Machine Learning open course Lecture 1: intro to ML

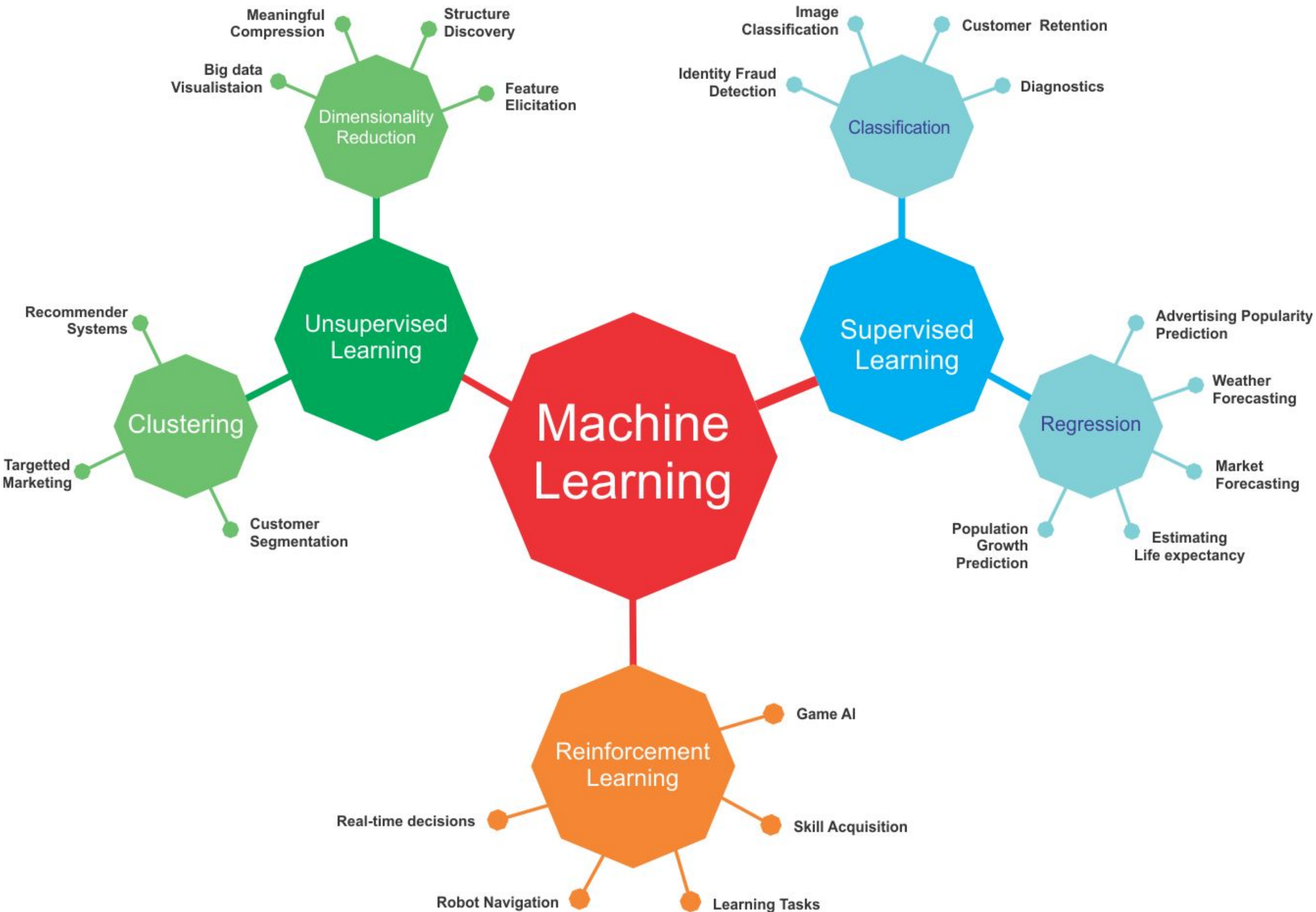
MIPT, Fall 2019

Outline

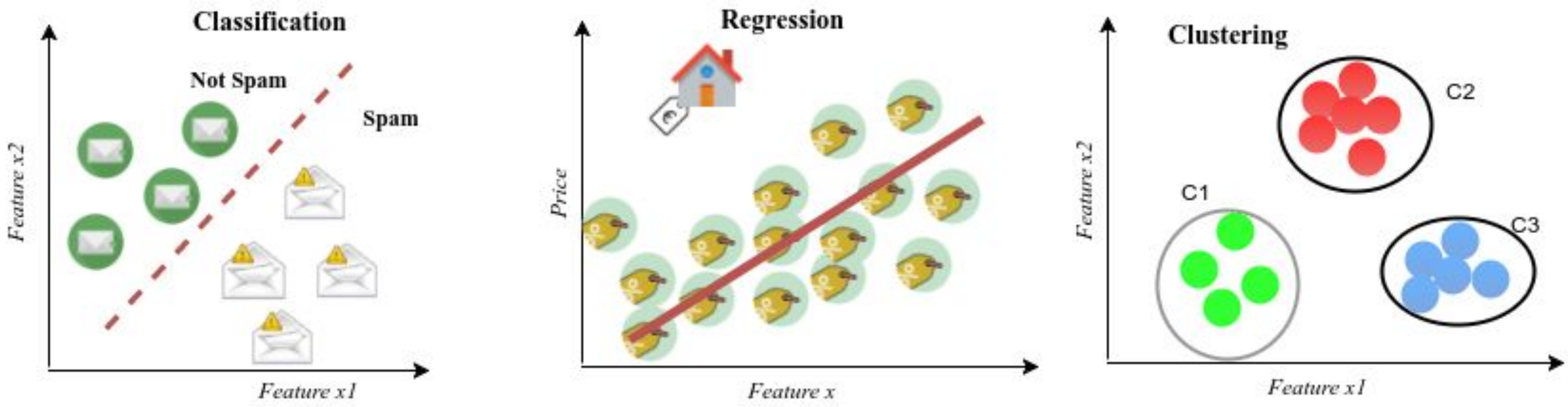
1. Machine Learning tasks overview
2. General supervised learning problem statement
3. Models evaluation and cross validation
4. kNN method in classification and regression

