k neighbours classifier

(k-Nearest Neighbors)

Predictive Analytics

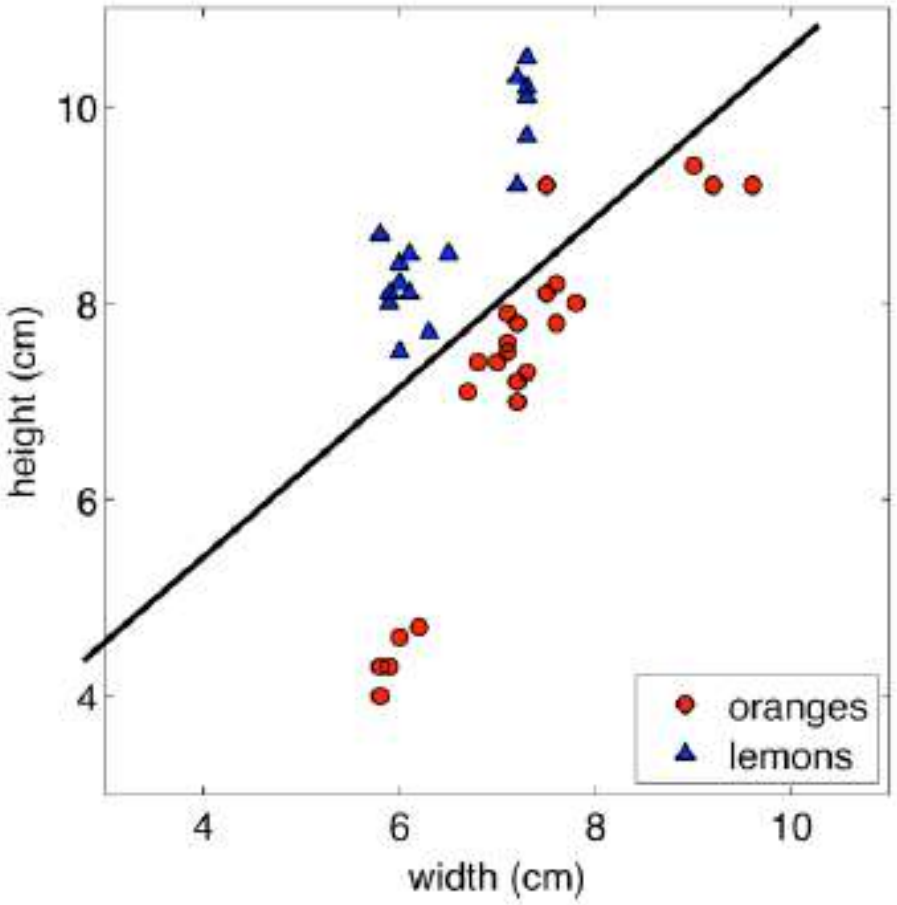
Sources

Based on materials by **Fragkiskos D. Malliaros**.

École Polytechnique / CentraleSupélec (Université Paris–Saclay).

Non-parametric learning

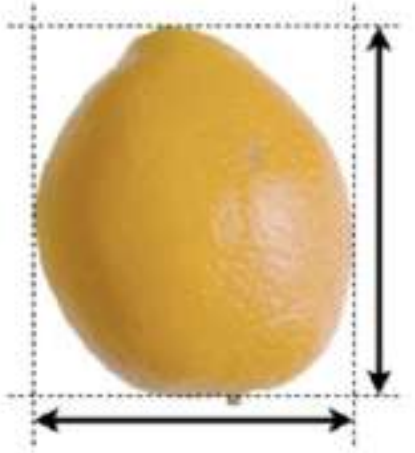
Classification: Oranges and Lemons



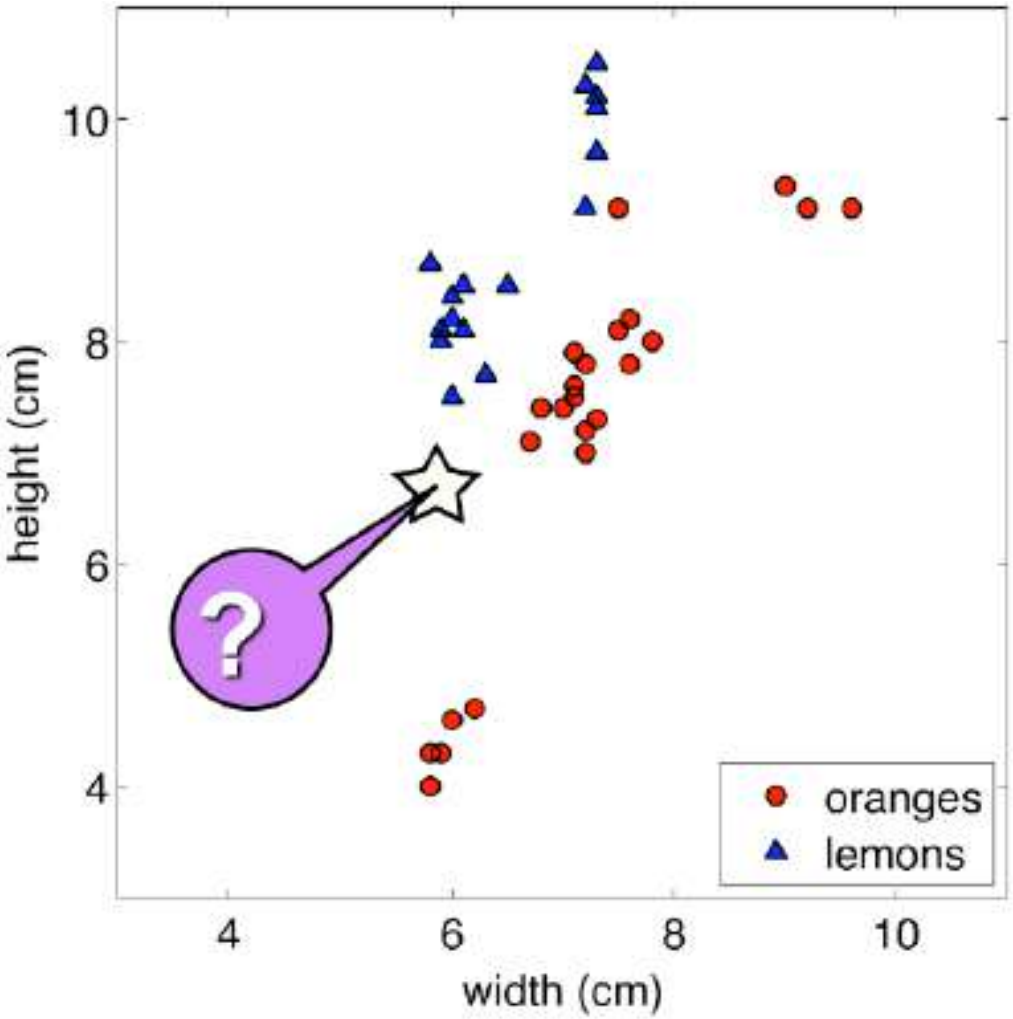
- We can construct a linear decision boundary:

$$y = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

  - Parametric model
  - Fixed number of parameters



Classification as Induction

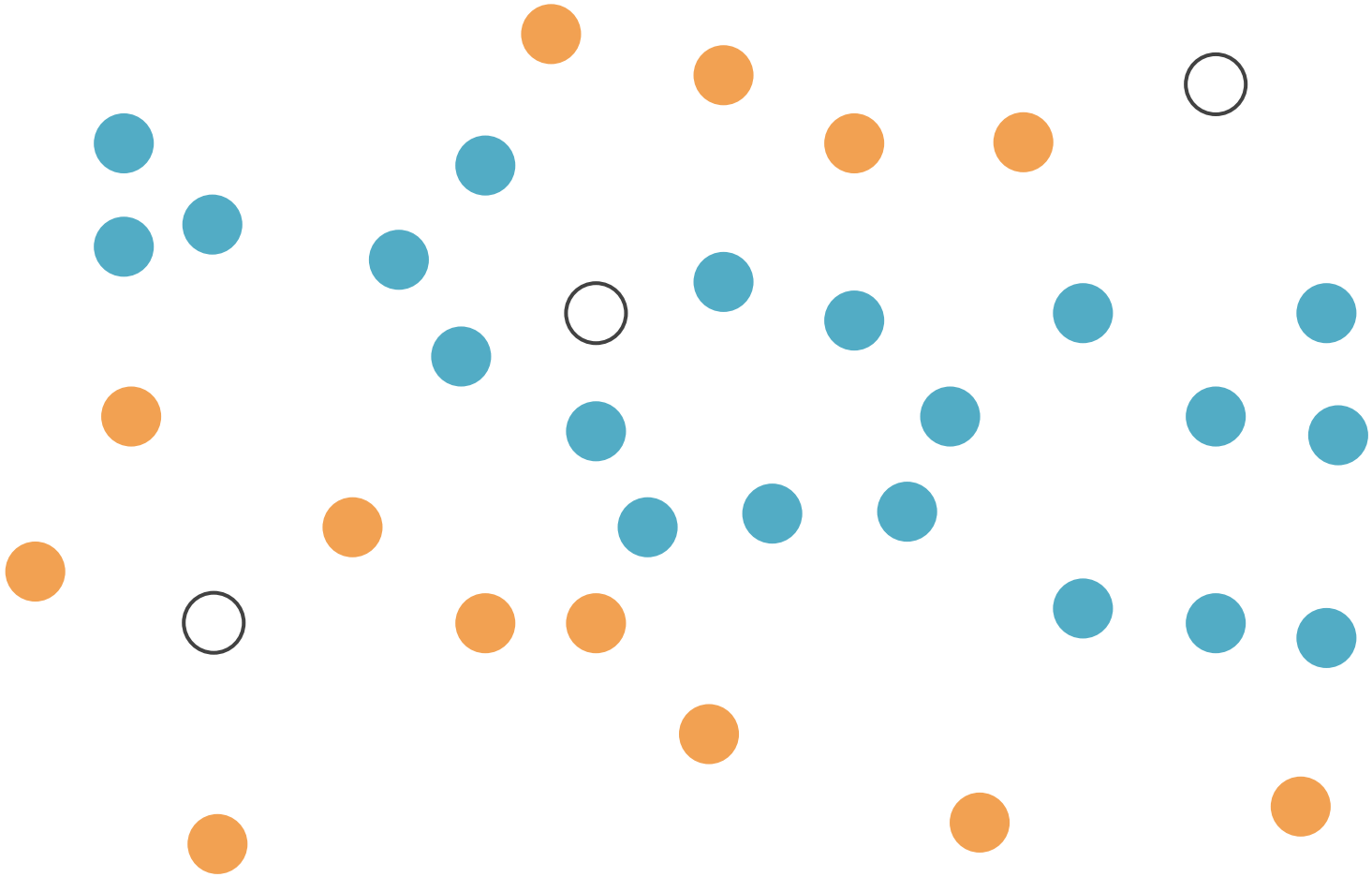


- Is there alternative way to formulate the classification problem?
- Classification as induction
  - Comparison to instances already seen in training
  - Non-parametric learning

