# Introduction to Machine Learning
# Lecture 4: Multiclass classification. KNN. Naive Bayes.

Harbour.Space University
February 2021

**Nikolay Karpachev**

# Outline

1. Binary classification recap
2. Multiclass classification
    - Linear multi-label model
    - Probabilistic approach. Logistic loss
    - Metrics
3. Regularization in linear models
4. Naive Bayes classifier
5. K-Nearest-Neighbours algorithm

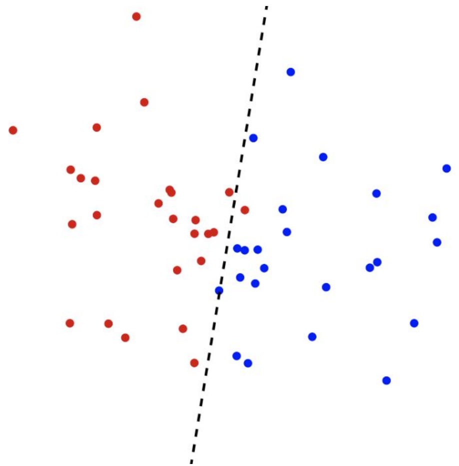
# Linear classification

**Linear classification model**

$$\hat{y} = f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$$

$$a(\mathbf{x}) = +1 \ \text{ if } f(\mathbf{x}) > 0$$
$$a(\mathbf{x}) = -1 \ \text{ if } f(\mathbf{x}) < 0$$

$$Q = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}_i)$$



# Margin Loss

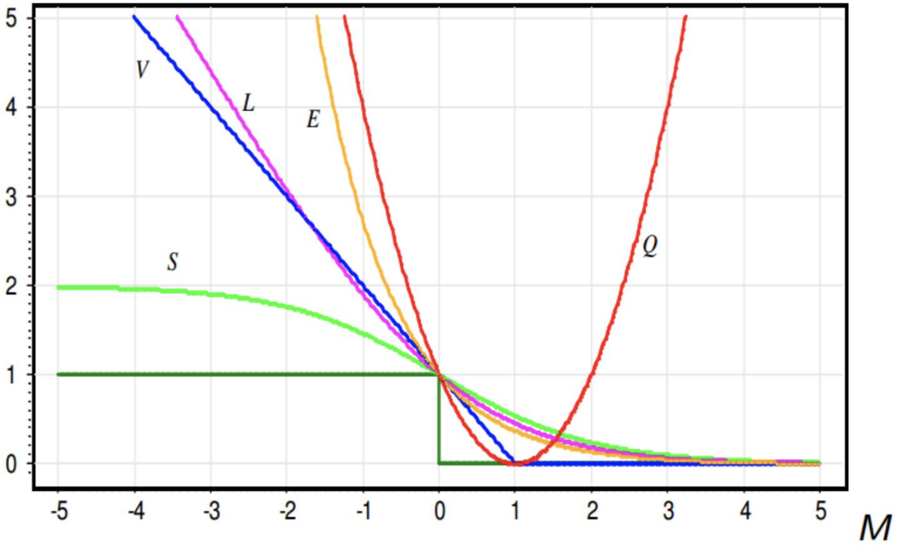
- Margin loss (number of classification errors)

$$Q = \frac{1}{N} \sum_{i=1}^{N} [M_i \leq 0]$$

# Loss functions in classification

$$Q = \frac{1}{N}\sum_{i=1}^{N}[M_i \le 0] \le \widetilde{Q} = \frac{1}{N}\sum_{i=1}^{N} L(M_i)$$

$$\widetilde{Q} \longrightarrow \min \implies Q \longrightarrow \min$$




$$Q(M) = (1-M)^2$$
$$V(M) = (1-M)_+$$
$$S(M) = 2(1+e^M)^{-1}$$
$$L(M) = \log_2(1+e^{-M})$$
$$E(M) = e^{-M}$$

