K Nearest Neighbors

In K-Nearest Neighbors (KNN), the class of a new data point is predicted based on the majority class among its k closest training samples, measured using a distance metric like Euclidean distance.

22 OCT, 2025

BAYESIANBRAT*

Nearest Neighbors (NNs)

⇒ It is a simple supervised learning model.

→ Given an instance x_0, determine nearest neighbor = instance in training data closest to x_0

→ Assign x_0 to the class of the nearest neighbor

→ Often use K nearest neighbors (KNNs)

→ Assign x_0 to the majority class of the KNNs

Euclidean distance is measured as

$$d(x,y) = \sqrt{\Sigma(x_i - y_i)^2}$$

→ K is a hyperparameter : a parameter set before the model training process begins.

→ Hyperparameter = a setting you choose before training.

→ Not learned from data

→ Instead you decide it's value and test which works best.

For K-NN

K = how many neighbors you look at

small K → very flexible, but noisy

Large K → smoother, but might miss local structure

For K=1 or any small K, causes overfitting

For K = Big, there's a risk of underfitting

Also, KNN is a ~~almost~~ almost lazy learner because it learns nothing during the training stage. Because during prediction, it recalculates everything.

→ Euclidean distance:

$$d(x,y) = \sqrt{\Sigma(x_i - y_i)^2}$$

→ the straight-line distance (pythagoras)

→ Manhattan distance:

$$d(x,y) = \Sigma |x_i - y_i|$$

→ Minkowski distance:

→ Minkowski Distance:

This is a general formula that covers both Euclidean and Manhattan

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

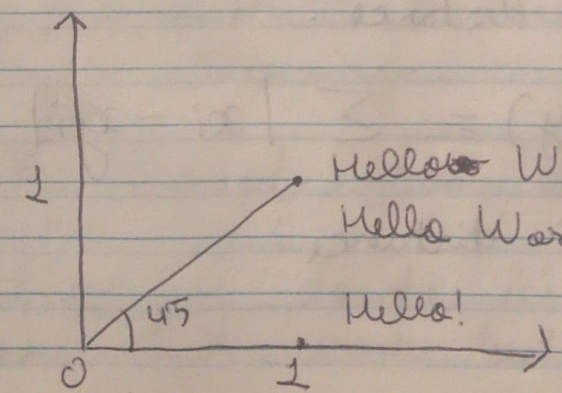
☆ Cosine Similarity:

→ Cosine similarity is determined by the $\cos(\theta)$, where θ is the angle between two vectors.

→ These two vectors are formed by a point on graph to origin and some another point to the origin of the graph.

⇒ Example:

Words	Hello	World
Hello World!	1	1
Hello!	1	0
Hello World?	1	1



$\cos(0) = 1$
$\cos(45) = 0.71$

There's a generalized formula:

$$\text{Cosine similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\ \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Exercise

	I	love	Troll	2	Cymbeline
A: I love Troll 2	1	1	1	1	0
B: I love Cymbeline	1	1	0	0	1

Applying formula:

$$= \frac{(1\times1)+(1\times1)+(1\times0)+(1\times0)+(0\times1)}{\sqrt{1^2+1^2+1^2+1^2+0^2}\ \sqrt{1^2+1^2+0^2+0^2+1^2}}$$

$$= \frac{2}{\sqrt{4}\ \sqrt{3}}$$

$$= \frac{2}{\sqrt{3}}$$

$$= 0.58$$

→ In summary, the cosine similarity is a relatively easy to calculate metric that tells us how similar or different things are.

Cosine distance = 1 - Cosine Similarity

Categorical Attributes:

→ So far, we used numeric distances (Euclidean, Manhattan etc.)

→ But what if attributes like ("Wet/dry", "smooth/rough") need to be calculated?

→ We can't subtract them, instead we use match distance

The example in the PDF
We have a query instance:

x_0 = (Weather = Wet, surface = median, experience = low, grade = flat)

instance	Weather	surface	experience	grade
0	dry	smooth	high	steep
1	dry	smooth	high	flat
2	dry	smooth	high	flat
3	wet	rough	low	steep
4	dry	rough	low	steep

→ Take instance **0 vs x_0**:

Weather: Wet vs dry → different → 1
surface: mid vs smooth → different → 1
experience: low vs high → different → 1
grade: flat vs steep → different → 1

Total distance → 4

Take instance 3 vs x_0:

Weather : Wet vs Wet → 0
surface : mid vs rough → 1
esp : low vs low → 0
grade : flat vs steep → 1

Total distance = 2

Likewise we calculate for all other instances.

★ Attribute Scales

→ Many distance measures depends on the scale of attribute (unit of measure)

→ Often important to scale the attributes to unitless quantities.

Standardization formula:

$$x^* = \frac{x - \text{mean}(x)}{\text{stddev}(x)}$$

Example:

x_1	x_2	x_1^*	x_2^*
1.8	3.7	-0.2641	0.48668
2.0	3.0	-0.15288	0.27762
2.5	3.5	0.12523	0.42695
2.0	4.1	-0.15288	0.6015
1.8	1.0	-0.26413	-0.31971
2.3	3.8	0.01398	0.516553
4.0	6.0	0.9595	
-1.0	3.3	-1.8215	
6.0	-4.0	2.0720	
1.3	-3.7	-0.5125	

How did we calculate

x_1 mean = 2.3 stddev = 1.798

x_2 mean = 2.1 stddev = 3.348

$$x_1^* = \frac{1.8 - 2.3}{1.7978} = -0.264$$

$$x_2^* = \frac{3.7 - 2.1}{3.348} = 0.487$$

Weighted KNN

→ In standard KNN, you pick the k nearest neighbors.

→ Each neighbor gets equal weight → simple major vote (classification) or average (regression).

Problem:

→ What if one neighbor is much closer to than others?

→ Shouldn't it have more influence?

Weighted KNN

→ Instead of giving all neighbors equal say, give closer neighbors higher weight.

$$\hat{y}(x_0) \simeq \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=K}^{K} w_i}$$

Where

$$w_i = \frac{1}{d(x_0 x_i)}$$

Sometimes $w_i = \frac{1}{d(x_0, x_i)^2}$ etc.

So, the nearest neighbors vote counts more than a farther one.

→ Example (classification)

Red at 0.5
Blue at 0.6
Red at 1.2

Unweighted K-NN (K = 3)
Red = 2 votes, Blue = 1 vote → predicted Red

Weighted K-NN ($\frac{1}{d}$ rule)

Red → 0.5 → $\frac{1}{0.5} = 2.0$

Blue → 0.6 → $\frac{1}{0.6} = 1.67$

Red → 1.2 → $\frac{1}{1.2} = 0.83$

Total weights = Red : 2.83
Blue : 1.67

Answer: Red

~~Example~~

Example data (from before)

	dist	Weight
Red	0.5	2.0
Blue	0.6	1.67
Red	1.2	0.83

Total Weight = $2.0 + 1.6667 + 0.8333$
$= 4.5$

Classification (Weighted vote):

Red $w_1 + w_3 = 2.0 + 0.8333 = 2.8333$
$w_2 = 1.6667$

Regression (if labels are numeric, Red = 1, Blue = 0)

$$\hat{y} = \frac{2 \times 1 + 1.6667 \cdot 0 + 0.8333 \times 1}{4.5}$$

$$= 0.6296$$