INTRODUCTION À L'APPRENTISSAGE AUTOMATIQUE

Lecture 2 - Polytech

Caio Corro

OUTLINE

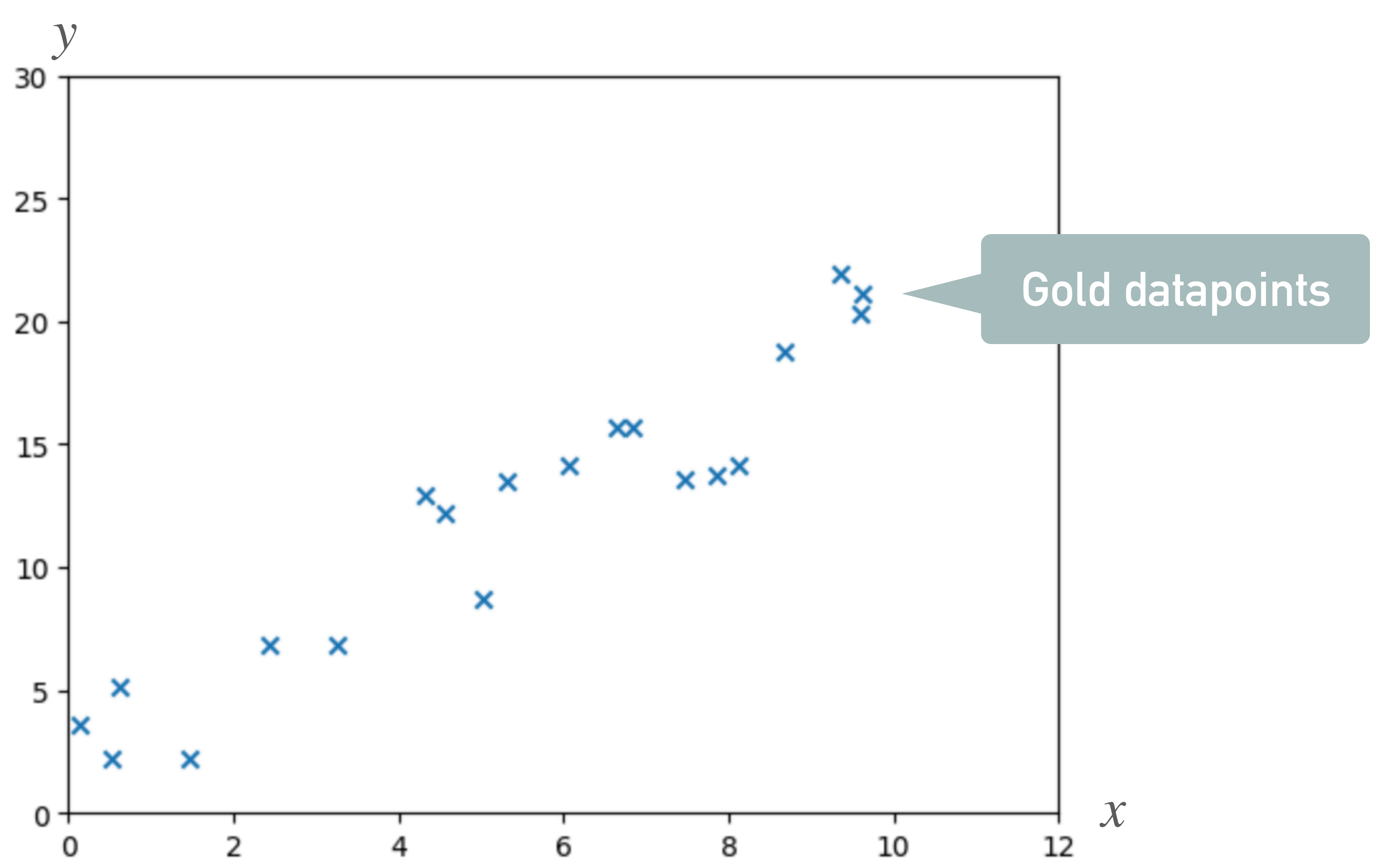
1. Linear models
2. Loss functions
3. Regularization
4. Training algorithms

LINEAR REGRESSION

LINEAR REGRESSION

Setting

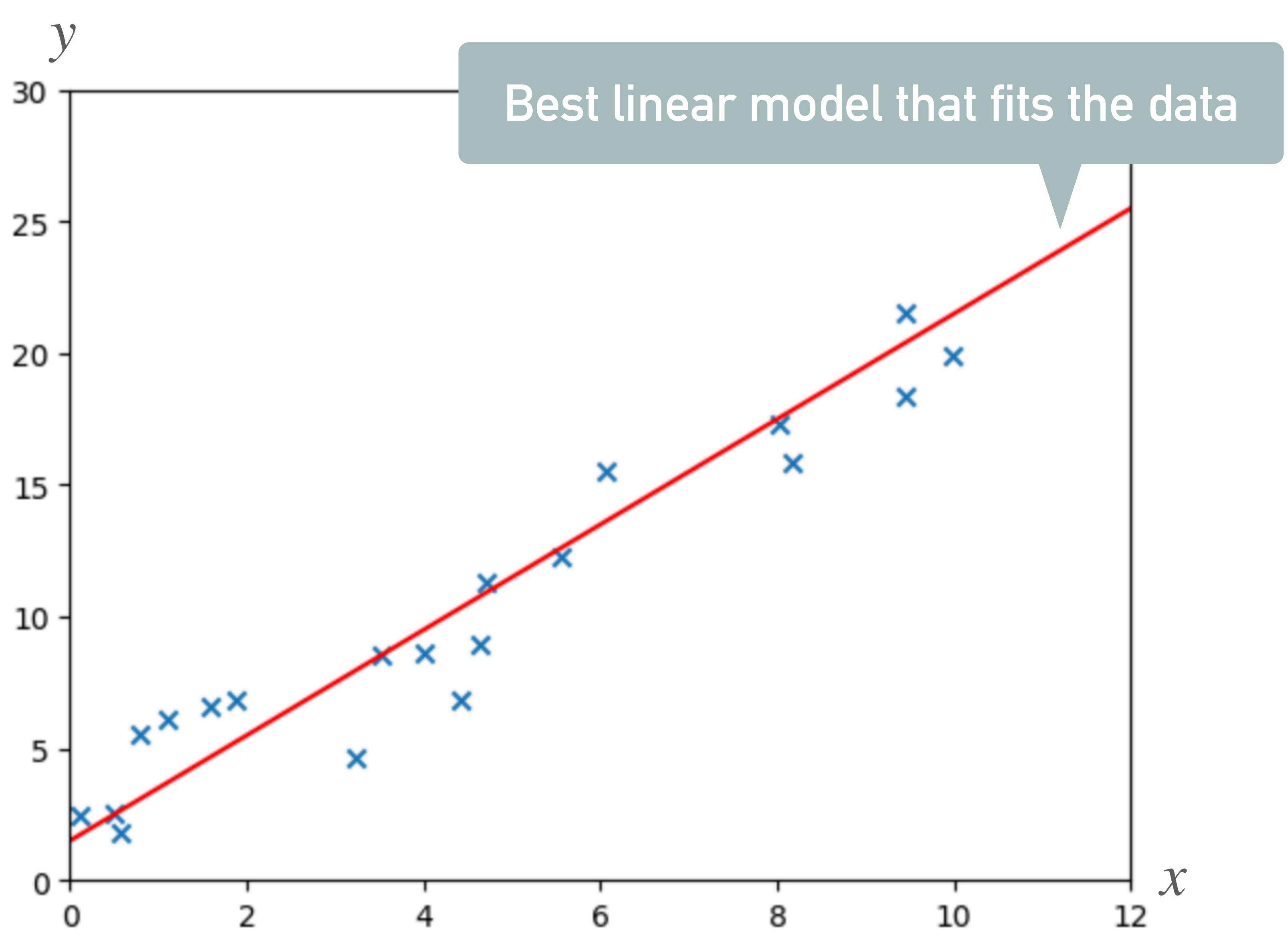
- Regression : we want to predict a scalar value (i.e. a real)
- Linear model : output must be a linear projection of input



LINEAR REGRESSION

Setting

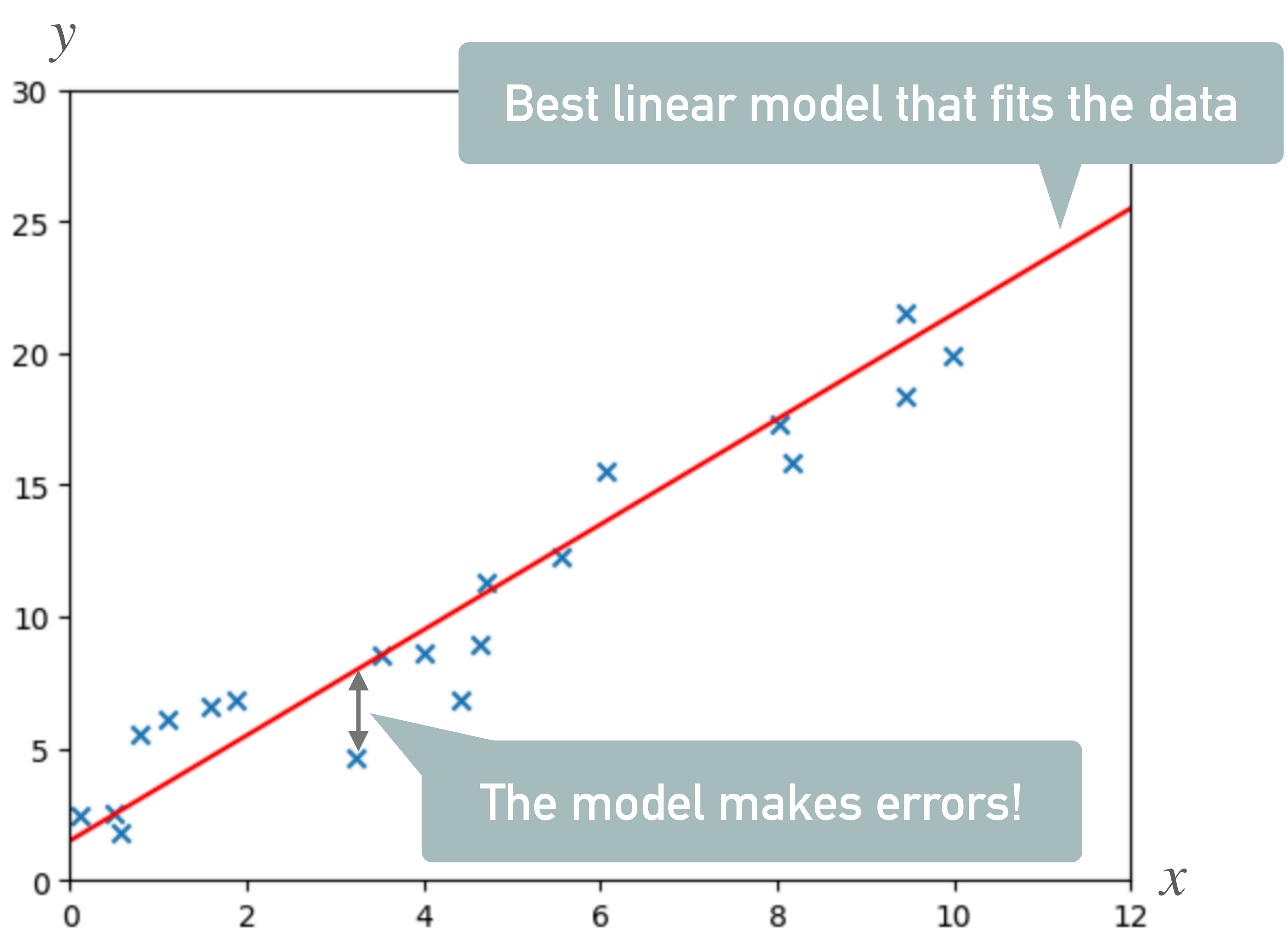
- Regression : we want to predict a scalar value (i.e. a real)
- Linear model : output must be a linear projection of input



LINEAR REGRESSION

Setting

- Regression : we want to predict a scalar value (i.e. a real)
- Linear model : output must be a linear projection of input

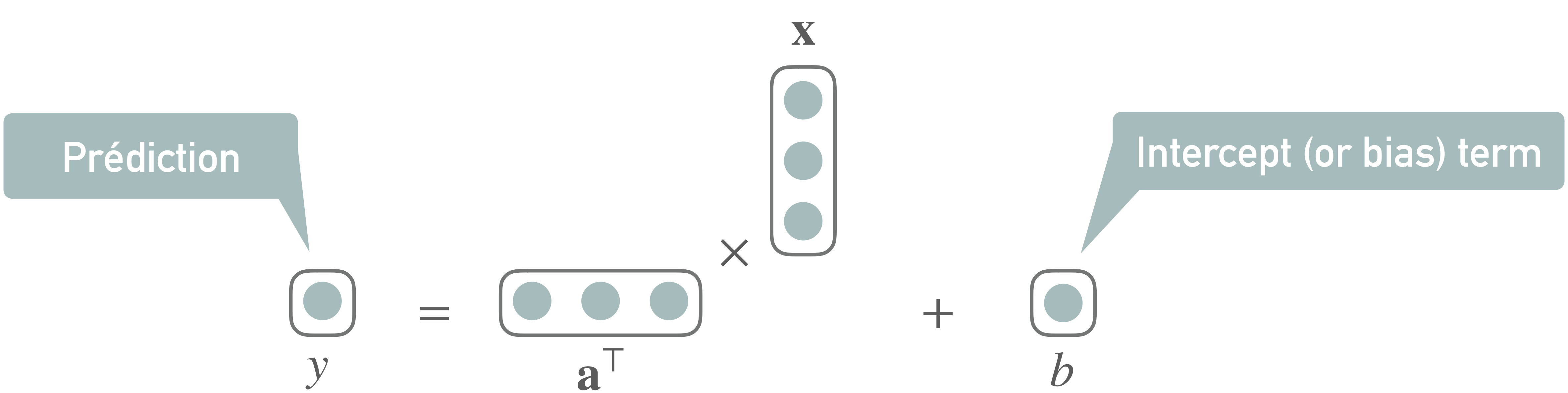


LINEAR REGRESSION

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Parameters: $\theta = \{\mathbf{a}, b\}$ avec $\mathbf{a} \in \mathbb{R}^d, \;\; b \in \mathbb{R}$
- Output: $y \in \mathbb{R}$

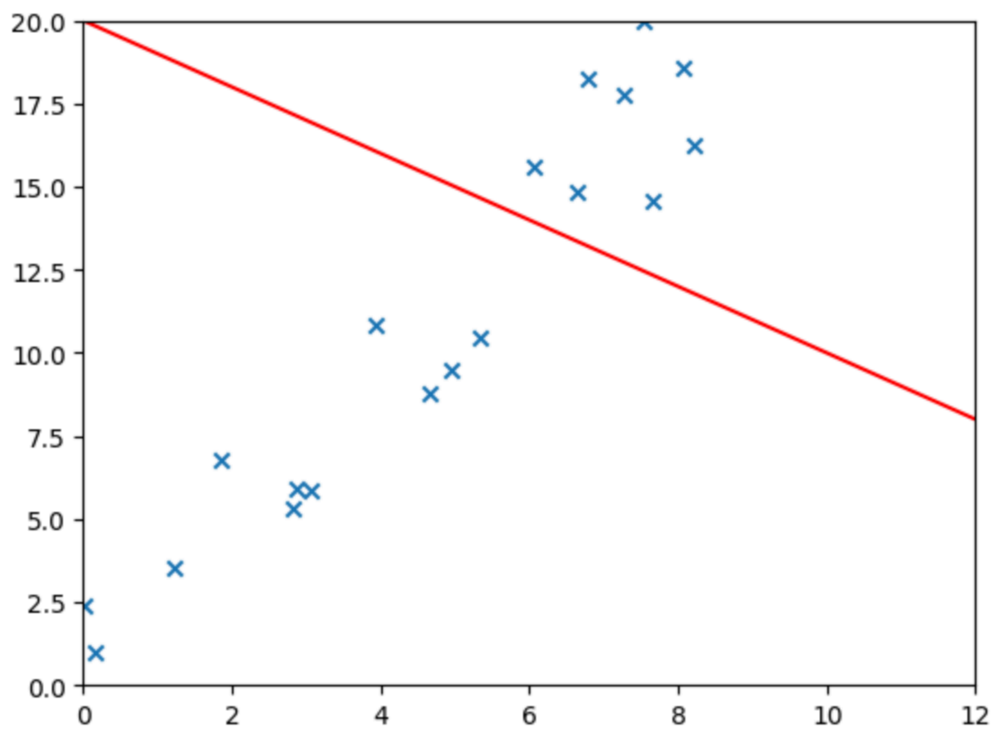
Model

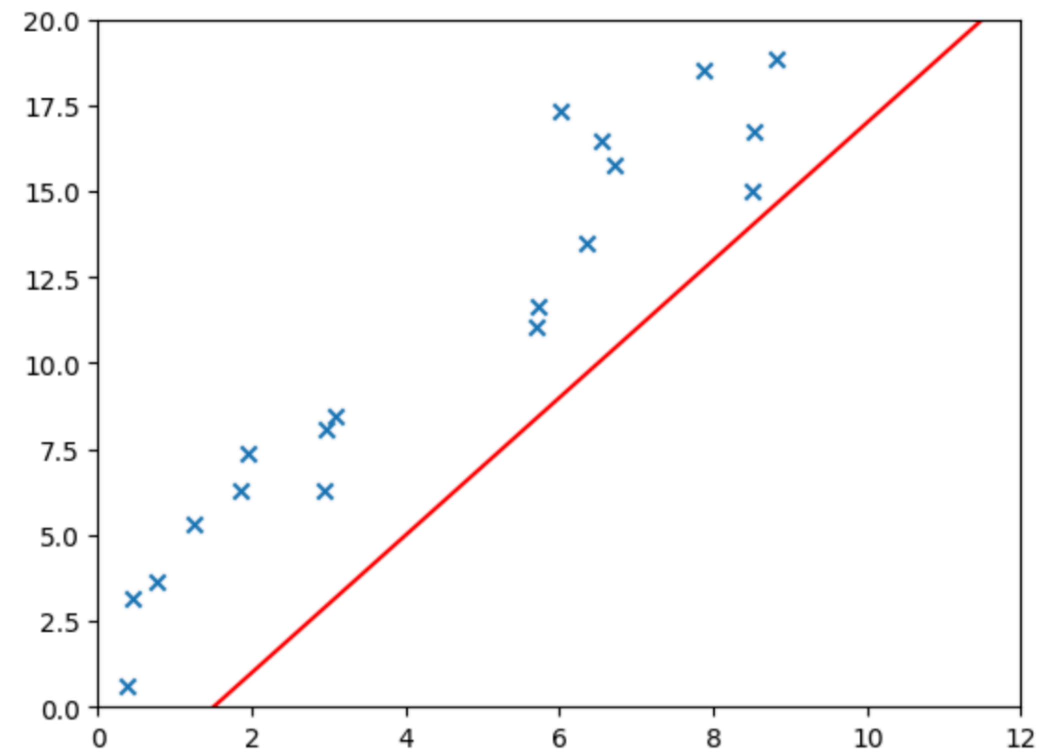
Simple linear projection of the input: $f_\theta(\mathbf{x}) = f(\mathbf{x}; \theta) = \langle \mathbf{a}, \mathbf{x} \rangle + b$