# Logistic regression

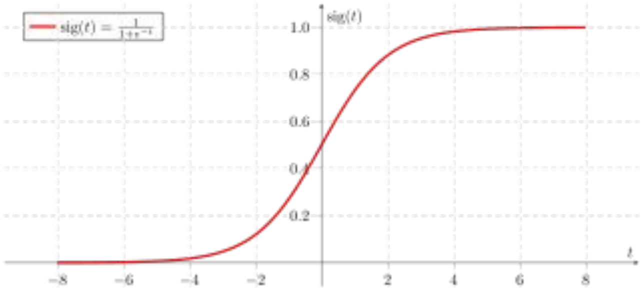
Linear classification model:

$$a(x) = \mathrm{sign}\langle w, x\rangle, \quad x, w \in \mathbb{R}^n.$$

Logistic regression:

$$p_i = \sigma(\langle w, x_i\rangle) = \frac{1}{1 + e^{-\langle w, x_i\rangle}} = P(y = 1|x)$$

**Sigmoid** function



# Logistic regression

$$y_i \in \{0, 1\} \qquad Q = -\sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i)\ln(1 - p_i) \to \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}} = P(y = 1 | x)$$

logistic loss

The class labels are described as Bernoulli-distributed data with parameters *p* inferred from the model

# Multiclass classification

- **Before:** two classes {-1, +1}

$$a(x) = \operatorname{sign}\langle w, x\rangle, \quad x, w \in \mathbb{R}^n$$

- **Now:** arbitrary number of classes {0, … , n-1}

Problem: how can we create an n-option decision rule based on the same dot-product linear model?

# Multiclass classification

- **Before:** two classes {-1, +1}

$$a(x) = \operatorname{sign}\langle w, x \rangle, \quad x, w \in \mathbb{R}^n$$

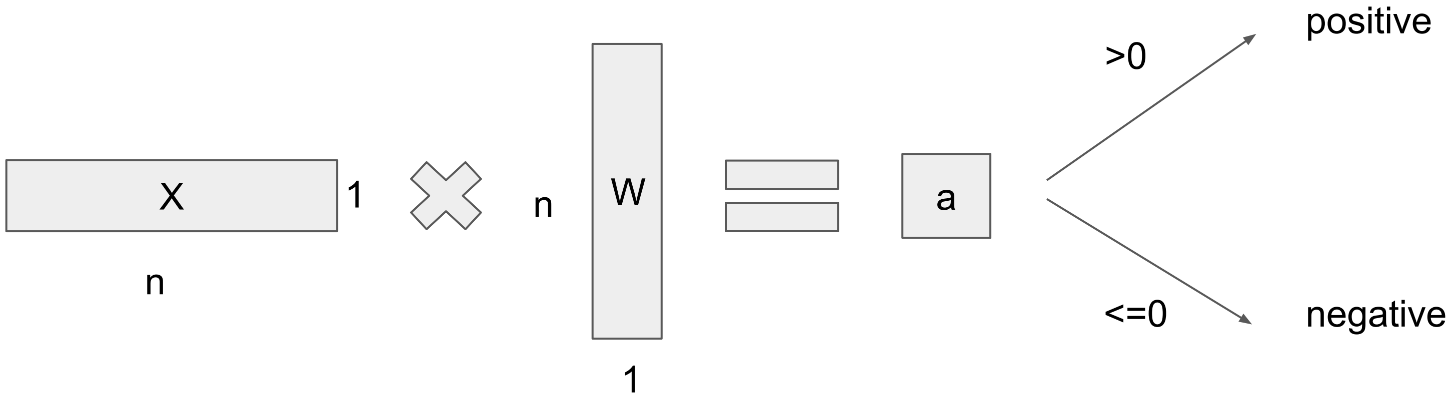
- **Now:** arbitrary number of classes {0, … , n-1}

$$a(x) = \arg\max_{y \in Y} \langle w_y, x \rangle, \quad x, w_y \in \mathbb{R}^n.$$

