

## K-Nearest Neighbors

### ① Proximity:

$$\text{Two observations: } \begin{array}{cc} x_1 & x_2 \\ (x_{11}, x_{21}) \\ (x_{12}, x_{22}) \end{array}$$

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  $\underline{x}_1 = (x_{11}, x_{21})^T$  and  $\underline{x}_2 = (x_{12}, x_{22})^T$   
be two vectors.

$$\begin{aligned} \underline{d} &= \underline{x}_1 - \underline{x}_2, \text{ the diff. of the vectors } \underline{x}_1 \text{ \& } \underline{x}_2 \\ &= (x_{11} - x_{12}, x_{21} - x_{22})^T \end{aligned}$$

the euclidean distance between the  $\underline{x}_1$  &  $\underline{x}_2$  can be written as

$$\begin{aligned} \text{dist} &= \sqrt{\underline{d}^T \underline{d}} \\ &= \sqrt{(x_{11} - x_{12}, x_{21} - x_{22}) \begin{pmatrix} x_{11} - x_{12} \\ x_{21} - x_{22} \end{pmatrix}} \\ &= \sqrt{(x_{11} - x_{12})^2 + (x_{21} - x_{22})^2} \end{aligned}$$

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

①

## Euclidean distance (in general)

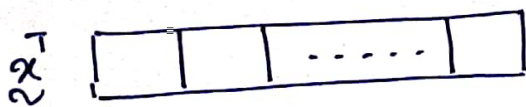
	$x_1$	$x_2$	...	$x_p$
$\tilde{x}_1^T$	$x_{11}$	$x_{21}$	...	$x_{p1}$
$\tilde{x}_2^T$	$x_{12}$	$x_{22}$	...	$x_{p2}$
...	...	...	...	...

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

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

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

## ② finding the nearest neighbour:



We need to find the NN of this obs. from the set of

these observations...

	$x_1$	$x_2$	...	$x_p$
$\tilde{x}_1^T$				
$\tilde{x}_2^T$				
$\tilde{x}_3^T$				
...				
$\tilde{x}_{n-1}^T$				
$\tilde{x}_n^T$				



## 1-NN

Step 1: calculate the dist. of  $x$  from  $x_i$ , 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  $i$ th obs. in train-data

4: if  $d < \text{min-dist}$ :

5:     min-dist =  $d$      #update 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 :

	$x_1$	$x_2$	...	$x_p$	$y$
$\tilde{x}_1^T$	$x_{11}$	$x_{21}$	...	$x_{p1}$	$y_1$
$\tilde{x}_2^T$	$x_{12}$	$x_{22}$	...	$x_{p2}$	$y_2$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$\tilde{x}_{n_1}^T$	$x_{1n_1}$	$x_{2n_1}$	...	$x_{pn_1}$	$y_{n_1}$

Test data :

	$x_1$	$x_2$	...	$x_p$
$\tilde{x}_1^{*T}$	$x_{11}^*$	$x_{21}$	...	$x_{p1}^*$
$\tilde{x}_2^{*T}$	$x_{12}^*$	$x_{22}$	...	$x_{p2}^*$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\tilde{x}_{n_2}^{*T}$	$x_{1n_2}^*$	$x_{2n_2}$	...	$x_{pn_2}^*$

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

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

□ function 1: function to calculate the euclidean distance.

$\text{distance}(\tilde{x}, y) \rightarrow$  returns euclidean dist. between the vectors  $\tilde{x}$  and  $y$ .

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

$\text{NN}(\text{train-X}, \text{train-Y}, \tilde{x}^*) \rightarrow$  finds the nearest obs. to  $\tilde{x}^*$  in the training data and get the value of  $y$  for that obs.

□ function 3:

$\text{1NN}(\text{train-X}, \text{test}, \text{train-Y}) \rightarrow$  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:  $d = \text{distance}(x^*, \text{observation})$

4: if  $d < \text{min-dist}$  : # min-dist =  $\infty$  (initially)

5: min-dist =  $d$  # update min-dist.

6: nearest-obs-index = index of observation

7: end if

8: end for

9:  $y\text{-hat}^* = \text{train-y}[\text{nearest-obs-index}]$

10: return  $(y\text{-hat}^*)$

□ fn.3 : 1NN algo

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

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