Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2024/2025

Prof. Luca Iocchi

7. Linear models for classification

Luca Iocchi

Overview

- Linearly separable data
- Linear models
- Least squares
- Perceptron
- Fisher's linear discriminant
- Support Vector Machines

References

C. Bishop. Pattern Recognition and Machine Learning. Sect. 4.1, 7.1

T. Mitchell. Machine Learning. Section 4.4

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Linear Models for Classification

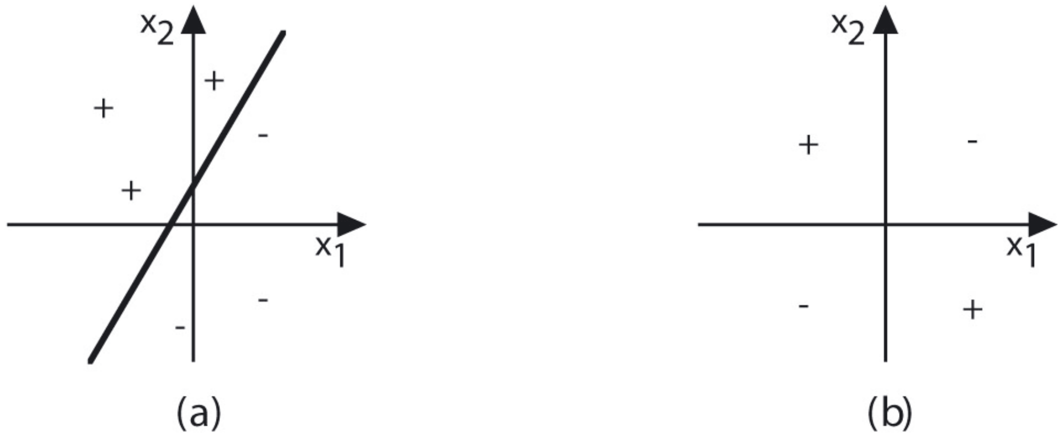
Learning a function $f : X \to Y$, with ...

- $X \subseteq \Re^d$
- $Y = \{C_1, \ldots, C_k\}$

assuming *linearly separable* data.

Linearly separable data

Instances in a data set are *linearly separable* iff there exists a hyperplane that separates the instance space into two regions, such that differently classified instances are separated



Linear discriminant functions

Linear discriminant function

$$y : X \to \{C_1, \ldots, C_K\}$$

Two classes:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

K-classes:

$$y_1(\mathbf{x}) = \mathbf{w}_1^T \mathbf{x} + w_{10}$$
$$\ldots$$
$$y_K(\mathbf{x}) = \mathbf{w}_K^T \mathbf{x} + w_{K0}$$

Compact notation

Two classes:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}, \text{ with:}$$

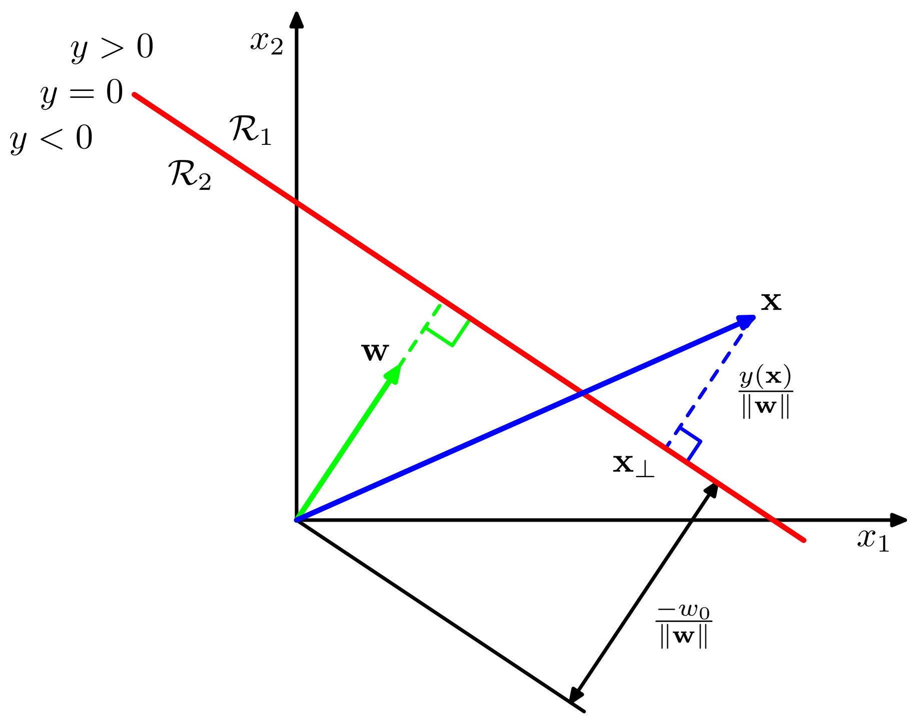
$$\tilde{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}, \tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

K classes:

$$\mathbf{y}(\mathbf{x}) = \begin{pmatrix} y_1(\mathbf{x}) \\ \cdots \\ y_K(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^T\mathbf{x} + w_{10} \\ \cdots \\ \mathbf{w}_K^T\mathbf{x} + w_{K0} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{w}}_1^T \\ \cdots \\ \tilde{\mathbf{w}}_K^T \end{pmatrix} \tilde{\mathbf{x}} = \tilde{\mathbf{W}}^T\tilde{\mathbf{x}}, \text{ with:}$$

$$\tilde{\mathbf{W}}^T = \begin{pmatrix} \tilde{\mathbf{w}}_1^T \\ \cdots \\ \tilde{\mathbf{w}}_K^T \end{pmatrix}, \text{ i.e.: } \tilde{\mathbf{W}} = (\tilde{\mathbf{w}}_1, \cdots, \tilde{\mathbf{w}}_K)$$

Linear discriminant functions



Multiple classes

K-class discriminant comprising K linear functions

$$\mathbf{y}(\mathbf{x}) = \begin{pmatrix} y_1(\mathbf{x}) \\ \cdots \\ y_K(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{w}}_1^T \tilde{\mathbf{x}} \\ \cdots \\ \tilde{\mathbf{w}}_K^T \tilde{\mathbf{x}} \end{pmatrix} = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$$

Classify $\mathbf{x}$ as C_k with $k = \mathrm{argmax}_{j=1,\dots,K}\{y_j(\mathbf{x})\}$