Possible classes

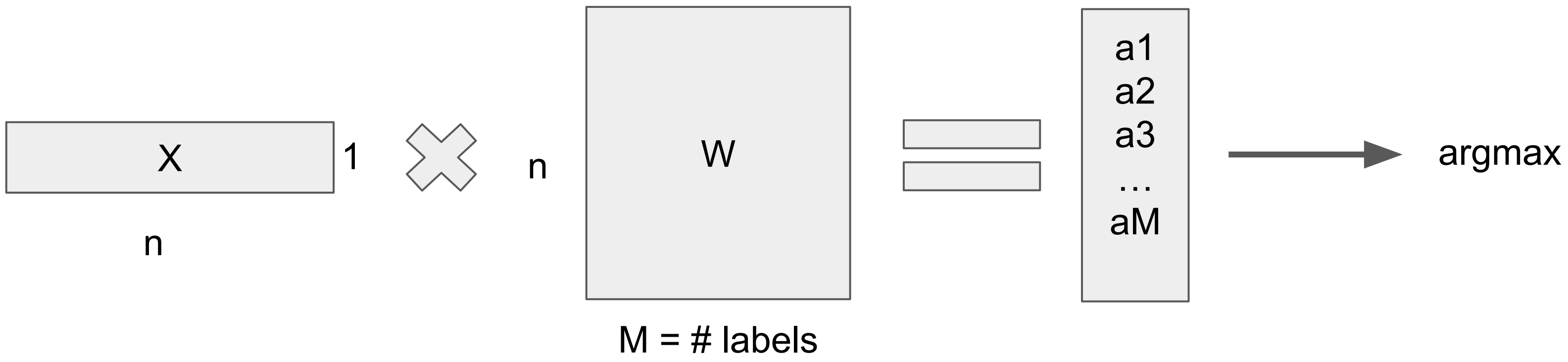
# Binary and multiclass models

- **Binary case**



# Binary and multiclass models

- **Multilabel case**



# Multiclass classification

**Prediction model**

$$a(x) = \arg\max_{y \in Y} \langle w_y, x \rangle, \qquad x, w_y \in \mathbb{R}^n.$$

How do we compute probabilities?

# Multiclass classification

**Prediction model**

$$a(x) = \arg\max_{y \in Y} \langle w_y, x \rangle, \qquad x, w_y \in \mathbb{R}^n.$$

**Probabilistic model**

$$P(y|x, w) = \frac{\exp\langle w_y, x \rangle}{\sum_{z \in Y} \exp\langle w_z, x \rangle} = \operatorname*{SoftMax}_{y \in Y} \langle w_y, x \rangle$$

# Multiclass classification

**Probabilistic model**

$$P(y|x, w) = \frac{\exp\langle w_y, x\rangle}{\sum_{z \in Y} \exp\langle w_z, x\rangle} = \underset{y \in Y}{\mathrm{SoftMax}}\langle w_y, x\rangle$$

Softmax example

$$\begin{bmatrix} -1 \\ 2.5 \\ 5 \\ 4 \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0.002 \\ 0.06 \\ 0.69 \\ 0.25 \\ 0.005 \end{bmatrix}$$

# Multiclass classification

**Optimization (categorical cross-entropy loss)**

$$L(w) = \sum_{i=1}^{\ell} \log P(y_i | x_i, w) \;\to\; \max_{w} .$$

# Regularization in linear models

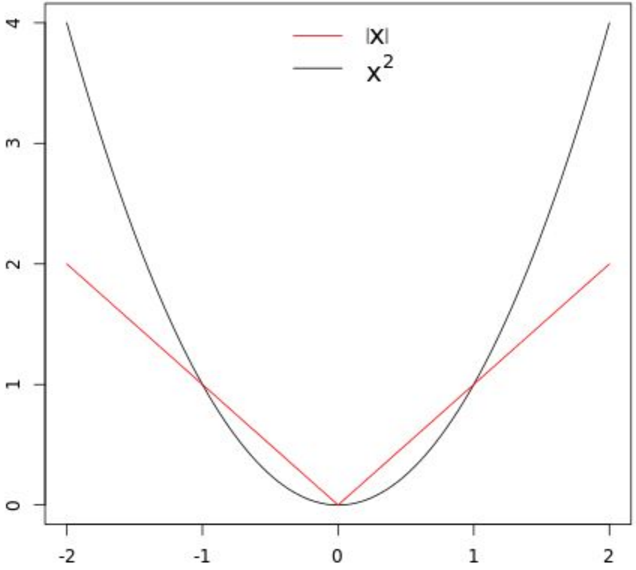
$$Q(\mathbf{w}) = \|Y - X\mathbf{w}\|_2^2 + \underline{\lambda^2\|\mathbf{w}\|_2^2}$$

Regularization term

# Regularization in linear models

- $L_2$ regularization
  - constraints weights
  - delivers more stable solution
  - Differentiable

- $L_1$ regularization
  - non-differentiable
    - (actually, the same as MAE)
  - selects features



# Regularization in linear models

$$Q(\mathbf{w}) = \|Y - X\mathbf{w}\|_2^2 + \lambda^2 \|\mathbf{w}\|_2^2$$

**The same way with classification**

$$Q(w) = \sum_{i=1}^{\ell} \log\big(1 + \exp(-\langle w, x_i \rangle y_i)\big) + \frac{\tau}{2}\|w\|^2 \;\to\; \min_{w}$$

Time for your questions and a coffee break.

