# Linear classification

Course of Machine Learning
Master Degree in Computer Science

University of Rome “Tor Vergata”

a.a. 2020-2021

Giorgio Gambosi

# Classification

- value $t$ to predict are from a discrete domain, where each value denotes a class
- most common case: disjoint classes, each input has to assigned to exactly one class
- input space is partitioned into decision regions
- in linear classification models decision boundaries are linear functions of input $\mathbf{x}$ ($D-1$-dimensional hyperplanes in the $D$-dimensional feature space)
- datasets such as classes correspond to regions which may be separated by linear decision boundaries are said linearly separable

## Regression and classification

- Regression: the target variable $\mathbf{t}$ is a vector of reals
- Classification: several ways to represent classes (target variable values)
- Binary classification: a single variable $t \in \{0, 1\}$, where $t = 0$ denotes class $C_0$ and $t = 1$ denotes class $C_1$
- $K > 2$ classes: "1 of $K$" coding. $\mathbf{t}$ is a vector of $K$ bits, such that for each class $C_j$ all bits are 0 except the $j$-th one (which is 1)

## Approaches to classification

Three general approaches to classification

1. find $f : \mathbf{X} \mapsto \{1, \ldots, K\}$ (discriminant function) which maps each input $\mathbf{x}$ to some class $C_i$ (such that $i = f(\mathbf{x})$)
2. discriminative approach: determine the conditional probabilities $p(C_j|\mathbf{x})$ (inference phase); use these distributions to assign an input to a class (decision phase)
3. generative approach: determine the class conditional distributions $p(\mathbf{x}|C_j)$, and the class prior probabilities $p(C_j)$; apply Bayes' formula to derive the class posterior probabilities $p(C_j|\mathbf{x})$ ; use these distributions to assign an input to a class

# Discriminative approaches

- Approaches 1 and 2 are discriminative: they tackle the classification problem by deriving from the training set conditions (such as decision boundaries) that , when applied to a point, discriminate each class from the others
- The boundaries between regions are specify by *discrimination functions*

## Generalized linear models

- In linear regression, a model predicts the target value; the prediction is made through a linear function $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ (linear basis functions could be applied)
- In classification, a model predicts probabilities of classes, that is values in $[0, 1]$; the prediction is made through a generalized linear model $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$, where $f$ is a non linear activation function with codomain $[0, 1]$
- boundaries correspond to solution of $y(\mathbf{x}) = c$ for some constant $c$; this results into $w^T\mathbf{x} + w_0 = f^{-1}(c)$, that is a linear boundary. The inverse function $f^{-1}$ is said link function.

## Generative approaches

- Approach 3 is generative: it works by defining, from the training set, a model of items for each class
- The model is a probability distribution (of features conditioned by the class) and could be used for random generation of new items in the class
- By comparing an item to all models, it is possible to verify the one that best fits

# Linear discriminant functions in binary classification

- Decision boundary: $D-1$-dimensional hyperplane $y(\mathbf{x})=0$ of all points s.t. $\mathbf{w}^T\mathbf{x}+w_0=0$
- Given $\mathbf{x}_1, \mathbf{x}_2$ on the hyperplane, $y(\mathbf{x}_1)=y(\mathbf{x}_2)=0$. Hence,

$$\mathbf{w}^T(\mathbf{x}_1)-\mathbf{w}^T(\mathbf{x}_2)=\mathbf{w}^T(\mathbf{x}_1-\mathbf{x}_2)=0$$

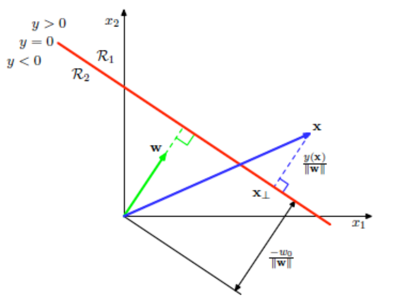
  that is, $\mathbf{x}_1-\mathbf{x}_2$, $\mathbf{w}$ orthogonal
- For any $\mathbf{x}$ s.t. $y(\mathbf{x})=0$, $\mathbf{w}^T\mathbf{x}$ is the length of the projection of $\mathbf{x}$ in the direction of $\mathbf{w}$ (orthogonal to the hyperplane $y(\mathbf{x})=0$), in multiples of $||\mathbf{w}||_2$
- By normalizing wrt to $||\mathbf{w}||_2=\sqrt{\sum_i w_i^2}$, we get the length of the projection of $\mathbf{x}$ in the direction orthogonal to the hyperplane, assuming $||\mathbf{w}||_2=1$
- Since $\mathbf{w}^T\mathbf{x}=-w_0$,

$$\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||}=-\frac{w_0}{||\mathbf{w}||}$$

  thus, the distance is determined by the threshold $w_0$

# Linear discriminant functions in binary classification

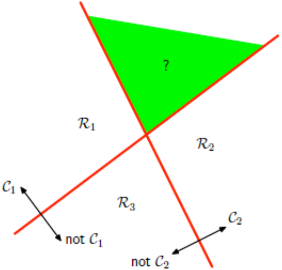
- In general, for any $\mathbf{x}$, $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ returns the distance (in multiples of $||\mathbf{w}||$) of $\mathbf{x}$ from the hyperplane
- The sign of the returned value discriminates in which of the regions separated by the hyperplane the point lies