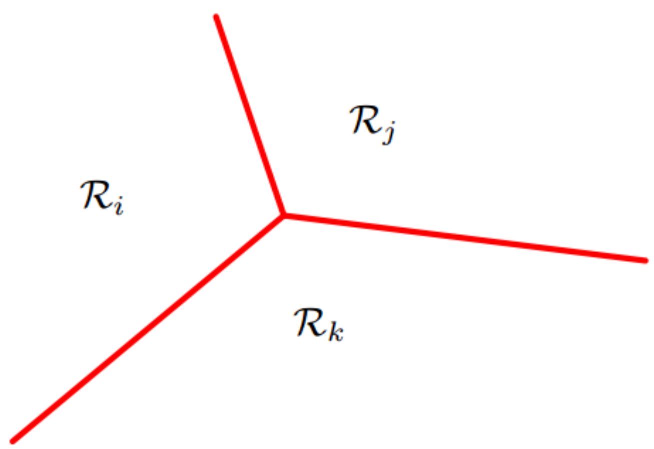
Decision boundary between C_k and C_j (hyperplane in $\Re^{D-1}$):

$$(\tilde{\mathbf{w}}_k - \tilde{\mathbf{w}}_j)^T \, \tilde{\mathbf{x}} = 0$$

Multiple classes

when we have n-dimensional space

Example of K-class discriminant



Learning linear discriminants

Given a multi-class classification problem and data set D with linearly separable data,

determine $\tilde{\mathrm{W}}$ such that $\mathrm{y}(\mathrm{x}) = \tilde{\mathrm{W}}^T \tilde{\mathrm{x}}$ is the K-class discriminant.

Approaches to learn linear discriminants

- Least squares
- Perceptron
- Fisher's linear discriminant
- Support Vector Machines

Least squares

Given $D = \{(\mathrm{x}_n, \mathrm{t}_n)_{n=1}^N\}$, find the linear discriminant

$$y(\mathrm{x}) = \tilde{\mathrm{W}}^T \tilde{\mathrm{x}}$$

one hot encoding of code

1-of-K coding scheme for t: $\mathrm{x} \in C_k \rightarrow t_k = 1, t_j = 0$ for all $j \neq k$.
E.g., $\mathrm{t}_n = (0, \ldots, 1, \ldots, 0)^T$

$$\underbrace{\tilde{\mathrm{X}} = \begin{pmatrix} \tilde{\mathrm{x}}_1^T \\ \cdots \\ \tilde{\mathrm{x}}_N^T \end{pmatrix} \qquad \mathrm{T} = \begin{pmatrix} \mathrm{t}_1^T \\ \cdots \\ \mathrm{t}_N^T \end{pmatrix}}$$

↳ complete representation of dataset

Least squares

Minimize sum-of-squares error function

$$E(\tilde{\mathrm{W}}) = \frac{1}{2} Tr\left\{(\tilde{\mathrm{X}}\tilde{\mathrm{W}} - \mathrm{T})^T (\tilde{\mathrm{X}}\tilde{\mathrm{W}} - \mathrm{T})\right\}$$

Closed-form solution:

solution of minimization can be written in close form

$$\tilde{\mathrm{W}} = \underbrace{(\tilde{\mathrm{X}}^T\tilde{\mathrm{X}})^{-1}\tilde{\mathrm{X}}^T}_{\tilde{\mathrm{X}}^\dagger} \mathrm{T}$$

↳ pseudoinverse

Learned model:

$$y(\mathrm{X}) = \tilde{\mathrm{W}}^T \tilde{\mathrm{X}} = \mathrm{T}^T (\tilde{\mathrm{X}}^\dagger)^T \tilde{\mathrm{X}}$$

Least squares

Classification of new instance $\mathrm{x}' \notin D$

Use learnt $\tilde{\mathrm{W}}$ to compute:

$$y(\mathrm{x}') = \tilde{\mathrm{W}}^T \tilde{\mathrm{x}}' = \begin{pmatrix} y_1(\mathrm{x}') \\ \cdots \\ y_K(\mathrm{x}') \end{pmatrix}$$

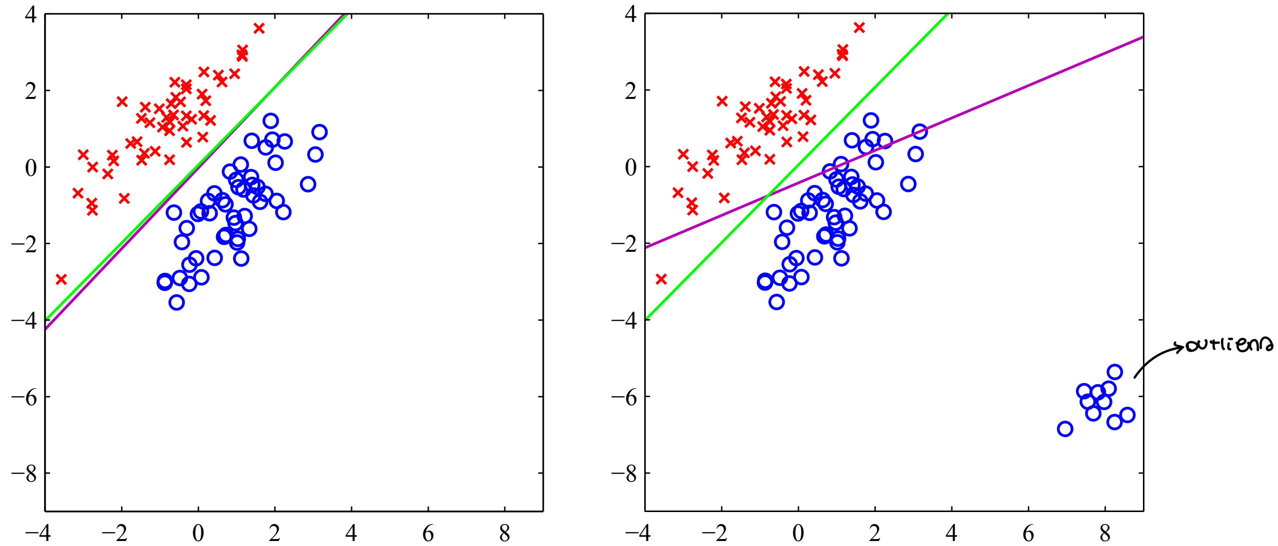
Predict C_k with:

$$k = \underset{j \in \{1,\ldots,K\}}{\operatorname{argmax}} \{y_j(\mathrm{x}')\}$$