# Naïve Bayes classifier

# Naive Bayes classifier

Let's denote:

- Training set $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ , where
    - $\mathbf{x}_i \in \mathbb{R}^p$ , $y_i \in \{C_1, \ldots, C_k\}$ for k-class classification

# Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

or, in our case

$$P(y_i = C_k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = C_k)P(y_i = C_k)}{P(\mathbf{x}_i)}$$

# Naive Bayes classifier

Let's denote:

- Training set $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ , where
  - $\mathbf{x}_i \in \mathbb{R}^p$ , $y_i \in \{C_1, \ldots, C_K\}$ for K-class classification

$$P(y_i = C_k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = C_k)P(y_i = C_k)}{P(\mathbf{x}_i)}$$

Naive assumption: features are **independent**

# Naive Bayes classifier

$$P(y_i = C_k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = C_k)P(y_i = C_k)}{P(\mathbf{x}_i)}$$

Naive assumption: features are **independent**:

$$P(\mathbf{x}_i|y_i = C_k) = \prod_{l=1}^{p} P(x_i^l|y_i = C_k)$$

# Naive Bayes classifier

$$P(y_i = C_k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = C_k)P(y_i = C_k)}{\cancel{P(\mathbf{x}_i)}}$$

Optimal class label:

$$C^* = \arg\max_k P(y_i = C_k|\mathbf{x_i})$$

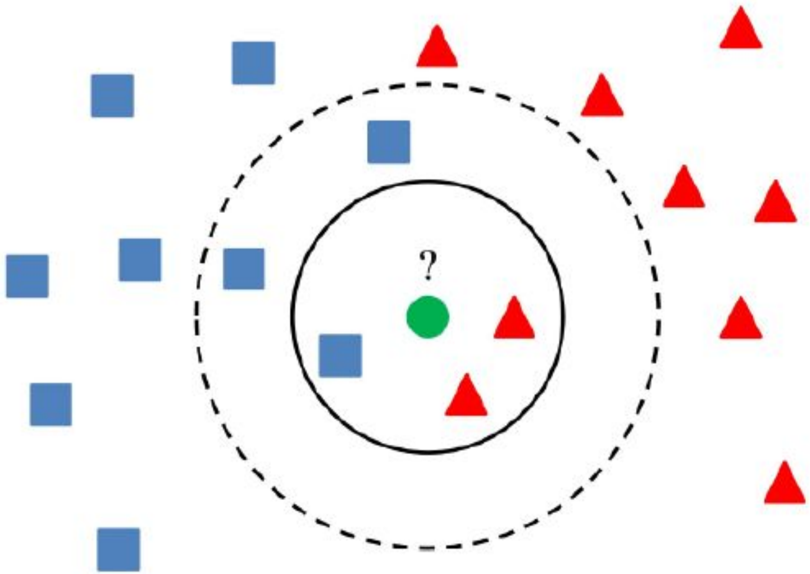
To find maximum we even do not need the denominator

But we need it to get probabilities

# kNN – k Nearest Neighbors

# kNN - k Nearest Neighbours

# kNN - k Nearest Neighbours



# k Nearest Neighbors Method

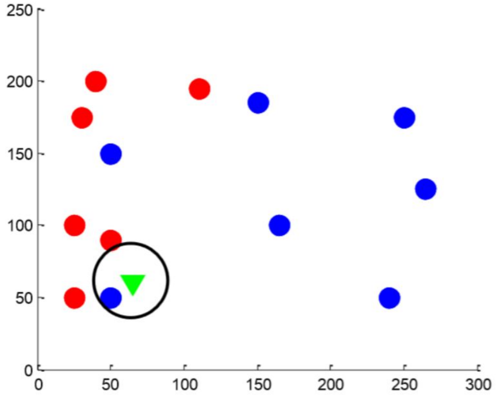
Given a *new observation*:

1. Calculate the distance to each of the samples in the dataset.
2. Select  samples from the dataset with the minimal distance to them.
3. The label of the *new observation* will be the most frequent label among those nearest neighbors.

