Supervised Learning Algorithms

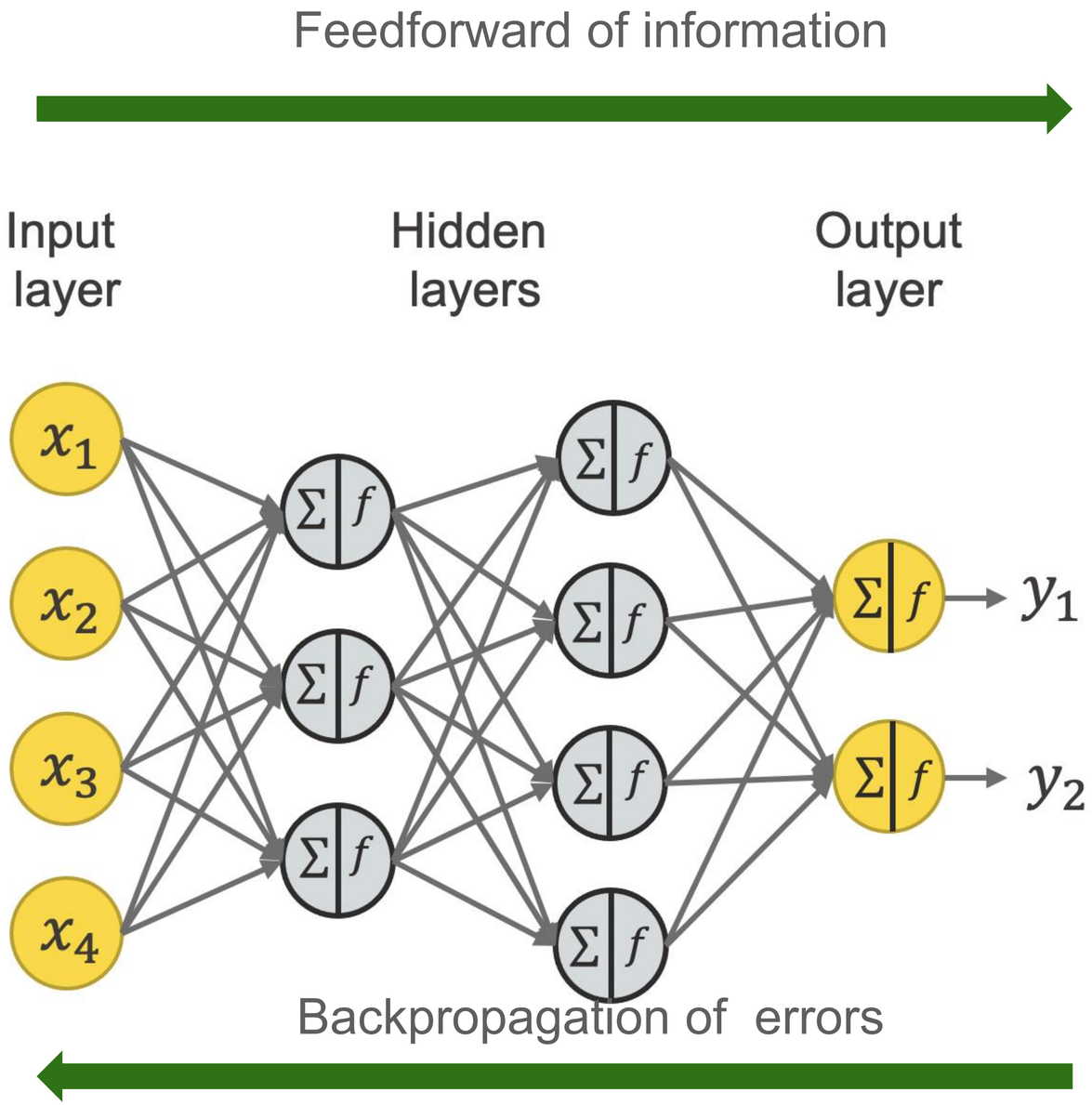
Exam /presentation

- 15.05.25
- 45 min
- Without any supporting material

- Final project presentations
- 22.05.25 (no class on 29.05 and no excercise on 30.05)

- Guest lecture on 08.05.25

Recap: Basic principles neural network



Gradient descent

- **Objective:** The primary objective of gradient descent is to find the set of parameters that minimizes the objective function.
- **Gradient:** The gradient of the function at a particular point is a vector that points in the direction of the steepest ascent. By moving in the opposite direction of the gradient, we move towards the minimum.

Supervised learning algorithms

Summary

- Support Vector Machines
- Naive Bayes
- Decision trees
- Ensembles / bagging / boosting

Support Vector Machine

Example

- Imagine a civil engineering project aimed at developing a new residential area near a hill slope known for its landslide activity. An SVM model is trained using historical data on landslides in the region, incorporating factors such as slope, soil type, rainfall intensity, and vegetation cover. The trained model is then used to assess the susceptibility of the new area to landslides, providing valuable information for planning and mitigation strategies, such as reinforcing slopes or altering drainage patterns to reduce risk.

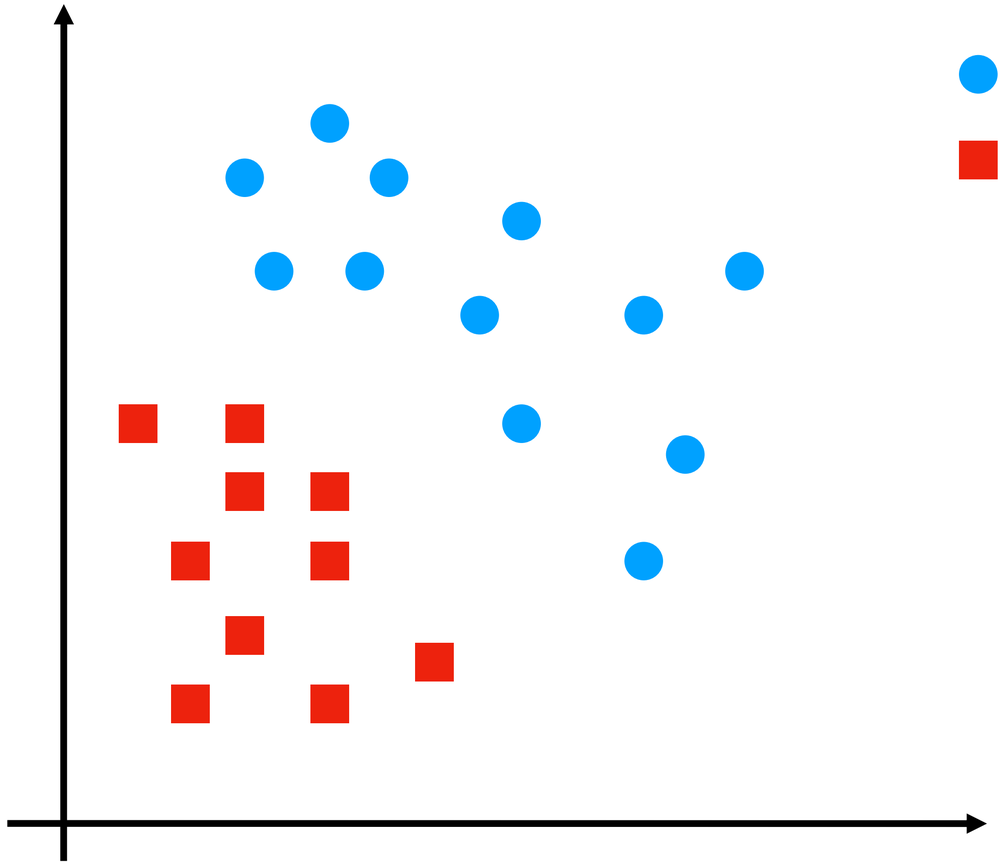
SVM
Introduction

Why this additional classification technique ?

- Easy to implement and test
- It can provide good results on small datasets
- Has sound theoretical guarantees
- It allows for a geometric way to think about supervised learning
- A nice review of what we have learned related to loss functions, optimisation…

