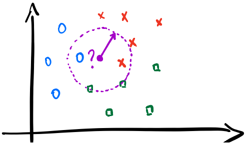
K-NN: Nearest Neighbour Classification and Regression



Another name:
- Exemplar-based Method

- One of the earliest and simplest learning algorithms
- Memorize the training set
- Predict label of any new instance based on the labels of its "closest" neighbours in the training set (instance-based)
- Assumption: Labels of nearby points in the feature space are the same
- Does not identify a predictor function from a class of specified functions based on the training dataset (non-parametric)

Nearest Neighbour Classification

Labelled training set: $(x_1, y_1), \ldots, (x_m, y_m), \quad x_i \in \mathbb{R}^d, \quad y_i \in \mathcal{C}$

Nearest neighbour: For new instance x, define $nn(x) \in [m] \triangleq \{1, \ldots, m\}$:

$$nn(x) = \operatorname*{argmin}_{i \in [m]} \|x - x_i\|^2$$

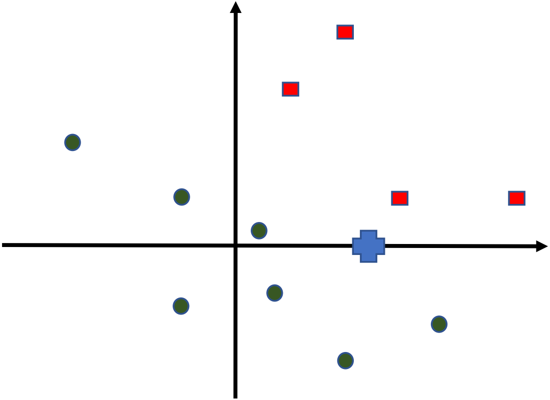
returns the index of the training example nearest to x

Classification rule:

$$y = f(x) = y_{nn(x)}$$

Example

K-NN is not a linear classifier:
- decision boundary on the feature space is not a linear function and positive/negative examples are not separated by an hyperplane.



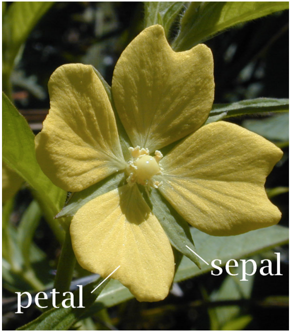
Will be classified as red!

Minimal training, but expensive testing

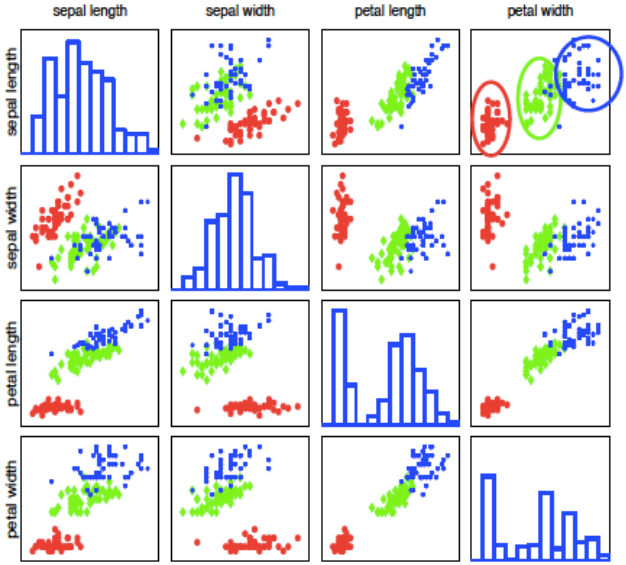
Famous Example: Iris Classification



- Three types:
  setosa, versicolor, and virginica
- Dataset introduced originally by statistician Ronald Fisher in 1936
- 50 observations from each three species
- Four features measured: length and width of sepals and petals



Iris Classification



Example: Iris Classification with Two Features

Training data

ID (i)	petal width (x_1)	sepal length (x_2)	type
1	0.2	5.1	setosa
2	1.4	7.0	versicolor
3	2.5	6.7	virginica