# Linear discriminant functions in multiclass classification

## First approach

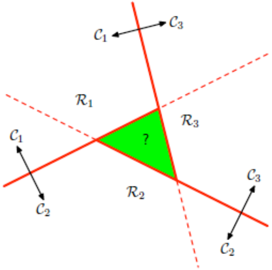
- Define $K-1$ discrimination functions
- Function $f_i$ $(1 \leq i \leq K-1)$ discriminates points belonging to class $C_i$ from points belonging to all other classes: if $f_i(\mathbf{x}) > 0$ then $\mathbf{x} \in C_i$, otherwise $\mathbf{x} \notin C_i$
- The green region belongs to both $\mathcal{R}_1$ and $\mathcal{R}_2$



# Linear discriminant functions in multiclass classification

## Second approach

- Define $K(K-1)/2$ discrimination functions, one for each pair of classes
- Function $f_{ij}$ $(1 \leq i < j \leq K)$ discriminates points which might belong to $C_i$ from points which might belong to $C_j$
- Item $\mathbf{x}$ is classified on a majority basis
- The green region is unassigned



# Linear discriminant functions in multiclass classification

## Third approach

- Define $K$ linear functions

$$y_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \qquad\qquad 1 \leq i \leq K$$

Item $\mathbf{x}$ is assigned to class $C_k$ iff $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$: that is,

$$k = \underset{j}{\operatorname{argmax}}\, y_j(\mathbf{x})$$

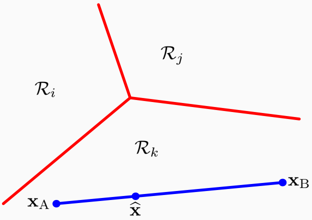
- Decision boundary between $C_i$ and $C_j$: all points $\mathbf{x}$ s.t. $y_i(\mathbf{x}) = y_j(\mathbf{x})$, a $D-1$-dimensional hyperplane

$$(\mathbf{w}_i - \mathbf{w}_j)^T\mathbf{x} + (w_{i0} - w_{j0}) = 0$$

# Linear discriminant functions in multiclass classification

The resulting decision regions are connected and convex

- Given $\mathbf{x}_A, \mathbf{x}_B \in \mathcal{R}_k$ then $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$, for all $j \neq k$
- Let $\hat{\mathbf{x}} = \lambda\mathbf{x}_A + (1-\lambda)\mathbf{x}_B$, $0 \leq \lambda \leq 1$
- For all $i$, since $y_i$ is linear for all, $y_i(\hat{\mathbf{x}}) = \lambda y_i(\mathbf{x}_a) + (1-\lambda) y_i(\mathbf{x}_B)$
- Then, $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$ for all $j \neq k$; that is, $\hat{\mathbf{x}} \in \mathcal{R}_k$



# Generalized discriminant functions

- The definition can be extended to include terms relative to products of pairs of feature values (Quadratic discriminant functions)

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=1}^{i} w_{ij} x_i x_j$$

  $\frac{d(d+1)}{2}$ additional parameters wrt the $d+1$ original ones: decision boundaries can be more complex

- In general, generalized discrimination functions through set of functions $\phi_i, \ldots, \phi_m$

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{M} w_i \phi_i(\mathbf{x})$$

# Linear discriminant functions and regression

- Assume classification with $K$ classes
- Classes are represented through a 1-of-$K$ coding scheme: set of variables $z_1, \ldots, z_K$, class $C_i$ coded by values $z_i = 1$, $z_k = 0$ for $k \neq i$
- Discriminant functions $y_i$ are derived as linear regression functions with variables $z_i$ as targets
- To each variable $z_i$ a discriminant function $y_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$ is associated: $\mathbf{x}$ is assigned to the class $C_k$ s.t.

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

- Then, $z_k(\mathbf{x}) = 1$ and $z_j(\mathbf{x}) = 0$ ($j \neq k$) if $k = \underset{i}{\operatorname{argmax}}\, y_i(\mathbf{x})$
- Group all parameters together as

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

# Linear discriminant functions and regression

- In general, a regression function provides an estimation of the target given the input $E[t|\mathbf{x}]$
- Value $y_i(\mathbf{x})$ can then be seen as an estimation of the conditional expectation $E[z_i|\mathbf{x}]$ of binary variable $z_i$ given $\mathbf{x}$
- If we assume $z_i$ is distributed according to a Bernoulli distribution, the expectation corresponds to the posterior probability

$$\begin{aligned} y_i(\mathbf{x}) &\simeq E[z_i|\mathbf{x}] \\ &= P(z_i = 1|\mathbf{x}) \cdot 1 + P(z_i = 0|\mathbf{x}) \cdot 0 \\ &= P(z_i = 1|\mathbf{x}) \\ &= P(C_i|\mathbf{x}) \end{aligned}$$