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

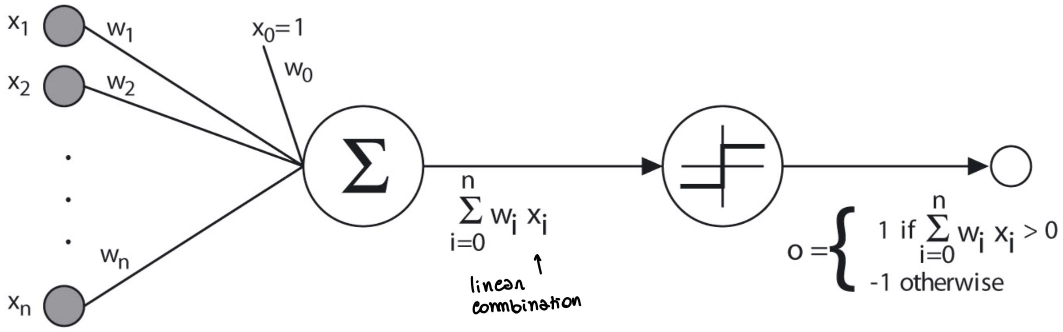
Issues with least squares

Assume Gaussian conditional distributions. Not robust to outliers!



Perceptron

Learning $f : \Re^d \rightarrow \{-1, +1\}$



$$o(x_1, \ldots, x_d) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_d x_d > 0 \\ -1 & \text{otherwise.} \end{cases}$$

$$o(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} > 0 \\ -1 & \text{otherwise.} \end{cases} = sign(\mathbf{w}^T \mathbf{x})$$

Perceptron training rule

Consider the *unthresholded linear unit*, where

$$o = w_0 + w_1 x_1 + \cdots + w_d x_d = \mathbf{w}^T \mathbf{x}$$

Let's learn w_i from training examples $D = \{(\mathbf{x}_n, t_n)_{n=1}^N\}$ that minimize the squared error (*loss function*)

$$E(\mathbf{w}) \equiv \underbrace{\frac{1}{2} \sum_{n=1}^{N} (t_n - o_n)^2}_{\text{0 when model predict excty the label of samples of dataset}} = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

Perceptron training rule

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathrm{w}_n^T \mathrm{x}_n)^2 = \frac{1}{2} \sum_{n=1}^{N} \frac{\partial}{\partial w_i} (t_n - \mathrm{w}^T \mathrm{x}_n)^2 \\
&= \frac{1}{2} \sum_{n=1}^{N} 2(t_n - \mathrm{w}^T \mathrm{x}_n) \frac{\partial}{\partial w_i} (t_n - \mathrm{w}^T \mathrm{x}_n) \\
&= \sum_{n=1}^{N} (t_n - \mathrm{w}^T \mathrm{x}_n) \frac{\partial}{\partial w_i} (t_n - \mathrm{w}^T \mathrm{x}_n) \\
&= \sum_{n=1}^{N} (t_n - \mathrm{w}^T \mathrm{x}_n)(-x_{i,n})
\end{aligned}
$$

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Perceptron training rule

Unthresholded unit:

Update of weights w

$$
\begin{aligned}
w_i &\leftarrow w_i + \Delta w_i \\
\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{n=1}^{N} (t_n - \mathrm{w}^T \mathrm{x}_n)\, x_{i,n}
\end{aligned}
$$

η is a small constant (e.g., 0.05) called *learning rate*

Perceptron training rule

Thresholded unit:

Update of weights w

$$
\begin{aligned}
w_i &\leftarrow w_i + \Delta w_i \\
\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{n=1}^{N} (t_n - sign(\mathrm{w}^T \mathrm{x}_n))\, x_{i,n}
\end{aligned}
$$

Perceptron algorithm

Given perceptron model $o(\mathrm{x}) = sign(\mathrm{w}^T \mathrm{x})$ and data set D, determine weights w.

1. Initialize $\hat{\mathrm{w}}$ (e.g. small random values)
2. Repeat until termination condition
   - $\hat{w}_i \leftarrow \hat{w}_i + \Delta w_i$
3. Output $\hat{\mathrm{w}}$

Perceptron algorithm

Batch mode: Consider all dataset D

$$\Delta w_i = \eta \sum_{(\mathbf{x},t)\in D} (t - o(\mathbf{x}))\, x_i$$

⇒ high computational

Mini-Batch mode: Choose a small subset $S \subset D$ ⇒ best choice

$$\Delta w_i = \eta \sum_{(\mathbf{x},t)\in S} (t - o(\mathbf{x}))\, x_i$$

Incremental mode: Choose one sample $(\mathbf{x}, t) \in D$ ⇒ high variance

$$\Delta w_i = \eta\, (t - o(\mathbf{x}))\, x_i$$

$o(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ for unthresholded, $o(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x})$ for thresholded
Incremental and mini-batch modes speed up convergence and are less sensitive to local minima.

Perceptron algorithm

Termination conditions

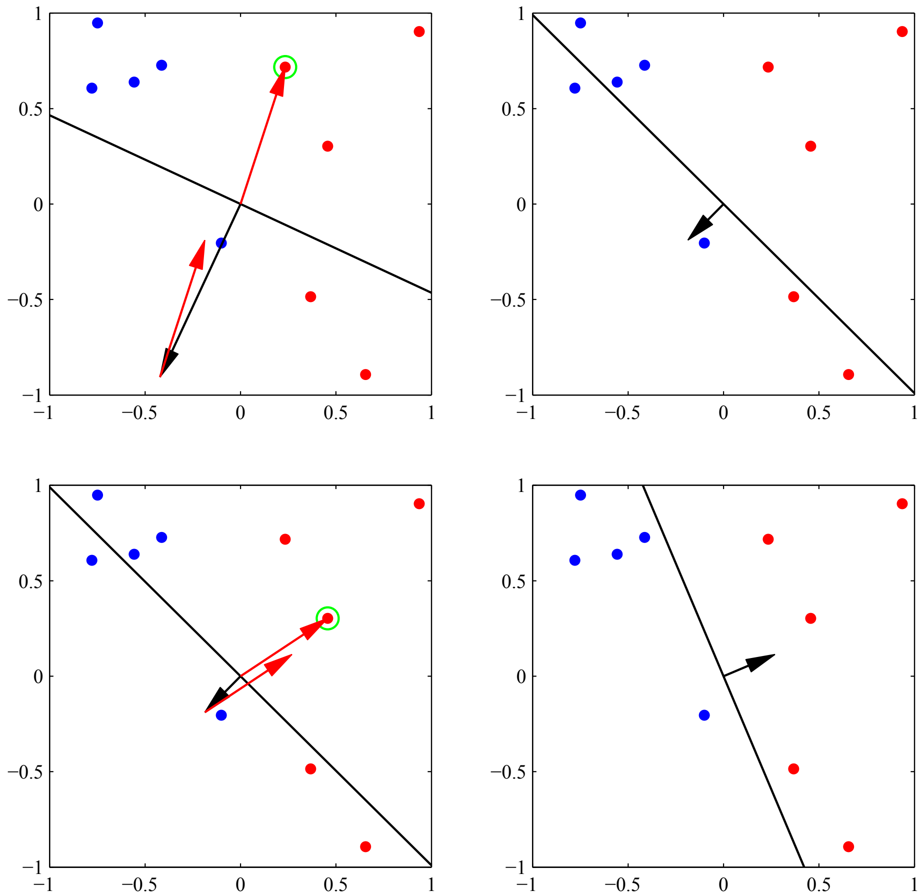
- Predefined number of iterations
- Threshold on changes in the loss function $E(\mathbf{w})$