Variety of tasks in Machine Learning

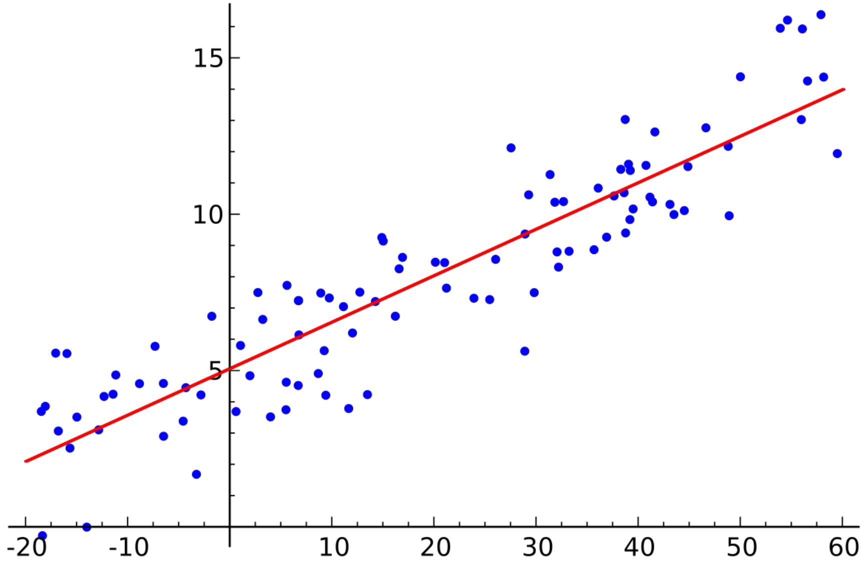
- Supervised learning
    - Classification
    - Regression
- Unsupervised learning
    - Clustering
    - Anomaly detection
    - Dimensionality reduction
- Other cool stuff


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Identity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Population Growth Prediction
Estimating Life expectancy
Reinforcement Learning
Game AI
Real-time decisions
Skill Acquisition
Robot Navigation
Learning Tasks

ML tasks


Classification
Not Spam
Spam
Feature x2
Feature x1
Regression
Price
Feature x
Clustering
C1
C2
C3
Feature x2
Feature x1

- Regression task



Estimated (or predicted) Y value for observation i

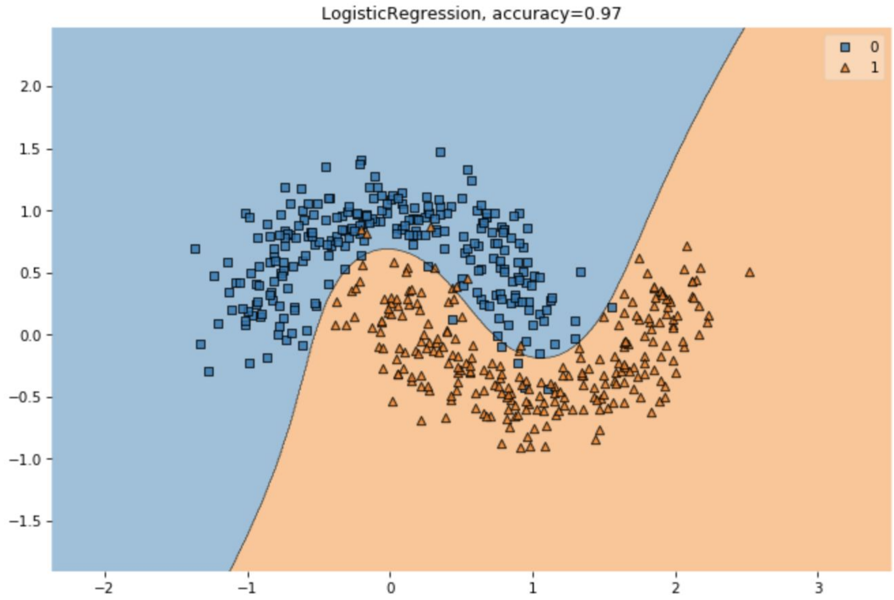
Estimate of the regression intercept

Estimate of the regression slope

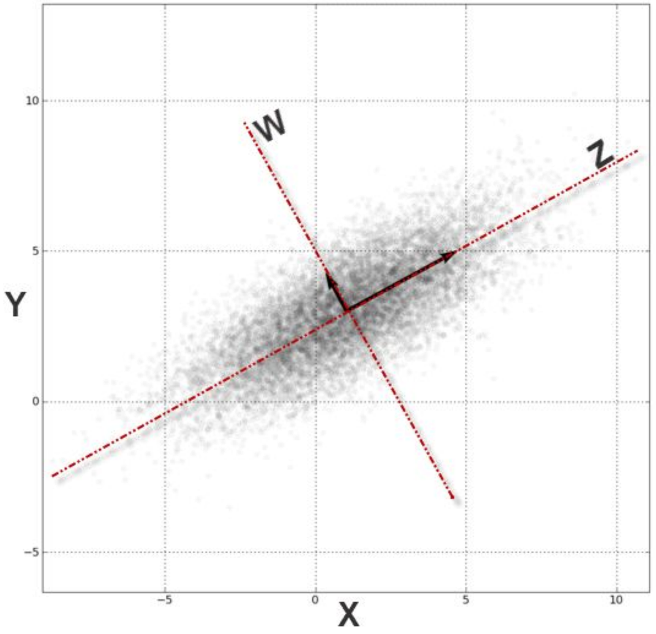
Value of X for observation i

$$\hat{Y}_i = b_0 + b_1 X_i$$

- Regression task
- Classification task



- Regression task
- Classification task
- Dimensionality reduction task



Supervised learning problem statement

Let's denote:

- Training set $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, where
  - $(\mathbf{x} \in \mathbb{R}^p,\ y \in \mathbb{R}\ )$ for regression,
  - $\mathbf{x}_i \in \mathbb{R}^p$,$y_i \in \{+1, -1\}$ for binary classification,
- Model $f(\mathbf{x})$ that predicts some target for every object,
- Loss function $Q(\mathbf{x}, y, f)$ that should be minimized.

Supervised learning problem statement

Let's denote:

- Training set $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, where
  - $(\mathbf{x} \in \mathbb{R}^p,\ y \in \mathbb{R}\ )$ for regression,
  - $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{+1, -1\}$ for binary classification,
- Model $f(\mathbf{x})$ that predicts some target for every object,
- Loss function $Q(\mathbf{x}, y, f)$ that should be minimized.