- However, $y_i(\mathbf{x})$ is not a probability itself (we may not assume it takes value only in the interval $[0, 1]$)

## Learning functions $y_i$

- Given a training set $\mathbf{X}, \mathbf{t}$, a regression function can be derived by least squares
- An item in the training set is a pair $(\mathbf{x}_i, \mathbf{t}_i)$, $\mathbf{x}_i \in \mathbb{R}^D$ e $\mathbf{t}_i \in \{0,1\}^K$
- $\mathbf{W} \in \mathbb{R}^{(D+1)\times K}$ is the matrix of parameters of all functions $y_i$: the $i$-th column represents the $D+1$ parameters $w_{i0}, \ldots, w_{iD}$ of $y_i$

$$\overline{\mathbf{W}} = \begin{pmatrix} w_{10} & w_{20} & \cdots & w_{K0} \\ w_{11} & w_{21} & \cdots & w_{K1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1D} & w_{2D} & \cdots & w_{KD} \end{pmatrix}$$

- $\mathbf{y}(\mathbf{x}) = \mathbf{W}^T\overline{\mathbf{x}}$ with $\overline{\mathbf{x}} = (1, x_1, \ldots, x_d)$

## Learning functions $y_i$

- $\overline{\mathbf{X}} \in \mathbb{R}^{n\times(D+1)}$ is the matrix of feature values for all items in the training set

$$\overline{\mathbf{X}} = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_1^{(D)} \\ 1 & x_2^{(1)} & \cdots & x_2^{(D)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{(1)} & \cdots & x_n^{(D)} \end{pmatrix}$$

- Then, for matrix $\overline{\mathbf{X}}\mathbf{W}$, of size $n \times K$, we have

$$(\overline{\mathbf{X}}\mathbf{W})_{ij} = w_{j0} + \sum_{k=1}^{D} x_i^{(k)} w_{jk} = y_j(\mathbf{x}_i)$$

# Learning functions $y_i$

- $y_j(\mathbf{x}_i)$ is compared to item $\mathbf{T}_{ij}$ in the matrix $\mathbf{T}$, of size $n \times K$, of target values, where row $i$ is the 1-of-$K$ coding of the class of item $\mathbf{x}_i$

$$(\overline{\mathbf{X}}\mathbf{W} - \mathbf{T})_{ik} = y_k(\mathbf{x}_i) - t_{ik}$$

- Let us consider the diagonal items of $(\overline{\mathbf{X}}\mathbf{W} - \mathbf{T})^T(\overline{\mathbf{X}}\mathbf{W} - \mathbf{T})$. Then,

$$((\overline{\mathbf{X}\mathbf{W}} - \mathbf{T})^T(\overline{\mathbf{X}\mathbf{W}} - \mathbf{T}))_{kk} = \sum_{i=1}^{n}(y_k(\mathbf{x}_i) - t_{ik})^2$$

That is,

$$((\overline{\mathbf{X}}\mathbf{W} - \mathbf{T})^T(\overline{\mathbf{X}}\mathbf{W} - \mathbf{T}))_{kk} = \sum_{\mathbf{x}_i \in C_k}(y_k(\mathbf{x}_i) - 1)^2 + \sum_{\mathbf{x}_i \notin C_k} y_k(\mathbf{x}_i)^2$$

# Learning functions $y_i$

- Summing all elements on the diagonal of $(\overline{\mathbf{X}}\mathbf{W}-\mathbf{T})^T(\overline{\mathbf{X}}\mathbf{W}-\mathbf{T})$ provides the overall sum, on all items in the training set, of the squared differences between observed values and values computed by the model, with parameters $\mathbf{W}$, that is

$$\sum_{k=1}^{K}\sum_{i=1}^{n}(y_k(\mathbf{x}_i)-t_{ik})^2$$

- This corresponds to the *trace* of $(\overline{\mathbf{X}}\mathbf{W}-\mathbf{T})^T(\overline{\mathbf{X}}\mathbf{W}-\mathbf{T})$. Hence, we have to minimize:

$$E(\mathbf{W})=\frac{1}{2}tr((\overline{\mathbf{X}}\mathbf{W}-\mathbf{T})^T(\overline{\mathbf{X}}\mathbf{W}-\mathbf{T}))$$

- Standard approach, solve

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{W}}=\mathbf{0}$$

# Learning functions $y_i$

- It is possible to show that

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} = \overline{\mathbf{X}}^T\overline{\mathbf{X}}\mathbf{W} - \overline{\mathbf{X}}^T\mathbf{T}$$

- From $\overline{\mathbf{X}}^T\overline{\mathbf{X}}\mathbf{W} - \overline{\mathbf{X}}^T\mathbf{T} = \mathbf{0}$ it results

