Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

7. Linear models for classification

L. Iocchi, F. Patrizi

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

Linear Models for Classification

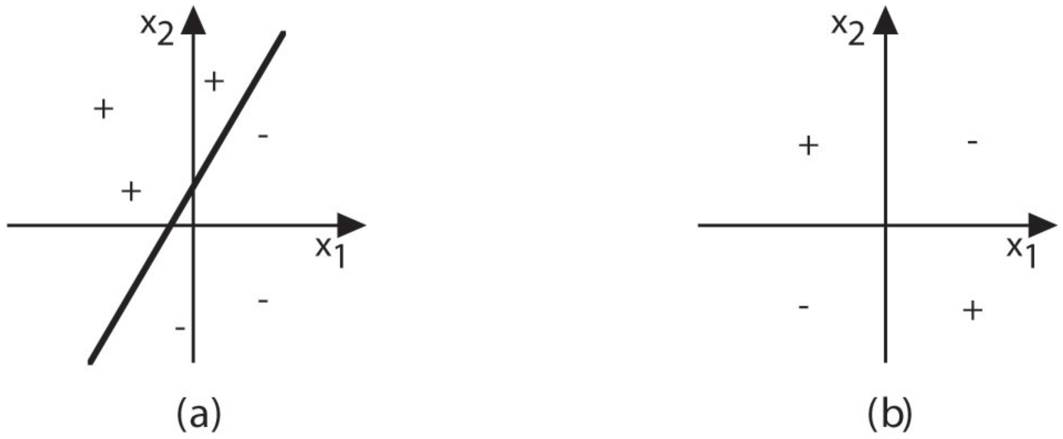
Learning a function $f : X \to Y$, with ...

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

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

$$y : X \rightarrow \{C_1, \dots, C_K\}$$

Two classes:

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

K-class:

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

Compact notation

Two classes:

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

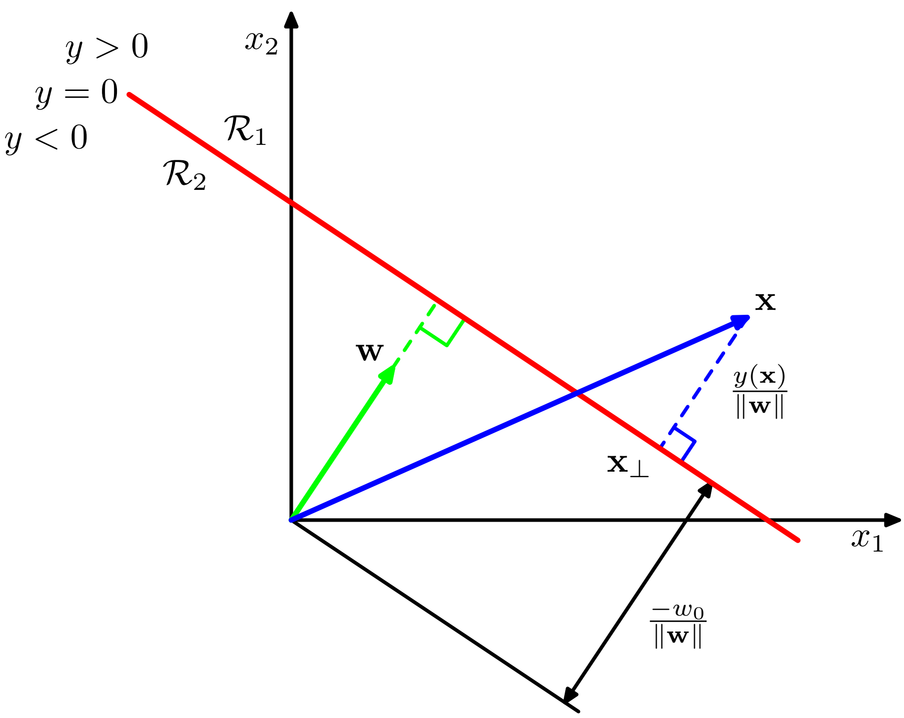
$$\tilde{\mathbf{w}} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}, \tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

K classes:

