Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2025/2026

Prof. Luca Iocchi

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

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

Linear Models for Classification

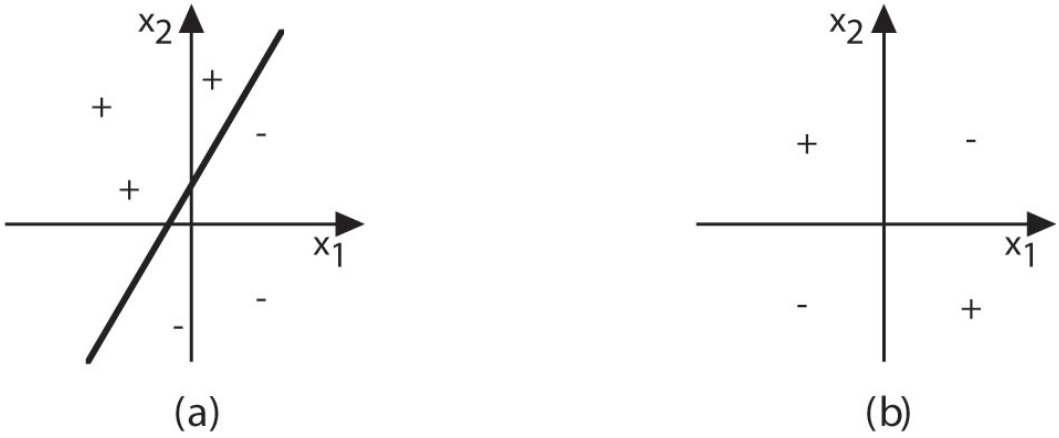
Learning a function $f : X \rightarrow Y$, with ...

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

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

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

Two classes:

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

K-classes:

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

Compact notation

Two classes:

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

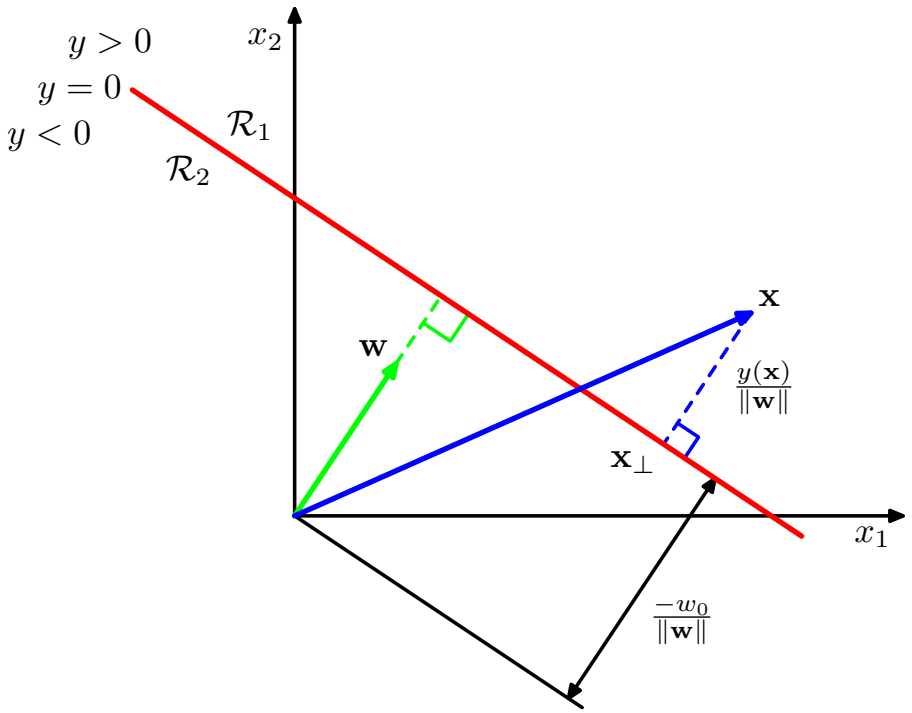
$$\tilde{\mathrm{w}} = \begin{pmatrix} w_0 \\ \mathrm{w} \end{pmatrix}, \tilde{\mathrm{x}} = \begin{pmatrix} 1 \\ \mathrm{x} \end{pmatrix}$$