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
```

SVM
Introduction



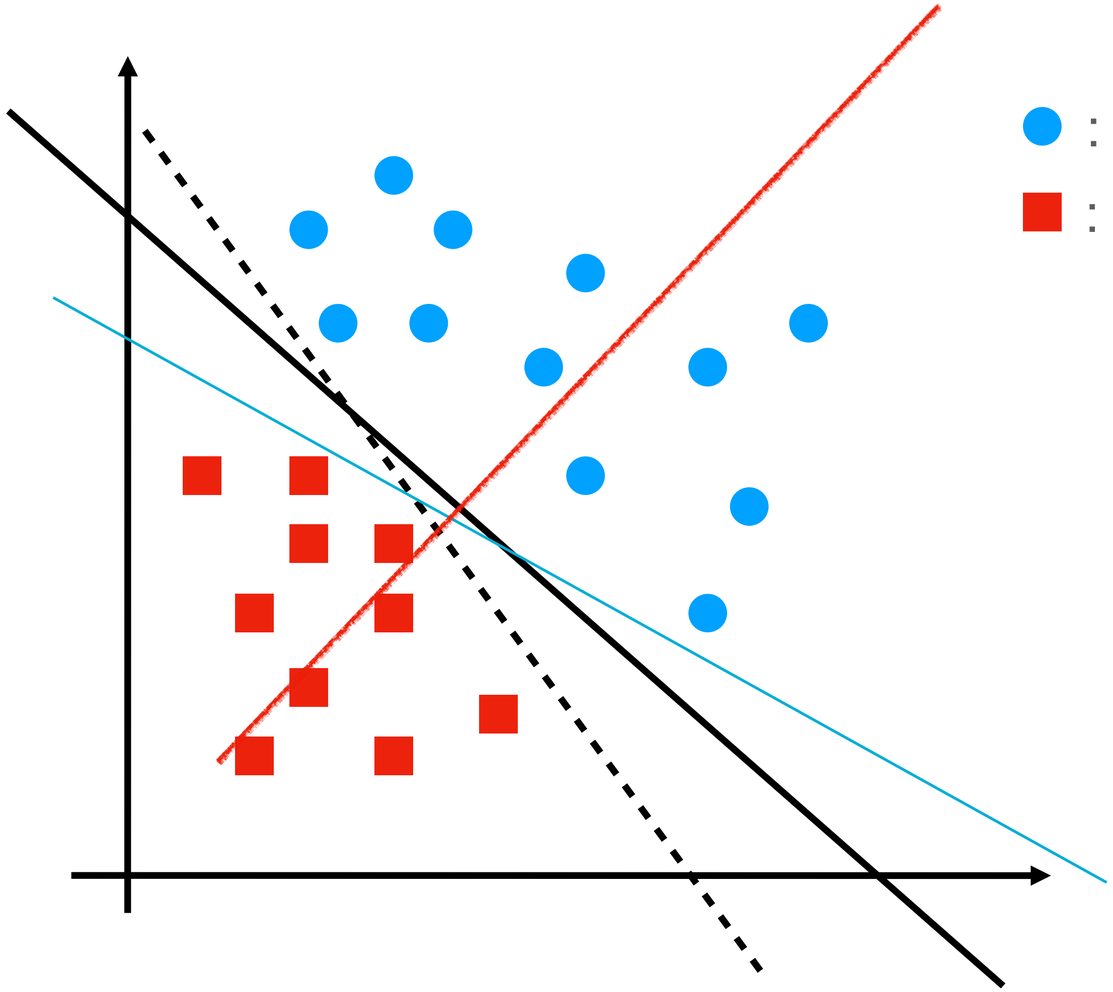
● : label as 1

■ : label as -1

Classification Algorithm

O. Fink /A. Alahi

SVM
Introduction



: label as 1

: label as -1

Classification Algorithm

Which line?

A:1

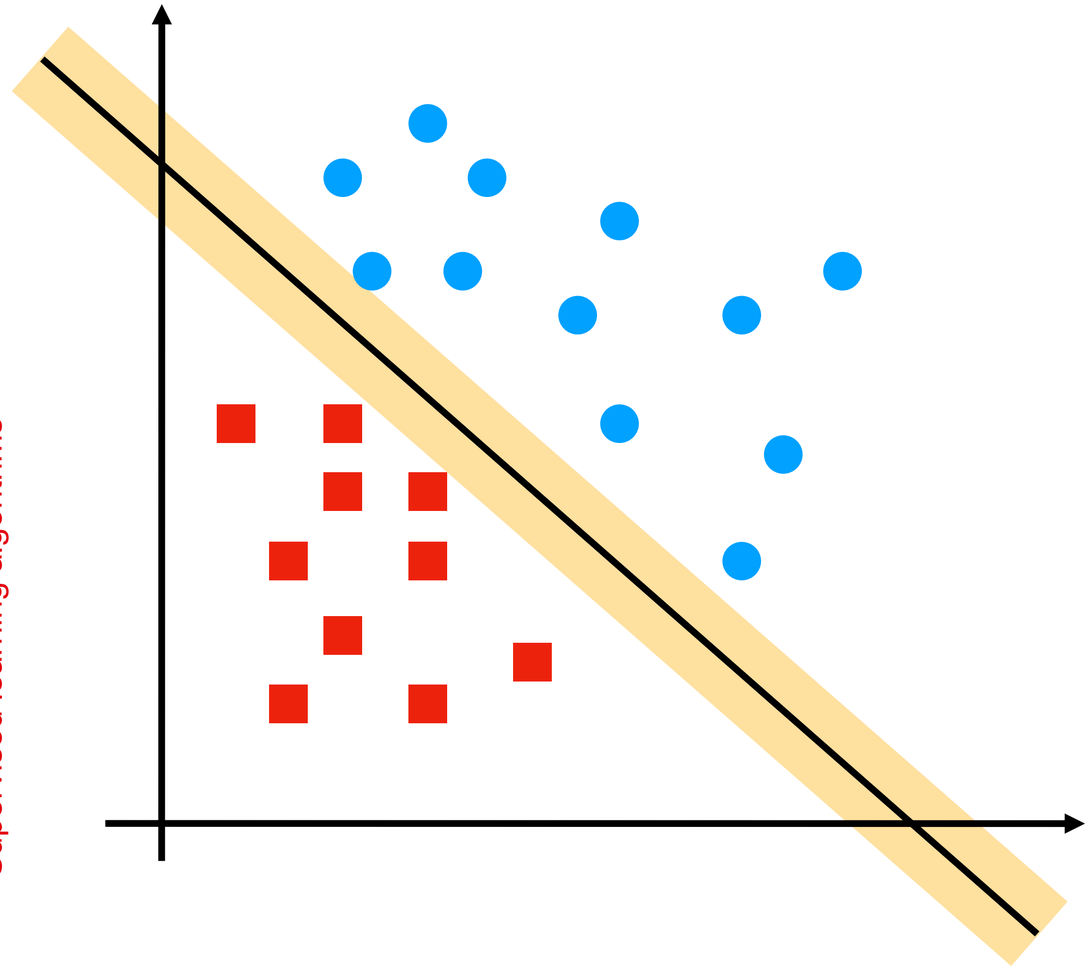
A:2

A:3

A:4

SVM
Introduction

What's SVM?



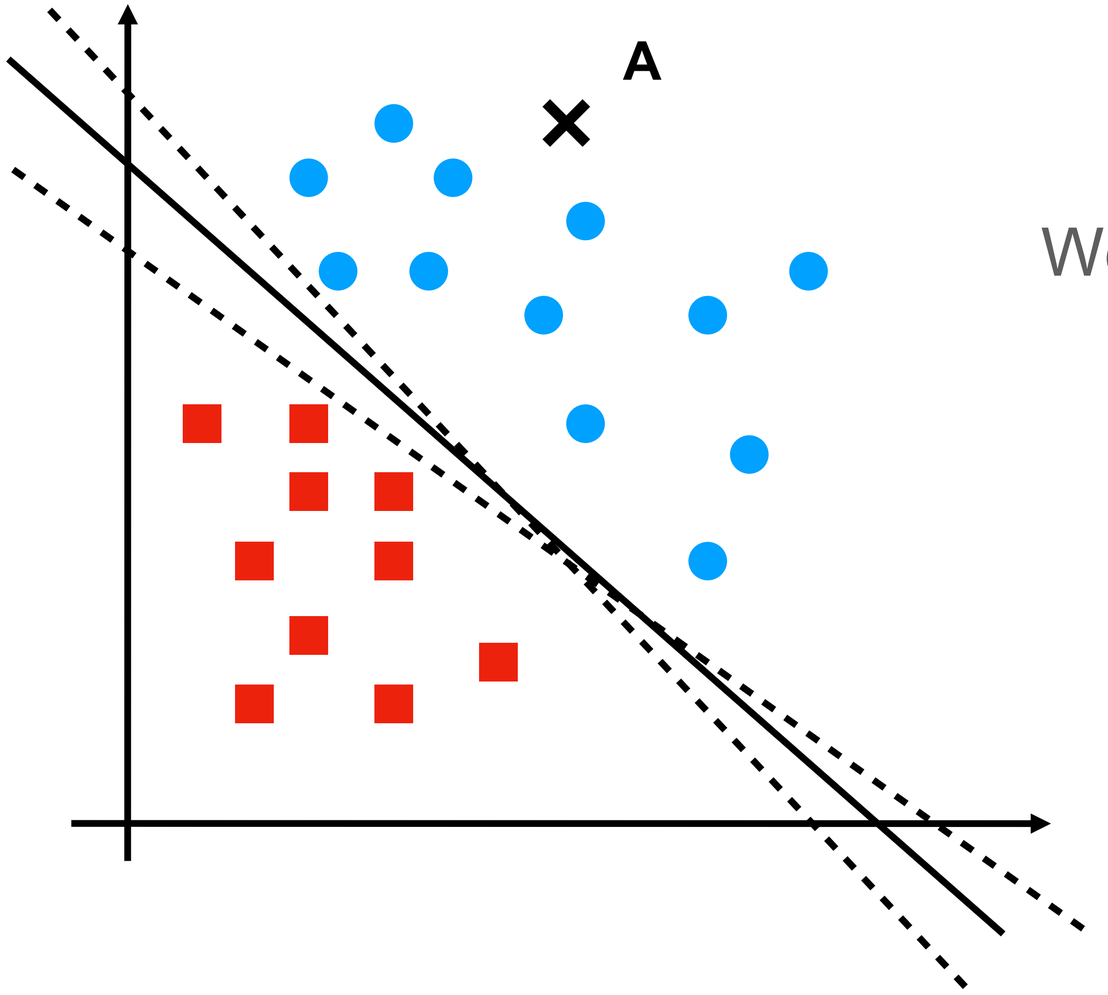
Classification Algorithm

+

Maximization of the margin

SVM
Introduction

Why is margin maximization interesting?



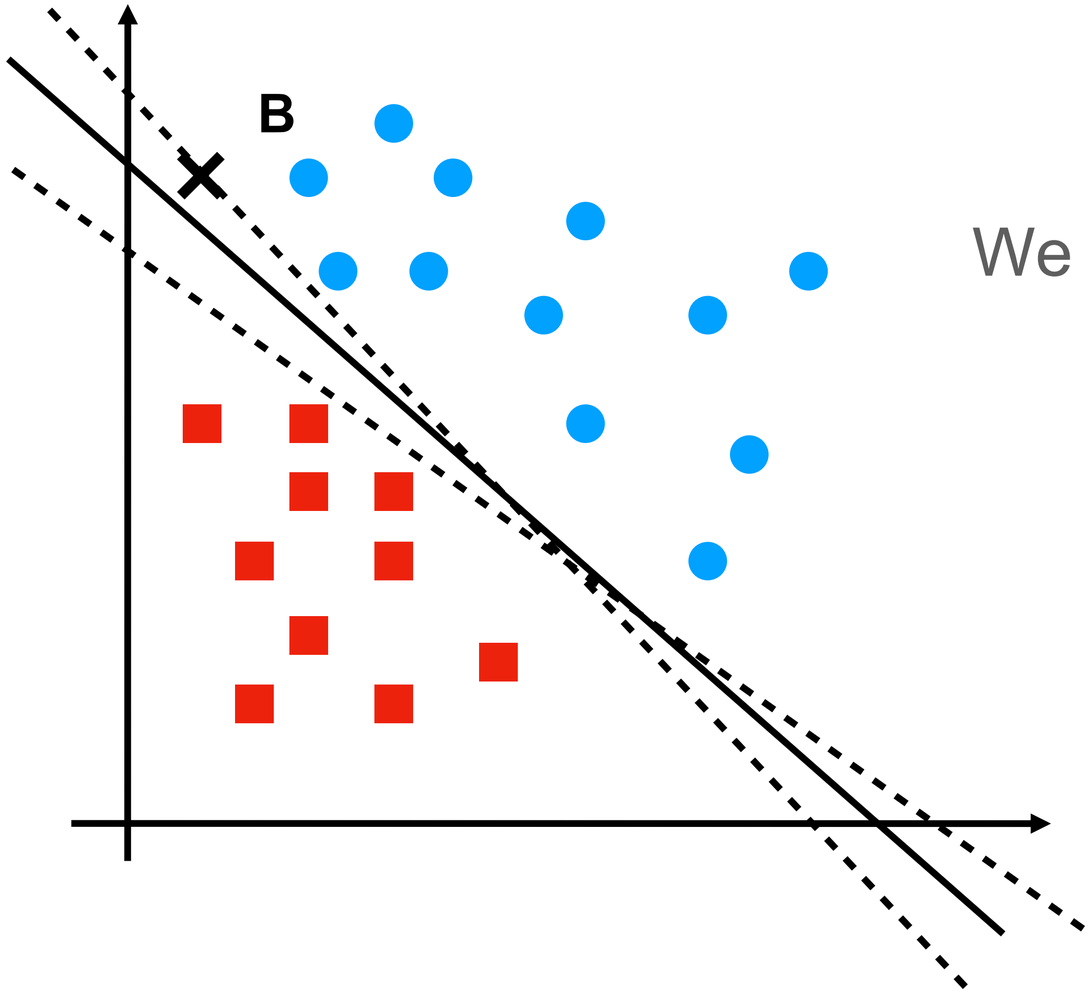
‘A’ **far** from the separation line

We are confident that ‘A’ is classified as blue

Small change in the separation line will not change this classification

SVM
Introduction

Why margin maximization is interesting?



‘B’ **close** from the separation line

We are less confident that ‘B’ is classified as blue

Small change in the separation line can change this classification

SVM
Hinge Loss

In this lecture:

- **labels** $y \in \{-1,1\}$
- **predicted values** $f(x) \in \mathbb{R}$ (also referred to as $\hat{y}$)

	f(x)<0	**f(x)>0**
y=-1	Correct	Incorrect
y=1	Incorrect	Correct

So:

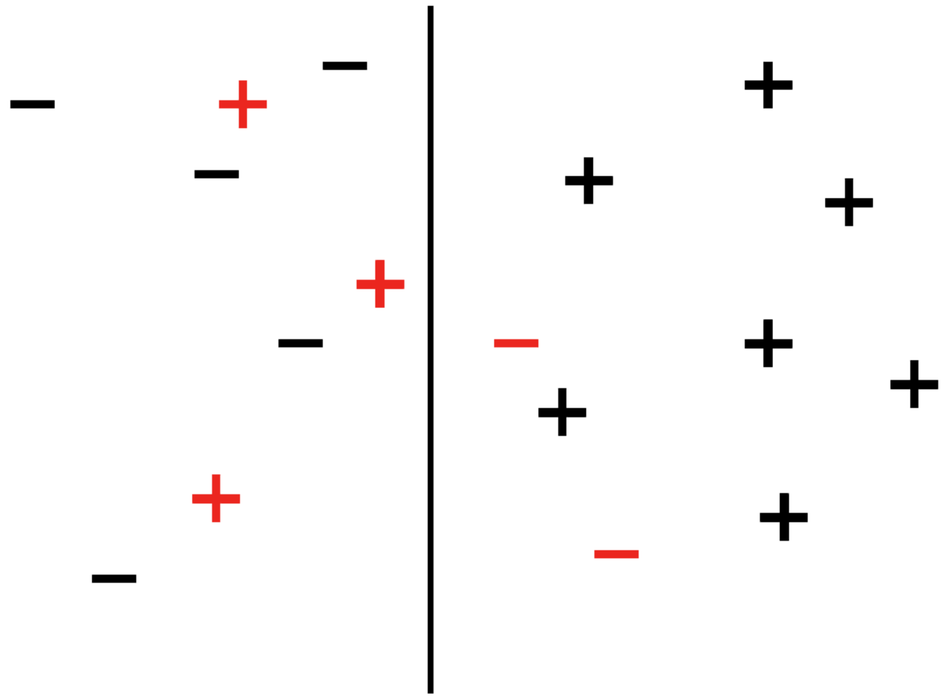
- $y \cdot f(x) > 0 \rightarrow$**correctly classified**
- $y \cdot f(x) < 0 \rightarrow$**incorrectly classified**

SVM
Hinge Loss

Correctly classified

Misclassified

$f(x) < 0$ $\quad$ $f(x) > 0$



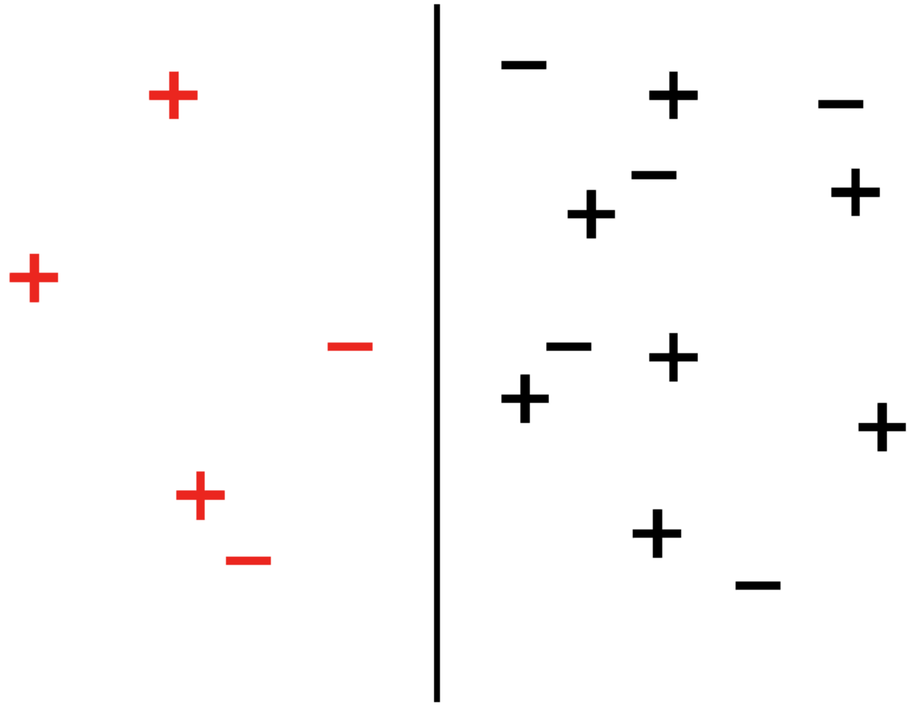
$f(x)$ **: Predicted value (also referred as** $\hat{y}$**)**

SVM
Hinge Loss

Correctly classified

Misclassified

$yf(x) < 0$ $yf(x) > 0$



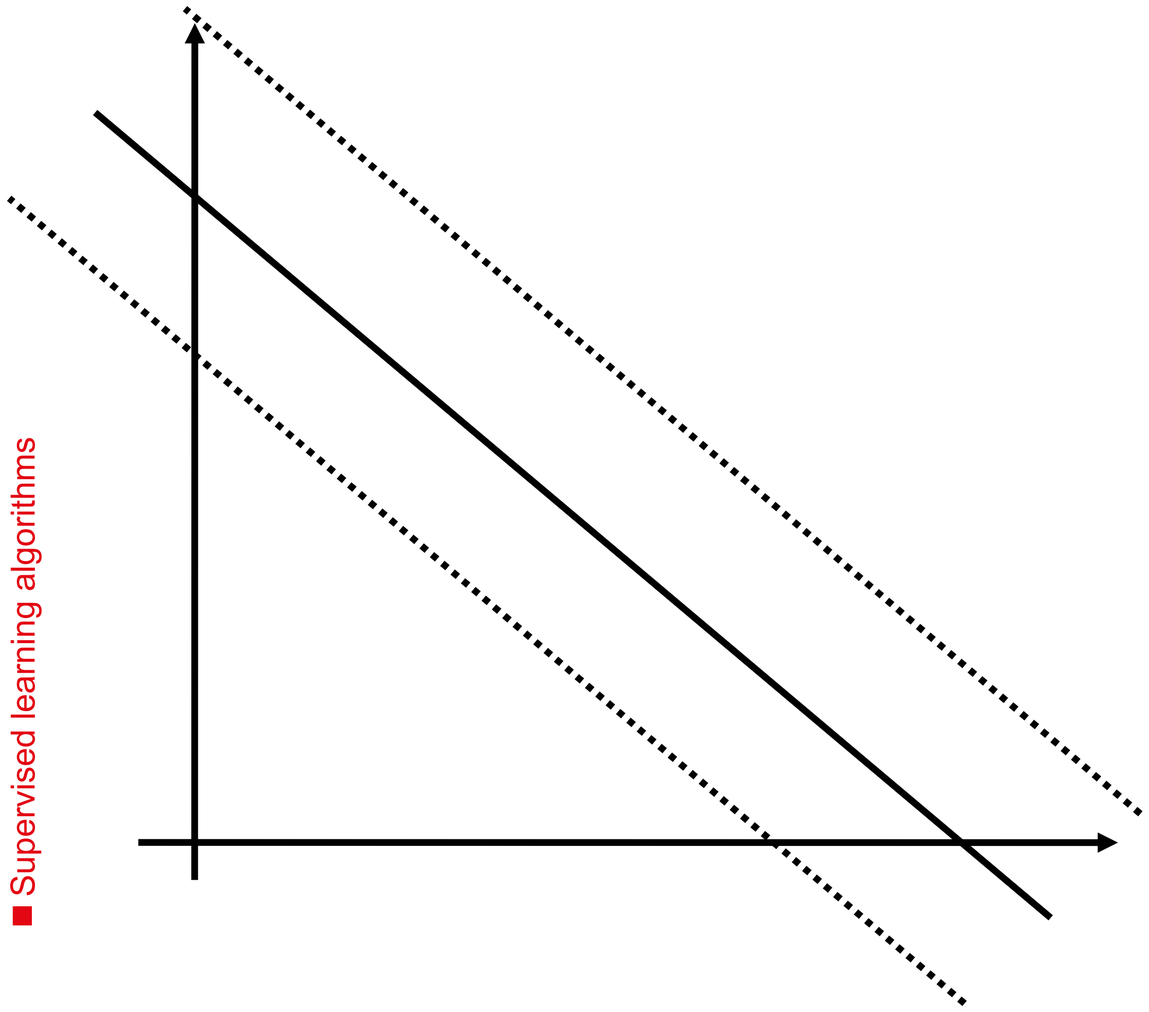
bringing all our misclassified points on one side of the decision boundary

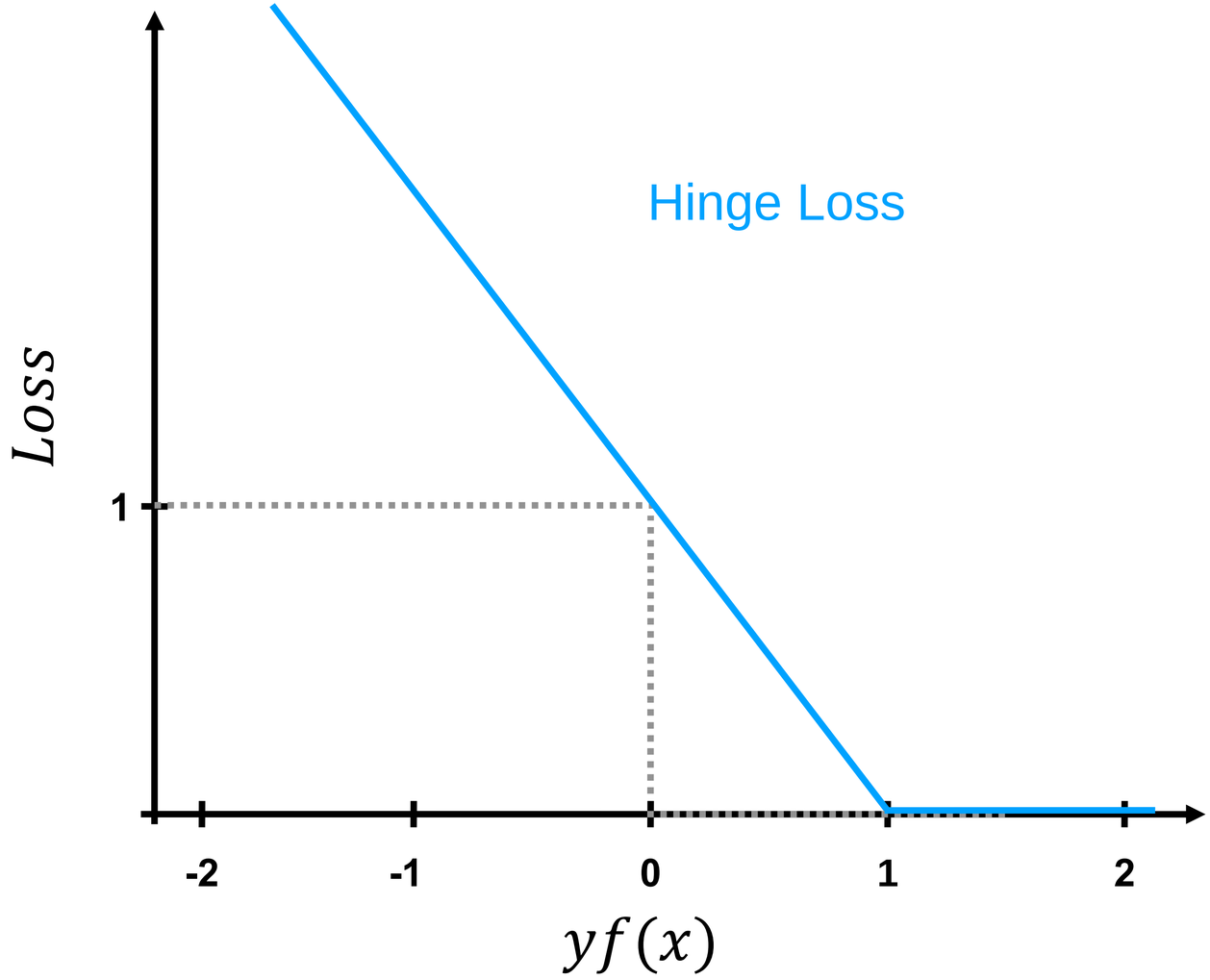
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function $Hinge(x, y) = max(0, 1 - y.f(x)), labels\ \ y \in \{-1, 1\}$



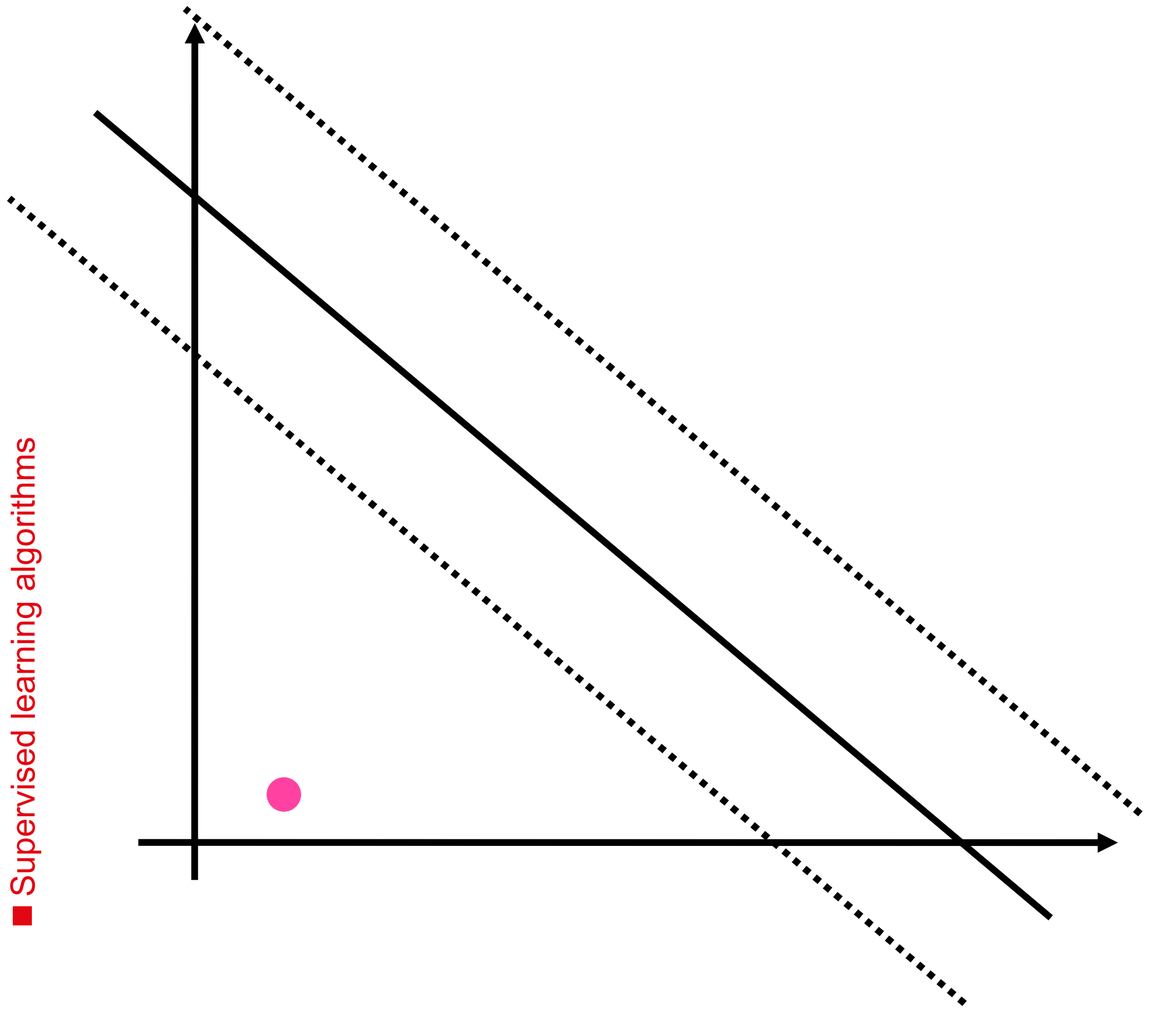


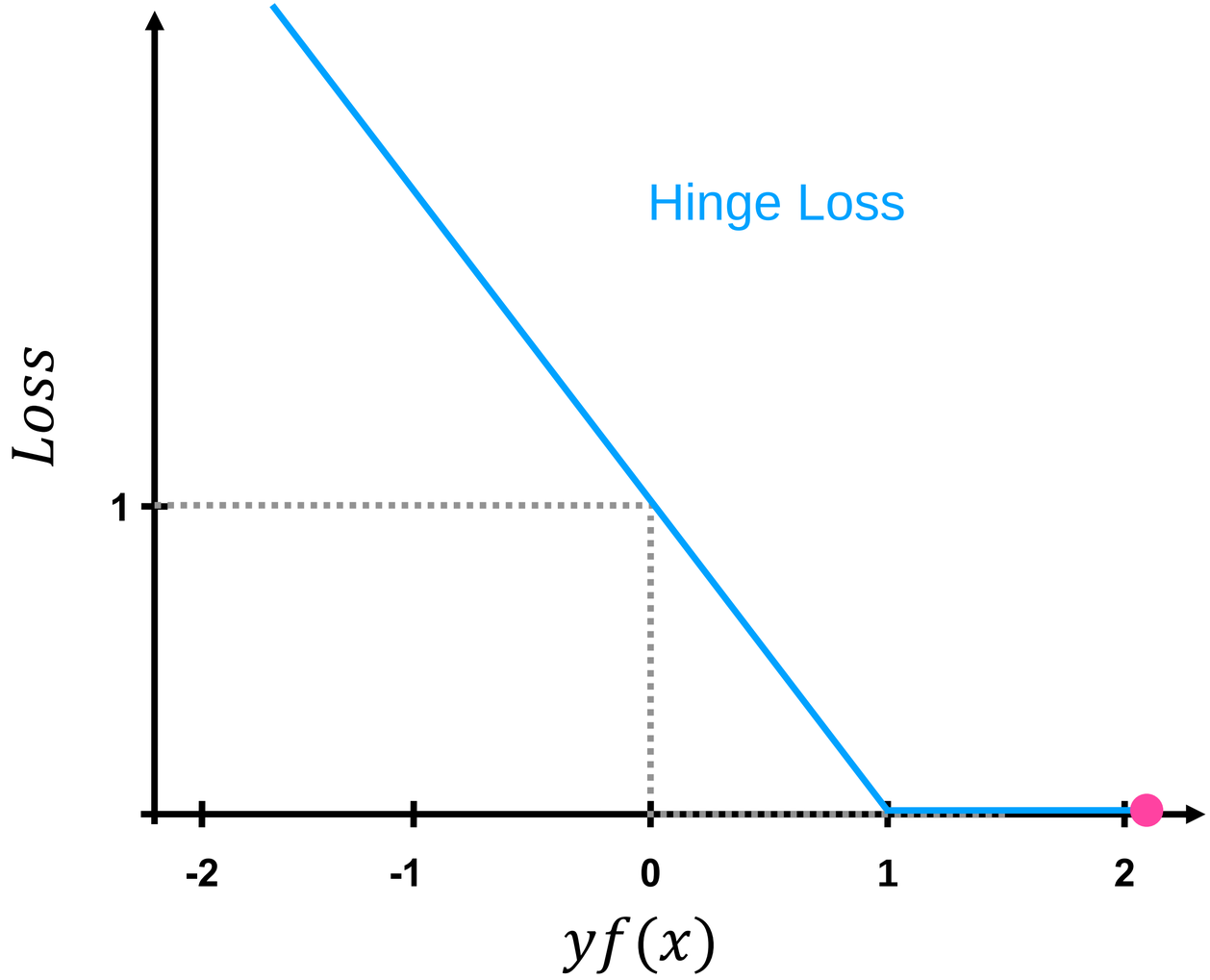
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function



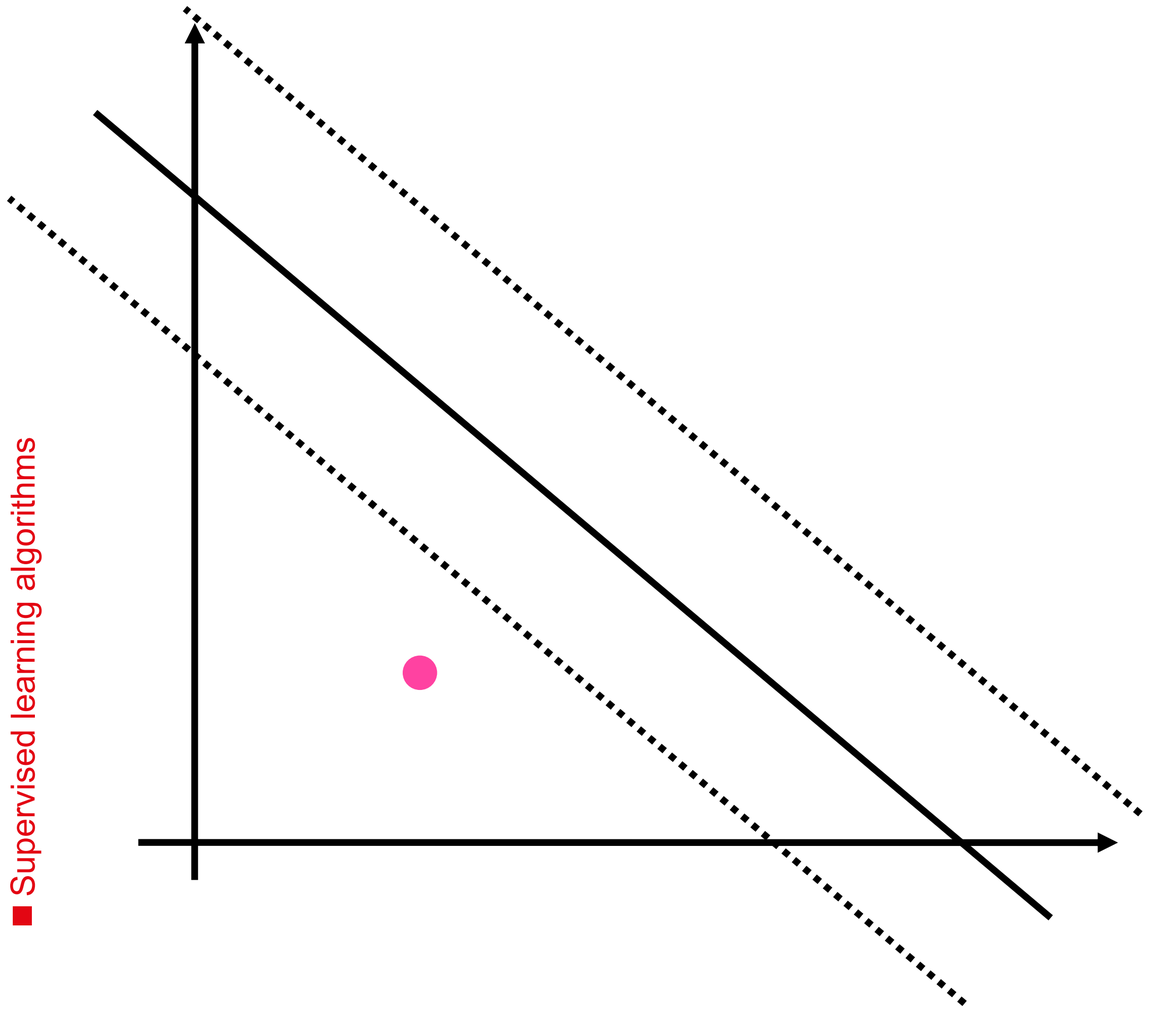


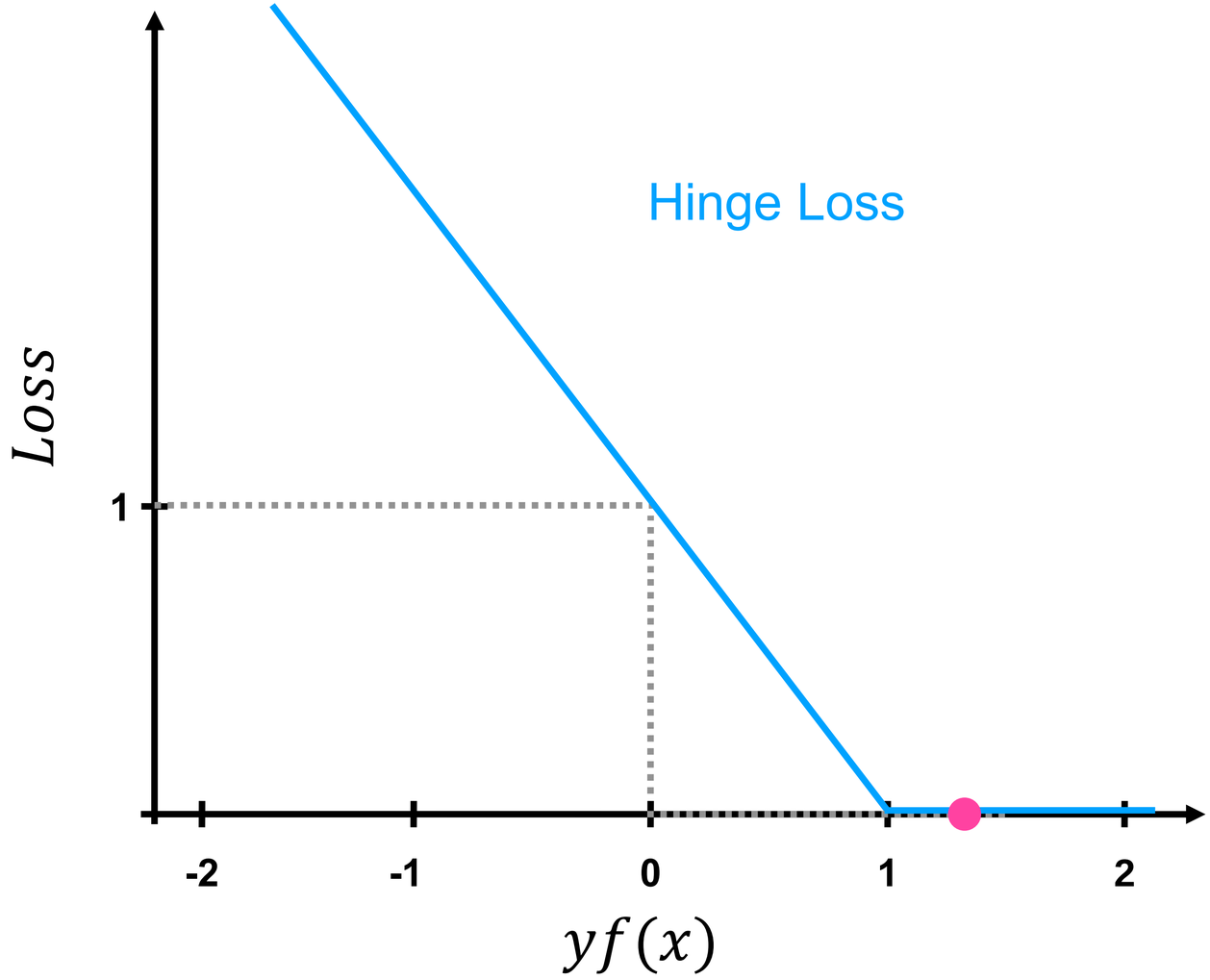
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function

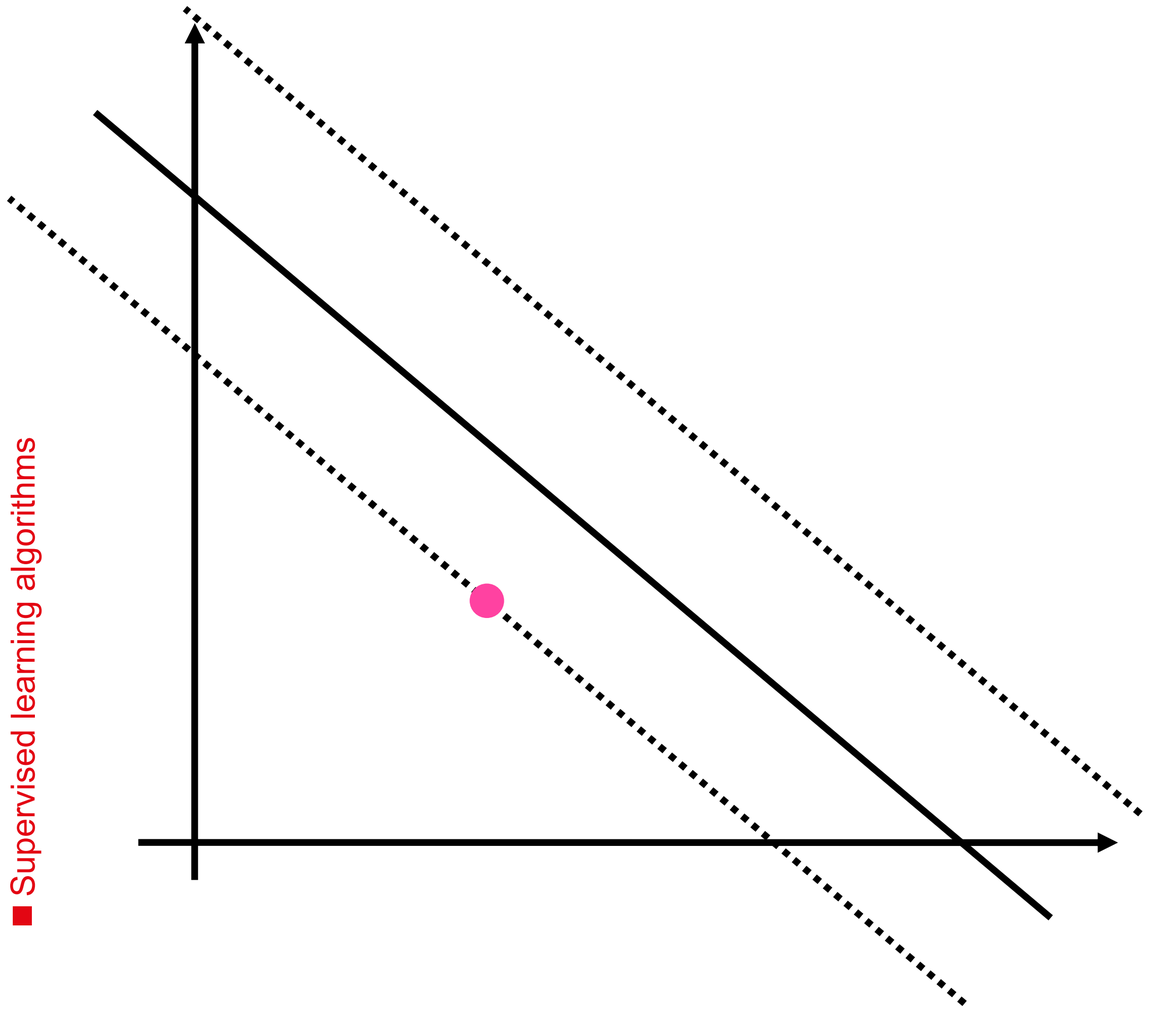


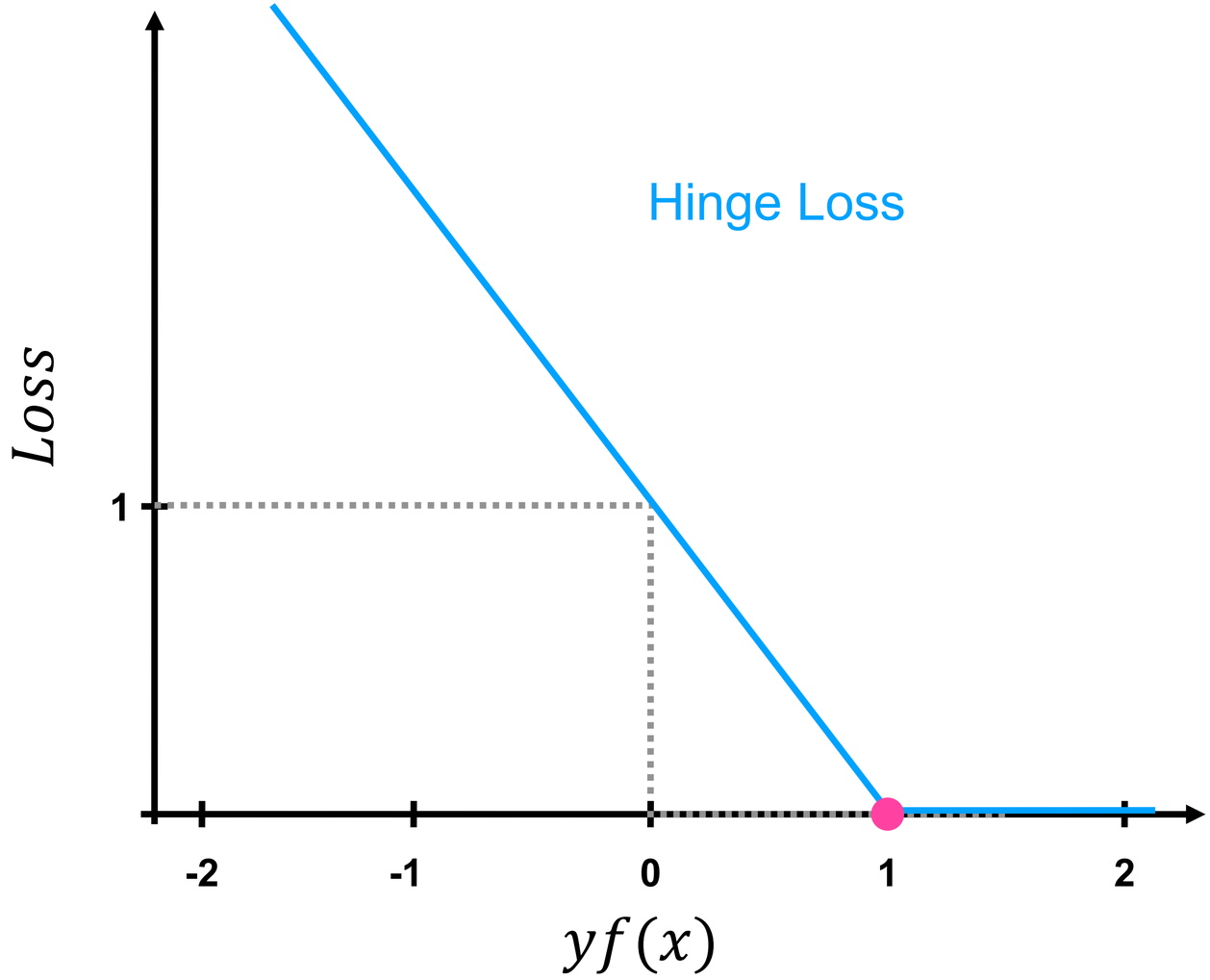


SVM

Optimal hyperplane separation

Link with the Hinge function



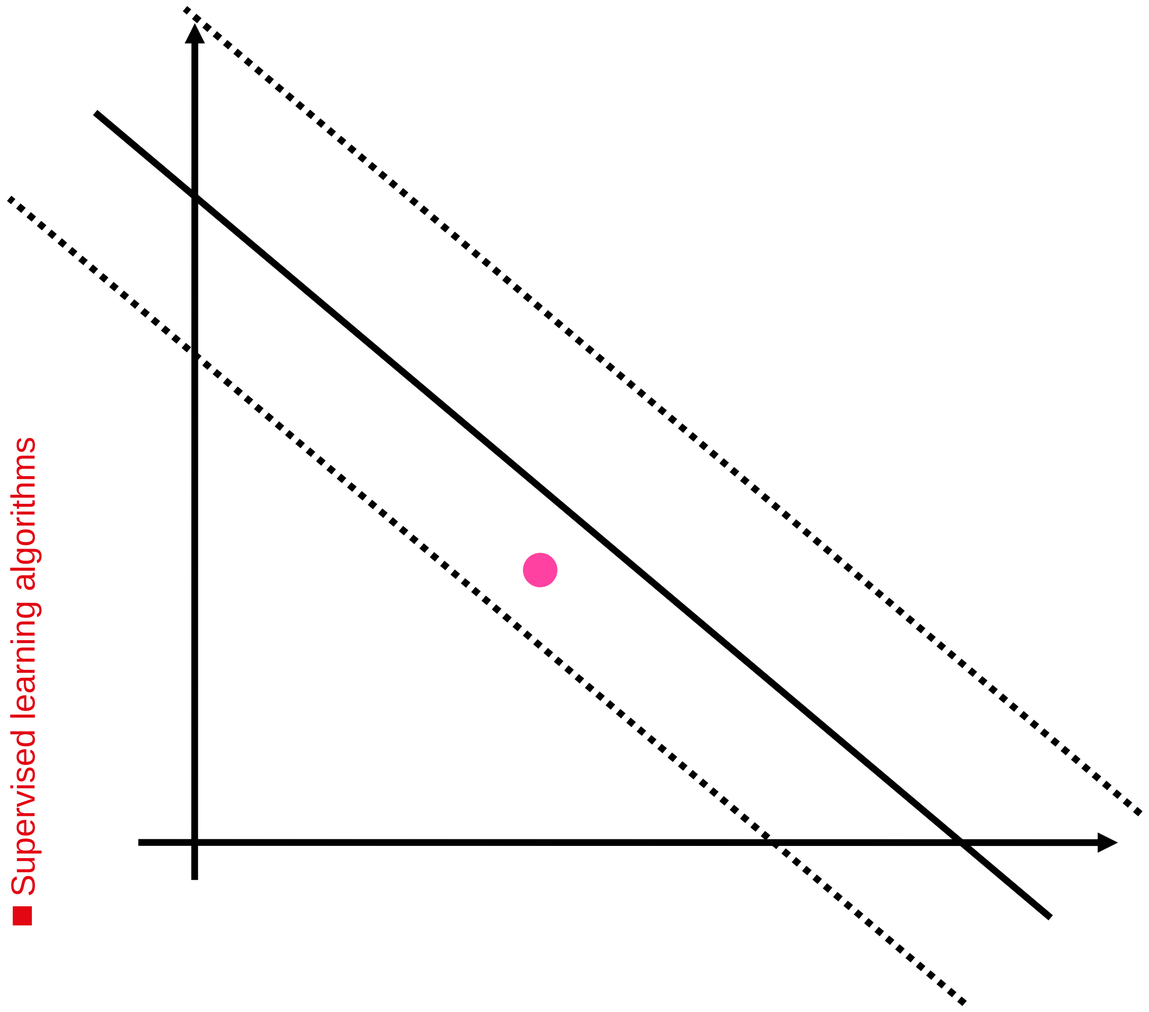


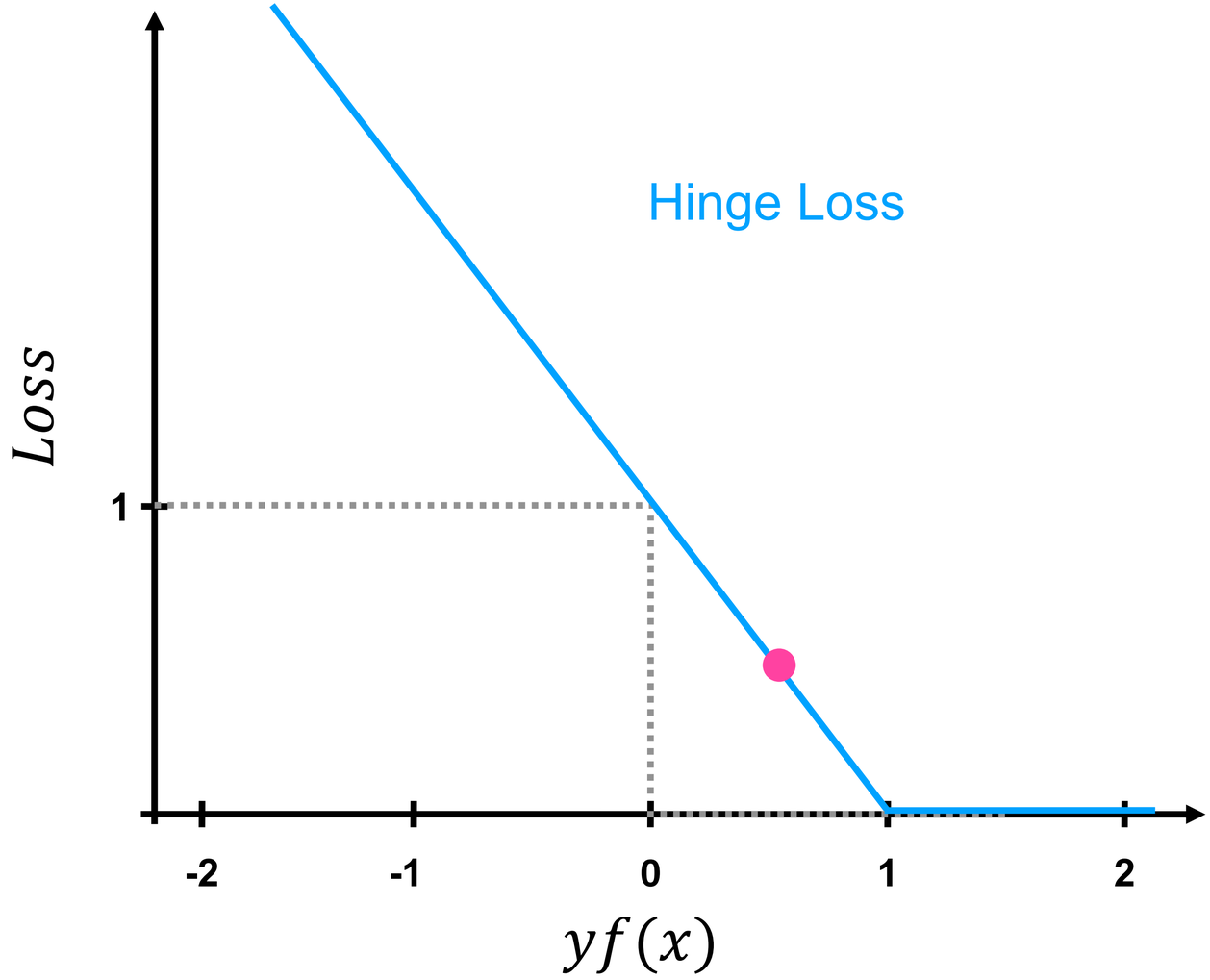
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function



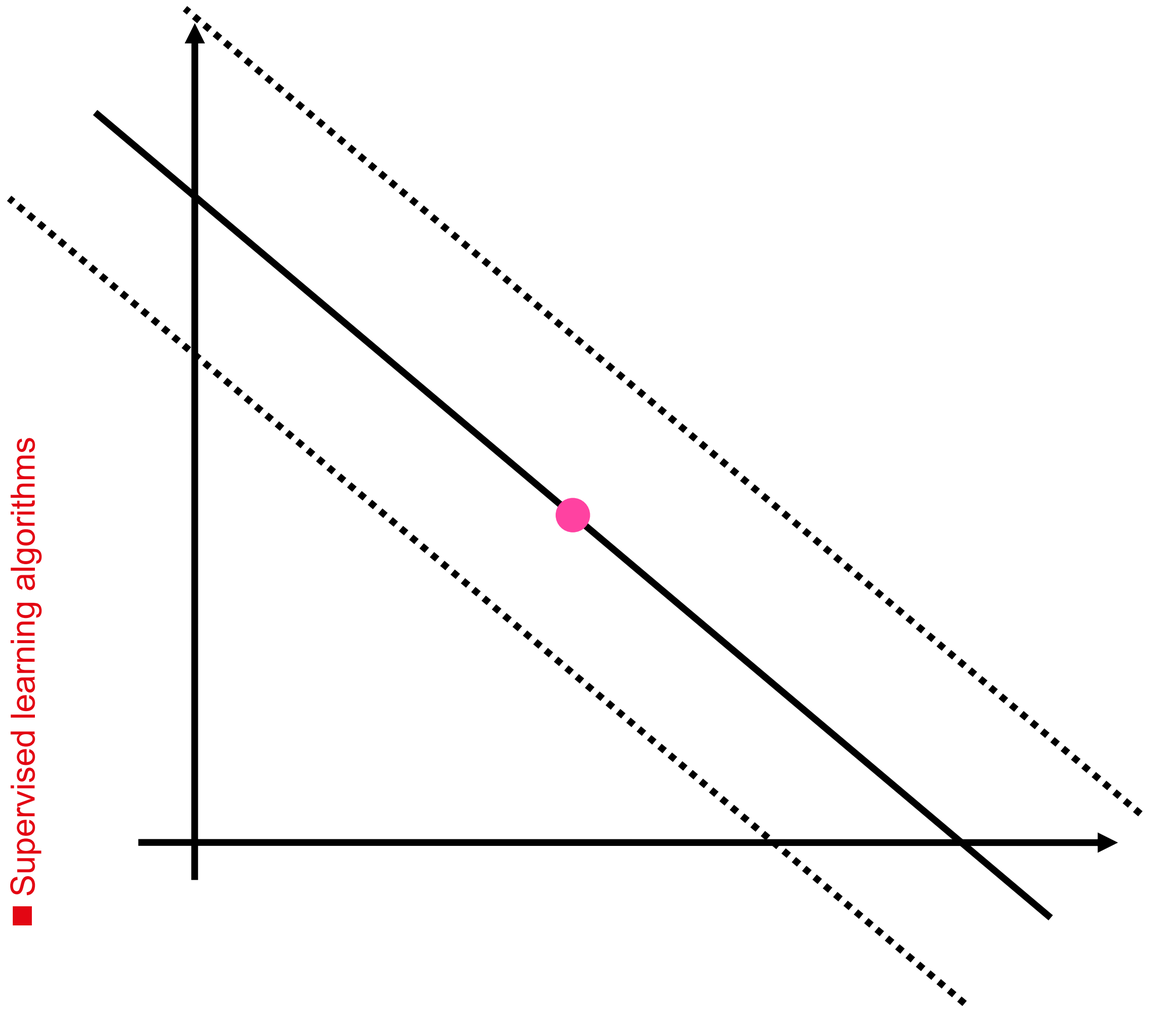


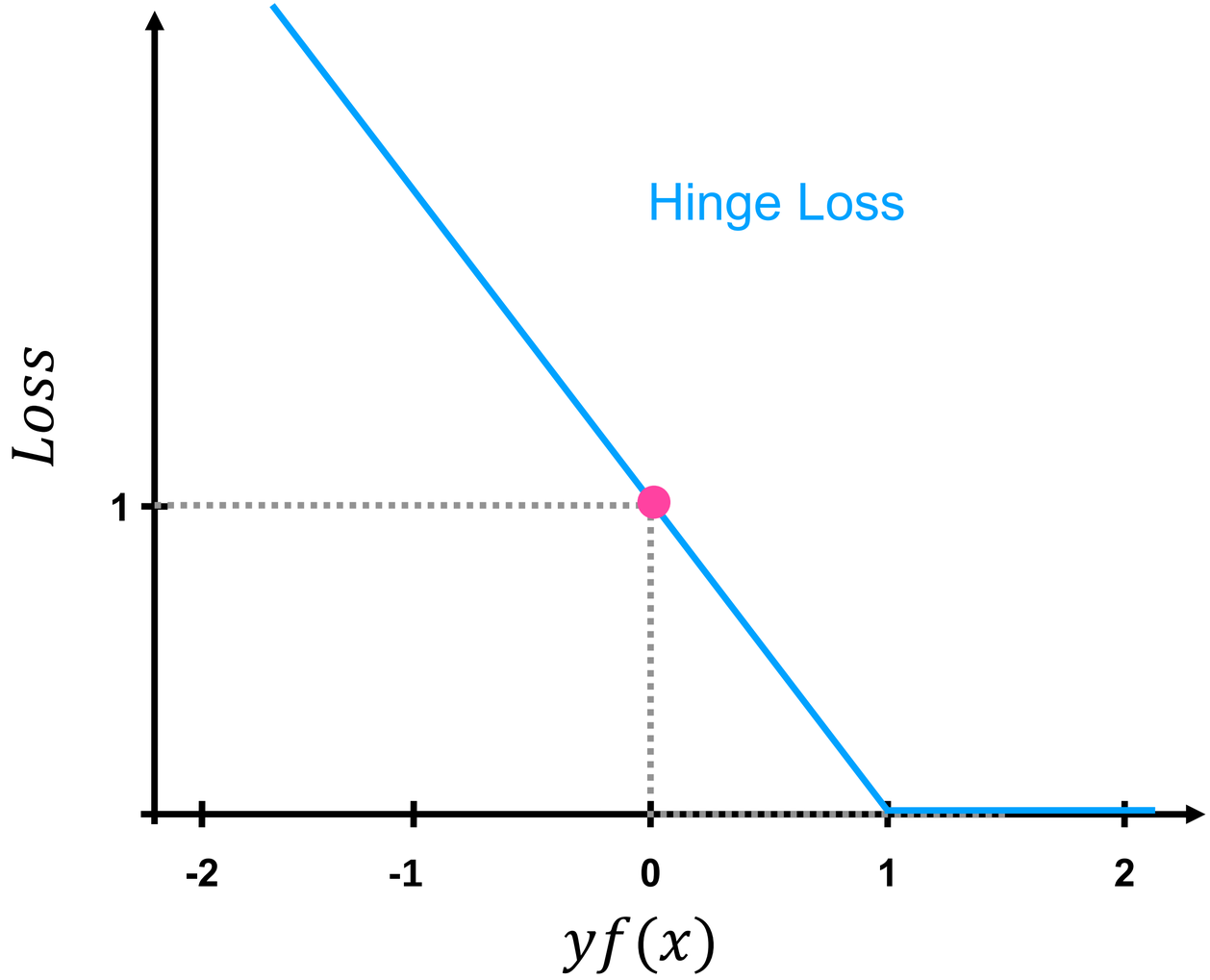
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function



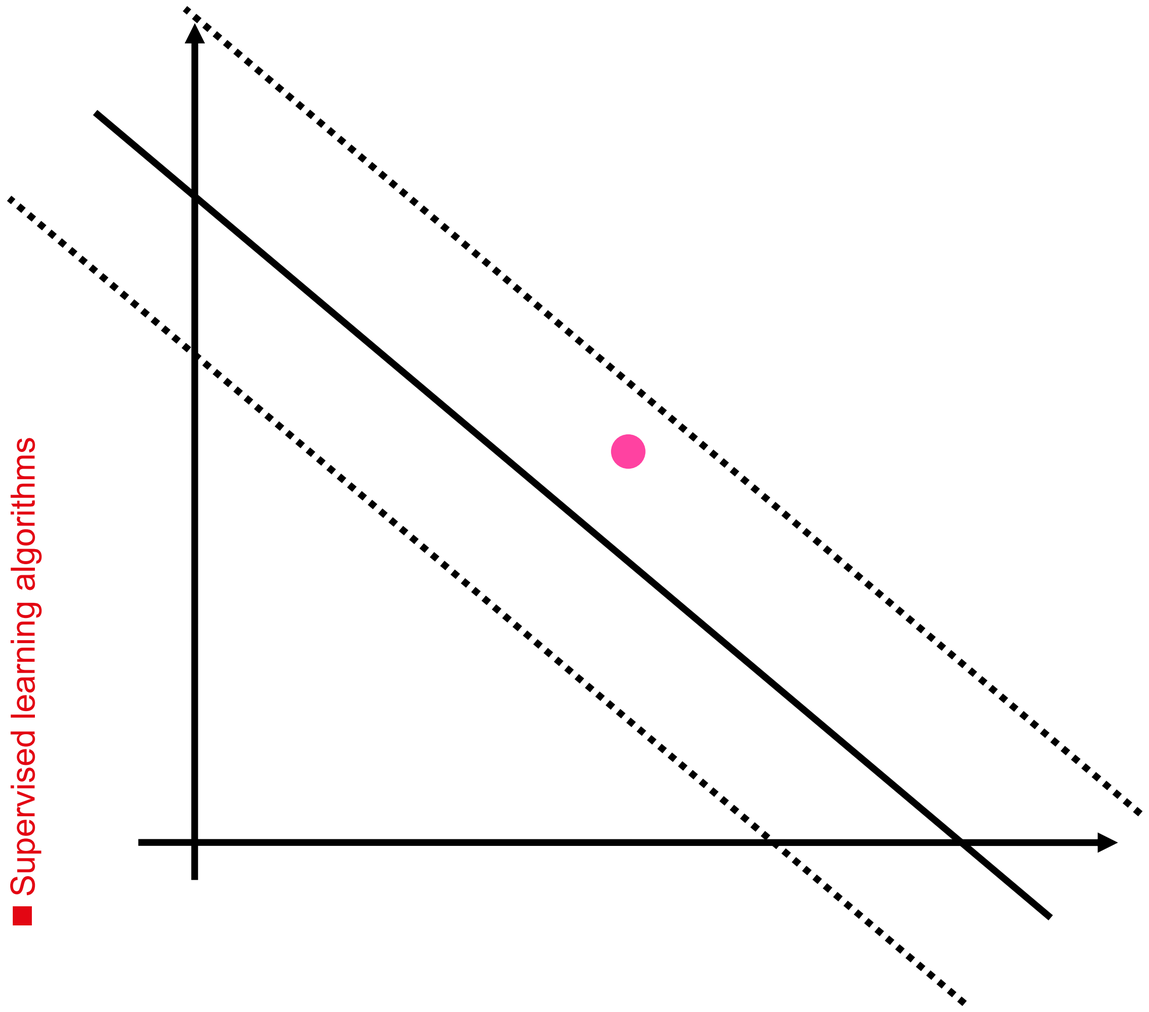


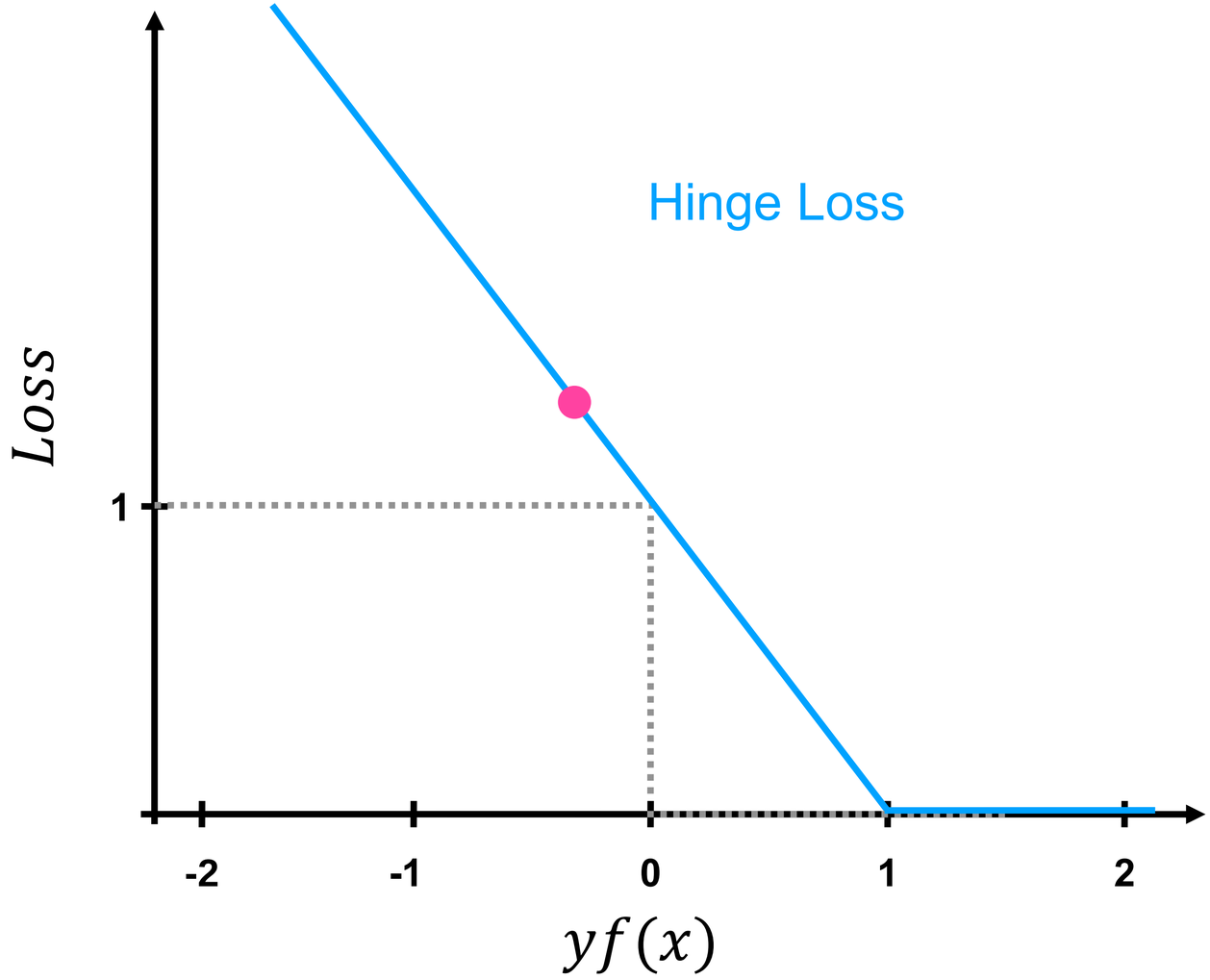
O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Link with the Hinge function

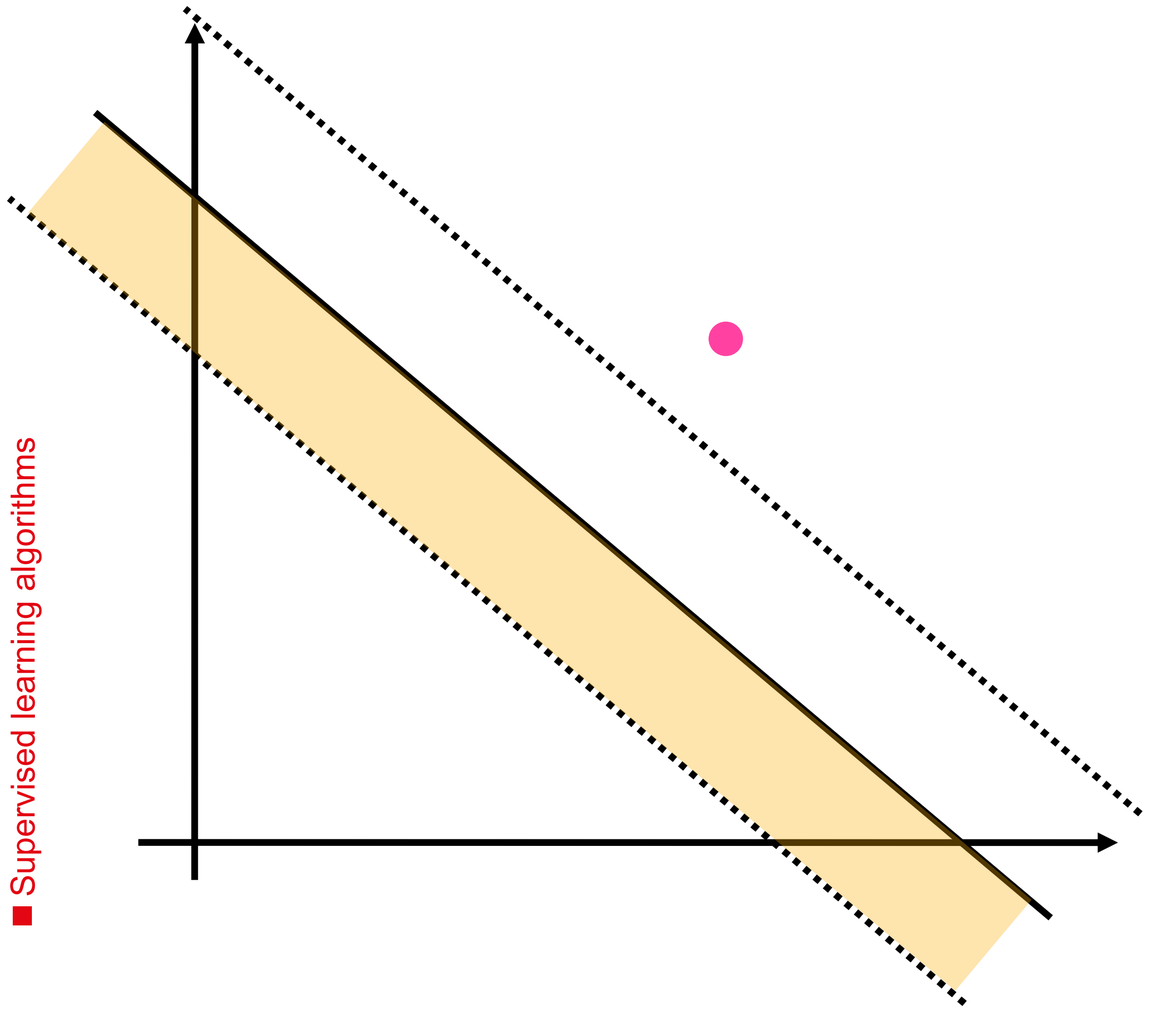


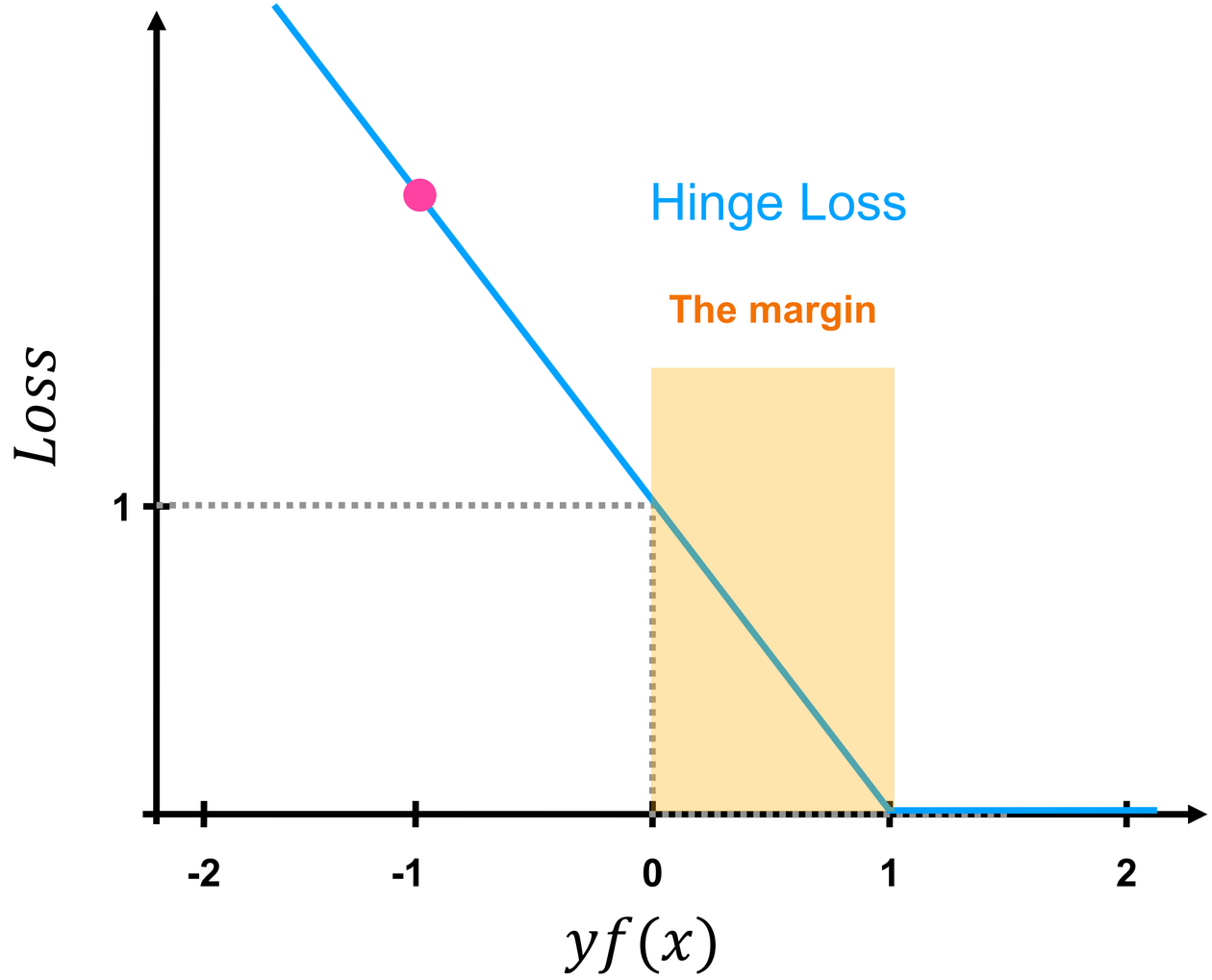


SVM

Optimal hyperplane separation

Link with the Hinge function

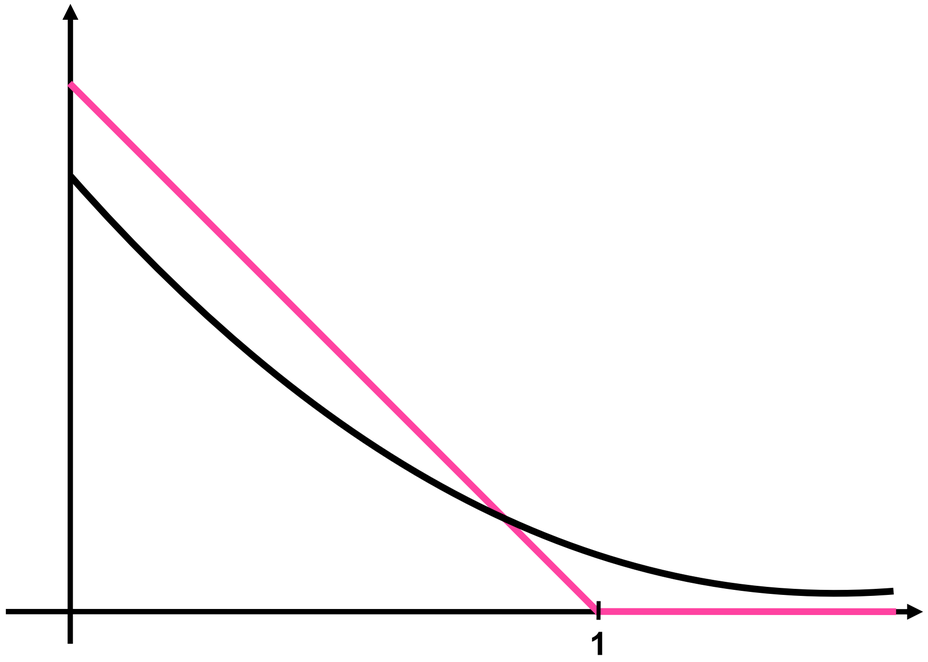




SVM

Alternative view of logistic regression

Remember Logistic Loss function?



Logistic loss

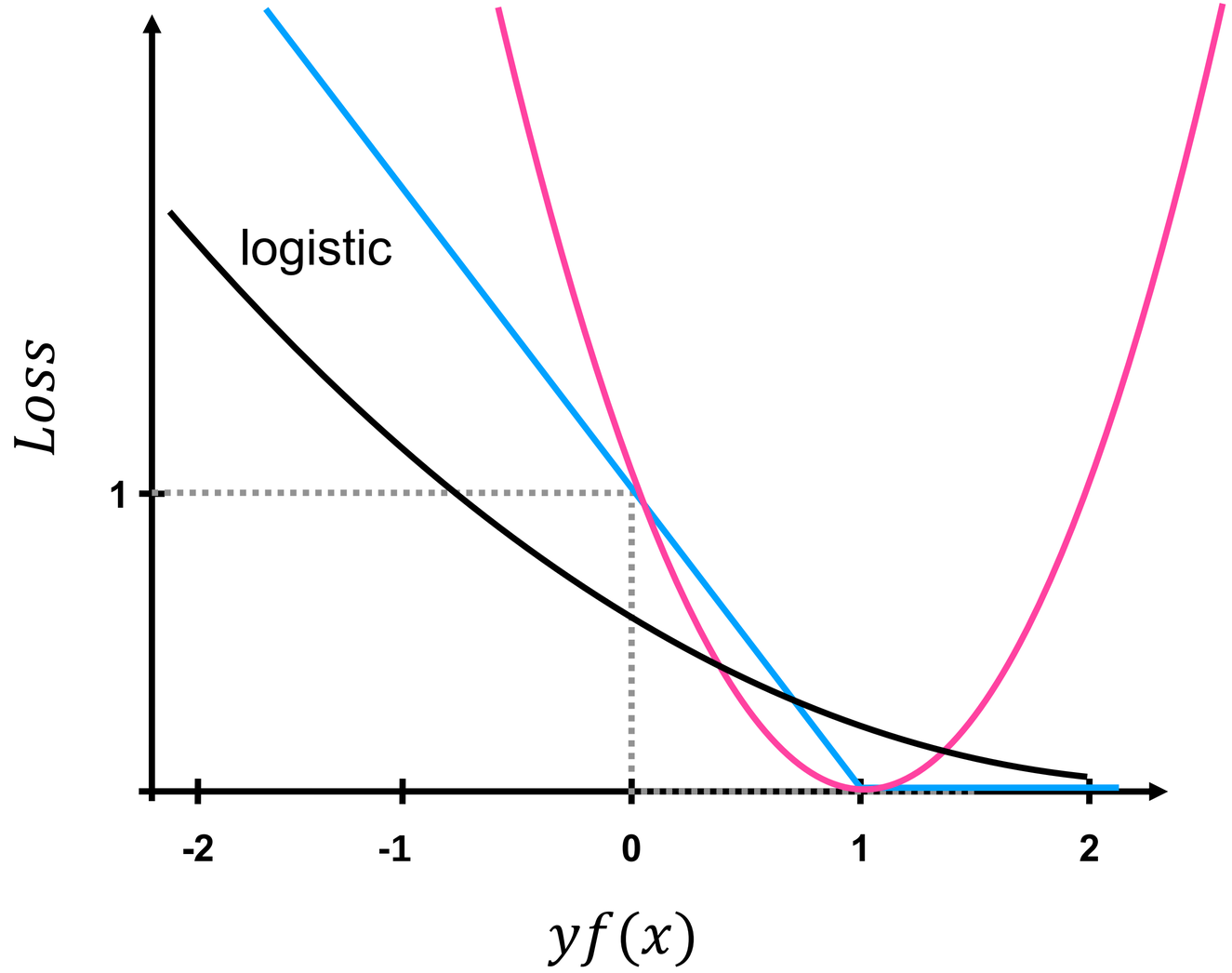
$$= -\frac{1}{N}\sum_{i=1}^{N} Y_i \log(\hat{Y_i})$$

The Hinge Loss

$$Hinge(x, y) = max\big(0, 1 - y.f(x)\big), labels\ y \in \{-1, 1\}$$

« Maximum-margin » classifier

SVM
Hinge Loss



MSE

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

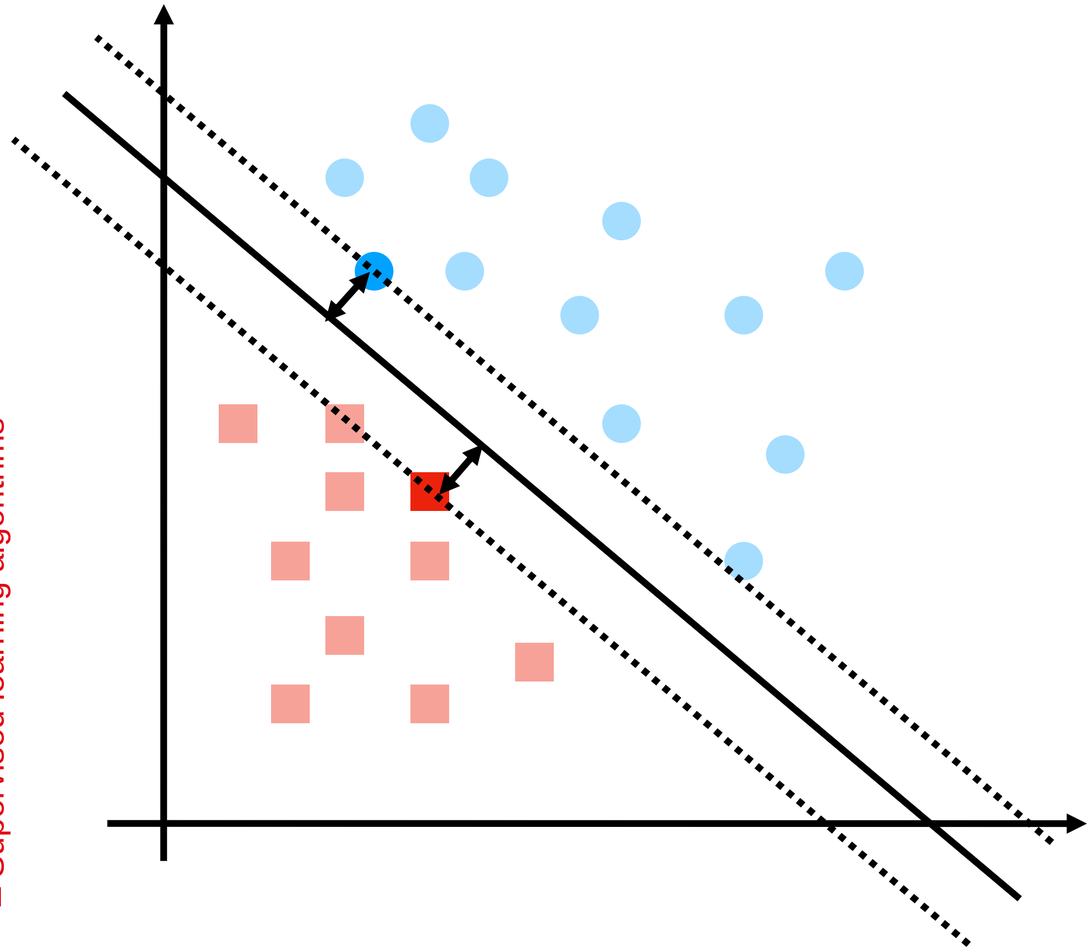
Left side of the Logistic loss

$$= -\frac{1}{N}\sum_{i=1}^{N} Y_i \log(\hat{Y}_i)$$

SVM

Optimal hyperplane separation

What's SVM?



Classification Algorithm

Based on maximizing **margin**

Margin : **Defined as the width that the boundary could increase by before hitting a datapoint**

O. Fink /A. Alahi

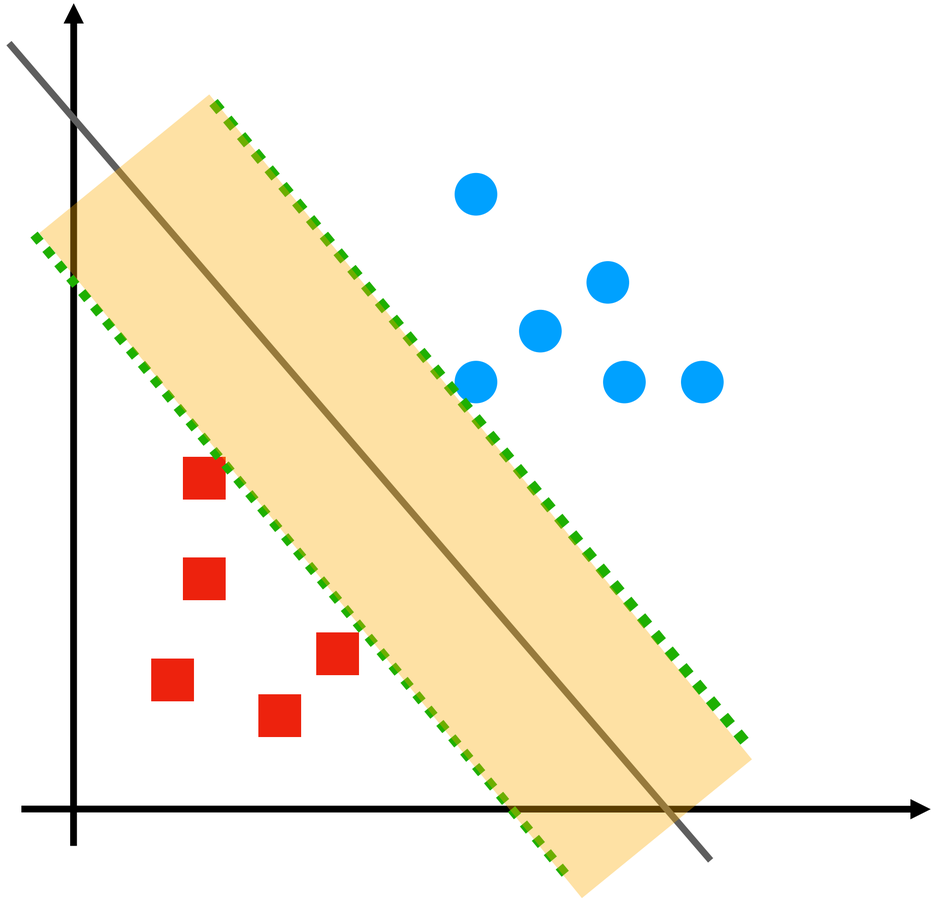
SVM

Optimal hyperplane separation

How can we formulate this problem?

