Classification: Part 1

Statistical Analysis and Document Mining

Spring 2019

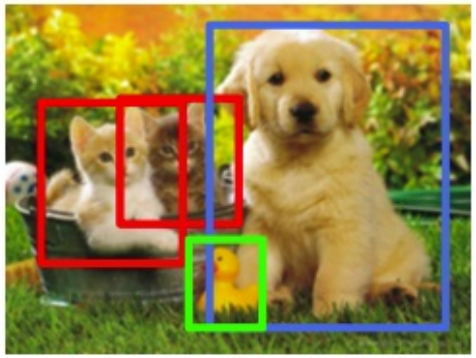
Vasilii Feofanov

Université Grenoble Alpes
vasilii.feofanov@univ-grenoble-alpes.fr

Outline

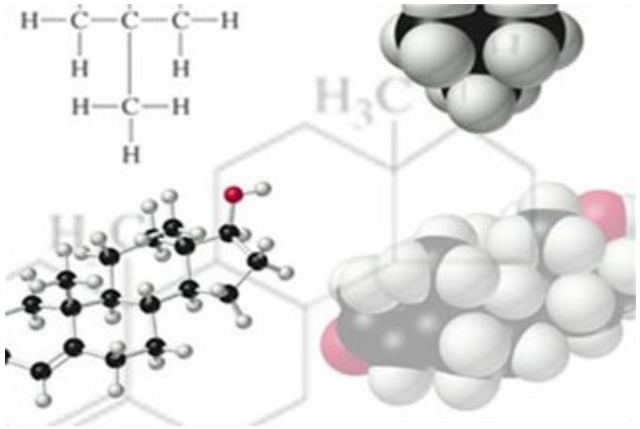
Introduction

CAT

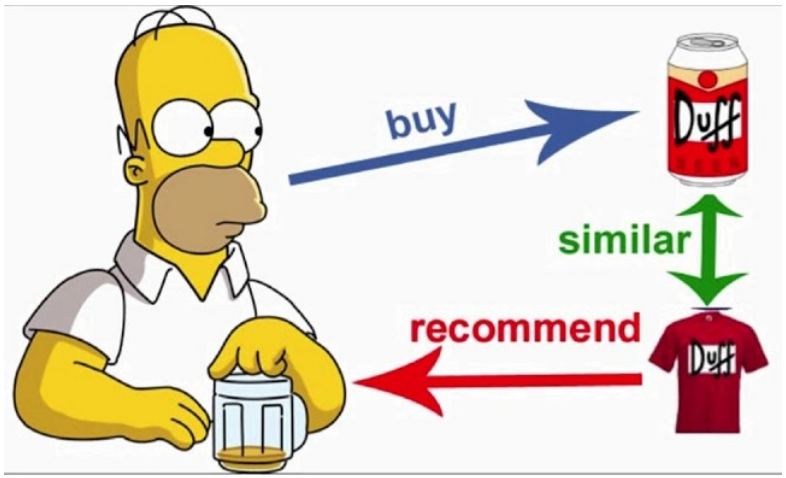


CAT, DOG, DUCK

(a) Computer Vision



(b) Bioinformatics



(c) Recommender Systems

(d) Pattern Recognition

Iris Flower Data Set

(a) Iris Setosa

(b) Iris Versicolor

(c) Iris Virginica

Figure: The data set consists of 50 samples from each of 3 species of Iris. From each sample the length and the width of the sepals and petals were measured. Based on this, the goal is to build a model that distinguishes the species from each other.[1]

[1] https://en.wikipedia.org/wiki/Iris_flower_data_set

Classification: Simple Formulation

- Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$: collected samples with class labels;
- A new data point $\mathbf{x}$ without any label;
- Build a classifier $h(\mathbf{x})$ that predicts y as accurately as possible:

$$\mathbf{x} = \overset{\text{Sepal.L}}{(6.2,} \quad \overset{\text{Sepal.W}}{2.8,} \quad \overset{\text{Petal.L}}{5.6,} \quad \overset{\text{Petal.W}}{2.4)} \quad \overset{?}{\longrightarrow} \quad y = \texttt{"Virginica"}$$

How to do that?