K classes:

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

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

Linear discriminant functions



Multiple classes

K-class discriminant comprising K linear functions

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

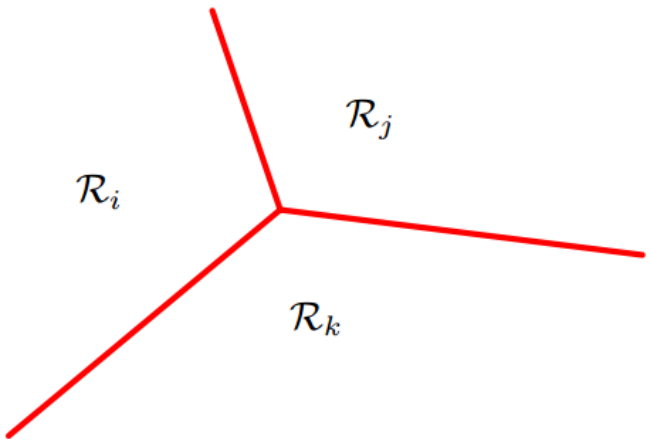
Classify x as C_k with $k = \mathrm{argmax}_{j=1,\ldots,K}\{y_j(\mathrm{x})\}$

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

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

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

Approaches to learn linear discriminants

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

Least squares

Given $D = \{(x_n, t_n)_{n=1}^N\}$, find the linear discriminant

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

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

$$\tilde{X} = \begin{pmatrix} \tilde{x}_1^T \\ \cdots \\ \tilde{x}_N^T \end{pmatrix} \qquad T = \begin{pmatrix} t_1^T \\ \cdots \\ t_N^T \end{pmatrix}$$

Least squares

Minimize sum-of-squares error function (removed $\tilde{\cdot}$)

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

Closed-form solution:

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

Learned model:

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

Least squares

Classification of new instance $x' \notin D$

Use learnt W^* to compute:

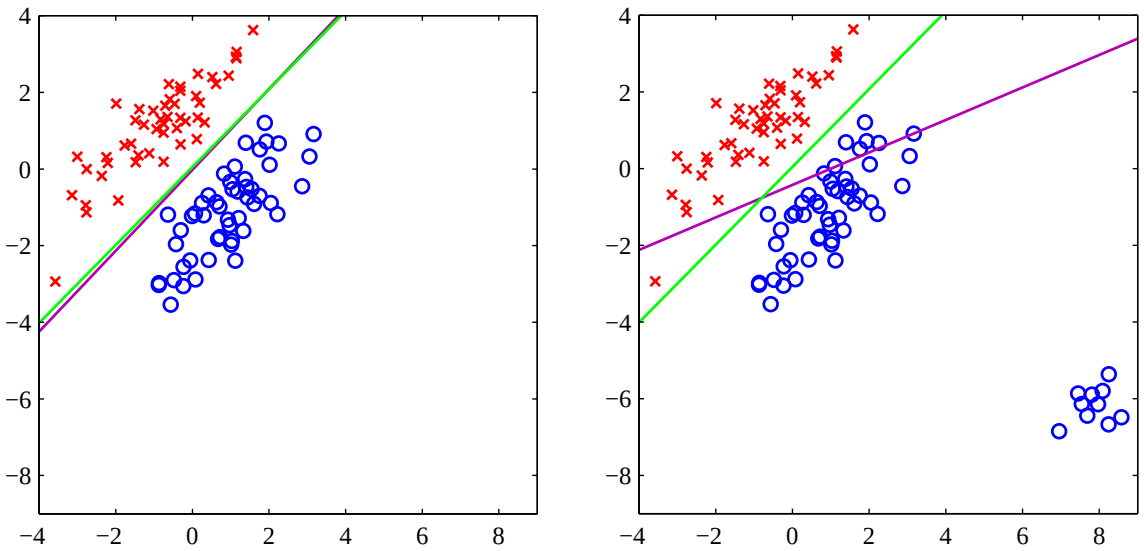
$$y(x') = W^{*T} x' = \begin{pmatrix} y_1(x') \\ \cdots \\ y_K(x') \end{pmatrix}$$