Negative example : -1

Positive example : 1



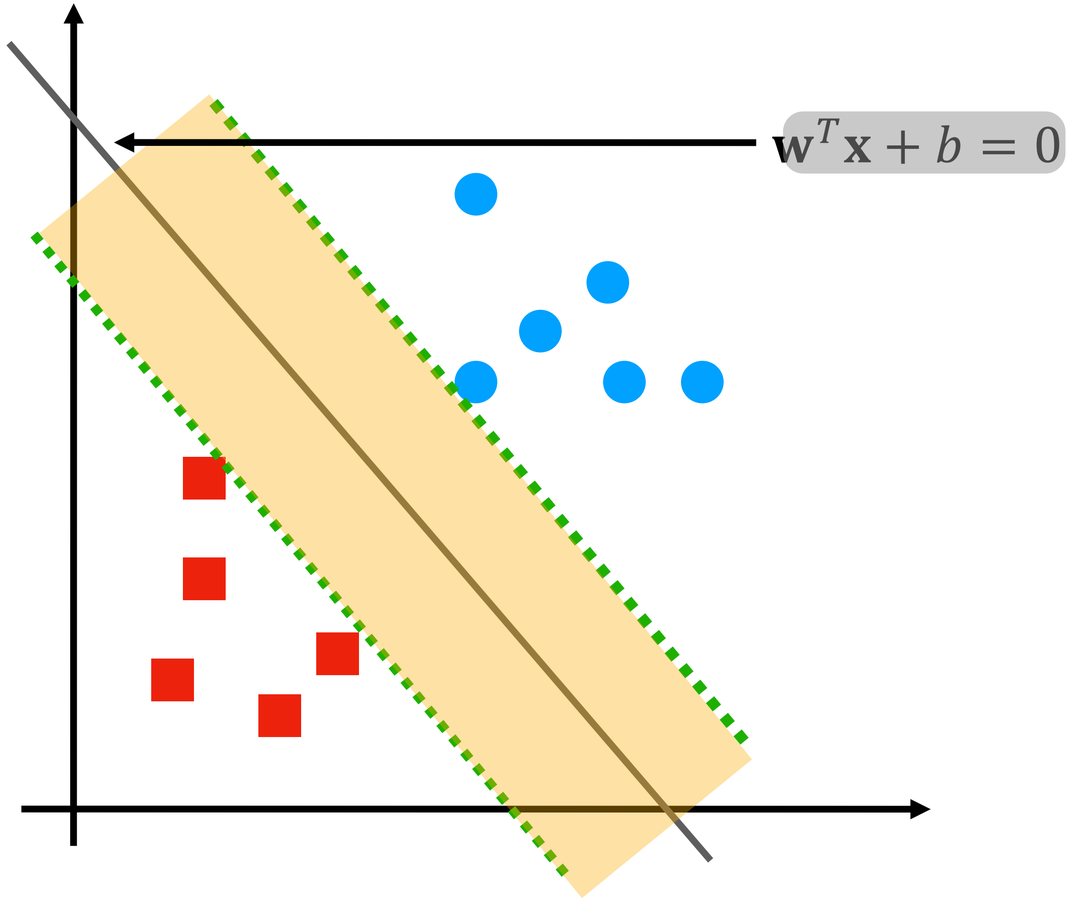
SVM

Optimal hyperplane separation

First we model our separation hyperplane

Negative example : -1

Positive example : 1

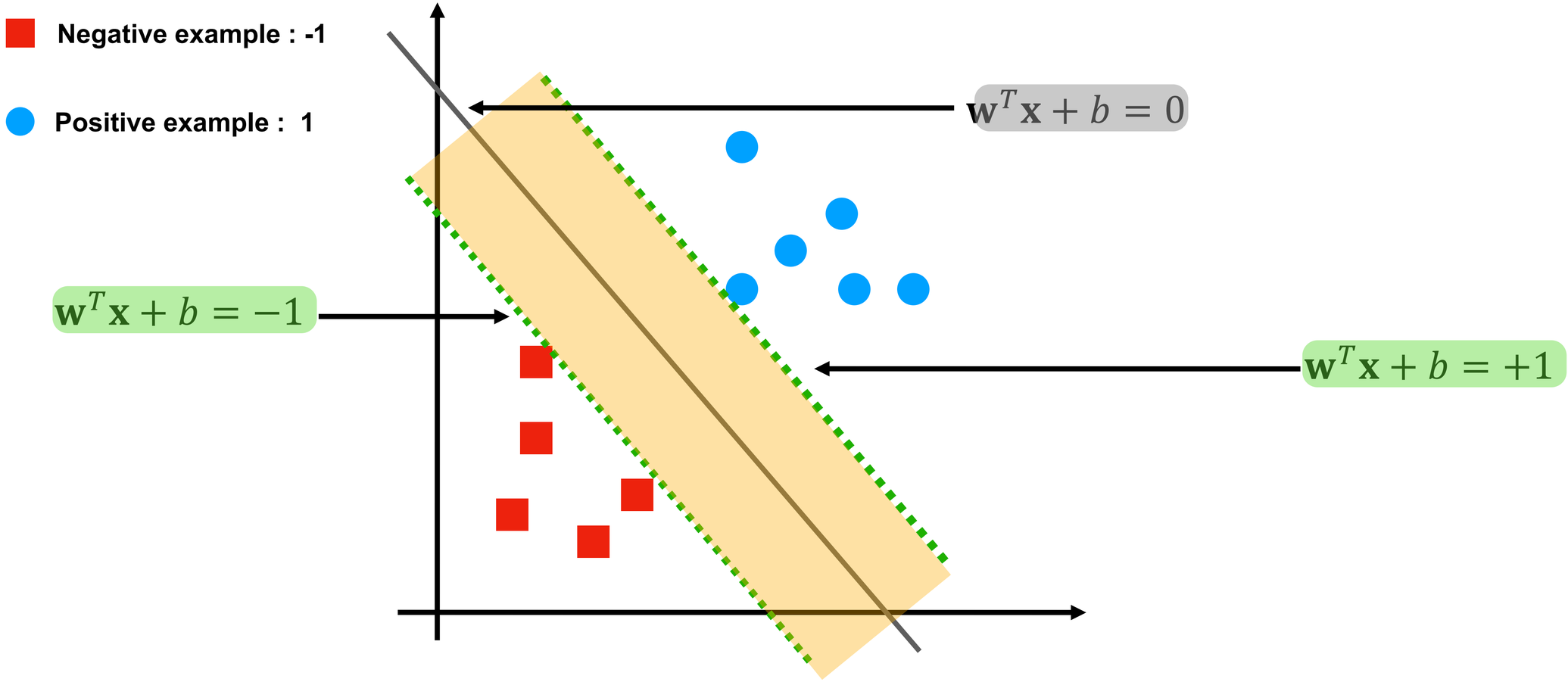


O. Fink /A. Alahi

SVM

Optimal hyperplane separation

Then we model our margin limits

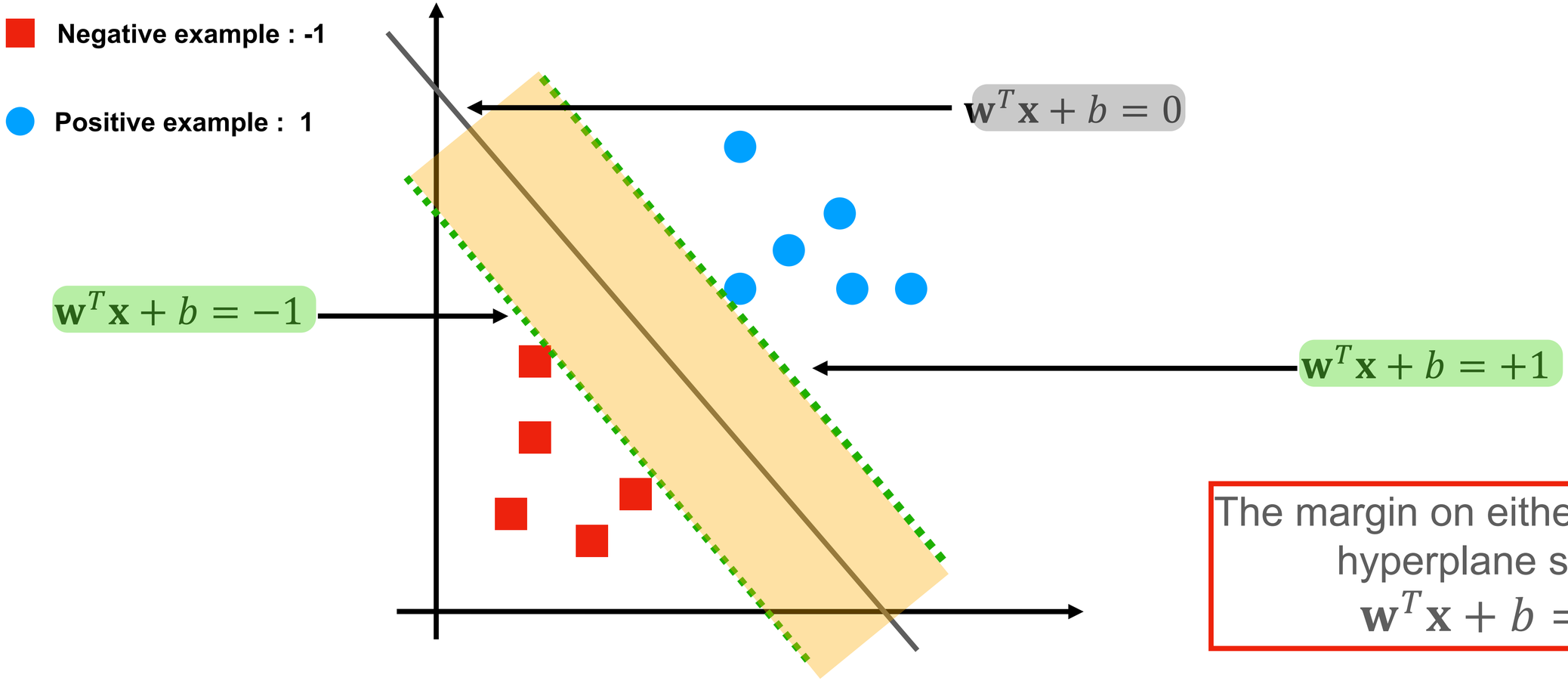


O. Fink /A. Alahi

SVM

Optimal hyperplane separation

We obtain our constraints



The margin on either side of the hyperplane satisfy

$$\mathbf{w}^T\mathbf{x} + b = \pm 1$$

SVM

Optimal hyperplane separation

The margin between two classes is at least $\frac{2}{\|\mathbf{w}\|}$

We obtain our objective criteria

Maximizing this condition is equivalent to **minimizing** $\frac{\|\mathbf{w}\|}{2}$

Better to minimize a **square form** : $\frac{\|\mathbf{w}\|^2}{2}$

SVM

Optimal hyperplane separation

We obtain an optimization problem:

$$\min_{\mathbf{w},b} \frac{\| \mathbf{w} \|^2}{2}$$

Objective

$$\left.\begin{array}{l} \mathbf{w}^T\mathbf{x}^{(i)} + b \geq 1 \text{ when } \; y^{(i)} = +1 \\ \mathbf{w}^T\mathbf{x}^{(i)} + b \leq -1 \text{ when } \; y^{(i)} = -1 \end{array}\right\} y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \text{ when } \; i = 1,2,\ldots,M$$

Constraints

This is hard-margin SVM, and it works only for separable data

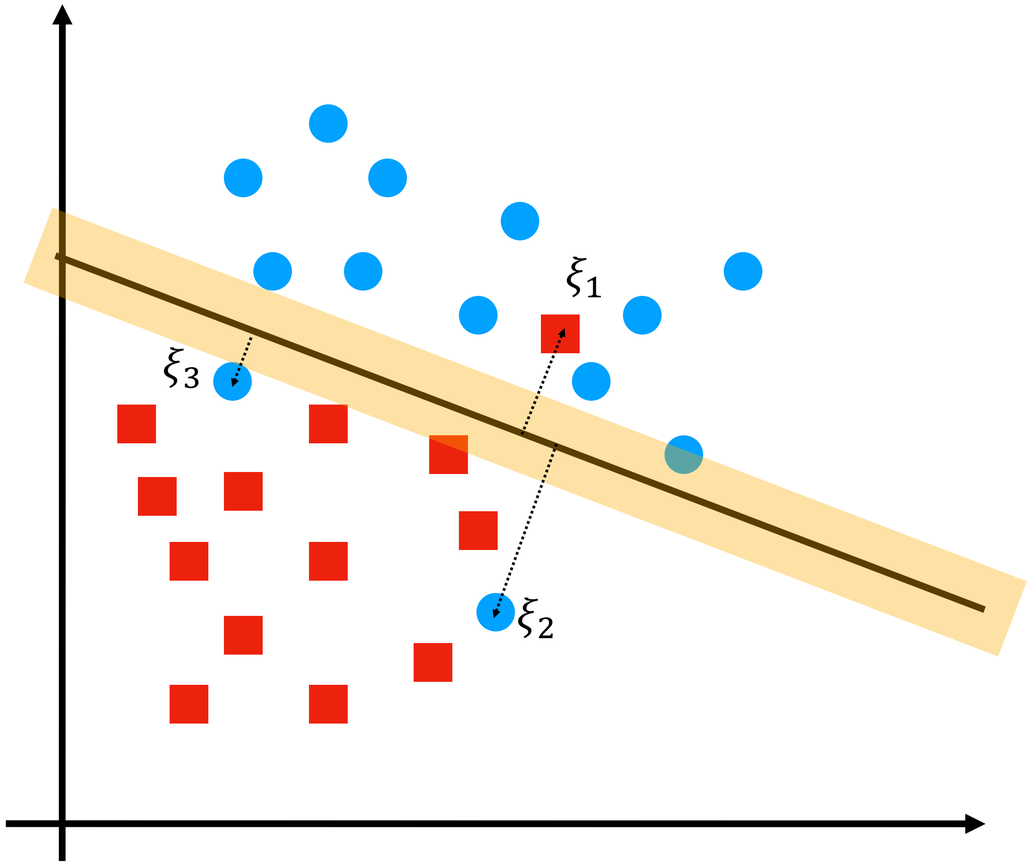
Next slides are optional and develop the formulation of hard-margin SVM

O. Fink /A. Alahi

SVM

Dealing with non-separable data

What should we do ?



Constraints relaxed by slack variables ξ_i

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i s.t \xi_i \geq 0 \;\; \forall\, i = 1,2,\ldots,N$$

M : number of examples in the margin or misclassified

We need to add a penalty for too large slack variable

$$\min_{\mathbf{w},b}\left(\frac{\| \mathbf{w} \|^2}{2} + \frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

C > 0 weight the influence of the penalty term

Find trade off between maximizing margin and minimizing the classification errors

SVM

Dealing with non-separable data

Control size of the margin

$$\min_{\mathbf{w},b}\left(\frac{\|\mathbf{w}\|^2}{2}+\frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

Control number of misclassification

$$\left.\begin{array}{l}\mathbf{w}^T\mathbf{x}^{(i)}+b\geq 1-\xi_i \text{ when } y^{(i)}=+1\\ \mathbf{w}^T\mathbf{x}^{(i)}+b\leq -1+\xi_i \text{ when } y^{(i)}=-1\end{array}\right\} y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)\geq 1-\xi_i \text{ when } i=1,2,\ldots,M$$

Note that : $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)\geq 1-\xi_i \Leftrightarrow \xi_i\geq 1-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)$

Because $\xi_i\geq 0$ **Note that :** $\xi_i=max(0,1-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b))$

SVM

Dealing with non-separable data

Control size of the margin

$$\min_{\mathbf{w},b}\left(\frac{\|\mathbf{w}\|^2}{2}+\frac{C}{N}\sum_{i=1}^{N}\xi_i\right)$$

Control number of misclassification

We can reformulate into a form more adapted for learning as :

Hinge loss

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N}max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w})+\frac{\lambda}{2}\|\mathbf{w}\|^2\right)$$

Regularization term

SVM

Dealing with non-separable data

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N} max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w}\right)$$

Leads to :

1. **A separating hyperplane**
2. **A scaling of w so that no point of the data is in the margin**

$$\min_{\mathbf{w}}\left(\frac{\lambda}{2}\parallel \mathbf{w} \parallel^2\right)$$

Guarantee that separating hyperplane and scaling for which the margin is the largest

SVM

Dealing with non-separable data