New flower:

petal width $= 1.8$, sepal length $= 6.4$

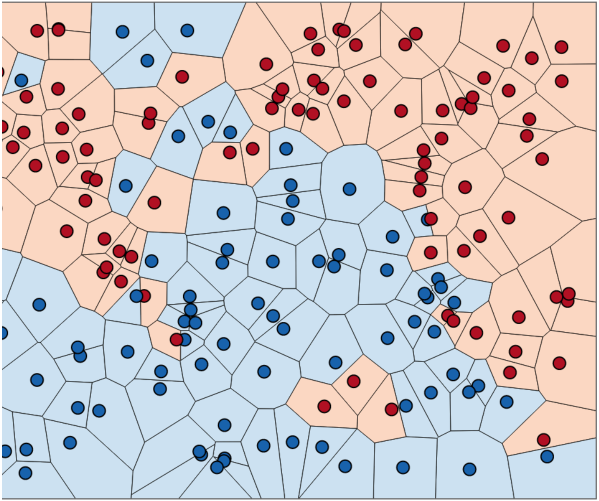
ID (i)	Distance
1	1.75
2	0.72
3	0.76

Predicted category: versicolor

Decision Boundary

We can determine the label of any point in the space.

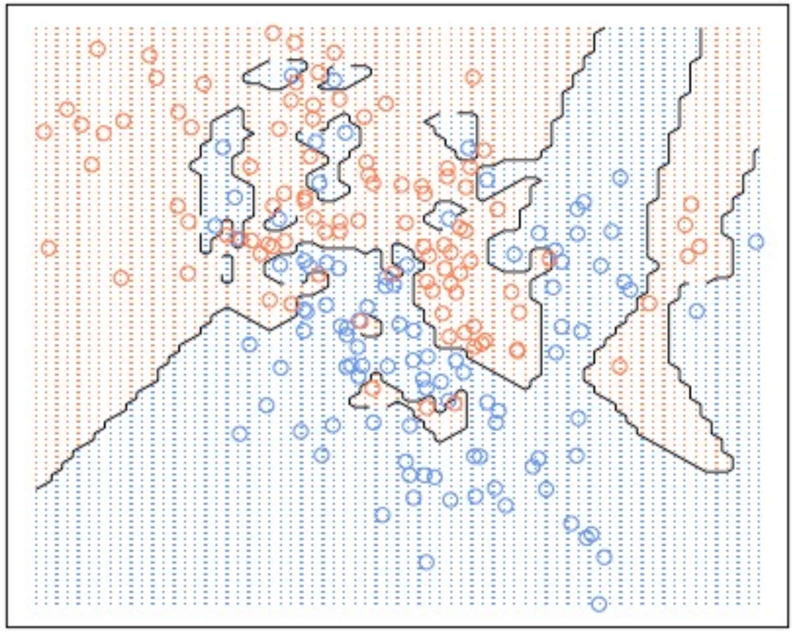
This leads to decision boundaries



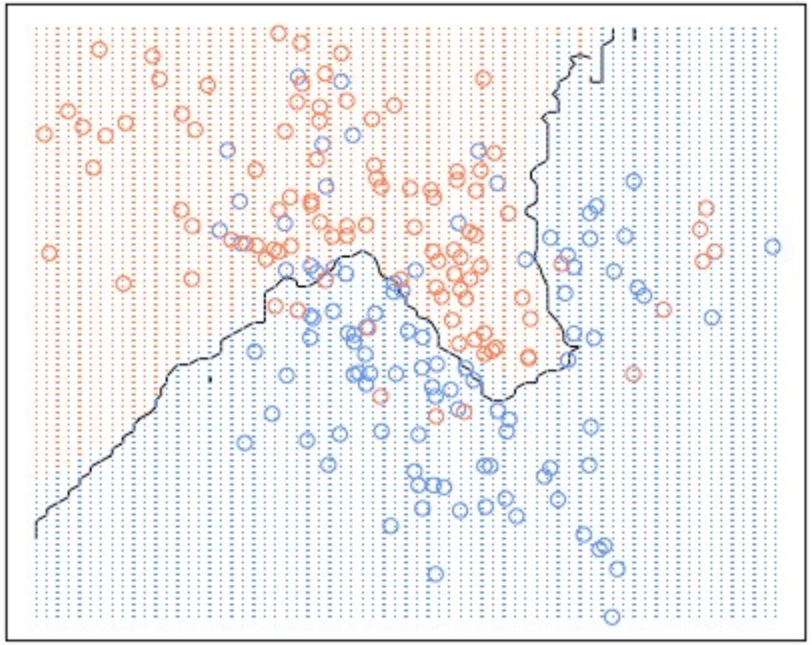
1-NN is similar to $\gamma = \infty$ in the RBF-Kernel