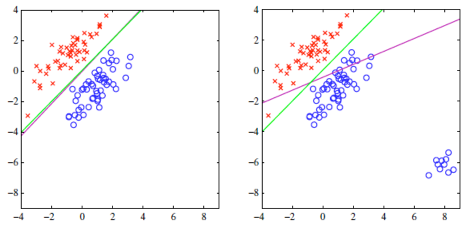
$$\mathbf{W} = (\overline{\mathbf{X}}^T\overline{\mathbf{X}})^{-1}\overline{\mathbf{X}}^T\mathbf{T}$$

- and the set of discriminant functions

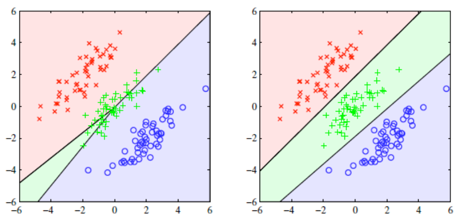
$$\mathbf{y}(\mathbf{x}) = \mathbf{W}^T\overline{\mathbf{x}} = \mathbf{T}^T\overline{\mathbf{X}}(\overline{\mathbf{X}}^T\overline{\mathbf{X}})^{-1}\overline{\mathbf{x}}$$

# Some considerations

- Simple learning: closed form
- quite prone to outliers (magenta, this approach; green, logistic regression)



- poor precision for $K > 2$ (left, this approach; right, logistic regression)



# Fisher' linear discriminant

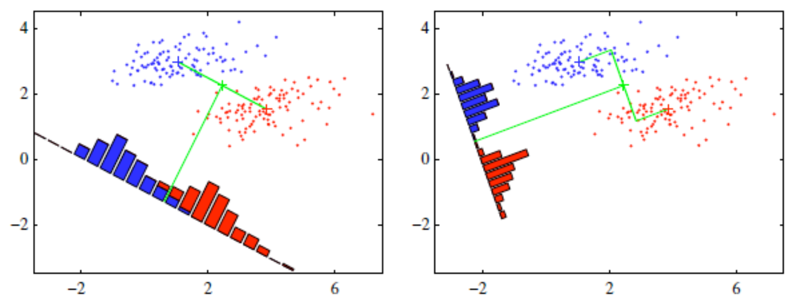
- The idea of *Linear Discriminant Analysis* (*LDA*) is to find a linear projection of the training set into a suitable subspace where classes are as linearly separated as possible
- A common approach is provided by Fisher linear discriminant, where all items in the training set (points in a $D$-dimensional space) are projected to one dimension, by means of a linear transformation of the type

$$y = \mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w}$ is the $D$-dimensional vector corresponding to the direction of projection (in the following, we will consider the one with unit norm).

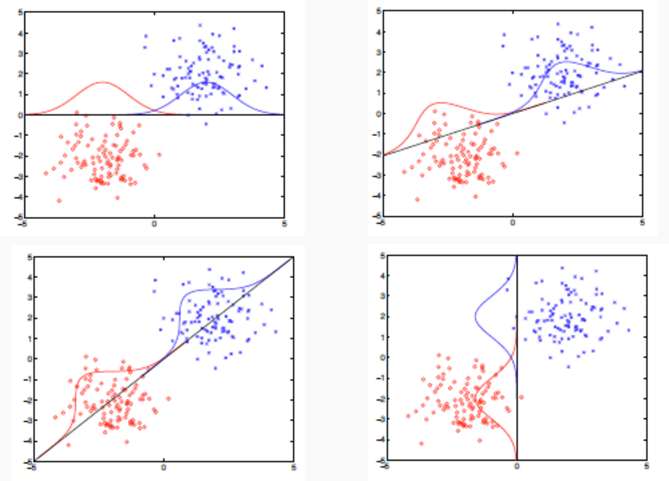
# LDA

If $K = 2$, given a threshold $\tilde{y}$, item $\mathbf{x}$ is assigned to $C_1$ iff its projection $y = \mathbf{w}^T\mathbf{x}$ is such that $y > \tilde{y}$; otherwise, $\mathbf{x}$ is assigned to $C_2$.



# LDA

Different line directions, that is different parameters $\mathbf{w}$, may induce quite different separability properties.



## Deriving $\mathbf{w}$ in the binary case

Let $n_1$ be the number of items in the training set belonging to class $C_1$ and $n_2$ the number of items in class $C_2$. The mean points of both classes are

$$\mathbf{m}_1 = \frac{1}{n_1} \sum_{\mathbf{x} \in C_1} \mathbf{x} \qquad\qquad \mathbf{m}_2 = \frac{1}{n_2} \sum_{\mathbf{x} \in C_2} \mathbf{x}$$

A simple measure of the separation of classes, when the training set is projected onto a line, is the difference between the projections of their mean points

$$m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$$

where $m_i = \mathbf{w}^T\mathbf{m}_i$ is the projection of $\mathbf{m}_i$ onto the line.