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

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

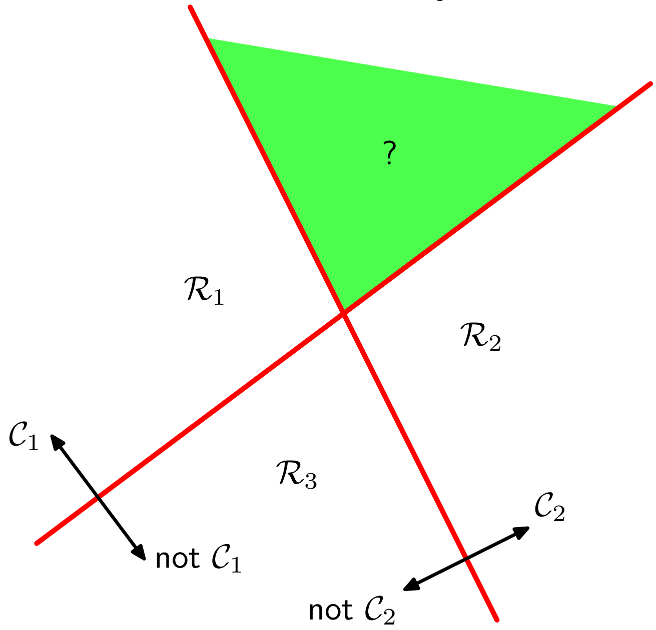
Linear discriminant functions



Multiple classes

Cannot use combinations of binary linear models.

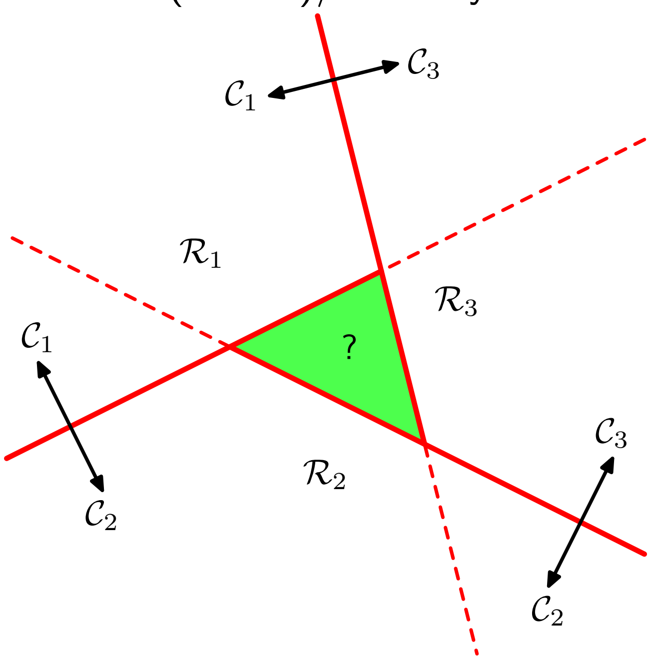
One-versus-the-rest classifiers: $K - 1$ binary classifiers: C_k vs. not-C_k



Multiple classes

Cannot use combinations of binary linear models.

One-versus-one classifiers: $K(K - 1)/2$ binary classifiers: C_k vs. C_j



Multiple classes

K-class discriminant comprising K linear functions ($\mathbf{x}$ not in dataset)

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

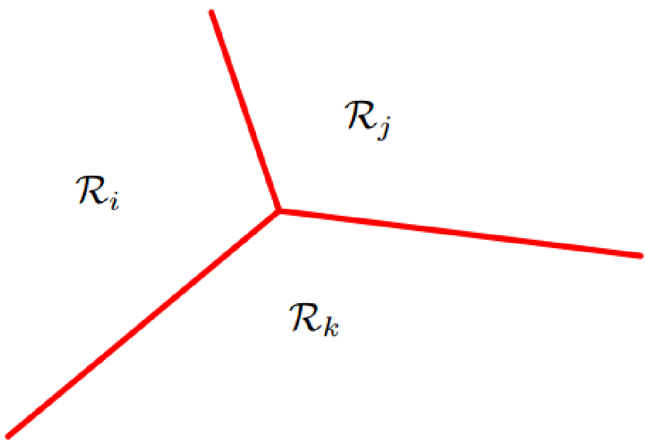
Classify $\mathbf{x}$ as C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$ ($j, k = 1, \ldots, K$)