$$K(x, x') = \exp\left(-\gamma \underbrace{\|x - x'\|^2}_{\textit{radial}} \right)$$

nearest neighbour (k = 1)



20-nearest neighbour



K-nearest Neighbour (K-NN) Classification

Consider K nearest neighbours:

- Nearest neighbour: $nn_1(x) = \text{argmin}_{i\in[m]} \|x - x_i\|^2$
- 2nd nearest neighbour: $nn_2(x) = \text{argmin}_{i\in[m]\setminus\{nn_1(x)\}} \|x - x_i\|^2$
- 3rd nearest neighbour: $nn_3(x) = \text{argmin}_{i\in[m]\setminus\{nn_1(x),nn_2(x)\}} \|x - x_i\|^2$

Set of K-nearest neighbours:

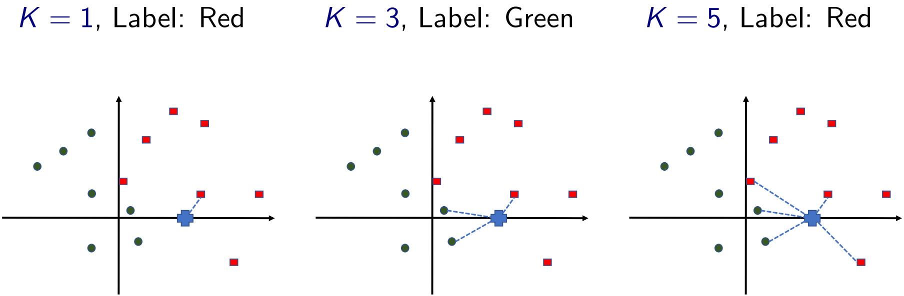
$$knn(x) = \{nn_1(x), \ldots, nn_K(x)\}$$

Classification rule:Majority vote among K-NNs:

$$v_c = \sum_{i\in knn(x)} \mathbb{I}(y_i = c), \quad \forall c \in \mathcal{C}$$

$$y = f(x) = \underset{c\in\mathcal{C}}{\text{argmax}}\, v_c$$

Example

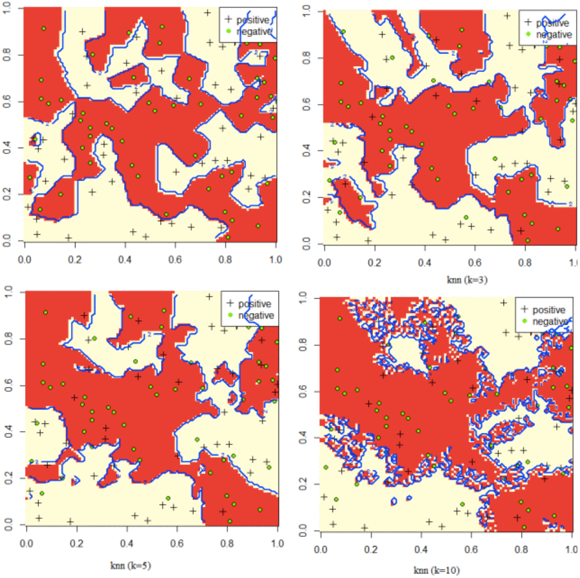


- One way to break a tie in k-NN is to decrease/increase k by 1 until you have broken the tie.
- If weighting is uniform, another way is to compute the sum of distances between our point and its neighbors from each class.

