|}{|\{1,1,1,2\} \cup \{1,1,2,2,3\}|} = \frac{|\{1,1,2\}|}{|\{1,1,1,1,1,2,2,2,3\}|} = \frac{3}{9} = \frac{1}{3}.$$

$$J(A,\ C) = \frac{|\{1,1,1,2\} \cap \{1,2,3,4\}|}{|\{1,1,1,2\} \cup \{1,2,3,4\}|} = \frac{|\{1,2\}|}{|\{1,1,1,1,2,2,3,4\}|} = \frac{2}{8} = \frac{1}{4}.$$

$$J(B,\ C) = \frac{|\{1,1,2,2,3\} \cap \{1,2,3,4\}|}{\{1,1,2,2,3\} \cup \{1,2,3,4\}|} = \frac{|\{1,2,3\}|}{|\{1,1,1,2,2,2,3,3,4\}|} = \frac{3}{9} = \frac{1}{3}.$$

In the above example, we compared the similarity between two multisets as in the case of A and B and also compared the similarity between multiset and set as in the case of B and C. In this case, we have the same Jaccard similarity for A and B and also for B and C which implies that they are 0.3333 similar to each other while A and C are 0.25 similar to one another.

Another question pertaining to Jaccard similarity is how to compare the similarity between a pair of sets in a large scale system?. To compute the similarity between a pair of set in a large scale dataset is a big challenge because it requires a lot of time to compare the similarity. Say, for instance we have ten million of customers in the database and we want to compare the similarities between each pair of customers implies that we need to compare the similarity for each $\binom{N}{2} = \binom{10^7}{2} \approx 5 \times 10^{13}$ pairs of customers. In this case, Jaccard similarity will not scale well for such large data.

One of the approach to compute Jaccard similarity for large scale system is to estimate Jaccard using minhashing method (Leskovec et al., 2014). Minhashing is a type of Locality-Sensitive Hashing (LSH) used for compressing large datasets into smaller datasets by means of a random permutation of sets in the first row in which the column has 1 (Leskovec et al., 2014). Minhashing uses a hashing function to hash values that are the same in the sets into the same bucket and the probability of the hashing values is estimated to be the result of the Jaccard similarity of the sets (McMullen, 2015).

Euclidean distance is widely used in various applications because it has straightforward interpretation. In optimality problem, it can be used for estimation and also works well with numeric data. One of its major drawbacks is the case of computing the distance directly from the data given which might be a mixture of numerical and categorical variables in nature or having the same data type of attributes but with different scales.

This can be a problem because to compute distance with different scales, one side will be larger than the other which will have a high influence on the result. The way to solve problem with different scales is to pre-process the data, i.e., to normalize the data before computing the distance in such a way that we scale the data in a specific range so as to guarantee that they all influence the result in a similar way.

In conclusion, we were able to analyze the design decisions for kNN and also able to describe different similarity and distance measures. We considered Jaccard similarity as the similarity measure for customer system application and explained how we can compare sets and multisets using the similarity measure.

3. Applications of kNN

kNN has been widely used in many applications such as image processing, web pages, recommendation system, medical treatment, text classification and many others. In this chapter, we shall consider how kNN can be applied to customer system and sensor applications. We shall focus on the classification of customer service data and by extension to sensor case using kNN.

3.1 Customer Service Problem

In customer system application, services are rendered to customers in order to satisfy their needs. To satisfy customer needs, the service provider should be able to know the buying habits of their customers or have past information about their customers. Knowing past information of customers will create a stronger experience which will enable them to be able to recommend products that will interest their customers. Recommender systems do rely on customer purchase history to identify products that may interest their customers (Liu and Shih, 2005). In this section, we shall consider how we can apply kNN to customer service problem so as to be able to classify customers into a different segment and also to enable recommendation of products or items to customers.

Recommending products to customers needs to be based on the similarities of customer preferences which is referred to as Collaborative Filtering (CF) (Liu and Shih, 2005). Collaborative filtering (CF) is one of the most successful method which is widely used in customer system application. It is an approach of recommendation system where opinions and tastes of customers are taken into consideration. It is a method of predicting customers' interest by recommending items that were liked by other customers due to their similar taste (Leskovec et al., 2014). In collaborative filtering, we focused on the similarity of customers in terms of rating of products.