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

Example of K-class discriminant



Learning linear discriminants

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

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

Approaches to learn linear discriminants

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

Least squares

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

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

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

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

Least squares

Minimize sum-of-squares error function

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

Closed-form solution:

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

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

Least squares

Classification of new instance $\mathbf{x}$ not in dataset:

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

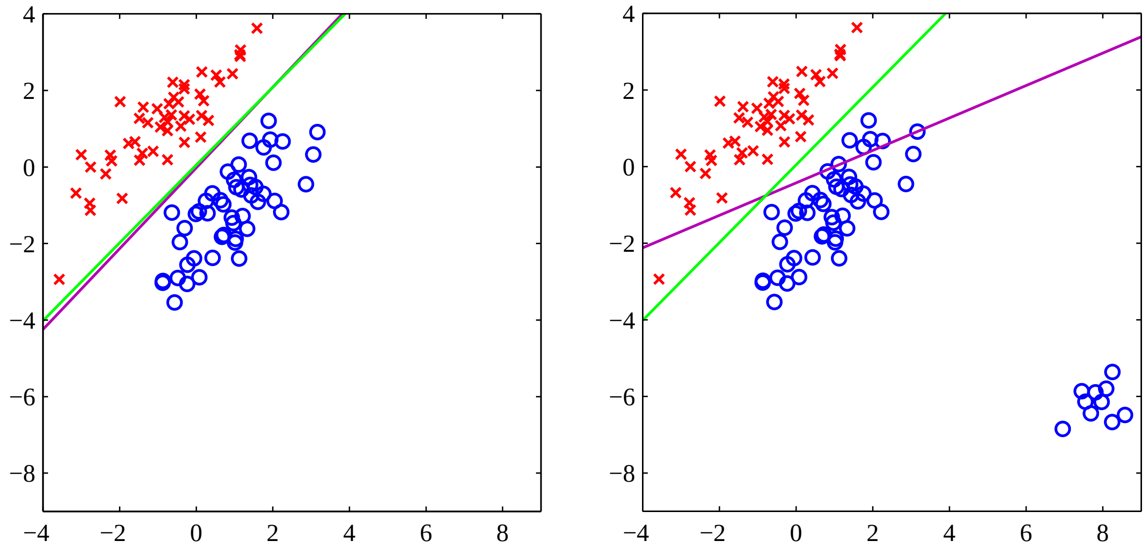
$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \begin{pmatrix} y_1(\mathbf{x}) \\ \cdots \\ y_K(\mathbf{x}) \end{pmatrix}$$

Assign class C_k to $\mathbf{x}$, where:

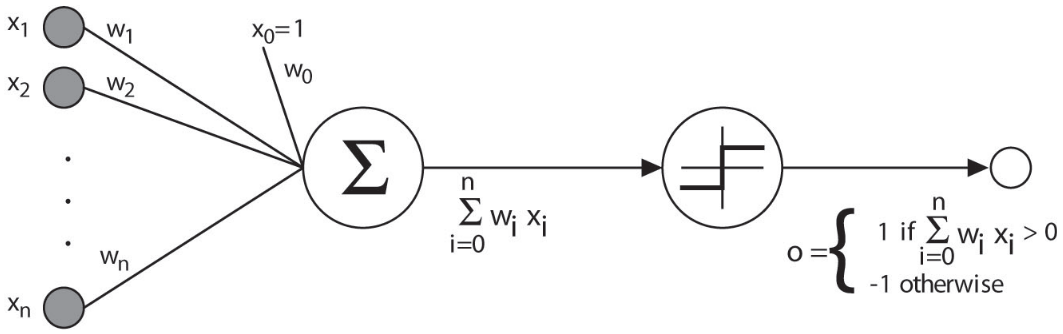
$$k = \underset{i \in \{1,\ldots,k\}}{\operatorname{argmax}} \{y_i(\mathbf{x})\}$$

Issues with least squares

Assume Gaussian conditional distributions. Not robust to outliers!