## Deriving $\mathbf{w}$ in the binary case

- We wish to find a line direction $\mathbf{w}$ such that $m_2 - m_1$ is maximum
- $\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$ can be made arbitrarily large by multiplying $\mathbf{w}$ by a suitable constant, at the same time maintaining the direction unchanged. To avoid this drawback, we consider unit vectors, introducing the constraint $||\mathbf{w}||_2 = \mathbf{w}^T\mathbf{w} = 1$
- This results into the constrained optimization problem

$$\max_{\mathbf{w}} \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$$
$$\text{where } \mathbf{w}^T\mathbf{w} = 1$$

- This can be transformed into an equivalent unconstrained optimization problem by means of lagrangian multipliers

$$\max_{\mathbf{w},\lambda} \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})$$

# Deriving $\mathbf{w}$ in the binary case

Setting the gradient of the function wrt $\mathbf{w}$ to $\mathbf{0}$

$$\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})) = \mathbf{m}_2 - \mathbf{m}_1 + 2\lambda\mathbf{w} = \mathbf{0}$$

results into

$$\mathbf{w} = \frac{\mathbf{m}_2 - \mathbf{m}_1}{2\lambda}$$

# Deriving $\mathbf{w}$ in the binary case

Setting the derivative wrt $\lambda$ to 0

$$\frac{\partial}{\partial\lambda}(\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \mathbf{w}^T\mathbf{w})) = 1 - \mathbf{w}^T\mathbf{w} = 0$$

results into

$$1 - \mathbf{w}^T\mathbf{w} = 1 - \frac{(\mathbf{m}_2 - \mathbf{m}_1)^T(\mathbf{m}_2 - \mathbf{m}_1)}{4\lambda^2} = 0$$

that is

$$\lambda = \frac{\sqrt{(\mathbf{m}_2 - \mathbf{m}_1)^T(\mathbf{m}_2 - \mathbf{m}_1)}}{2} = \frac{||\mathbf{m}_2 - \mathbf{m}_1||_2}{2}$$

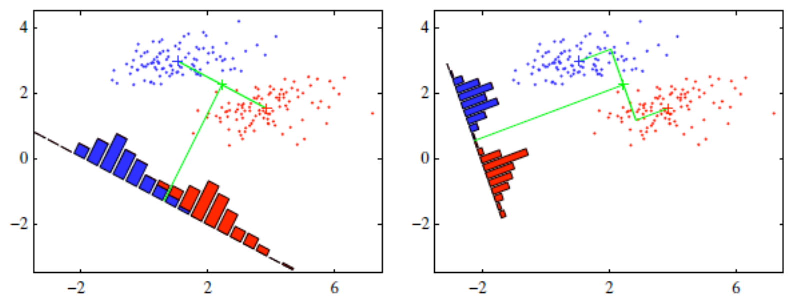
Combining with the result for the gradient, we get

$$\mathbf{w} = \frac{\mathbf{m}_2 - \mathbf{m}_1}{||\mathbf{m}_2 - \mathbf{m}_1||_2}$$

# Deriving $\mathbf{w}$ in the binary case

The best direction $\mathbf{w}$ of the line, wrt the measure considered, is the one from $\mathbf{m}_1$ to $\mathbf{m}_2$.

However, this may result in a poor separation of classes.




Projections of classes are dispersed (high variance) along the direction of $\mathbf{m}_1 - \mathbf{m}_2$. This may result in a large overlap.

# Deriving $\mathbf{w}$ in the binary case: refinement

- Choose directions s.t. classes projections show as little dispersion as possible
- Possible in the case that the amount of class dispersion changes wrt different directions, that is if the distribution of points in the class is elongated
- We wish then to maximize a function which:
  - is growing wrt the separation between the projected classes (for example, their mean points)
  - is decreasing wrt the dispersion of the projections of points of each class

# Deriving $\mathbf{w}$ in the binary case: refinement

- The within-class variance of the projection of class $C_i$ $(i = 1, 2)$ is defined as

$$s_i^2 = \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - m_i)^2$$

The total within-class variance is defined as $s_1^2 + s_2^2$

- Given a direction $\mathbf{w}$, the Fisher criterion is the ratio between the (squared) class separation and the overall within-class variance, along that direction

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Indeed, $J(\mathbf{w})$ grows wrt class separation and decreases wrt within-class variance

# Deriving $\mathbf{w}$ in the binary case: refinement

Let $\mathbf{S}_1, \mathbf{S}_2$ be the within-class covariance matrices, defined as

$$\mathbf{S}_i = \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T$$

Then,

$$\begin{aligned}
s_i^2 &= \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - m_i)^2 = \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_i)^2 \\
&= \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_i)(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_i) \\
&= \sum_{\mathbf{x}\in C_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_i)(\mathbf{x}^T\mathbf{w} - \mathbf{m}_i^T\mathbf{w}) \\
&= \sum_{\mathbf{x}\in C_i} \left(\mathbf{w}^T(\mathbf{x}-\mathbf{m}_i)\right)\left((\mathbf{x}-\mathbf{m}_i)^T\mathbf{w}\right) \\
&= \sum_{\mathbf{x}\in C_i} \mathbf{w}^T(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T\mathbf{w} \\
&= \mathbf{w}^T\left(\sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T\right)\mathbf{w} = \mathbf{w}^T\mathbf{S}_i\mathbf{w}
\end{aligned}$$

# Deriving $\mathbf{w}$ in the binary case: refinement

Let also $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$ be the total within-class covariance matrix and

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

be the between-class covariance matrix.