Iris Classification Based on the Petal Information

Iris: New Data Point

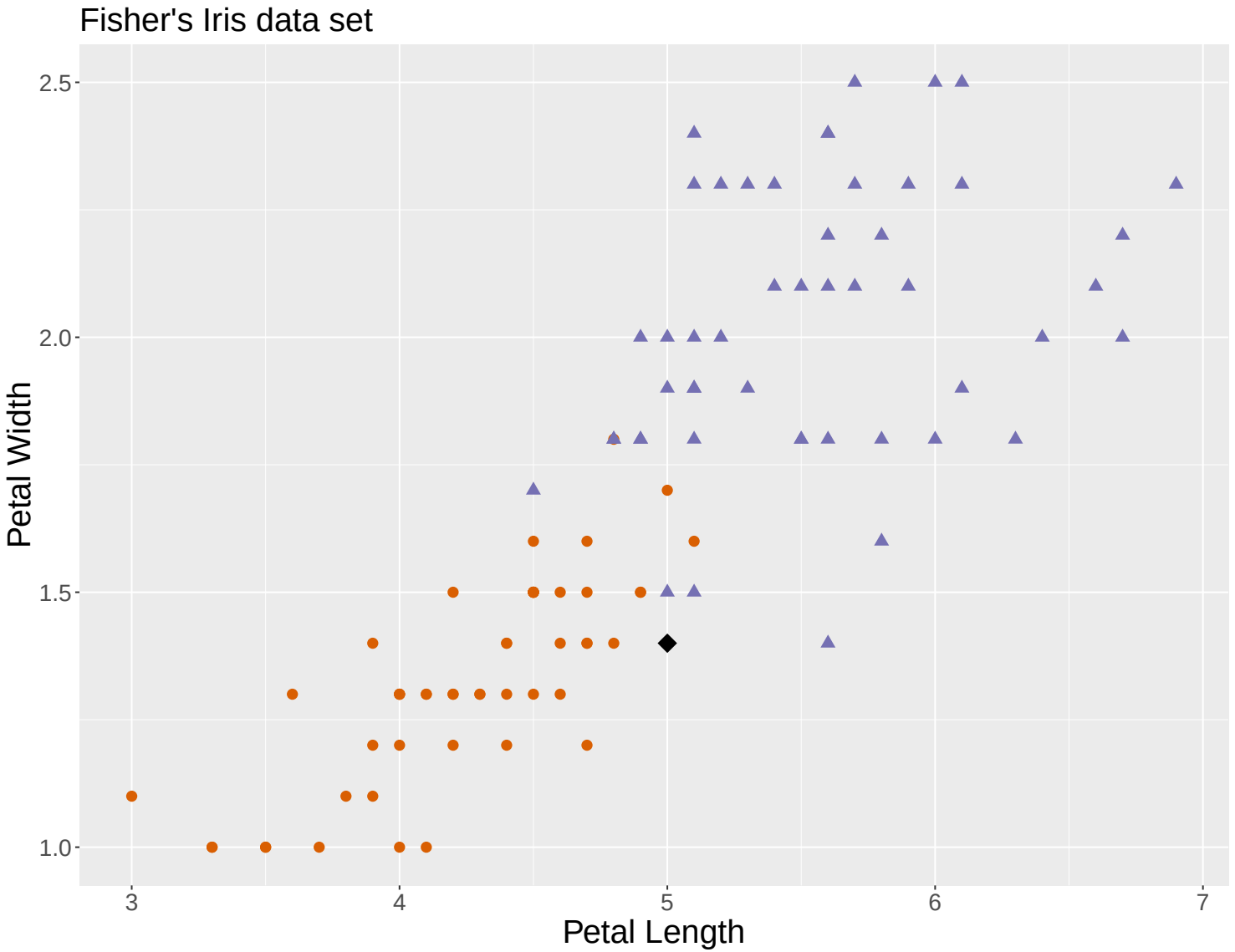
4 Nearest Neighbours

4NN Clasification

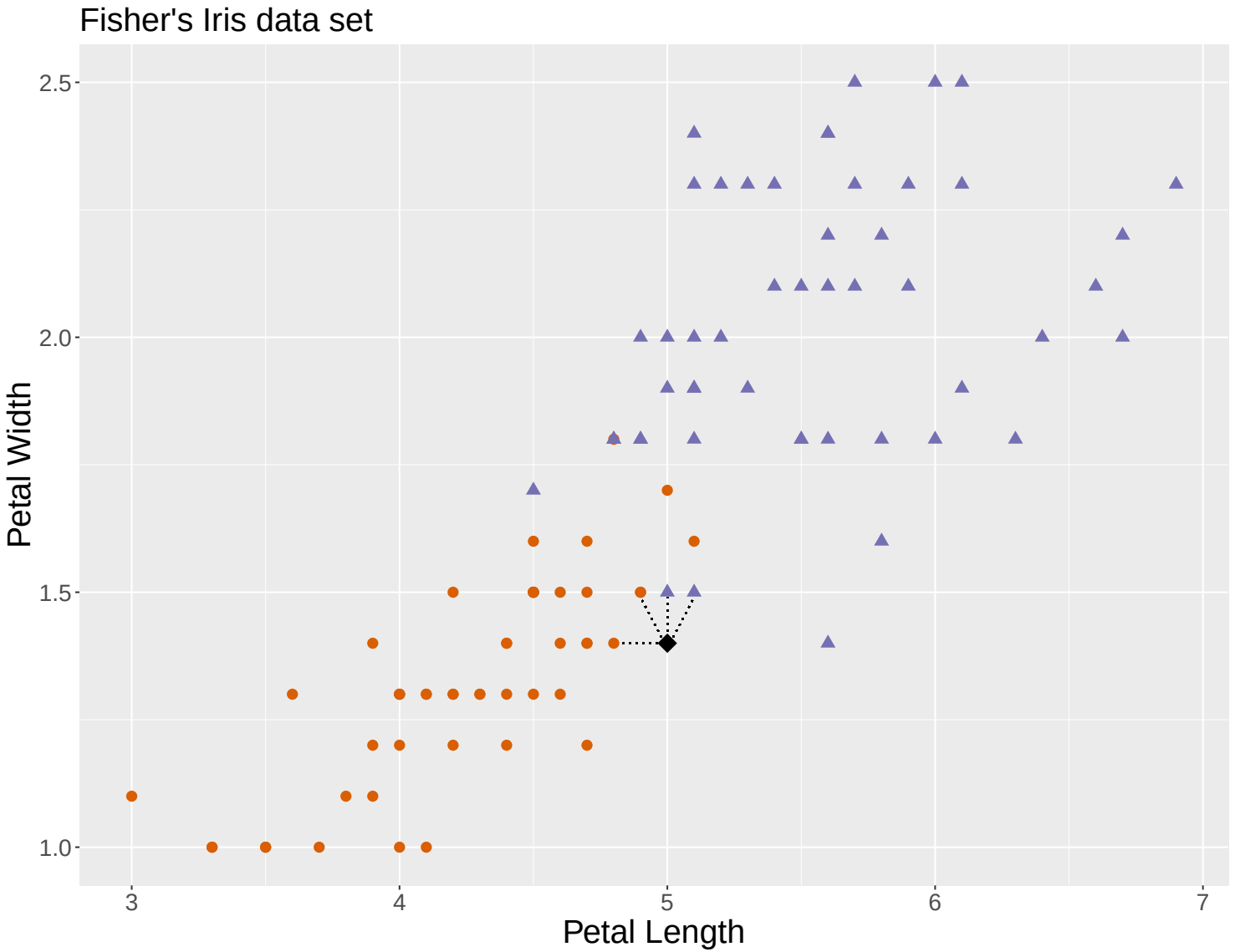
Another Data Point

And Its Prediction

One More Example



What to do in this case?



Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

- What is the time complexity of the algorithm?

Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

- How does the choice of k affect kNN?

Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

- How does the choice of distance metric affect the algorithm?

Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

- Is the nearest neighbours approach applicable for the regression task?

Algorithm k Nearest Neighbours (kNN)

Input: Training set $\{\mathbf{x}_i, y_i\}_{i=1}^n$;
Number of classes K;
New data point $\mathbf{x}$.

1. Compute distance $d(\mathbf{x}, \mathbf{x}_i)$ **for** $i = \{1, \ldots, n\}$.

2. Find k closest training examples to $\mathbf{x}$:

$$J \subset \{1, \ldots, n\} \quad \text{s.t.} \quad |J| = k;$$
$$\forall j \in J, \forall t \in \{1, \ldots, n\} \setminus J : d(\mathbf{x}, \mathbf{x}_j) \leq d(\mathbf{x}, \mathbf{x}_t).$$

Output: Majority class $h(\mathbf{x}) = \text{argmax}_{c=1,\ldots,K} \sum_{j \in J} \mathbb{I}(y_j = c)$.

- kNN is an *instance-based* learning algorithm. What is the main drawback of such methods?

Outline

Discriminant Analysis (Fisher, 1936)

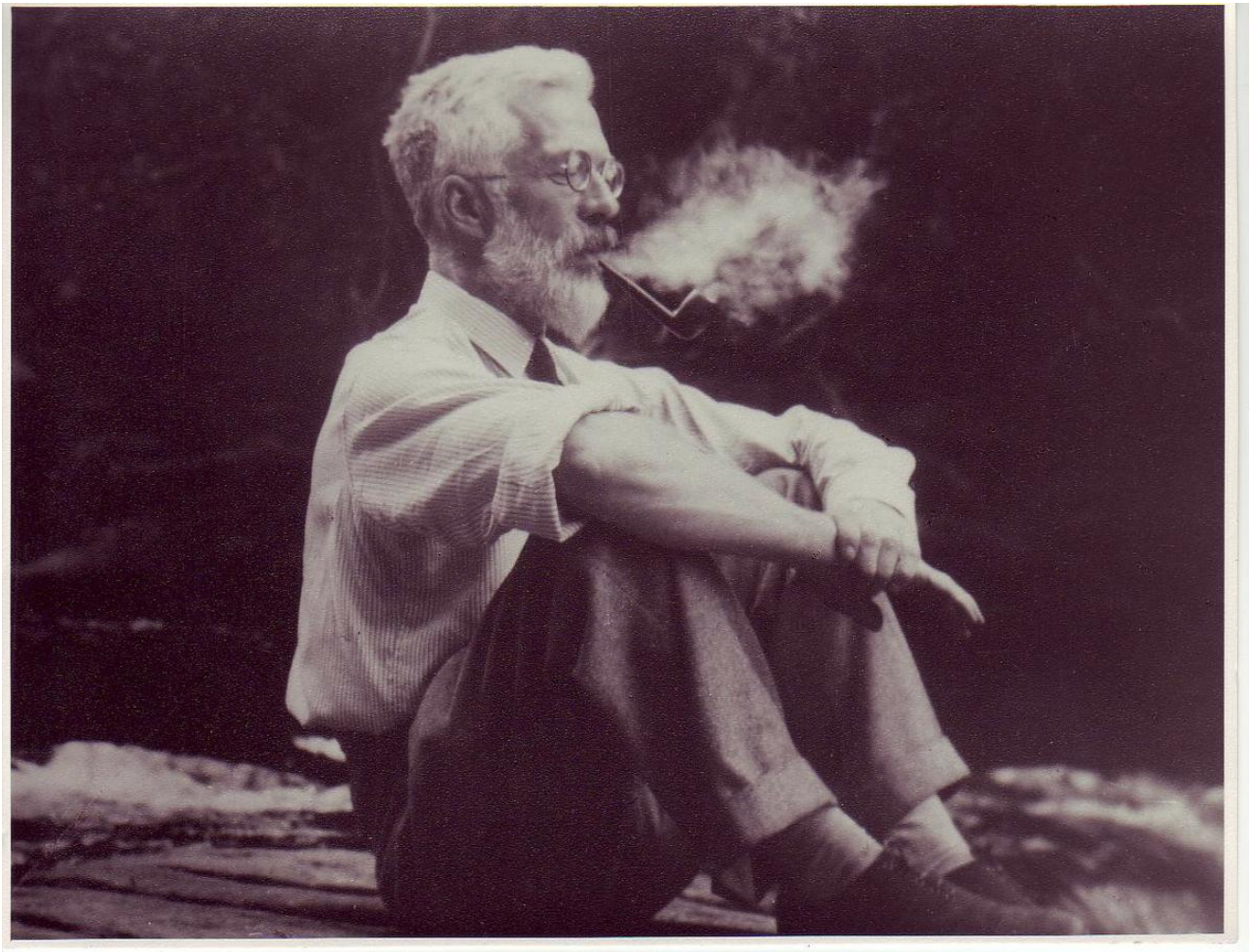


Figure: Ronald A. Fisher in 1946.