Perceptron training rule

Example:
$\eta = 0.1$, $x_i = 0.8$

- if $t = 1$ and $o = -1$ then $\Delta w_i = 0.16$
- if $t = -1$ and $o = 1$ then $\Delta w_i = -0.16$

Perceptron training rule



Perceptron training rule

Can prove it will converge:

- if training data is linearly separable
- and η sufficiently small

Small $\eta \rightarrow$ slow convergence.

Perceptron: Prediction

Classification of new instance $\mathrm{x}' \notin D$:

Predict C_k, with $k = sign(\mathrm{w}^T \mathrm{x}')$, using learnt w

Fisher's linear discriminant

Consider two classes case.

Determine $y = \mathrm{w}^T \mathrm{x}$
and classify $\mathrm{x} \in C_1$ if $y \geq -w_0$, $\mathrm{x} \in C_2$ otherwise.

Corresponding to the projection on a line determined by w.

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Fisher's linear discriminant

Adjusting w to find a direction that maximizes class separation.

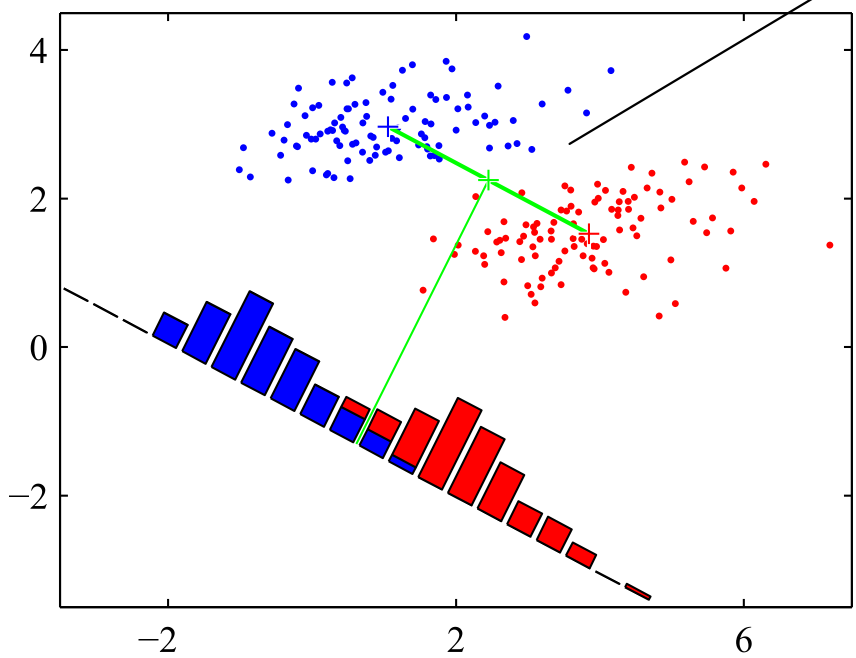
Consider a data set with N_1 points in C_1 and N_2 points in C_2

$$\mathrm{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathrm{x}_n \qquad \mathrm{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathrm{x}_n$$

Choose w that maximizes $J(\mathrm{w}) = \mathrm{w}^T(\mathrm{m}_2 - \mathrm{m}_1)$, subject to $||\mathrm{w}|| = 1$.

Fisher's linear discriminant

$$\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$$



this direction is not consider the covariance

↓

we will use covariance to rotate this line

Fisher's linear discriminant

Fisher criterion

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

with
$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$
Between class scatter

$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$
Within class scatter

Choose $\mathbf{w}$ that maximizes $J(\mathbf{w})$.

Fisher's linear discriminant

Find w that maximizes

$$J(\mathrm{w}) = \frac{\mathrm{w}^T \mathrm{S}_B \mathrm{w}}{\mathrm{w}^T \mathrm{S}_W \mathrm{w}}$$