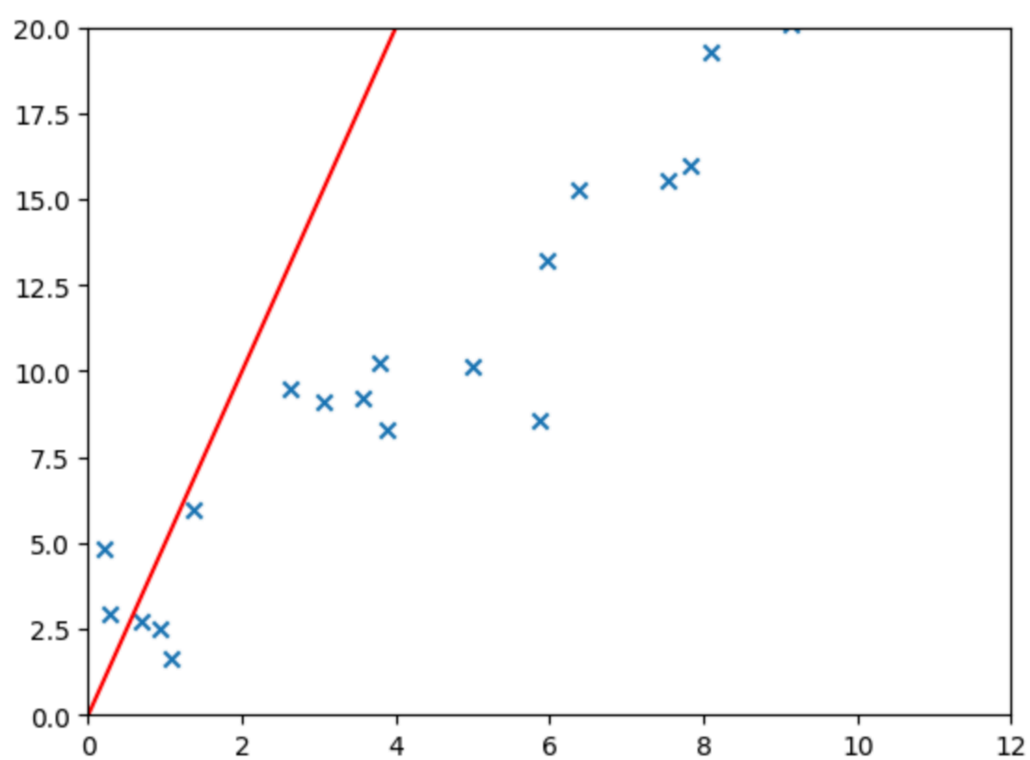


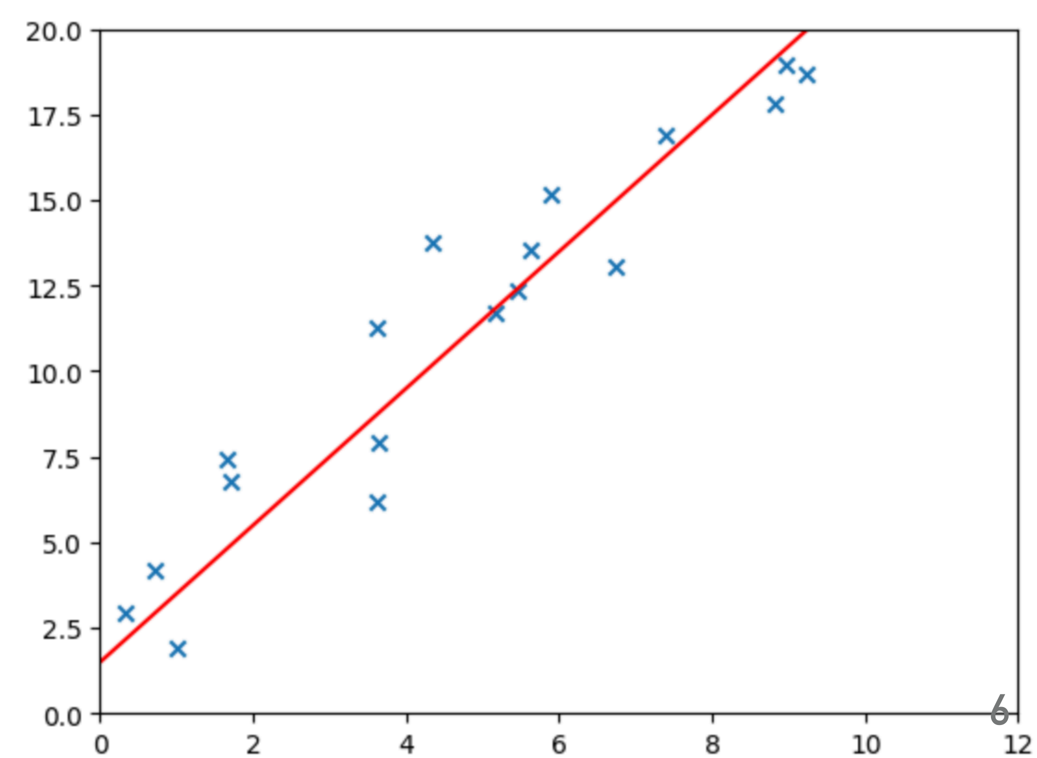
Training problem

Given a dataset $D = \{\ (\mathbf{x}^{(i)}, y^{(i)})\ \}_{i=1}^{n}$, find the best possible set of parameters θ









TRAINING

Loss function

Function that is used to compare the prediction of a model with the expected gold output

$$\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$$

Loss value, non-negative

Gold output

Model output

Properties of loss functions

- Non-negative
- The smaller the loss, the better the model
- Loss is null if and only if the model predicts the correct output
- Convex in the second argument (not always true)

TRAINING

Squared error loss function

$$\ell_{L2}(y, w) = \frac{1}{2}\|y - w\|_2^2 = \frac{1}{2}(y - w)^2$$

Absolute error loss function

$$\ell_{L1}(y, w) = \frac{1}{2}\|y - w\|_1 = |y - w|$$

What is the difference?

- The squared error loss function is differentiable everywhere
- The absolute error loss function is non differentiable when y = w
- The absolute error loss function is less sensitive to outliers

TRAINING

Training problem

Given a dataset $D = \{ (\mathbf{x}^{(i)}, y^{(i)}) \}_{i=1}^{n}$, find the best possible set of parameters θ

=> by minimizing a loss function over the dataset!

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_{(\mathbf{x},y)\in D} \ell\left(y, \; f_\theta(\mathbf{x}) \right)$$

Example with the squared error loss

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_{(\mathbf{x},y)\in D} \frac{1}{2}\left(y - (\langle \mathbf{a}, \mathbf{x} \rangle + b) \right)^2$$

$$= \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_{(\mathbf{x},y)\in D} \frac{1}{2}\left(y - \langle \mathbf{a}, \mathbf{x} \rangle - b \right)^2$$

REGULARIZATION

REGULARIZATION

The overfitting problem

When the input is of large dimension, the model may overfit (i.e. learn data by heart), which mean that the model will not be able to generalize correctly on unseen data.

Parameter regularization (sometimes called penalty term)

Penalize the parameters θ to keep them close to 0.

Intuition => "use as less information as possible"

Main regularization terms:

- L2 regularization: $r(\mathbf{u}) = \frac{1}{2}||\mathbf{u}||_2^2$
- L1 regularization: $r(\mathbf{u}) = \sum_i |u_i|$

REGULARIZATION

Regularization weight, > 0

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_{(\mathbf{x},y)\in D} \ell\left(\, y, \; f_\theta(\mathbf{x}) \, \right) \qquad + \qquad \beta \times r(\mathbf{a})$$

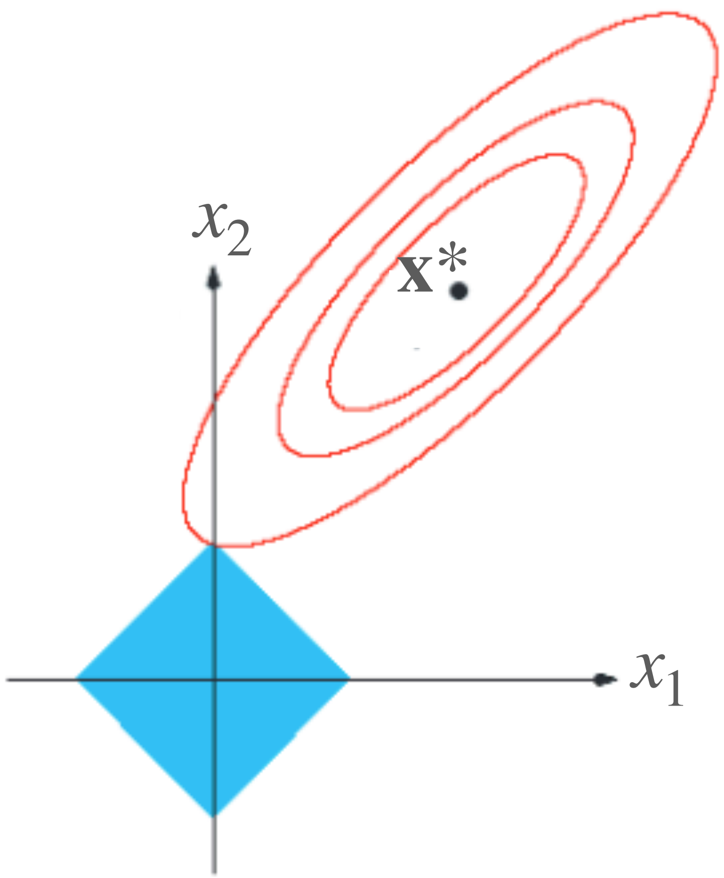
Warning

- Only regularize the projection parameters **a**
- It does not make sense to regularize the intercept term b
  (why would we want the prediction function to pass through the origin?)

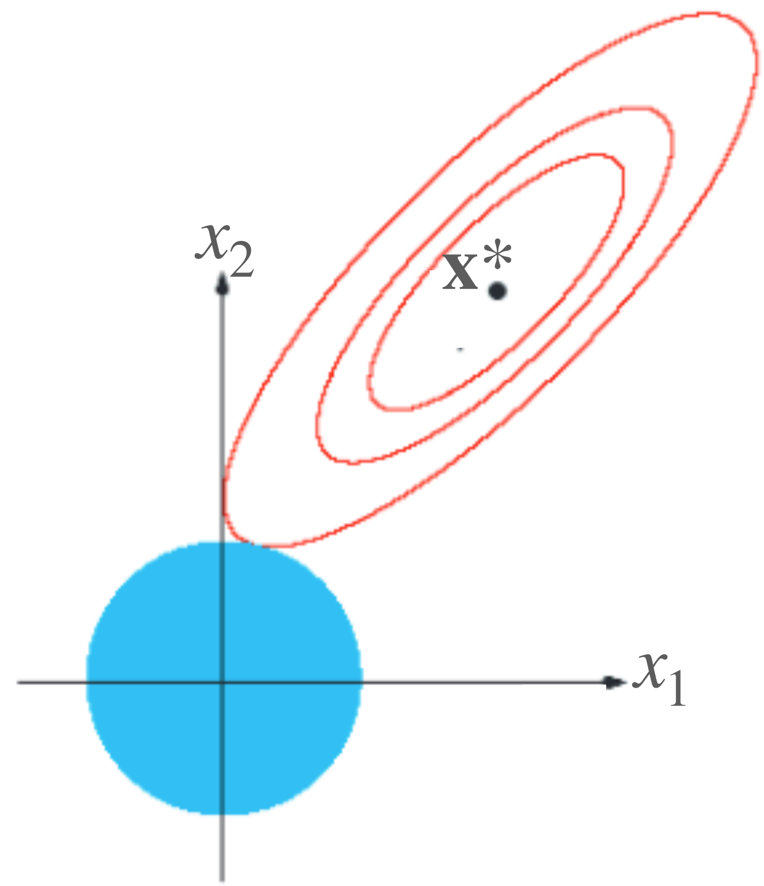
GEOMETRIC INTUITION

Difference between L1 and L2 regularization

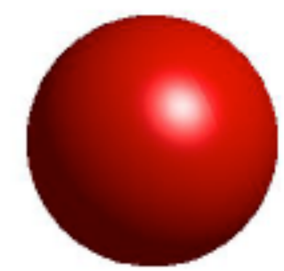
- L2 norm is isotropic, it does not favor any direction
- L1 norm favors sparse vectors, that is vectors with (many) zeros



L1 regularization



L2 regularization



Illustrations: Optimization with Sparsity-Inducing Penalties (Bach et al.), The Elements of Statistical Learning (Hastie et al.)

FEATURE SELECTION

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \quad \frac{1}{n} \sum_{(\mathbf{x},y)\in D} \ell\left(y, \; f_\theta(\mathbf{x}) \right) \quad + \quad \beta \times \sum_i |a_i|$$

L1 and regularization path

L1 regularization favors solutions with many zeros

- Features with a weight of 0 in a are not used by the model
- We can use L1 regularization to "sort" the features, find the one which are important
    1. Start with a large β so no feature is selected
    2. Gradually decrease β to see which features are used by the model

TRAINING

(other slides)

TRICK FOR L1-REGULARIZATION

Issue of L1 regularization

- Non-differentiable in zero
- But sub-differentiable

Trick for non-differentiable function

- Rewrite the function as a maximum of differentiable functions

$$f(u) = \max(\ f_1(u),\ f_2(u),\ f_3(u), \ldots)$$

- Use any convex combination of gradients of functions in the following set:

$$F(u) = \{f_i \,|\, f_i(u) = f(u)\}$$

TRICK FOR L1-REGULARIZATION

Problem

This approach will not lead to sparse vector of parameters :(

Instead, there exists specialized optimization algorithms for L1 regularization :

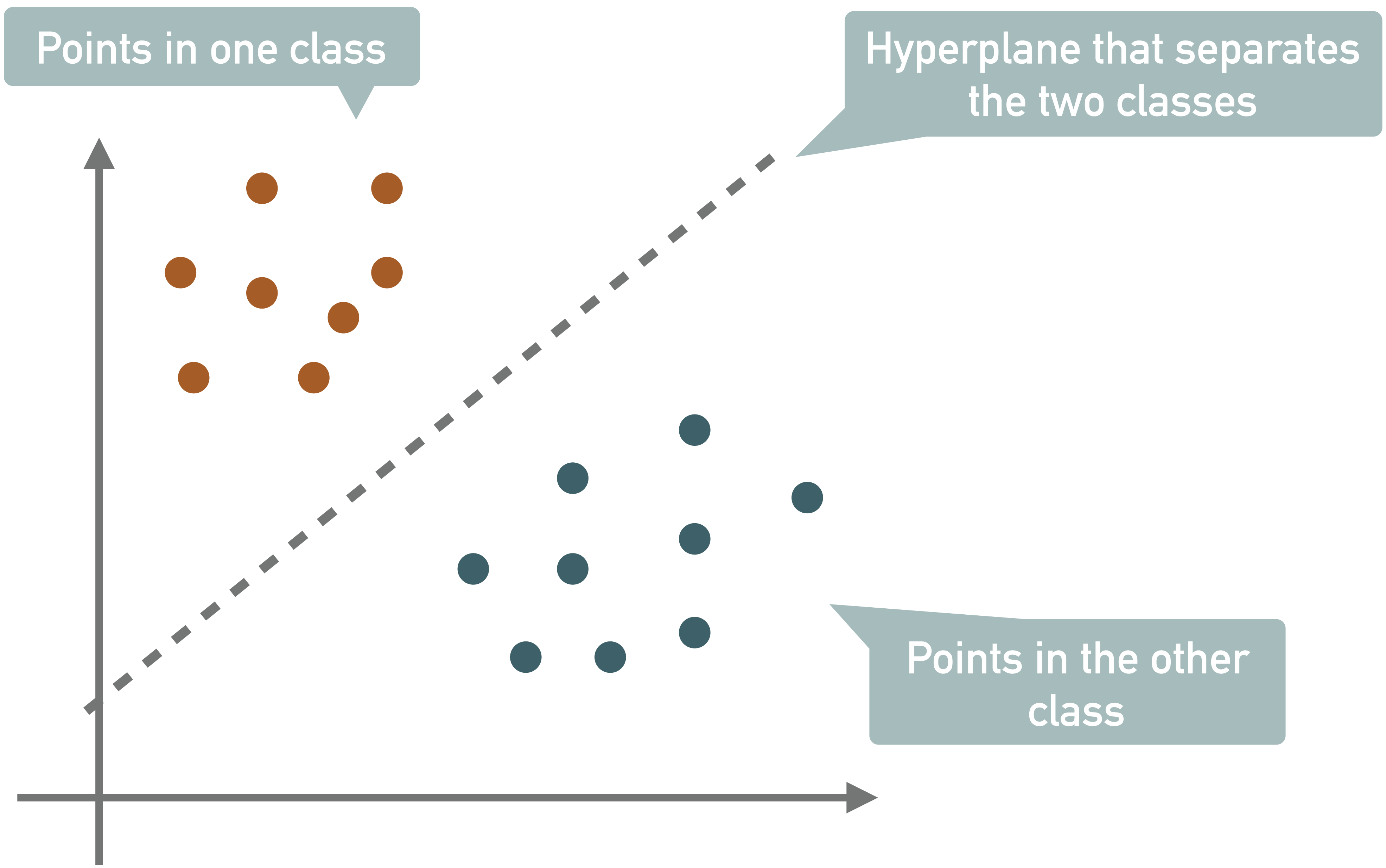
- Proximal methods
- Coordinate descent
- ...

Easy trick

- If the sign of a parameter a_i changed after update, set it to 0
- Use 0 as a partial derivative of the L1 norm for parameters equal to 0

BINARY CLASSIFICATION

BINARY LINEAR CLASSIFIER: INTUITION



DOT PRODUCT 1/2

Let $a \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$ be two vectors.

