# K-NN
# k Nearest Neighbour Classifier

# Eager and Lazy Learners

***Lazy learners***: Save all data from training, use it for classifying

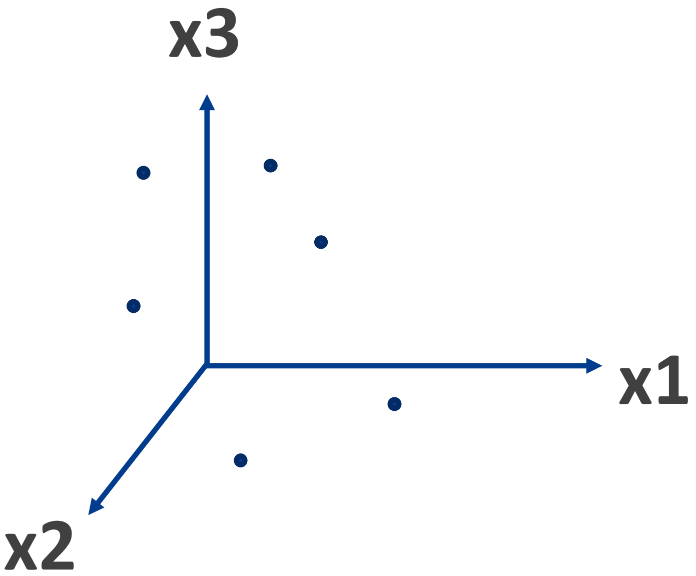
(The learner was lazy, classifier had to do the work)

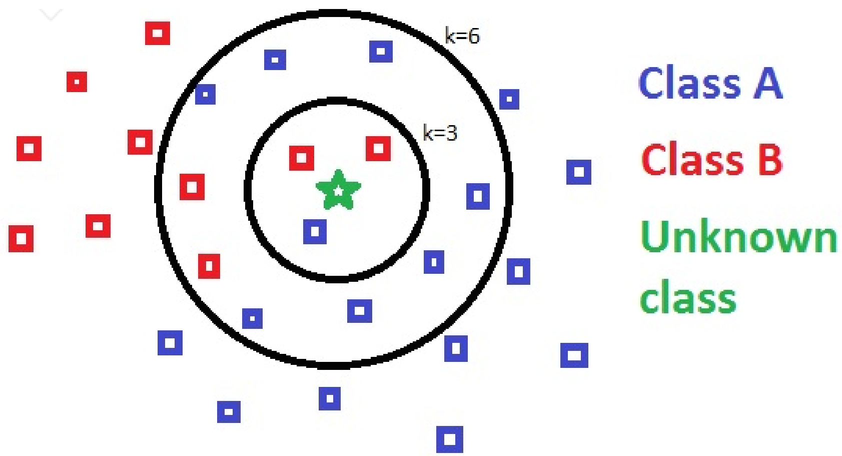
***Eager learners***: Build a (compact) model/structure during training, use the model for classification.

(The learner was eager/worked harder, classifier had simple life)

kNN is a lazy learner.

# k-NN: Basic Idea




*https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75*

# Distance Metrics

- Minkowski distance (L norm)

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{m} |x_i - y_i|^p}$$

- Most popular: Euclidean Distance L2 (p=2)

$$d_E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

  - e.g., d((1,2),(3,4))=2.83
- Manhattan distance L1 (p=1) $d_M(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$
  - e.g., d((1,2),(3,4))=4

# k-NN Algorithm

**Training algorithm:**

- For each training example $\langle x_j, f(x_j) \rangle$, where is $f(x_j)$ is class of $x_j$, add the example to the list *training_examples*.

**Classification algorithm**:

- Given a query instance $x_q$ to be classified, Let $(x_1, \dots, x_k)$ be the $k$ instances from *training_examples* that are *nearest* to $x_q$ by the distance function. Let $V$ be the set of classes for the $k$ training examples.
- Return $class(x_q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_j))$

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

# k-NN Example

| outlook | temperatur e | humidity | windy | play | distance |
|---|---|---|---|---|---|
| sunny | hot | high | FALSE | no | 2 (*) |
| sunny | hot | high | TRUE | no | 1 (*) |
| overcast | hot | high | FALSE | yes | 3 |
| rainy | mild | high | FALSE | yes | 3 |
| rainy | cool | normal | FALSE | yes | 3 |
| rainy | cool | normal | TRUE | no | 2 (*) |
| overcast | cool | normal | TRUE | yes | 2 (*) |
| sunny | mild | high | FALSE | no | 2 (*) |
| sunny | cool | normal | FALSE | yes | 2 (*) |
| rainy | mild | normal | FALSE | yes | 4 |
| sunny | mild | normal | TRUE | yes | 2 (*) |
| overcast | mild | high | TRUE | yes | 2 (*) |
| overcast | hot | normal | FALSE | yes | 4 |
| rainy | mild | high | TRUE | no | 1 (*) |

We have Play Tennis data on the left

And we have new instance: $\langle Outlk = sun, Temp = cool, Humid = high, Wind = true \rangle$

- In order to calculate distance, each feature is coded as 1 or 0, so feature space has 10 dimensions. So the $x_i - x_j$ is 0 if $x_i = x_j$, or 1 otherwise.
- By Manhattan distance we need to find 5 nearest neighbours.
- But here we have lots of ties, therefore we can choose randomly to make 5, or count all ties (*).
- So we have 5 “no” and 4 “yes”, by voting, decision is NO

(but notice that in this dataset, features can also be coded as ordinal, e.g., outlook(sunny, overcast, rainy)->(3,2,1). The problem is: what number do we assign to those features?)

# How to choose k for k-NN

- k=1 perfectly separates training data, so low bias but high variance
- By increasing the number of neighbours $k$ we increase bias and decrease variance (what happens when $k$ = $m$? $m$ is the number of observations)
- Ways to choose k:
  - By cross-validation (or any validation)
  - Keep k odd to avoid equal voting for classes
  - Use $\sqrt{n}$ where n is number of training instances