Then,

$$\begin{aligned}
J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} &= \frac{(\mathbf{w}^T\mathbf{m}_2 - \mathbf{w}^T\mathbf{m}_1)^2}{\mathbf{w}^T\mathbf{S}_1\mathbf{w} + \mathbf{w}^T\mathbf{S}_2\mathbf{w}} \\
&= \frac{(\mathbf{w}^T\mathbf{m}_2 - \mathbf{w}^T\mathbf{m}_1)(\mathbf{w}^T\mathbf{m}_2 - \mathbf{w}^T\mathbf{m}_1)}{\mathbf{w}^T\mathbf{S}_1\mathbf{w} + \mathbf{w}^T\mathbf{S}_2\mathbf{w}} \\
&= \frac{\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}}{\mathbf{w}^T\mathbf{S}_1\mathbf{w} + \mathbf{w}^T\mathbf{S}_2\mathbf{w}} \\
&= \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}
\end{aligned}$$

# Deriving $\mathbf{w}$ in the binary case: refinement

As usual, $J(\mathbf{w})$ is maximized wrt $\mathbf{w}$ by setting its gradient to $\mathbf{0}$

$$\frac{\partial}{\partial \mathbf{w}}\frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} = 2\frac{(\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} - (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}}{(\mathbf{w}^T\mathbf{S}_W\mathbf{w})(\mathbf{w}^T\mathbf{S}_W\mathbf{w})^T}$$

which results into

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

# Deriving $\mathbf{w}$ in the binary case: refinement

Observe that:

- $\mathbf{w}^T\mathbf{S}_B\mathbf{w}$ is a scalar, say $c_B$
- $\mathbf{w}^T\mathbf{S}_W\mathbf{w}$ is a scalar, say $c_W$
- $(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w}$ is a scalar, say $c_m$

Then, the condition $(\mathbf{w}^T\mathbf{S}_B\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$ can be written as

$$c_B\mathbf{S}_W\mathbf{w} = c_W\mathbf{S}_B\mathbf{w} = c_W(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T\mathbf{w} = c_W(\mathbf{m}_2 - \mathbf{m}_1)c_m$$

which results into

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

Since we are interested into the direction of $\mathbf{w}$, that is in any vector proportional to $\mathbf{w}$, we may consider the solution

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

# Deriving $\mathbf{w}$ in the binary case: choosing a threshold

Possible approach:

- model $p(y|C_i)$ as a gaussian: derive mean and variance by maximum likelihood

$$m_i = \frac{1}{n_i}\sum_{\mathbf{x}\in C_i} w^T\mathbf{x} \qquad \sigma_i^2 = \frac{1}{n_i-1}\sum_{\mathbf{x}\in C_i}(w^T\mathbf{x}-m_i)^2$$

  where $n_i$ is the number of items in training set belonging to class $C_i$
- derive the class probabilities

$$p(C_i|y) \propto p(y|C_i)p(C_i) = p(y|C_i)\frac{n_i}{n_1+n_2} \propto n_i e^{-\frac{(y-m_i)^2}{2\sigma_i^2}}$$

- the threshold $\tilde{y}$ can be derived as the minimum $y$ such that

$$\frac{p(C_2|y)}{p(C_1|y)} = \frac{n_2}{n_1}\frac{p(y|C_2)}{p(y|C_1)} > 1$$

# Perceptron

- Introduced in the '60s, at the basis of the neural network approach
- Simple model of a single neuron
- Hard to evaluate in terms of probability
- Works only in the case that classes are linearly separable

# Definition

It corrisponds to a binary classification model where an item $\mathbf{x}$ is first transformed by a non linear function $\phi$ and the classified on the basis of the sign of the obtained value. That is,

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

$f()$ is essentially the sign function

$$f(i) = \begin{cases} -1 & \text{if } i < 0 \\ 1 & \text{if } i \geq 0 \end{cases}$$

The resulting model is a particular generalized linear model. A special case is the one when $\phi$ is the identity, that is $y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x})$.
By the definition of the model, $y(\mathbf{x})$ can only be $\pm 1$: we denote $y(\mathbf{x}) = 1$ as $\mathbf{x} \in C_1$ and $y(\mathbf{x}) = -1$ as $\mathbf{x} \in C_2$.

To each element $\mathbf{x}_i$ in the training set, a target value is then associated $t_i \in \{-1, 1\}$.