In this form the problems will be stated in future as well.

Minimizing loss function is great. But how not to overfit?

Supervised learning problem statement

Let's denote:

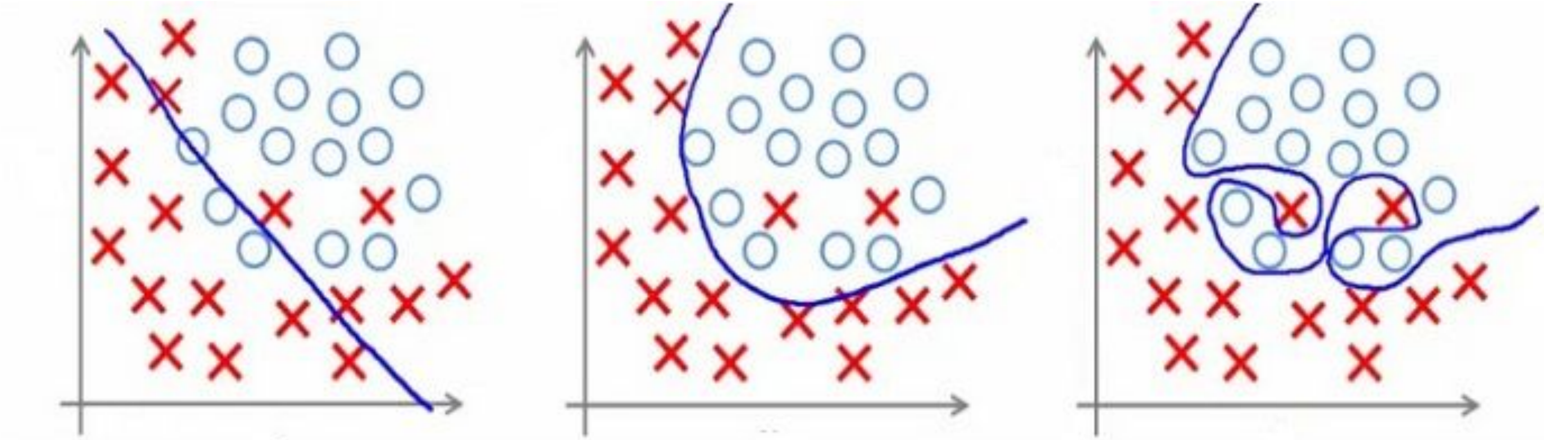
- Training set $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, where
  - $(\mathbf{x} \in \mathbb{R}^p,\ y \in \mathbb{R})$ for regression,
  - $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{+1, -1\}$ for binary classification,
- Model $f(\mathbf{x})$ that predicts some target for every object,
- Loss function $Q(\mathbf{x}, y, f)$ that should be minimized.

In this form the problems will be stated in future as well.

Minimizing loss function is great. But how not to overfit?

Stop, what is overfitting?

Overfitting vs. underfitting



Under-fitting

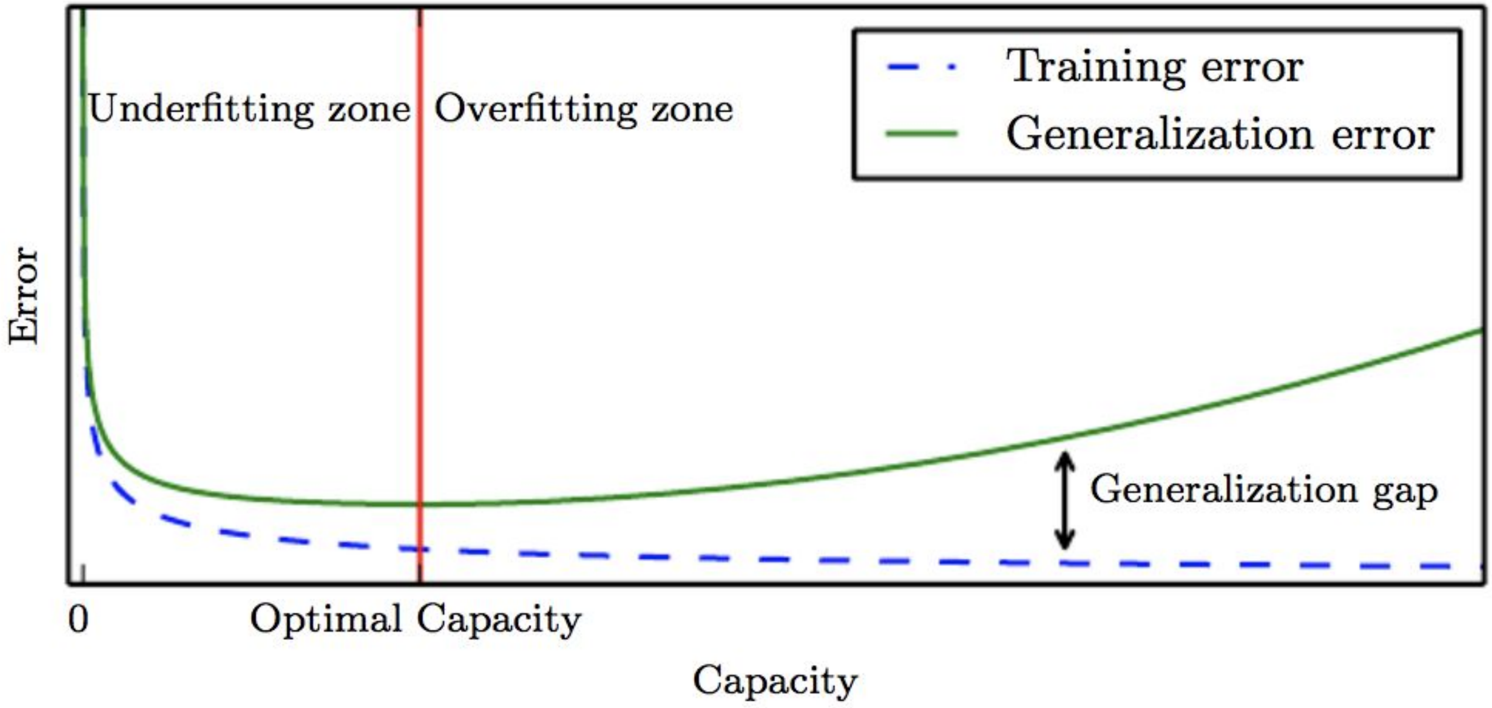
(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Overfitting vs. underfitting



