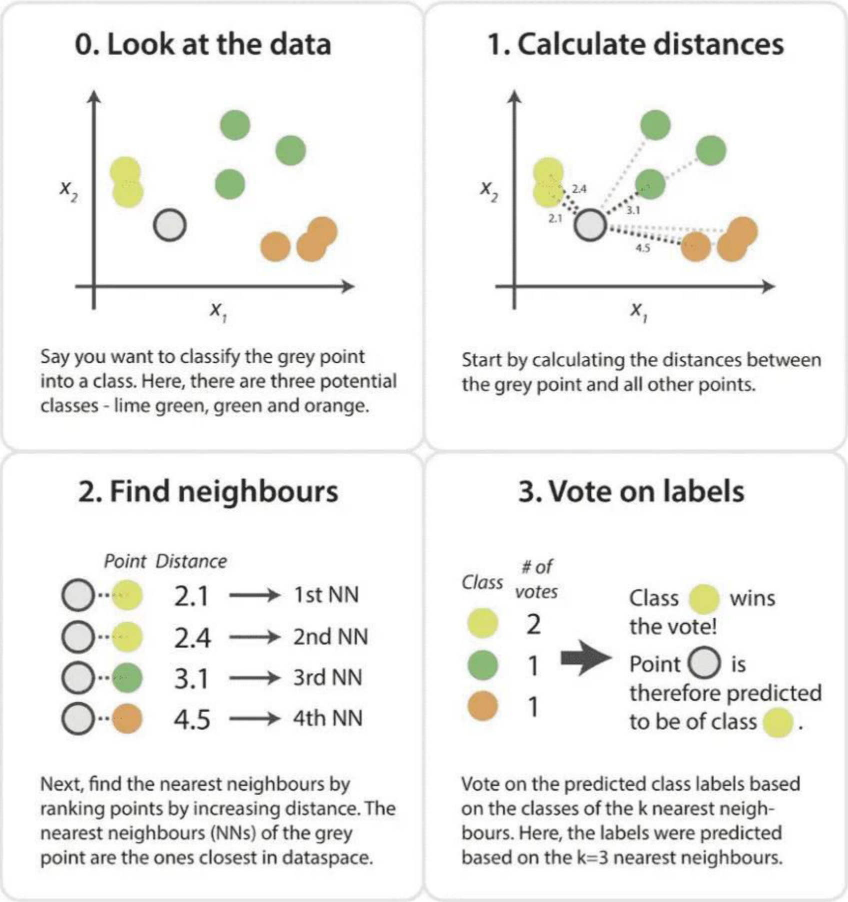
K-nearest neighbors (KNN)

- K-nearest neighbors (KNN) is a type of supervised learning algorithm used for both regression and classification.
- KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closet to the test data.
- The KNN algorithm calculates the probability of the test data belonging to the classes of ‘K’ training data and class holds the highest probability will be selected.
- In the case of regression, the value is the mean of the ‘K’ selected training points.

K-nearest neighbors (KNN)



KNN: distance metrics

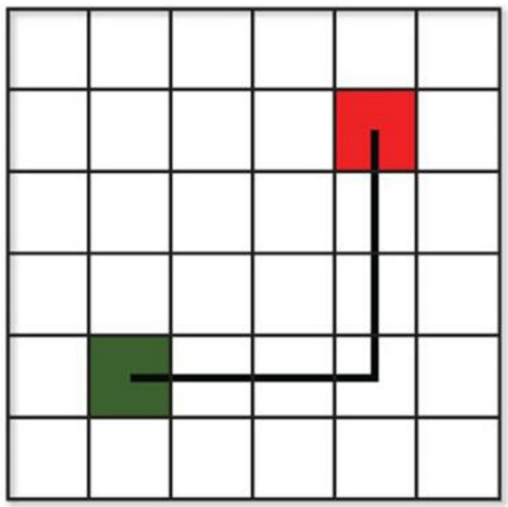
- There are several distance measures that you can choose from, to determine which data points are closest to a given query point
- **Euclidean distance (p=2):** This is the most used distance measure, and it is limited to real-valued vectors. Using the below formula, it measures a straight line between the query point and the other point being measured.
- **Manhattan distance (p=1)**: This is also another popular distance metric, which measures the absolute value between two points. It is also referred to as taxicab distance or city block distance as it is commonly visualized with a grid, illustrating how one might navigate from one address to another via city streets.
- **Minkowski distance**: This distance measure is the generalized form of Euclidean and Manhattan distance metrics. The parameter, p, in the formula below, allows for the creation of other distance metrics. Euclidean distance is represented by this formula when p is equal to two, and Manhattan distance is denoted with p equal to one.
- **Note:** q is used instead of p in equations

Distance functions

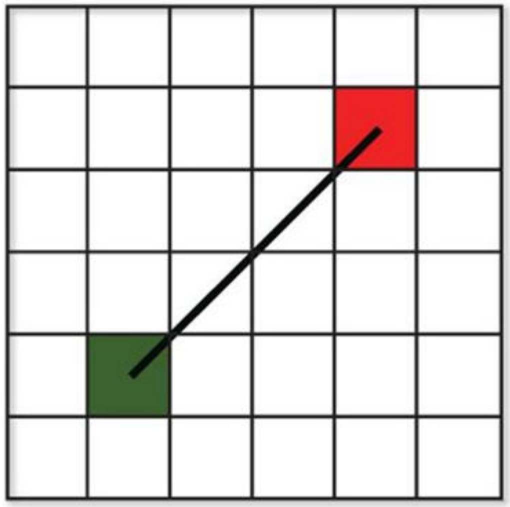
Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$



Manhattan Distance



Euclidean Distance

KNN classifier example

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Sepal Length	Sepal Width	Species
5.2	3.1	?