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N} max(0,1-y^{(n)}\mathbf{x}^{(n)^T}\mathbf{w}) + \frac{\lambda}{2}\parallel \mathbf{w} \parallel^2\right)$$

Note that this function is convex

So we can use SGD to optimize it efficiently

SVM

Dealing with non-separable data

To lead to a more efficient implementation
we use the dual form of our problem

$$\min_{\mathbf{w}}\left(\sum_{n=1}^{N}[1-y^{(n)}\mathbf{x}^{(n)T}\mathbf{w}]_{+}+\frac{\lambda}{2}\parallel\mathbf{w}\parallel^{2}\right)\longleftrightarrow\max_{\alpha\in\{0,1\}}\min_{\mathbf{w}}\left(\sum_{n=1}^{N}\alpha_{n}(1-y^{(n)}\mathbf{x}^{(n)T}\mathbf{w})+\frac{\lambda}{2}\parallel\mathbf{w}\parallel^{2}\right)$$

Primal problem **Dual problem**

Two versions of the same problem

More explanation on duality for SVM

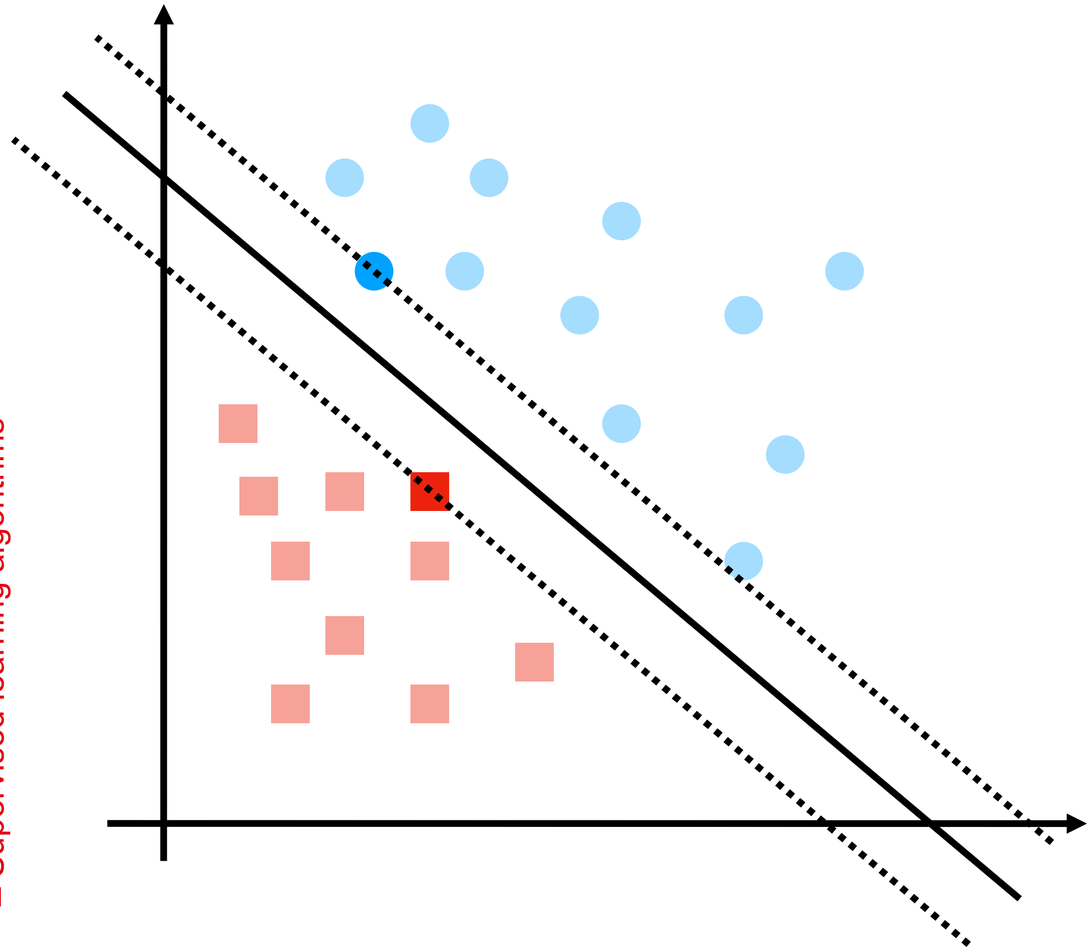
Note that : $max(0,f(x))=[f(x)]_{+}$

SVM

Dealing with non-separable data

$$\max_{\alpha \in \{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n (1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}) + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

The α_n are the support vectors !



The support vectors are the **nearest samples** from the hyperplane

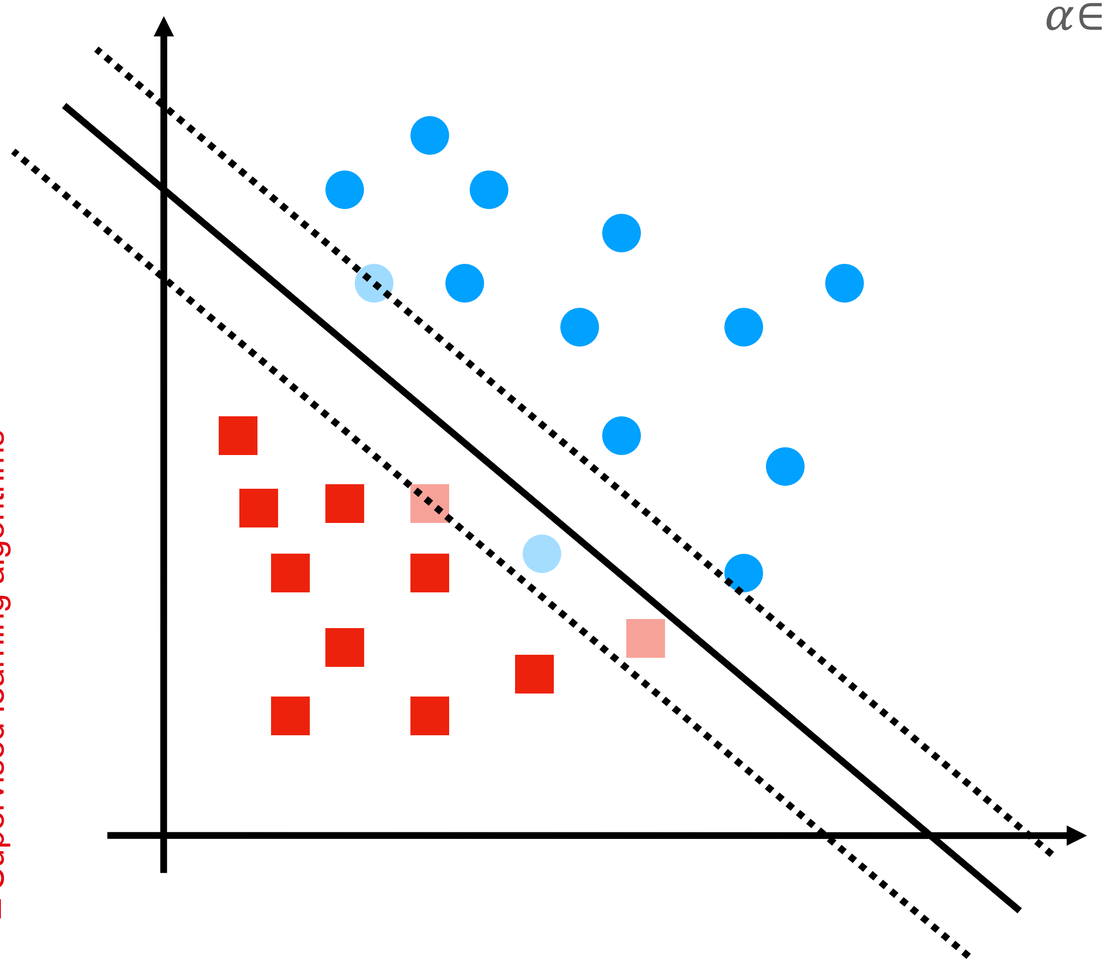
SVM

Dealing with non-separable data

$$\max_{\alpha \in \{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n (1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}) + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

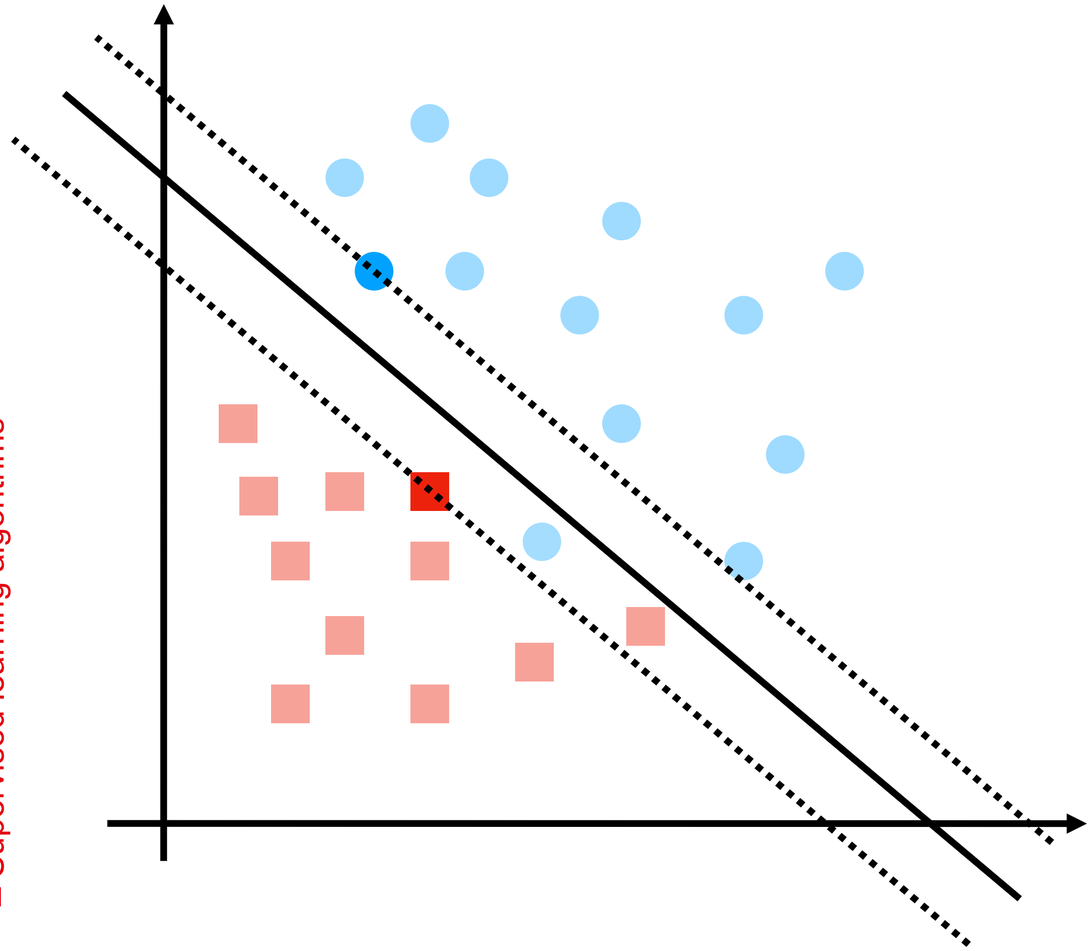
α_n = 0 when $1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w} < 0$

Examples that lie on the correct side and outside of the margin



SVM

Dealing with non-separable data



$$\max_{\alpha\in\{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n [1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}]_+ + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

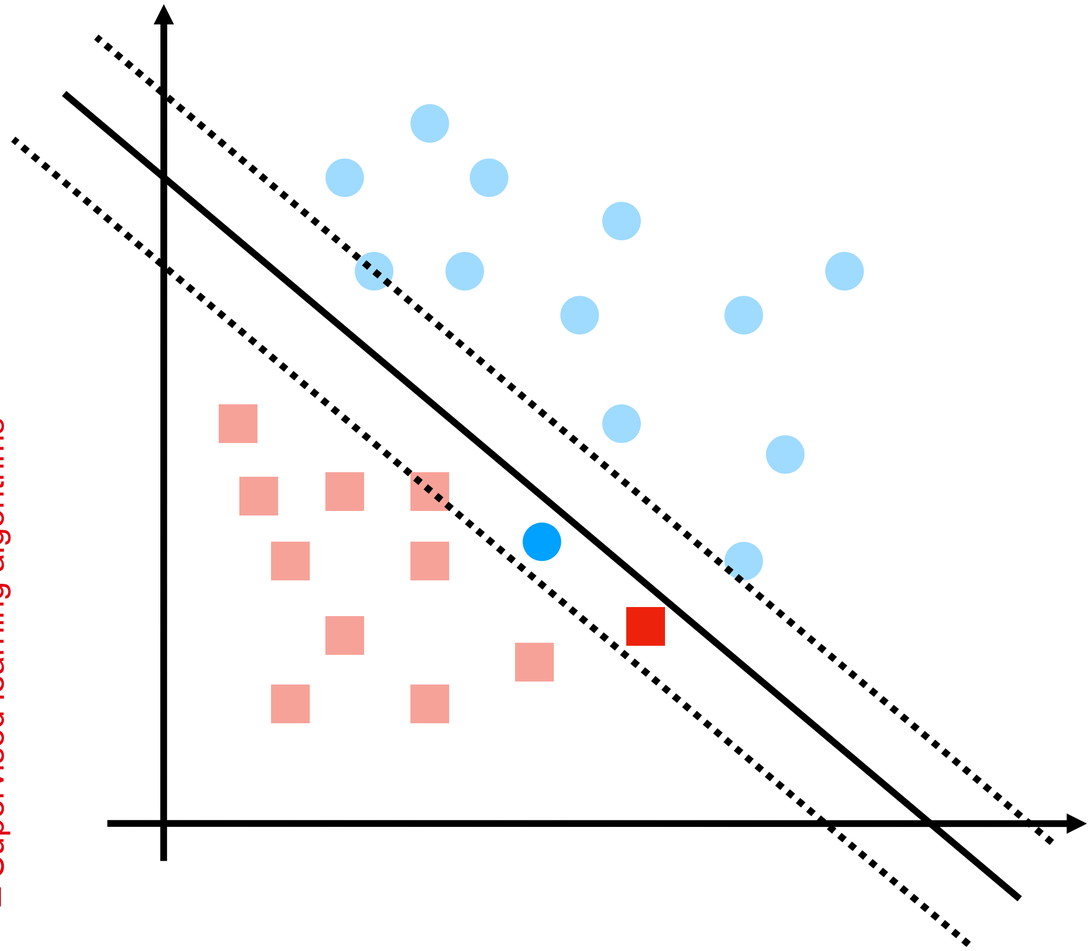
$$\alpha_n \in \{0,1\} \text{ when } 1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w} = 0$$

Examples that lie on the correct side and just on of the margin

SVM

Dealing with non-separable data

$$\max_{\alpha\in\{0,1\}} \min_{\mathbf{w}} \left(\sum_{n=1}^{N} \alpha_n [1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w}]_+ + \frac{\lambda}{2} \| \mathbf{w} \|^2 \right)$$

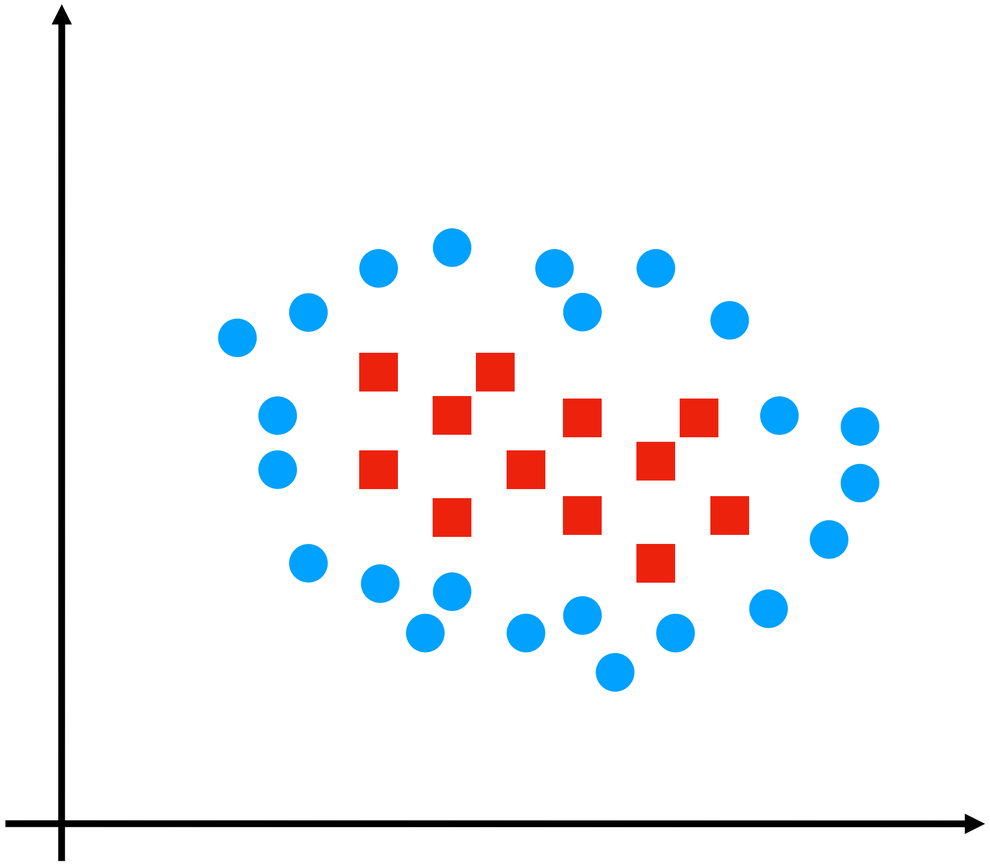


$\alpha_n = 1$ when $1 - y^{(n)} \mathbf{x}^{(n)T} \mathbf{w} > 0$

Examples that lie strictly inside the margin, or on the wrong side

SVM

Dealing with non-linear classification



We introduce the kernel trick

SVM

Dealing with non-linear classification

$$\max_{\alpha \in [0,1]^N} \alpha^T \mathbf{1} - \frac{1}{2\lambda} \alpha^T \mathbf{Y}\mathbf{X}\mathbf{X}^T \mathbf{Y}\alpha$$

We saw that in this alternative formulation the data only enters in the form of a ‘’kernel‘’ $\mathbf{K} = \mathbf{X}\mathbf{X}^T$

$X: [N \times D]$ with N the number of samples and D the number of features

By definition, a ‘’kernel’’ $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ is equivalent to an **inner product**

There exists an **infinity of inner products** dependent to the **geometry of spaces** where we use it

SVM

Dealing with non-linear classification

$$\max_{\alpha \in [0,1]^N} \alpha^T \mathbf{1} - \frac{1}{2\lambda} \alpha^T \mathbf{Y} \mathbf{X}\mathbf{X}^T \mathbf{Y} \alpha$$

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \dots & \mathbf{x}_1^T\mathbf{x}_N \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \dots & \mathbf{x}_2^T\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ \mathbf{x}_N^T\mathbf{x}_1 & \mathbf{x}_N^T\mathbf{x}_2 & \dots & \mathbf{x}_N^T\mathbf{x}_N \end{bmatrix}$$

We can introduce a new inner product :

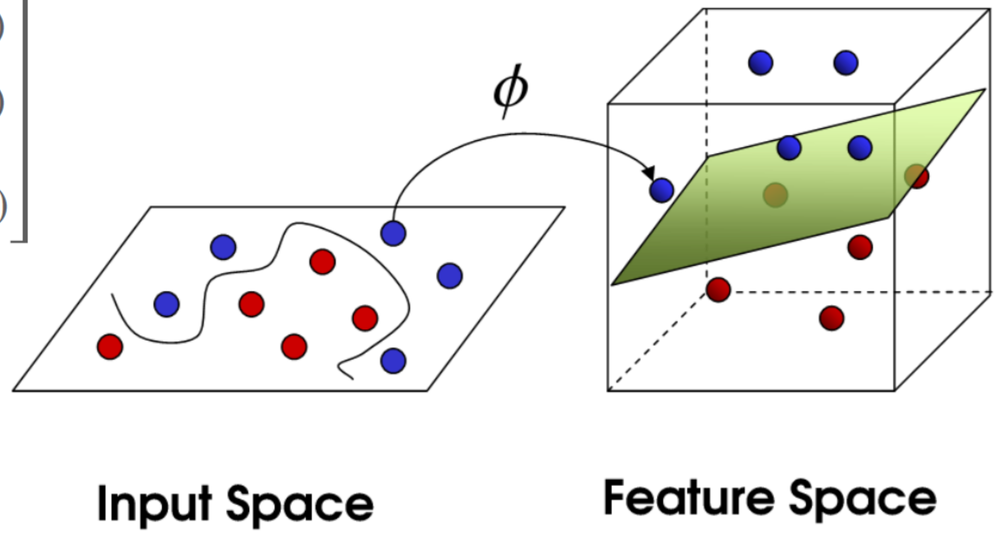