Perceptron



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

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

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*)

$$E(\mathbf{w}) \equiv \frac{1}{2} \sum_{n=1}^{N} (t_n - o_n)^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

Perceptron training rule

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

Perceptron training rule

Unthresholded unit:

Update of weights $\mathbf{w}$

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

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

Perceptron training rule

Thresholded unit:

Update of weights $\mathbf{w}$

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

Perceptron algorithm

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

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

Perceptron algorithm

Batch mode: Consider all dataset D

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

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

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

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

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

$o(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ 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

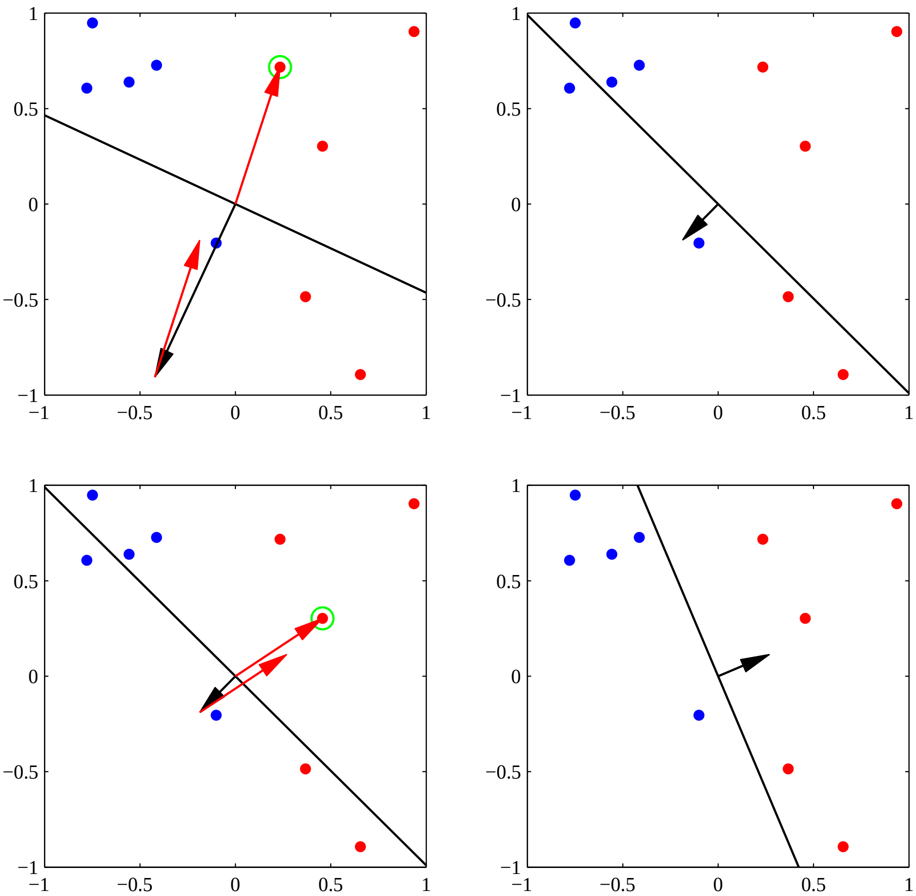
- Predefined number of iterations
- Threshold on changes in the loss function $E(\mathbf{w})$

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 $\mathbf{x}$ not in dataset:

Classify $\mathbf{x}$ as C_k, for $k = sign(\mathbf{w}^T \mathbf{x})$, using learnt $\mathbf{w}$

Fisher's linear discriminant

Consider two classes case.

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

Corresponding to the projection on a line determined by $\mathbf{w}$.

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Fisher's linear discriminant

Adjusting $\mathbf{w}$ to find a direction that maximizes class separation.