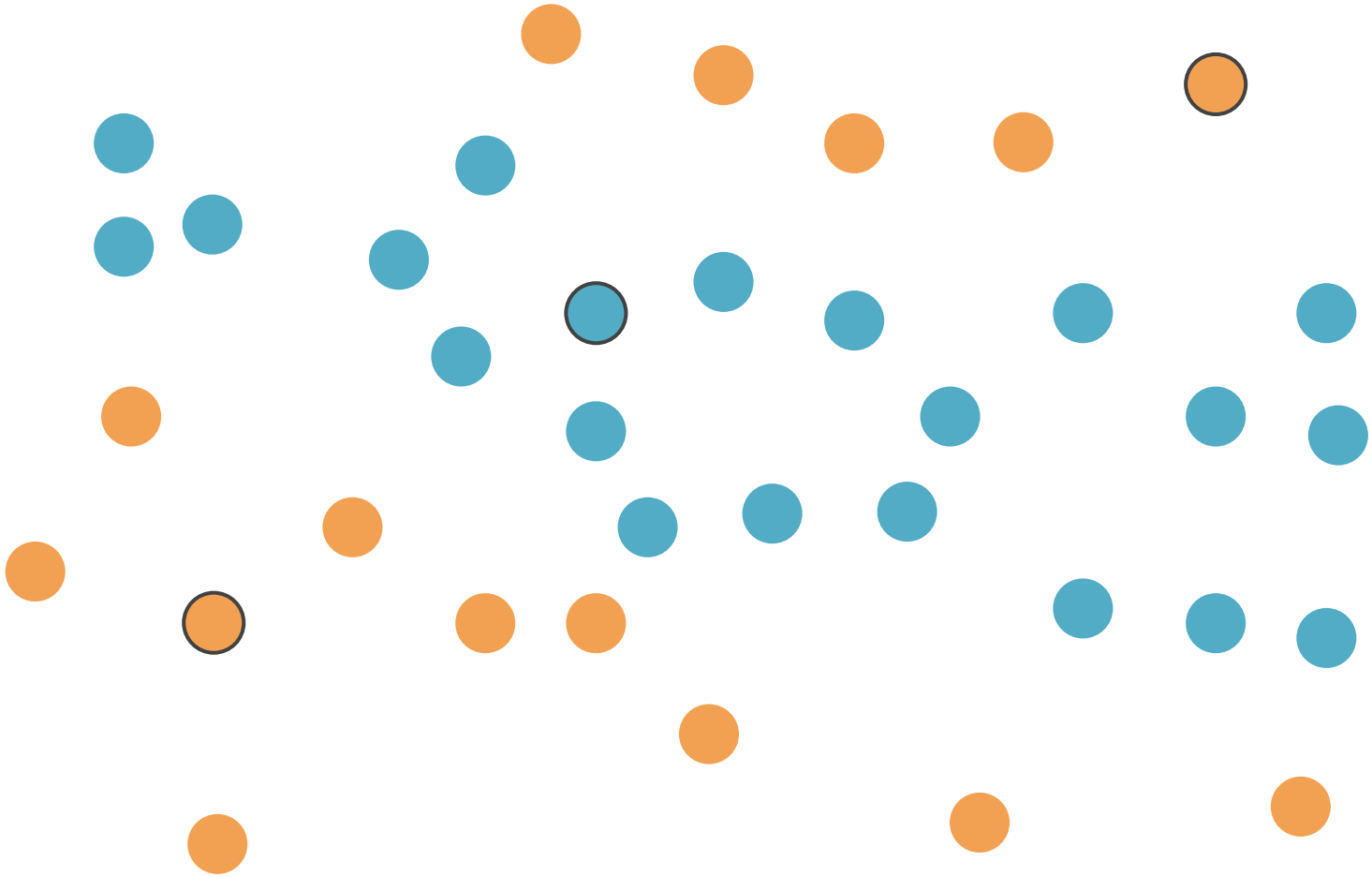
Non-parametric Learning

- Non-parametric learning algorithm (does not mean NO parameters)
  - The complexity of the decision function grows with the number of data points
    - Contrast with linear regression ($\approx$ as many parameters as features)
  - Usually: decision function is expressed directly in terms of the training examples
  - Examples:
    - k-nearest neighbors (today's lecture)
    - Tree-based methods (lecture 6)
    - Some cases of SVMs (lecture 7)

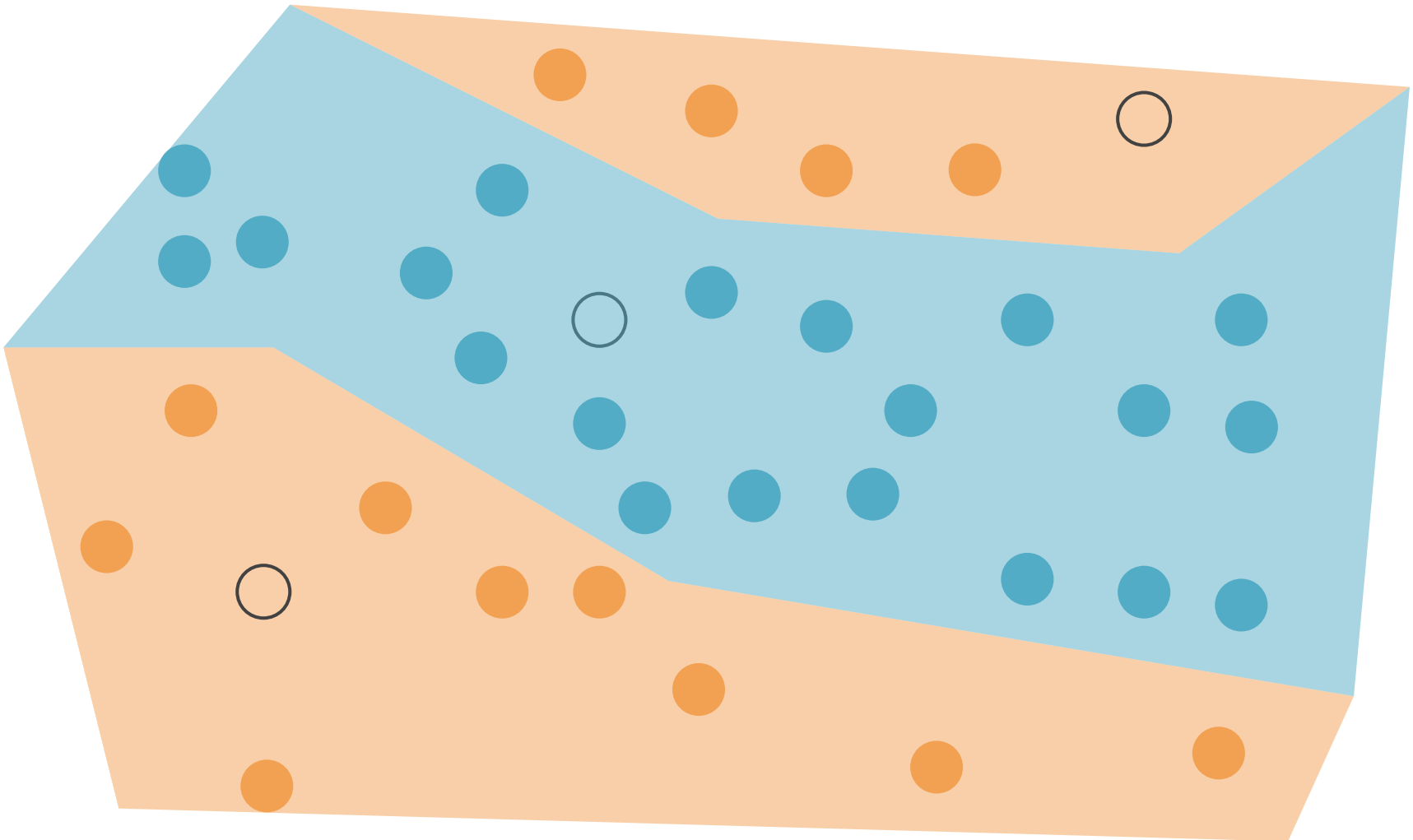
How Would You Color the Blank Circles?