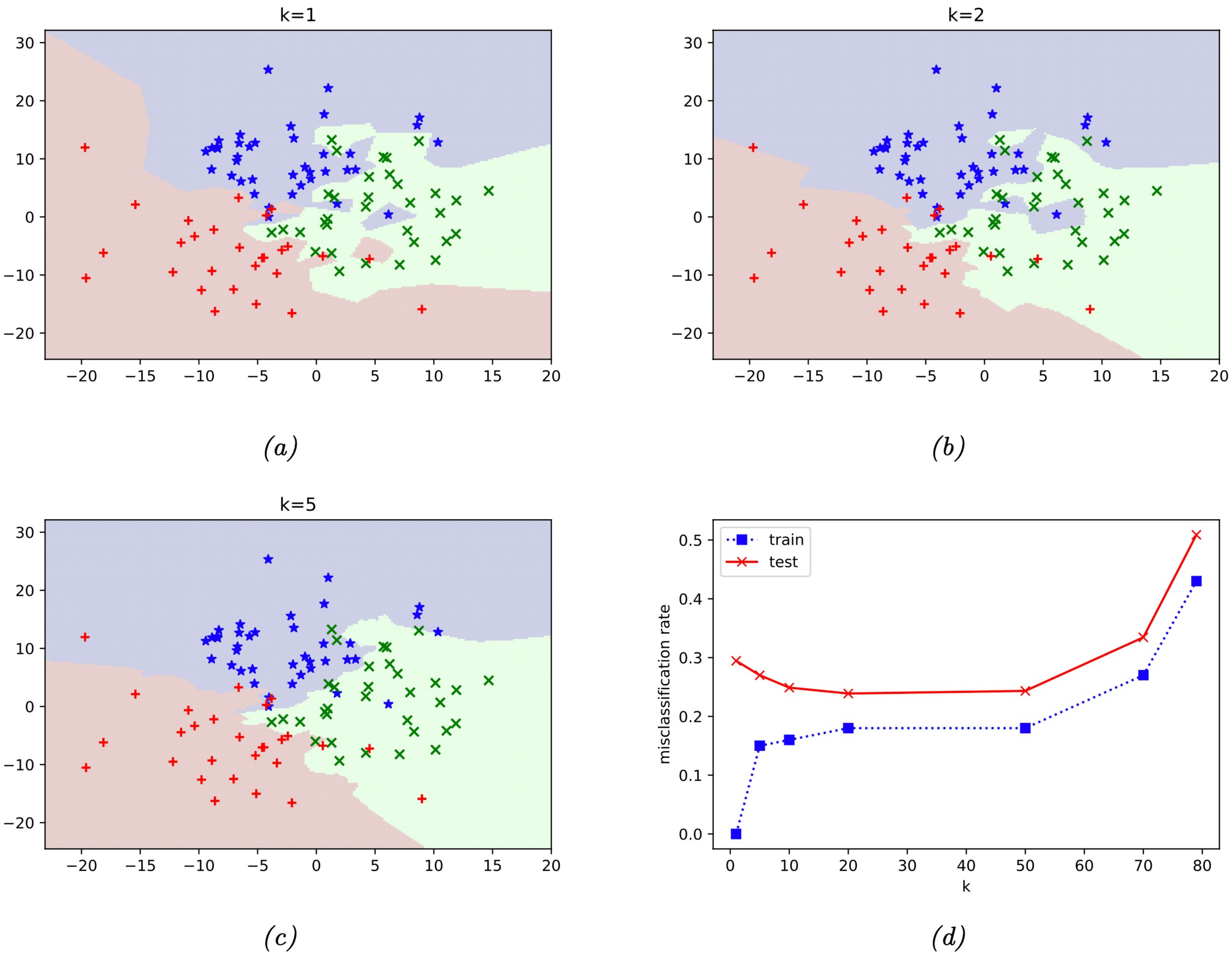
- K: form 1 to infinity … usually means … from Overfitting (k=1) to Underfitting (K>>1)

Increasing K

As K increases, decision boundaries become smoother



Decision boundaries induced by a K-NN classifier.