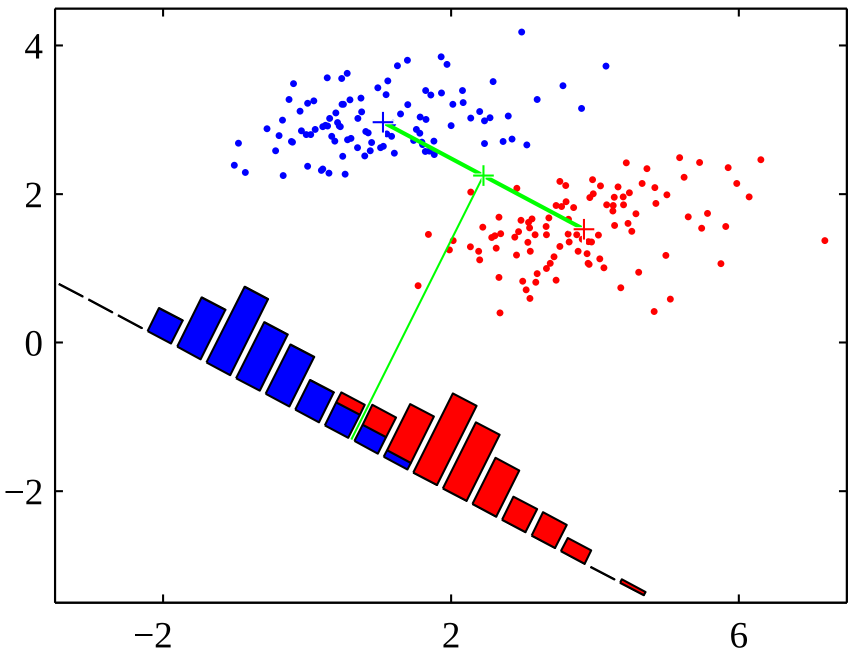
Consider a data set with N_1 points in C_1 and N_2 points in C_2

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

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

Fisher's linear discriminant

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



Sapienza University of Rome, Italy - Machine Learning (2020/2021)

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

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

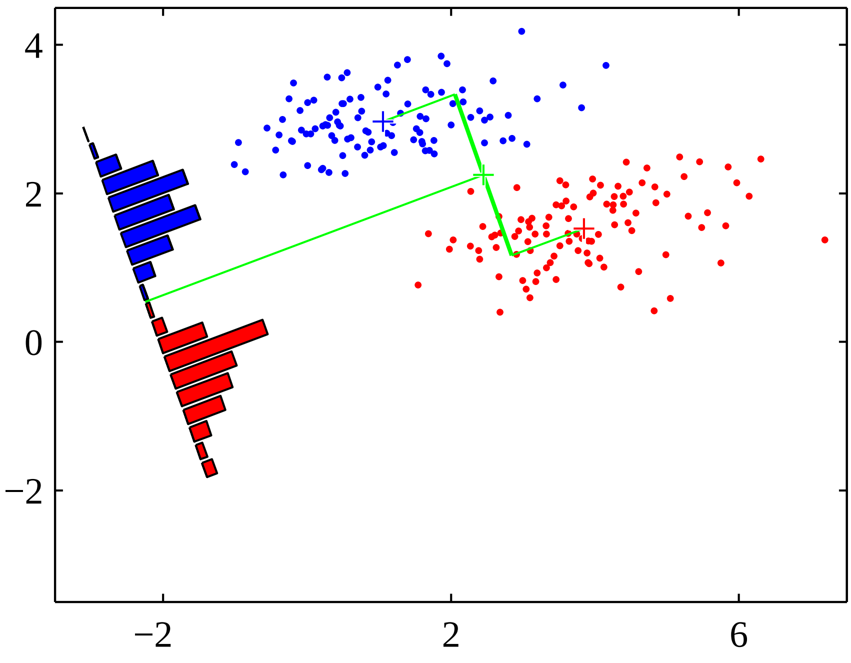
by solving

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

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

Fisher's linear discriminant

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



Fisher's linear discriminant

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

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

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

$\mathbf{m}$ is the global mean of all the data set.

Fisher's linear discriminant

Multiple classes.