How Would You Color the Blank Circles?



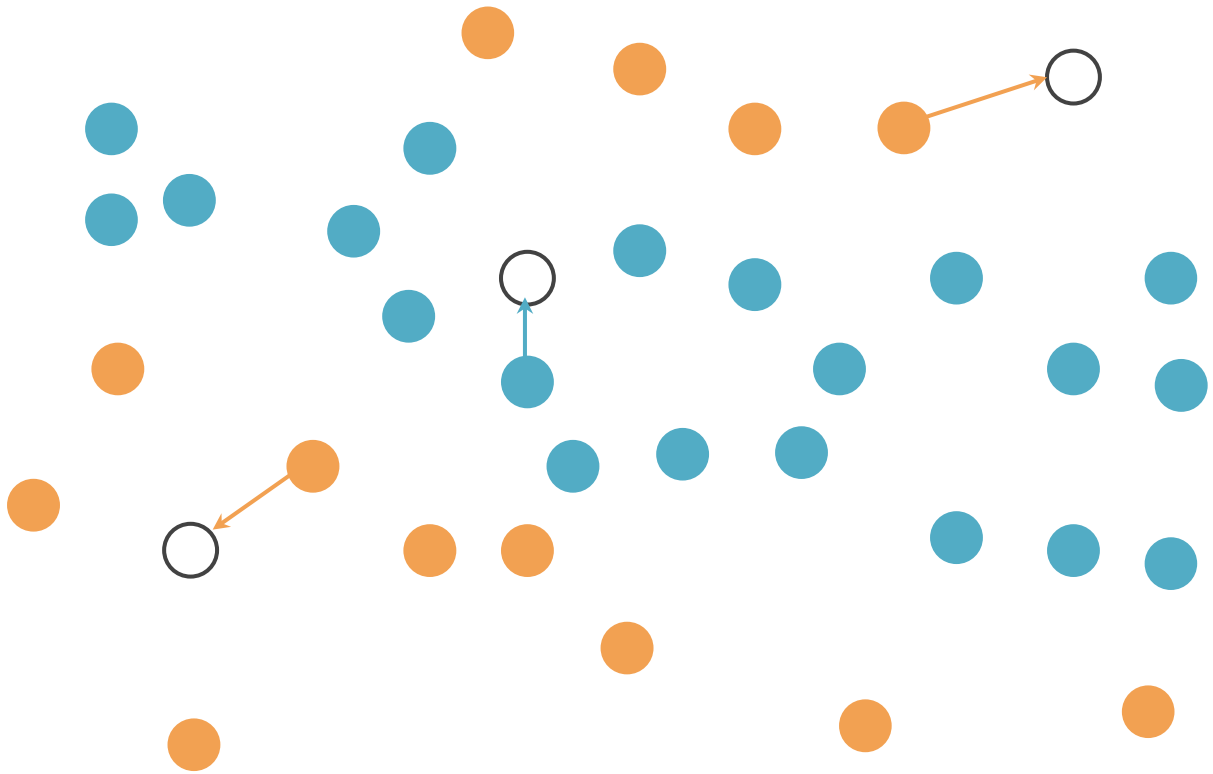
Partitioning the Space



The training data partitions the entire space

Nearest Neighbors – The Idea

- Learning:
  - Store all the training examples
- Prediction:
  - For a point x: assign the label of the training example closest to it



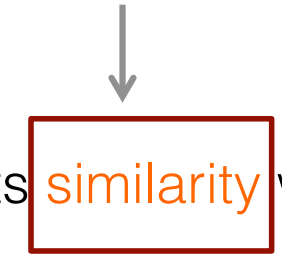
Nearest Neighbors – The Idea

- Learning:
  - Store all the training examples
- Prediction:
  - For a point x: assign the label of the training example closest to it
  - Classification
    - Majority vote: predict the class of the most frequent label among the k neighbors
  - Regression
    - Predict the average of the labels of the k neighbors

Instance-based Learning

- Learning
  - Store training instances
- Prediction
  - Compute the label for a new instance based on its similarity with the stored instances

Where the magic happens!



- Also called lazy learning
- Similar to case-based reasoning
  - Doctors treating a patient based on how patients with similar symptoms were treated
  - Judges ruling court cases based on legal precedent

Computing distances and similarities

Distance Function

- Distance function

$$d : \mathcal{X} \rightarrow \mathbb{R}_+$$

- Properties of a distance function (or metric)

1. $d(\mathbf{x}, \mathbf{z}) \geq 0$ non-negativity
2. $d(\mathbf{x}, \mathbf{x}) = 0$ identity of indiscernibles
3. $d(\mathbf{x}, \mathbf{z}) = d(\mathbf{z}, \mathbf{x})$ symmetry
4. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{u}) + d(\mathbf{u}, \mathbf{z})$ triangle inequality

Distance Between Instances

$\mathbf{x} \in \mathbb{R}^n$

- Euclidean distance (L2)

$$d(\mathbf{x}^1, \mathbf{x}^2) = \|\mathbf{x}^1 - \mathbf{x}^2\|_2 = \sqrt{\sum_{j=1}^{n} (x_j^1 - x_j^2)^2}$$

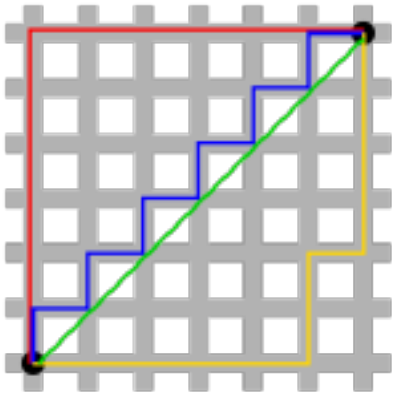
- Manhattan distance (L1)

$$d(\mathbf{x}^1, \mathbf{x}^2) = \|\mathbf{x}^1 - \mathbf{x}^2\|_1 = \sum_{j=1}^{n} |x_j^1 - x_j^2|$$

- Lp-norm

$$d(\mathbf{x}^1, \mathbf{x}^2) = \|\mathbf{x}^1 - \mathbf{x}^2\|_p = \left(\sum_{j=1}^{n} |x_j^1 - x_j^2|^p \right)^{1/p}$$