KNN classifier example

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Sepal Length	Sepal Width	Species
5.2	3.1	?

Step 1: Find Distance

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(x-a)^2 + (y-b)^2}$$

$$\text{Distance (Sepal Length, Sepal Width)} = \sqrt{(5.2-5.3)^2 + (3.1-3.7)^2}$$

Distance (Sepal Length, Sepal Width) = 0.608

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

KNN classifier example

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Step 2: Find Rank

KNN classifier example

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Step 3: Find the Nearest Neighbor

If k = 1 – Setosa

If k = 2 – Setosa

If k = 5 – Setosa

Approximation of discrete-valued target function

- Let us first consider learning ***discrete-valued target functions*** of the form

$$f : \Re^n \rightarrow V.$$

- Where, V is the finite set $\{v_1, \ldots v_s\}$
- The k- Nearest Neighbor algorithm for approximation a **discrete-valued target function** is given below:

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list $training\_examples$

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from $training\_examples$ that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Approximation of discrete-valued target function

Sl. No.	Height	Weight	Target	Distance	Nearest Points
1	150	50	Medium	8.06	
2	155	55	Medium	2.24	1
3	160	60	Large	6.71	3
4	161	59	Large	6.40	2
5	158	65	Large	11.05	
6	157	54	?		

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

Approximation of discrete-valued target function

Sl. No.	Height	Weight	Target	Distance	Nearest Points
1	150	50	Medium	8.06	
2	155	55	Medium	2.24	1
3	160	60	Large	6.71	3
4	161	59	Large	6.40	2
5	158	65	Large	11.05	
6	157	54	?		

k=3

$\delta(a,b)=1$

$a==b$

$\delta(a,b)=0$

$a\neq b$

$$\hat{f}(x_q) \leftarrow \underset{v\in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

m

L

$1+0+0=1$

$0+1+1=2$

Hamming Distance for Nominal and binary attributes

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

The Hamming distance between:

- "karolin" and "kathrin" is 3.
- "karolin" and "kerstin" is 3.
- 1011101 and 1001001 is 2.
- 2173896 and 2233796 is 3.

Burger liking example using hamming distance

	Pepper	Ginger	Chilly	Liked
A	True	True	True	False
B	True	False	False	True
C	False	True	True	False
D	False	True	False	True
E	True	False	False	True

New Example - Q: pepper: false, ginger: true, chilly : true

Burger liking example using hamming distance

	Pepper	Ginger	Chilly	Liked	Distance
A	True	True	True	False	1 + 0 + 0 = 1
B	True	False	False	True	1 + 1 + 1 = 3
C	False	True	True	False	0 + 0 + 0 = 0
D	False	True	False	True	0 + 0 + 1 = 1
E	True	False	False	True	1 + 1 + 1 = 3

New Example - Q: pepper: false, ginger: true, chilly : true

Burger liking example using hamming distance

	Pepper	Ginger	Chilly	Liked	Distance	3NN
A	True	True	True	False	1 + 0 + 0 = 1	2
B	True	False	False	True	1 + 1 + 1 = 3	
C	False	True	True	False	0 + 0 + 0 = 0	1
D	False	True	False	True	0 + 0 + 1 = 1	2
E	True	False	False	True	1 + 1 + 1 = 3	

New Example - Q: pepper: false, ginger: true, chilly : true

Weighted similarity measure

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- Given the training data, predict the class of the following new example using k-Nearest Neighbour for k=5:
- age<=30, income=medium, student=yes, credit- rating=fair.

Weighted similarity measure

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- For similarity measure use a simple match of attribute values:

$$\sum_{i=1}^{4} w_i * \frac{\partial(a_i, b_i)}{4}$$

- *where* $\partial(a_i, b_i)$ is 1 if a_i equals b_i and 0 otherwise.
- a_i and b_i are either age, income, student or credit_rating.
- Weights are all 1 except for income it is 2.