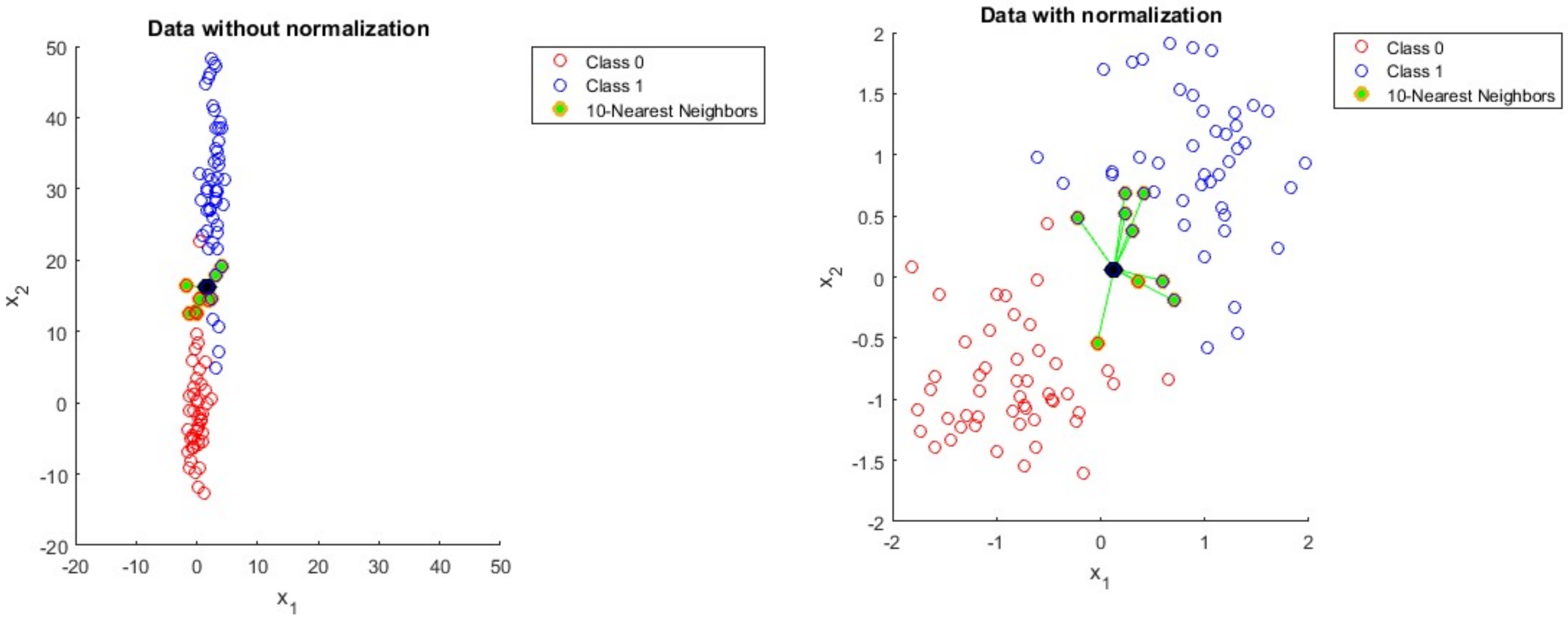


https://probml.github.io/pml-book/book1.html

Refinements

- Choosing K: Best K should be determined by considering a range of K values:
  - Too small K: overfitting to data
  - Too large K: too much influence from other classes
- Can be problematic when dataset is clustered, i.e., K neighbours has very different distances. Weighting can be used so that nearer neighbours have more influence on the vote.
- Distance measure should be chosen carefully
  - For example: house price dataset: area ranges around 80-200 m^2, whereas number of rooms $\sim 1 - 6$