Manhattan distance: The sum of the horizontal and vertical distances between points on a grid



Source: Wikipedia

From Distance to Similarity

$$s = \frac{1}{1+d}$$

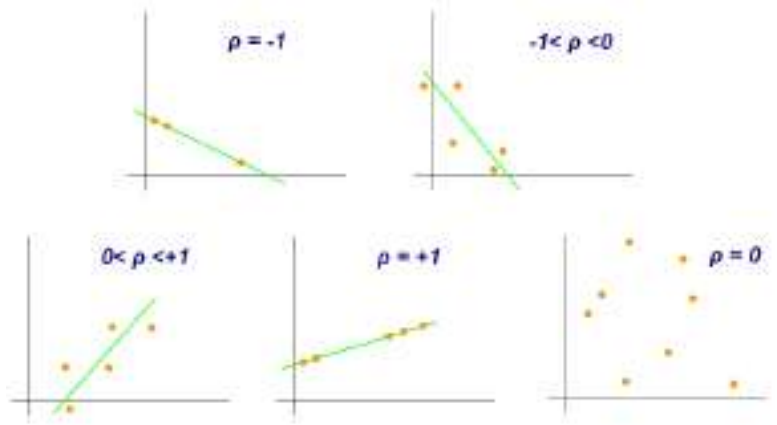
- Pearson's correlation

$$\rho(\mathbf{x},\mathbf{z}) = \frac{\sum_{j=1}^{n}(x_j - \bar{x})(z_j - \bar{z})}{\sqrt{\sum_{j=1}^{n}(x_j - \bar{x})^2}\sqrt{\sum_{j=1}^{n}(z_j - \bar{z})^2}}$$

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j$$

- Assuming that the data is centered

$$\rho(\mathbf{x},\mathbf{z}) = \frac{\sum_{j=1}^{n} x_j z_j}{\sqrt{\sum_{j=1}^{n} x_j^2}\sqrt{\sum_{j=1}^{n} z_j^2}}$$



Geometric interpretation?

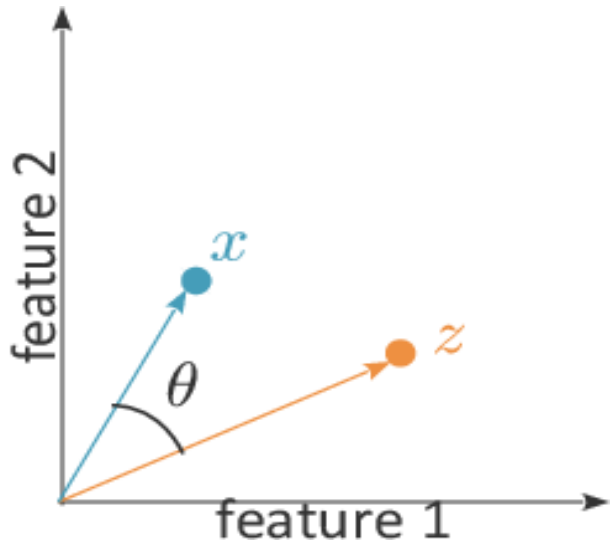
Pearson's Correlation

- Pearson's correlation (centered data)

$$\rho(\mathbf{x}, \mathbf{z}) = \frac{\sum_{j=1}^{n} x_j z_j}{\sqrt{\sum_{j=1}^{n} x_j^2} \sqrt{\sum_{j=1}^{n} z_j^2}} = \frac{\langle \mathbf{x}, \mathbf{z} \rangle}{\|\mathbf{x}\| \, \|\mathbf{z}\|} = \cos \theta$$

inner product

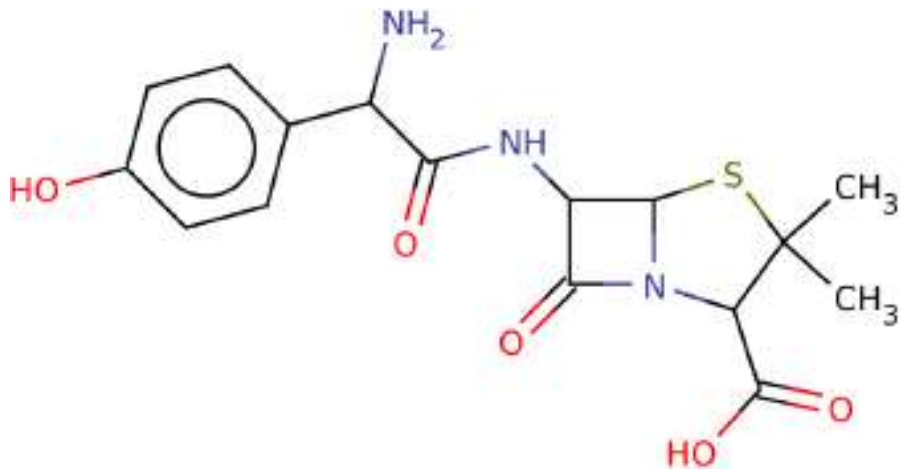
- Cosine similarity: the dot product can be used to measure similarities between vectors



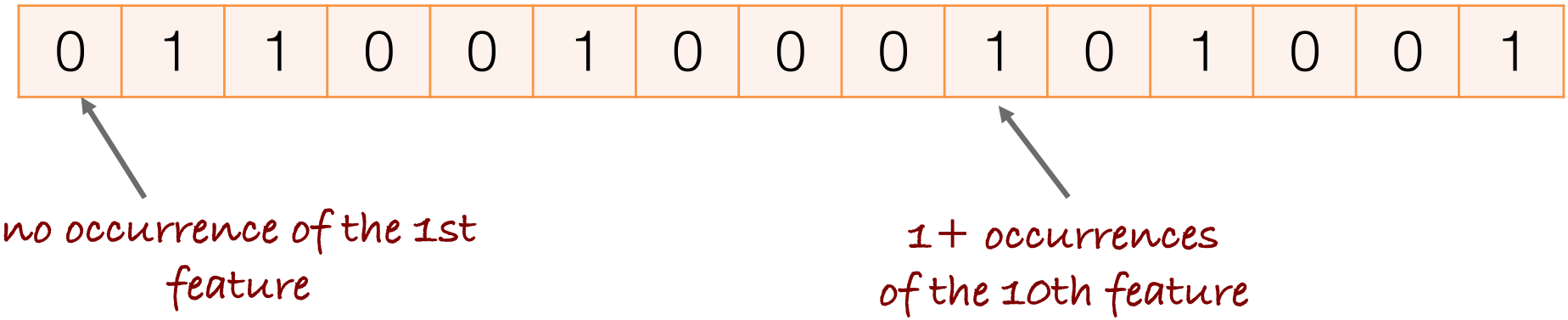
Categorical Features

- Represent objects as the list of presence/absence (or counts) of features that appear in it

- Example: molecules
  - Features: atoms and bonds of a certain type
  - C, H, S, O, N, …
  - O-H, O=C, C-N, ...