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

The dot product is defined as: $a^\top x = \langle a, x \rangle = \sum_{i=1}^{n} a_i \times x_i$

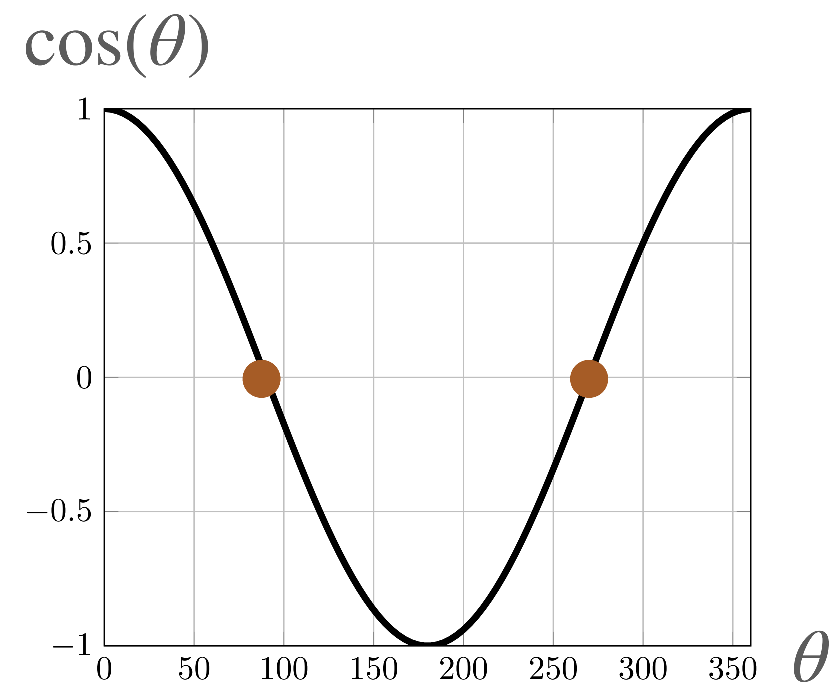
Transpose and matrix multiplication

Properties

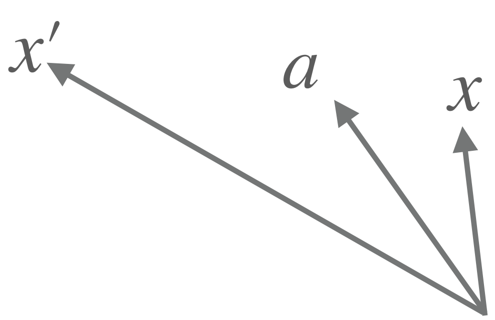
- $a^\top x = \|a\| \|x\| \cos \theta$ where $\|w\|$ is the magnitude of the vector: $\|a\| = \sqrt{\sum_{i=1}^{n} a_i^2}$
- $a^\top x = 0$ if and only if vectors a and x are orthogonal

DOT PRODUCT 2/2

- $a^\top x = \|a\| \|x\| \cos\theta$
- $\|a\| \geq 0$

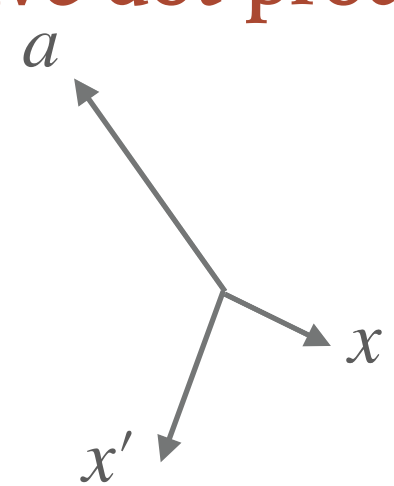


Positive dot product



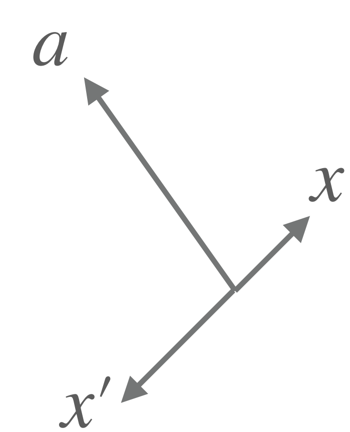
$a^\top x > 0 \quad a^\top x' > 0$

Negative dot product



$a^\top x < 0 \quad a^\top x' < 0$

Null dot product



$a^\top x = 0 \quad a^\top x' = 0$

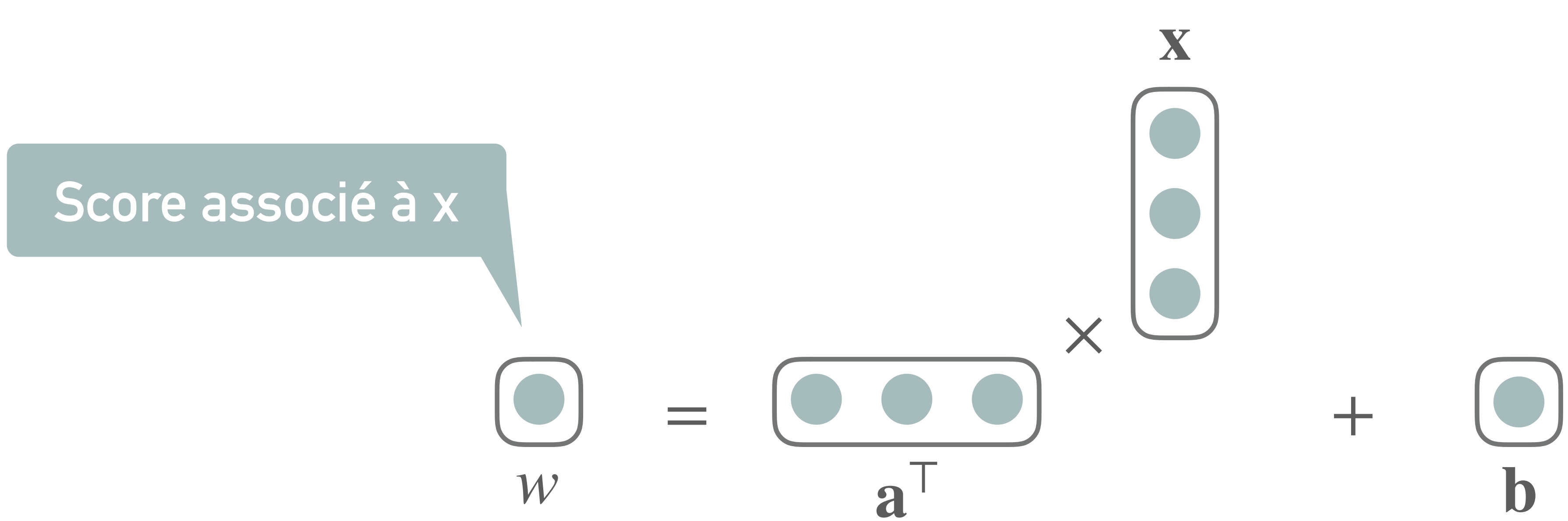
BINARY CLASSIFICATION

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Parameters: $\theta = \{\mathbf{a}, b\}$ avec $\mathbf{a} \in \mathbb{R}^d, \;\; b \in \mathbb{R}$
- Output: $y \in \{0,1\}$ ou $y \in \{-1,1\}$

Scoring function

Étant donné une entrée, calcul un score associé à la sortie

$$s_\theta(\mathbf{x}) = s(\mathbf{x}; \theta) = \langle \mathbf{a}, \mathbf{x} \rangle + b$$



BINARY CLASSIFICATION

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Parameters: $\theta = \{\mathbf{a}, b\}$ avec $\mathbf{a} \in \mathbb{R}^d, \ b \in \mathbb{R}$
- Output: $y \in \{0,1\}$ ou $y \in \{-1,1\}$

Scoring function

Compute a score associated with an input (this function is parameterized)

$$s_\theta(\mathbf{x}) = s(\mathbf{x}; \theta) = \langle \mathbf{a}, \mathbf{x} \rangle + b$$

Prediction function

Compute the output associated with a score (this function is not parameterized)

$$\hat{\mathbf{y}}(w) = \begin{cases} 1 & \text{if } w \geq 0, \\ 0 & \text{otherwise.} \end{cases} \qquad \text{ou} \qquad \hat{\mathbf{y}}(w) = \begin{cases} 1 & \text{if } w \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

BINARY CLASSIFICATION

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Parameters: $\theta = \{\mathbf{a}, b\}$ avec $\mathbf{a} \in \mathbb{R}^d, \;\; b \in \mathbb{R}$
- Output: $y \in \{0,1\}$ ou $y \in \{-1,1\}$

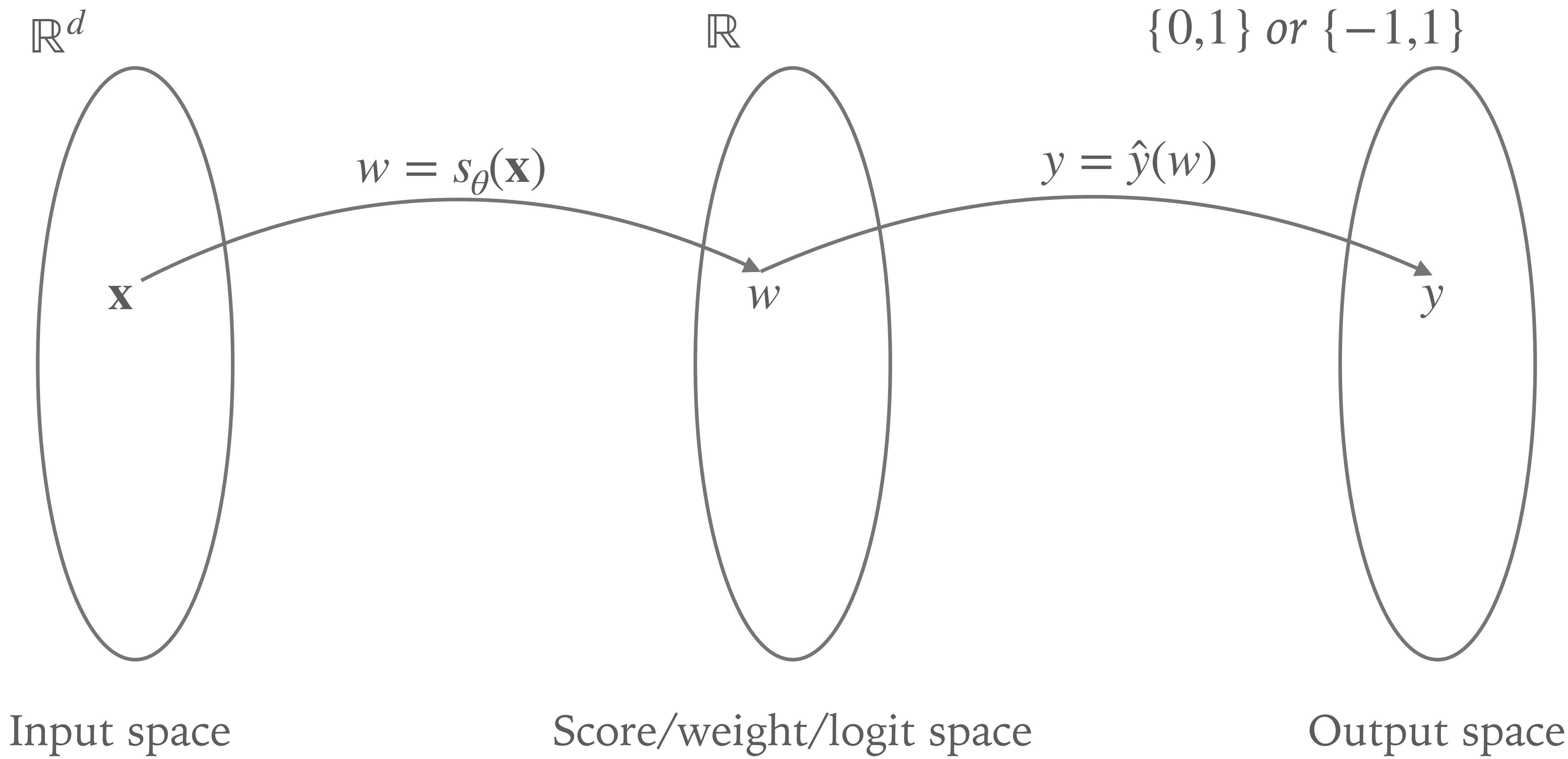
Full model

Sometimes we directly define the full model (scoring function + prediction function)
Vocabulary issue: this is also called the prediction function

$$f_\theta(\mathbf{x}) = \begin{cases} 1 & \textit{if } \langle \mathbf{a}, \mathbf{x} \rangle + b \geq 0, \\ 0 & \textit{otherwise.} \end{cases} \quad \text{ou} \quad f_\theta(\mathbf{x}) = \begin{cases} 1 & \textit{if } \langle \mathbf{a}, \mathbf{x} \rangle + b \geq 0, \\ -1 & \textit{otherwise.} \end{cases}$$

BINARY LINEAR CLASSIFIER: DEFINITION

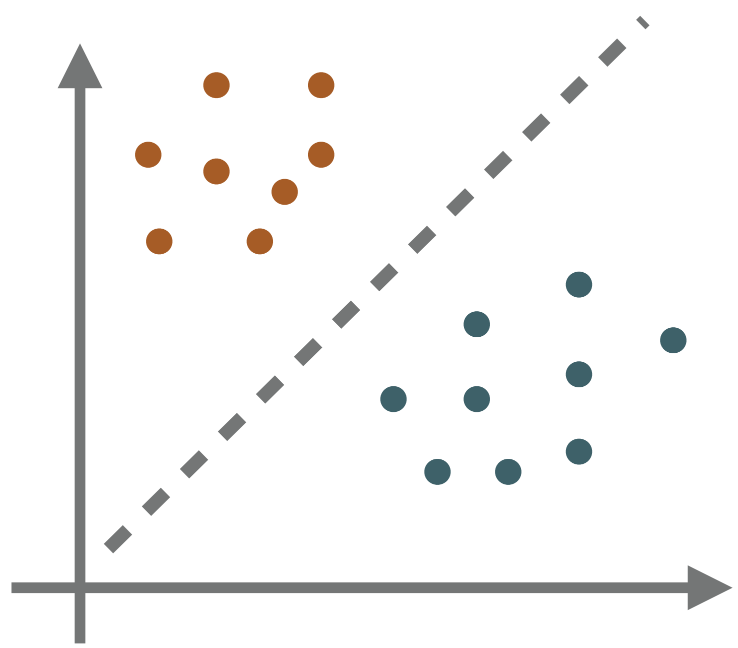
BINARY CLASSIFICATION



PERCEPTRON FOR BINARY CLASSIFICATION

In a nutshell

$$f_\theta(x) = \begin{cases} -1 & \textit{if} \quad a^\top x + b \leq 0, \\ 1 & \textit{if} \quad a^\top x + b > 0. \end{cases}$$



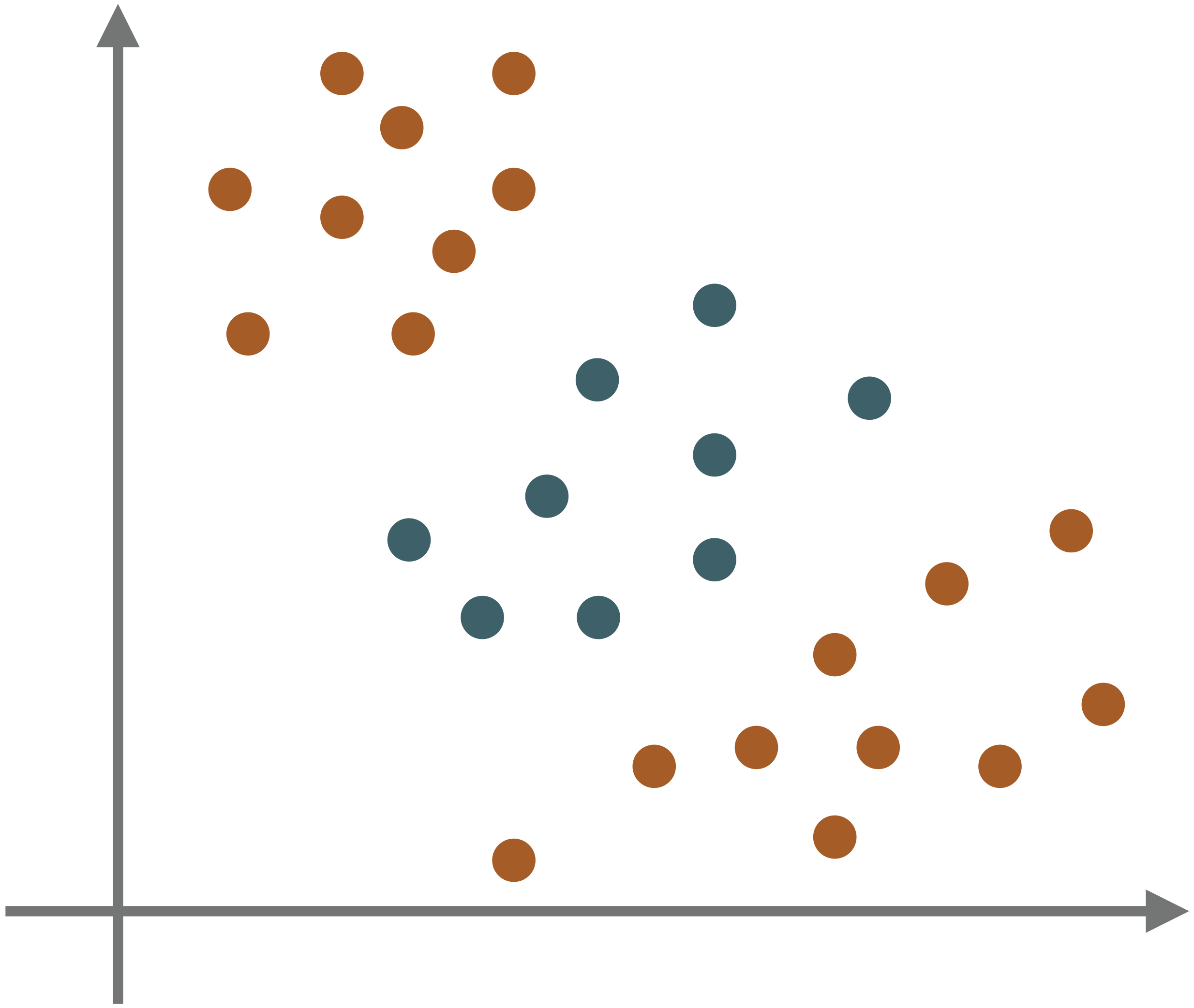
- Parameters: $\theta = \{a, b\}$
- Decision boundary is the set of points that solves:
$$a^\top x + b = 0$$
- The decision boundary is an hyperplane

Remaining questions

- Does an hyperplane that separates data always exists?
- How do we find this hyperplane, i.e. how do we compute a and b?

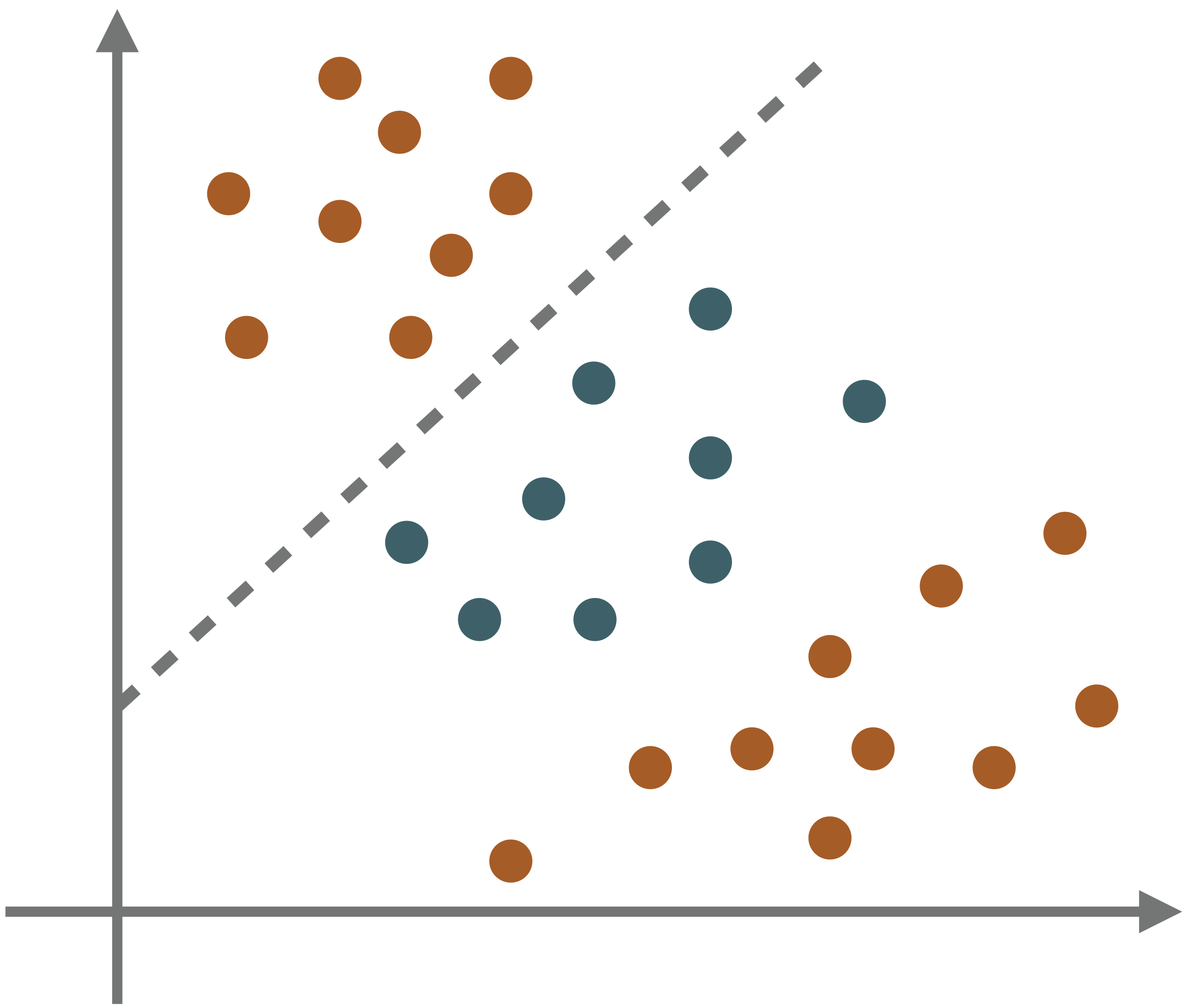
PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