Predict C_k with:

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

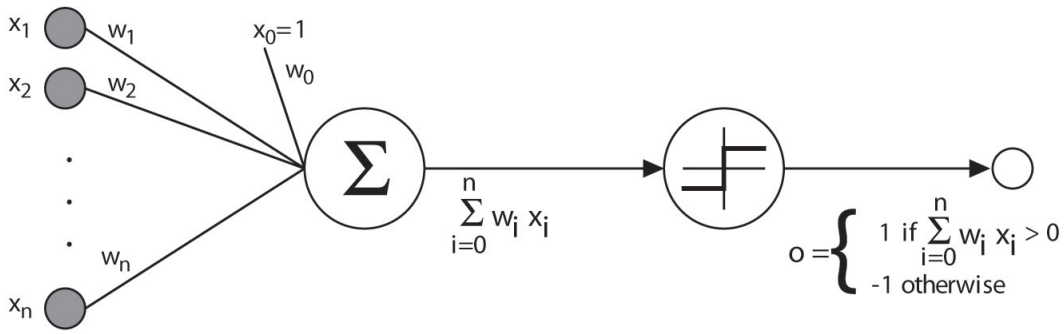
Issues with least squares

Assume Gaussian conditional distributions. Not robust to outliers!



Perceptron

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



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

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

Perceptron training rule

Consider the *unthresholded linear unit*, where

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

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

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

Perceptron training rule

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

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

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

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

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

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

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

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

Perceptron algorithm

Termination conditions

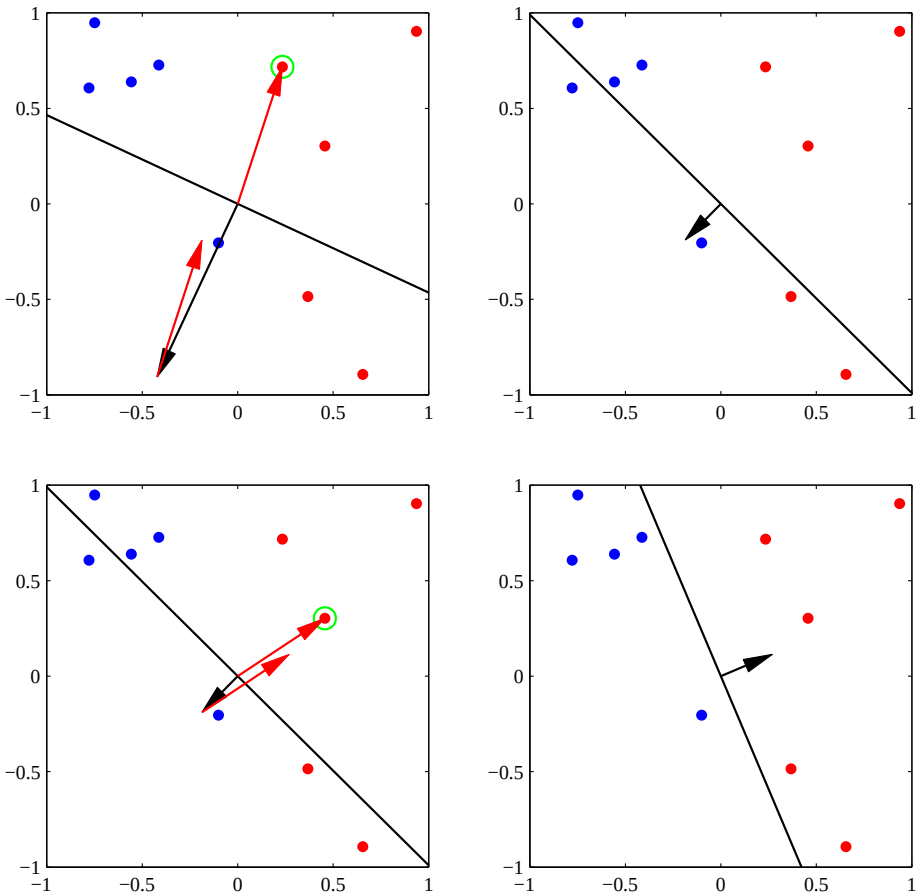
- Predefined number of iterations
- Threshold on changes in the loss function $E(\mathrm{w})$