Binary Representation (1/2)



- Hamming distance between two binary representations
  - Number of bits that are different

$$d(\mathbf{x}^1, \mathbf{x}^2) = \sum_{j=1}^{n} (x_j^1 \text{ XOR } x_j^2)$$

  - Equivalent to the L1 distance

$$d(\mathbf{x}^1, \mathbf{x}^2) = \sum_{j=1}^{n} |x_j^1 - x_j^2|$$

XOR operator

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Binary Representation (2/2)

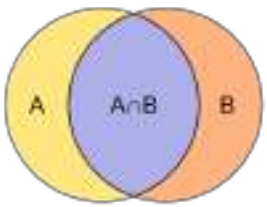
0	1	1	0	0	1	0	0	0	1	0	1	0	0	1

no occurrence of the 1st feature

1+ occurrences of the 10th feature

- Jaccard similarity (or Tanimoto similarity)
  - Number of shared features (normalized)

$$J(\mathbf{x}^1, \mathbf{x}^2) = \frac{\sum_{j=1}^{n}(x_j^1 \text{ AND } x_j^2)}{\sum_{j=1}^{n}(x_j^1 \text{ OR } x_j^2)}$$



AND operator

Input A	Input B	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR operator

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Jaccard index: intersection over union (Wikipedia: https://en.wikipedia.org/wiki/Jaccard_index)

Example

x = 010101001

y = 010011000

- Hamming distance

  x = 010101001

  y = 010011000

  Thus, d(x,y) = 3

- Jaccard similarity

  J = (# of *11*) / (# of *01* + # of *10* + # of *11*)

  = (2) / (1 + 2 + 2) = 2 / 5 = 0.4

Let’s go back to the kNN classifier

Nearest Neighbor Algorithm

- Training examples in the Euclidean space $\mathbf{x} \in \mathbb{R}^n$
- Idea: The label of a test data point is estimated from the known value of the nearest training example
  - The distance is typically defined to be the Euclidean one

Algorithm 1

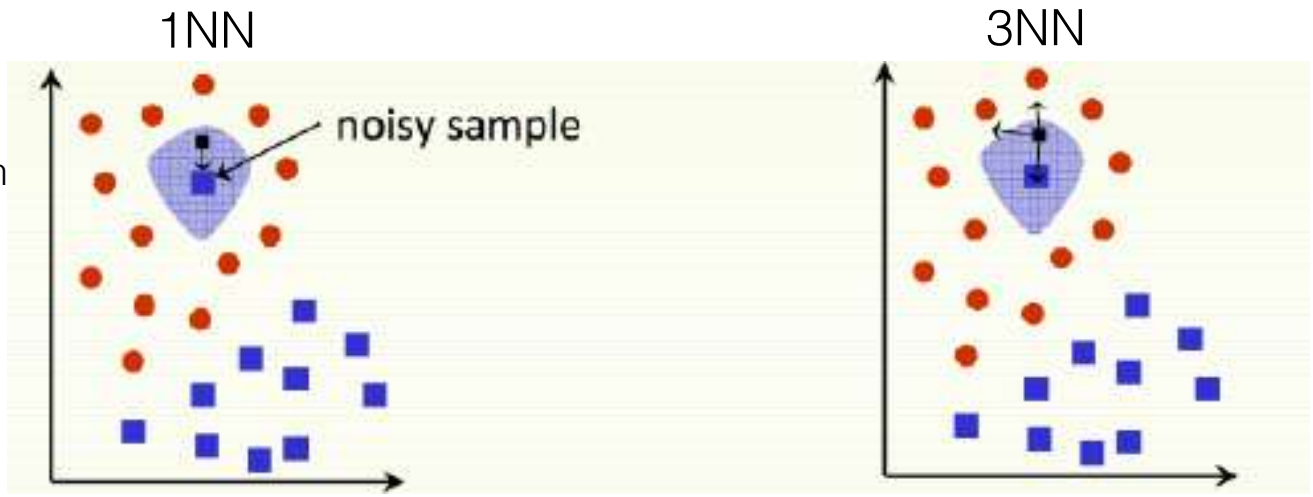
1. Find example (x*, y*) from the stored training set closest to the test instance x. That is:

$$\mathbf{x}^* = \text{argmin}_{\mathbf{x}^i \in \text{ train. set}} \; d(\mathbf{x}^i, \mathbf{x})$$

2. Output y(x) = y* (The output label)

k-Nearest Neighbors (kNN) Algorithm

Every example in the blue shaded area will be misclassified as the blue class



Every example in the blue shaded area will be classified correctly as the red class

- Algorithm 1 is sensitive to mis-labeled data ('class noise')
- Consider the vote of the k nearest neighbors (majority vote)