$$\mathbf{K}' = \mathbf{\Phi}\mathbf{\Phi}^T = \begin{bmatrix} \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_1) & \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_N) \\ \phi(\mathbf{x}_2)^T\phi(\mathbf{x}_1) & \phi(\mathbf{x}_2)^T\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_2)^T\phi(\mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ \phi(\mathbf{x}_N)^T\phi(\mathbf{x}_1) & \phi(\mathbf{x}_N)^T\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_N)^T\phi(\mathbf{x}_N) \end{bmatrix}$$

SVM

Dealing with non-linear classification

We can introduce a new inner product :

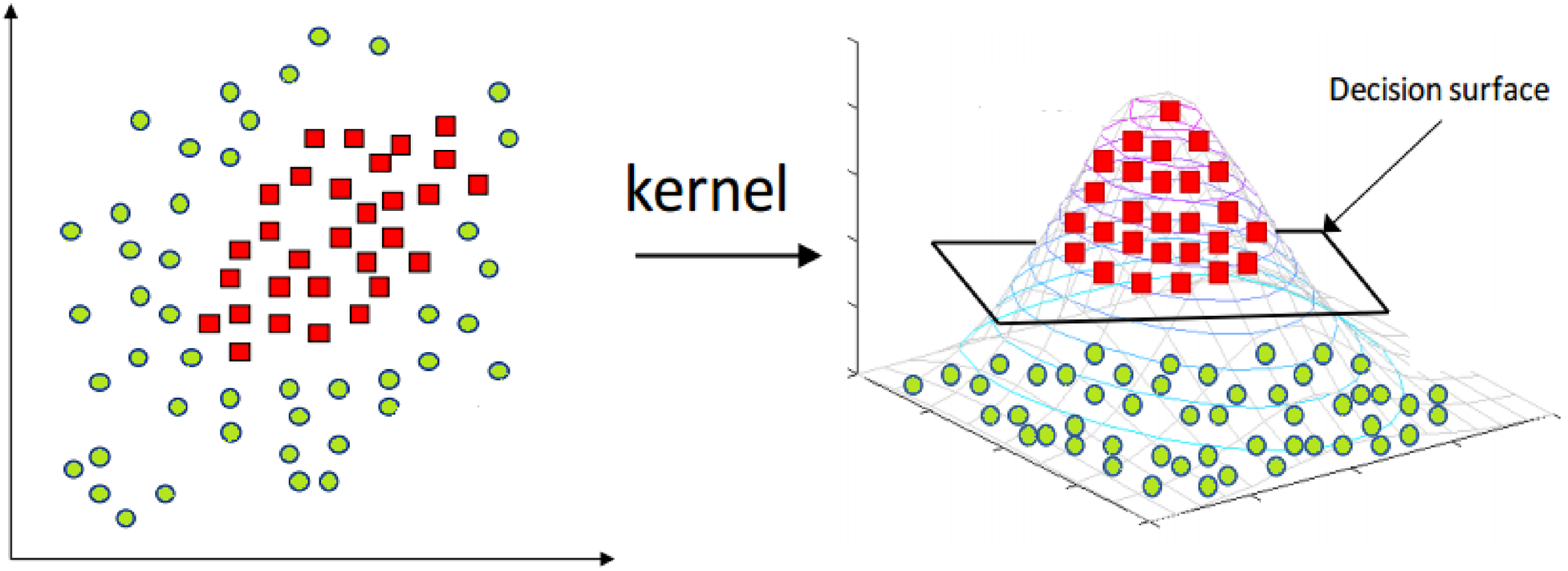
$$\mathbf{K}' = \mathbf{\Phi}\mathbf{\Phi}^T = \begin{bmatrix} \phi(\mathbf{x}_1)^{\mathbf{T}}\phi(\mathbf{x}_1) & \phi(\mathbf{x}_1)^{\mathbf{T}}\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_1)^{\mathbf{T}}\phi(\mathbf{x}_N) \\ \phi(\mathbf{x}_2)^{\mathbf{T}}\phi(\mathbf{x}_1) & \phi(\mathbf{x}_2)^{\mathbf{T}}\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_2)^{\mathbf{T}}\phi(\mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ \phi(\mathbf{x}_N)^{\mathbf{T}}\phi(\mathbf{x}_1) & \phi(\mathbf{x}_N)^{\mathbf{T}}\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_N)^{\mathbf{T}}\phi(\mathbf{x}_N) \end{bmatrix}$$



This new inner product change the geometric aspect of our original problem and can simplify the classification !

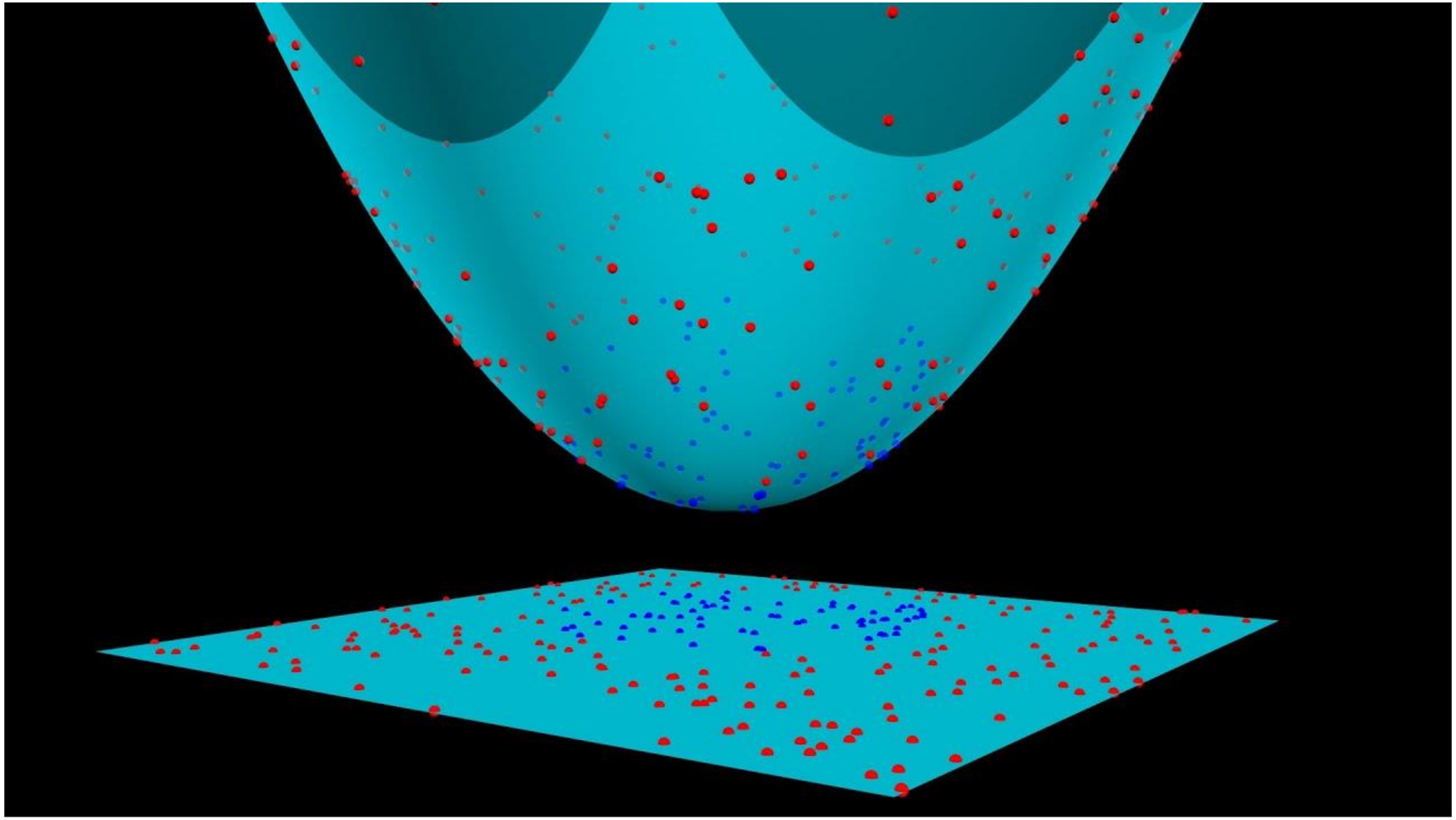
SVM

Dealing with non-linear classification



SVM

Dealing with non-linear classification



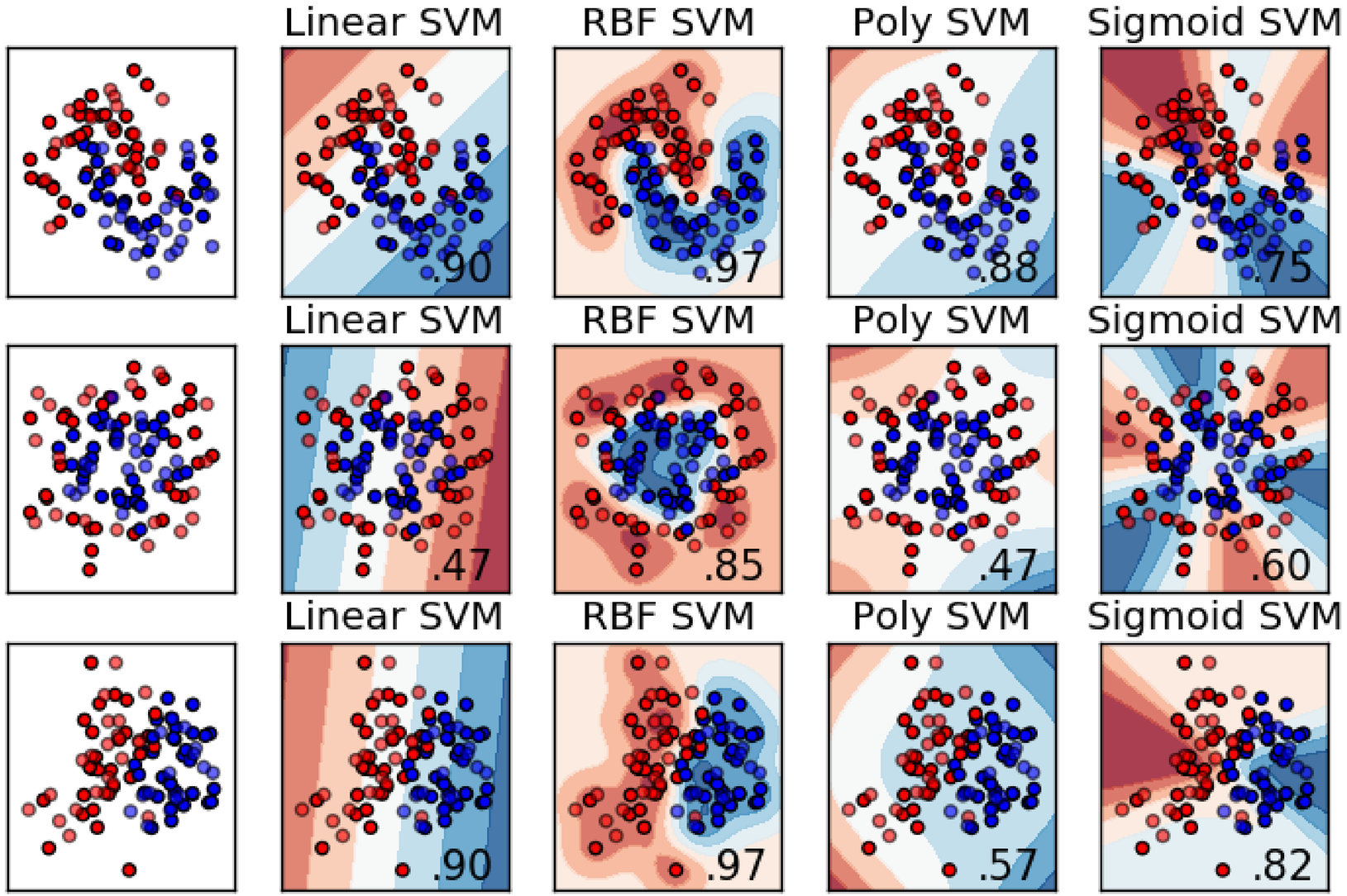
SVM

Dealing with non-linear classification

Kernel	Equation
Linear	$K(x,y) = xy$
Sigmoid	$K(x,y) = \tanh(axy + b)$
Polynomial	$K(x,y) = (1 + xy)^d$
RBF	$K(x,y) = a[\exp(\frac{\gamma}{\|\|x-y\|\|^2 + sigma^2}) - 1]$
KMOD	$K(x,y) = \exp(-a\|\|x-y\|\|^2)$
Exponential RBF	$K(x,y) = \exp(-a\|\|x-y\|\|)$

SVM

Dealing with non-linear classification



Live demos:

Dash-gallery.plotly.host/dash-svm/

SVM
Resources

- Book chap. 12:
  - https://mml-book.github.io/
- Explanation video:
  - https://www.youtube.com/watch?v=efR1C6CvhmE
- Tutorial on SVM:
  - http://lasa.epfl.ch/teaching/lectures/ML_Phd/Notes/nu-SVM-SVR.pdf
- Other slides on SVM:
  - http://lasa.epfl.ch/teaching/lectures/ML_Phd/Slides/SVM.pdf
- Visualization:
  - https://www.cristiandima.com/basics-of-support-vector-machines/

Naive Bayes

Example Structural Health Monitoring

- Let's consider a simplified case study where the goal is to develop a classifier to determine the health status of a bridge based on two primary indicators: vibration frequency deviations (measured in Hz) and maximum daily temperature variations (measured in degrees Celsius). For simplicity, we classify the bridge's condition into three categories: Good, Minor Damage, and Major Damage.

Example Structural Health Monitoring

- Data Assumption
- Assume we have historical data that indicate the following probabilities:
- **Prior Probabilities of Each Condition:**
  - *P*(Good)=0.70
  - *P*(Minor Damage)=0.20
  - *P*(Major Damage)=0.10
- **Likelihood of Vibration Frequency Deviations (Hz):**
  - Good Condition: Normally around 0.5 Hz deviation.
  - Minor Damage: Deviations around 2 Hz.
  - Major Damage: Deviations exceed 4 Hz.
- **Likelihood of Maximum Daily Temperature Variation (°C):**
  - Good Condition: Variation within ±5°C.
  - Minor Damage: Variation within ±10°C.
  - Major Damage: Variation exceeds ±15°C.

O. Fink /A. Alahi

Example Structural Health Monitoring

- For simplicity, let's assume we categorize the deviations into "low," "medium," and "high" for both features and assign probabilities based on our historical data.
- Scenario:
- One day, sensors on the bridge report a vibration frequency deviation of 3 Hz and a maximum daily temperature variation of 12 °C. We need to classify the bridge's condition based on this data.

Naive Bayes

Bayes' theorem

X : Features

y : Labels/Target

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Naive Bayes

