Linear Discriminants

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professor : Christian Gagné

Week 5

UNIVERSITÉ LAVAL

5.1 Discriminative models

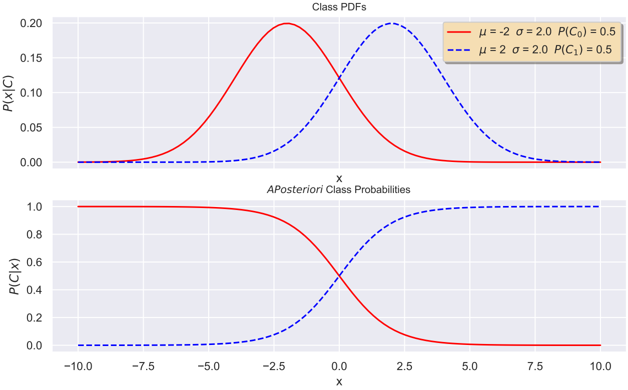
Generative and discriminative models

- Generative classification models
  - Likelihood-based classification (probability densities)

$$\mathrm{h}_i(\mathbf{x}) = \log \hat{P}(C_i|\mathbf{x})$$

  - Parametric (including mixture models) and nonparametric approaches
- Discriminative models
  - Philosophy: to solve only the problem of discrimination, the estimation of densities is an unnecessary step
  - Obtaining a discriminant function $\mathrm{h}_i(\mathbf{x}|\Phi_i)$ according to a parametrization Φ_i
- "When solving a given problem, try to avoid solving a more general problem as an intermediary step." (Vladimir Vapnik)

Generative and discriminative models



- Generative model: if $P(C_1|x) \geq P(C_0|x)$ then C_1, otherwise C_0
- Discriminative model: if $x \geq 0$ then C_1, otherwise C_0

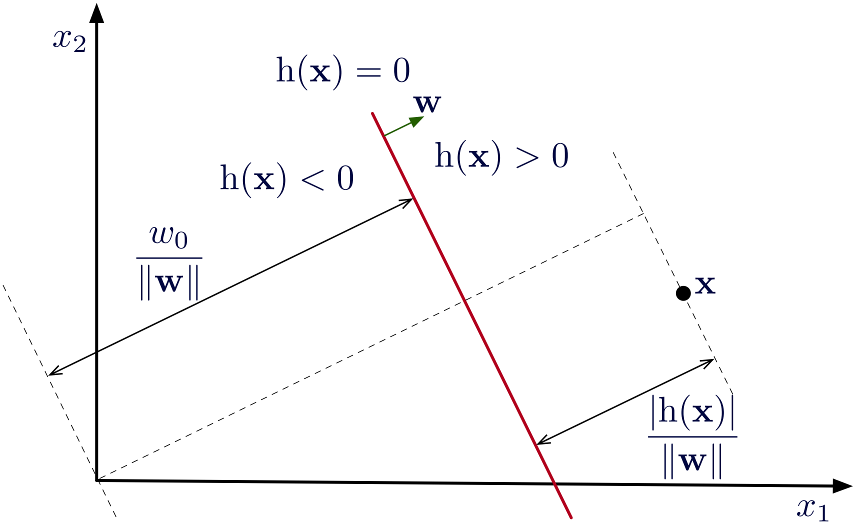
Linear discriminants

- Equation of a linear discriminant

$$\mathrm{h}_i(\mathbf{x}|\mathbf{w}_i, w_{i,0}) = \sum_{j=1}^{D} w_{i,j} x_j + w_{i,0}$$

- Two-class model
  - Only one equation $\mathrm{h}(\mathbf{x}|\mathbf{w}, w_0)$
  - $\mathbf{x}$ belongs to C_1 if $\mathrm{h}(\mathbf{x}) \geq 0$
  - Otherwise (when $\mathrm{h}(\mathbf{x}) < 0$) $\mathbf{x}$ belongs to C_2.
  - Weight $\mathbf{w}$ determines the orientation of the separating hyperplane
  - Bias w_0 determines the position of the separating hyperplane in the input space

Linear discriminants geometry



5.2 Perceptron

Perceptron

- Perceptron
  - Proposed in 1957 by Rosenblatt
  - Considered as the simplest neural network
  - The class is assigned according to the sign of the discriminant function $\mathrm{h}(\mathbf{x}|\mathbf{w}, w_0)$

$$\mathrm{h}(\mathbf{x}|\mathbf{w}, w_0) = \mathbf{w}^\top \mathbf{x} + w_0, \quad \mathbf{x} \in \begin{cases} C_1 & \text{if } \mathrm{h}(\mathbf{x}|\mathbf{w}, w_0) \geq 0 \\ C_2 & \text{otherwise} \end{cases}$$

- Optimization based on the perceptron criterion $(r^t \in \{-1, 1\})$

$$E_{percp}(\mathbf{w}, w_0|\mathcal{X}) = -\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t \mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0)$$

  - $\mathcal{Y}$ represents the data of $\mathcal{X}$ misclassified by $\mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0)$

$$\mathcal{Y} = \{\mathbf{x}^t \in \mathcal{X} \mid r^t \mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0) \leq 0\}$$

Gradient descent

- Iterative minimization of an error criterion $E(\mathbf{w}, w_0|\mathcal{X})$ based on a dataset $\mathcal{X}$

$$\{\mathbf{w}^*, w_0^*\} = \operatorname*{argmin}_{\{\mathbf{w}, w_0\}} E(\mathbf{w}, w_0|\mathcal{X})$$

