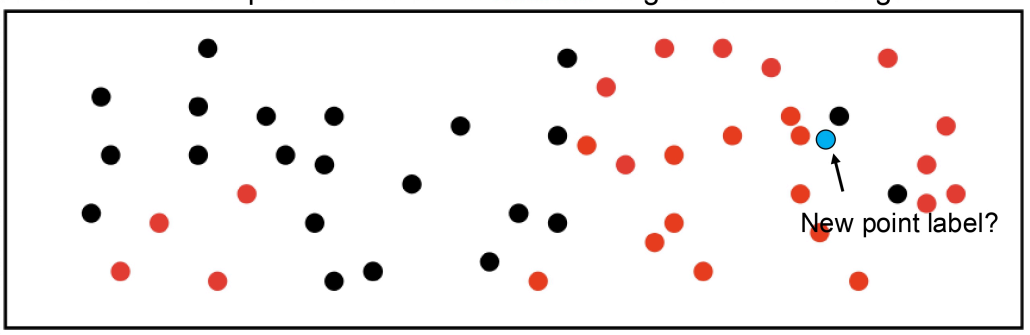
KNN : k-nearest Neighbours

kNN Basics

- Full name: k-Nearest Neighbours (kNN, or k-NN).
- It is nonparametric.
  No assumption about the functional form of the model.
- It is instance-based.
  The prediction is based on a comparison of a new point with data points in the training set, rather than a model.
- It is a lazy algorithm.
  No explicit training step. Defers all the computation until prediction.
- Can be used for both classification and regression problems.

Intuitive Understanding

Instead of approximating a model function $f(x)$ globally, kNN approximates the label of a new point based on its nearest neighbours in training data.



Q1: How to choose k? e.g. let k = 3 to avoid issues.

Q2: how to we measure the distance between examples?

Distance metrics (or similarity metrics)

Given two points $\boldsymbol{x}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, \ldots, x_d^{(1)}\right), \boldsymbol{x}^{(2)} = \left(x_1^{(2)}, x_2^{(2)}, \ldots, x_d^{(2)}\right)$ in a d-dimensional space:

- Minkowski distance (or L^p norm)

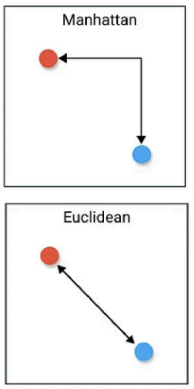
$$D\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) = \sqrt[p]{\sum_{i=1}^{d} \left|x_i^{(1)} - x_i^{(2)}\right|^p}$$

- When p=1, it becomes Manhattan distance

$$D\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) = \sum_{i=1}^{d} \left|x_i^{(1)} - x_i^{(2)}\right|$$

- When p=2, it becomes Euclidean distance

$$D\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) = \sqrt{\sum_{i=1}^{d} \left|x_i^{(1)} - x_i^{(2)}\right|^2}$$



Simple basic ass distance formulae made to look more complicated

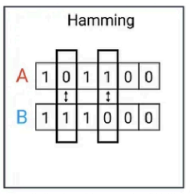
Distance metrics in kNN (common choice)

- Euclidean distance for real values (also called L^2 distance).
  Sometimes Manhattan

$$D(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \sqrt{\sum_{i=1}^{d} \left| x_i^{(1)} - x_i^{(2)} \right|^2}$$

- Hamming distance for discrete/categorical values, e.g. $x \in \{rainy, sunny\}$.

$$D(x^{(1)}, x^{(2)}) = \begin{cases} 0, if\ x^{(1)} = x^{(2)} \\ 1, \quad otherwise \end{cases}$$



Same as 0/1 model in linear and logistic regression

kNN algorithm

Input: neighbour size $k > 0$, training set $\{(\boldsymbol{x}^{(n)}, y^{(n)}): n = 1,2 \ldots N\}$, a new unlabelled data $\boldsymbol{x}^{(j)}$

for n = 1, 2… N // each example in the training set
  Calculate $D(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(n)})$ // distance between $x^{(j)}$ and $x^{(n)}$
  Select k training examples closest to $\boldsymbol{x}^{(j)}$
Return $y^{(j)}$ = the plurality vote of labels from the k examples. (classification) or
$y^{(j)}$ = average/median of the y values of the k examples. (regression)

▼ ***Plurality Vote***

💡

Understanding Plurality Vote

- Each of the **k nearest neighbors** has a known label.
- The **label that appears the most** among the k neighbors is assigned to the new data point.
- It does **not require an absolute majority** (more than 50%)—just the most frequent label wins.