Bayes' theorem

Probability to label with y knowing X

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Naive Bayes

Bayes' theorem

Probability to have this X knowing the label y

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$P(X|y)$ is called a **generative model**

Describes the way to produce data with the same Features distribution

Naive Bayes

Bayes' theorem

Probability to label with y

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$P(y)$ is called the **prior probability**

Describes the probability to encounter the data labeled y without considering the X

Naive Bayes

Bayes' theorem

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Probability of the X

Naive Bayes

Bayes' theorem

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

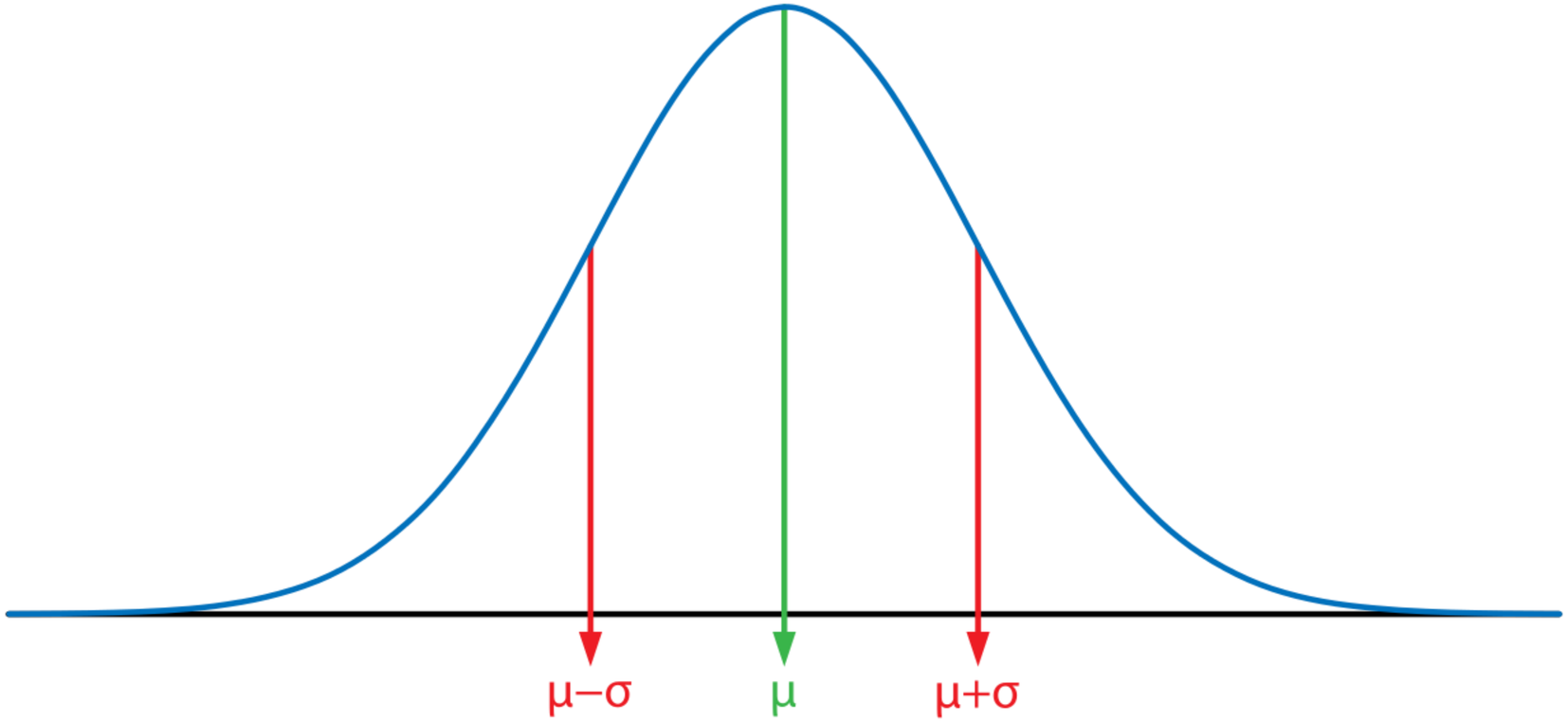
This theorem gives us the probability to be labeled with y knowing features observation X

We can use this probability for classification

Naive Bayes

Gaussian Naive Bayes

First we can make the assumption that the **generative model** follows a **simple gaussian distribution**



Naive Bayes

Bayes' theorem

The hard part is finding the **generative model**

We can make some simplifying assumptions about the form of this model

First assumption: the **generative model** follows a **simple gaussian distribution**

This is why these methods are called **Naive**

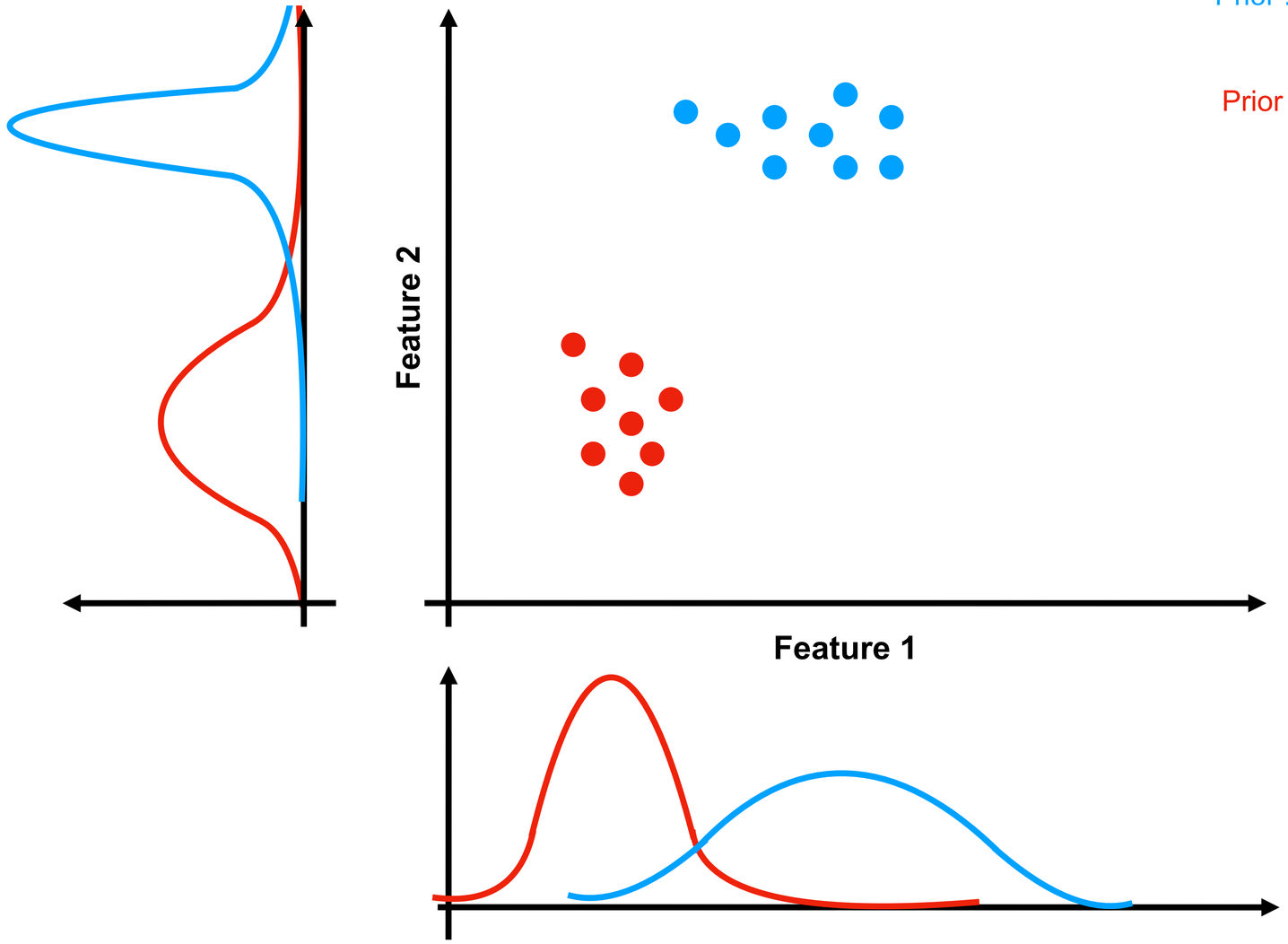
Second assumption: **Features are independent** !

Naive Bayes

Gaussian Naive Bayes

Prior : P(BLUE) = $\frac{9}{9+8} = 0.53$

Prior : P(RED) = $\frac{8}{9+8} = 0.47$

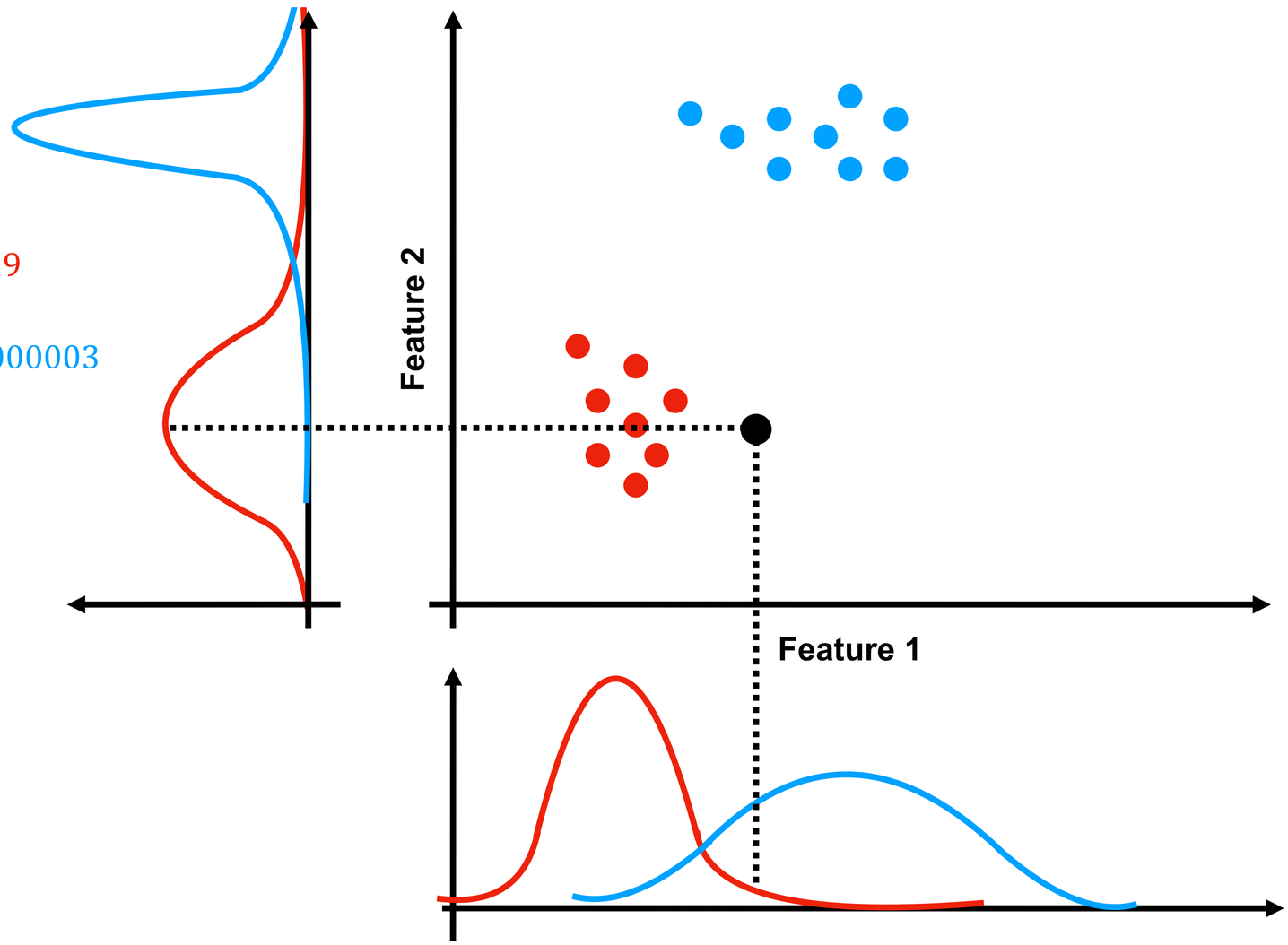


Naive Bayes

Gaussian Naive Bayes

$P(Features2|RED) = 0.19$

$P(Features2|BLUE) = 0.000003$

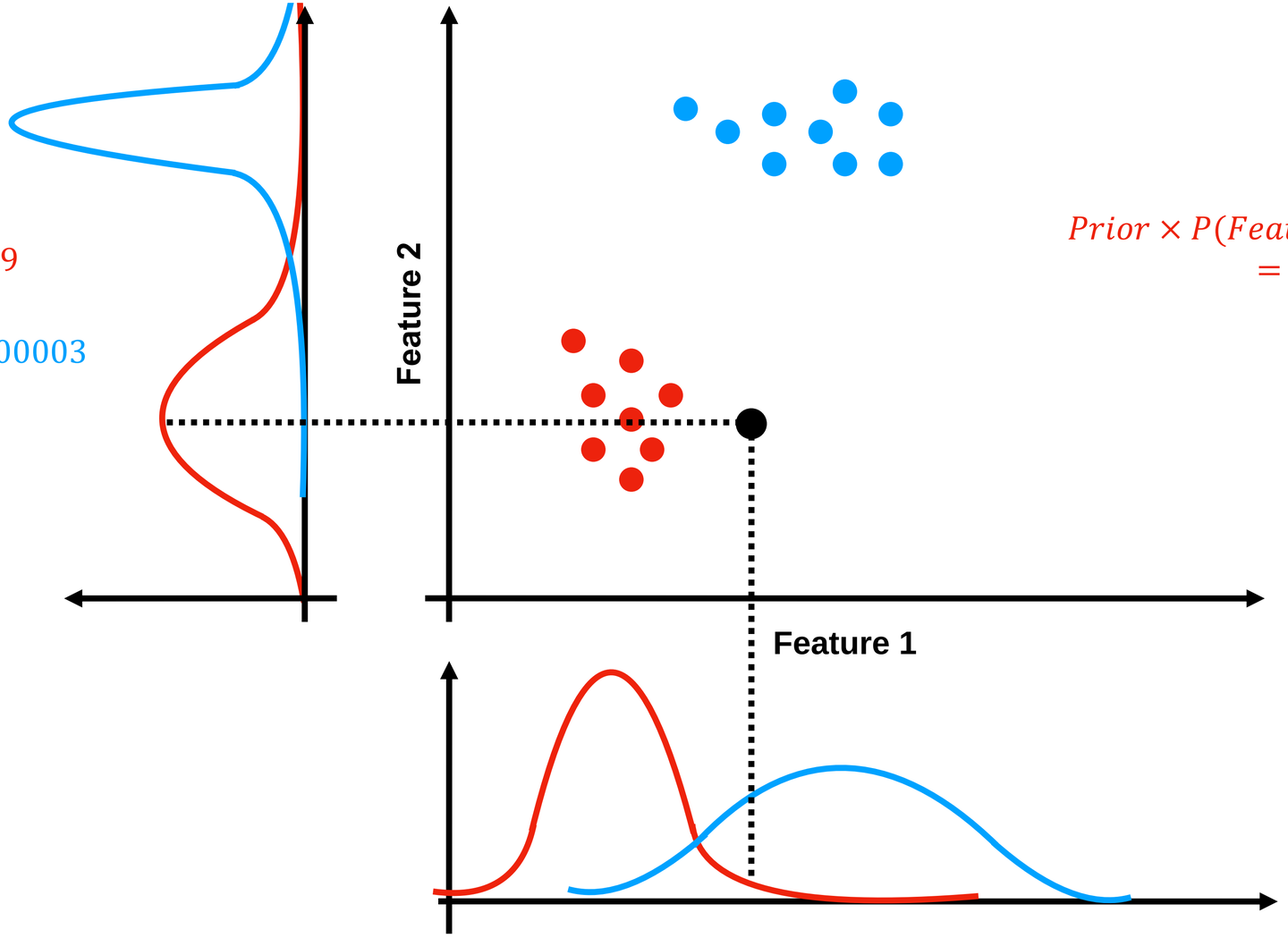


$P(Features1|RED) = 0.004$

$P(Features1|BLUE) = 0.08$

Naive Bayes

Gaussian Naive Bayes



$P(Features2|RED) = 0.19$

$P(Features2|RED) = 0.000003$

$Prior \times P(Features1|RED) \times P(Features2|RED)$
$= 0.47 \times 0.004 \times 0.19$
$= 0.0003572$

$P(Features1|RED) = 0.004$

$P(Features1|BLUE) = 0.08$

Naive Bayes

Gaussian Naive Bayes

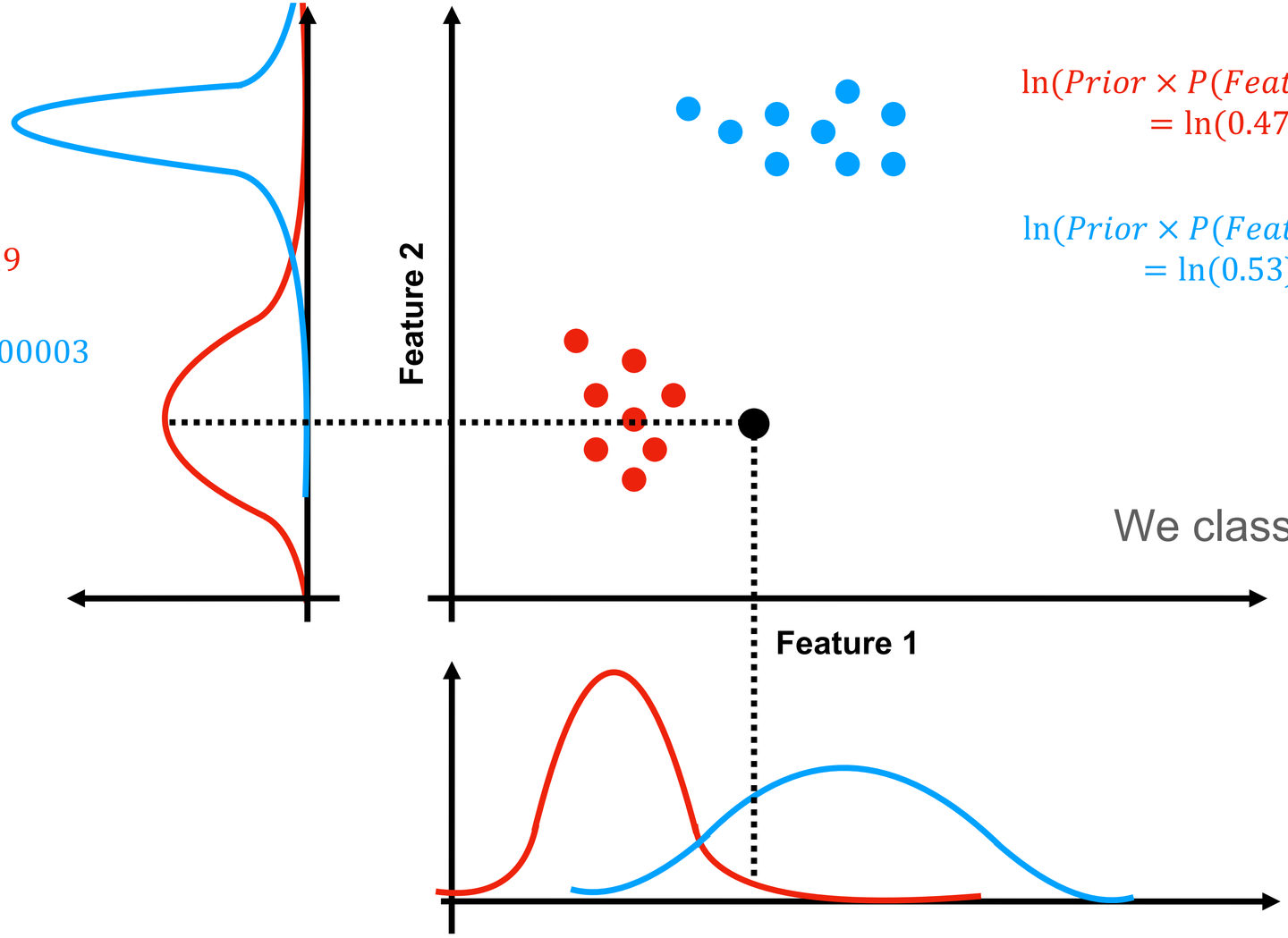
Add $\ln$ to avoid too small values (source of computational error) !!

$$\ln(Prior \times P(Features1|RED) \times P(Features2|RED))$$
$$= \ln(0.47) + \ln(0.004) + \ln(0.19)$$
$$= -7.94$$

$$\ln(Prior \times P(Features1|BLUE) \times P(Features2|BLUE))$$
$$= \ln(0.53) + \ln(0.08) + \ln(0.000003)$$
$$= -15.9$$

$P(Features2|RED) = 0.19$

$P(Features2|RED) = 0.000003$



$-15.9 < -7.94$

We classify this new point as RED

$P(Features1|RED) = 0.004$

$P(Features1|BLUE) = 0.08$

Example SHM cont.

- Let's simplify and assume:
- The probability of observing a 3 Hz deviation is:
  - P(3 Hz|Good)=0.1
  - P(3 Hz|Minor Damage)=0.7
  - P(3 Hz|Major Damage)=0.2
- The probability of observing a 12°C variation is:
  - P(12°C|Good)=0.05
  - P(12°C|Minor Damage)=0.6
  - P(12°C|Major Damage)=0.35
- Calculating Probabilities
- To classify the bridge's condition, we calculate the posterior probability for each condition using Bayes' theorem, focusing on the product of the likelihood and the prior probability for simplicity.
- Let's calculate these probabilities.
- Based on the calculated probabilities, the bridge's health condition is classified as follows:
  - Good: 3.7%
  - Minor Damage: 88.9%
  - Major Damage: 7.4%

Coding Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import make_classification

# Generate a sample dataset
X, y = make_classification(n_samples=100, n_features=2, n_informa-
tive=2, n_redundant=0)

# Create a Gaussian Naive Bayes classifier
clf = GaussianNB()

# Fit the classifier to the data
clf.fit(X, y)

# Predict the class of new data
pred = clf.predict([[0, 0], [1, 1]])

# Output the predicted class
print(pred)
```