Applying kNN to customer service problem, we based classification on the preferences of the nearest neighbours To recommend products to the target customer, the recommendation is based on other customers having similar preferences to the target customer. "Preferences" in this case simply means purchasing habits of customers which might be the selection of products or customer attributes. The suitable similarity measure used to determine the similarities between the preferences of other customers to that of the target customer is Jaccard similarity.

We need to find the similarities between each customer and the target customer where each customer is ranked according to their similarities to the target customer. So, we choose the k most similar among the customers as the k nearest neighbours to the target customer. Then, we base the classification of the target customer on the class of the majority vote of his or her k nearest neighbours among the customers. However, classification based on recommendation is by recommending the highly rated products that are not yet been purchased by the target customer.

3.1.1 Design System for Customer Service.

In this subsection, we want to analyze the customer service problem in mathematical terms and we shall focus on the case of book orders. To solve the customer service problem, we need to specify what should be the input and output of the data. The following are the specifications needed in customer system for classification of book orders. Let

- the database $\mathcal{D}$ of customers be defined as

$$\mathcal{D} : \mathcal{X} \to \Omega \quad \text{such that}$$

$$\mathcal{X} = \{(c_i, b_k) \mid i = 1, \ldots, n,\ k = 1, \ldots, m\} \quad \text{and}$$

$$\Omega = \begin{cases} \Omega_o, & \forall \text{ high-spending customers} \\ \Omega_1, & \forall \text{ low-spending customers,} \end{cases}$$

  where $\mathcal{X} \in \mathbb{R}^{n \times m}$ is the training set containing different books purchased by each customers and Ω is the set of classes of each customers. Let

- c_i and b_k be defined as i^{th} customer and k^{th} book respectively,
- the attributes of i^{th} customer be defined as

$$c_i = \{a_{ij} \mid j = 1, \ldots, \tau\},$$

- c_q be defined as the target customer,
- k be the number of nearest neighbours among other customers to the target customer c_q .

The design decisions of kNN in customer service problem of book orders are given as follows:

- **Similarity:** We need to find the similarity between the target customer and each i^{th} customers. So, we defined the similarity measured from target customer to each i^{th} customers as

$$S(c_q, c_i) : c_q \times c_i \to \mathbb{R}^+.$$

  We considered Jaccard similarity as the similarity measure for the customer service problem in this case which is given as

$$S(c_q, c_i) = \frac{|c_i \cap c_q|}{|c_i \cup c_q|}, \qquad \text{such that } \ 0 \leq S(c_q, c_i) \leq 1.$$

- **Choosing value for k:** We need to choose the k nearest neighbours among the customers c_i that are very similar to the target customer c_q. To choose the nearest neighbours, we can either sort the similarities in descending order such that

$$S_{\text{sort}} = \underset{<}{\text{sort}} \{S(c_q, c_i)\},$$

  and then we choose the first k as the nearest neighbours to the target customer i.e

$$S_k = S_{\text{sort}}[0 : k-1],$$

  or we rather find the k most similar among all the customers as the nearest neighbours such that

$$S_k = \max_{i=1}^{n} \{S(c_q, c_i)\},$$

  where S_k is the set consisting of the highest similarities of k nearest neighbours among the customers to the target customer c_q in the database.

- **Classification:** We need to assign a class to the target customer c_q. To assign a class to c_q, we take the class of the majority vote among the k nearest neighbour customers and assign it as the class of the target customer, i.e.,

  $$c_q \in \text{max } \{\Omega_o,\ \Omega_1\},$$

  which implies that the target customer can either be a high-spending customer or low-spending customer depending on the class which has the highest vote.

  To base classification on recommendation purpose, we consider books purchased by each k nearest neighbours customers. We take the frequency count of each book b_k purchased by all the nearest neighbour customers and recommend the books that are not yet been purchased by c_q.

We shall illustrate how kNN can be applied to customer service problem in the case of book orders and how we can recommend books to customers in the following example.

