# K- Nearest Neighbours

## (1) Proximity :

Two observations :

| | $x_1$ | $x_2$ |
|---|---|---|
| | $x_{11}$ | $x_{21}$ |
| | $x_{12}$ | $x_{22}$ |

The distance between these two observations can be quantified using Euclidean distance :

$$d = \sqrt{(x_{11} - x_{12})^2 + (x_{21} - x_{22})^2}$$

Euclidean distance
$(x_1, y_1) \leftrightarrow (x_2, y_2)$

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Euclidean distance (in vectorized notation)

Let $\underset{\sim}{x_1} = (x_{11}, x_{21})^T$ and $\underset{\sim}{x_2} = (x_{12}, x_{22})^T$ be two vectors.

$\underset{\sim}{d} = \underset{\sim}{x_1} - \underset{\sim}{x_2}$ , the diff. of the vectors $\underset{\sim}{x_1}$ & $\underset{\sim}{x_2}$

$= ((x_{11} - x_{12}), (x_{21} - x_{22}))^T$

the euclidean distance between the $\underset{\sim}{x_1}$ & $\underset{\sim}{x_2}$ can be written as

$$\text{dist} = \sqrt{\underset{\sim}{d}^T \underset{\sim}{d}}$$

$$= \sqrt{(x_{11} - x_{12},\ x_{21} - x_{22}) \begin{pmatrix} x_{11} - x_{12} \\ x_{21} - x_{22} \end{pmatrix}}$$

$$= \sqrt{(x_{11} - x_{12})^2 + (x_{21} - x_{22})^2}$$

H.W: design a function the calc. euclidean distance between two vectors using (1) for loop (2) vectorized notation.

## Euclidian distance (in general)

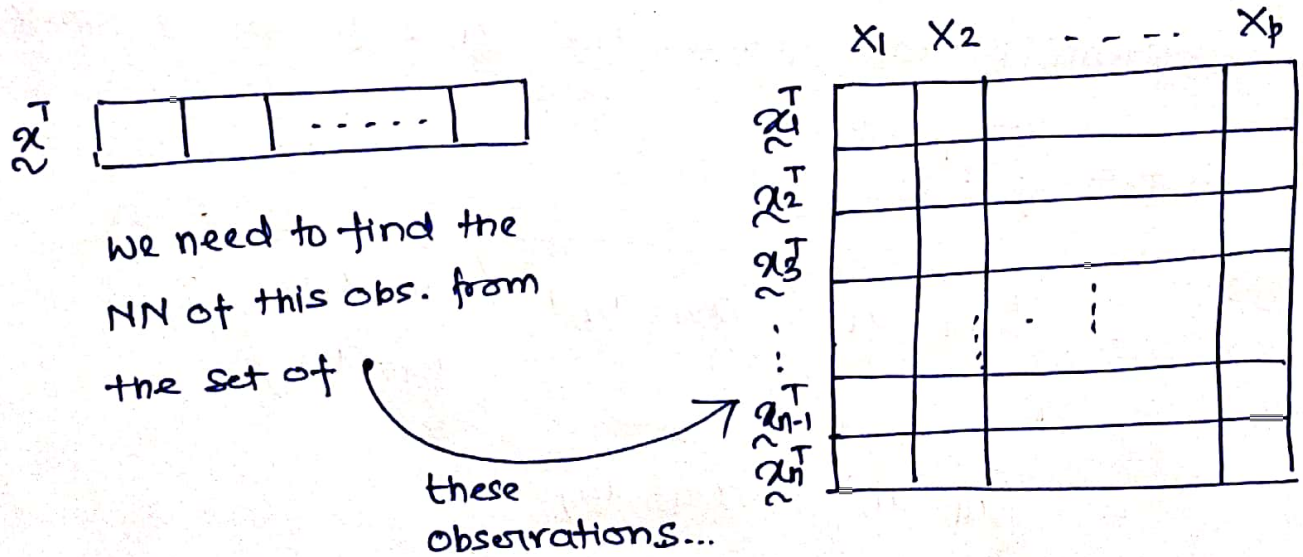
| | $X_1$ | $X_2$ | ..... | $X_p$ |
|---|---|---|---|---|
| $\underset{\sim}{x}_1^T$ | $x_{11}$ | $x_{21}$ | --- | $x_{p1}$ |
| $\underset{\sim}{x}_2^T$ | $x_{12}$ | $x_{22}$ | --- | $x_{p2}$ |
| ..... | | | | |