Classification: More Formally

- *Input space:* $\mathcal{X} \subseteq \mathbb{R}^d$;
- Output space: $\mathcal{Y} = \{-1, +1\}$ (binary classification),
  $\mathcal{Y} = \{1, \ldots, K\}$ (multi-class classification);
- *Assumption:* all $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y}$ are **i.i.d.** from $\mathcal{D}$ with respect to a fixed unknown probability distribution $P(\mathbf{X}, Y)$;
- *Sample Data:* we observe $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$;

Classification: More Formally

- *Input space:* $\mathcal{X} \subseteq \mathbb{R}^d$;
- Output space: $\mathcal{Y} = \{-1, +1\}$ (binary classification),
  $\mathcal{Y} = \{1, \ldots, K\}$ (multi-class classification);
- *Assumption:* all $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y}$ are **i.i.d.** from $\mathcal{D}$ with respect to a fixed unknown probability distribution $P(\mathbf{X}, Y)$;
- *Sample Data:* we observe $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$;
- *Loss Function:* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$;
- *Target*: minimise the *risk* $R^\ell(h) := \mathbb{E}_{(\mathbf{X},Y)\sim\mathcal{D}}\, \ell(h(\mathbf{X}), Y)$.

0/1 Loss Function

0/1 loss function has the following view:

$$\ell^{0/1}(h(\mathbf{x}), y) = \mathbb{I}(h(\mathbf{x}) \neq y) = \begin{cases} 1, & \text{if } h(\mathbf{x}) \neq y; \\ 0, & \text{if } h(\mathbf{x}) = y \end{cases}$$

0/1 Loss Function

0/1 loss function has the following view:

$$\ell^{0/1}(h(\mathbf{x}), y) = \mathbb{I}(h(\mathbf{x}) \neq y) = \begin{cases} 1, & \text{if } h(\mathbf{x}) \neq y; \\ 0, & \text{if } h(\mathbf{x}) = y \end{cases}$$

In this case, the risk is written as:

$$\begin{aligned} R(h) &= P(h(\mathbf{X}) \neq Y) \\ &= \sum_{c \in \{1,\dots,K\}} P(Y = c) P(h(\mathbf{X}) \neq c | Y = c) \end{aligned}$$

0/1 Loss Function

0/1 loss function has the following view:

$$\ell^{0/1}(h(\mathbf{x}), y) = \mathbb{I}(h(\mathbf{x}) \neq y) = \begin{cases} 1, & \text{if } h(\mathbf{x}) \neq y; \\ 0, & \text{if } h(\mathbf{x}) = y \end{cases}$$

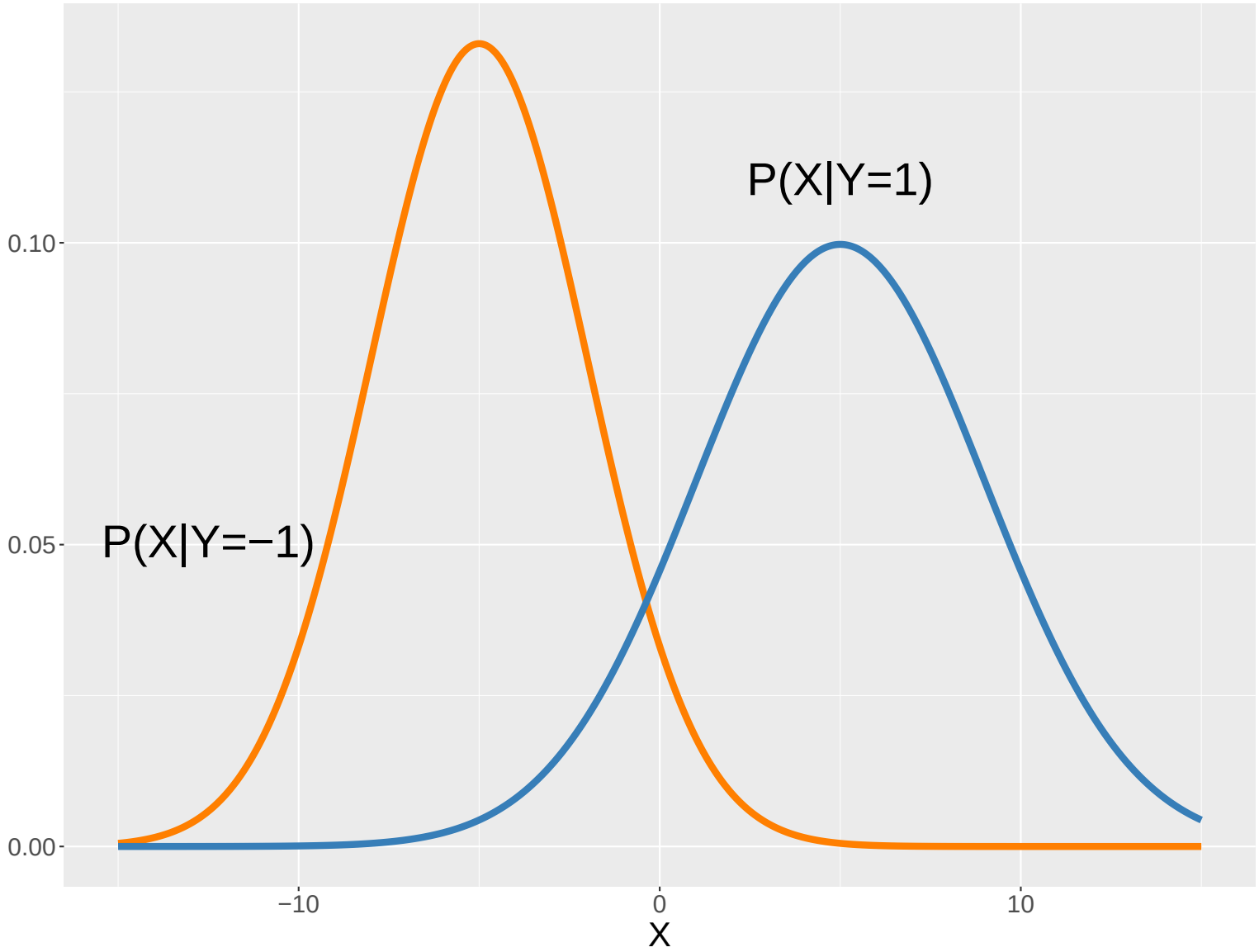
In this case, the risk is written as:

$$\begin{aligned} R(h) &= P(h(\mathbf{X}) \neq Y) \\ &= \sum_{c \in \{1,\dots,K\}} P(Y = c) P(h(\mathbf{X}) \neq c | Y = c) \end{aligned}$$

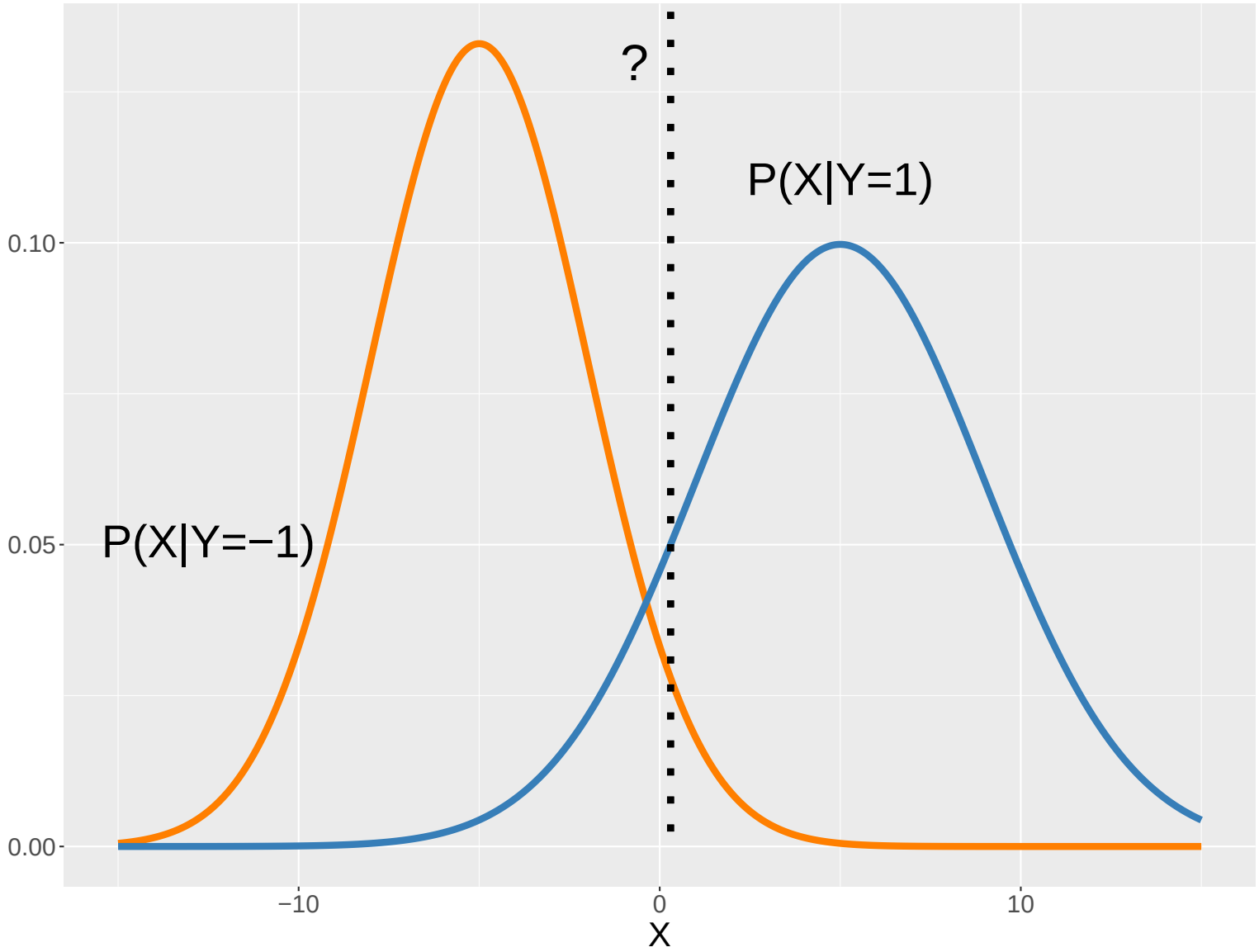
In the binary case, it is represented as:

$$R(h) = P(Y = -1)P(h(\mathbf{X}) = 1|Y = -1) + P(Y = 1)P(h(\mathbf{X}) = -1|Y = 1)$$

Classification in Terms of Distributions



Classification in Terms of Distributions



Bayes' Rule

$$\underbrace{P(Y|\mathbf{X})}_{\text{Posterior}} = \frac{\overbrace{P(\mathbf{X}|Y)}^{\text{Likelihood}}\overbrace{P(Y)}^{\text{Class Prior}}}{\underbrace{P(\mathbf{X})}_{\text{Evidence}}}$$

Maximum A Posteriori Rule

Idea: Classify $\mathbf{x}$ to a class with the highest posterior probability:

$$h_B(\mathbf{x}) := \underset{y \in \mathcal{Y}}{\operatorname{argmax}} P(Y = y | \mathbf{X} = \mathbf{x}).$$

This is equivalent to:

$$h_B(\mathbf{x}) \propto \underset{y \in \mathcal{Y}}{\operatorname{argmax}} P(\mathbf{X} = \mathbf{x} | Y = y) P(Y = y).$$

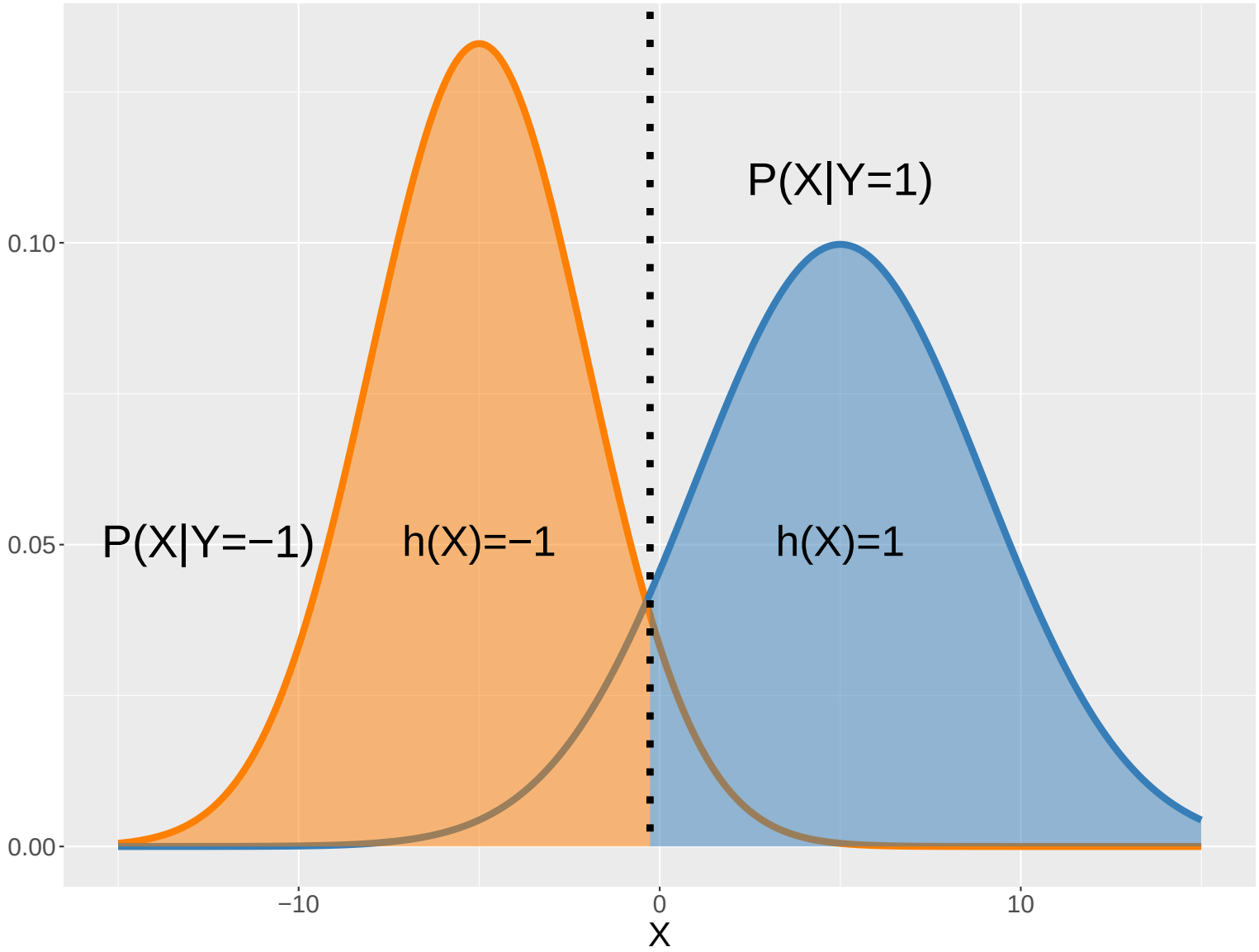
The method is often called the *Bayes* classifier.

Bayes Classifier in the Binary Case

In the binary case ($\mathcal{Y} = \{-1, +1\}$), the Bayes classifier is usually defined in the following way:

$$h_B(\mathbf{x}) = \begin{cases} +1, & \text{if } P(Y = +1 | \mathbf{X} = \mathbf{x}) \geq P(Y = -1 | \mathbf{X} = \mathbf{x}); \\ -1, & \text{if } P(Y = +1 | \mathbf{X} = \mathbf{x}) < P(Y = -1 | \mathbf{X} = \mathbf{x}). \end{cases}$$

Particular Case: Maximum Likelihood Estimator



Bayes Classifier Is Optimal

Theorem

Suppose $P(Y)$ and $P(\mathbf{X}|Y)$ are given. Then the Bayes classifier yields the minimum of the misclassification error.

Exercise: Prove this theorem. For sake of simplicity, consider the binary case ($\mathcal{Y} = \{-1, +1\}$).

Bayes Classifier Is Optimal

Theorem

Suppose $P(Y)$ and $P(\mathbf{X}|Y)$ are given. Then the Bayes classifier yields the minimum of the misclassification error.

Exercise: Prove this theorem. For sake of simplicity, consider the binary case ($\mathcal{Y} = \{-1, +1\}$).

1. $P(h(\mathbf{X}) \neq Y) = \int P(h(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x})d\mathbf{x}$. Then, what is the value of $P(h_B(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x})$?

Bayes Classifier Is Optimal

Theorem

Suppose $P(Y)$ and $P(\mathbf{X}|Y)$ are given. Then the Bayes classifier yields the minimum of the misclassification error.

Exercise: Prove this theorem. For sake of simplicity, consider the binary case ($\mathcal{Y} = \{-1, +1\}$).

1. $P(h(\mathbf{X}) \neq Y) = \int P(h(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x})d\mathbf{x}$. Then, what is the value of $P(h_B(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x})$?
2. $P(h_B(\mathbf{x}) \neq Y|\mathbf{X} = \mathbf{x}) = 1 - \max_{c \in \mathcal{Y}} P(Y = c|\mathbf{X} = \mathbf{x})$. Does this prove the theorem?