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

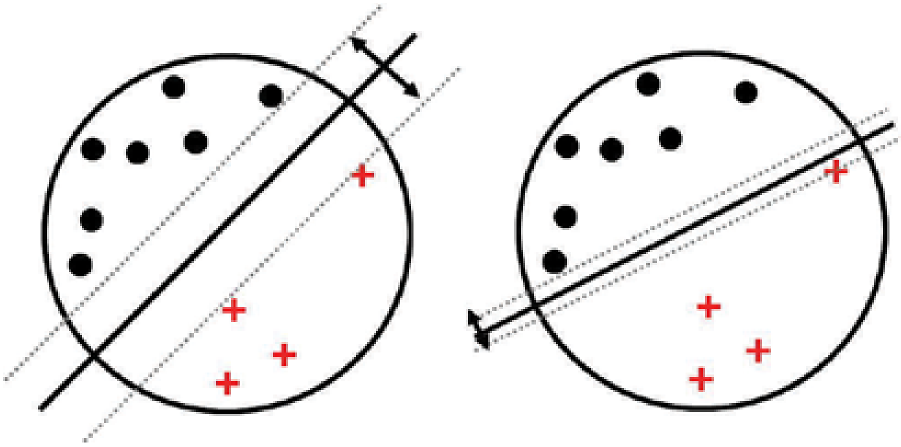
Maximizing

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

...

Support Vector Machines

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



Support Vector Machines

Let's consider binary classification $y : X \to \{+1, -1\}$ 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

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

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

Support Vector Machines

Let $\mathbf{x}_k$ be the closest point of the data set D to the hyperplane $\bar{h} : \bar{\mathbf{w}}^T \mathbf{x} + \bar{w}_0 = 0$

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

Given data set D and hyperplane $\bar{h}$, the margin is computed as

$$\min_{n=1,\ldots,N} \frac{|y(\mathbf{x}_n)|}{||\mathbf{w}||} = \cdots = \frac{1}{||\mathbf{w}||} \min_{n=1,\ldots,N} [t_n(\bar{\mathbf{w}}^T \mathbf{x}_n + \bar{w}_0)]$$

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

Support Vector Machines

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

$$\mathbf{w}^*, {w_0}^* = \underset{\mathbf{w}, w_0}{\operatorname{argmax}} \frac{1}{||\mathbf{w}||} \min_{n=1,\ldots,N} [t_n(\mathbf{w}^T \mathbf{x}_n + w_0)]$$

Support Vector Machines

Rescaling all the points does not affect the solution.

Rescale in such a way that for the closet point $\mathbf{x}_k$ we have

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

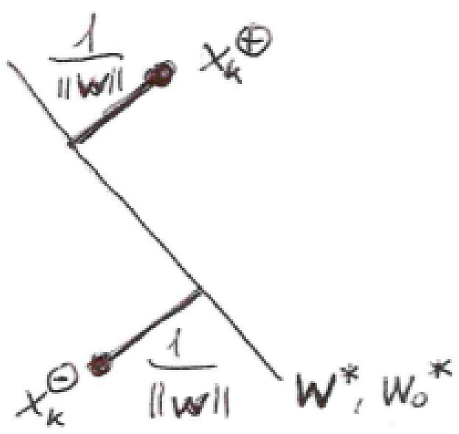
Canonical representation:

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

Support Vector Machines

When the maxim 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 maxim margin hyperplane can be found by solving the optimization problem

$$\max \frac{1}{||\mathbf{w}||} \;=\; \min \frac{1}{2}||\mathbf{w}||^2$$

subject to

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

Quadratic programming problem solved with Lagrangian method.