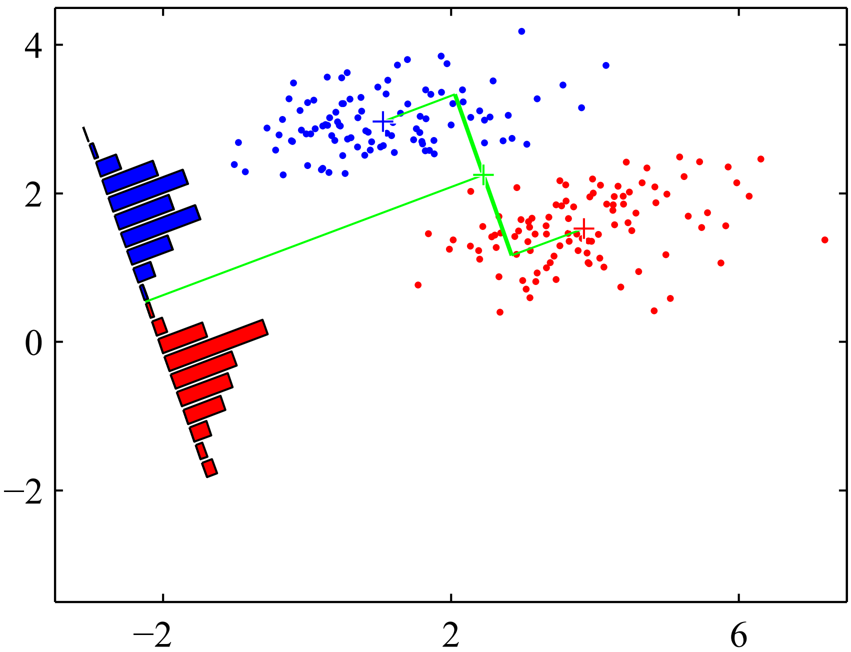
by solving

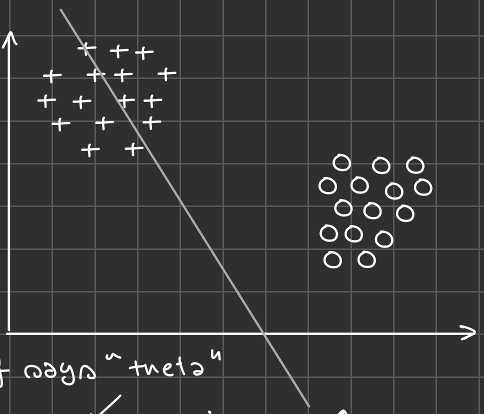
$$\frac{d}{d\mathrm{w}} J(\mathrm{w}) = 0$$

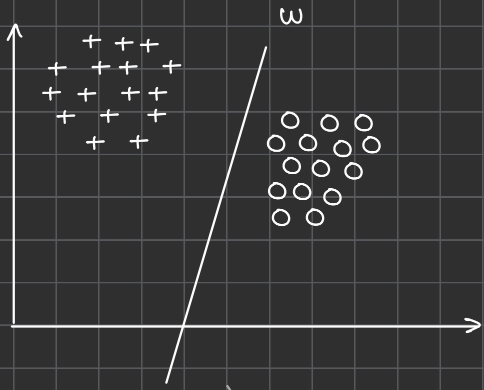
$$\Rightarrow \mathrm{w}^* \propto \mathrm{S}_W^{-1}(\mathrm{m}_2 - \mathrm{m}_1)$$

Fisher's linear discriminant

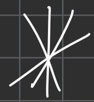
$$\mathrm{w} \propto \mathrm{S}_W^{-1}(\mathrm{m}_2 - \mathrm{m}_1)$$







prof says "theta"

if $\theta << 0$ 

change slow

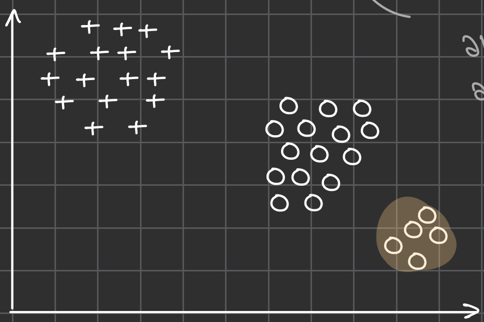
if $\theta >> 0$ 

change fast

SO PERCEPTRONS is ROBUST TO OUTLIERS

$\Delta w \quad (t_n - sign(w^T x_n))$

in this case all term = 0



also with outlier all this terms = 0

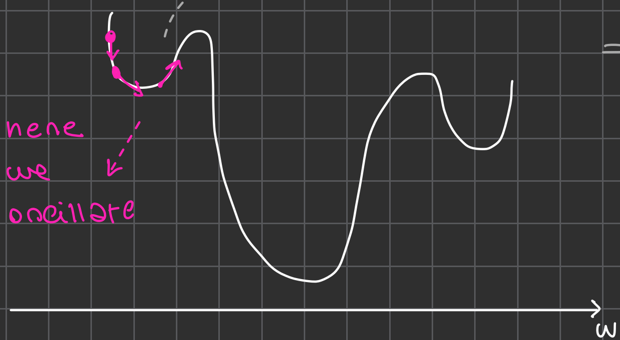
outlier

contribute 0 to update rule

ERROR FUNCTION:

we will stuck here

here we oscillate

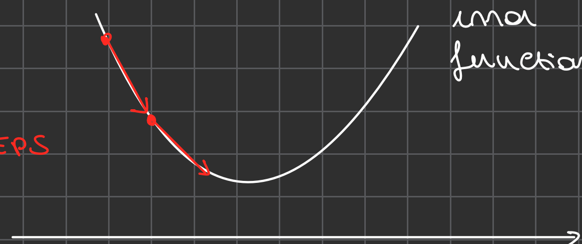


=> that mean we stop in LOCAL MINIMAL

we cannot define how small or big must be ERROR FUNCTION

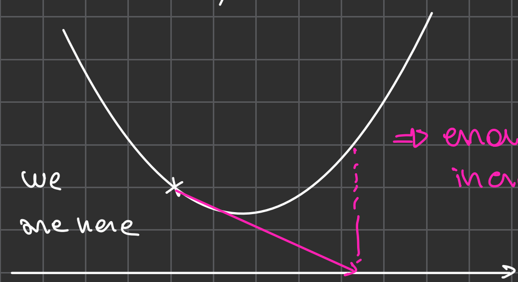
• WHAT HAPPEN IF "learning rate" is TO HIGH

BIG STEPS



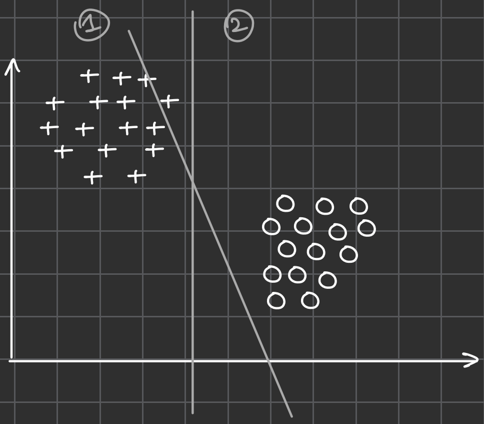
error function

we are here



=> error is increasing 'n'

