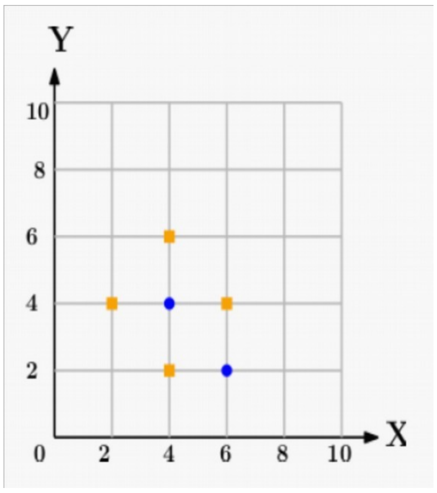
ASSIGNMENT NO MLA3

Roll - 41239

TITLE : Assignment on k-NN Classification

PROBLEM STATEMENT : In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find the class of the point (6,6). Extend the same example for Distance-Weighted k-NN and Locally weighted Averaging.



OBJECTIVE :
To understand how kNN algorithm works on the given dataset
To implement kNN classification algorithm.

S/W and H/W : Python,Colab,Core i5,8 GB RAM,1 TB HDD,Keyboard,mouse

Theory :

1. Supervised learning involves selecting a heuristic function using distinctly known feature/s to map labels. By observing the given data we can conclude that the input features and output labels are distinctly known, hence satisfying the rules necessary for the use of supervised learning.

Programmer's Perspective:

Let S be the programmer's perspective of the linear regression, such that S = {s, e, X, Y, *fme*, *ffi Memshared*| Φ}

s = Start State

- {(X_i, y_i): i ∈ N, X_i = ($X0_i$,$X1_i$), $X0_i \in$ R, $X1_i \in$ R, $y_i \in$ {orange, blue} }
- $X0_i$: x-coordinate
- $X1_i$: y-coordinate
- k = { number of neighbors }, k ∈ [3,length of the dataset]

e = { Mapping the given point with the appropriate class }

X = {X,y}

- X: Feature values
    - x: { x-coordinate | $x \in \mathrm{R}$ }
    - y: { y-coordinate | $y \in \mathrm{R}$ }
    - k: { number of neighbors | k $\in$ [3,length of the dataset] }
- y: Output values = { y_i | y_i is the predicted class for point X_i, $y_i \in$ {orange, blue}, $i \in \mathrm{N}$ },

f_{me} = Function f_{me} can be defined as $f_{me}: X \rightarrow y$
f_{me}: { KNearestNeighbors | X = (df, k, input_val, weight, metric), y=prediction}

- df : {(X,y)}
- k : { number of neighbors }, k $\in$ [3,length of the dataset]
- input_val : { x, y | x=x-coordinate, y=y-coordinate }
- weight : { type of k-NN }, weight $\in$ {distance, uniform}
- metric : { method for distance calculation }, metric $\in$ {manhatten, euclidean}
- y = prediction: { output label | y $\in$ {blue, orange} }
- Method to implement k nearest neighbors algorithm

f_{fi} = friend functions: {f_1, f_2, f_3} For ever function f_i in f_{fi}:

$$f_i: X \rightarrow y$$

- f1: { euclidean_distance | X=(point_a, point_b), y=distance }
    - point_a, point_b : { coordinates of a point in the dataset }
    - y = distance : { Euclidean distance between two points }, distance $\in$ [0, ∞)
    - Method to calculate euclidian distance between two points point_a and point_b

- f2: { manhattan_distance | X=(point_a, point_b), y=distance }
    - point_a, point_b : { coordinates of a point in the dataset }
    - y = distance : { Manhattan distance between two points }, distance $\in$ [0, ∞)
    - Method to calculate manhattan distance between two points point_a and point_b

- f3: { weighted_prediction | X = nearest_neighbors, y = prediction }
    - nearest_neighbors: { [(X_{00},X_{10}), (X_{01},X_{11}),... (X_{0k-1},X_{1k-1})] | X_{0i} = x-coordinate, X_{1i} = y-coordinate }
    - y = prediction: { output label | y $\in$ {blue, orange} }
    - Method to predict class based on the weighted frequency of nearest neighbors

Mem_{shared} = { Shared resource for all threads to store prediction for given set of input }

Why k-NN?

- A notable feature of this approach is its laziness i.e calculations are only done during the prediction phase, when a test sample needs to be classified. No model is constructed from the training examples beforehand.
- There exist many important theorems claiming that, on "endless" datasets, it is the optimal method of classification.

- Good interpretability as it answers the question "why was my example given class X?” with “because similar items are labeled with X". For example, consider a model assessing the risk involved with a loan. Here, a customer is judged to be high risk because 8 out of 10 customers who have been previously evaluated and were most similar to them in terms of X, Y, and Z, were found to be high risk. By observing the nearest neighbors, you can see objects similar to the example that are labeled as X, and you can decide whether the prediction makes sense.

k-NN vs Linear Algorithms

- A kNN classifier is able to recover unstructured partitions of the space, as opposed to, say, a linear classifier that requires a linear separation between the classes. It can also adapt to different densities in the space, making it more robust than methods such as support vector machine (SVM) classification with radial basis function (RBF) kernel. The following two examples of 2D data illustrate different partitions of the space imposed by labeled data and the prediction of a kNN model on that space.

Choosing appropriate k value is very critical in the k-NN algorithm

- A small value of k means that noise will have a higher influence on the result.
- Larger values of K will have smoother decision boundaries which mean lower variance but increased bias, it is also computationally expensive and defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes).
- A simple approach to select k is set **k = sqrt(N)**, where N is the size of the dataset
- Another way to choose k is though cross-validation, by selecting the k value which performs best on the validation set and minimizes the validation error.
- Generally **odd values of k** are preferred in order to avoid a split vote.

Disadvantages of k-NN

- **Curse of dimensionality**: In k-NN the distance between instances is calculated based on all attributes of the instance, as a result difficulty may arise when many irrelevant attributes are present in the dataset. Because, the similarity metric used by k-NN depends on all the attributes in the dataset and it can be easily misled if the distance between neighbors is dominated by the irrelevant attributes. This problem is referred to as the curse of dimensionality. Nearest-neighbor approaches are especially sensitive to this problem.
- Computation time: As k-NN algorithm delays all processing until a new query is received, significant computation can be required to process each new query.

Conclusion:

K Nearest Neighbor classification algorithm was used for classifying the points given in the dataset.