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

Fisher's linear discriminant

Adjusting w to find a direction that maximizes class separation.

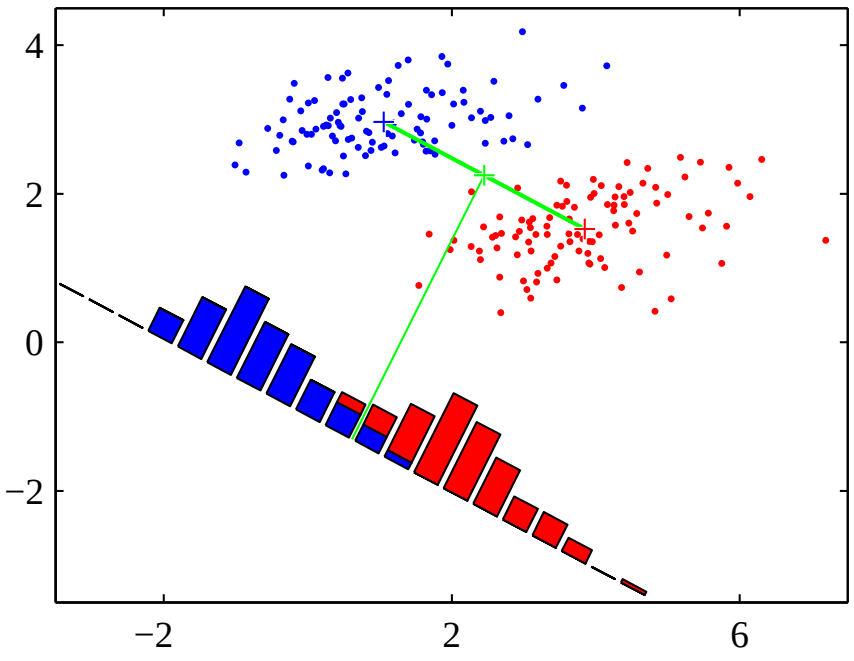
Consider a data set with N_1 points in C_1 and N_2 points in C_2