Toy Example: Consider the database consisting of different books purchased by different customers which we classified into different classes so that we can target advertising to a particular group (those we expect to be high spenders, who will be responsive to the advertisements). In Table 3.1, we represent 1 as the book purchased by a customer and 0 implies otherwise.

Customers	Book 1	Book 2	Book 3	Book 4	Book 5	Class
A	1	0	1	1	0	Ω_o
B	0	1	1	0	0	Ω_o
C	1	1	0	1	0	Ω_1
D	1	0	1	1	1	Ω_1
E	1	1	0	0	1	?

Table 3.1: Data of customers and items purchased

From Table given above, we choose customer E as the target customer (query point) and customers $A - D$ as the training data. In this case, the class of the target customer is unknown, so we need to assign class and as well recommend books to the target customer.

To apply kNN to this case, we need to find the similarity between the target customer and each of the other customers by using Jaccard similarity as the similarity measure and we obtained:

$$J_{(A,E)} = \frac{|\{\text{Book } 1,\ \text{Book } 3,\ \text{Book } 4\} \cap \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|}{|\{\text{Book } 1,\ \text{Book } 3,\ \text{Book } 4\} \cup \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|} = \frac{1}{5} = 0.2.$$

$$J_{(B,E)} = \frac{|\{\text{Book } 2,\ \text{Book } 3\} \cap \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|}{|\{\text{Book } 2,\ \text{Book } 3\} \cup \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|} = \frac{1}{4} = 0.25.$$

$$J_{(C,E)} = \frac{|\{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 4\} \cap \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|}{|\{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 4\} \cup \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|} = \frac{2}{4} = \frac{1}{2} = 0.5.$$

$$J_{(D,E)} = \frac{|\{\text{Book } 1,\ \text{Book } 3,\ \text{Book } 4,\ \text{Book } 5\} \cap \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|}{|\{\text{Book } 1,\ \text{Book } 3,\ \text{Book } 4,\ \text{Book } 5\} \cup \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}|} = \frac{2}{5} = 0.4.$$

We choose the k most similar to the target customer E as its nearest neighbours Suppose we choose $k = 3$ in this case, implies that

$$S_k(c_q, c_i) = \{0.5, 0.4, 0.25\},$$

which are corresponding to the classes $\{\Omega_1,\ \Omega_1,\ \Omega_o\}$. Then, we base our classification on the class of the majority vote which corresponds to class Ω_1. Thus, the class of customer E is Ω_1.

To recommend books to customer E, we based the recommendation on the books purchased by the k nearest neighbour customers that has not yet been purchased by customer E, i.e.,

$$\begin{aligned}
&\text{Customer B} = \{\text{Book } 2,\ \text{Book } 3\}\\
&\text{Customer C} = \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 4\}\\
&\text{Customer D} = \{\text{Book } 1,\ \text{Book } 3,\ \text{Book } 4,\ \text{Book } 5\}\\
&\text{Customer E} = \{\text{Book } 1,\ \text{Book } 2,\ \text{Book } 5\}
\end{aligned}$$

The frequency count of books purchased by the neighbours are; Book 1 $\rightarrow$ 2, Book 2 $\rightarrow$ 2, Book 3 $\rightarrow$ 2, Book 4 $\rightarrow$ 2, Book 5 $\rightarrow$ 1. The books that are not yet been purchased by customer E are Book 3 and Book 4. So, in this case we can recommend Book 3 and Book 4 to customer E which might be of interest due to the similar taste of his or her neighbours

Other similarity measure that could be of use in this case is Cosine similarity. Using the same above example to compute the cosine similarity, each set of books purchased by each customer is represented as binary vectors, so we have

$$\begin{aligned}
A = [1,0,1,1,0],\ \ B = [0,1,1,0,0],\ \ C = [1,1,0,1,0],\\
D = [1,0,1,1,1],\ \ E = [1,1,0,0,1].
\end{aligned}$$