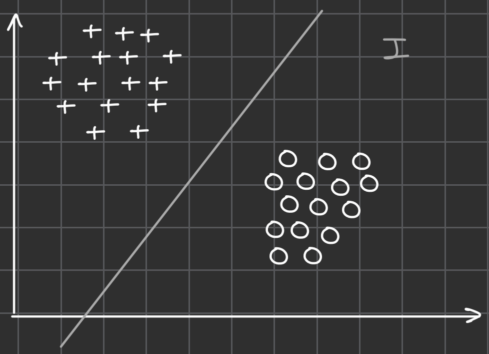
we can also DIVERGE

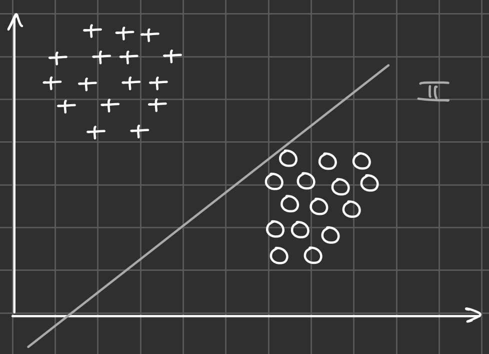


=> this is NOT GOOD because if we consider samples that are not in dataset this will not CORRECT

come vole la risposta

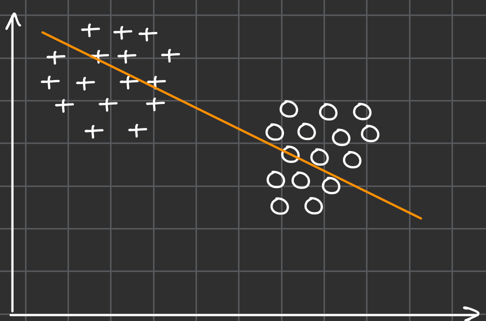
→ our algorithm will stop in second iterations

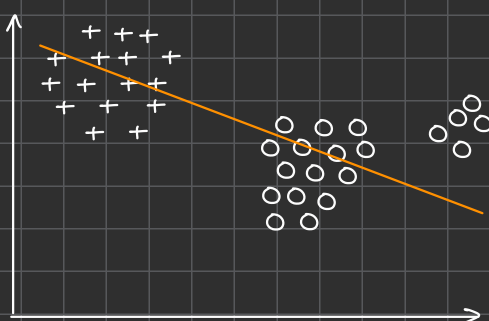


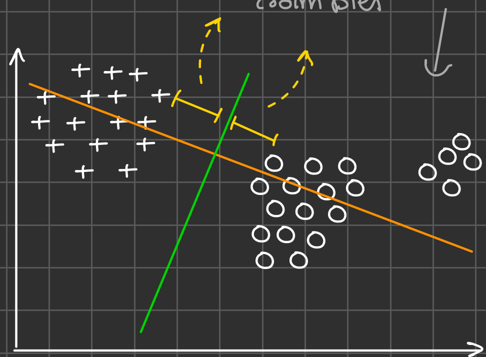


if we consider a small learning rate II is more probable than I

we need some margins so will be more correct with new samples







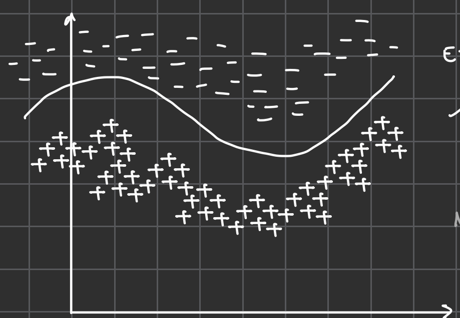
FISHER we use data for estimation of error function so outliers we will change small slope

$y(x;w) = w^T x$ => linear model also x

maybe non-linear

$y(x;w) = w^T \phi(x)$ => linear model but not in x

still linear in w



EXAMPLE

$$y(x^T, w) = w^T \phi(x) = w^T \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

NOT SEPARABLE

Fisher's linear discriminant

Summarizing, given a two classes classification problem, Fisher's linear discriminant is given by the function $y = \mathrm{w}^T \mathrm{x}$ and the classification of new instances is given by $y \geq -w_0$
where

$$\mathrm{w} = \mathrm{S}_W^{-1}(\mathrm{m}_2 - \mathrm{m}_1)$$

$$w_0 = \mathrm{w}^T \mathrm{m}$$

m is the global mean of all the data set.

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Fisher's linear discriminant

Multiple classes.

$$\mathrm{y} = \mathrm{W}^T \mathrm{x}$$

Maximizing

$$J(\mathrm{W}) = Tr\left\{(\mathrm{W}\mathrm{S}_W\mathrm{W}^T)^{-1}(\mathrm{W}\mathrm{S}_B\mathrm{W}^T)\right\}$$

...

Support Vector Machines

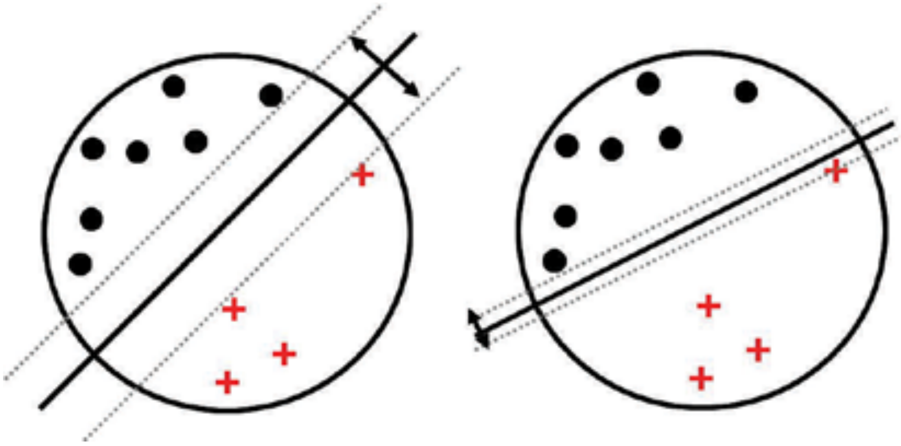
→ ROBUST TO OUTLIERS THE VALUE WILL BE =0

Support Vector Machines (SVM) for Classification aims at maximum margin providing for better accuracy.

similar to exam

Q: WHERE THIS TWO IMAGES COMES FROM

A: LEFT → SVM (probably)



Support Vector Machines

Let's consider binary classification $f : X \rightarrow \{+1, -1\}$ with data set $D = \{(\mathbf{x}_n, t_n)_{n=1}^{N}\}$, $t_n \in \{+1, -1\}$ and a linear model

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

Assume D is linearly separable

$$\exists \mathbf{w}, w_0 \ s.t. \quad \begin{array}{l} y(\mathbf{x}_n) > 0, \ if \ t_n = +1 \\ y(\mathbf{x}_n) < 0, \ if \ t_n = -1 \end{array}$$