Bayes Classifier: Discussion

In practice, we don't have any information about data distribution.

- Can we estimate $P(Y)$?

Bayes Classifier: Discussion

In practice, we don't have any information about data distribution.

- Can we estimate $P(Y)$?
- Can we estimate $P(\mathbf{X}|Y)$?

Additional Assumptions

- *Assumption 1:* Observations from a class $c \in \mathcal{Y}$ are normally distributed $[\mathbf{X}|Y = c] \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.
  - Remember the formula of the multivariate normal distribution.

Additional Assumptions

- *Assumption 1:* Observations from a class $c \in \mathcal{Y}$ are normally distributed $[\mathbf{X}|Y = c] \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.
  - Remember the formula of the multivariate normal distribution.

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)}.$$

Additional Assumptions

- *Assumption 1:* Observations from a class $c \in \mathcal{Y}$ are normally distributed $[\mathbf{X}|Y = c] \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.
  - Remember the formula of the multivariate normal distribution.

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_c)^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_c)}.$$

- *Assumption 2:* The covariance matrices of the classes are equal:

$$\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}.$$

Due to the logarithm properties, the Bayes classifier can be written as follows:

$$h_B(\mathbf{x}) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} [\ln P(\mathbf{X} = \mathbf{x} | Y = c) + \ln P(Y = c)].$$

Due to the logarithm properties, the Bayes classifier can be written as follows:

$$h_B(\mathbf{x}) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} [\ln P(\mathbf{X} = \mathbf{x} | Y = c) + \ln P(Y = c)].$$

Taking into account Assumption 1 and Assumption 2, we derive the algorithm called *Linear Discriminant Analysis (LDA)*:

$$h_{LDA}(\mathbf{x}) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} \delta_c(\mathbf{x}),$$

$$\delta_c(\mathbf{x}) = \boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_c^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \ln P(Y = c);$$

δ_c is usually called the *discriminant function*.

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

Condition:

$$(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^{\intercal}\boldsymbol{\Sigma}^{-1}\mathbf{x}-\frac{1}{2}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^{\intercal}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}+\boldsymbol{\mu}_{-1})-\ln\frac{P(Y=-1)}{P(Y=+1)} \geq 0.$$

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

Condition:

$$(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}+\boldsymbol{\mu}_{-1}) - \ln \frac{P(Y=-1)}{P(Y=+1)} \geq 0.$$

- Why it is called "linear"?

LDA in the Binary Case

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

Condition:

$$\overbrace{(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}}^{\mathbf{a}^\intercal} \mathbf{x} - \overbrace{\frac{1}{2}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1} + \boldsymbol{\mu}_{-1}) - \ln \frac{P(Y=-1)}{P(Y=+1)}}^{\mathbf{b}} \geq 0.$$

- Why it is called "linear"?
- How many parameters we need to estimate?

LDA in the Binary Case

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

Condition:

$$\overbrace{(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}}^{\mathbf{a}^\intercal} \mathbf{x} - \overbrace{\frac{1}{2}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1} + \boldsymbol{\mu}_{-1}) - \ln \frac{P(Y=-1)}{P(Y=+1)}}^{\mathbf{b}} \geq 0.$$

- Why it is called "linear"?
- How many parameters we need to estimate?
- What happens when Assumption 1, 2 are violated?

LDA in the Binary Case

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

Condition:

$$\overbrace{(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}}^{\mathbf{a}^\intercal} \mathbf{x} - \overbrace{\frac{1}{2}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1} + \boldsymbol{\mu}_{-1}) - \ln \frac{P(Y=-1)}{P(Y=+1)}}^{\mathbf{b}} \geq 0.$$

- Why it is called "linear"?
- How many parameters we need to estimate?
- What happens when Assumption 1, 2 are violated?
- What is the time complexity during training phase?

LDA in the Binary Case

In the binary case, the decision rule can be also written as:

$$h_{LDA}(\mathbf{x}) = \begin{cases} +1, & \text{if } \textit{Condition} \text{ is true;} \\ -1, & \text{otherwise.} \end{cases}$$

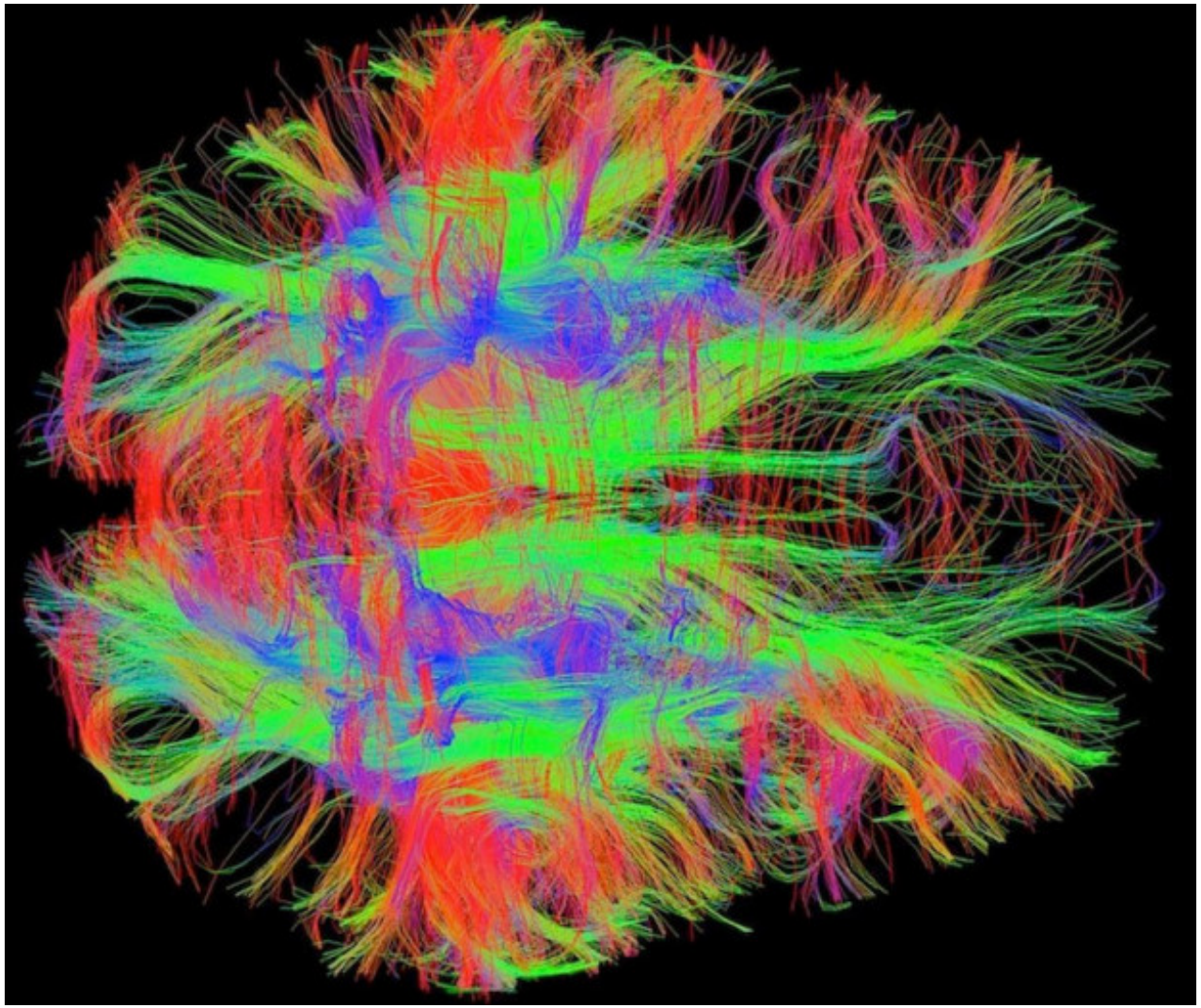
Condition:

$$\overbrace{(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}}^{\mathbf{a}^\intercal} \mathbf{x} - \overbrace{\frac{1}{2}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1} + \boldsymbol{\mu}_{-1}) - \ln \frac{P(Y=-1)}{P(Y=+1)}}^{\mathbf{b}} \geq 0.$$