Overfitting vs. underfitting

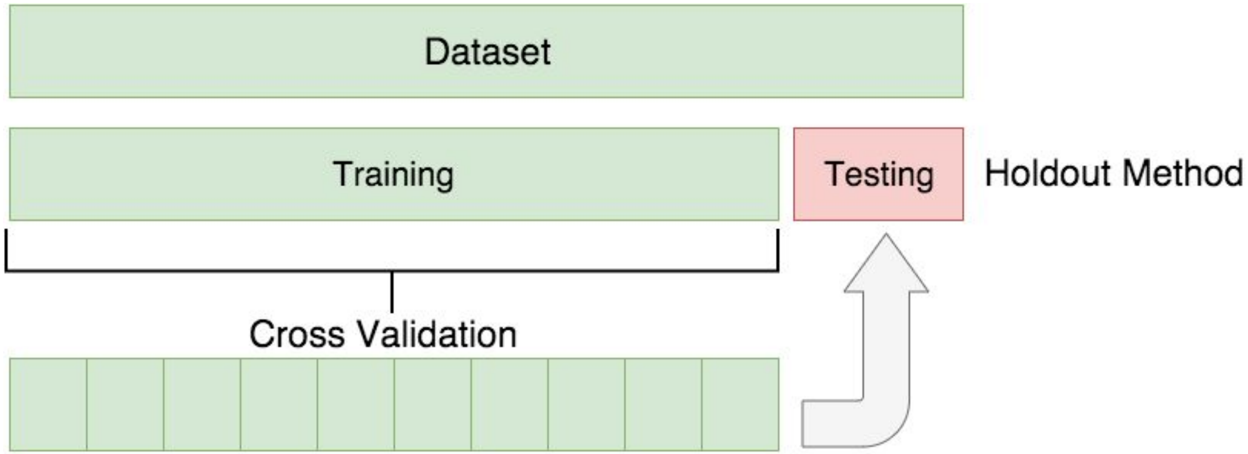
- We can control overfitting / underfitting by altering model’s capacity (ability to fit a wide variety of functions):
- select appropriate hypothesis space
- learning algorithm’s effective capacity may be less than the representational capacity of the model family

Evaluating the quality

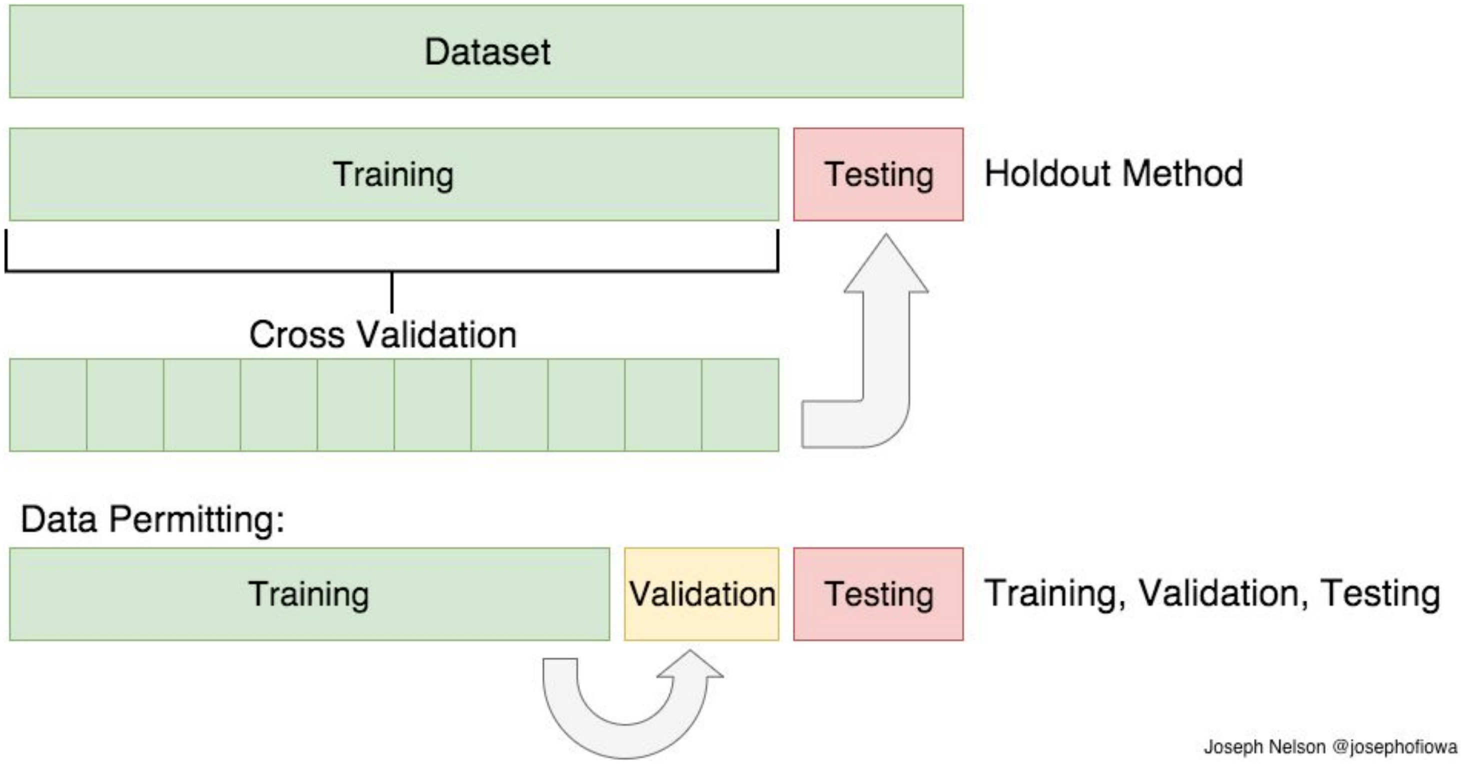
Evaluating the quality

Is it good enough?