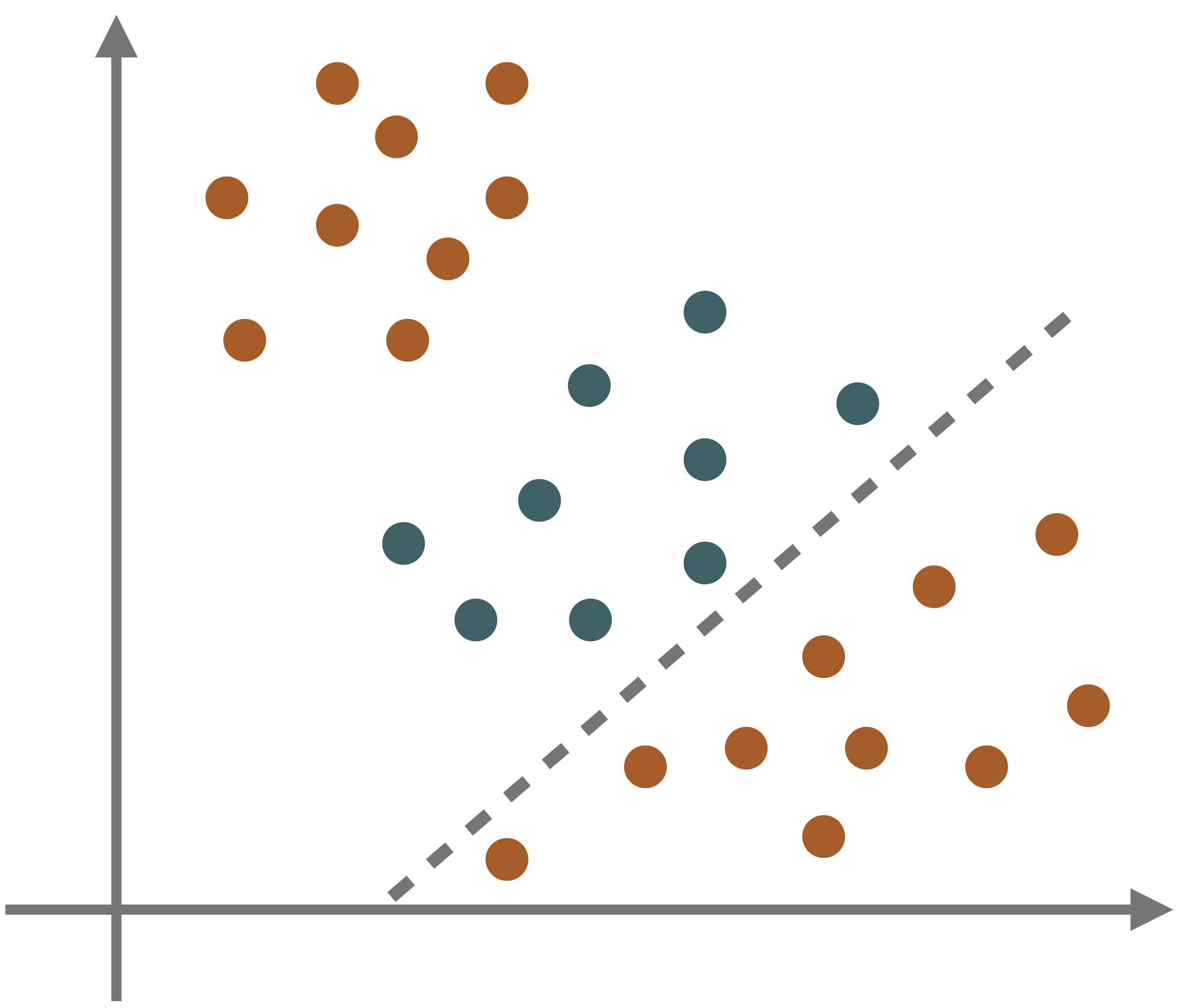
PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



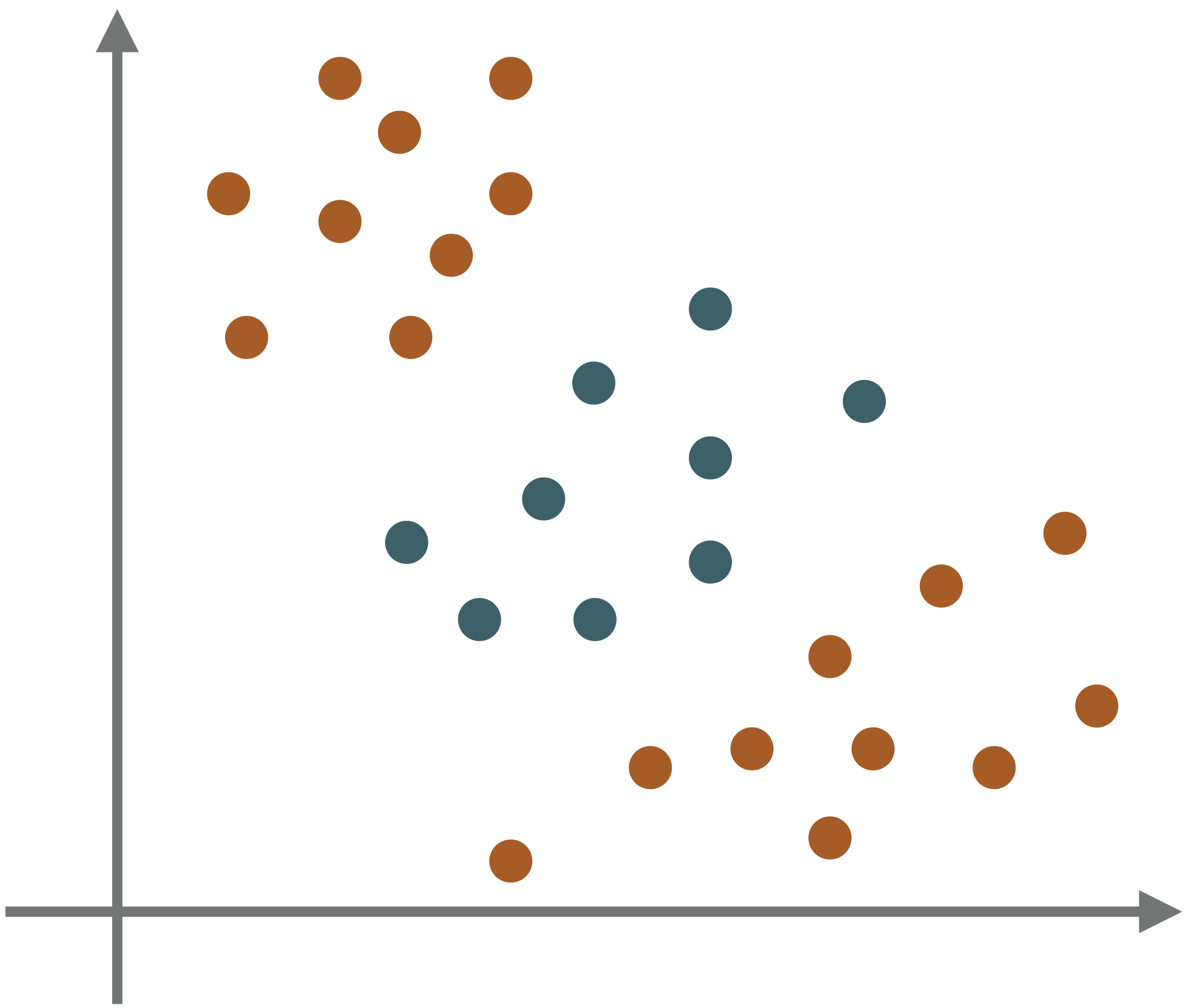
PROBLEMATIC CASES

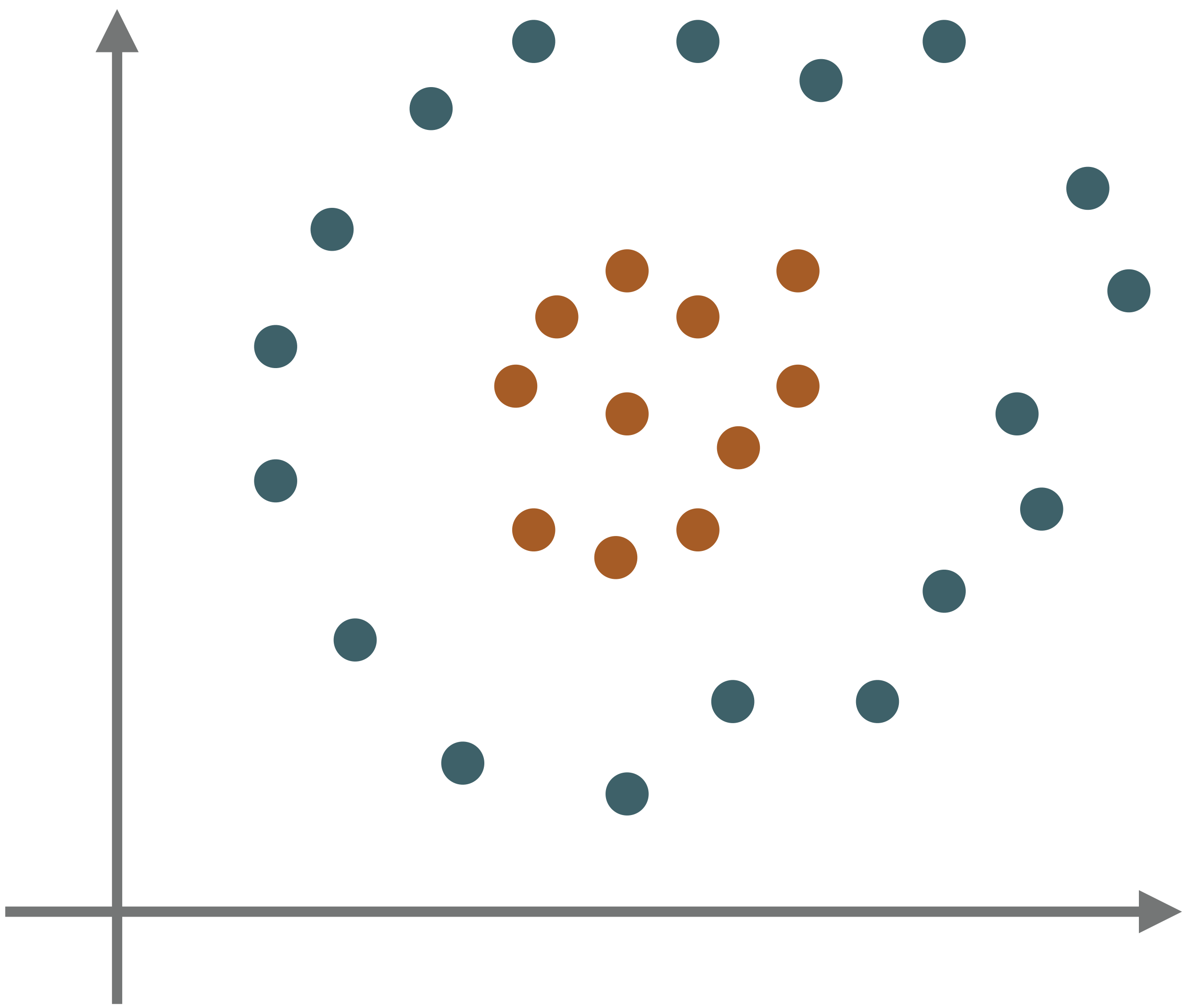
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



PROBLEMATIC CASES

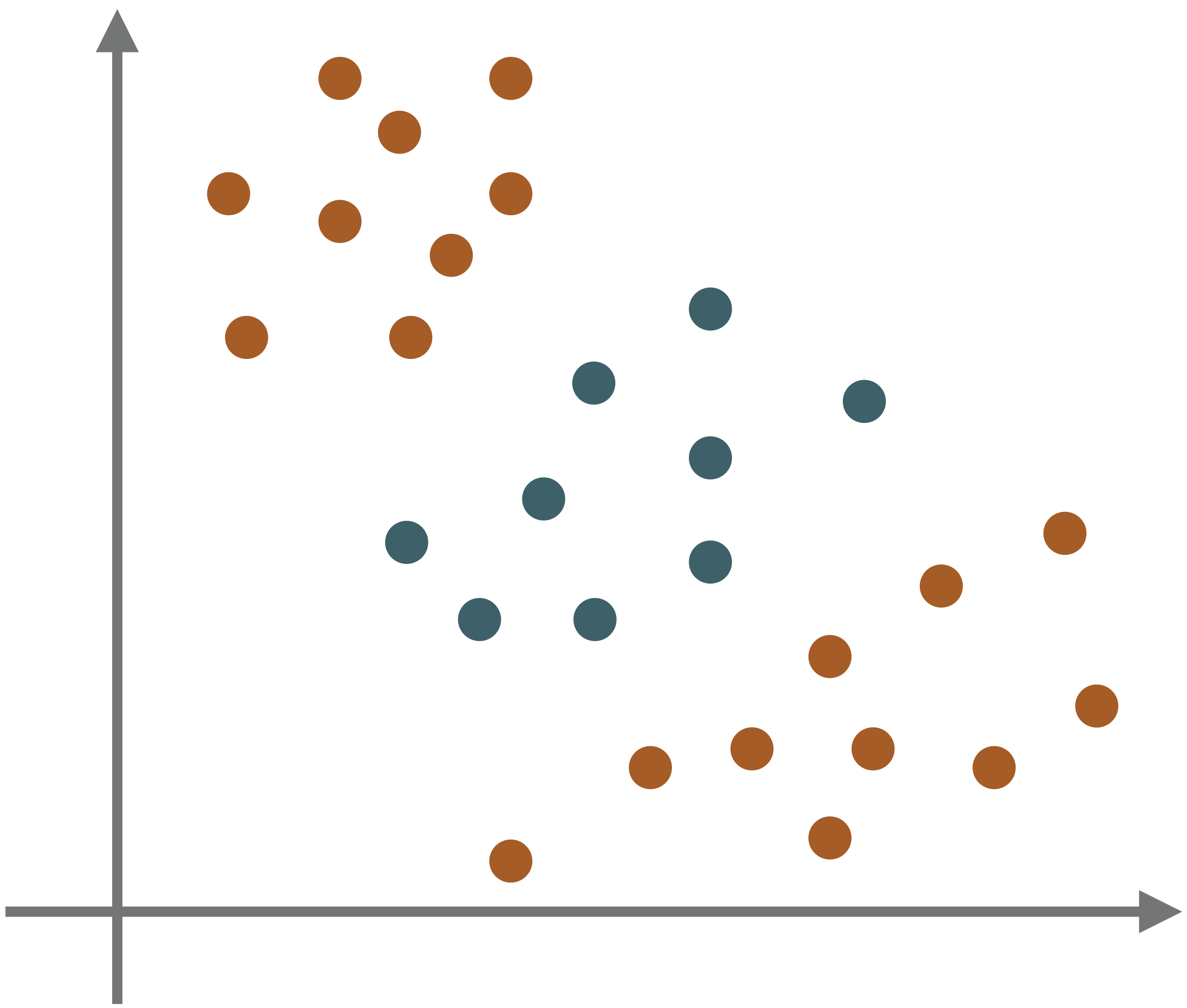
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**

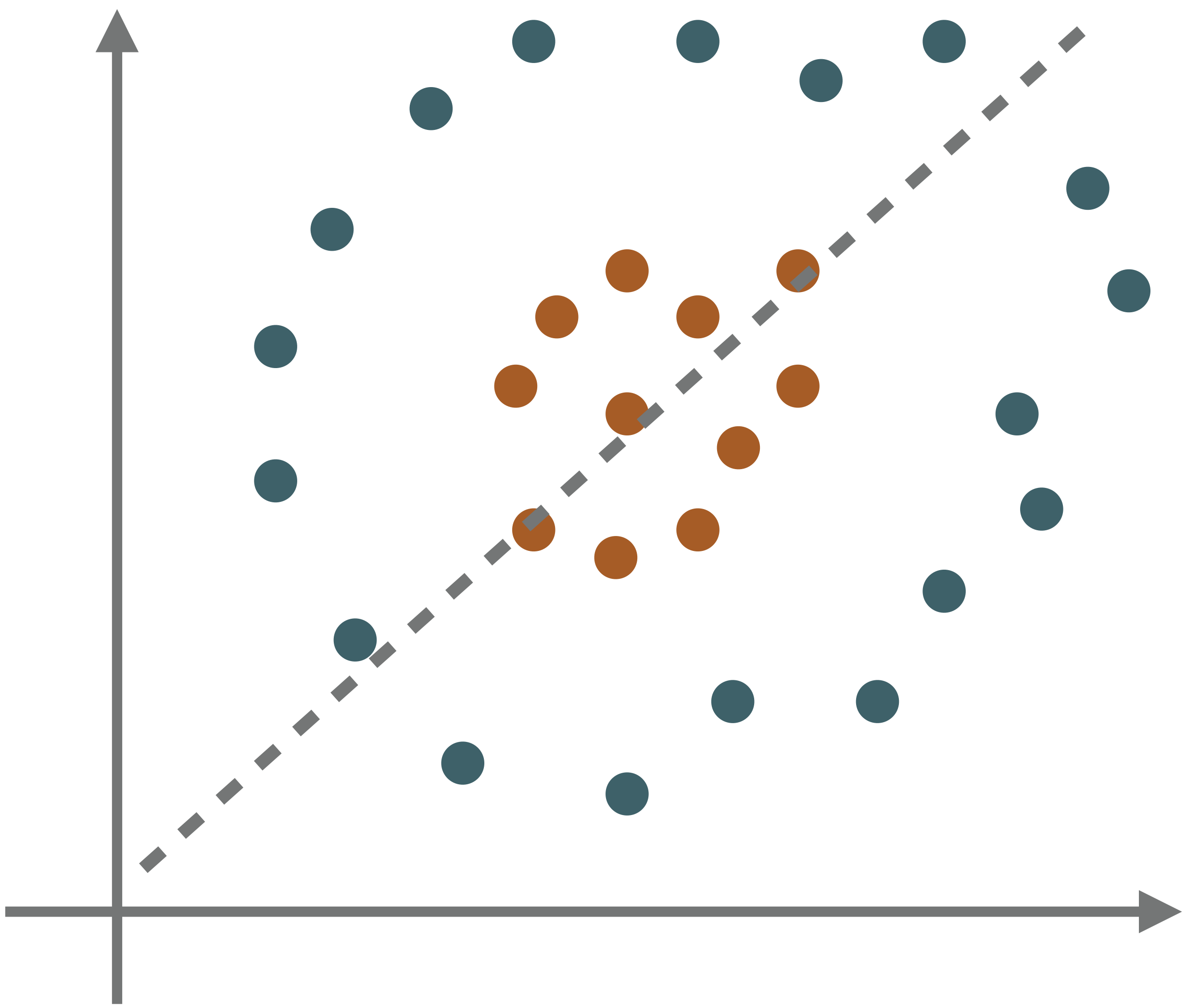




PROBLEMATIC CASES

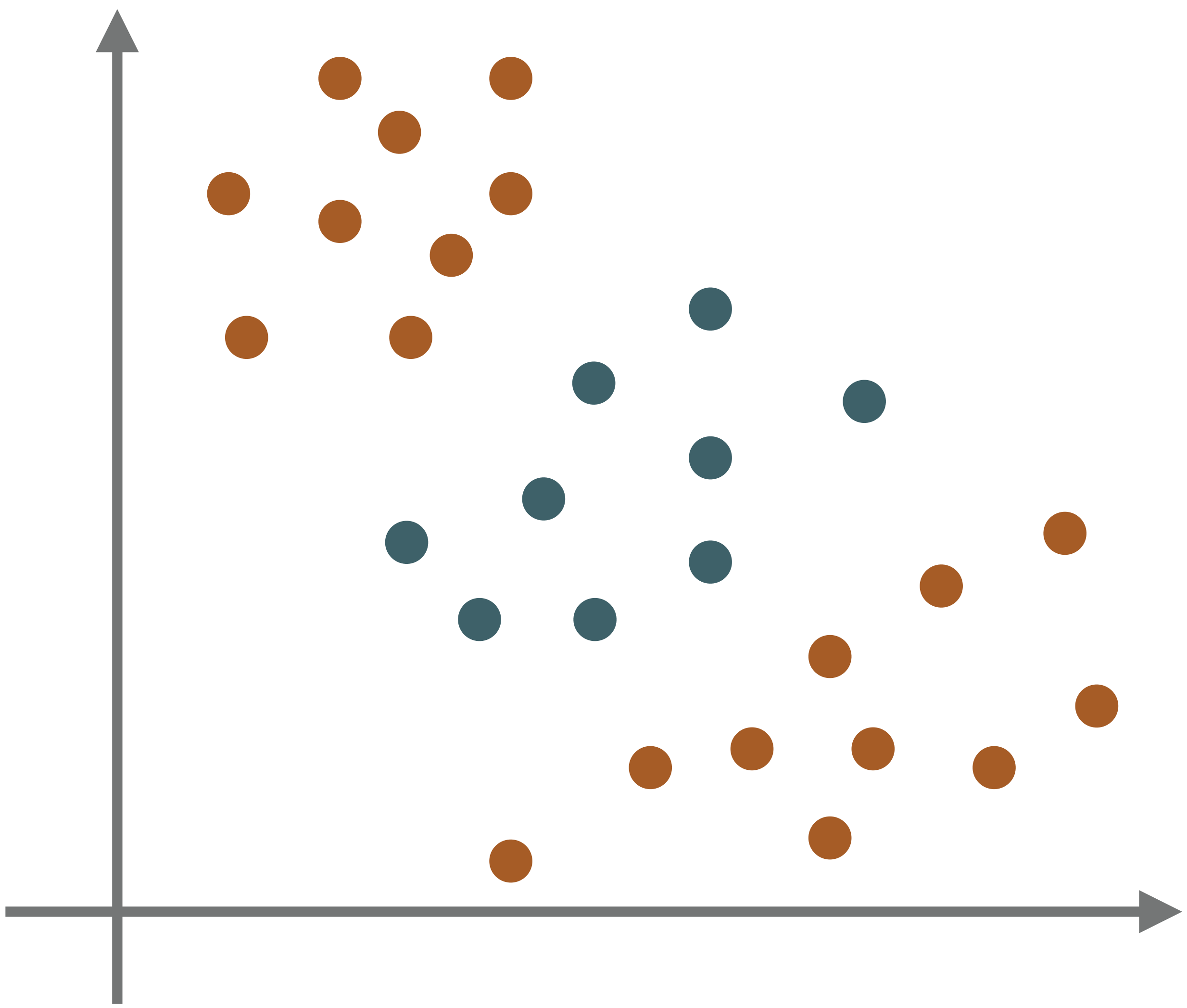
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**