Then, the cosine similarity is computed as

$$\cos(\theta)_{A,E} = \frac{1\cdot 1 + 0\cdot 1 + 1\cdot 0 + 1\cdot 0 + 0\cdot 1}{(\sqrt{1^2+1^2+1^2})(\sqrt{1^2+1^2+1^2})} = \frac{1}{\sqrt{3}\sqrt{3}} = \frac{1}{3} = 0.3333.$$

$$\cos(\theta)_{B,E} = \frac{0\cdot 1 + 1\cdot 1 + 1\cdot 0 + 0\cdot 0 + 0\cdot 1}{(\sqrt{1^2+1^2})(\sqrt{1^2+1^2+1^2})} = \frac{1}{\sqrt{2}\sqrt{3}} = \frac{1}{\sqrt{6}} = 0.4082.$$

$$\cos(\theta)_{C,E} = \frac{1\cdot 1 + 1\cdot 1 + 0\cdot 0 + 1\cdot 0 + 0\cdot 1}{(\sqrt{1^2+1^2+1^2})(\sqrt{1^2+1^2+1^2})} = \frac{2}{\sqrt{3}\sqrt{3}} = \frac{2}{3} = 0.6667.$$

$$\cos(\theta)_{D,E} = \frac{1\cdot 1 + 0\cdot 1 + 1\cdot 0 + 1\cdot 0 + 1\cdot 1}{(\sqrt{1^2+1^2+1^2+1^2})(\sqrt{1^2+1^2+1^2})} = \frac{2}{\sqrt{4}\sqrt{3}} = \frac{1}{\sqrt{3}} = 0.5774.$$

The more the cosine angle approaches 1, the more the similarity. In this case, suppose we choose $k = 3$ as the number of nearest neighbours to customer E, then we have the nearest neighbours to be customer C, customer D and customer B respectively which corresponds to the same nearest neighbours using Jaccard similarity.

3.2 Extension to Sensor Application

In this section, we shall extend from customer system application to sensor application. We shall focus on how kNN can be applied to sensor applications by considering an abstract case of sensors reporting on features found in an environment and suitable similarity measure to use.

Sensors are widely used for different purposes in various applications such as smart phones, cars, air planes, manufacturing and machinery, hospitals, work places, education and many other aspects of our day-to-day activities. Sensors are devices that receive and responds to a signal or stimulus (Kalantar-zadeh and Fry, 2008). They are used in detecting or monitoring changes in a physical environment in which they are capable of providing raw data of the information received (Kalantar-zadeh and Fry, 2008). A sensor network consists of multiple detection channels which are called the sensor nodes.

Consider an abstract case of sensors reporting on features found in an environment. Assuming certain combination of features indicate the healthy state of the environment being monitored and another set of features indicate an unhealthy state of the environment. kNN is applicable to this situation by making use of the features of its nearest neighbours which implies that the classification is based on the features of its nearest neighbours

To apply kNN to sensor application, a distance measure is needed to calculate how close or far the sensor nodes are to the query point (Han et al., 2015). The suitable distance measure which can be used to determine the distance between the sensor nodes and the query point is Euclidean distance. We need to find the distance between the query point and each of the sensor nodes and then choose the k smallest distances as the nearest neighbours to the query point (Han et al., 2015). Then, the classification of the query point is based on the class of the majority vote of its nearest neighbours

Using the same approach of customer service, we consider features in this case rather than the case of book orders. The notations listed below will be useful in the following properties of how to apply kNN in sensors application. Let

- $S = \{s_i \mid i = 1, \ldots, n\}$ be a sensor network consisting of sensor nodes;
- $s_i \in S$ and $q \in S$ be the i^{th} sensor nodes and the query point respectively;
- the classes f of each sensor nodes s_i be defined as

$$f = \begin{cases} f_o, & \forall \text{ healthy state} \\ f_1, & \forall \text{ unhealthy state,} \end{cases}$$

  where $f \in s_i$;
- k be the number of nearest neighbours in the sensor nodes to the query point q.

Properties: Having stated the problem of the sensor case, we shall consider how kNN can be applied to this problem by detecting the features of the query point using its nearest neighbours Below are the following properties to consider before decisions can be made for the query point.