[Additional Slides]



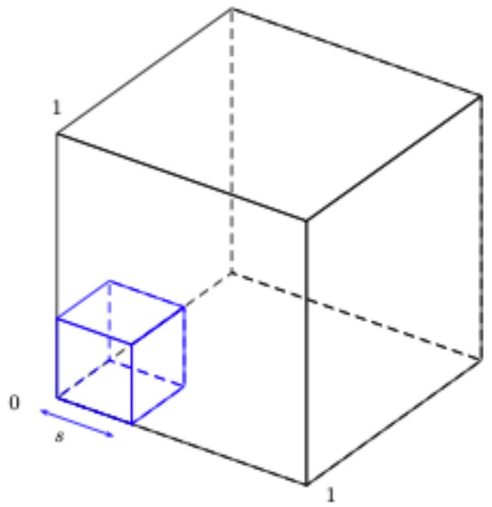
https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

Distance Measure

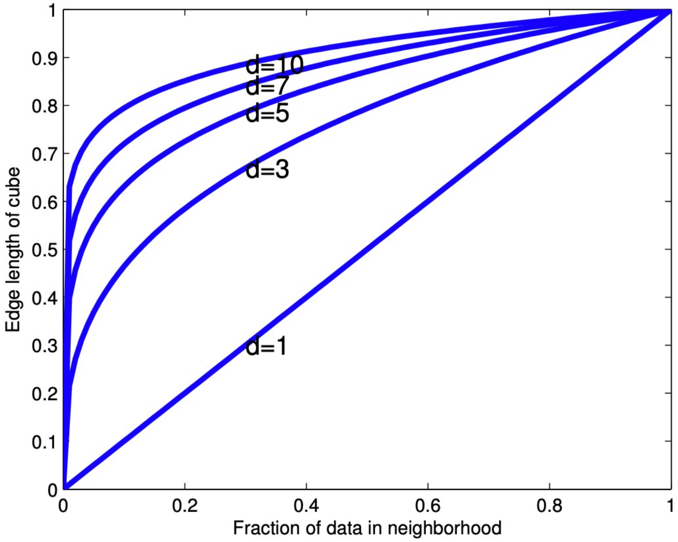
Different distance measures can be used, i.e.,

$$\|x - x'\|_p = \left(\sum_{j=1}^{d} |x_j - x'_j|^p \right)^{1/p}$$

Hard to stay close in higher dimensions!



(a)



(b)

Figure 16.3: Illustration of the curse of dimensionality. (a) We embed a small cube of side s inside a larger unit cube. (b) We plot the edge length of a cube needed to cover a given volume of the unit cube as a function of the number of dimensions. Adapted from Figure 2.6 from [HTF09]. Generated by code at figures.probml.ai/book1/16.3.

https://probml.github.io/pml-book/book1.html

K-NN Regression

For regression, value of the new instance is the (weighted) average of the values of the K nearest neighbours.

K-NN is a local-method:
good at regressing points that lie within the interval of training data.

Pros and Cons of K-NN Classification/ Regression

- Advantages
  - Easy to implement: just compute distances, no complex parameter tuning.
  - No training necessary. New training data can be added easily.
  - Has strong theoretical guarantees.
- Disadvantages
  - Computationally intensive for large-scale problems: $O(m \cdot d)$ for labelling a data point. Can be reduced in practice:
    - ⋆ Presort training data employing efficient data structures.
    - ⋆ Use approximate distance and search methods
    - ⋆ Remove redundant data
  - Large memory: we need to keep all the training data (non-parametric).
  - Choosing the right distance measure and K can be involved.

A note on Kernelised k Nearest Neighbour

- It's known that:
  - For any random set of finite points, with high probability, we can map these points to a higher dimension and make these points linearly separable.
  - E.g., using some kernels.

- Does Linear separability (esp. in higher dimensions) imply that points from the same class will get closer than the points from different classes?

https://stats.stackexchange.com/questions/44166/kernelised-k-nearest-neighbour