COMPACT FORMULA →

$$t_n \, y(\mathbf{x}_n) > 0 \ \forall n = 1, \dots N$$

Support Vector Machines

Let x_k be the closest point of the data set D to the hyperplane $h : \mathrm{w}^T\mathrm{x} + w_0 = 0$

the *margin* (smallest distance between x_k and h) is $\frac{|y(\mathrm{x}_k)|}{||\mathrm{w}||}$

Given data set D and hyperplane h, the margin is computed as

$$\min_{n=1,\dots,N} \frac{|y(\mathrm{x}_n)|}{||\mathrm{w}||} = \cdots = \frac{1}{||\mathrm{w}||} \min_{n=1,\dots,N} [t_n(\mathrm{w}^T\mathrm{x}_n + w_0)]$$

using the property $|y(\mathrm{x}_n)| = t_n\, y(\mathrm{x}_n)$

Support Vector Machines

Given data set D, the hyperplane $h^* : \mathrm{w}^{*T}\mathrm{x} + w_0{}^* = 0$ with maximum margin is computed as

$$\mathrm{w}^*, w_0{}^* = \operatorname*{argmax}_{\mathrm{w}, w_0} \underbrace{\frac{1}{||\mathrm{w}||} \min_{n=1,\dots,N} [t_n(\mathrm{w}^T\mathrm{x}_n + w_0)]}_{\text{MAXIMUM MARGIN OPTIMIZATION PROBLEM}}$$

Support Vector Machines

Rescaling all the points does not affect the solution.

Rescale in such a way that for the closet point x_k we have

$$t_k(\mathbf{w}^T \mathbf{x}_k + w_0) = 1$$

IF DATASET IS NOT SEPARABE

THE ALGORITHM WILL RETURN NO SOLUTION

Canonical representation:

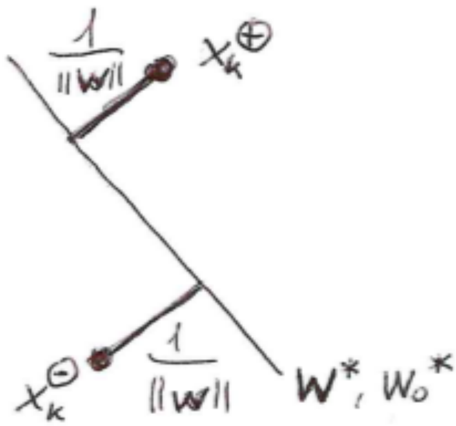
$$t_n(\mathbf{w}^T \mathbf{x}_n + w_0) \geq 1 \quad \forall n = 1, \ldots, N$$

is only = 1 for the closest point to the margin

Support Vector Machines

When the maximum margin hyperplane $\mathbf{w}^*, w_0{}^*$ is found, there will be at least 2 closest points $\mathbf{x}_k^{\oplus}$ and $\mathbf{x}_k^{\ominus}$ (one for each class).

$$\mathbf{w}^{*T} \mathbf{x}_k^{\oplus} + w_0^* = +1$$
$$\mathbf{w}^{*T} \mathbf{x}_k^{\ominus} + w_0^* = -1$$



Support Vector Machines

In the canonical representation of the problem the maximum margin hyperplane can be found by solving the optimization problem

$$\mathrm{w}^*, w_0^* = \operatorname{argmax} \frac{1}{||\mathrm{w}||} = \operatorname{argmin} \frac{1}{2}||\mathrm{w}||^2$$

subject to

$$t_n(\mathrm{w}^T \mathrm{x}_n + w_0) \geq 1 \ \ \forall n = 1, \ldots, N$$

Quadratic programming problem solved with Lagrangian method.

Support Vector Machines

Solution

$$\mathrm{w}^* = \sum_{n=1}^{N} a_n^* \, t_n \, \mathrm{x}_n$$

a_i^* (Lagrange multipliers): results of the Lagrangian optimization problem

$$\tilde{L}(\mathrm{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n \, a_m \, t_n \, t_m \, \mathrm{x}_n^T \mathrm{x}_m$$

subject to

$$a_n \geq 0 \ \ \forall n = 1, \ldots, N$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Support Vector Machines

Note:
samples $\mathbf{x}_n$ for which $a_n^* = 0$ do not contribute to the solution

Karush-Kuhn-Tucker (KKT) condition:

for each $\mathbf{x}_n \in D$, *either* $a_n^* = 0$ *or* $t_n\, y(\mathbf{x}_n) = 1$

thus $t_n\, y(\mathbf{x}_n) > 1$ implies $a_n^* = 0$

Support vectors: $\mathbf{x}_k$ such that $t_k\, y(\mathbf{x}_k) = 1$ and $a_k^* > 0$

$$SV \equiv \{\mathbf{x}_k \in D \mid t_k\, y(\mathbf{x}_k) = 1\}$$

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Support Vector Machines

Hyperplanes expressed with support vectors

$$y(\mathbf{x}) = \sum_{\mathbf{x}_j \in SV} a_j^*\, t_j\, \mathbf{x}^T \mathbf{x}_j + w_0^* = 0$$

Note: other vectors $\mathbf{x}_n \notin SV$ do not contribute $(a_n^* = 0)$

Support Vector Machines

To compute w_0^*:

Support vector $\mathbf{x}_k \in SV$ satisfies $t_k\, y(\mathbf{x}_k) = 1$

$$t_k \left(\sum_{\mathbf{x}_j \in SV} a_j^* t_j \mathbf{x}_k^T \mathbf{x}_j + w_0^* \right) = 1$$

Multiplying by t_k and using $t_k^2 = 1$

$$w_0^* = t_k - \sum_{\mathbf{x}_j \in SV} a_j^* t_j \mathbf{x}_k^T \mathbf{x}_j$$

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Support Vector Machines

Instead of using one particular support vector $\mathbf{x}_k$ to determine w_0

$$w_0^* = t_k - \sum_{\mathbf{x}_j \in SV} a_j^* t_j \mathbf{x}_k^T \mathbf{x}_j$$

a more stable solution is obtained by averaging over all the support vectors

$$w_0^* = \frac{1}{|SV|} \sum_{\mathbf{x}_k \in SV} \left(t_k - \sum_{\mathbf{x}_j \in S} a_j^* t_j \mathbf{x}_k^T \mathbf{x}_j \right)$$

Support Vector Machines

Given the maximum margin hyperplane determined by a_k^*, w_0^*

Classification of a new instance $\mathbf{x}'$

$$y(\mathbf{x}') = sign\left(\sum_{\mathbf{x}_k \in SV} a_k^* \, t_k \, \mathbf{x}'^T \mathbf{x}_k + w_0^*\right)$$

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Support Vector Machines

Optimization problem for determining $\mathbf{w}, w_0$ (dimension $d+1$, with $X = \Re^d$) transformed in an optimization problem for determining $\mathbf{a}$ (dimension $|D|$)

Efficient when $d << |D|$ (most of a_i will be zero).
Very useful when d is large or infinite.