- **Distance:** We need to find the distance between each of the sensor nodes s_i and the query point q such that

$$d(s_i, q) : s_i \times q \to \mathbb{R}^+.$$

We considered Euclidean distance as the distance measure between each sensor nodes s_i and the query point which is given as

$$d(s_i, q) = \sqrt{\sum_{i=1}^{n}(s_i - q)^2} \quad \text{such that} \quad d(s_i, q) \in \mathbb{R}^+.$$

- **Choosing value for k:** We are required to find the k smallest distances among the sensor nodes to the query point such that

$$d_k = \min_{i=1}^{n} \{d(s_i, q)\},$$

where d_k is set consisting of the k smallest distances of the nearest neighbours among the sensor nodes to the query point.

- **Classification:** We are required to find the features of the query point, so the classification is based on the class of the majority vote of its nearest neighbours, i.e.,

$$q \in \text{arg max} \{f_o,\ f_1\},$$

which implies that the features of the query point can either be healthy state or unhealthy state of the environment.

In the case where we have extracted features, we might also be able to apply kNN in a similar way to customer service and use Jaccard similarity.

In this chapter, we were able to apply kNN to customer system application and by extension to sensor application. We considered the case of book orders for customer service data and apply the design decisions of kNN to classify and as well recommend products to the target customers. Using the same customer service approach, we considered an abstract case of sensors reporting on features found in an environment and we applied kNN to detect the features of the query location.

4. Results and Discussion

According to the aim of this project work, we design and implement a simple kNN system for customer data and show the effect of different design decisions on the output of the system. We create a small example set of "customer orders" as the dataset and illustrate the difference in the results using different similarity measures and different values of k. In this chapter, we shall apply kNN on Iris dataset and evaluate its performance on the data. We shall also give the result of the implementation of kNN on customer service data and then give more explanation on the result.

4.1 Example of kNN with Iris Data

In this section, we applied kNN on an Iris data using Euclidean distance as the distance measure which was implemented on Waikato Environment for Knowledge Analysis (WEKA) open source (Markov and Russell, 2006). In the iris dataset, there are 150 instances with three different classes namely: Iris-setosa, Iris-versicolor, and Iris-virginica respectively. We can access the performance of kNN on the dataset by considering the rate of its accuracy, false positives and false negatives just as mentioned in section 1.2.1 which is defined as follows:

- **Accuracy** is defined as the ratio of the number of correct predictions to the total number of the instances, i.e.,
$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of instances}},$$
- **False Positives (FP)** is the number of instances predicted positive but which are actually negative,
- **False Negatives (FN)** is the number of instances predicted negative but which are actually positive.

4.1.1 Result of kNN with Iris Data.

Below is the confusion matrix obtained using WEKA showing the visualization of the performance of kNN on iris dataset which was cross-validated with 10-folds.

```
=== Confusion Matrix ===

a  b  c   <-- classified as
50  0  0 |  a = Iris-setosa
0 47  3 |  b = Iris-versicolor
0  4 46 |  c = Iris-virginica
```

From the confusion matrix shown above, the sum of numbers in the diagonal are the correct predictions which are 143 instances while 7 instances were incorrectly predicted. Therefore, the accuracy rate of kNN on the data was given to be 95.3333% while the misclassification rate was 4.6667% which is the rate of the incorrect predictions and it implies that both FP and FN are relatively small. Hence, the predictions of kNN made on the iris data are 95% accurate.

4.2 Customer Services Example

As mentioned in the previous section, we talked about the result of how kNN classification was applicable to the iris dataset. In this section, we shall show the result of kNN being applied to the customer service data. We create a binary dataset of ten customers with ten different books $A - J$ in which customers are classified to their respective classes (Bashirat, 2017). In the data, we classified customers into two classes which are either high-spending customers or low-spending customers by means of knowing which customers we can advertise products to. We represent 1 as a particular book purchased by a customer and 0 implies otherwise. We consider the list of all books purchased by each customer as a training set and also consider the set of books $\{A, C, F, I, J\}$ or $\{1, 0, 1, 0, 0, 1, 0, 0, 1, 1\}$ as the test instance we want to classify.