# Cost function

- A natural definition of the cost function would be the number of misclassified elements in the training set
- This would result into a piecewise constant function and gradient optimization could not be applied (we would have zero gradient almost everywhere)
- A better choice is using a piecewise linear function as cost function

# Cost function

We would like to find a vector of parameters $\mathbf{w}$ such that, for any $\mathbf{x}_i$, $\mathbf{w}^T\mathbf{x}_i > 0$ if $\mathbf{x}_i \in C_1$ and $\mathbf{w}^T\mathbf{x}_i < 0$ if $\mathbf{x}_i \in C_2$: in short, $\mathbf{w}^T\mathbf{x}_i t_i > 0$.

Each element $\mathbf{x}_i$ provides a contribution to the cost function as follows

1. $0$ if $\mathbf{x}_i$ is classified correctly by the model
2. $-\mathbf{w}^T\mathbf{x}_i t_i > 0$ if $\mathbf{x}_i$ is misclassified

Let $\mathcal{M}$ be the set of misclassified elements. Then the cost is

$$E_p(\mathbf{w}) = -\sum_{\mathbf{x}_i \in \mathcal{M}} \mathbf{w}^T\phi(\mathbf{x}_i)t_i$$

The contribution of $\mathbf{x}_i$ to the cost is 0 if $\mathbf{x}_i \notin \mathcal{M}$ and it is a linear function of $\mathbf{w}$ otherwise

# Gradient optimization

The minimum of $E_p(\mathbf{w})$ can be found through gradient descent

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \frac{\partial E_p(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(k)}}$$

the gradient of the cost function wrt to $\mathbf{w}$ is

$$\frac{\partial E_p(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{\mathbf{x}_i \in \mathcal{M}} \phi(\mathbf{x}_i) t_i$$

Then gradient descent can be expressed as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \sum_{\mathbf{x}_i \in \mathcal{M}_k} \phi(\mathbf{x}_i) t_i$$

where $\mathcal{M}_k$ denotes the set of points misclassified by the model with parameter $\mathbf{w}^{(k)}$

## Gradient optimization

Online (or stochastic gradient descent): at each step, only the gradient wrt a single item is considered

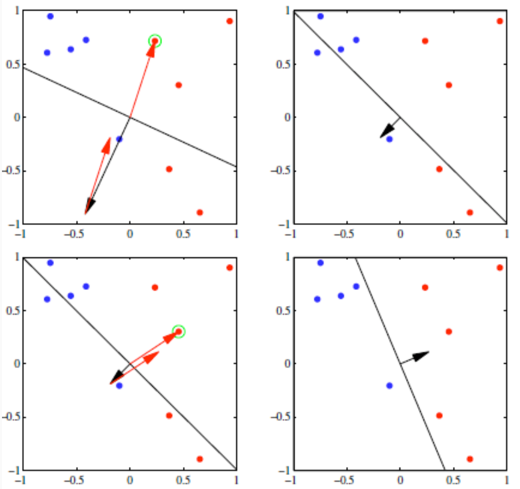
$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta\phi(\mathbf{x}_i)t_i$$

where $\mathbf{x}_i \in \mathcal{M}_k$ and the *scale factor* $\eta > 0$ controls the impact of a badly classified item on the cost function
The method works by circularly iterating on all elements and applying the above formula.

# Gradient optimization

Initialize $\mathbf{w}^0$
$k := 0$ **repeat**
  $k := k + 1$
  $i := (k \bmod n) + 1$
  $y := f(\mathbf{w}^T \phi(\mathbf{x}_i))t_i$
  **if** $y > 0$ **then** $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)}$
  **else** $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta\phi(\mathbf{x}_i)t_i$
**until** all elements are well classified

# Gradient optimization




In black, decision boundary and corresponding parameter vector $\mathbf{w}$; in red misclassified item vector $\phi(\mathbf{x}_i)$, added by the algorithm to the parameter vector as $\eta\phi(\mathbf{x}_i)$

## Gradient optimization

At each step, if $\mathbf{x}_i$ is well classified then $\mathbf{w}^{(k)}$ is unchanged; else, its contirbution to the cost is modified as follows

$$
\begin{aligned}
-(\mathbf{w}^{(k+1)})^T\phi(\mathbf{x}_i)t_i &= -(\mathbf{w}^{(k)})^T\phi(\mathbf{x}_i)t_i - \eta(\phi(\mathbf{x}_i)t_i)^T\phi(\mathbf{x}_i)t_i \\
&= -(\mathbf{w}^{(k)})^T\phi(\mathbf{x}_i)t_i - \eta||\phi(\mathbf{x}_i)||^2 \\
&< -(\mathbf{w}^{(k)})^T\phi(\mathbf{x}_i)t_i
\end{aligned}
$$

This contribution is decreasing, however this does not guarantee the convergence of the method, since the cost function could increase due to some other element becoming misclassified if $\mathbf{w}^{(k+1)}$ is used

## Perceptron convergence theorem

It is possible to prove that, in the case the classes are linearly separable, the algorithm converges to the correct solution in a finite number of steps.