Source: A primer to the 42 most commonly used ML algorithms

Naive Bayes
Resources

- Explanation with code
  - https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html
- Interactive explanation (To understand Bayes' theorem and the likelihood)
  - https://seeing-theory.brown.edu/bayesian-inference/index.html

Decision Trees

Example: earthquake vulnerability assessment

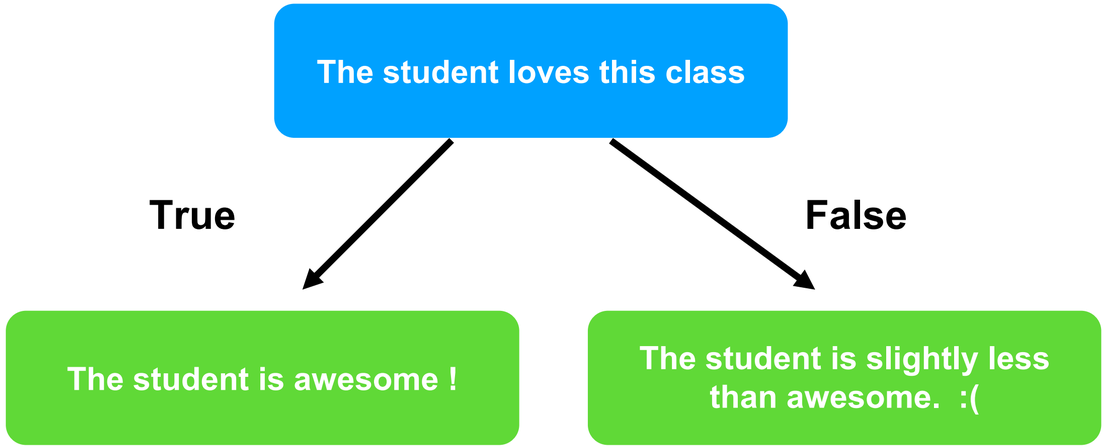
- The goal of an earthquake vulnerability assessment is to evaluate **how likely buildings or structures are to be damaged in the event of an earthquake.** This involves analyzing various factors such as the **building's age, construction material, height, design, the seismic zone where it's located, soil type, and previous maintenance records.** A Decision Tree model can be used to classify buildings into different categories of vulnerability based on these factors.

  - Why Use Decision Trees for This Task?
  - **Interpretability:** Decision Trees provide clear and interpretable models, which is crucial for safety-critical applications like structural engineering. Engineers and decision-makers can easily understand the basis of the model's classifications.
  - **Handling Different Types of Data:** Decision Trees can handle both numerical and categorical data, making them suitable for the diverse types of data involved in earthquake vulnerability assessment.
  - **Non-linear Relationships:** They can model non-linear relationships between the factors affecting a building's vulnerability to earthquakes, which is often the case in real-world scenarios.

Decision Trees

Introduction

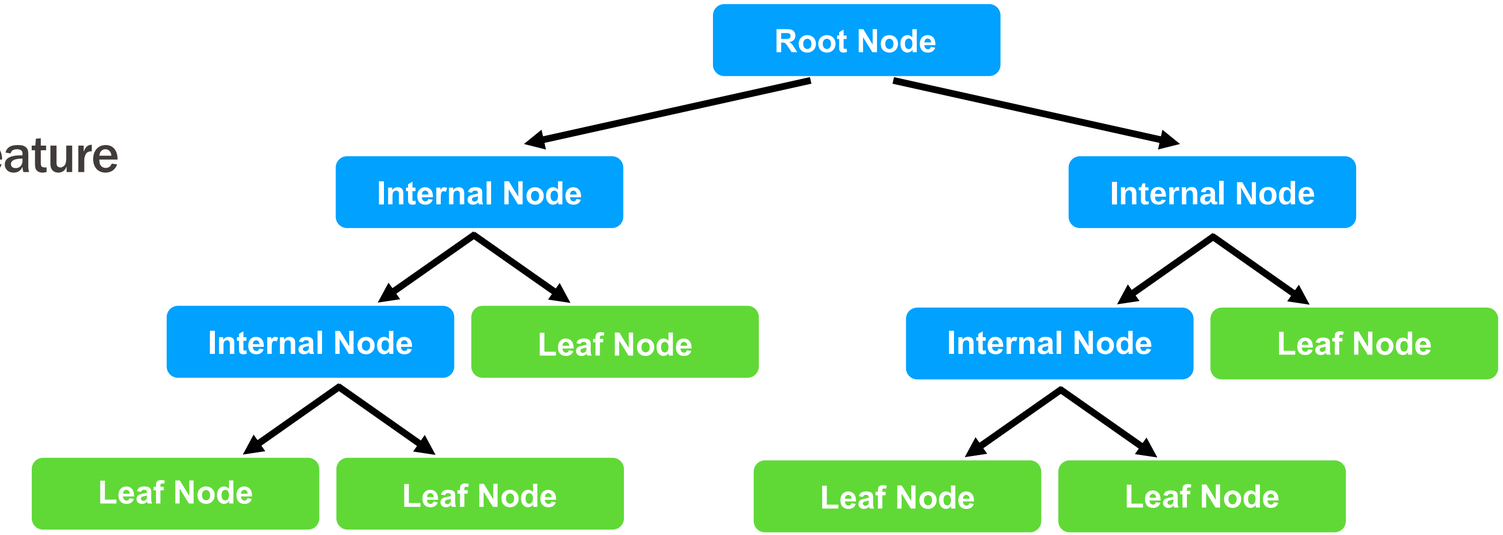
What is a decision tree?



Split data based on the answer to a question

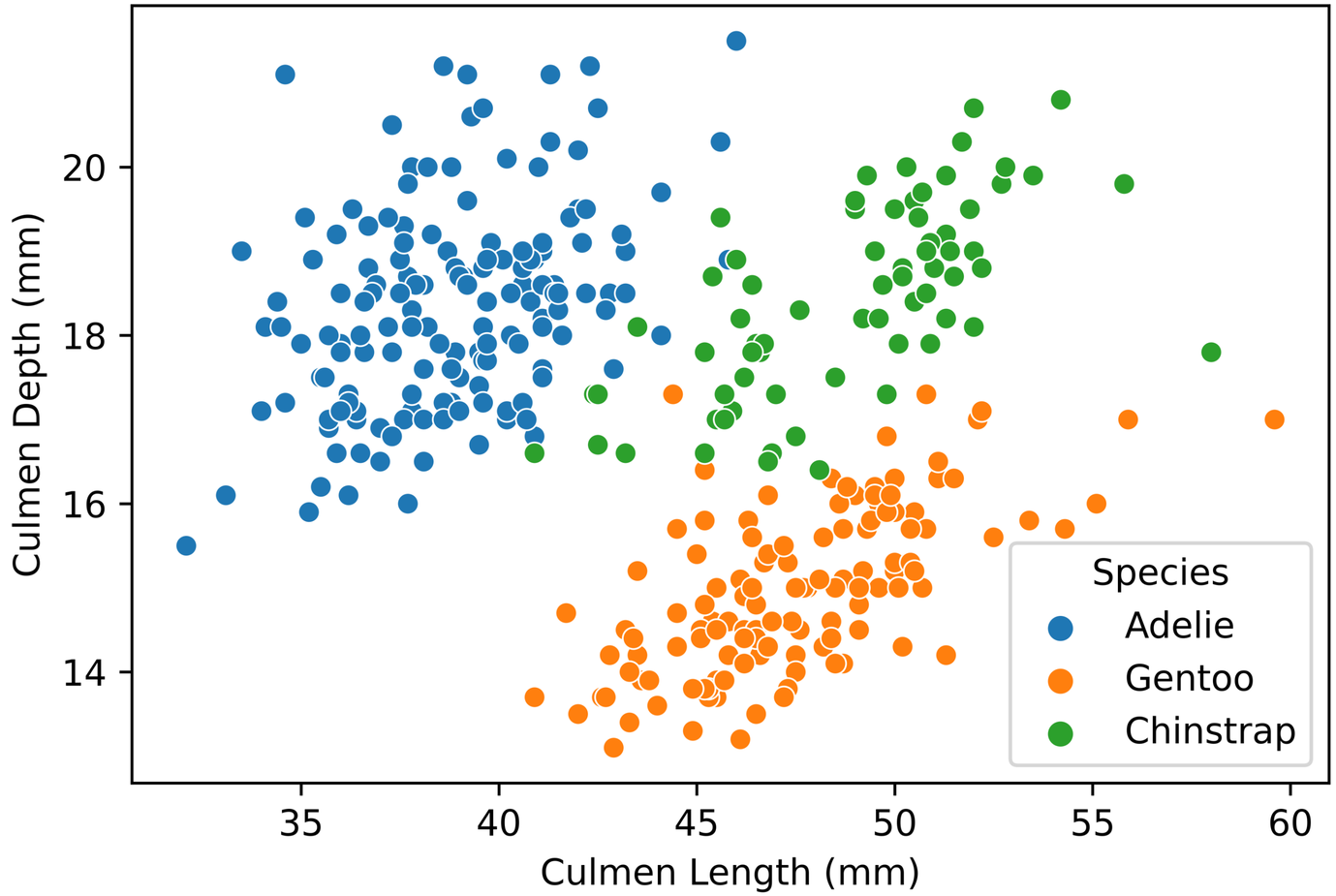
Decision Trees
Terminology

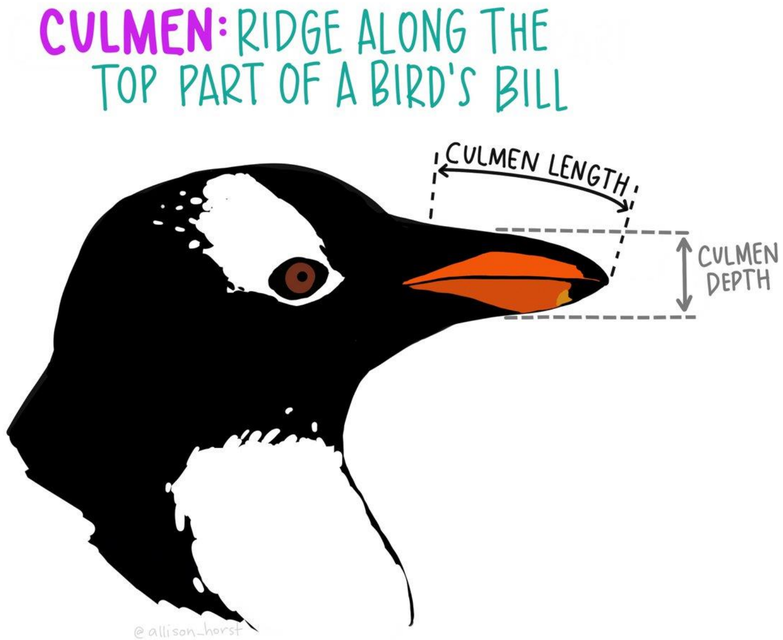
- Nodes are checked on a single feature
- Branches are feature values
- Leaves indicate class label



Decision Trees

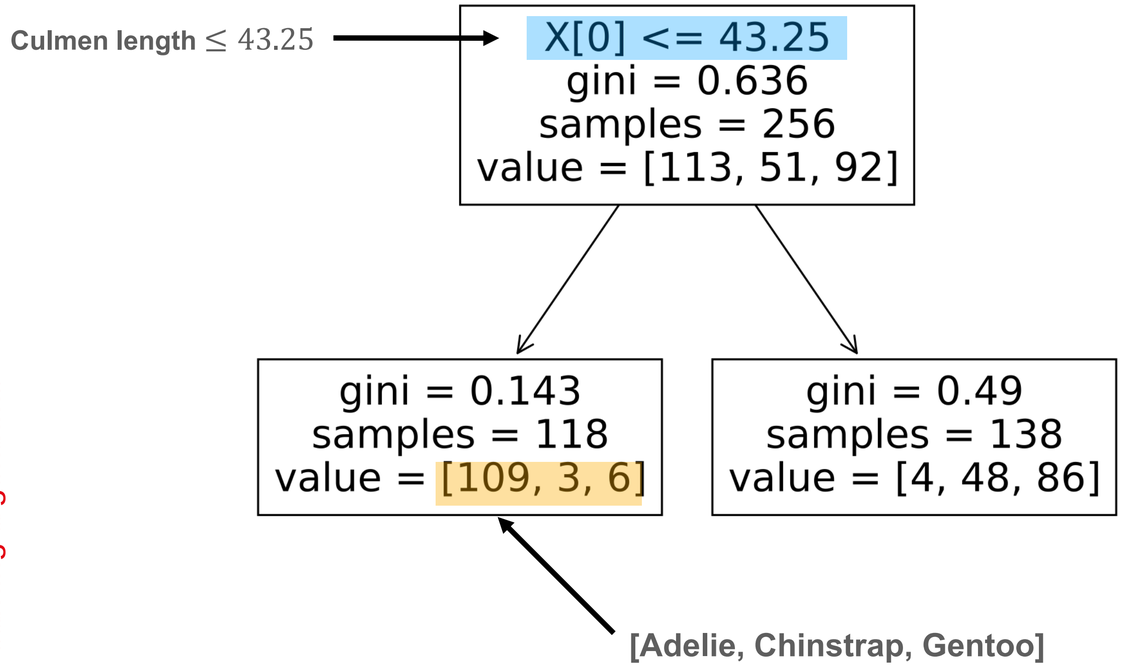
Penguins example

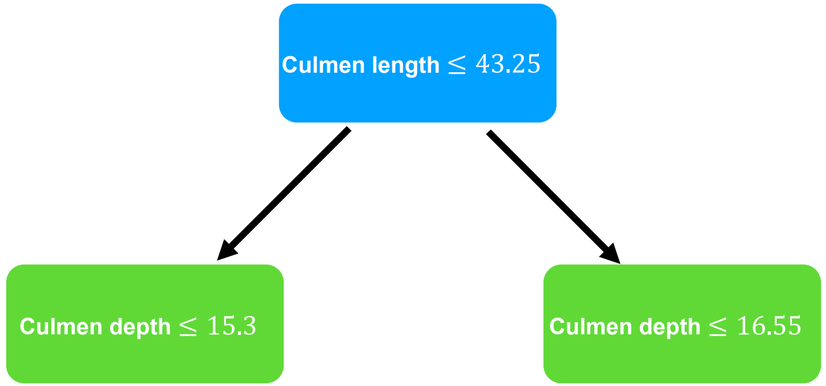




Decision Trees
Penguins example

Python output for *sklearn.tree*



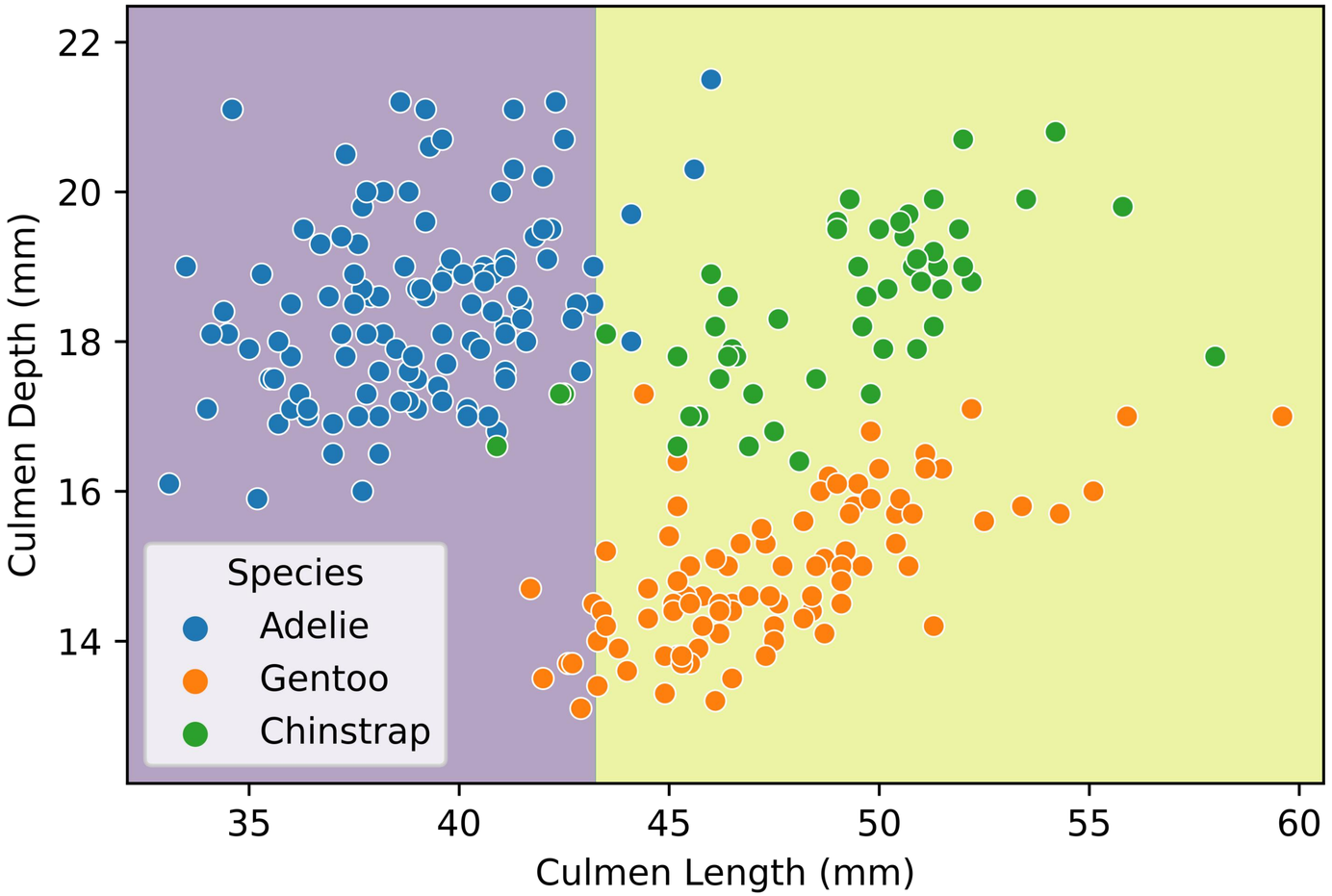


Gini coefficient in decision trees (also called **Gini impurity**) is a measure of how mixed the classes are in a node.
A **Gini of 0** means the node is **pure** (all one class); higher values indicate more **impurity**.