Evaluating the quality



Evaluating the quality

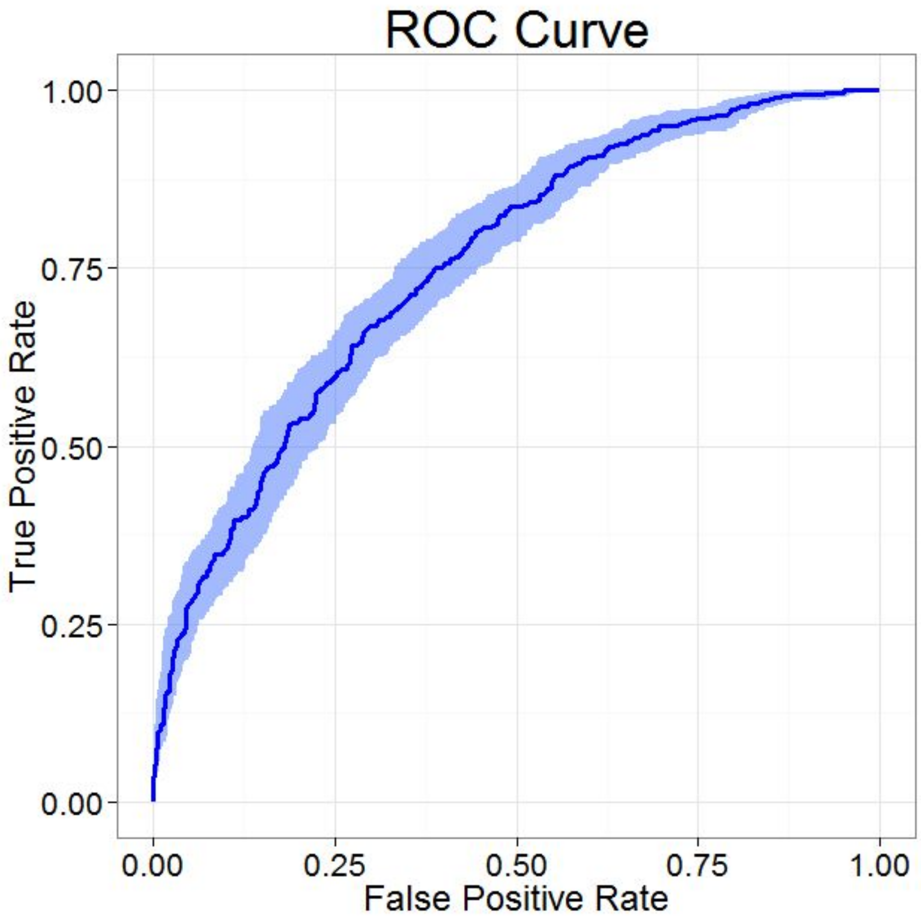


Cross-validation

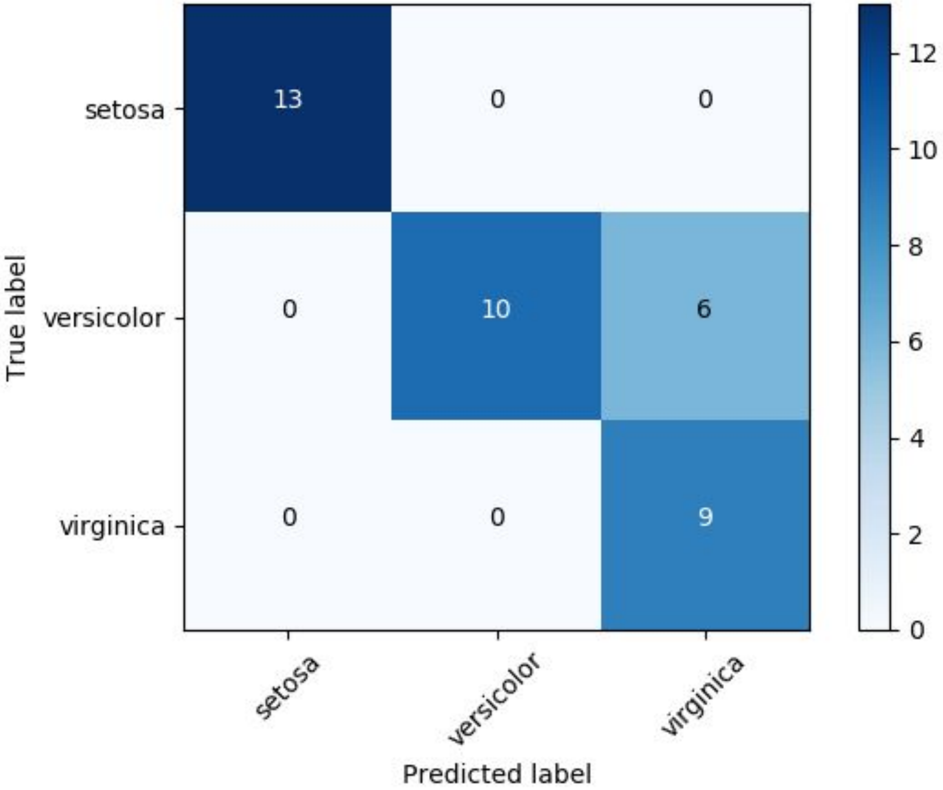
Measuring the quality in classification

- Accuracy
- ROC-AUC
- Precision
- Recall
- Confusion Matrix
- ...

ROC AUC


ROC Curve
1.00
0.75
0.50
0.25
0.00
True Positive Rate
0.00
0.25
0.50
0.75
1.00
False Positive Rate