- Resolution with partial derivatives, $\nabla_w E$

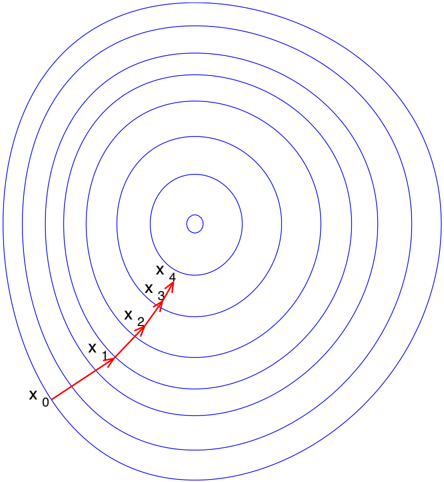
$$\nabla_w E = \left[\frac{\partial E}{\partial w_0} \quad \frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \quad \cdots \quad \frac{\partial E}{\partial w_D}\right]$$

- Modification of the weights w_i in the direction opposite to the gradient (gradient descent)

$$w_i = w_i + \Delta w_i, \quad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \quad i = 0, \ldots, D$$

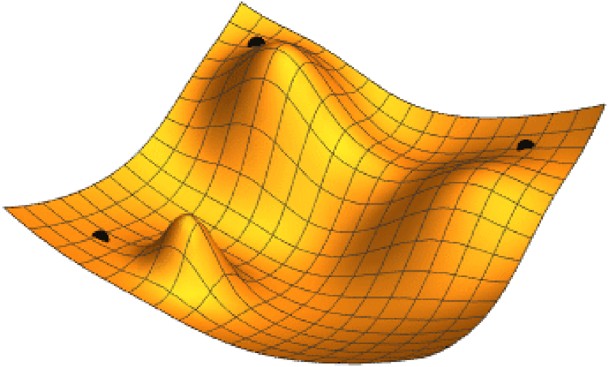
- $\eta \in [0, 1]$ is the step size or *learning rate*

Gradient descent



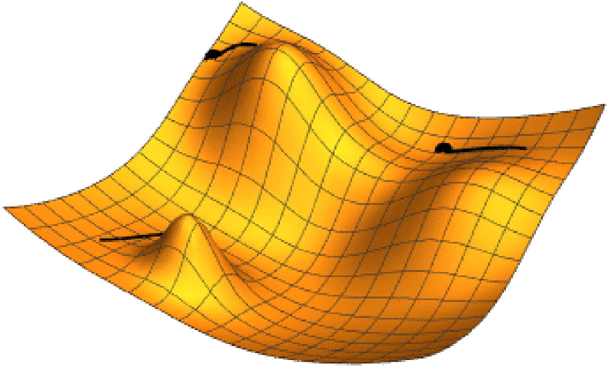
Public domain, `https://commons.wikimedia.org/wiki/File:Gradient_descent.svg`

Gradient descent



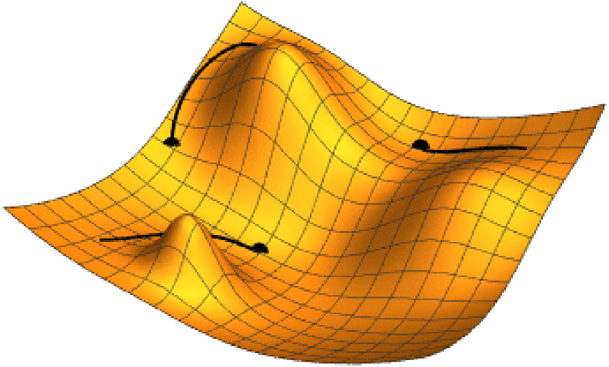
Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent



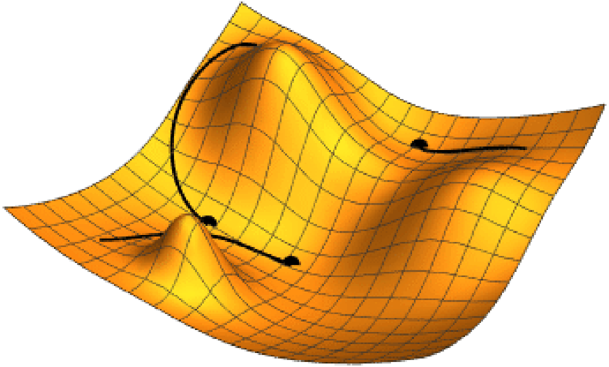
Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent



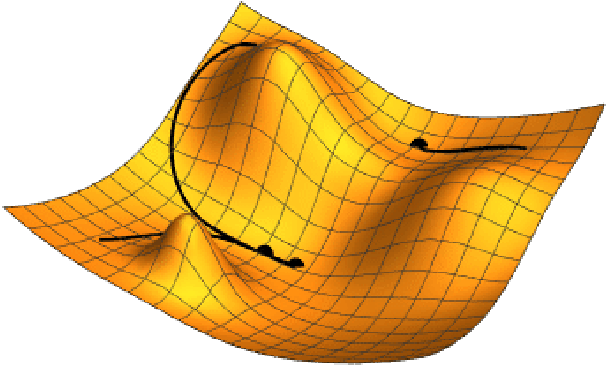
Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent



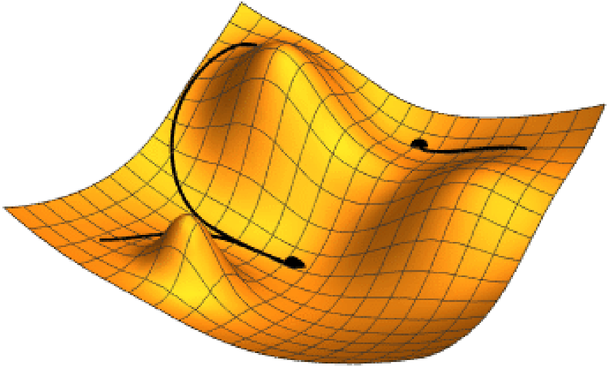
Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent



Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent



Public domain, https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Gradient descent with perceptron

- Perceptron error criterion

$$E_{percp}(\mathbf{w}, w_0 | \mathcal{X}) = -\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)$$

  - $\mathcal{Y}$ is the dataset from $\mathcal{X}$ misclassified by $\mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)$
- Calculating the gradient $\nabla E_{percp}(\mathbf{w}, w_0 | \mathcal{X})$

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial(-\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t(\mathbf{w}^\top \mathbf{x} + w_0))}{\partial w_i} = -\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t x_i^t \\ \frac{\partial E}{\partial w_0} &= \frac{\partial(-\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t(\mathbf{w}^\top \mathbf{x} + w_0))}{\partial w_0} = -\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t \end{aligned}$$

- Gradient descent $w_i = w_i + \Delta w_i$, $i = 0, \ldots, D$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{\mathbf{x}^t \in \mathcal{Y}} r^t x_i^t, \quad \Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_{\mathbf{x}^t \in \mathcal{Y}} r^t$$

Perceptron algorithm

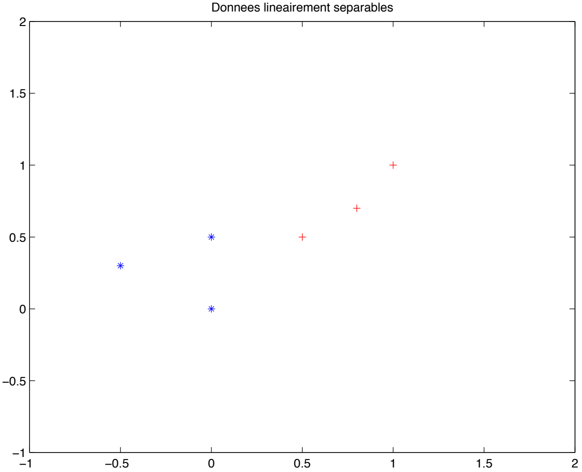
1. Initialize the weights $\mathbf{w}$ and w_0 arbitrarily

$$w_j = 0, \quad j = 0, \ldots, D$$

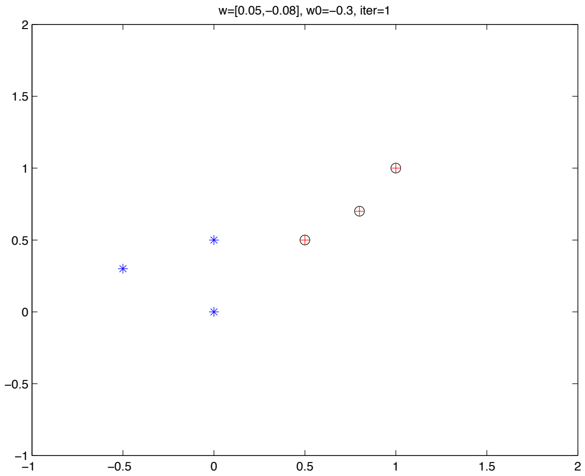
2. Repeat until convergence or depletion of resources:

$$\begin{aligned} w_j &= w_j + \eta \sum_{\mathbf{x}^t \in \mathcal{Y}} r^t x_j^t, \quad j = 1, \ldots, D \\ w_0 &= w_0 + \eta \sum_{\mathbf{x}^t \in \mathcal{Y}} r^t \end{aligned}$$

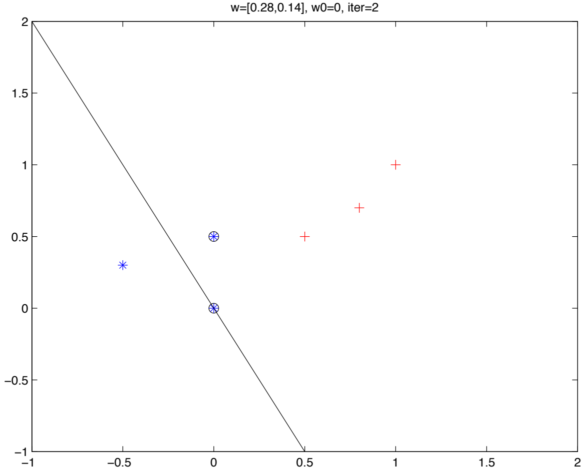
Example with the perceptron (linearly separable)