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

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

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



Fisher's linear discriminant

Fisher criterion

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

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

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

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

Fisher's linear discriminant

Find w that maximizes

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

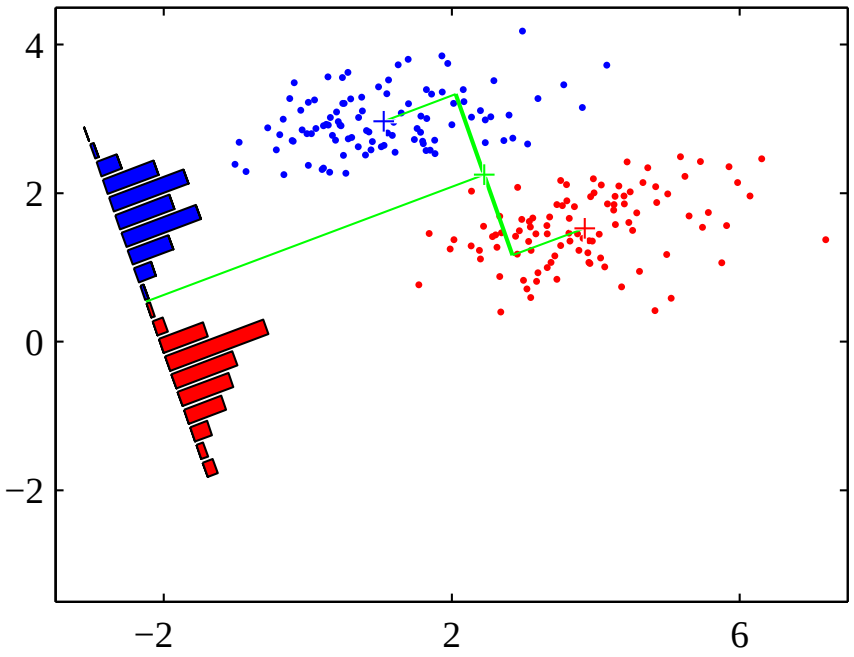
by solving

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

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

Fisher's linear discriminant

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



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.

Fisher's linear discriminant

Multiple classes.

$$y = W^T x$$

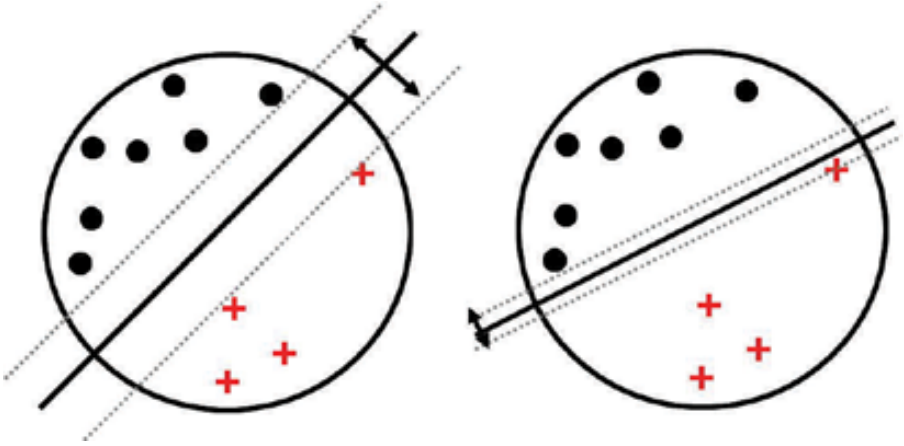
Maximizing

$$J(W) = Tr\left\{(WS_W W^T)^{-1}(WS_B W^T)\right\}$$

...

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 $f : X \to \{+1, -1\}$ with data set $D = \{(\mathrm{x}_n, t_n)_{n=1}^N\}$, $t_n \in \{+1, -1\}$ and a linear model

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

Assume D is linearly separable

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

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

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

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

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

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

Support Vector Machines

Rescaling all the points does not affect the solution.

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

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

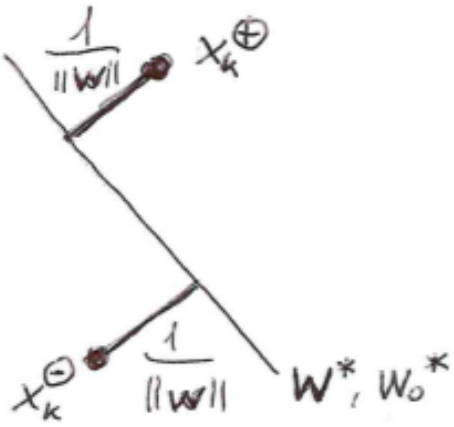
Canonical representation:

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

Support Vector Machines

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

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



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

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

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 x_n for which $a_n^* = 0$ do not contribute to the solution

Karush-Kuhn-Tucker (KKT) condition:

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

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

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

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

Support Vector Machines

Hyperplanes expressed with support vectors

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

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

Support Vector Machines

To compute w_0^*:

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

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

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

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

Support Vector Machines

Instead of using one particular support vector x_k to determine w_0

$$w_0^* = t_k - \sum_{x_j \in SV} a_j^* t_j x_k^T x_j$$

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

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

Support Vector Machines

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

Classification of a new instance x'