Example 1: Binary Classification

Imagine we have a **k-NN classifier with k = 5**, and we are classifying a new data point as **either "Red" (R) or "Blue" (B)"**.

Step 1: Find the 5 nearest neighbors

- 3 neighbors are **Red (R)**
- 2 neighbors are **Blue (B)**

Step 2: Count the votes

- **Red (R): 3 votes**
- **Blue (B): 2 votes**

Step 3: Assign the label

- Since **Red (R) has the highest count**, the new data point is classified as **Red (R)**.

Example 2: Multi-Class Classification

Let's consider a classification where we have **three classes: "Cat", "Dog", and "Rabbit"** and **k = 7**.

Step 1: Find the 7 nearest neighbors

- 3 neighbors are **Cats (C)**
- 2 neighbors are **Dogs (D)**
- 2 neighbors are **Rabbits (R)**

Step 2: Count the votes

- **Cat (C): 3 votes**
- **Dog (D): 2 votes**
- **Rabbit (R): 2 votes**

Step 3: Assign the label

- **Cat (C) has the most votes** (even though it is not a strict majority), so the new data point is classified as **Cat (C)**.

Limitations of Plurality Voting in k-NN

- **Ties can occur** when multiple classes have the same count. A common solution is to **reduce k to an odd number** or use **weighted voting** (where closer neighbors have more influence).
- **Imbalanced classes** can bias results if one class dominates the dataset

Check your understanding

Consider a binary problem (lemon or orange) with 2 dimensions (height and width) with following training examples:

- $\boldsymbol{x}^{(1)}$ =(6,6), $y^{(1)}$ =orange
- $\boldsymbol{x}^{(2)}$ =(8,10), $y^{(2)}$ =lemon
- $\boldsymbol{x}^{(3)}$ =(7,6), $y^{(3)}$ =orange

New example

- $\boldsymbol{x}^{(4)}$ =(8,7), $y^{(4)}$ =? Using k=1 nearest neighbour, and Euclidean distance

Distances:

$$\sqrt{2^2 + 1^2} = \sqrt{5}$$

$$\sqrt{0^2 + 3^2} = \sqrt{6}$$

$$\sqrt{1^2 + 1^2} = \sqrt{2}$$

$\therefore$ the closest neighbour is $x^{(3)}, y^{(3)}$

$\therefore y^{(4)}$ is orange

Check your understanding

Consider a regression problem (lemon' weight) with 2 dimensions (height and width) with following training examples:

- $\boldsymbol{x}^{(1)}$ =(6,6), $y^{(1)}$ =10
- $\boldsymbol{x}^{(2)}$ =(8,10), $y^{(2)}$ =20
- $\boldsymbol{x}^{(3)}$ =(7,6), $y^{(3)}$ =15

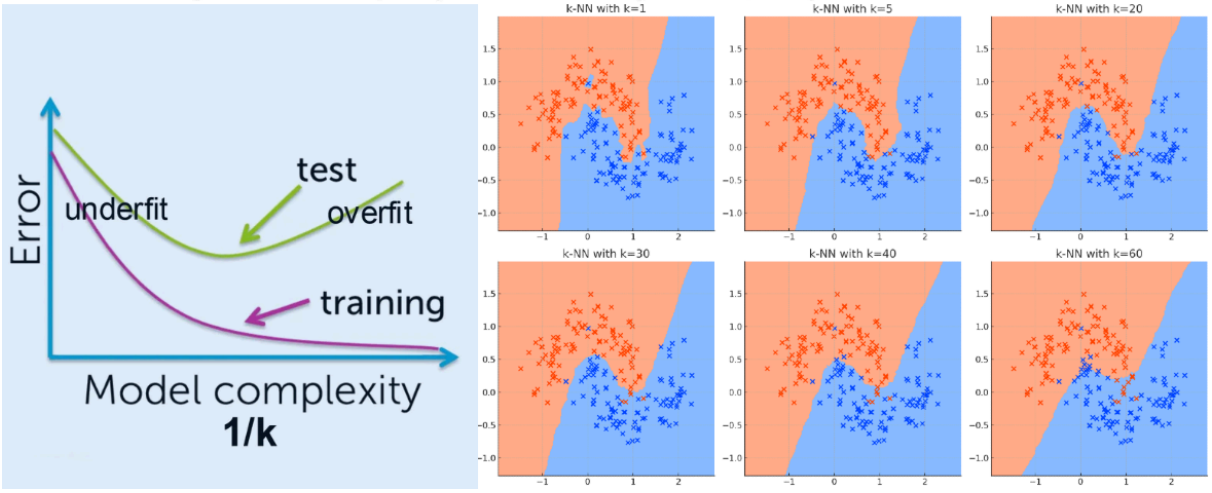
New example

- $\boldsymbol{x}^{(4)}$ =(8,7), $y^{(4)}$ =?
- If k = 2, what is the label of $\boldsymbol{x}^{(4)}$?