Algorithm 2

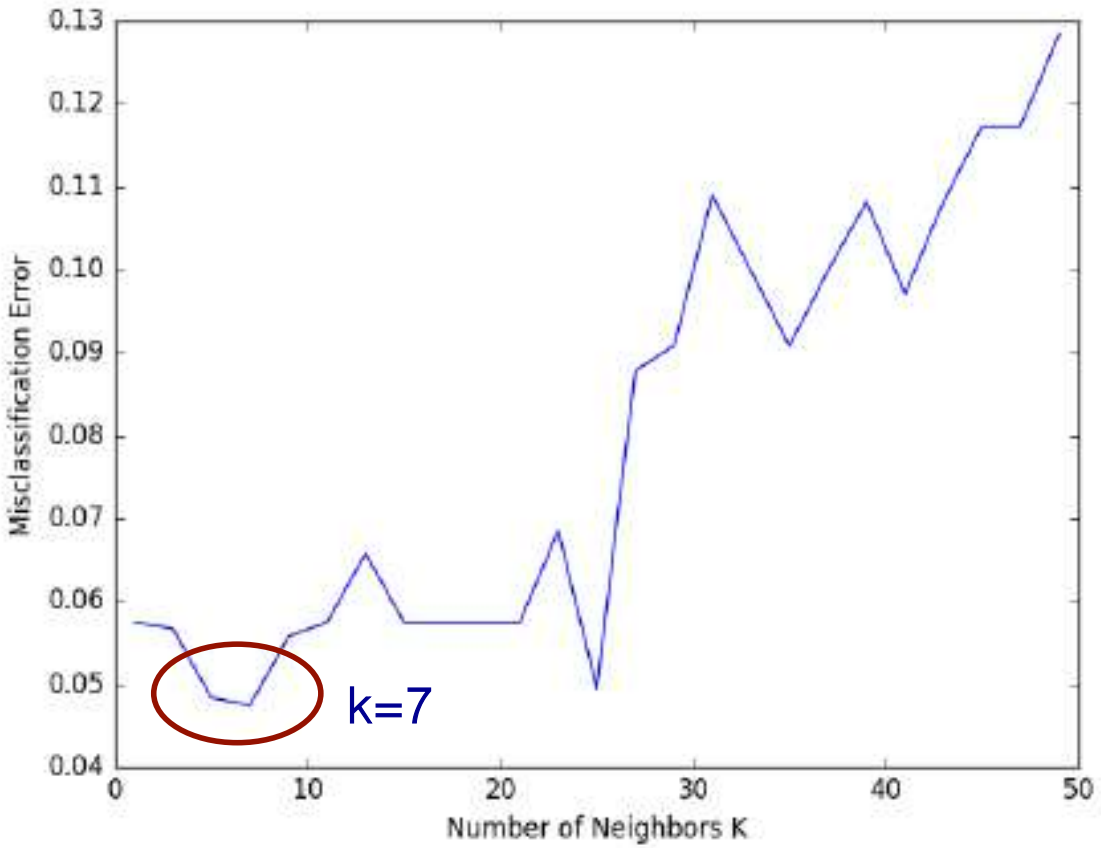
- Find k examples (x^{*i}, y^{*i}), i=1,…,k closest to the test instance x
- The output is the majority class

Choice of Parameter k (1/2)

- Small k: noisy decision
  - The idea behind using more than 1 neighbors is to average out the noise
- Large k
  - May lead to better prediction performance
  - If we set k too large, we may end up looking at samples that are not neighbors (are far away from the point of interest)
  - Also, computationally intensive. Why?
  - Extreme case: set k=m (number of points in the dataset)
    - For classification: the majority class
    - For regression: the average value

Choice of Parameter k (2/2)

Set k by cross validation, by examining the misclassification error



Rule of thumb:

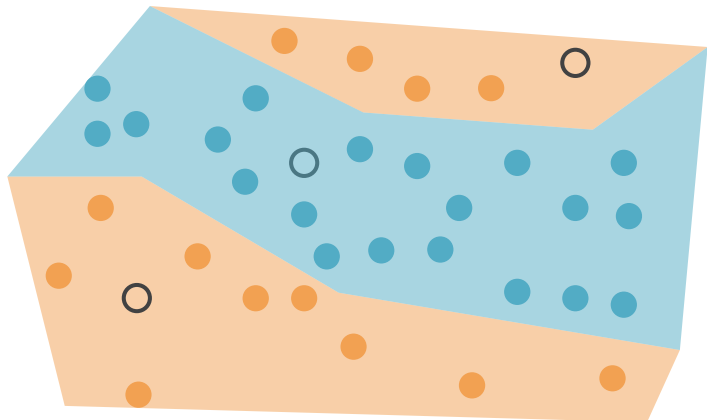
$$k = \sqrt{m}$$

m: # of training instances

Source: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Advantages of kNN

- Training is very fast
  - Just store the training examples
  - Can use smart indexing procedures to speed-up testing
- The training data is part of the 'model'
  - Useful in case we want to do something else with it
- Quite robust to noisy data
  - Averaging k votes
- Can learn complex functions (implicitly)



Drawbacks of kNN

- Memory requirements
  - Must store all training data
- Prediction can be slow (will figure it out by yourself in the lab)
  - Complexity of labeling 1 new data point: O(knm)
  - But kNN works best with lots of samples
  - Can we further improve the running time?
    - Efficient data structures (e.g., k-D trees)
    - Approximate solutions based on hashing
- High dimensional data and the curse of dimensionality
  - Computation of the distance in a high dimensional space may become meaningless
  - Need more training data
  - Dimensionality reduction

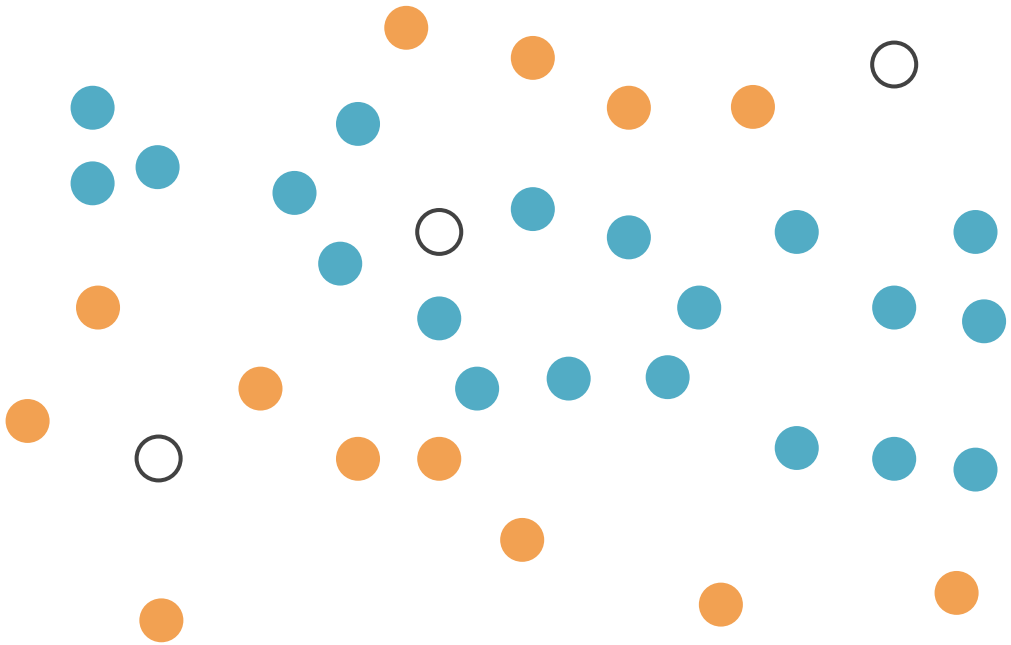
Wikipedia: https://en.wikipedia.org/wiki/K-d_tree

kNN – Some More Issues

- Normalize the scale of the attributes
- Simple option: linearly scale the range of each feature to be, e.g., in the range of [0,1]
- Linearly scale each dimension to have 0 mean and variance 1
  - Compute the mean μ and variance σ^2 for an attribute x_j and scale: $(x_j - \mu)/\sigma^2$