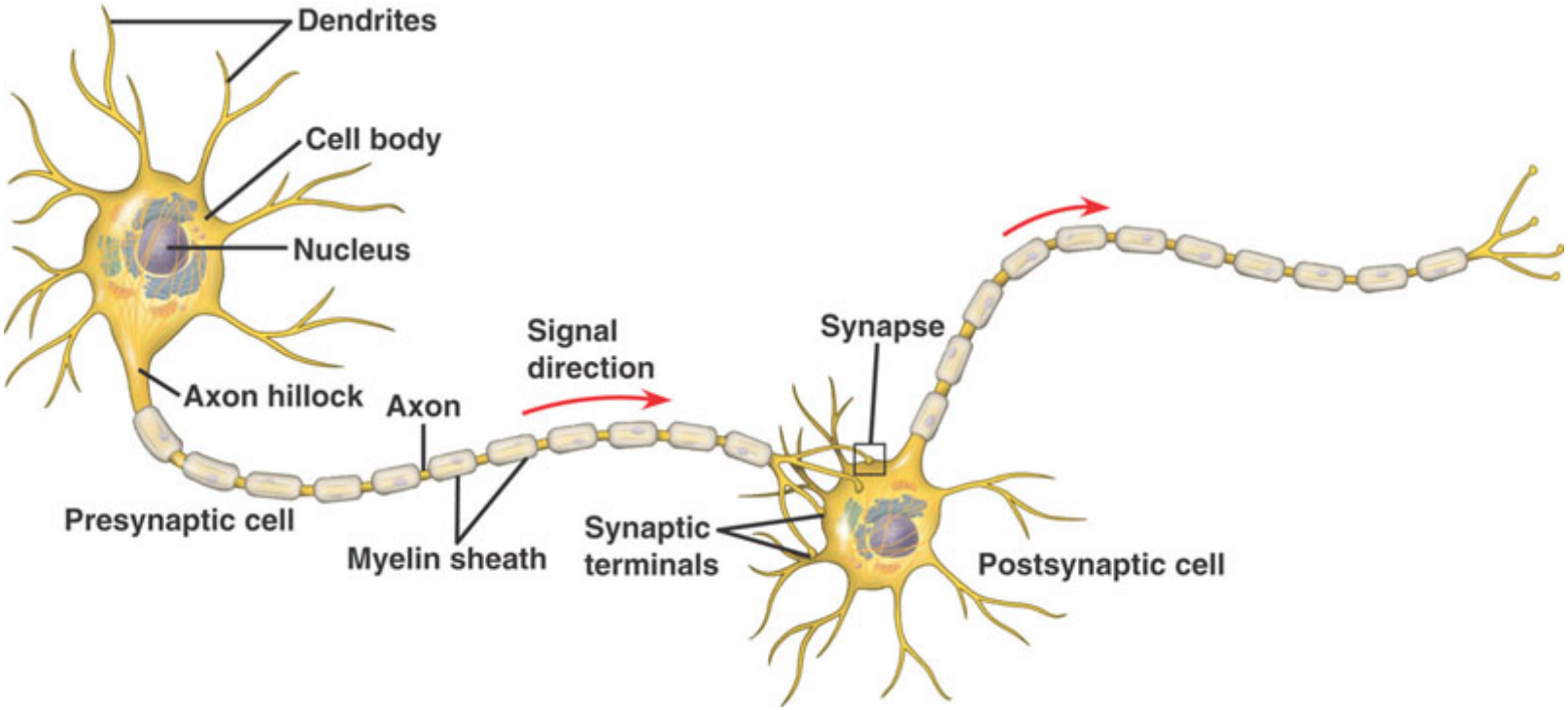
- Why it is called "linear"?
- How many parameters we need to estimate?
- What happens when Assumption 1, 2 are violated?
- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?

Outline

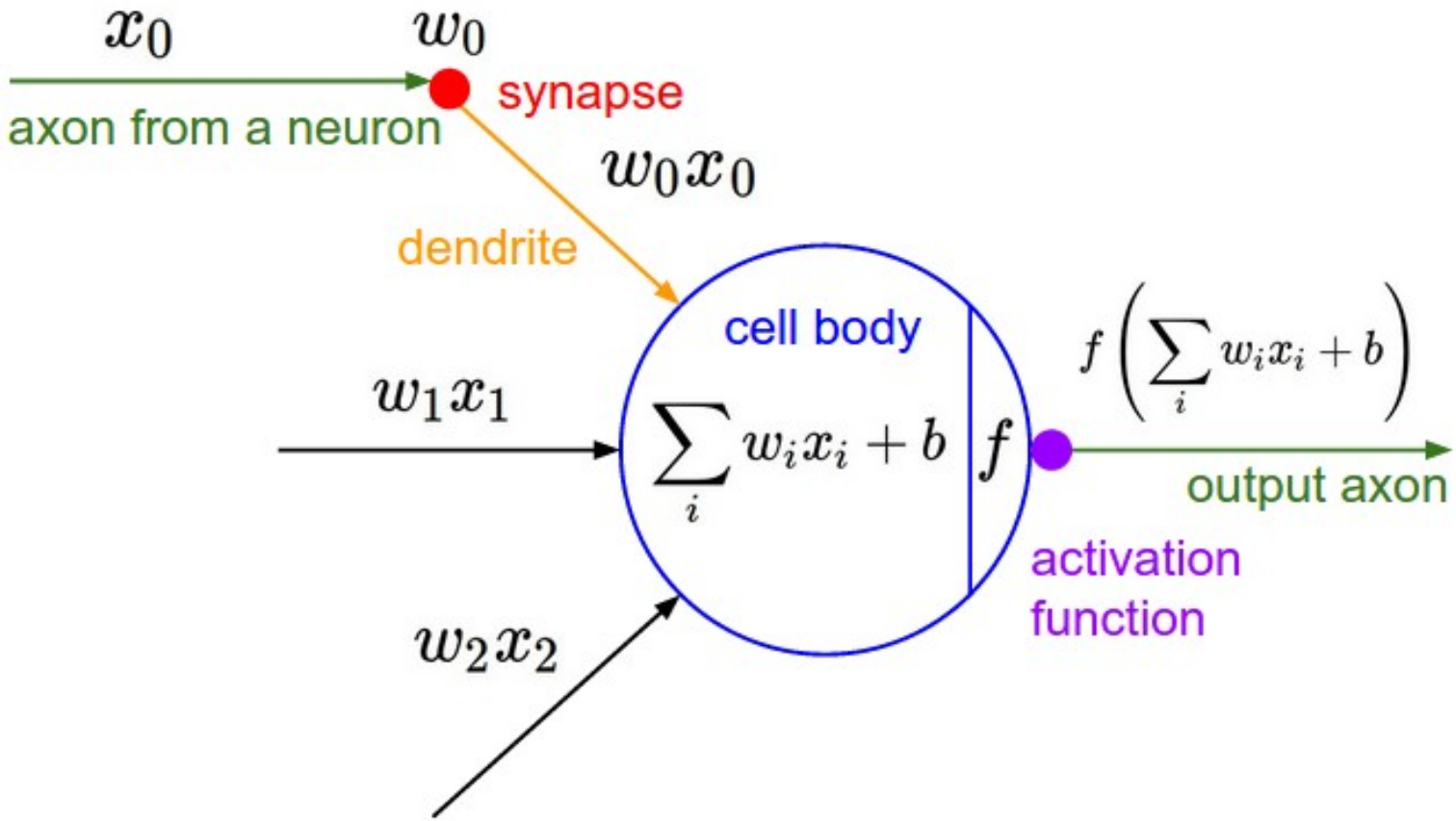
AI and Human Brain Study



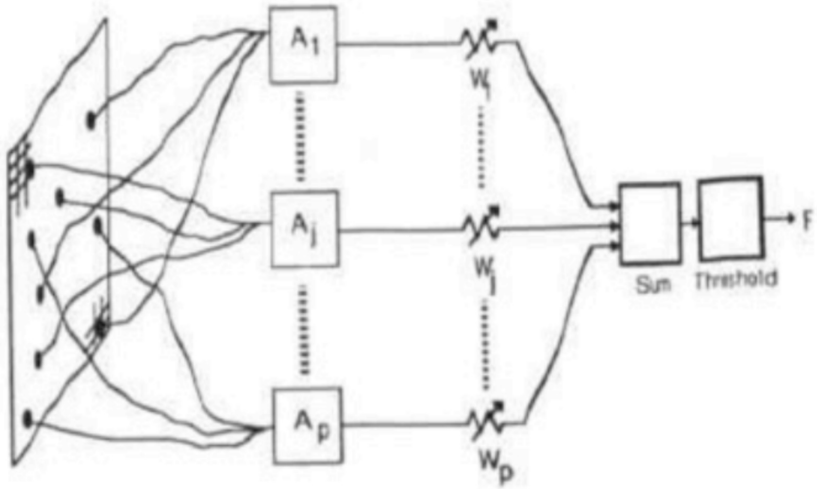
Biological Neuron

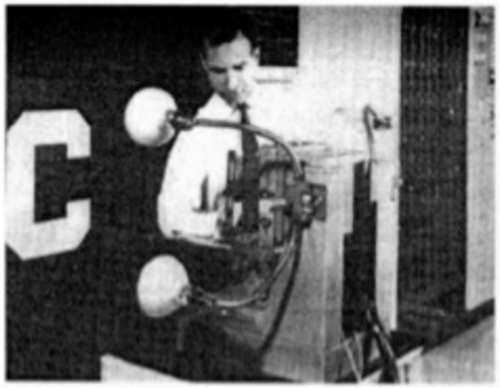


Neuron Model (MuCulloch & Pitts, 1943)