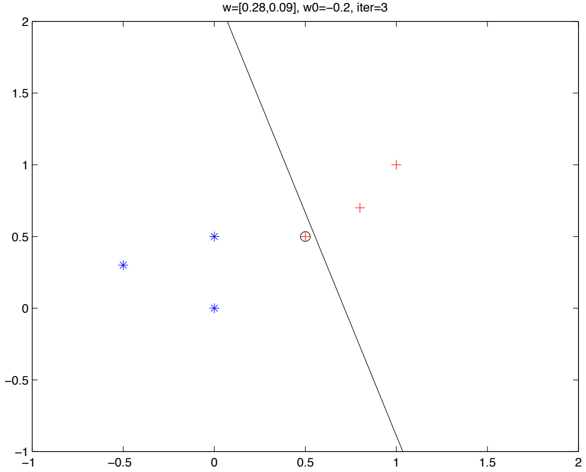
Example with the perceptron (linearly separable)



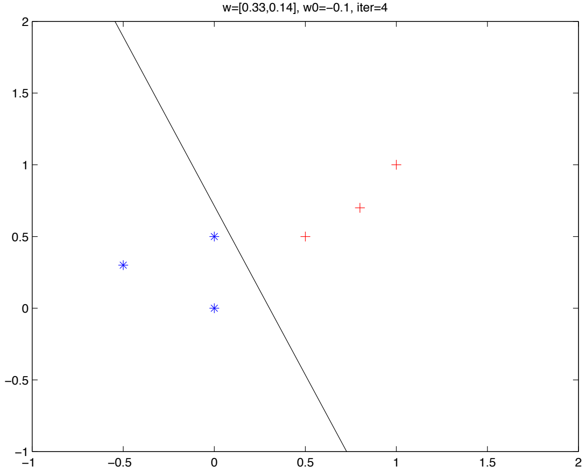
Example with the perceptron (linearly separable)



Example with the perceptron (linearly separable)



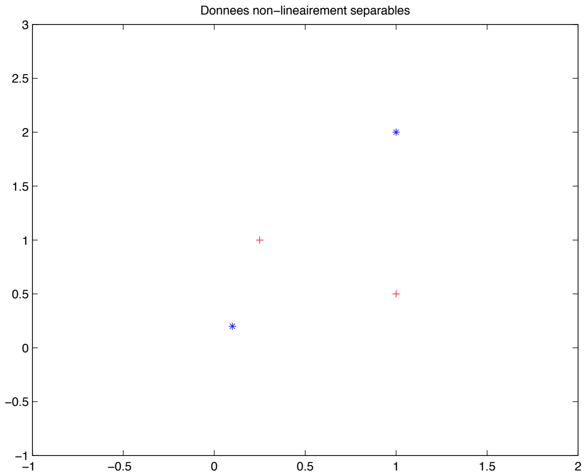
Example with the perceptron (linearly separable)



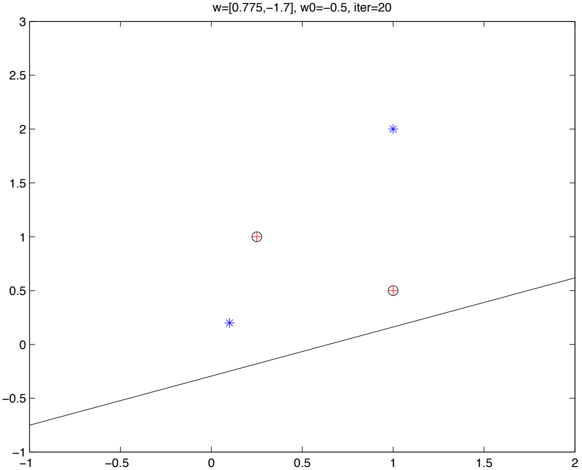
Perceptron convergence

- Convergence on linearly separable data
  - Mathematical proof of convergence exists for linearly separable data
  - Convergence towards any position of the discriminant that separates the data
  - For non-linearly separable data, no convergence
- Error criterion weakly related to the nature of the errors

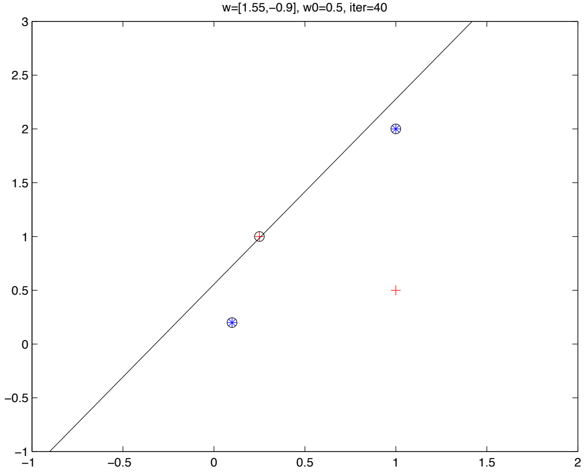
Example with the perceptron (non-linearly separable)



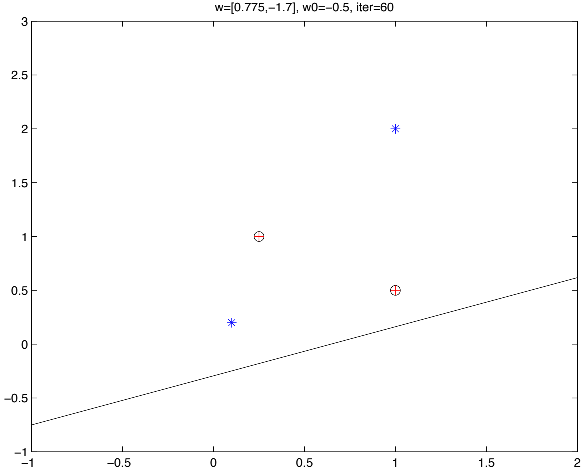
Example with the perceptron (non-linearly separable)