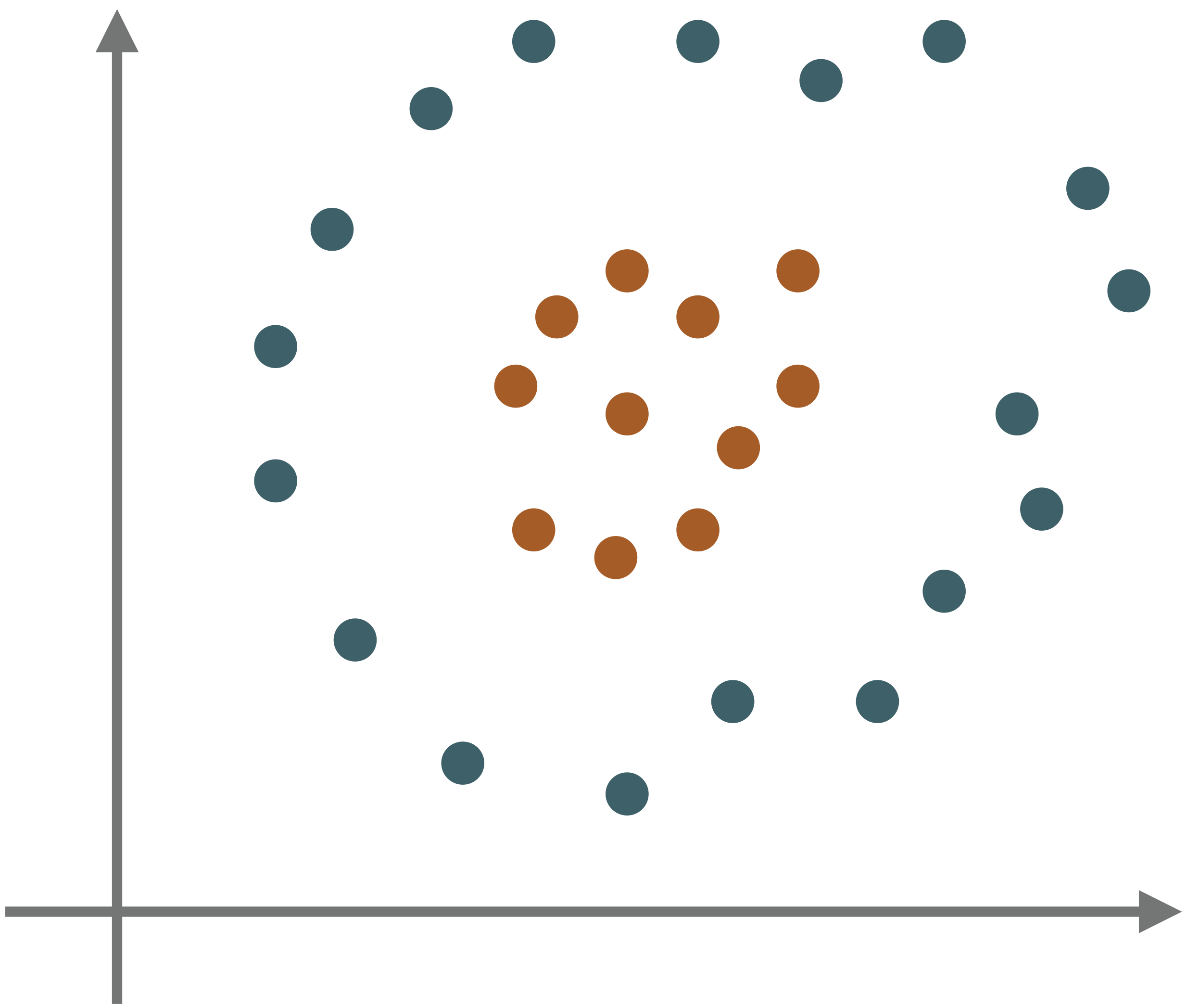




PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**





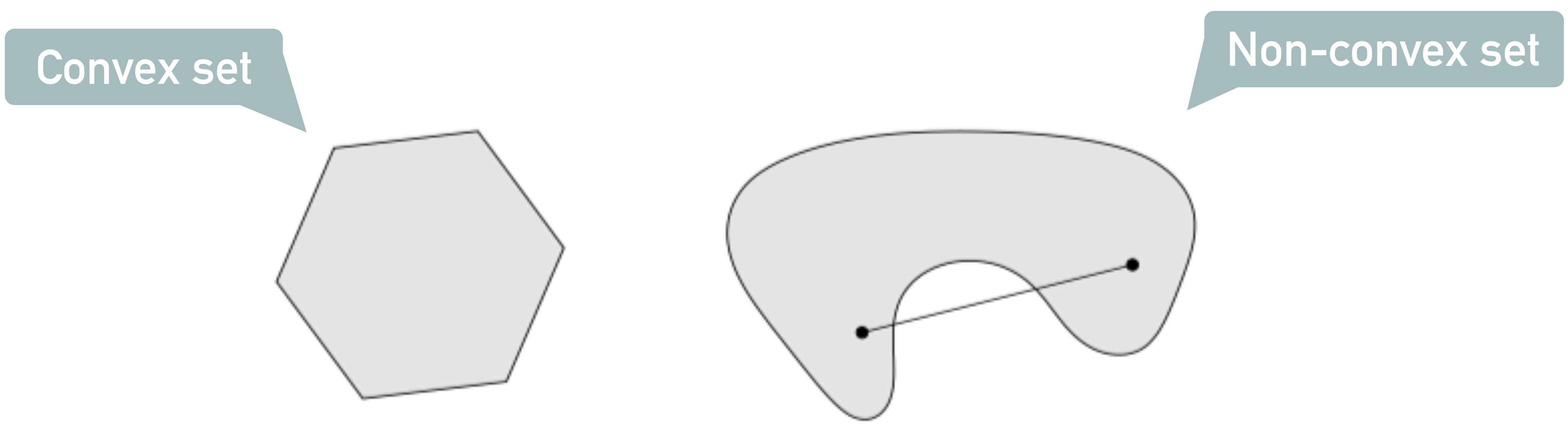
CONVEX SET

Definition

Let $C \in \mathbb{R}^n$ be a set of points. C is convex if and only if:

$$\forall x, y \in C, \epsilon \in [0,1] : \epsilon \times x + (1 - \epsilon) \times y \in C$$

Or, in other words, for every couple of points in C, their convex combination must also be in C.



(Picture from Convex Optimization, Boyd and Vandenberghe)

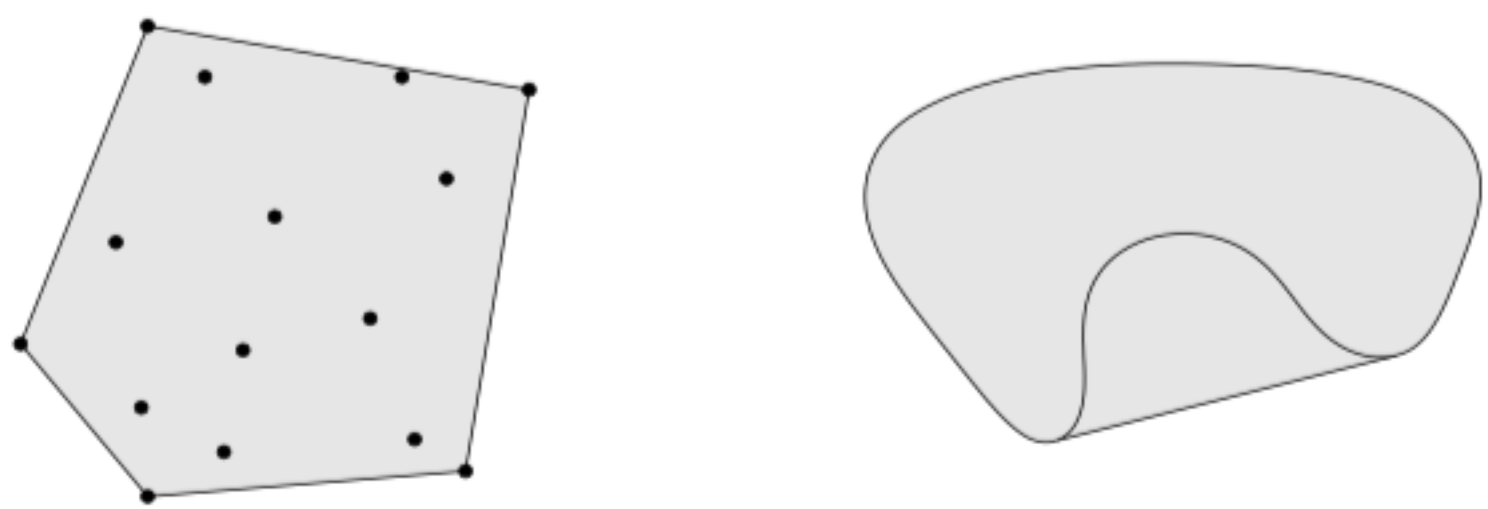
CONVEX HULL

Definition

The **convex hull** of a set $C \in \mathbb{R}^n$ is the set of all convex combinations of points in C:

$$\text{conv}\, C = \{\epsilon_1 x_1 + \ldots + \epsilon_k x_k \quad | \quad \forall i = 1 \ldots k : x_i \in C, \epsilon_i \geq 0, \epsilon_1 + \ldots + \epsilon_k = 1\}$$

Or, in other words, it is the smallest convex set that contains S



(Picture from Convex Optimization, Boyd and Vandenberghe)

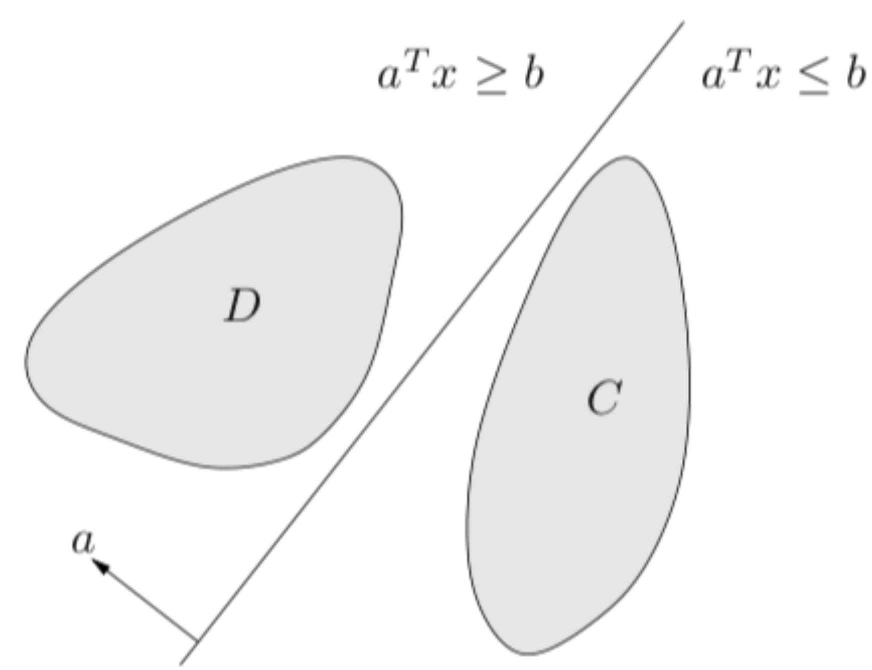
SEPARATING HYPERPLANE

Theorem

Let $C \in \mathbb{R}^n$ and $D \in \mathbb{R}^n$ be two convex sets.
If C and D does not intersect, i.e. $C \cap D = \varnothing$ then there exist a separating hyperplane such that:

$$\forall x \in C: \quad a^\top x + b \geq 0$$
$$\forall x \in D: \quad a^\top x + b \leq 0$$

where w and b parameterize the separation hyperplane.

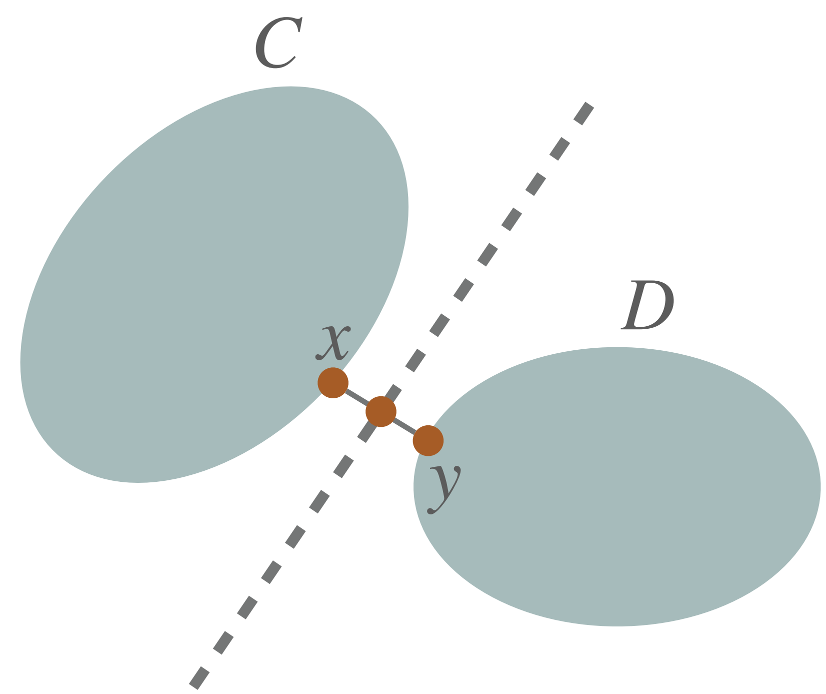


(Picture from Convex Optimization, Boyd and Vandenberghe)

PARAMETER OF THE SEPARATING HYPERPLANE

Closed form solution

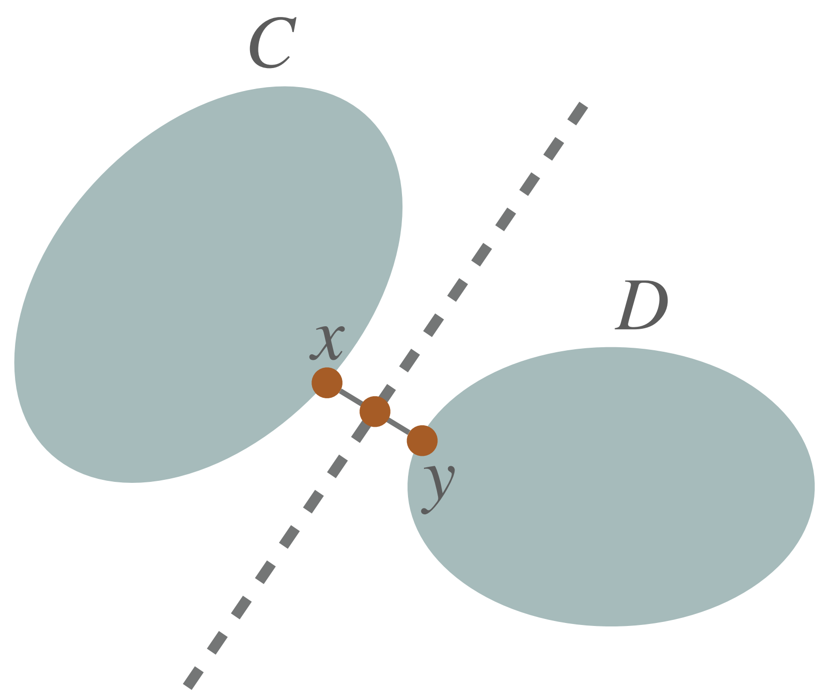
See Convex Optimization (Boyd and Vandenberghe) section 2.5.1.