Perceptron Machine (Rosenblatt, 1958)





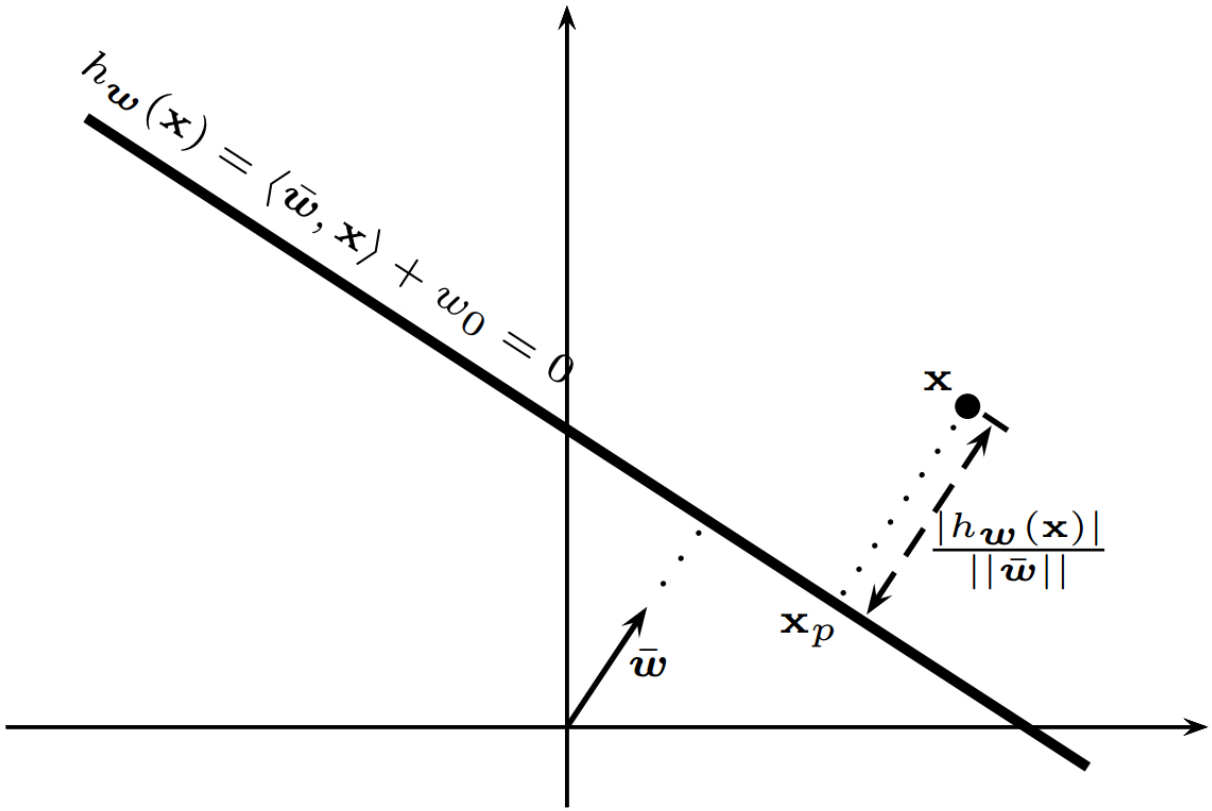
- **Summation:**

$$h_w(\mathbf{x}) := \langle \mathbf{w}, \mathbf{x} \rangle + w_0 = \sum_{j=1}^{d} w_j x_j + w_0.$$

- **Activation:**

$$a(\mathbf{x}) = \begin{cases} +1, & \text{if } y h_w(\mathbf{x}) \geq 0; \\ -1, & \text{if } y h_w(\mathbf{x}) < 0. \end{cases}$$

We want to find parameters $(w_0, \mathbf{w})$ such that the distance between the misclassified examples and the decision boundary is minimised.



Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

- If we take derivatives with respect to the weights:

$$\frac{\partial \hat{L}(\mathbf{w})}{\partial w_0} = -\sum_{i' \in \mathcal{I}} y_{i'},$$

$$\nabla \hat{L}(\mathbf{w}) = -\sum_{i' \in \mathcal{I}} y_{i'} \mathbf{x}_{i'}.$$

Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

- If we take derivatives with respect to the weights:

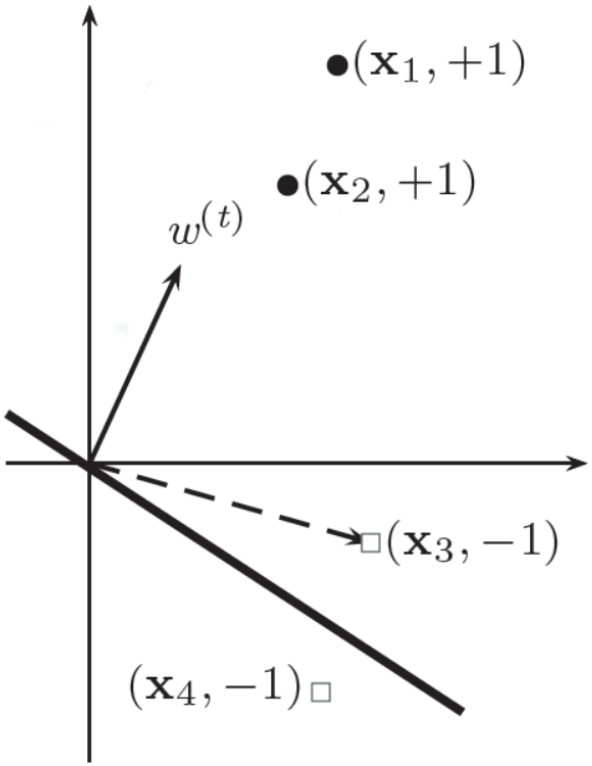
$$\frac{\partial \hat{L}(\mathbf{w})}{\partial w_0} = -\sum_{i' \in \mathcal{I}} y_{i'},$$

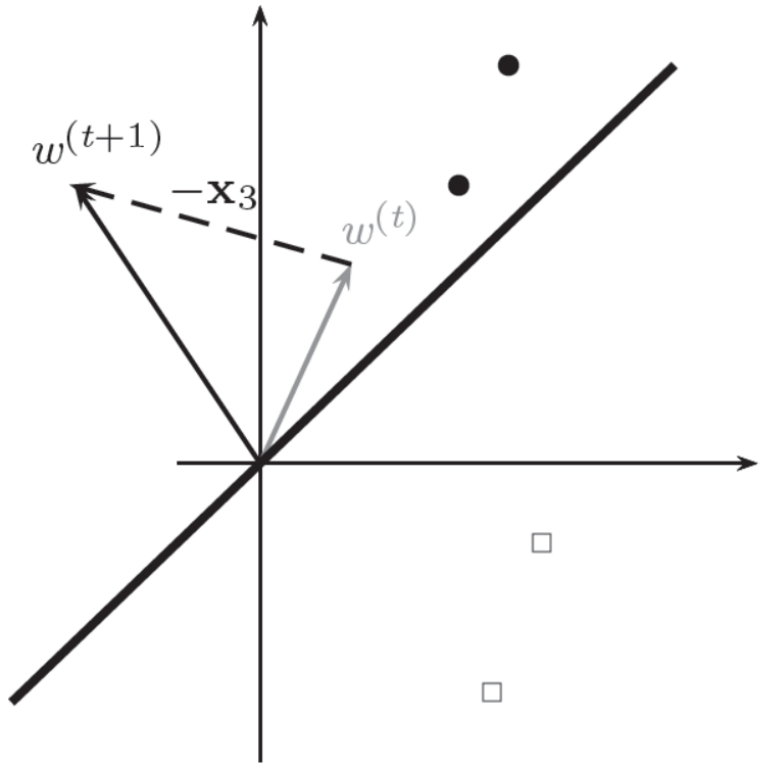
$$\nabla \hat{L}(\mathbf{w}) = -\sum_{i' \in \mathcal{I}} y_{i'} \mathbf{x}_{i'}.$$

- Minimisation of the perceptron error is done by stochastic gradient descent algorithm:

$$\text{if } y(\langle \mathbf{w}, \mathbf{x} \rangle + w_0) < 0, \text{then } \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} + \eta \begin{pmatrix} y \\ y\mathbf{x} \end{pmatrix}.$$

Update Rule: Illustration





Algorithm Perceptron

Input: Training set $\mathrm{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
Max number of iterations T.