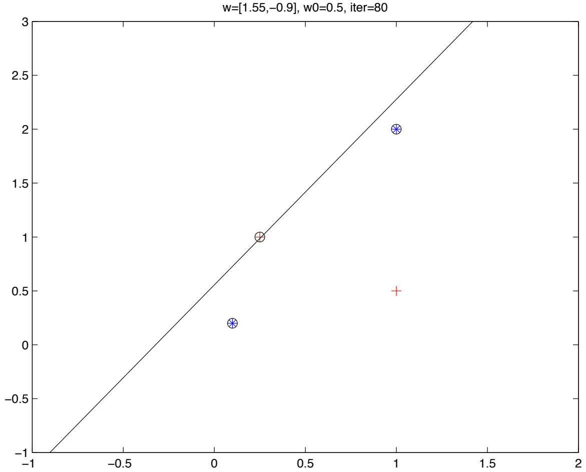
Example with the perceptron (non-linearly separable)



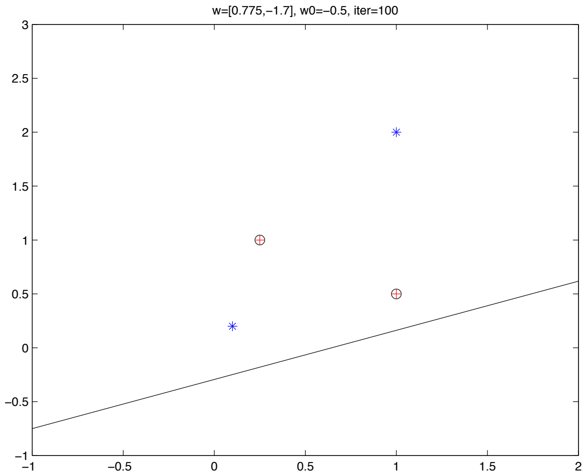
Example with the perceptron (non-linearly separable)



Example with the perceptron (non-linearly separable)



Example with the perceptron (non-linearly separable)



5.3 Least squares method

Regression for classification

- In regression, there is a stronger feedback on the nature of the errors
  - Fine differences between target values r^t and values obtained by $\mathrm{h}(\mathbf{x}^t)$
  - Target values $r^t \in \mathbb{R}$ in regression are more general than discrete target values $r^t \in \{-1, 1\}$ in classification
  - Least squares error for linear regression

$$E_{quad}(\mathbf{w}, w_0|\mathcal{X}) = \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - \mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0))^2$$

- Regression for classification
  - Optimize the separating hyperplane by treating r^t and $\mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0)$ as real numbers

Least squares method

- Gradient descent based on the least squares error ($r^t \in \{-1, 1\}$)

$$
\begin{aligned}
E_{quad}(\mathbf{w}, w_0|\mathcal{X}) &= \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - (\mathbf{w}^\top \mathbf{x}^t + w_0))^2 \\
\frac{\partial E_{quad}}{\partial w_i} &= \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (-2x_i^t)(r^t - (\mathbf{w}^\top \mathbf{x}^t + w_0)) \\
\frac{\partial E_{quad}}{\partial w_0} &= \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (-2)(r^t - (\mathbf{w}^\top \mathbf{x}^t + w_0))
\end{aligned}
$$

- By setting $e(\mathbf{x}^t) = r^t - \mathrm{h}(\mathbf{x}^t|\mathbf{w}, w_0)$, then

$$
\begin{aligned}
\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{\mathbf{x}^t \in \mathcal{X}} e(\mathbf{x}^t) x_i^t \\
\Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = \eta \sum_{\mathbf{x}^t \in \mathcal{X}} e(\mathbf{x}^t)
\end{aligned}
$$

Algorithm for least squares classification

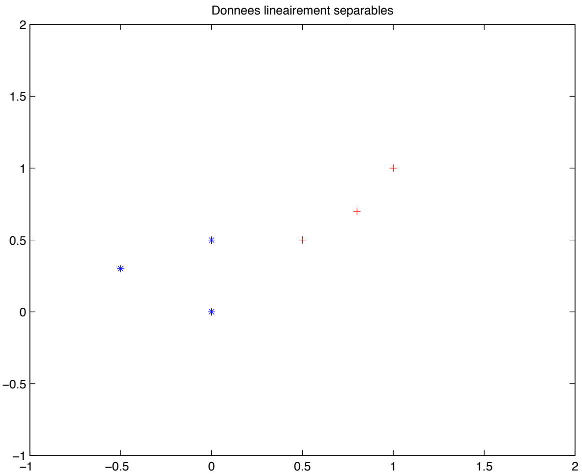
1. Initialize the weights $\mathbf{w}$ and w_0 arbitrarily