PARAMETER OF THE SEPARATING HYPERPLANE

Closed form solution

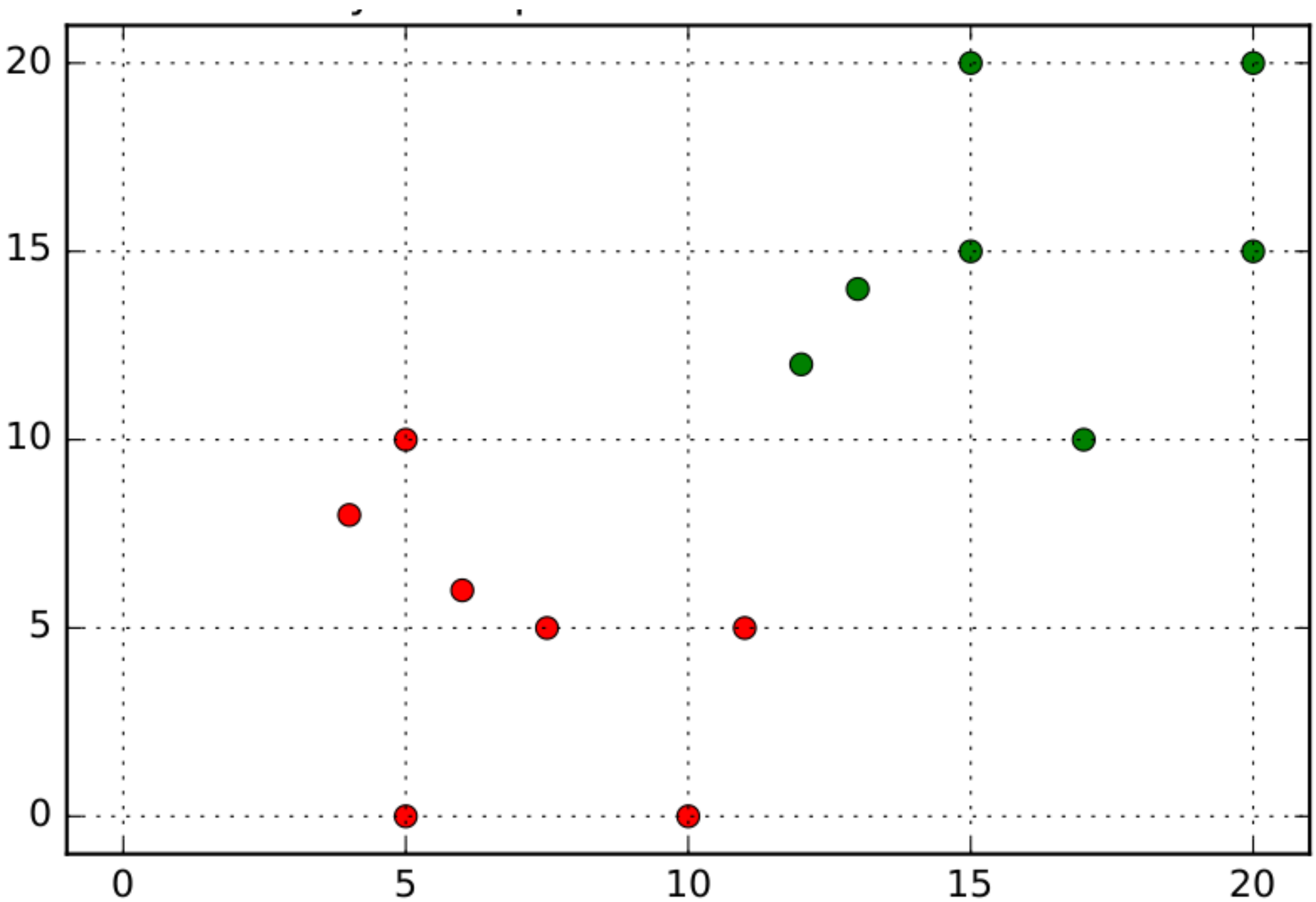
See Convex Optimization (Boyd and Vandenberghe) section 2.5.1.



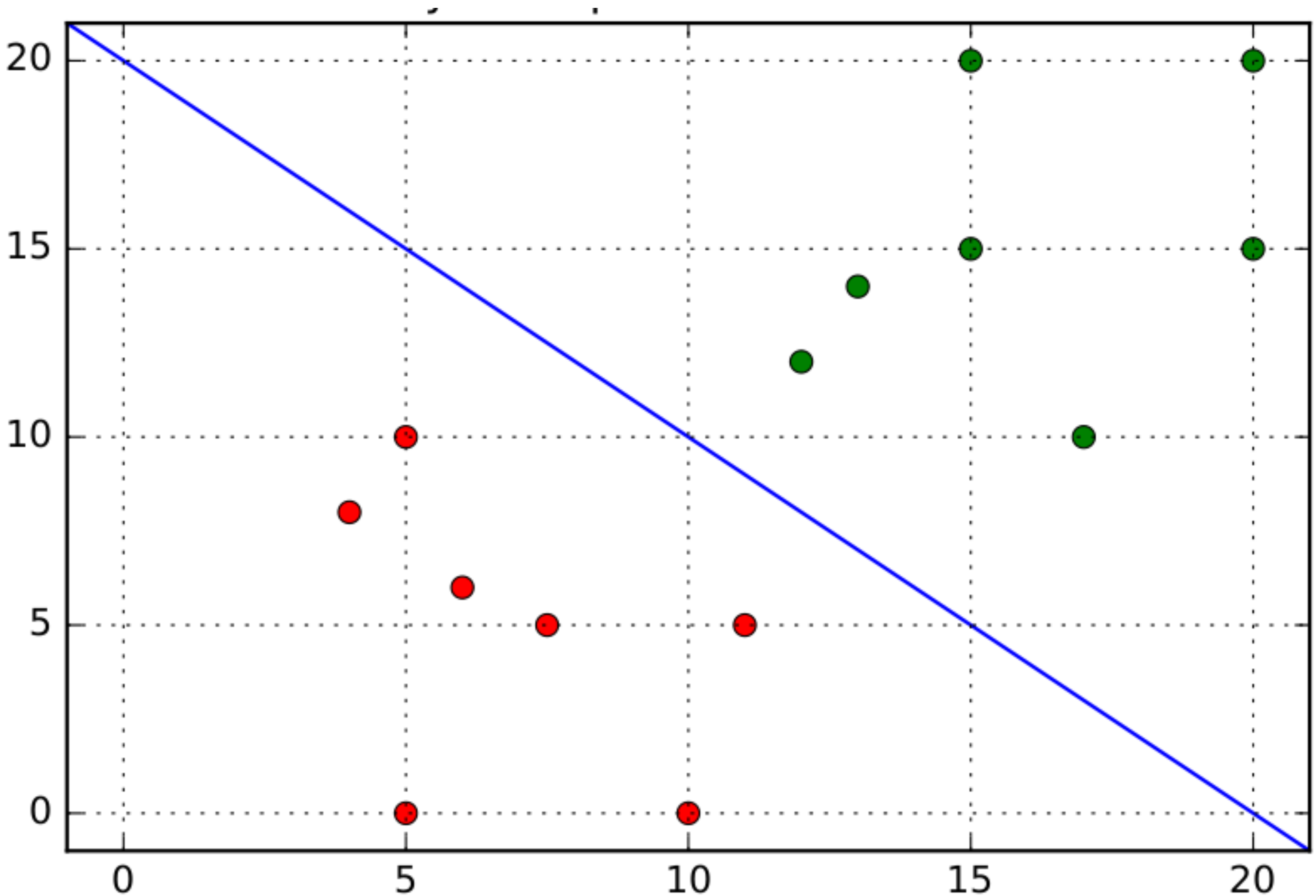
In practice

- Data is not linearly separable (i.e. such a hyperplane does not exists)
- Computing global solutions can be very expensive with big datasets
- Online algorithm are preferable

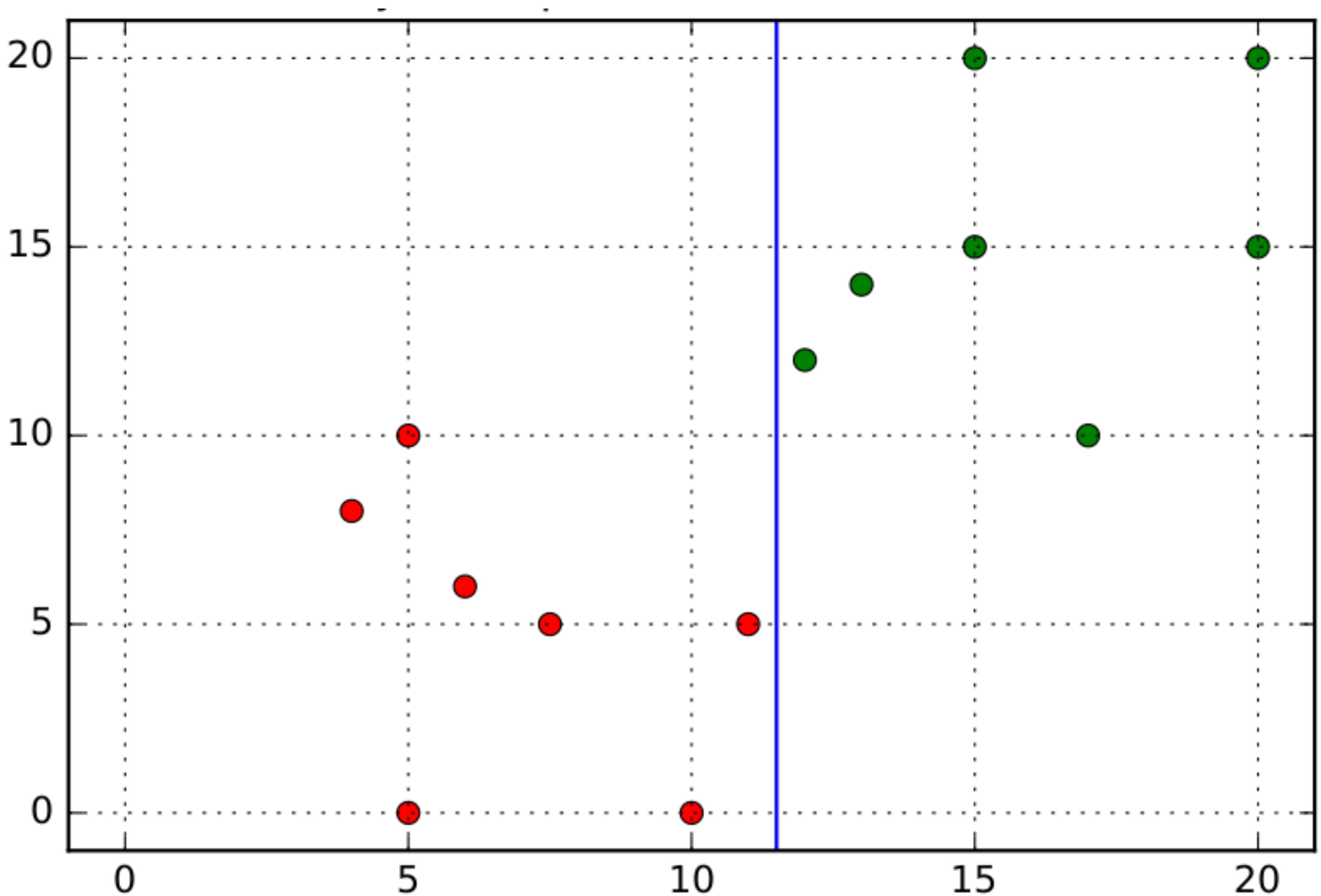
HOW TO SEPARATE THE DATA?



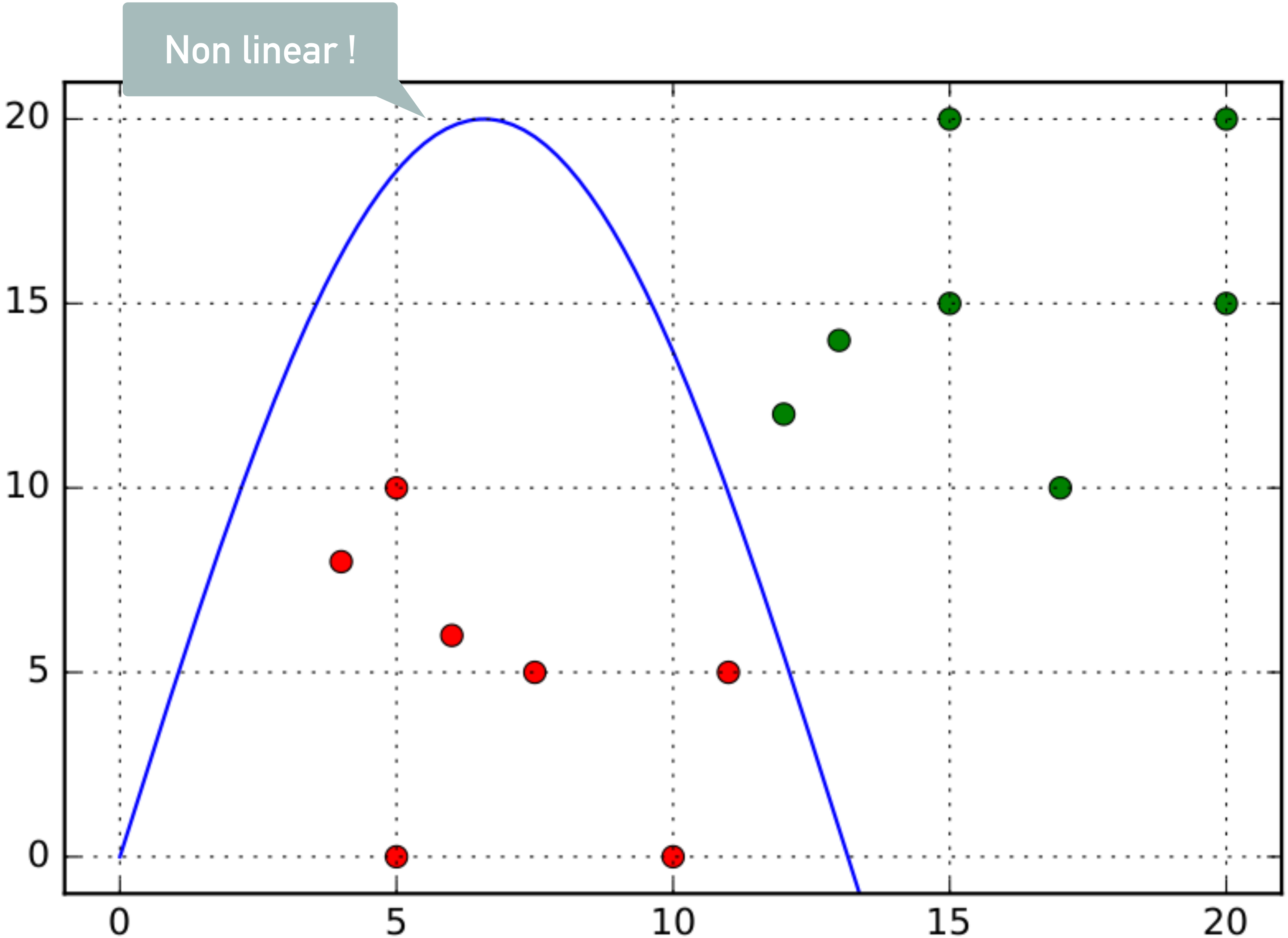
HOW TO SEPARATE THE DATA?



HOW TO SEPARATE THE DATA?



HOW TO SEPARATE THE DATA?



0-1 LOSS FUNCTION

Problem

How to compare the current prediction of the model with the gold output?

0-1 loss function

Function that is equal :

- to 0 if the model gives the correct prediction
- to 1 otherwise

Check if the score is of the same sign as the gold output

If $Y = \{0,1\}$:

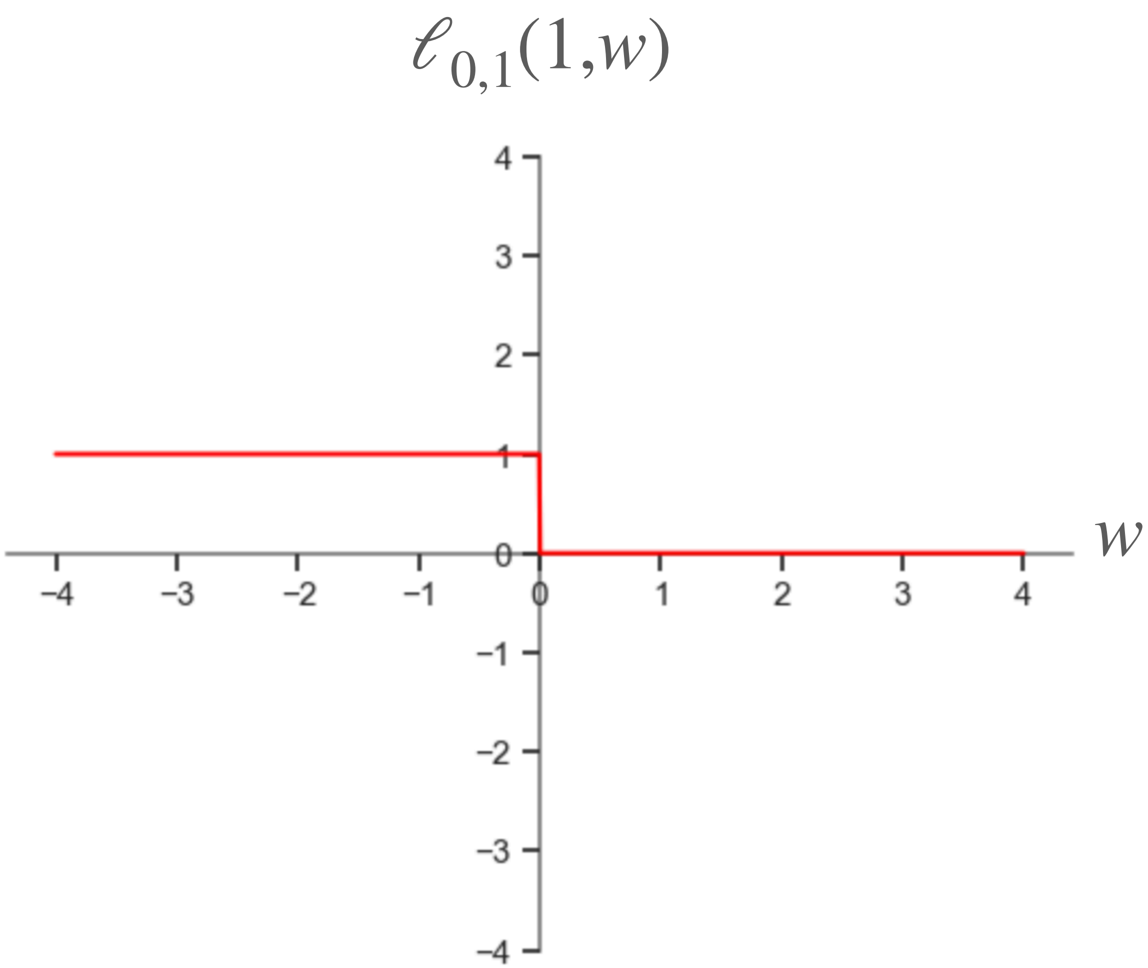
$$\ell_{0-1}(y, w) = \begin{cases} 0 & \textit{if } (2y-1)w \geq 0, \\ 1 & \textit{otherwise}. \end{cases}$$

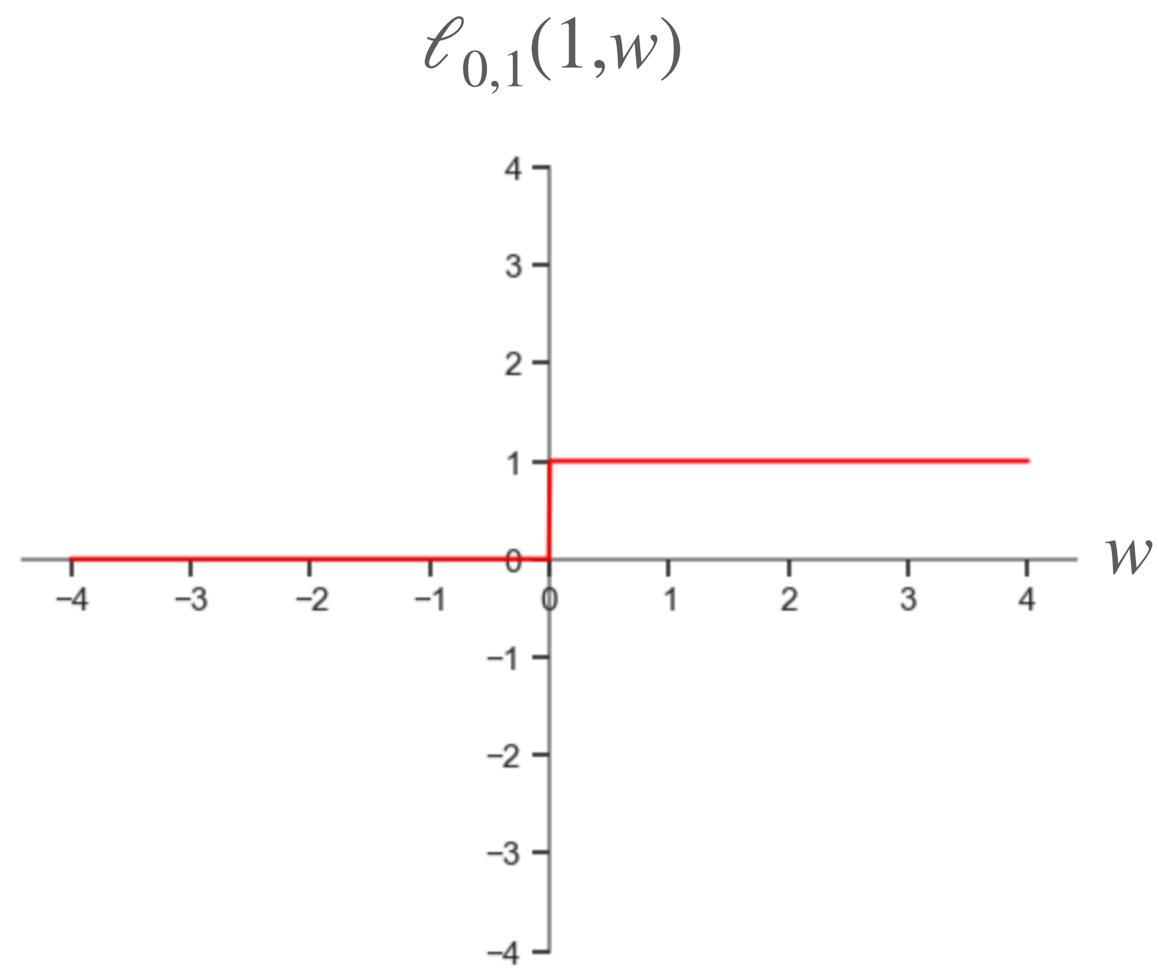
Input the score!