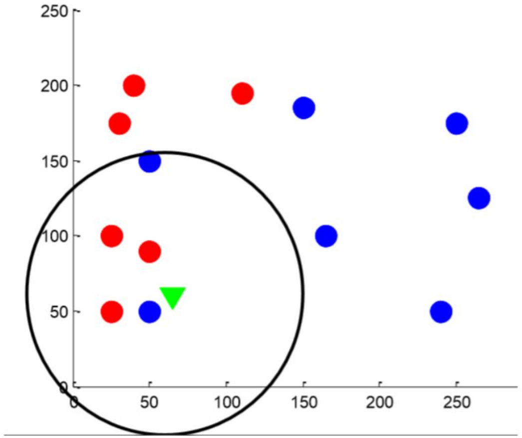
# How to make it better?

- The number of neighbors k (it is a ***hyperparameter***)

# kNN classification




k = 1



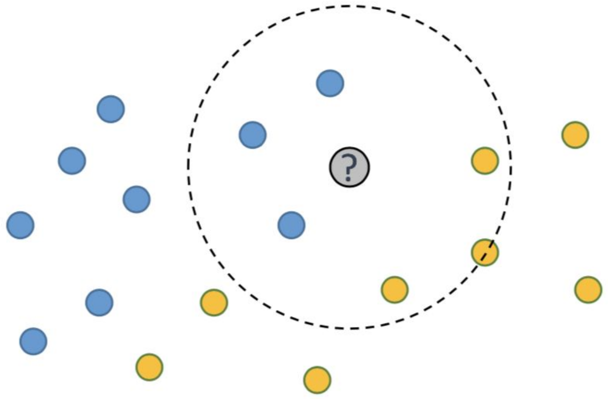

k = 5

# How to make it better?

- The number of neighbors k  (it is a ***hyperparameter***)
- The distance measure between samples
    a. Hamming
    b. Euclidean
    c. cosine
    d. Minkowski distances
    e. etc.
- Weighted neighbours

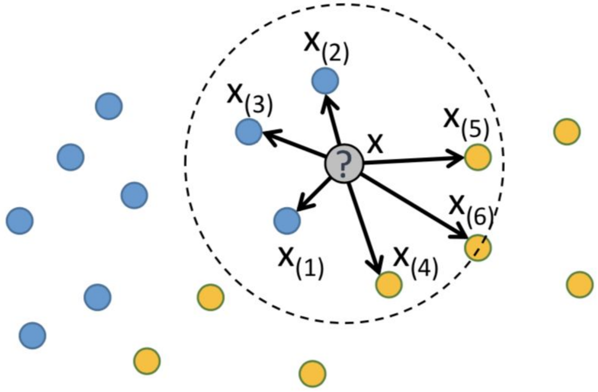
# Weighted kNN

k = 6



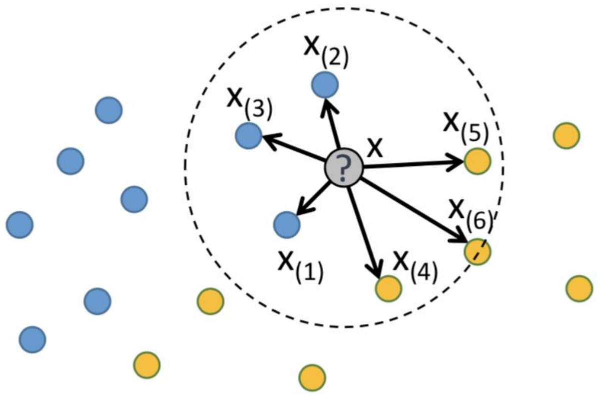
# Weighted kNN

k = 6



# Weighted kNN

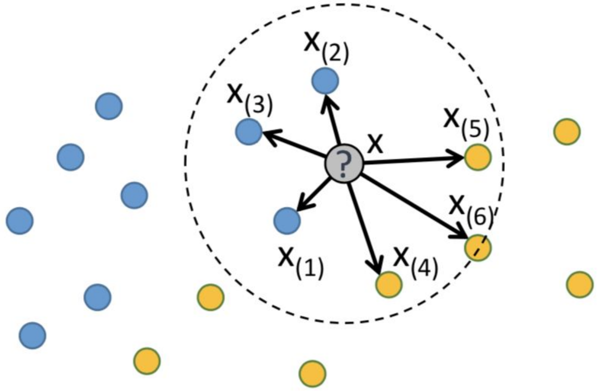
k = 6




- Weights can be adjusted according to the neighbors order, $w(\mathbf{x}_{(i)}) = w_i$