4.2.1 Result of kNN with Customer Service Data .

Customers	Jaccard similarity	Class	Cosine similarity	Class	Rank (according to similarity)
0	0.2500	High	0.4000	High	8
1	0.2857	Low	0.4472	Low	7
2	0.1111	High	0.2000	High	10
3	0.5714	High	0.7303	High	1
4	0.3333	High	0.5071	High	5
5	0.3333	High	0.5071	High	6
6	0.5714	High	0.7303	High	2
7	0.5000	Low	0.6708	Low	3
8	0.5000	Low	0.6708	Low	4
9	0.1250	Low	0.4000	Low	9

Table 4.1: Results of the Jaccard and Cosine similarities between each customer and the target customer with respect to their classes.

4.3 Discussion of Customer Service Data

The code we used for kNN was implemented by Dr. Jason Brownlee (Brownlee) but which was modified according to the project work. Some of the modifications include: changing the code to use any similarity or distance measure that fit in the features of the data; getting the result as "ties" if the number of class votes are the same and; returning the result in which the neighbours votes for only one class.

We considered two similarity measures in this case which are the Jaccard similarity and Cosine similarity. To use Jaccard similarity for this example, we change the binary dataset to list of set of books purchased by each customer while we change the data to list of binary vectors in order to compute for Cosine similarity.

According to Table 4.1, it is shown that the results of Jaccard and Cosine similarities are different but their classes are the same according to their similarities which corresponds to each customer. Another design decisions of kNN is analyzing different values of k on the result using both similarity measures.

Suppose we choose $k = 2$, i.e., choosing the first two nearest neighbours among other customers which are most similar to the target customer. It is shown that we will have the result to be "High". Let us

consider $k = 3$, we will end up having two classes in this case but the class of the majority votes is "High". Suppose we choose $k = 4$, we will end up falling into "ties" which implies that we are having the same number of votes for the two classes. Therefore, we can have the result to be "High" or "Low". To solve the problem of "ties", we can choose randomly by considering the FP and FN of this problem.

False positives (FP) in this customer service problem are the low-spending customers that we incorrectly classify as high-spending customers. This becomes a problem because we will continuing wasting money on them by advertising products that they will not be able to afford. False negatives (FN) are the high-spending customers that we incorrectly classify as low-spending customers. This is really a serious problem because we will continuing to lose customers and as well lose profit by not advertising products to those who can afford to buy the products.

For the "tie" problem, we might decide to predict the class of the target customer as "High" so as to reduce the effect of the FN by advertising products to the customer in which he or she might be interested in. Moreover, from Table 4.1, it was shown that the class "High" is more similar compared to class "Low". Thus, we predict the class of the target customer as "High". Generally, from Table 4.1, it was shown that the best value for k in order to predict the class of the target customer is to choose odd value, i.e., either choosing $k = 1, 3, 5, 7$ or 9.

How to choose whether to use Jaccard or Cosine similarity if implementing a system depends on the data given. Jaccard similarity is naturally represented as sets which works well in customer system for comparing set of items while Cosine similarity works well for sparse vectors. Moreover, Jaccard similarity can be estimated in large-scale systems by means of minhashing method but Cosine similarity requires lot of time to compute for large data.

In conclusion, we were able to applied kNN to iris dataset using Euclidean distance as the distance measure and also able to produce the results using WEKA. We gave the results of the implementation of kNN on the customer service data we created using Jaccard and Cosine similarities as the similarity measures. We discovered that the two similarity measures gave different results but the same classes according to their similarities.

5. Conclusions

In this project, an approach of machine learning using kNN was made in customer system and sensor applications. We considered how kNN can be applied to customers data records in customer system application and by extension to the case of sensor application.

We make use of kNN in customer system in order to solve the customer service problem such as products that should be recommended to customers and how to classify customers into different segment in order to know which customers we can advertise products to. kNN is also useful in sensor application for detecting features of a new observation by making use of the features of the nearest neighbours among the sensor nodes which are very close to the new observation. The main aim in this project work is to show the design decisions, such as similarity measure, which are important for kNN and how it can be applied to these two applications.

We have shown how kNN works and discussed the design decisions that should be made for kNN. We considered different similarity and distance measures and gave more detailed explanation on how some of these measures are useful in customer system and sensor applications. We considered Jaccard similarity as the similarity measure for customer system application and Euclidean distance as the distance measure in sensor application. We also considered what type of application some of the other similarity measures could be used for.