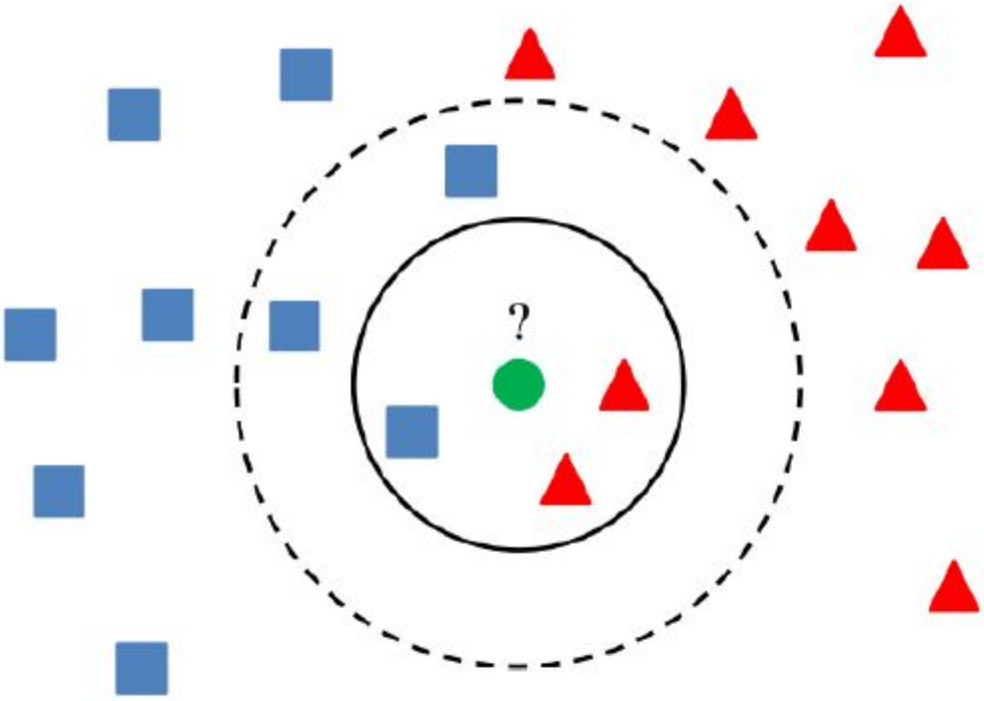
Confusion Matrix



Measuring quality in regression

- Mean Absolute Error (MAE)
- Mean Square Error (MSE)
- R2 score
- MAPE
- SMAPE
- ...

kNN - k Nearest Neighbours



k Nearest Neighbors Method

1. Calculate the distance to each of the samples in the training set.
2. Select samples from the training set with the minimal distance to them.
3. The class of the test sample will be the most frequent class among those nearest neighbors.

k Nearest Neighbors Method

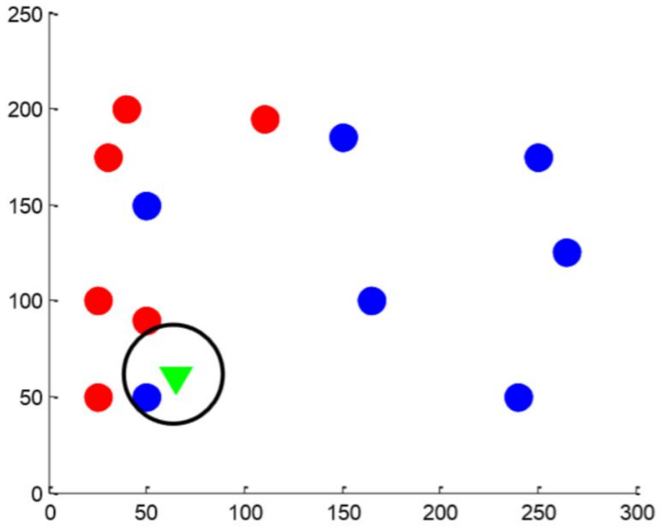
1. Calculate the distance to each of the samples in the training set.
2. Select samples from the training set with the minimal distance to them.
3. The class of the test sample will be the most frequent class among those nearest neighbors.

- kNN can be used for regression as well.
- And for clustering - it’s known as kMeans.

How to make it better?

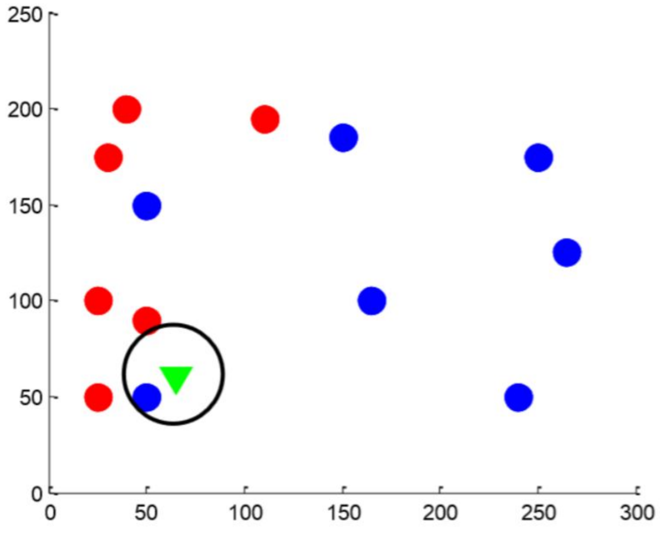
- The number of neighbors k

kNN classification

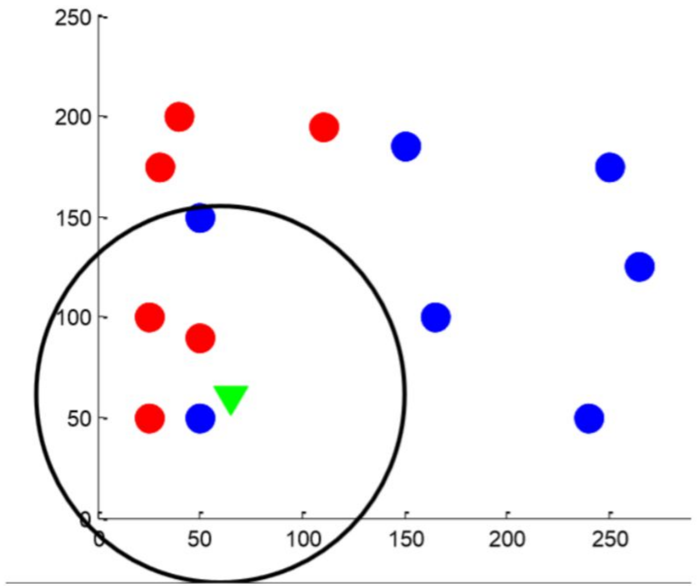


k = 1

kNN classification



k = 1



k = 5

How to make it better?