# Weighted kNN

k = 6



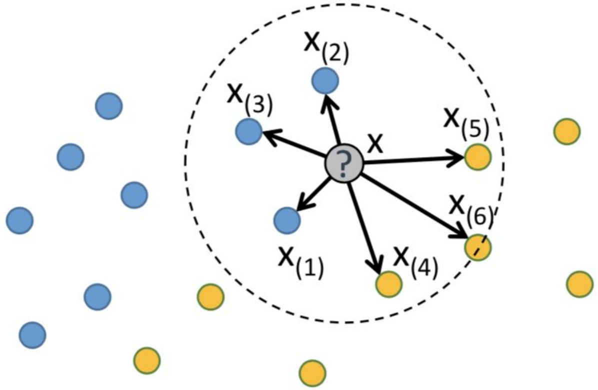

- Weights can be adjusted according to the neighbors order, $w(\mathbf{x}_{(i)}) = w_i$

- or on the distance itself

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

# Weighted kNN

k = 6




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

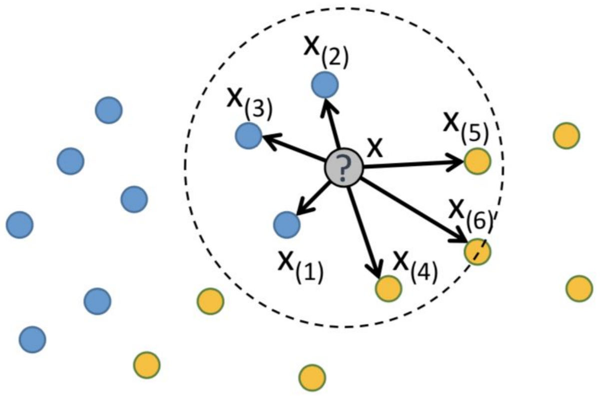
- Weights can be adjusted according to the neighbors order,
  $w(\mathbf{x}_{(i)}) = w_i$

- or on the distance itself

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

# Weighted kNN

k = 6

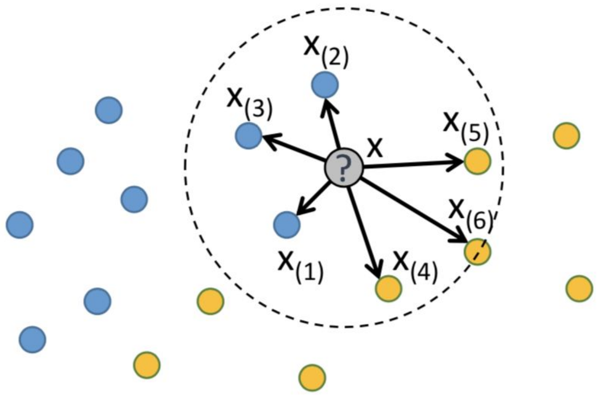


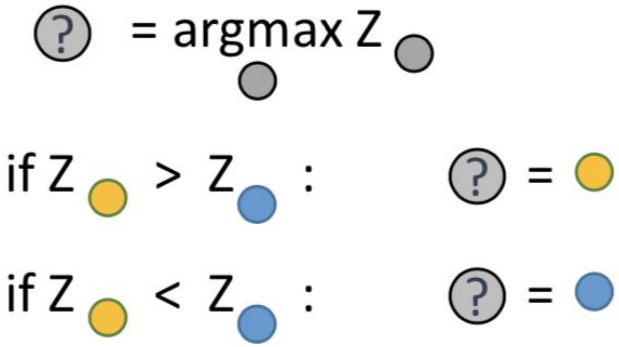

$$Z_{\bullet} = \frac{\boxed{w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) + w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})}$$

- 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



Thanks for your attention!