Let $\hat{\mathbf{w}}$ be a solution (that is, it discriminates $C_1$ and $C_2$): if $\mathbf{x}_{k+1}$ is the element considered at iteration $(k+1)$ and it is misclassified, then

$$\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}} = (\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}) + \eta\phi(\mathbf{x}_{k+1})t_{k+1}$$

where $\alpha > 0$ is a constant, to be specified later

# Perceptron convergence theorem

By squaring left and right expressions of the above formula, we get

$$
\begin{aligned}
&\left|\left|\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}}\right|\right|^2 = \\
&\left|\left|\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}\right|\right|^2 + \eta^2 \left|\left|\phi(\mathbf{x}_{k+1})\right|\right|^2 + 2\eta(\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}})^T\phi(\mathbf{x}_{k+1})t_{k+1} = \\
&\left|\left|\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}\right|\right|^2 + \eta^2 \left|\left|\phi(\mathbf{x}_{k+1})\right|\right|^2 + 2\eta(\mathbf{w}^{(k)})^T\phi(\mathbf{x}_{k+1})t_{k+1} - 2\eta\alpha\hat{\mathbf{w}}^T\phi(\mathbf{x}_{k+1})t_{k+1}
\end{aligned}
$$

Since $\mathbf{x}_{k+1}$ was misclassified by hypothesis, $(\mathbf{w}^{(k)})^T\phi(\mathbf{x}_{k+1})t_{k+1} < 0$ and

$$
\left|\left|\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}}\right|\right|^2 < \left|\left|\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}\right|\right|^2 + \eta^2 \left|\left|\phi(\mathbf{x}_{k+1})\right|\right|^2 - 2\eta\alpha\hat{\mathbf{w}}^T\phi(\mathbf{x}_{k+1})t_{k+1}
$$

# Perceptron convergence theorem

Let $\gamma$ be the minimum value of the signed dot product of $\hat{\mathbf{w}}$ with $\phi(\mathbf{x}_i)$ for some element $\mathbf{x}_i$, where the sign depends on the class of $\mathbf{x}_i$

$$\gamma = \min_i \left(\hat{\mathbf{w}}^T \phi(\mathbf{x}_i) t_i\right) = \min_i |\hat{\mathbf{w}}^T \phi(\mathbf{x}_i)| > 0$$

Let $\delta$ be the length of the longest $\phi(\mathbf{x}_i)$

$$\delta^2 = \max_i \, ||\phi(\mathbf{x}_i)||^2$$

Then,

$$\left|\left|\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}}\right|\right|^2 < \left|\left|\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}\right|\right|^2 + \eta^2\delta^2 - 2\eta\alpha\gamma$$

## Perceptron convergence theorem

By setting

$$\alpha = \frac{\eta\delta^2}{\gamma}$$

we get

$$\left|\left|\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}}\right|\right|^2 < \left|\left|\mathbf{w}^{(k)} - \alpha\hat{\mathbf{w}}\right|\right|^2 - \eta^2\delta^2$$

As can be seen, the squared distance between $\mathbf{w}^{(k+1)}$ and $\hat{\mathbf{w}}$ decreases at each step of an amount greater than $\eta^2\delta^2$

# Perceptron convergence theorem

Iterating the above properties on all steps,

$$\left|\left|\mathbf{w}^{(k+1)} - \alpha\hat{\mathbf{w}}\right|\right|^2 < \left|\left|\mathbf{w}^{(0)} - \alpha\hat{\mathbf{w}}\right|\right|^2 - (k+1)\eta^2\delta^2$$

Note that, after

$$\overline{k} = \frac{\left|\left|\mathbf{w}^{(0)} - \alpha\hat{\mathbf{w}}\right|\right|^2}{\eta^2\delta^2} - 1$$

steps we get

$$\left|\left|\mathbf{w}^{(0)} - \alpha\hat{\mathbf{w}}\right|\right|^2 - (k+1)\eta^2\delta^2 = 0$$

So, after at most $\overline{k}$ updates of $\mathbf{w}$, a decision boundary has been derived

## Perceptron convergence theorem

Setting $\mathbf{w}^{(0)} = \mathbf{0}$, we have

$$\overline{k} = \frac{\alpha^2}{\eta^2\delta^2}\,||\hat{\mathbf{w}}||^2 - 1 = \frac{\delta^2}{\gamma^2}\,||\hat{\mathbf{w}}||^2 - 1 = \frac{\max\limits_i\,||\phi(\mathbf{x}_i)||^2}{(\min\limits_i\,(\hat{\mathbf{w}}^T\phi(\mathbf{x}_i)))^2}\,||\hat{\mathbf{w}}||^2 - 1$$

The number of required step is large if $\min\limits_i\,(\hat{\mathbf{w}}^T\phi(\mathbf{x}_i))$ is small, that is if there exists some $\mathbf{x}_i$ such that $\phi(\mathbf{x}_i)$ is (almost) orthogonal to $\hat{\mathbf{w}}$.