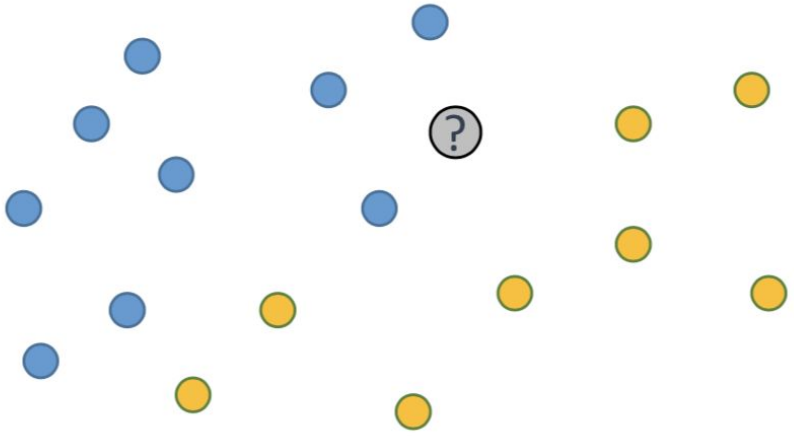
- The number of neighbors k
- The distance measure between samples
    a. Hamming
    b. Euclidean
    c. cosine
    d. Minkowski distances
    e. etc.

How to make it better?

- The number of neighbors k
- The distance measure between samples
    a. Hamming
    b. Euclidean
    c. cosine
    d. Minkowski distances
    e. etc.
- Weights of neighbors

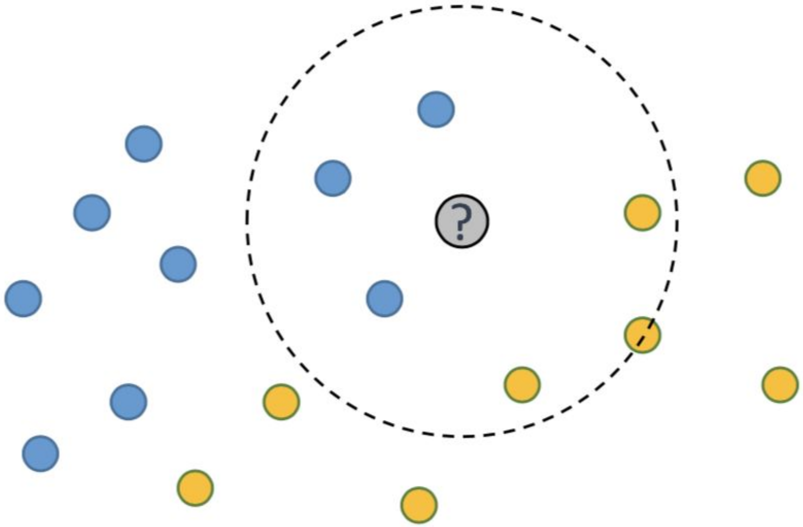
Weighted kNN

k = 6



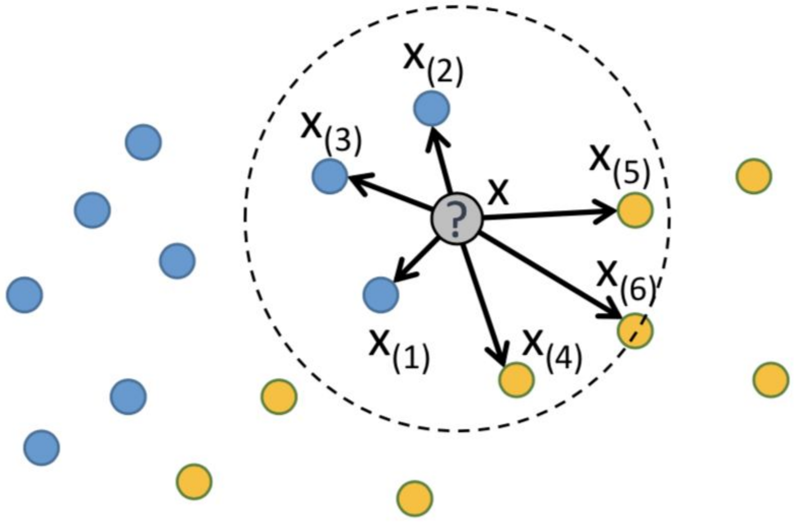
Weighted kNN

k = 6



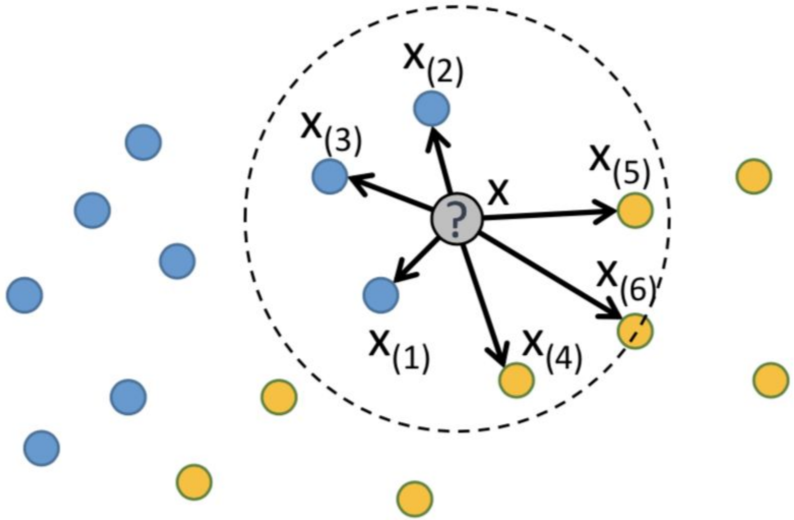
Weighted kNN

k = 6



Weighted kNN

k = 6

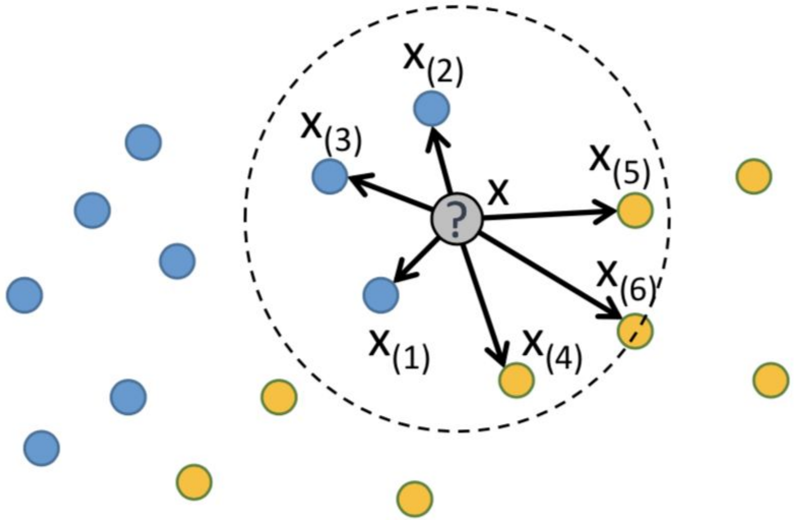


- Weights can be adjusted according to the neighbors order.