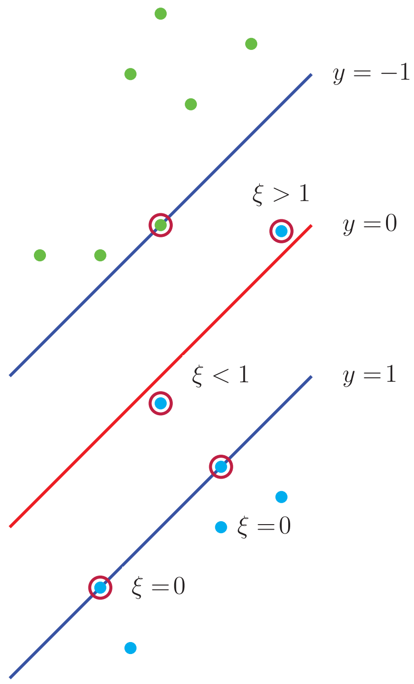
Support Vector Machines with soft margin constraints

What if data are "almost" linearly separable (e.g., a few points are on the "wrong side")

Let us introduce *slack variables* $\xi_n \geq 0$ $n = 1, \ldots, N$

Support Vector Machines with soft margin constraints

- $\xi_n = 0$ if point on or inside the correct margin boundary
- $0 < \xi_n \leq 1$ if point inside the margin but correct side
- $\xi_n > 1$ if point on wrong side of boundary



when $\xi_n = 1$, the sample lies on the decision boundary $y(\mathbf{x}_n) = 0$
when $\xi_n > 1$, the sample will be mis-classified

Support Vector Machines with soft margin constraints

Soft margin constraint

$$t_n\, y(\mathbf{x}_n) \geq 1 - \xi_n\,, \quad n = 1, \ldots, N$$

Optimization problem with soft margin constraints

$$\mathbf{w}^*, w_0^* = \operatorname{argmin} \frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} \xi_n$$

subject to

$$t_n\, y(\mathbf{x}_n) \geq 1 - \xi_n\,, \quad n = 1, \ldots, N$$
$$\xi_n \geq 0\,, \quad n = 1, \ldots, N$$

C is a constant (inverse of a regularization coefficient)

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

Support Vector Machines with soft margin constraints

Solution similar to the case of linearly separable data.

$$\mathbf{w}^* = \sum_{n=1}^{N} a_n^*\, t_n\, \mathbf{x}_n$$

$$w_0^* = \ldots.$$

with a_n^* computed as solution of a Lagrangian optimization problem.

Basis functions

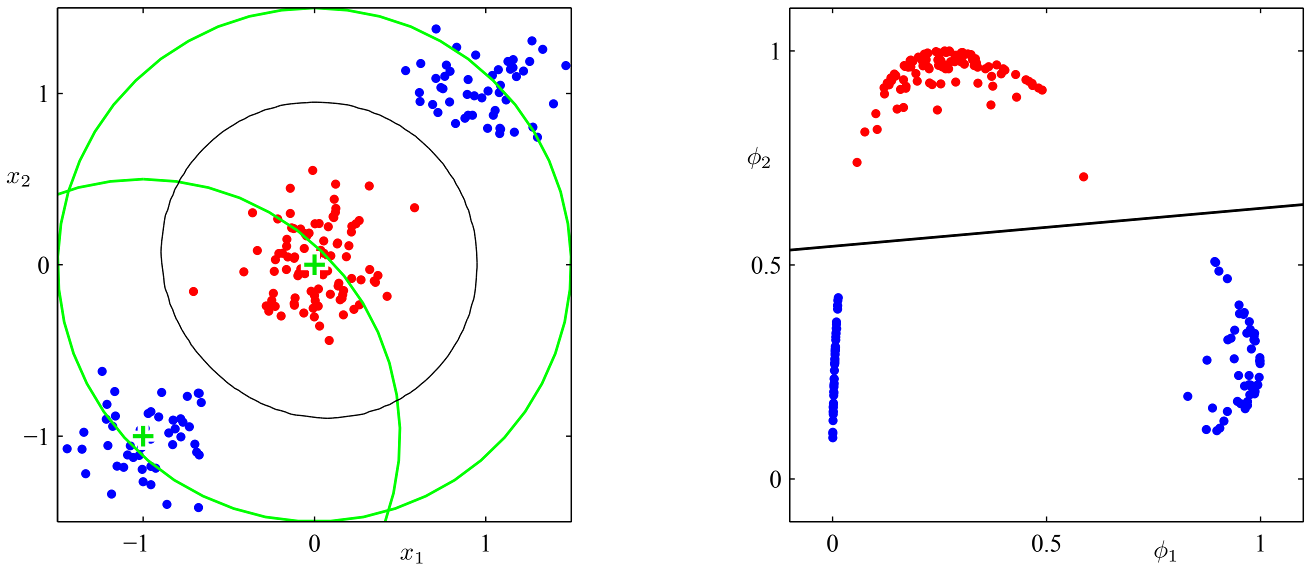
So far we considered models working directly on x.

All the results hold if we consider a non-linear transformation of the inputs $\phi(x)$ (*basis functions*).

Decision boundaries will be linear in the feature space ϕ and non-linear in the original space x

Classes that are linearly separable in the feature space ϕ may not be separable in the input space x.

Basis functions example



Basis functions examples

- Linear
- Polynomial
- Radial Basis Function (RBF)
- Sigmoid
- ...

Linear models for non-linear functions

Learning non-linear function

$$f : X \to \{C_1, \ldots, C_K\}$$

from data set D non-linearly separable.

Find a non-linear transformation ϕ and learn a linear model

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + w_0 \ \textit{(two classes)}$$

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \phi(\mathbf{x}) + w_{k0} \ \textit{(multiple classes)}$$

Summary

- Basic methods for learning linear classification functions
- Based on solution of an optimization problem
- Closed form vs. iterative solutions
- Sensitivity to outliers
- Learning non-linear functions with linear models using basis functions
- Further developed as kernel methods