$$w_j = 0, \quad j = 0, \ldots, D$$

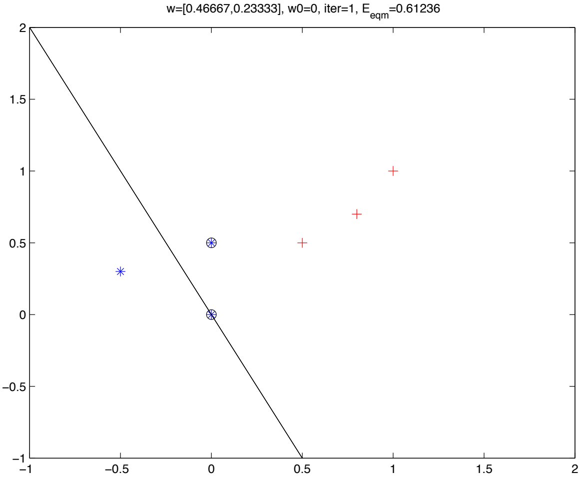
2. Repeat until convergence or depletion of resources:

$$\begin{aligned} e(\mathbf{x}^t) &= r^t - (\mathbf{w}^\top \mathbf{x}^t + w_0) \\ w_j &= w_j + \eta \sum_{\mathbf{x}^t \in \mathcal{X}} e(\mathbf{x}^t) x_j^t, \quad j = 1, \ldots, D \\ w_0 &= w_0 + \eta \sum_{\mathbf{x}^t \in \mathcal{X}} e(\mathbf{x}^t) \end{aligned}$$

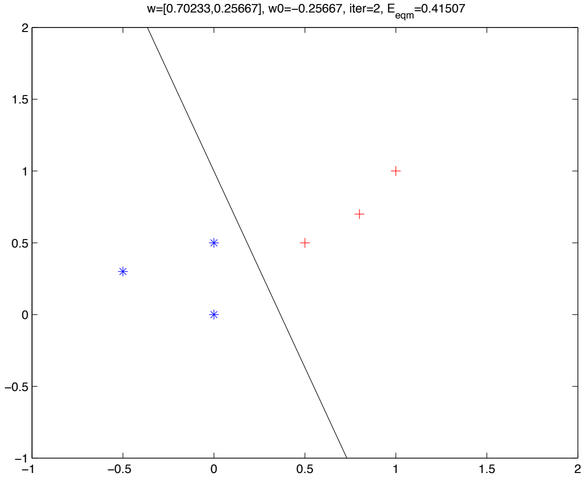
Least squares example (linearly separable)



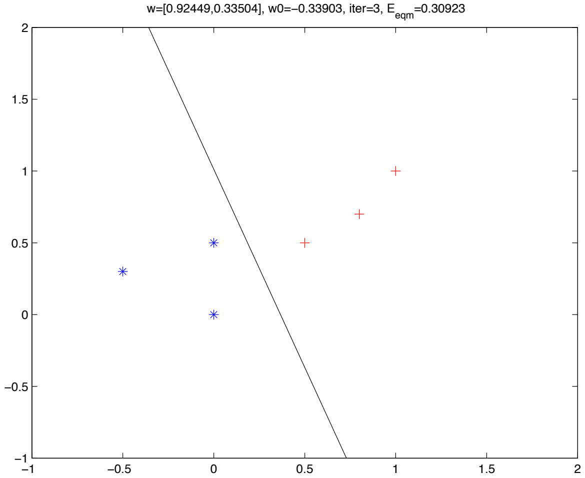
Least squares example (linearly separable)



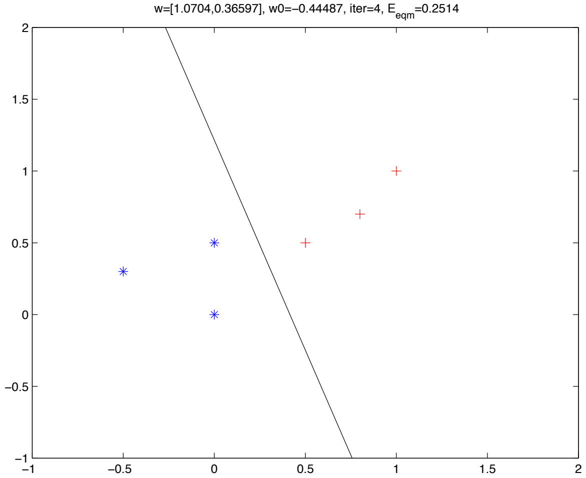
Least squares example (linearly separable)



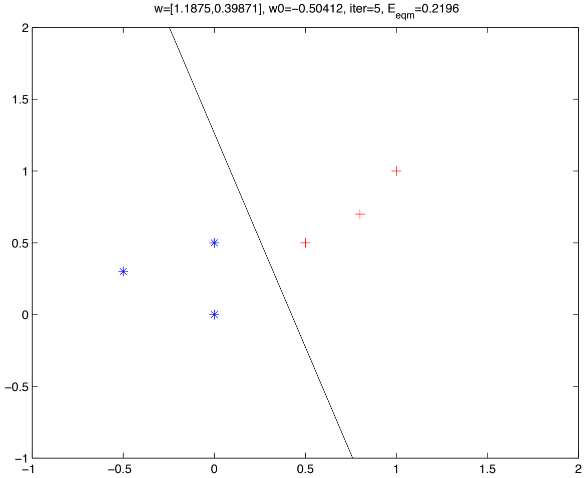
Least squares example (linearly separable)



Least squares example (linearly separable)