Initialisation: Weights $\mathbf{w}^{(0)} \leftarrow \mathbf{0}, w_0 \leftarrow 0$;
Counter $t \leftarrow 0$, learning rate $\eta > 0$.

repeat
Choose randomly an example $(\mathbf{x}^{(t)}, y^{(t)}) \in \mathrm{S}$
if $y \cdot (\langle \mathbf{w}^{(t)}, \mathbf{x}^{(t)} \rangle + w_0) < 0$ **then**

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta\, y^{(t)}$$
$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta\, y^{(t)}\, \mathbf{x}^{(t)}$$

end if
$t \leftarrow t + 1$
until $t > T$
Output: $(w_0^{(T)}, \mathbf{w}^{(T)})$.

Perceptron: Discussion

- What is the time complexity during training phase?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?
- Does the algorithm converge to an exact solution?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?
- Does the algorithm converge to an exact solution?
- Under what condition the algorithm does not converge?

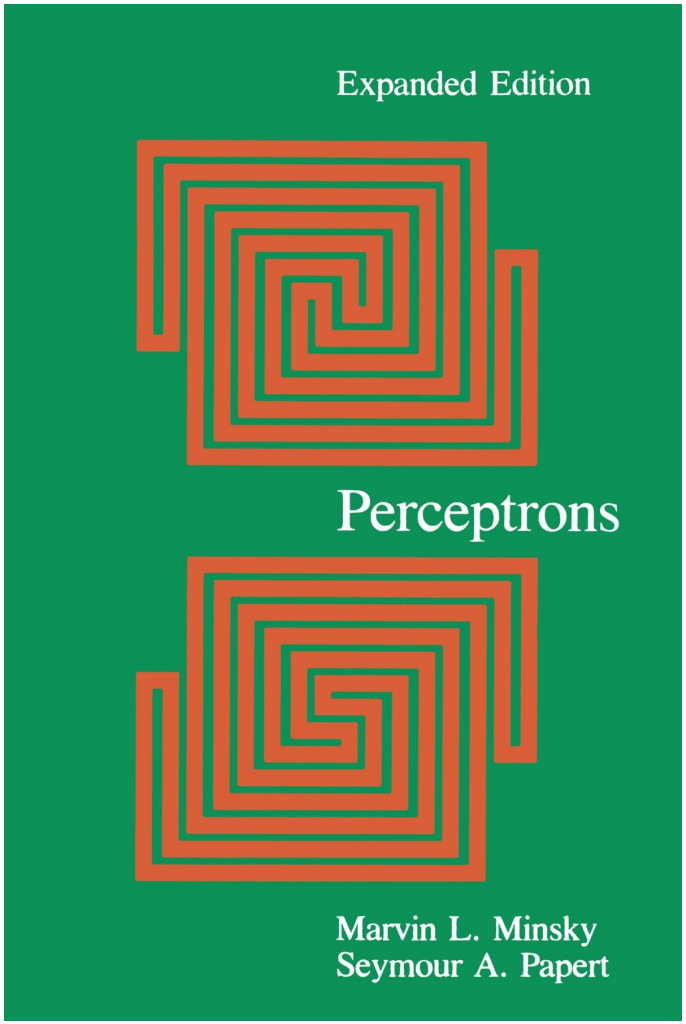


Figure: (Minsky & Papert, 1969)

Figure: Minsky and Papert in 1971.

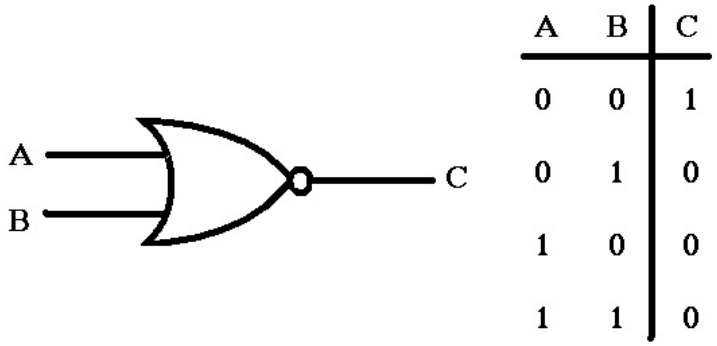
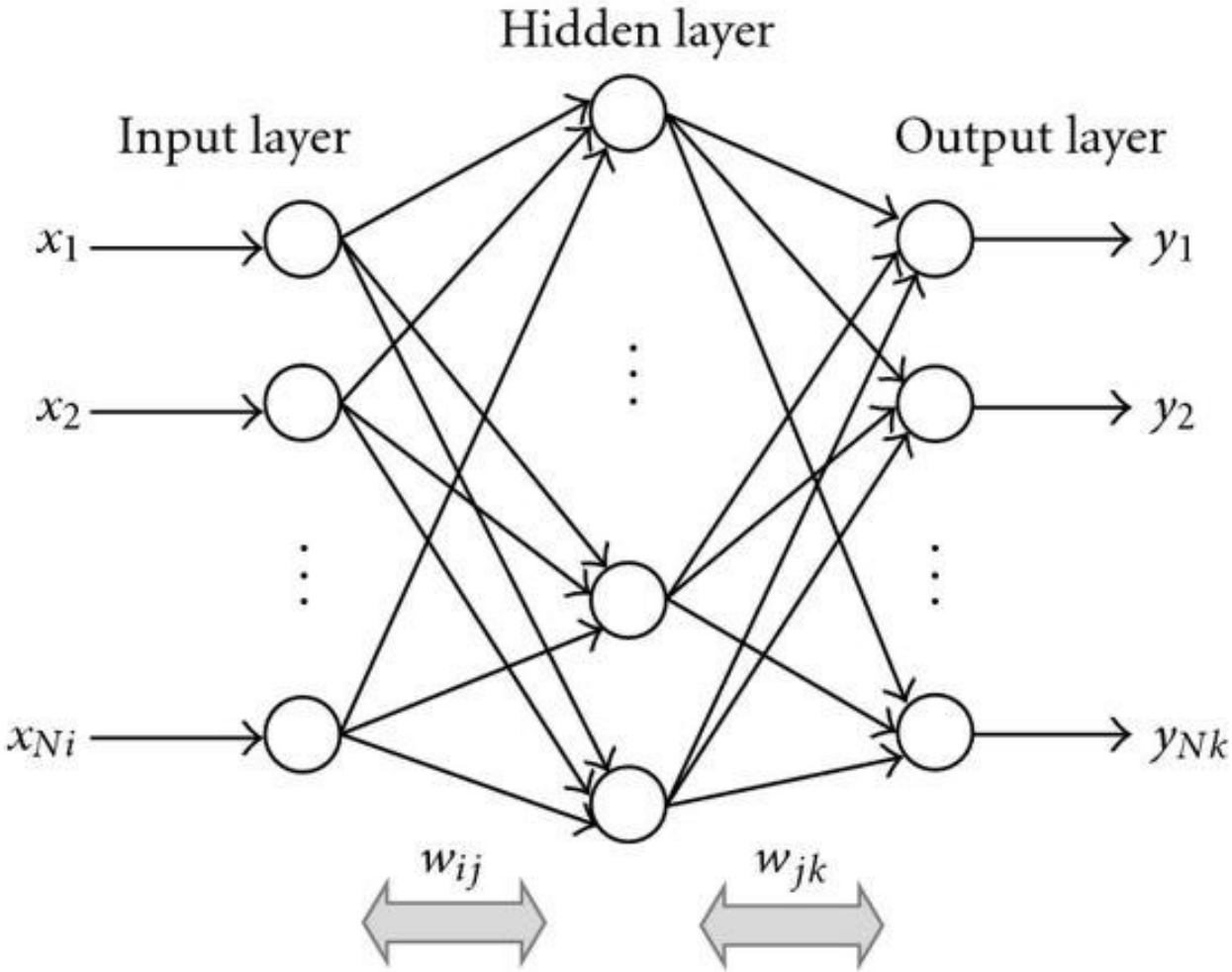


Figure: The XOR operation.

Backpropagation Learning (Rumelhart, 1985)



Nowadays: Deep Learning