If $Y = \{-1,1\}$:

$$\ell_{0-1}(y, w) = \begin{cases} 0 & \textit{if } yw \geq 0, \\ 1 & \textit{otherwise}. \end{cases}$$

0-1 LOSS FUNCTION





Problems

- Non convex function
- Derivative is null almost everywhere

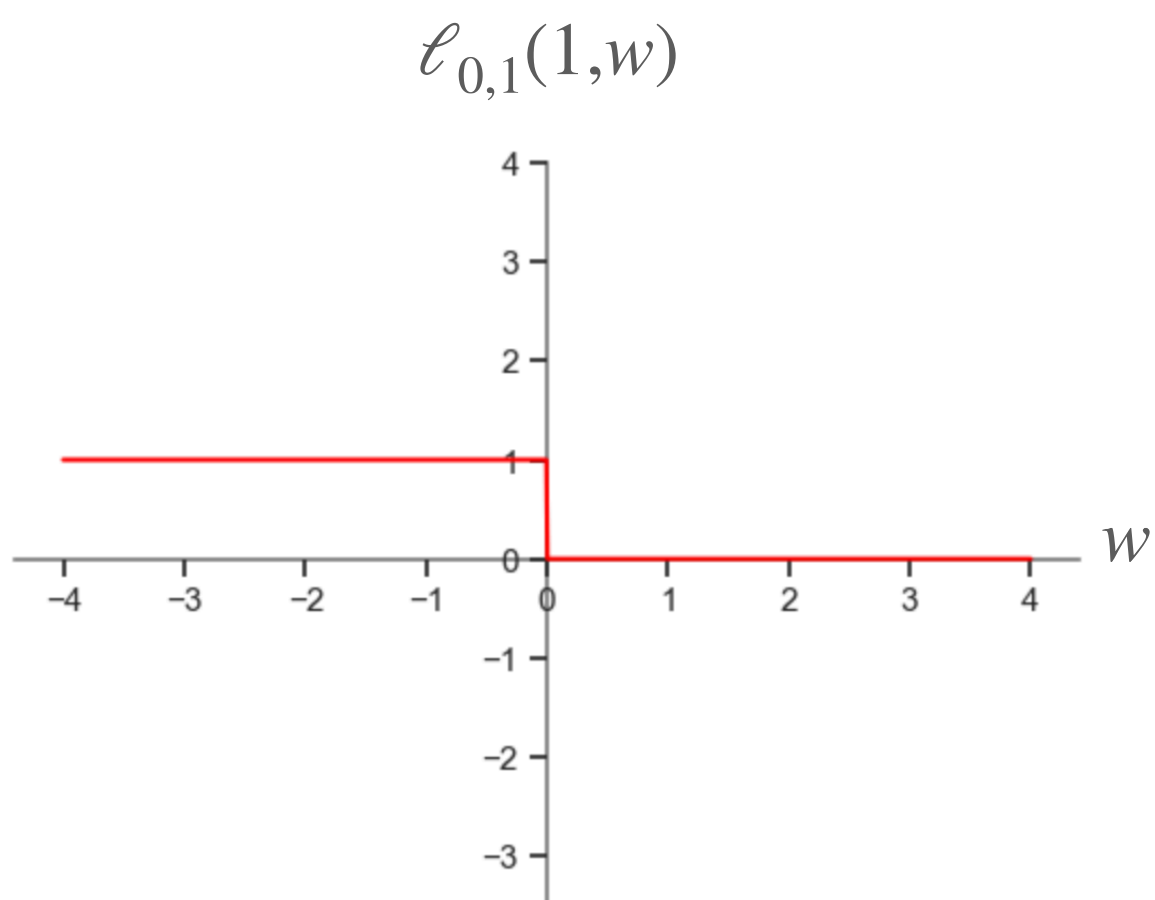
SURROGATE LOSSES

Main idea

Replace the 0-1 loss by a surrogate such that the surrogate:

- is convex
- is an upper bound on the 0-1 loss
- has non null derivatives when the prediction is wrong

Minimizing the surrogate loss should "implicitly" minimize the 0-1 loss.



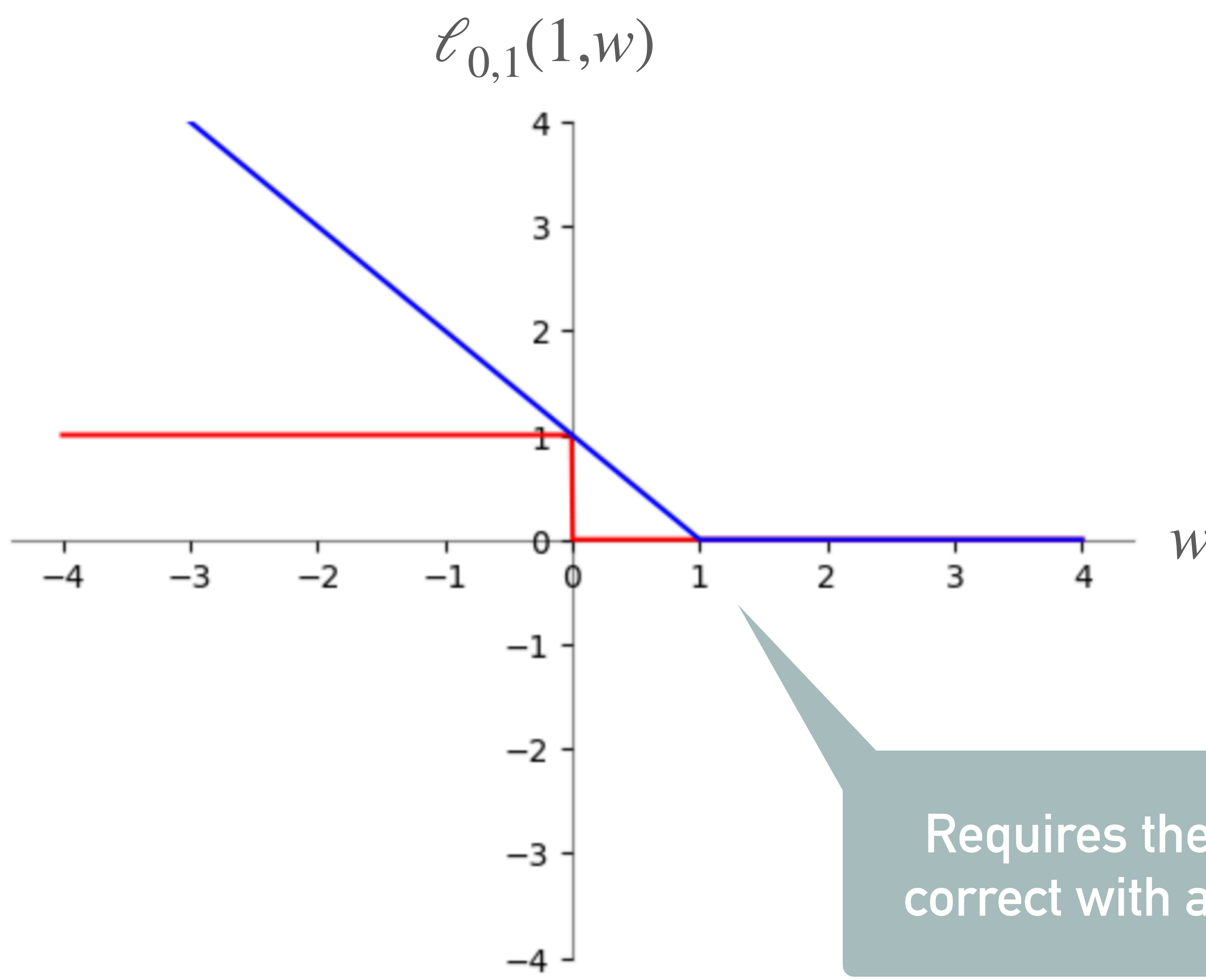
HINGE LOSS

$If\ Y = \{0,1\}:$

$$\ell_{hinge}(y, w) = \max(0, 1 - (2y - 1) \times w)$$

$If\ Y = \{-1,1\}:$

$$\ell_{hinge}(y, w) = \max(0, 1 - y \times w)$$



Requires the prediction is correct with a "margin" of 1

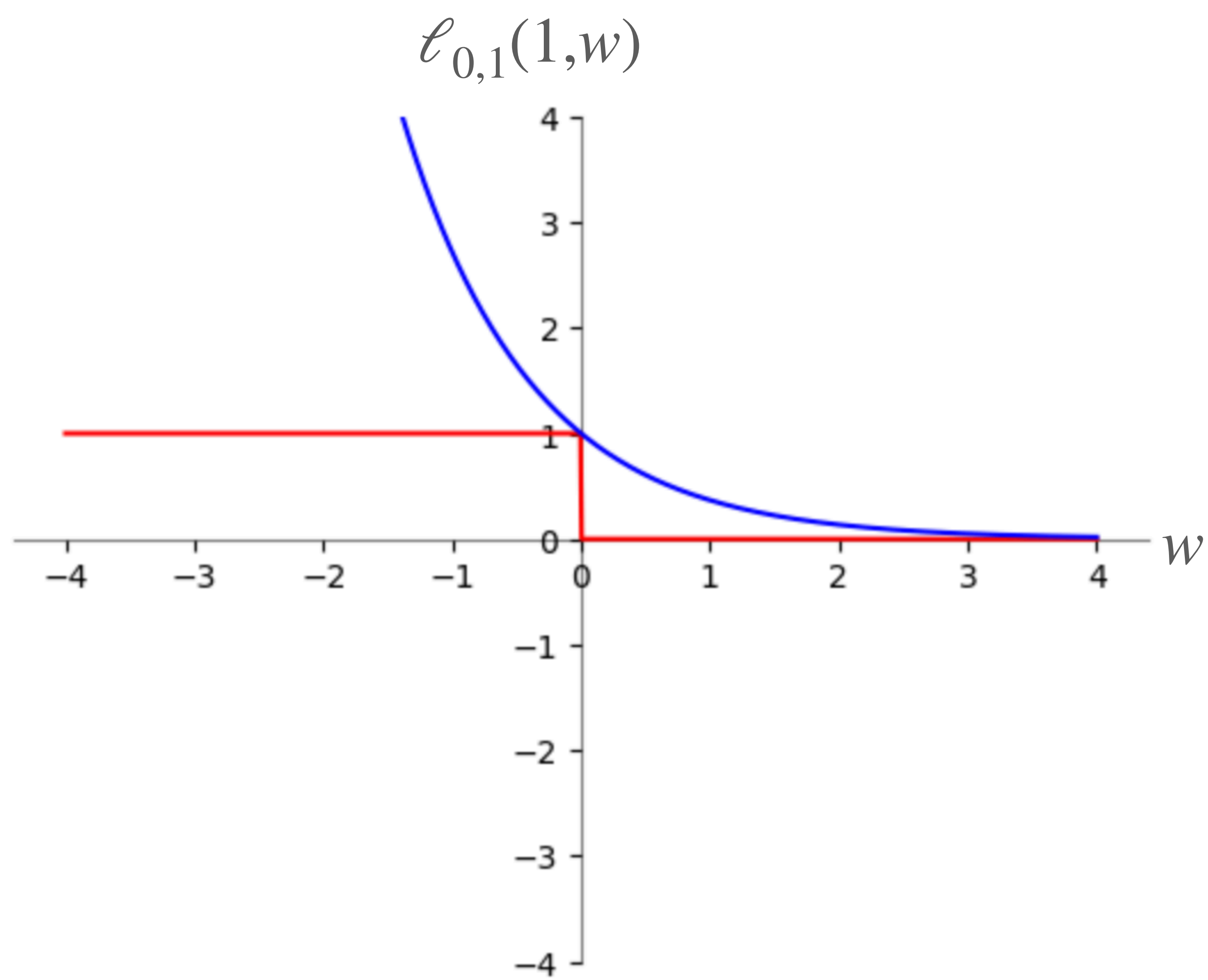
EXPONENTIAL LOSS

If $Y = \{0,1\}$:

$\ell_{exp}(y, w) = \exp(-(2y-1) \times w)$

If $Y = \{-1,1\}$:

$\ell_{exp}(y, w) = \exp(-y \times w)$



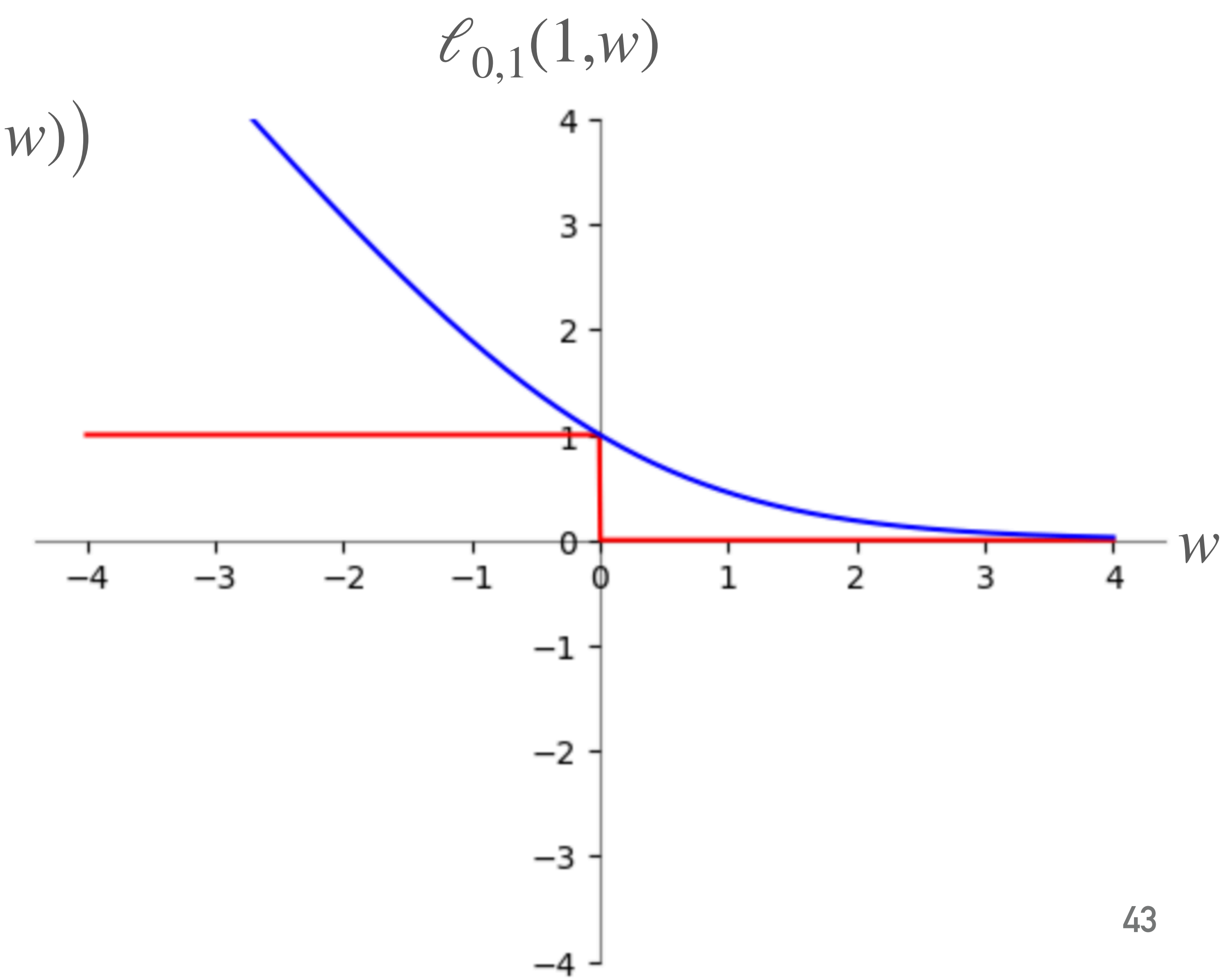
NEGATIVE LOG-LIKELIHOOD

$If\ Y = \{0,1\}$:

$$\ell_{nll}(y, w) = \frac{1}{\log 2} \log\big(1 + \exp(-(2y-1) \times w)\big)$$

$If\ Y = \{-1,1\}$:

$$\ell_{nll}(y, w) = \frac{1}{\log 2} \log\big(1 + \exp(-y \times w)\big)$$



PROBABILISTIC PREDICTION

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Parameters: $\theta = \{\mathbf{a}, b\}$ avec $\mathbf{a} \in \mathbb{R}^d, \; b \in \mathbb{R}$
- Output: $y \in \{0,1\}$ ou $y \in \{-1,1\}$

Scoring function

Compute a score associated with an input (this function is parameterized)

$$s_\theta(\mathbf{x}) = s(\mathbf{x}; \theta) = \langle \mathbf{a}, \mathbf{x} \rangle + b$$

Prediction function

Compute the output associated with a score (this function is not parameterized)

$$\hat{\mathbf{y}}(w) = \begin{cases} 1 & \textit{if } w \geq 0, \\ 0 & \textit{otherwise.} \end{cases}$$

What if we want to model the uncertainty?