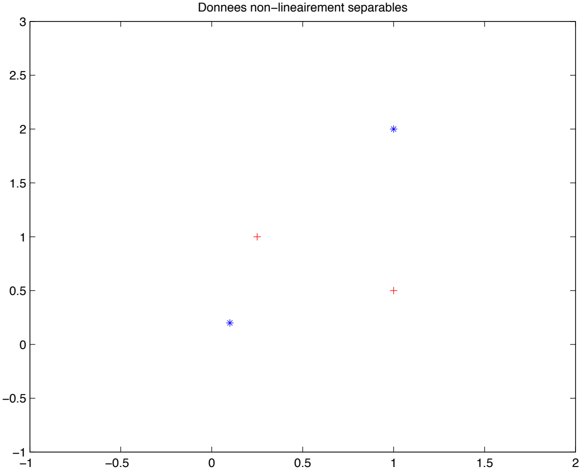
Least squares example (linearly separable)



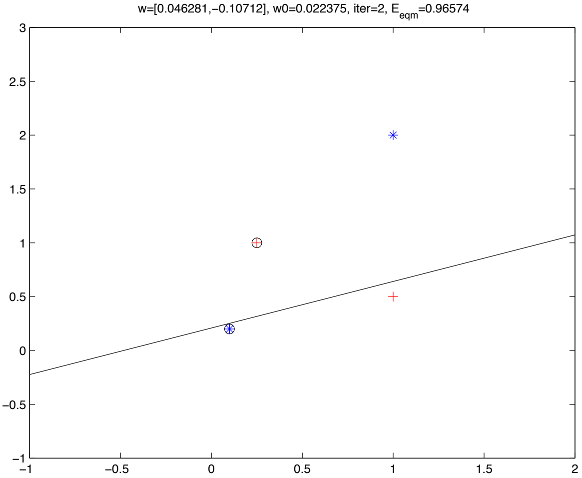
Convergence of the least squares method

- Convergence on linearly separable data
  - Positioning of the separating hyperplane at a position minimizing the least squares error
  - Emphasis on the data with the largest error
  - Strong influence of well-classified data which are far from the separating hyperplane
- Non linearly separable data
  - Best possible positioning of the hyperplane based on the least squares error

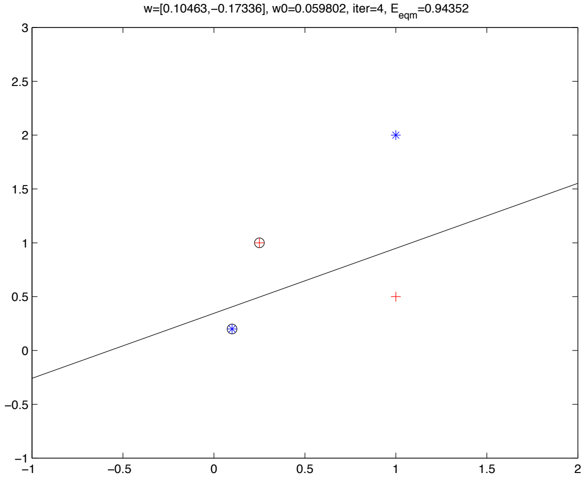
Least squares example (non-linearly separable)



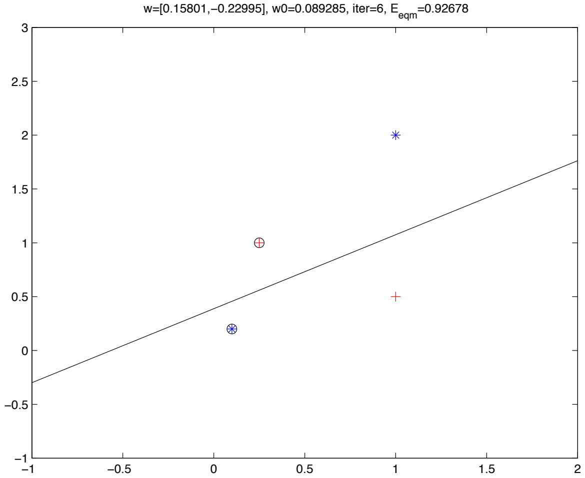
Least squares example (non-linearly separable)



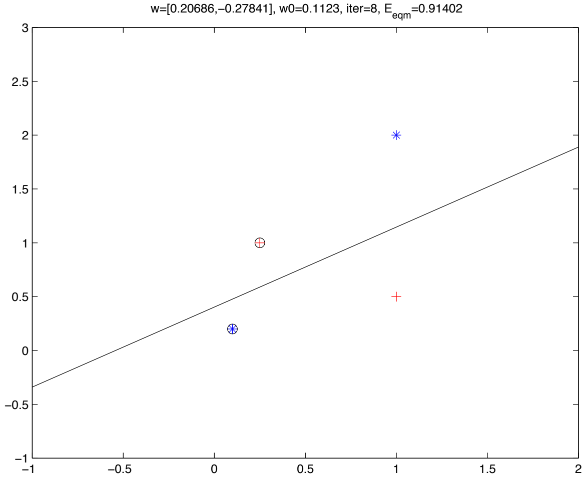
Least squares example (non-linearly separable)



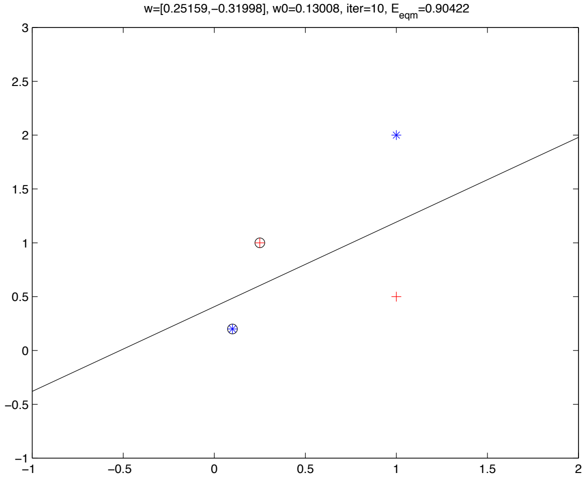
Least squares example (non-linearly separable)