1. $Distances$:
   a.
      i. $\sqrt{2^2+1^2}=\sqrt{5}$
      ii. $\sqrt{0^2+3^2}=\sqrt{6}$
      iii. $\sqrt{1^2+1^2}=\sqrt{2}$
      iv. $\therefore$ the closest neighbour is $x^{(3)}, y^{(3)}$
      v. $\therefore y^{(4)}$ is 15
   b. If $k=2$, what is label of $x^{(4)}$:
      i. $\frac{\sqrt{2}+\sqrt{5}}{2} \approx 1.8251407699 \approx 1.83$
      ii. $\therefore x^{(4)}=1.83$

How to choose k?

- Recall: Overfitting and Underfitting
- k changes model complexity: smaller k -> higher complexity



In k-Nearest Neighbors $(k - NN)$, a smaller k leads to a more complex model because:

1. **More Sensitivity to Noise:**
    a. *When k is small (e.g, $k = 1$), the classification of each point relies on just one or a few nearest neighbors.*
    b. *This makes the decision boundary highly flexible, capturing fine details but also noise, leading to high variance.*
2. **Highly Irregular Decision Boundaries**:
    a. *As seen in the top-left plot $(k = 1)$, the decision boundary is highly jagged, adapting to individual data points rather than generalizing well.*
    b. *This suggests overfitting, where the model fits the training data too closely.*
        i. **Overfitting Risk**:
            1. *A high-complexity model (low k) is prone to memorizing the training data instead of capturing the underlying pattern, leading to poor generalization to new data.*
3. **Higher Model Complexity**:
    a. *Complexity in a model refers to how much detail it captures.*
    b. *Smaller k values create more intricate decision boundaries, meaning the model tries to closely follow the training data distribution, increasing complexity*

How to choose k?

1. Small $k \rightarrow$ small neighbourhood $\rightarrow$ high complexity $\rightarrow$ may overfit
2. Large $k \rightarrow$ large neighbourhood $\rightarrow$ low complexity $\rightarrow$ may underfit
3. k usually between $3 - 15$:
    a. $k < \sqrt{N}$ where N is the number of training examples
4. More on k selection next week

The issue in numeric attribute ranges

- *Attributes $x = (x_1, x_2, ..., x_d)$ may have different ranges*
- *The attribute with a larger range is treated as more important by the k-NN algorithm*
    - *(some learning bias is embedded!)*
    - All this means is that the numbers with a larger range affect the distance formulae of the k-NN algorithm more prominently
- *It can affect the performance if you don't want to treat attributes differently*
- *For example, if $x_1 \in [0, 2]$ (e.g. height) , and $x_2 \in [50, 100]$ (e.g. age)*
    - *x_2 will affect the distance more*
- *Solutions?*

Normalisation and Standardization

Method 1

Normalisation:

- Linearly scale the range of each attribute to be, e.g. in $[0, 1]$
- $x_{j\_new}^{(n)} = \frac{x_j^{(n)} - min(x_j)}{max(x_j) - min(x_j)}$
- Taking the example of $(x_1 \in [0, 2] \quad and \quad x_1 = 1.5)$ and $(x_2 \in [50, 100] \quad and \quad x_2 = 75)$
  - $x_{1\_new} = \frac{1.5-0}{2-0} = \frac{15}{20} = \frac{3}{4} = 0.75$
  - $x_{2\_new} = \frac{75-50}{100-50} = \frac{25}{50} = \frac{1}{2} = 0.5$

Method 2

Standardization:

- Linearly scale each dimension to have 0 mean and variance 1
  - (by computing mean μ and variance σ^2)
- $x_{j\_new}^{(n)} = \frac{x_j^{(n)} - \mu_j}{\sigma_j}$
- Where
  - $\mu_j = \frac{1}{N} \Sigma_{n=1}^{N} x_j^{(n)}$
  - $\sigma_j = \sqrt{\frac{1}{N} \Sigma_{n=1}^{N} (x_j^{(n)} - \mu_j)^2}$
- Taking the example of $(x_1 \in [0, 2] \quad and \quad x_1 = (13, 2))$ and $(x_2 \in [50, 100] \quad and \quad x_2 = (7, 5))$
  - $\mu_{x_1} = 7.5$ and $\mu_{x_2} = 6$
    $\sigma_{x_1} = \sqrt{0.5(5.5^2 + 5.5^2)}$ and $\sigma_{x_2} = \sqrt{0.5(1^2 + 1^2)} = \sqrt{60.5/2} \Rightarrow \sigma_{x_1} = \sqrt{30.25}$ and $\sigma_{x_2} = 1 \therefore$
    Meaning μ is 7.5 and 6 and standard deviation is $\sqrt{30.25}$ and 1

  For $x_1 = 13$:

  $$x_{1\_new} = \frac{13-7.5}{5.5} = \frac{5.5}{5.5} = 1$$

For $x_1 = 2$:

$$x_{1\_new} = \frac{2-7.5}{5.5} = \frac{-5.5}{5.5} = -1$$

For $x_2 = 7$:

$$x_{2\_new} = \frac{7-6}{1} = \frac{1}{1} = 1$$

For $x_2 = 5$:

$$x_{2\_new} = \frac{5-6}{1} = \frac{-1}{1} = -1$$

Example

- Consider a dataset with 2 dimensions (ie. Attributes), where x_1 represents the age of a patient and x_2 represents the body weight. The output $y \in \{normal, abnormal\}$.
- Normalize each attribute of $\boldsymbol{x}^{(1)}$ to [0,1].

Patient	x_1	x_2	y
$\boldsymbol{x}^{(1)}$	14	70	n
$\boldsymbol{x}^{(2)}$	12	90	a
$\boldsymbol{x}^{(3)}$	15	66	n

kNN algorithm with normalization/standardization

Input: neighbour size $k > 0$, training set $\{(\boldsymbol{x}^{(n)}, y^{(n)}): n = 1,2 \ldots N\}$, a new unlabelled data $\boldsymbol{x}^{(j)}$
Normalise/standardize $\boldsymbol{x}^{(j)} \rightarrow \boldsymbol{x}_{new}^{(j)}$
for n = 1, 2… N // each example in the training set
Normalise/standardize $\boldsymbol{x}^{(n)} \rightarrow \boldsymbol{x}_{new}^{(n)}$
Calculate $D(\boldsymbol{x}_{new}^{(j)}, \boldsymbol{x}_{new}^{(n)})$ // normalized/standardized distance
Select k training examples closest to $x^{(j)}$
Return $y^{(j)}$ = the plurality vote of labels from the k examples. (classification) or
$y^{(j)}$ = average/median of the y values of the k examples. (regression)

Pros/cons

- kNN is a nonparametric, instance-based, lazy algorithm.
- Need to specify the distance function and pre-define k value.
- Easy to implement and interpret.
- It can approximate complex functions, so it has very good accuracy.
- It has to store all training data (large memory space), and calculate distance of each training example to the new example.
  There are smarter ways to store and use training data, e.g. KD-trees, remove redundant data.
- It can be sensitive to noise, especially when k is small.
- Its performance is degraded greatly as data dimension increases. (curse of dimensionality)
  As the volume grows larger, the “neighbors” become further apart and and not so close anymore. The prediction thus becomes less accurate.

1. **Pros/cons**

- *KNN is a nonparametric, instance-based, lazy algorithm.*
- *Need to specify the distance function and pre-define k value.*
- *Easy to implement and interpret.*

- *It can approximate complex functions, so it has very good accuracy.*
- *It has to store all training data (large memory space), and calculate distance of each training example to the new example.*
    - *There are smarter ways to store and use training data*
        - *For example*
            - *KD-trees*
            - *Remove redundant data.*
- *It can be sensitive to noise, especially when k is small.*
- *Its performance is degraded greatly as data dimension increases. (curse of dimensionality)*
- *As the volume grows larger, the "neighbors" become further apart and and not so close anymore. The prediction thus becomes less accurate.*

Fun project using kNN: where on earth is this photo from?

- Problem: where was this picture taken (country or GPS)?
- http://graphics.cs.cmu.edu/projects/im2gps/

- Get images from Flickr with gps info.
- Represent each image with meaningful features
- Apply kNN.