We designed a system for customer service problem on the case of book orders and applied the design decisions of kNN to classify and as well recommend books to target customers. An abstract case of sensors reporting on features found in an environment was also considered. By extension, we use a similar approach of customer service on the sensor case and applied the design decisions of kNN to find the features of the query location.

We applied kNN to iris dataset on WEKA and then evaluate the results. We designed and implemented kNN for small example set of customer orders that we constructed and showed the effects of different measures and different decision combination schemes on the results. We considered Jaccard similarity and Cosine similarity as the similarity measures and showed different choice of k values.

Further work

The work we might later consider are as follows: applying Jaccard similarity to large scale system using minhashing techniques in customer service data; using other machine learning algorithms in customer system and sensor applications; or using other features of customers by matching them on the rating system or considering the contents of the books rather than set of books.

Acknowledgements

All praises and adorations be unto Almighty Allah who has kept and strengthen me throughout my program at African Institute for Mathematical Sciences (AIMS). Forever will I be grateful to my loving, wonderful and caring parents for their continuous prayers and support towards my success in life. May Almighty Allah guide and protect them and make me a wonderful daughter that will clear their tears and bring more joy into their lives.

I am highly indebted and duly thankful to my project supervisor, Dr Simukai W. Utete for her motherly care, guidance and words of wisdom throughout my work. She always showed me the right direction during my project work. She is a vey nice woman and explains well whenever I ask her any questions concerning my project. May Almighty God guide, protect and bless you abundantly in all your endeavours.

My gratitude also goes to Dr Yae Olatoundji Gaba for his help towards the explanation of different similarity measures. He makes me understand the difference between distance and similarity measures and where it can be applied to. I can not thank you enough but I pray God reward you abundantly in all your work.

I also use this opportunity to thank my tutor Eyaya Birara Eneyew for his help towards the programming of my code. He has always been there for me whenever am stuck with my program. I really appreciate your kindness and sincerity, success shall be yours by God grace.

My profound gratitude goes to all academic and non-academic staff for their wonderful support ever since I was at AIMS. They make me feel at home, may God be with each and everyone of you. Lastly, I am very grateful to my family, friends and peers for their constant prayers and support throughout my stay at AIMS. I do appreciate all my friends and colleagues at African Institute for Mathematical Sciences (AIMS) for their tireless encouragement and invaluable pieces of advise and counsel, forever will I miss you all.

References

A. Bashirat. Machine Learning using k-Nearest Neighbors in Customer System and Sensor Applications, 2017. URL https://github.com/Bashirat/AIMS_essay_code.

J. Brownlee. Tutorial to Implement k-Nearest Neighbors in Python from Scratch. URL http://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/.

M. M. Deza and E. Deza. Encyclopedia of Distances. In *Encyclopedia of Distances*, pages 1–583. Springer, 2009.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2012.

P. Flach. *Machine learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

J. Gou, L. Du, Y. Zhang, T. Xiong, et al. A New Distance-Weighted k-Nearest Neighbor Classifier. *J. Inf. Comput. Sci*, 9(6):1429–1436, 2012.

Y. Han, K. Park, J. Hong, N. Ulamin, and Y.-K. Lee. Distance-Constraint k-Nearest Neighbor Searching in Mobile Sensor Networks. *Sensors*, 15(8):18209–18228, 2015.

D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing Images using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

K. Kalantar-zadeh and B. Fry. Sensor Characteristics and Physical Effects. *Nanotechnology-Enabled Sensors*, pages 13–62, 2008.

J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

D.-R. Liu and Y.-Y. Shih. Hybrid Approaches to Product Recommendation Based on Customer Lifetime Value and Purchase Preferences. *Journal of Systems and Software*, 77(2):181–191, 2005.

Z. Markov and I. Russell. An Introduction to the WEKA Data Mining System. *ACM SIGCSE Bulletin*, 38(3):367–368, 2006.

T. McMullen. It probably works. *Queue*, 13(8):15, 2015.

D. Widdows. *Geometry and Meaning*, volume 773. CSLI publications Stanford, 2004.

Wikipedia. Jaro–winkler distance. Wikipedia, the Free Encyclopedia, https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance, Accessed April 2017.