Decision Boundary of kNN

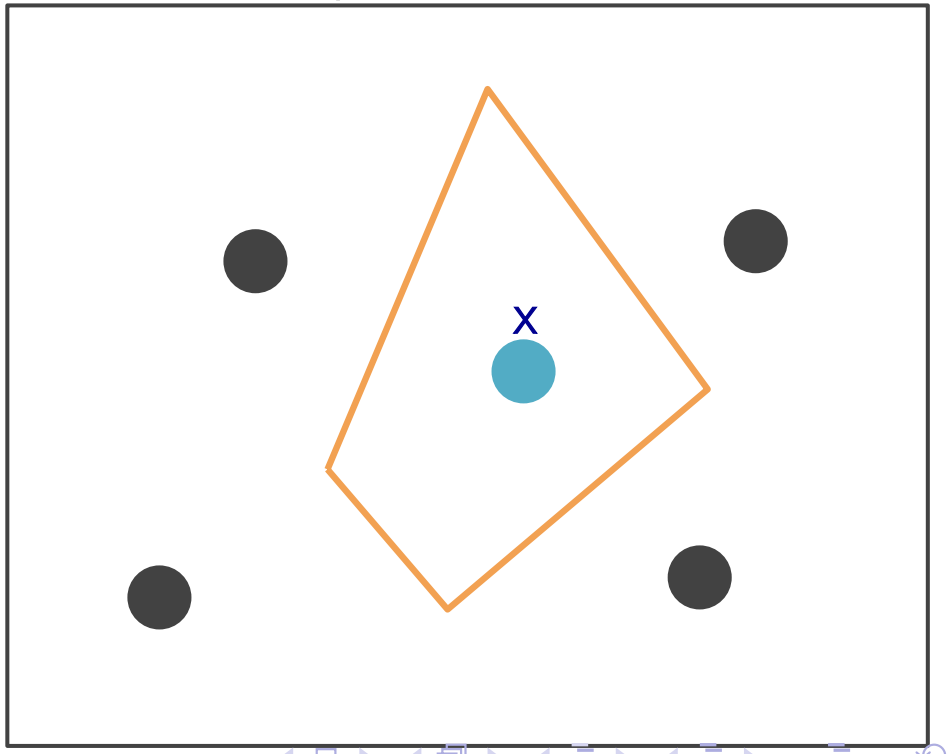
- Decision boundary in classification
  - Line separating the positive from negative regions
- What decision boundary is the kNN building?
  - The nearest neighbors algorithm does not explicitly compute decision boundaries, but those can be inferred



Voronoi Tessellation

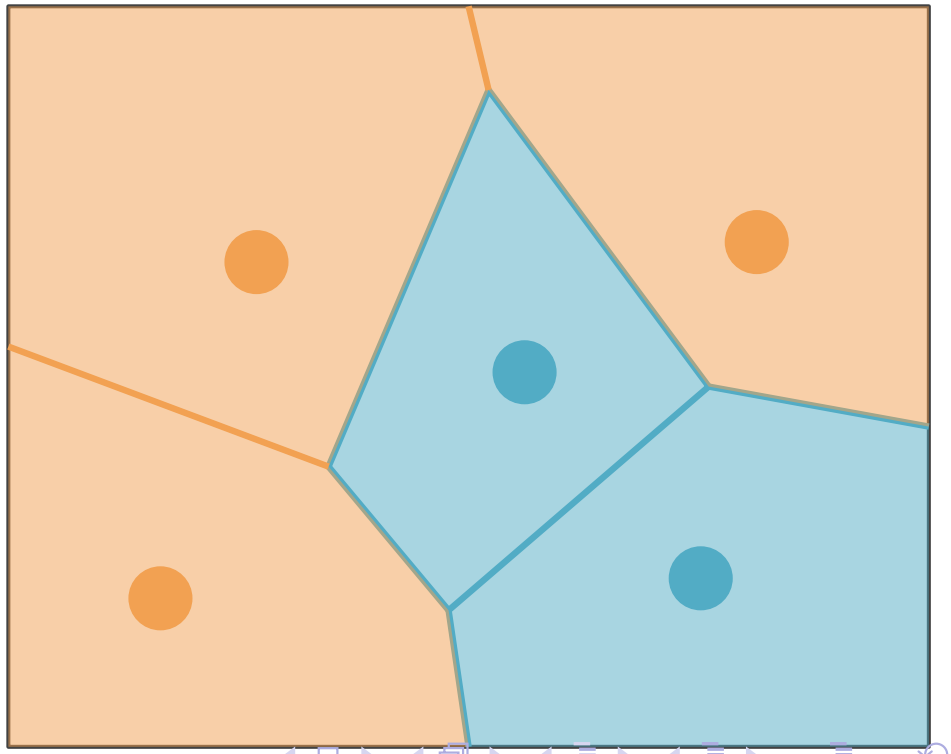
Consider the case of 1NN

- Voronoi cell of x:
  - Set of all points of the space closer to x than any other point of the training set
  - Polyhedron
- Voronoi tessellation (or diagram) of the space
  - Union of all Voronoi cells



Voronoi Tessellation

- The Voronoi diagram defines the decision boundary of the 1NN
- The kNN algorithms also partitions the space but in a more complex way



Wikipedia: https://en.wikipedia.org/wiki/Voronoi_diagram

kNN Variants

- Weighted kNN
  - Weight the vote of each neighbor x_i according to the distance to the test point x

$$w_i = \frac{1}{d(x, x_i)^2}$$

  - Other kernel functions can be used to weight the distance of neighbors

Source: https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf

scikit-learn

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html