Support Vector Machines

Solution

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

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

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

subject to

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

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

Support Vector Machines

Karush-Kuhn-Tucker (KKT) condition:
for each $\mathbf{x}_n \in X_D$*, either* $a_n = 0$ *or* $t_n\, y(\mathbf{x}_n) = 1$

$\mathbf{x}_n$ for which $a_m = 0$ do not contribute to the solution

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

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

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

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 $t_k^2 = 1$

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

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 $\mathbf{x}'$

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

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Support Vector Machines

Optimization problem for determining $\mathbf{w}$ (dimension $|X|$) transformed in an optimization problem for determining $\mathbf{a}$ (dimension $|D|$)

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

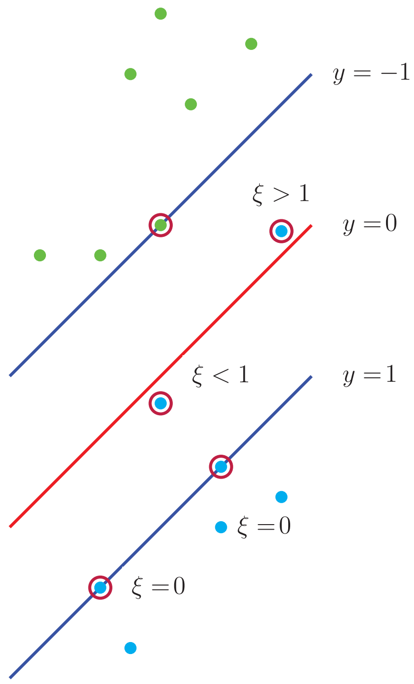
Support Vector Machines with soft margin constraints

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

$$\min \; \frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} \xi_n$$

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)

Support Vector Machines with soft margin constraints

Solution similar to the case of linearly separable data.

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

$$w_0^* = ...$$

with a_n computed as solution of a Lagrangian optimization problem.

Basis functions

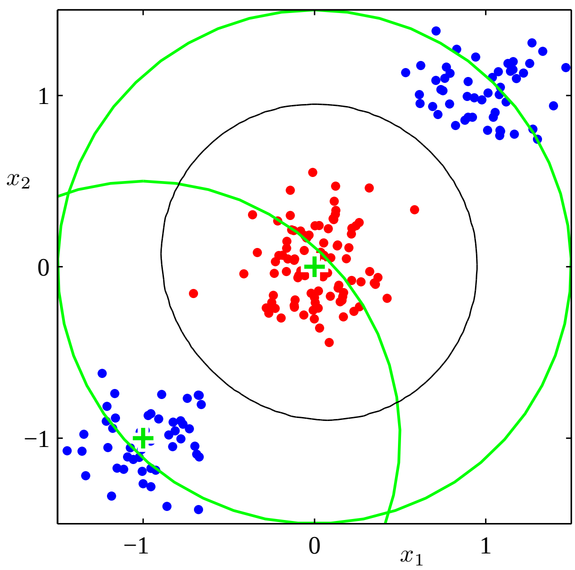
So far we considered models working directly on $\mathbf{x}$.

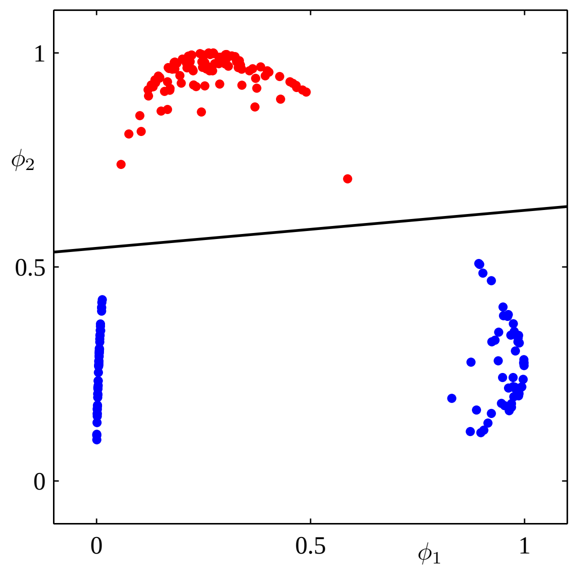
All the results hold if we consider a non-linear transformation of the inputs $\phi(\mathbf{x})$ (*basis functions*).

Decision boundaries will be linear in the feature space ϕ and non-linear in the original space $\mathbf{x}$

Classes that are linearly separable in the feature space ϕ may not be separable in the input space $\mathbf{x}$.

Basis functions example





Basis functions examples

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

Linear models for non-linear functions

Learning non-linear function

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

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