PROBABILISTIC PREDICTION

Bernoulli distribution

Distribution over $\{0,1\}$ parameterized by $\mu \in [0,1]$

$$p(z=1) = \mu \qquad p(z=0) = 1-\mu \qquad p(z) = \mu^z(1-\mu)^{1-z}$$

PROBABILISTIC PREDICTION

Bernoulli distribution

Distribution over $\{0,1\}$ parameterized by $\mu \in [0,1]$

$$p(z=1) = \mu \qquad p(z=0) = 1-\mu \qquad p(z) = \mu^z(1-\mu)^{1-z}$$

Probabilistic prediction function

Compute the parameter of a Bernoulli distribution over outputs.
We usually rely on the sigmoid function

$$\hat{\mathbf{y}}(w) = \sigma(w) = \frac{\exp(w)}{1+\exp(w)} = \frac{1}{1+\exp(-w)}$$

PROBABILISTIC PREDICTION

Bernoulli distribution

Distribution over $\{0,1\}$ parameterized by $\mu \in [0,1]$

$$p(z=1)=\mu \qquad p(z=0)=1-\mu \qquad p(z)=\mu^z(1-\mu)^{1-z}$$

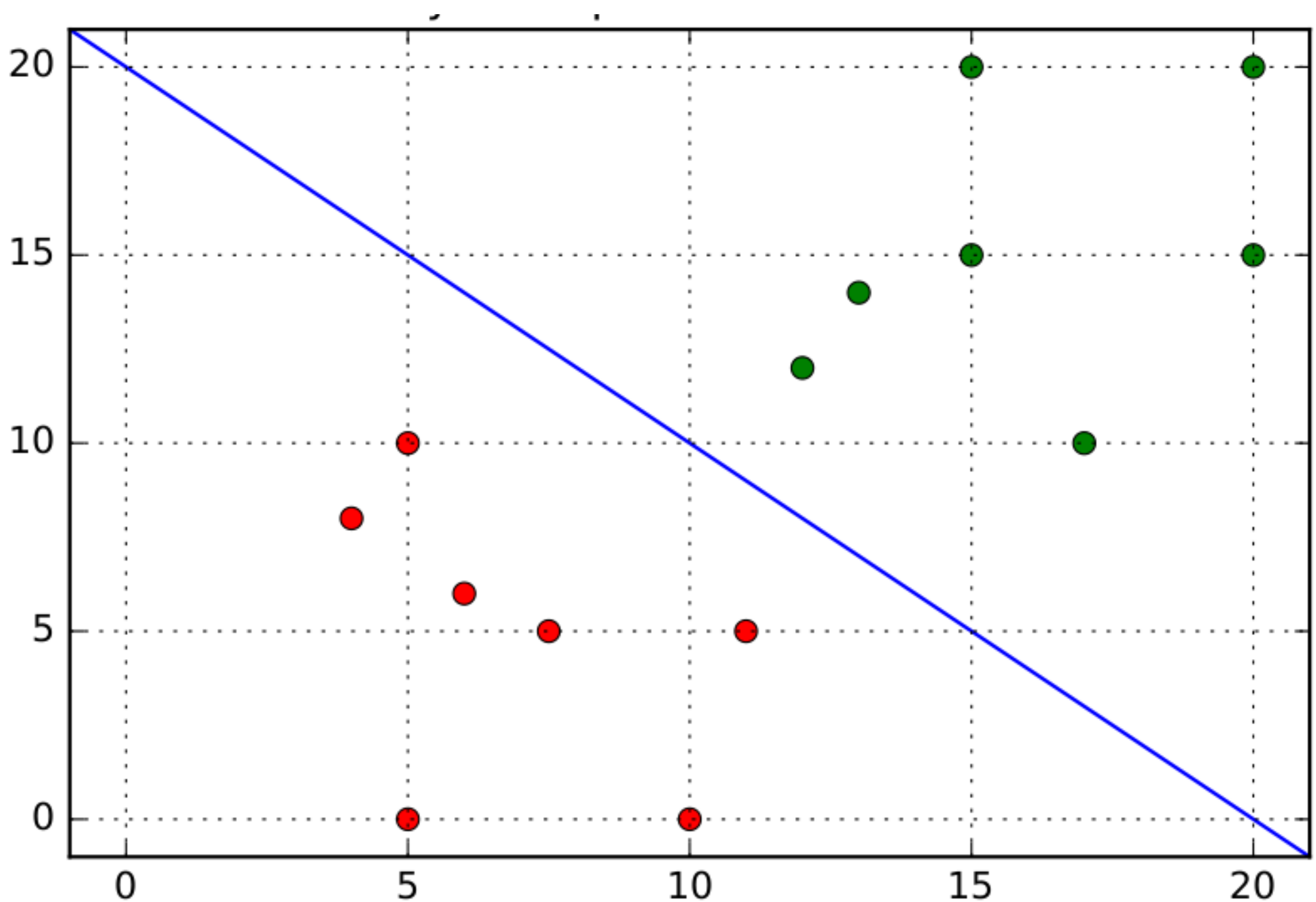
Probabilistic prediction function

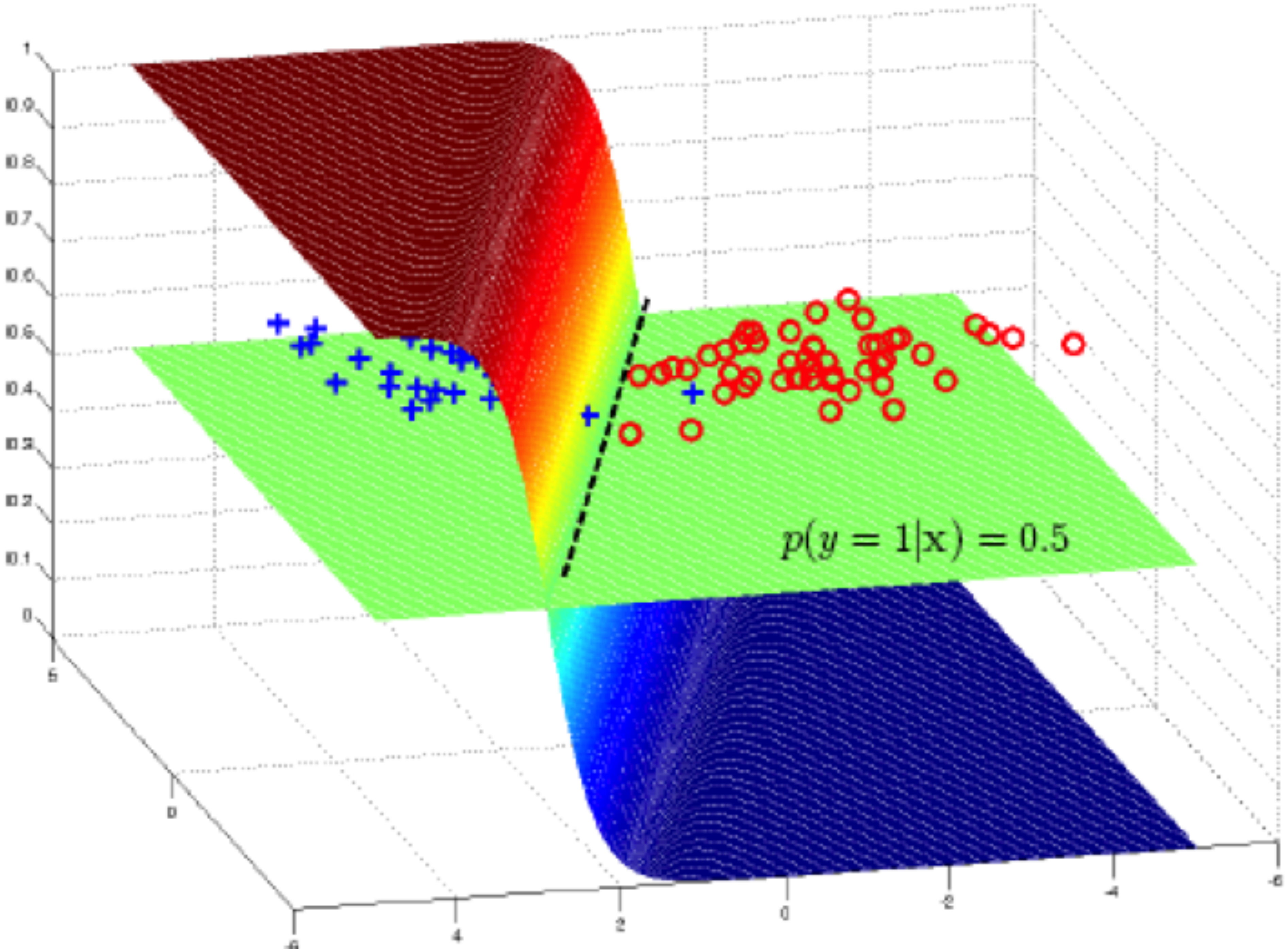
Compute the parameter of a Bernoulli distribution over outputs.
We usually rely on the sigmoid function

$$\hat{\mathbf{y}}(w)=\sigma(w)=\frac{\exp(w)}{1+\exp(w)}=\frac{1}{1+\exp(-w)}$$

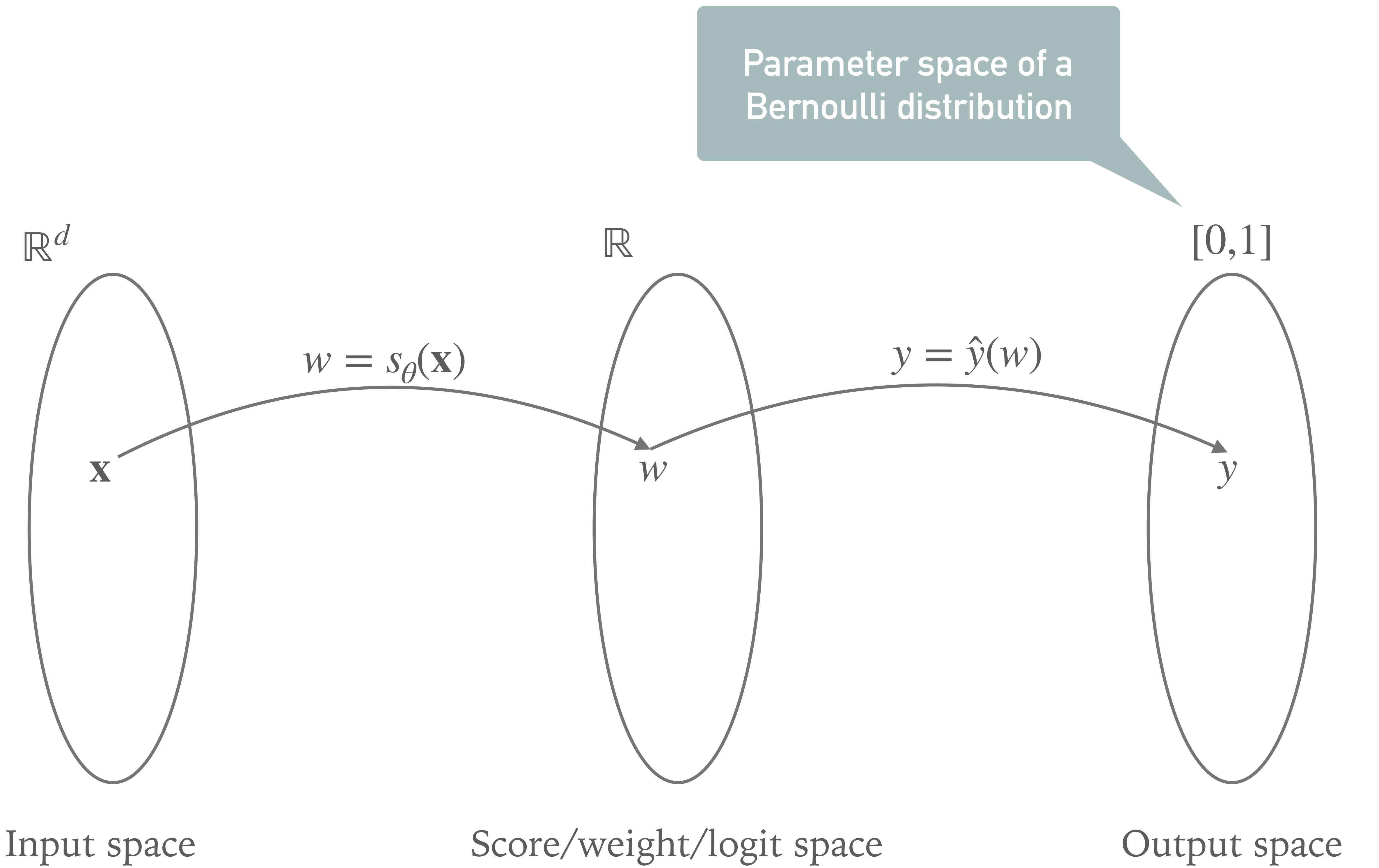
Complete model

$$p(y=1\,|\,x)=\frac{\exp(\langle \mathbf{a},\mathbf{x}\rangle+b)}{1+\exp(\langle \mathbf{a},\mathbf{x}\rangle+b)} \qquad p(y=0\,|\,x)=1-\frac{\exp(\langle \mathbf{a},\mathbf{x}\rangle+b)}{1+\exp(\langle \mathbf{a},\mathbf{x}\rangle+b)}$$


0
5
10
15
20
0
5
10
15
20


$p(y=1|\mathrm{x}) = 0.5$

PROBABILISTIC BINARY CLASSIFICATION



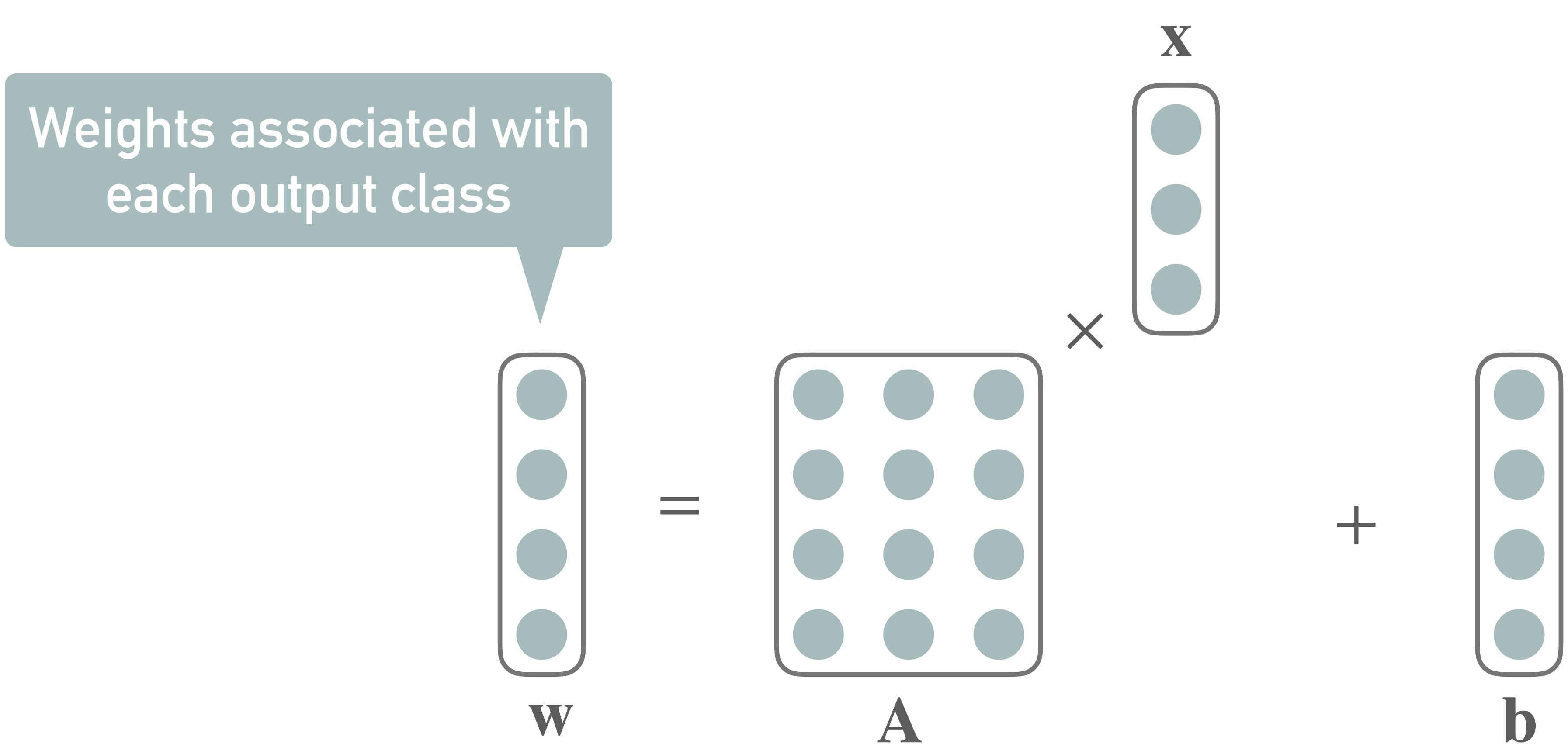
TRAINING A PROBABILISTIC MODEL

What loss should we use to train a probabilistic model?

- Negative log-likelihood! (en TD)

MULTICLASS CLASSIFICATION

MULTICLASS CLASSIFICATION

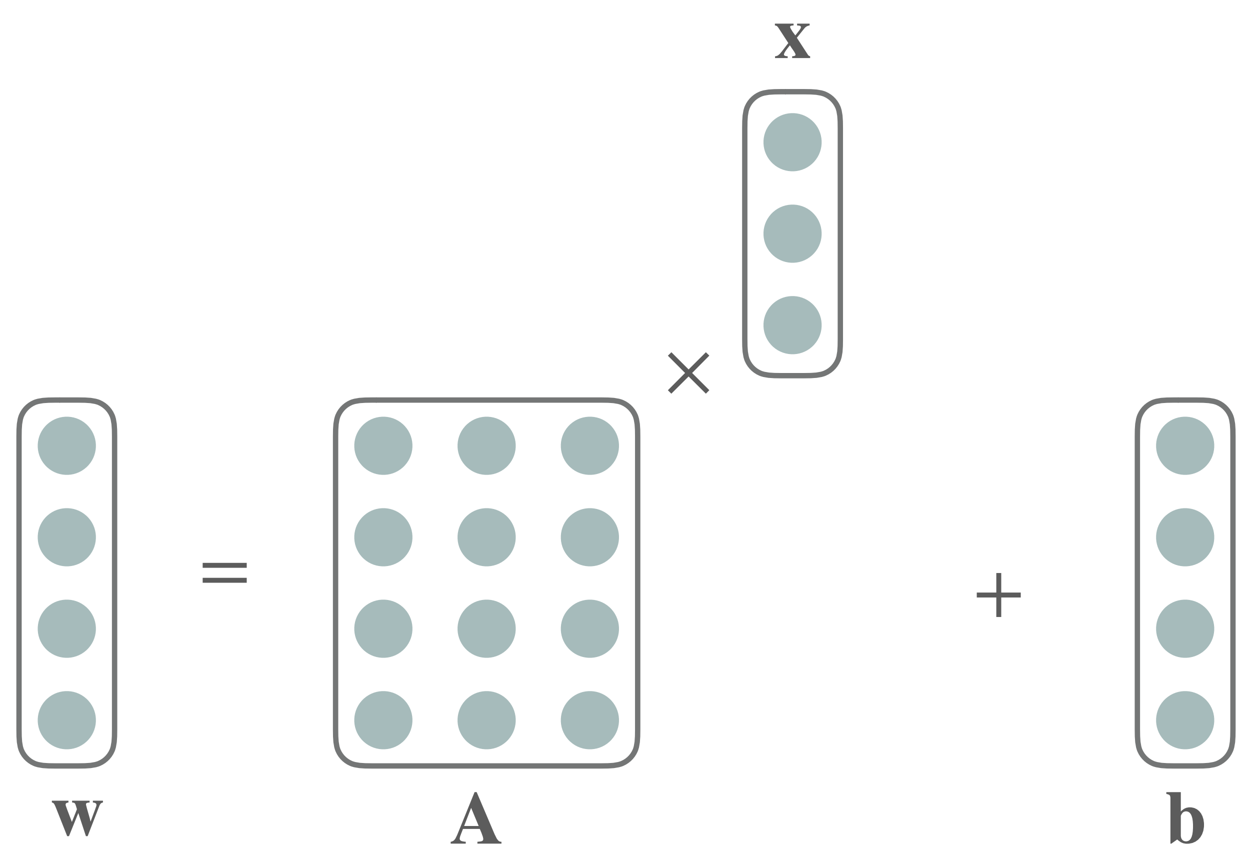


Prediction functions

E(k) is the set of one-hot vector of dim. k

- Integer output : $\hat{\mathbf{y}}(\mathbf{w}) = \mathrm{argmax}_{i \in \{1,\ldots,k\}} \; w_i$
- One-hot vector output : $\hat{\mathbf{y}}(\mathbf{w}) = \mathrm{argmax}_{\mathbf{y} \in E(k)} \; \langle \, \mathbf{y}, \mathbf{w} \, \rangle$
- Probabilistic output (i.e. distribution over classes) : $\hat{\mathbf{y}}(\mathbf{w}) = \mathrm{softmax}(\mathbf{w})$

MULTICLASS CLASSIFICATION



Loss functions

- hinge loss (m >= 0 is the margin) : $\ell(\mathbf{y}, \mathbf{w}) = \max(0,\ m - \langle \mathbf{y}, \mathbf{w} \rangle + \max_{\mathbf{y}' \in E(k) \setminus \{\mathbf{y}\}} \langle \mathbf{y}', \mathbf{w} \rangle$
- Negative log-likelihood :
  (also called cross-entropy) $\ell(\mathbf{y}, \mathbf{w}) = - \langle\ \mathbf{y}, \mathbf{w}\ \rangle + \log \sum_i \exp w_i$