We select Culmen length as the first **discriminative feature**

Decision Trees

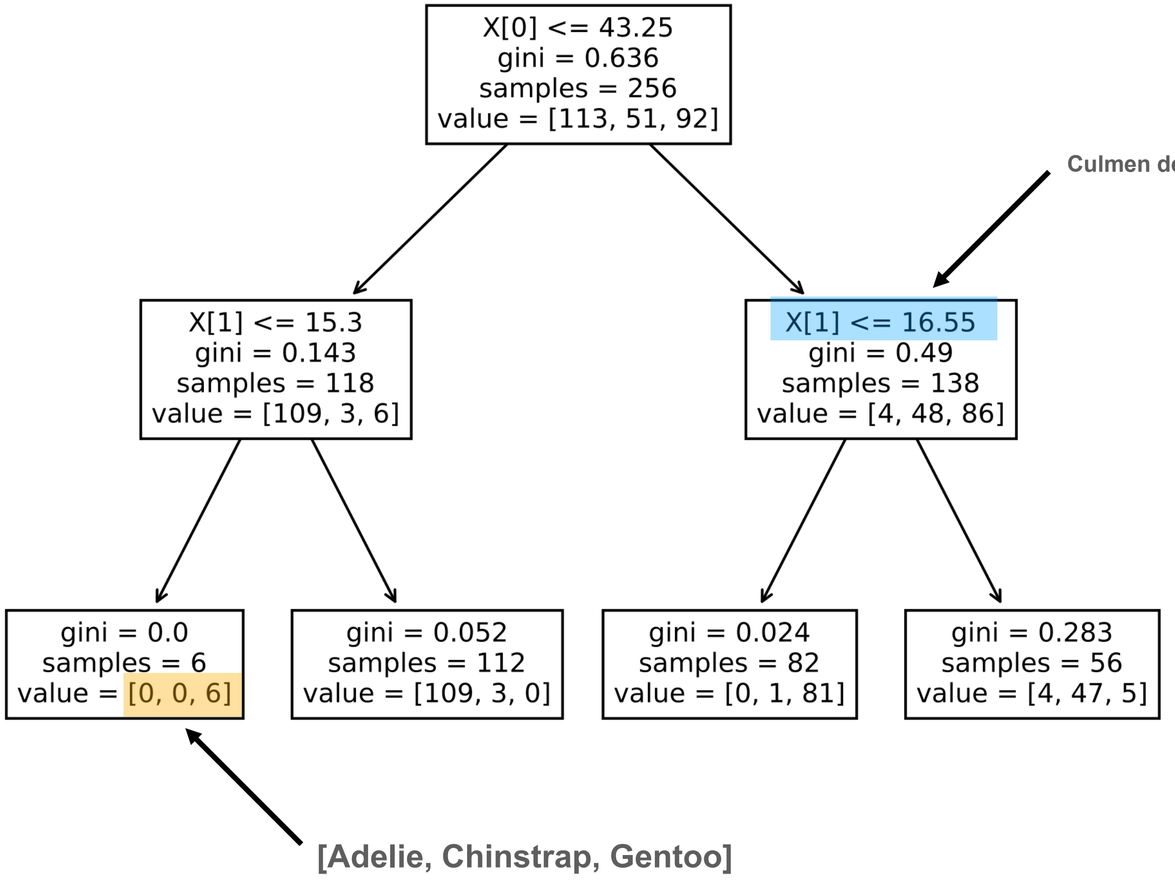
Penguins example

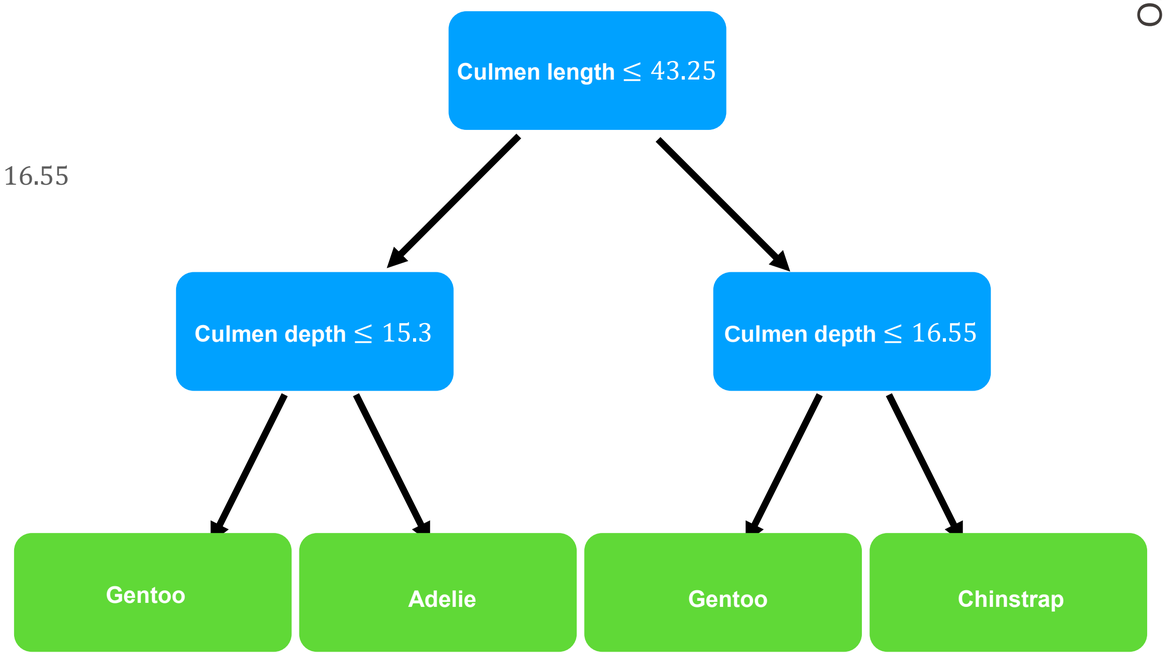


Decision Trees

Penguins example

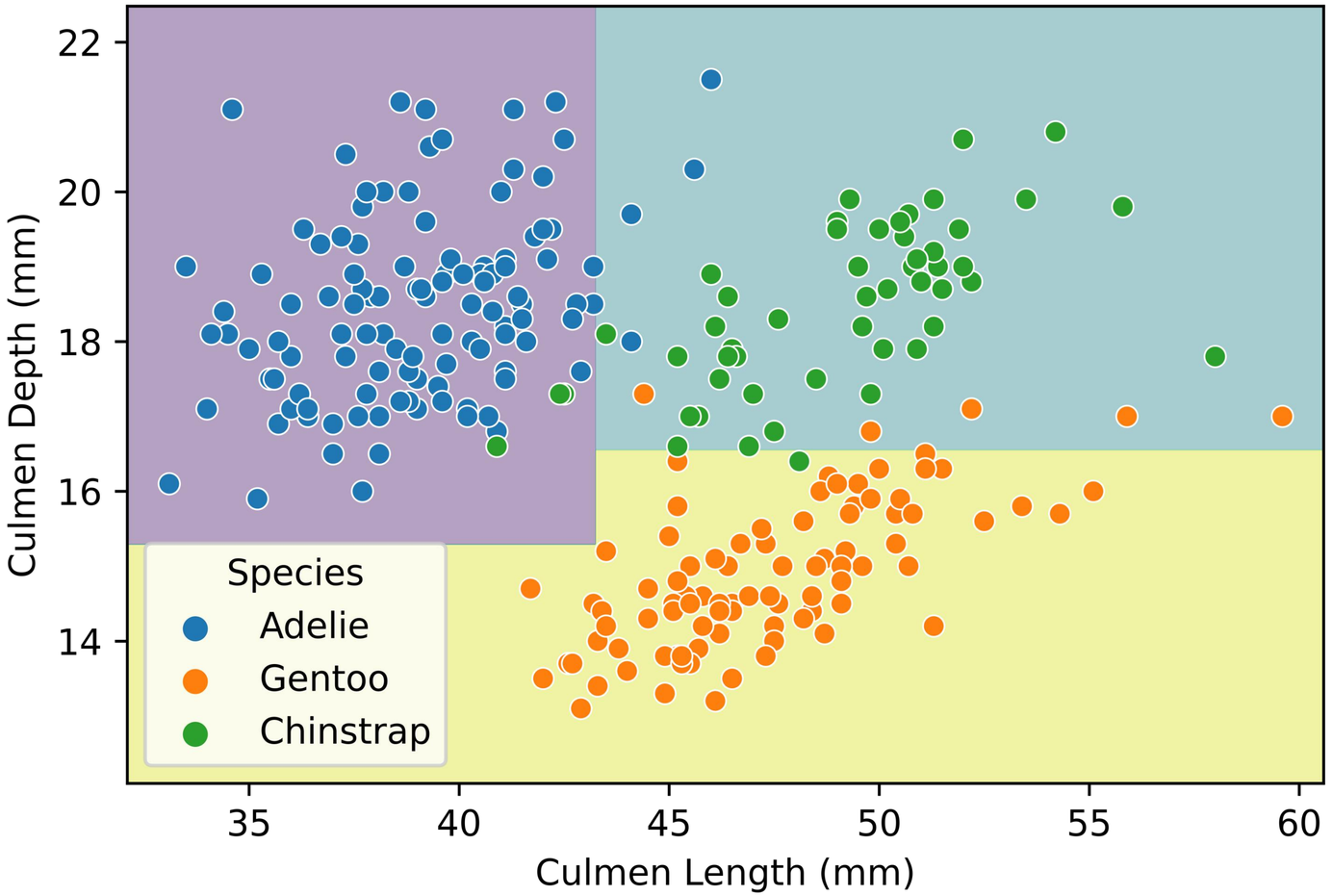
Python output for *sklearn.tree*





Decision Trees

Penguins example



O. Fink /A. Alahi

Decision Trees

Information gain

Which feature should we select for the root node ?

Features | Label

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

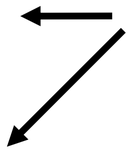
Decision Trees

Information gain

Which feature should we select for the root node ?

Features | Label

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



Partition based on feature « credit rating »

Decision Trees

Information gain

Which feature should we select for the root node ?

Features | Label

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Other possible partition based on feature « student »

We have multiple choices of partition

We need a method to select the best feature to build our first partition

Decision Trees
Tree construction

- We select the most discriminative **Feature**
  - Discriminative power based on a score:
    - Information gain (ID3/C4.5)
    - Gini impurity (CART)
- We create a node based on this feature
- We repeat for each new branch until all the samples are classified

Entropy

Scenario 1: Fair Coin

Suppose you have a fair coin, which has an equal probability (50% each) of landing on heads or tails. In this case, the outcome is completely uncertain, and you have no better strategy than guessing randomly. Here, the entropy, which measures the uncertainty or randomness in the system, is at its maximum. In information theory, entropy for this scenario can be calculated as:

$$H(X) = -\left(\tfrac{1}{2}\log_2\tfrac{1}{2} + \tfrac{1}{2}\log_2\tfrac{1}{2}\right) = 1 \text{ bit}$$

Scenario 2: Biased Coin

Now, consider a biased coin that lands on heads 90% of the time and tails 10% of the time. The outcome is more predictable because it's much more likely to be heads. The entropy in this case is lower, reflecting the reduced uncertainty:

$$H(X) = -\left(0.9\log_2 0.9 + 0.1\log_2 0.1\right) \approx 0.47 \text{ bits}$$

O. Fink /A. Alahi

Decision Trees
Information gain

At a given branch in the tree, the set of **samples S** to be classified has **P positive** and **N negative** instances

The **entropy of the set S** is : $H(P,N) = -(\frac{P}{P+N}\log_2(\frac{P}{P+N}) + \frac{N}{P+N}\log_2(\frac{N}{P+N}))$

Note : $H(P,N) = 0$ →No uncertainty ; $H(P,N) = 1$ →Maximal uncertainty

Feature A partitions S into S_1, S_2 ,…, S_v

The **entropy of the feature A** is : $H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$

The **information gain** obtained by splitting S using A is : $Gain(A) = H(P,N) - H(A)$

O. Fink /A. Alahi

Decision Trees

Information gain

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

We start by computing the entropy of each class

$$H(P,N) = H(9,5) = 0.94$$

$$H(P,N) = -\left(\frac{P}{P+N}\log_2\left(\frac{P}{P+N}\right) + \frac{N}{P+N}\log_2\left(\frac{N}{P+N}\right)\right)$$

$$H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$$

$$Gain(A) = H(P,N) - H(A)$$

Decision Trees

Information gain

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

We compute the entropy of each sets

Age [<=30] $H(2,3) = 0.97$

Age [31…40] $H(4,0) = 0$

Age [>40] $H(3,2) = 0.97$

Student [yes] $H(6,1) = 0.59$

Student [no] $H(3,4) = 0.98$

Income [high] $H(2,2) = 1$

Income [med] $H(4,2) = 0.92$

Income [low] $H(3,1) = 0.81$

Rating [fair] $H(6,2) = 0.81$

Rating [exc] $H(3,3) = 1$

Decision Trees

Information gain

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

We compute the entropy of each feature

$$H(age) = p([<= 30])H([<= 30]) + p([31\ldots40])H([31\ldots40]) + p([> 40])H([> 40]) = 0.69$$

$$H(income) = 0.91$$

$$H(student) = 0.78$$

$$H(rating) = 0.89$$

Decision Trees
Information gain

age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Finally we compute the gain

$$Gain(Age) = 0.94 - 0.69 = 0.25$$

$$Gain(Income) = 0.94 - 0.91 = 0.03$$

$$Gain(Student) = 0.94 - 0.78 = 0.16$$

$$Gain(Rating) = 0.94 - 0.89 = 0.05$$

The first node is split on the age

Decision Trees

Extracting classification rules from trees

Represent the knowledge in the form of **IF-THEN rules**

- One rule is created for each path from the root to a leaf
- Each feature-value pair along a path forms a conjunction
- The leaf node holds the class prediction

Rules are easier for humans to understand

Example :

IF age = *"<=30" AND student = "no"* THEN buys_computer = *"no"*

IF age = *"<=30" AND student = "yes"* THEN buys_computer = *"yes"*

IF age = *"31...40"* THEN buys_computer = *"yes"*

IF age = *">40" AND credit_rating = "excellent"* THEN buys_computer = *"yes"*

IF age = *">40" AND credit_rating = "fair"* THEN buys_computer = *"no"*

Decision Trees
Continuous features

With continuous features, we **cannot have a separate branch for each value**
→ use **binary decision trees**

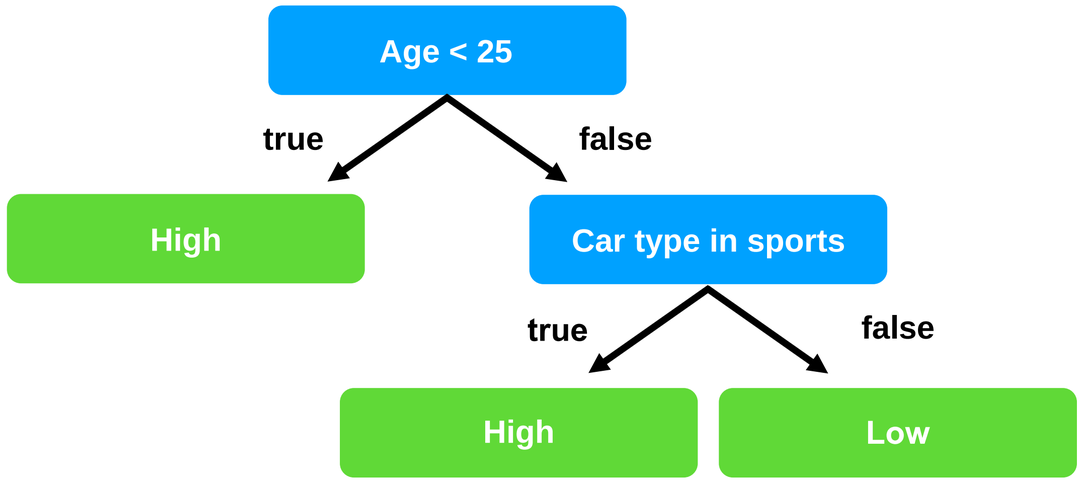
Binary decision trees :

- For continuous feature A, a split is defined by $val(A) < X$
- For categorical feature A, a split is defined by a subset $X \subseteq domain(A)$

Decision Trees

Continuous features

Continuous Feature	Categorical Feature	Class label
Age	**Car type**	**Risk**
23	family	high
17	sports	high
43	sports	high
68	family	low
32	truck	low
20	family	high



Decision Trees

Splitting continuous features

Approach :

- **Sort the data** according to feature value
- **Determine** the value of **X which maximizes information gain** by scanning through the data items

A relevant decision point exists only where the class label changes!

0	1	2	3	4	5	6	7	8	9	10
P	P	P	N	N	N	P	N	N	P	P

Decision Trees

Splitting continuous features

tid	Age	Risk
0	23	high
1	17	high
2	43	high
3	68	low
4	32	low
5	20	high

Reorder the data depending on Age

tid	Age	Risk
1	17	high
5	20	high
0	23	high
4	32	low
2	43	high
3	68	low

$$H(P,N) = -\left(\frac{P}{P+N}\log_2\left(\frac{P}{P+N}\right) + \frac{N}{P+N}\log_2\left(\frac{N}{P+N}\right)\right)$$

$$H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$$

$$Gain(A) = H(P,N) - H(A)$$

Decision Trees

Splitting continuous features

tid	Age	Risk
0	23	high
1	17	high
2	43	high
3	68	low
4	32	low
5	20	high

Reorder the data depending on Age

tid	Age	Risk
1	17	high
5	20	high
0	23	high
4	32	low
2	43	high
3	68	low

Position 0

	High	Low
Count above	0	0
Count below	4	2

$$H(P,N) = H(4,2) = 0.918$$

$$H(A) = 0 + 1 \times H(4,2) = 0.918$$

$$Gain = 0$$

$$H(P,N) = -(\frac{P}{P+N}\log_2(\frac{P}{P+N}) + \frac{N}{P+N}\log_2(\frac{N}{P+N}))$$

$$H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$$

$$Gain(A) = H(P,N) - H(A)$$

Decision Trees

Splitting continuous features

tid	Age	Risk
0	23	high
1	17	high
2	43	high
3	68	low
4	32	low
5	20	high

Reorder the data depending on Age

tid	Age	Risk
1	17	high
5	20	high
0	23	high
4	32	low
2	43	high
3	68	low

Position 3

	High	Low
Count above	3	0
Count below	1	2

$$H(P,N) = H(4,2) = 0.918$$

$$H(A) = 0 + \frac{1}{2} \times H(1,2) = 0.459$$

$$Gain = 0.918 - 0.459 = 0.459$$

$$H(P,N) = -(\frac{P}{P+N}\log_2(\frac{P}{P+N}) + \frac{N}{P+N}\log_2(\frac{N}{P+N}))$$

$$H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$$

$$Gain(A) = H(P,N) - H(A)$$

Decision Trees

Splitting continuous features

tid	Age	Risk
0	23	high
1	17	high
2	43	high
3	68	low
4	32	low
5	20	high

Reorder the data depending on Age

tid	Age	Risk
1	17	high
5	20	high
0	23	high
4	32	low
2	43	high
3	68	low

Position 6

	High	Low
Count above	4	2
Count below	0	0

$$H(P,N) = H(4,2) = 0.918$$

$$H(A) = 1 \times H(4,2) + 0 = 0.918$$

$$Gain = 0$$

$$H(P,N) = -(\frac{P}{P+N}\log_2(\frac{P}{P+N}) + \frac{N}{P+N}\log_2(\frac{N}{P+N}))$$

$$H(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{P+N} H(P_i, N_i)$$

$$Gain(A) = H(P,N) - H(A)$$

Decision Trees

Scalability of continuous feature splits

Naive implementation :

- At each step, the dataset is split in subsets that are associated with a tree node

Problem :

- For evaluating which continuous feature to split, data needs to be sorted according to these features
- Becomes **dominating cost**

Coding Decision Tree

```
from sklearn import datasets
from sklearn import tree

# Load the iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Train a decision tree classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)

# Predict the class for a new sample
sample = [[5, 4, 3, 2]]
prediction = clf.predict(sample)
print(prediction)
```

Source: A primer to the 42 most commonly used ML algorithms

Decision Trees

Characteristics of decision tree induction

Automatic feature selection

Minimal data preparation

Non-linear model

Easy to interpret and explain

Sensitive to small perturbations in the data

Tend to overfit

No incremental updates

Ensemble methods

Overview

based on the hypothesis that combining multiple models together can often produce a much more powerful model

Idea :

- Take a collection of simple or **weak learners** (decision tree for example)
- Combine their results to make a single, strong learner

Types :

- **Bagging**: train learners in parallel on different samples of the data, then combine outputs through voting or averaging
- **Stacking:** combine model outputs using a second-stage learner like linear regression
- **Boosting**: train learners on the filtered output of other learners

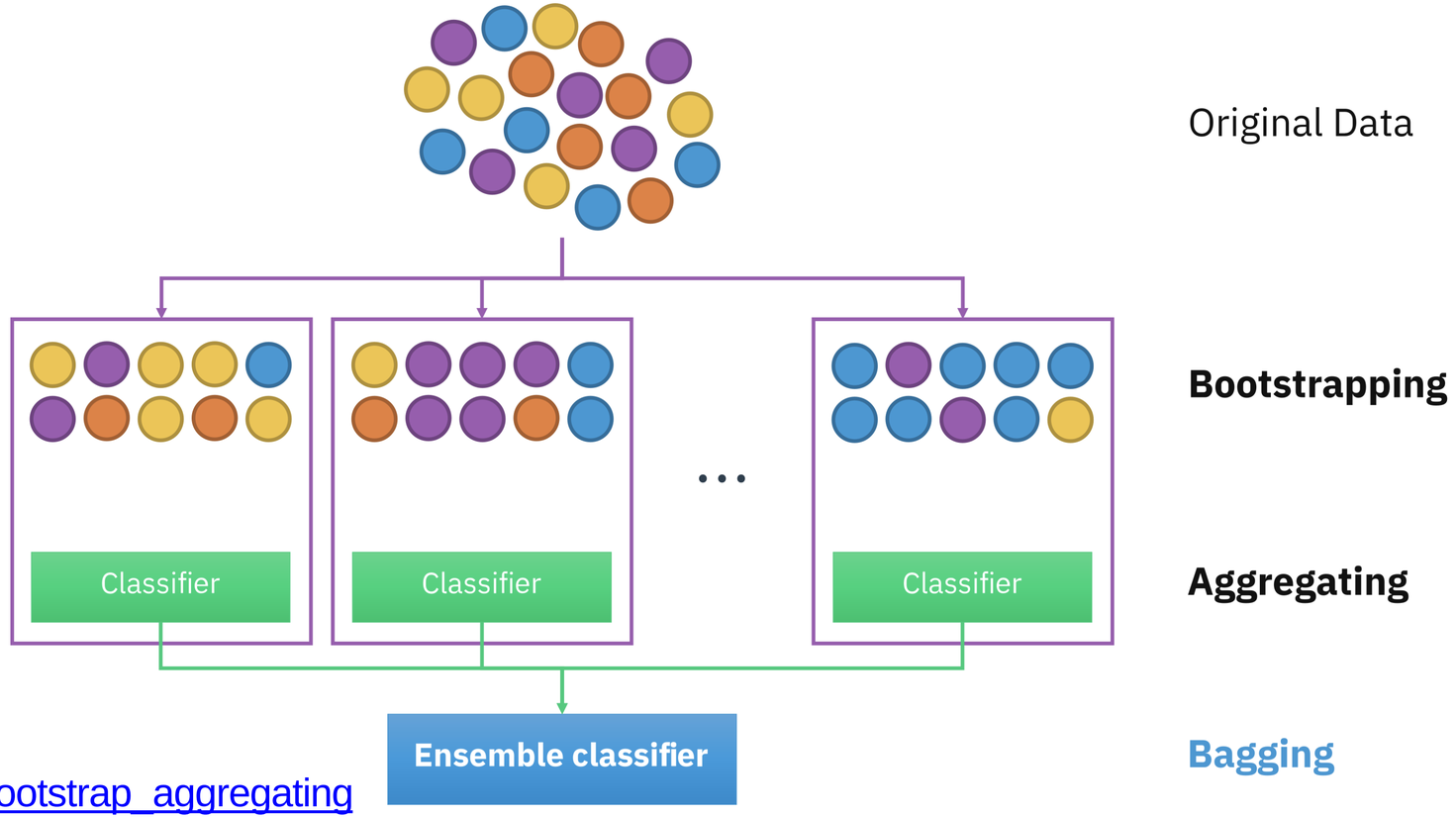
Learn more : https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Ensemble methods
Bagging

Bagging = Bootstrap aggregating

Improve the stability and accuracy of machine learning algorithm

Reduces variance and helps avoid overfitting

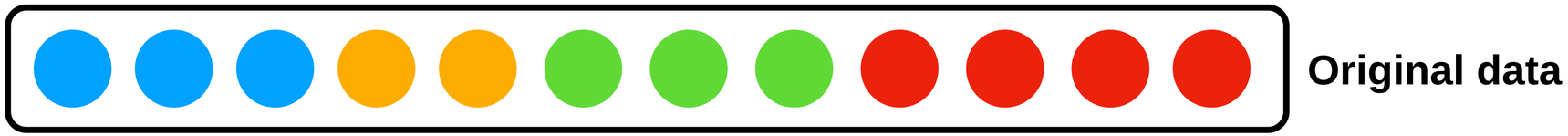


See: https://en.wikipedia.org/wiki/Bootstrap_aggregating

Ensemble methods

Bootstrap

Method to generate multiple datasets with good statistical properties from an original dataset



Original data

Ensemble methods

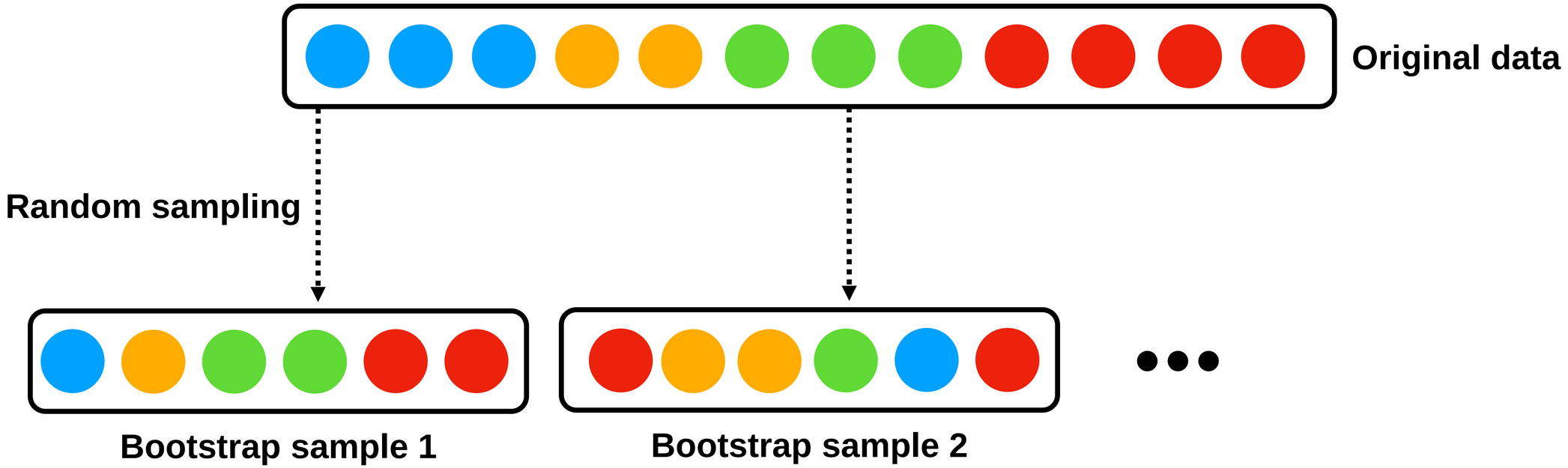
Bootstrap

Method to generate multiple datasets with good statistical properties from an original dataset

Ensemble methods

Bootstrap

Method to generate multiple datasets with good statistical properties from an original dataset



Ensemble methods
Random Forests

Learn **K different decision trees** from independent samples of the data (bagging) :

→ vote between different learners, so models should not be too similar

Aggregate output: **majority vote**

Ensemble methods
Sampling Strategies

Two sampling strategies are used to select the data on which the classifier is trained

Sampling data :

→ Select a subset of the data → Each tree is trained on different data

Sampling features :

→ Select a subset of features → corresponding nodes in different trees (usually) don't use the same feature to split

each classifier (ex: tree) doesn't see all the data and all the features!

Ensemble methods

Random Forests: Algorithm

of classifiers we train

original size of the dataset

1. Draw K bootstrap **samples of size** N from the original dataset, with replacement (**bootstrapping**)

2. While constructing the decision tree, select a random set of **m features** out of the p features available to infer split (**feature bagging**)

Coding Random Forest

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn import datasets

# Load a sample dataset
iris = datasets.load_iris()
X, y = iris.data, iris.target

# Create a random forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=0)

# Fit the model to the data
clf.fit(X, y)

# Predict on a new sample
x_new = [[5, 3.2, 1.2, 0.2]]
print(clf.predict(x_new))
```

Source: A primer to the 42 most commonly used ML algorithms

Ensemble methods
Conclusion

- Ensemble methods usually outperform individual decision trees, at the cost of interpretability
- Few hyper-parameters to tune compared to neural nets
- Can reach very good performance with little tuning, especially on tabular data (data structured in rows and columns) → use it as a strong baseline
- Libraries for ensemble methods (mostly gradient boosted trees): **scikit-learn**, **XGBoost** and **LightGBM**

Decision Tree /Ensemble methods

Resources

- Python tutorial:
  - https://jakevdp.github.io/PythonDataScienceHandbook/05.08-random-forests.html
- Maths of Bagging, Boosting and Ensemble Methods (Really optional):
  - ftp://ftp.math.ethz.ch/sfs/Manuscripts/buhlmann/handbook-cs-rev2010.pdf