Weighted similarity measure

age	income	student	Credit rating	Buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

age<=30, income=medium, student=yes, credit-rating=fair

RID	Class	Similarity to New
1	No	(1*1+2*0+1*0+1*1)/4 = 0.5
2	No	(1*1+2*0+1*0+1*0)/4 = 0.25
3	Yes	(1*0+2*0+1*0+1*1)/4 = 0.25
4	Yes	(1*0+2*1+1*0+1*1)/4 = 0.75
5	Yes	(0+0+1+1)/4 = 0.5
6	No	(0+0+1+0)/4 = 0.25
7	Yes	(0+0+1+0)/4 = 0.25
8	No	(1+2+0+1)/4 = 1
9	Yes	(1+0+1+1)/4 = 0.75
10	Yes	(0+2+1+1)/4 = 1
11	Yes	(1+2+1+0)/4 = 1
12	Yes	(0+2+0+0)/4 = 0.5
13	Yes	(0+0+1+1)/4 = 0.5
14	No	(0+2+0+0)/4 = 0.5

Weighted similarity measure

- Among the **five nearest neighbours** four are from class *Yes* and one from class *No*.
- Hence, the k-NN classifier predicts **buys_computer=yes** for the new example.

age<=30, income=medium, student=yes, credit-rating=fair

RID	Class	Similarity to New
1	No	(1*1+2*0+1*0+1*1)/4 = 0.5
2	No	(1*1+2*0+1*0+1*0)/4 = 0.25
3	Yes	(1*0+2*0+1*0+1*1)/4 = 0.25
4	Yes	(1*0+2*1+1*0+1*1)/4 = 0.75
5	Yes	(0+0+1+1)/4 = 0.5
6	No	(0+0+1+0)/4 = 0.25
7	Yes	(0+0+1+0)/4 = 0.25
8	No	(1+2+0+1)/4 = 1
9	Yes	(1+0+1+1)/4 = 0.75
10	Yes	(0+2+1+1)/4 = 1
11	Yes	(1+2+1+0)/4 = 1
12	Yes	(0+2+0+0)/4 = 0.5
13	Yes	(0+0+1+1)/4 = 0.5
14	No	(0+2+0+0)/4 = 0.5

Distance-weighted Nearest Neighbor Algorithm

- The refinement to the k-NEAREST NEIGHBOR Algorithm is to weight the contribution of each of the k neighbors according to their distance to the query point xq, giving greater weight to closer neighbors.
- For example, in the k-Nearest Neighbor algorithm, which approximates discrete-valued target functions, we might weight the vote of each neighbor according to the inverse square of its distance from xq.
- Distance-Weighted Nearest Neighbor Algorithm for approximation a discrete-valued target functions

Distance-weighted Nearest Neighbor Algorithm

Training algorithm:

- For each training example $\langle x, f(x)\rangle$, add the example to the list $training\_examples$

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from $training\_examples$ that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Distance-weighted Nearest Neighbor Algorithm

Sl. No.	Height	Weight	Target	Distance	$1/distance^2$	Nearest Points
1	150	50	Medium	8.06		
2	155	55	Medium	2.24	0.45	1
3	160	60	Large	6.71	0.15	3
4	161	59	Large	6.40	0.16	2
5	158	65	Large	11.05		
6	157	54	?			

Distance-weighted Nearest Neighbor Algorithm

Sl. No.	Height	Weight	Target	Distance	1/distance^2	Nearest Points
1	150	50	Medium	8.06		
2	155	55	Medium	2.24	0.45	1
3	160	60	Large	6.71	0.15	3
4	161	59	Large	6.40	0.16	2
5	158	65	Large	11.05		
6	157	54	? medium			

m: $0.45\,\delta(m,m) + 0.15\,\delta(m,L) + 0.16\,\delta(m,L)$

$0.45 * 1 + 0.15 * 0 + 0.16 * 0 = 0.45$

L: $0.45\,\delta(L,m) + 0.15\,\delta(L,L) + 0.16\,\delta(L,L) = 0 + 0.15 + 0.16 = 0.31$

Approximation of real-valued target function

- The K- Nearest Neighbor algorithm for approximation a **real-valued target function** is given below $f : \Re^n \rightarrow \Re$

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list $training\_examples$

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from $training\_examples$ that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Distance-weighted Nearest Neighbor Algorithm

- Distance-Weighted Nearest Neighbor Algorithm for approximation a Real-valued target functions

---

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Distance-weighted Nearest Neighbor Algorithm

Sl. No.	Height	Weight	Target	Distance	1/distance^2	Nearest Points
1	150	50	1.5	8.06		
2	155	55	1.2	2.24	0.45 ✓	1
3	160	60	1.8	6.71	0.15 ✓	3
4	161	59	2.1	6.40	0.16 ✓	2
5	158	65	1.7	11.05		
6	157	54	?			

Distance-weighted Nearest Neighbor Algorithm

Sl. No.	Height	Weight	Target	Distance	1/distance^2	Nearest Points
1	150	50	1.5	8.06		
2	155	55	1.2	2.24	0.45 ✓	1
3	160	60	1.8	6.71	0.15 ✓	3
4	161	59	2.1	6.40	0.16 ✓	2
5	158	65	1.7	11.05		
6	157	54	? 1.51			

$$\hat{f}(x_q) = (0.45 \times 1.2 + 0.15 \times 1.8 + 0.16 \times 2.1) / (0.45 + 0.15 + 0.16)$$

$$= 1.146 / 0.76 = 1.51$$