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

Support Vector Machines

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

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

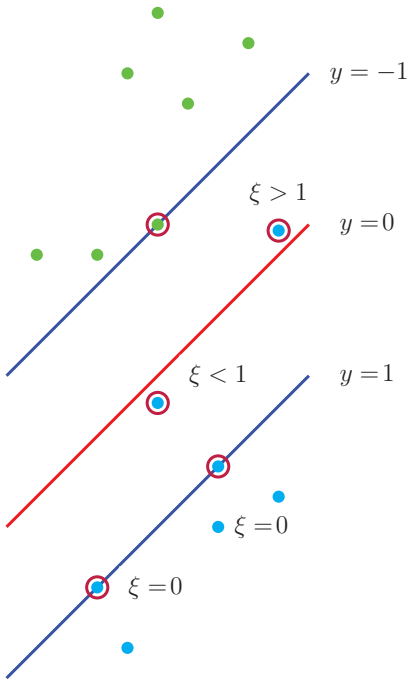
Support Vector Machines with soft margin constraints

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 $y(\mathrm{x}_n) = 0$
when $\xi_n > 1$, the sample will be mis-classified

Support Vector Machines with soft margin constraints

Soft margin constraint

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

Optimization problem with soft margin constraints

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

subject to

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

$$\xi_n \geq 0\,, \quad n = 1, \ldots, N$$

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.

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

$$w_0^* = ...$$

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

Basis functions

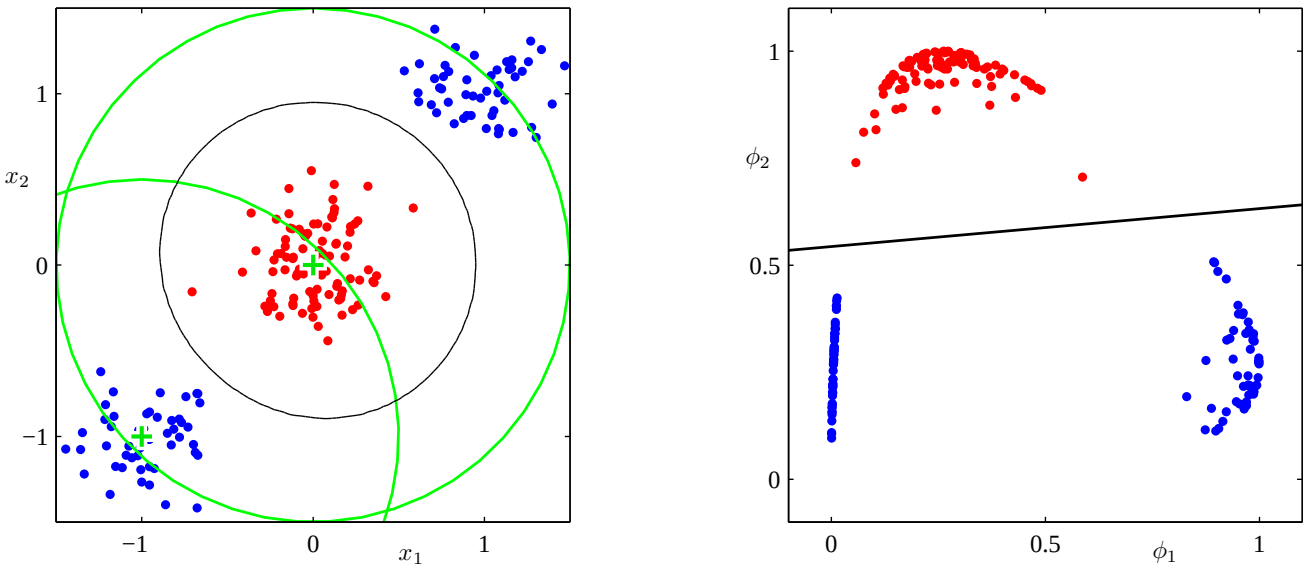
So far we considered models working directly on x.

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

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

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

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