$$w(\mathbf{x}_{(i)}) = w_i$$

Weighted kNN

k = 6



- Weights can be adjusted according to the neighbors order,

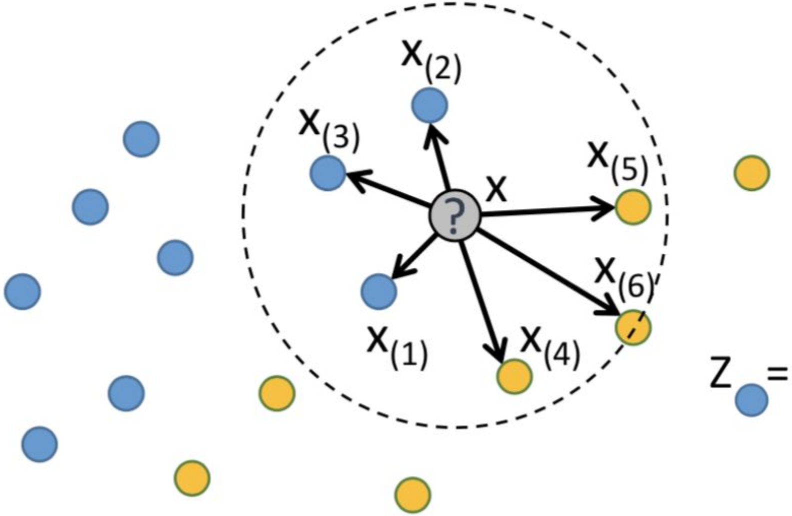
$$w(\mathbf{x}_{(i)}) = w_i$$

- or on the distance itself

$$w(\mathbf{x}_{(i)}) = w(d(\mathbf{x}, \mathbf{x}_{(i)}))$$

Weighted kNN

k = 6



- Weights can be adjusted according to the neighbors order,

$$w(\mathbf{x}_{(i)}) = w_i$$

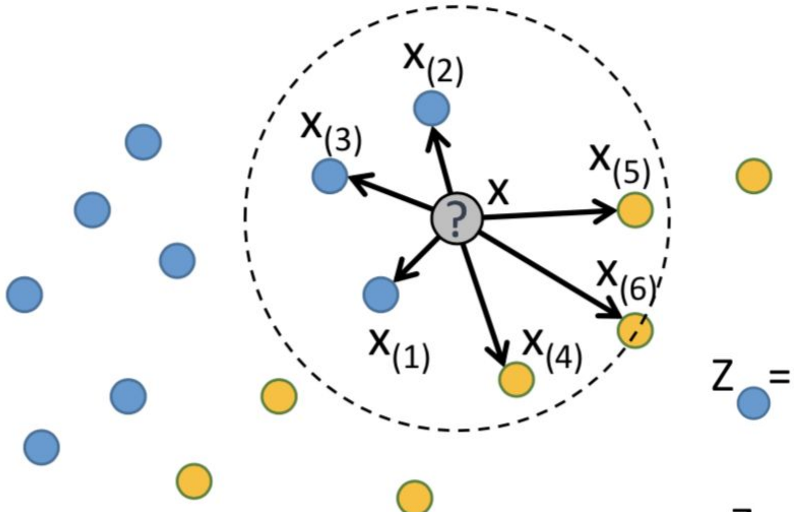
- or on the distance itself

$$w(\mathbf{x}_{(i)}) = w(d(\mathbf{x}, \mathbf{x}_{(i)}))$$

$$Z = \frac{\boxed{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

Weighted kNN

k = 6



- Weights can be adjusted according to the neighbors order,

$$w(\mathbf{x}_{(i)}) = w_i$$

- or on the distance itself

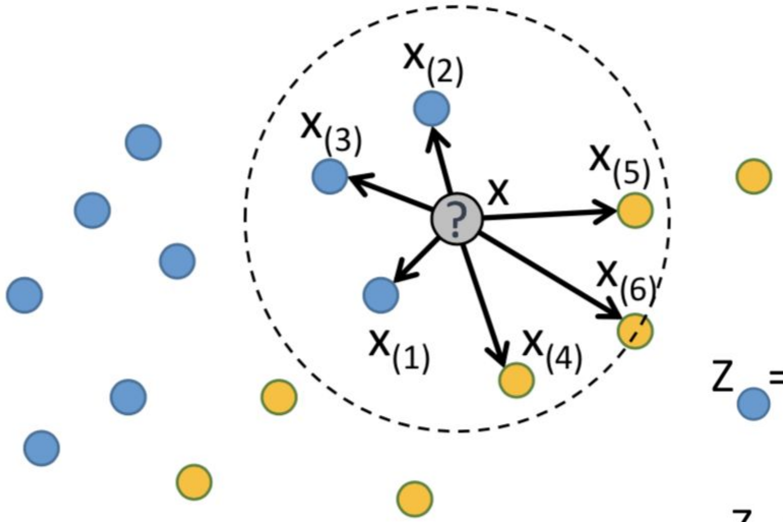
$$w(\mathbf{x}_{(i)}) = w(d(\mathbf{x}, \mathbf{x}_{(i)}))$$

$$Z_{\bullet} = \frac{\boxed{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

$$Z_{\bullet} = \frac{\boxed{w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

Weighted kNN

k = 6



$$? = \underset{\color{gray}\bullet}{\operatorname{argmax}}\, Z_{\color{gray}\bullet}$$

$$\text{if } Z_{\color{orange}\bullet} > Z_{\color{blue}\bullet} : \quad ? = {\color{orange}\bullet}$$

$$\text{if } Z_{\color{orange}\bullet} < Z_{\color{blue}\bullet} : \quad ? = {\color{blue}\bullet}$$

$$Z_{\color{blue}\bullet} = \frac{\boxed{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

$$Z_{\color{orange}\bullet} = \frac{\boxed{w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

Q&A