The euclidean distance between $\underset{\sim}{x}_1$ and $\underset{\sim}{x}_2$ can be written as:

$$\text{dist} = \sqrt{(x_{11}-x_{12})^2 + (x_{21}-x_{22})^2 + \cdots + (x_{p1}-x_{p2})^2}$$

$$= \sqrt{(\underset{\sim}{x}_1 - \underset{\sim}{x}_2)^T (\underset{\sim}{x}_1 - \underset{\sim}{x}_2)}$$

## (2) finding the nearest neighbour :



## 1-NN

**Step 1**: calculate the dist. of $\underset{\sim}{x}$ from $\underset{\sim}{x_i}$, for all $i = 1(1)n$

**Step 2**: NN = the obs. that corresponds to the shortest distance.

## Simplest approach (1-NN):

```
1: min_dist = inf
2: for i in range(1, len(train_data)):
3:     d = dist. of the test obs. from the ith obs. in train_data
4:     if d < min_dist:
5:         min_dist = d    #updata min_dist
6:         nearest_obs_index = i
7:     end if
8: end for
9: nearest_obs = train[i, ]
```

## (3) 1-NN algorithm for classification :

Training data :

| | $x_1$ | $x_2$ | --- | $x_p$ | $Y$ |
|---|---|---|---|---|---|
| $\underset{\sim}{x}_1^T$ | $x_{11}$ | $x_{21}$ | --- | $x_{p1}$ | $y_1$ |
| $\underset{\sim}{x}_2^T$ | $x_{12}$ | $x_{22}$ | --- | $x_{p2}$ | $y_2$ |
| ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ |
| $\underset{\sim}{x}_{n_1}^T$ | $x_{1n_1}$ | $x_{2n_1}$ | --- | $x_{pn_2}$ | $y_n$ |

Test data :

| | $x_1$ | $x_2$ | --- | $x_p$ |
|---|---|---|---|---|
| $\underset{\sim}{x}_1^{*T}$ | $x_{11}^*$ | $x_{21}$ | --- | $x_{p1}^*$ |
| $\underset{\sim}{x}_2^{*T}$ | $x_{12}^*$ | $x_{22}$ | --- | $x_{p2}^*$ |
| ⋮ | ⋮ | ⋮ | | ⋮ |
| $\underset{\sim}{x}_{n_2}^{*T}$ | $x_{1n_2}^*$ | $x_{2n_2}$ | --- | $x_{pn_2}^*$ |

$n_1$ = no. of obs. in training data

$n_2$ = no. of obs. in test data

- function 1: function to calculate the eucledian distance.

  distance $(x, y)$ → returns euclidean dist. between the vectors $\underset{\sim}{x}$ and $\underset{\sim}{y}$.

- function 2: function to calculate the (or get the) NN for one test observation.

  NN (train_X, train_y, $x^*$) → finds the nearest obs. to $x^*$ in the training data and get the value of Y for that obs.

- function 3:

  1NN (train_X, test, train y) → returns the values of y for all the obs. in the test data corresp. to the nearest neighbors in the train data.

## fn 1 : Eucledian dist

```
1: distance (x, y) :
2:     d = sqrt((x - y)^T (x - y))
3:     return (d)
```

## fn 2 : NN of $x^*$

```
1: NN (train-X , train-y , x*) :
2:     for observations in the training data :
3:         d = distance (x*, observation)
4:         if d < min-dist :      # min-dist = ∞ (initially)
5:             min-dist = d       # update min-dist.
6:             nearest-obs-index = index of observation
7:         end if
8:     end for
9:     y-hat* = train-y [nearest-obs-index]
10:    return (y-hat*)
```

□ fn 3 : 1 NN algo

```
1: 1NN (train-X, train-y, test_X):
2:     for each observation x* in test data :
3:         y* = NN (train-X, train-y, x*)
4:         append y* in a list Y    # initially Y is an empty list.
5:     end for
6:     return (Y)
```

Remark: This KNN algorithm works for either a problem of regression or classification.