Least squares example (non-linearly separable)



Least squares example (non-linearly separable)



Learning rate η

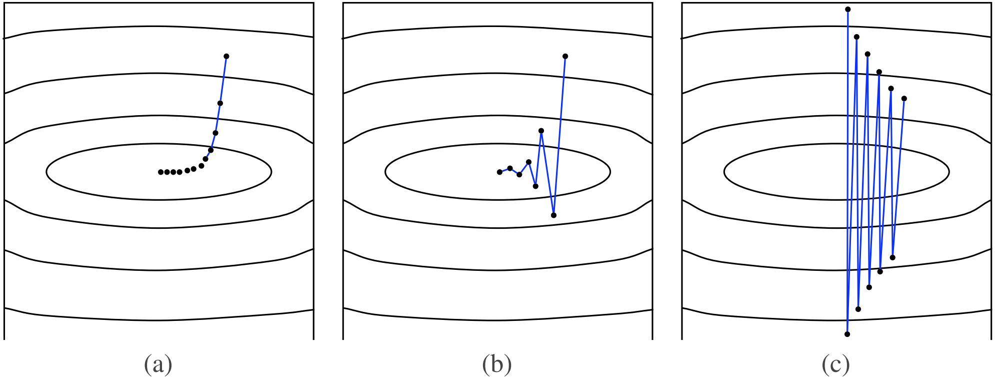


FIG. 5.6 – *Trajectoire de la descente du gradient pour différents taux d'apprentissage : (a) taux faible ; (b) taux moyen ; (c) taux (trop) élevé.*

5.4 Linear parametric methods

Parametric classification

- Discriminant function with parametric classification

$$\mathrm{h}_i(\mathbf{x}) = p(\mathbf{x}|C_i)P(C_i)$$

- Using the log: $\mathrm{h}_i(\mathbf{x}) = \log p(\mathbf{x}|C_i)P(C_i) = \log p(\mathbf{x}|C_i) + \log P(C_i)$
- If $p(\mathbf{x}|C_i)$ corresponds to a multivariate Gaussian distribution

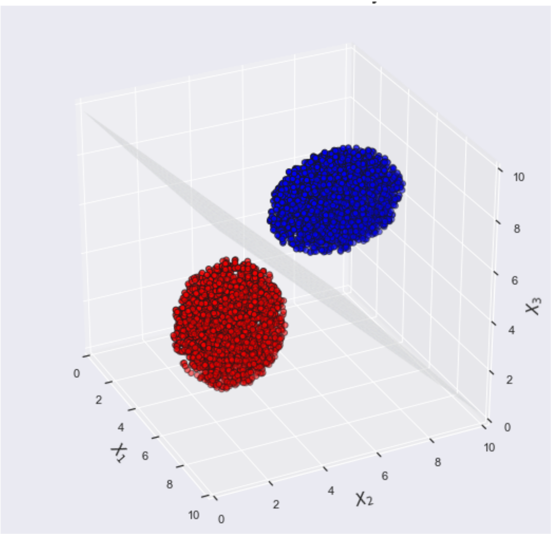
$$\begin{aligned}
p(\mathbf{x}|C_i) &= \frac{1}{(2\pi)^{0.5D}|\boldsymbol{\Sigma}_i|^{0.5D}} \exp\left[-0.5(\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \\
\mathrm{h}_i(\mathbf{x}) &= -0.5\log|\boldsymbol{\Sigma}_i| - 0.5(\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \log \hat{P}(C_i) \\
\hat{P}(C_i) &= \frac{\sum_t r_i^t}{N} \\
\mathbf{m}_i &= \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t} \\
\mathbf{S}_i &= \frac{\sum_t r_i^t (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^\top}{\sum_t r_i^t}
\end{aligned}$$

Parametric classification for linear discrimination

- If the estimation of the covariance matrices is shared, $\mathbf{S} = \sum_i \hat{P}(C_i)\mathbf{S}_i$

$$
\begin{aligned}
\mathrm{h}_i(\mathbf{x}) &= -0.5(\mathbf{x}-\mathbf{m}_i)^\top \mathbf{S}^{-1}(\mathbf{x}-\mathbf{m}_i) + \log \hat{P}(C_i) \\
&= -0.5\left(\mathbf{x}^\top \mathbf{S}^{-1}\mathbf{x} - 2\mathbf{x}^\top \mathbf{S}^{-1}\mathbf{m}_i + \mathbf{m}_i^\top \mathbf{S}^{-1}\mathbf{m}_i\right) + \log \hat{P}(C_i) \\
&= \mathbf{x}^\top \mathbf{S}^{-1}\mathbf{m}_i + (-0.5\mathbf{m}_i^\top \mathbf{S}^{-1}\mathbf{m}_i + \log \hat{P}(C_i)) \\
&= \mathbf{w}_i^\top \mathbf{x} + w_{i,0} \\
\text{where} \quad \mathbf{w}_i &= \mathbf{S}^{-1}\mathbf{m}_i \\
w_{i,0} &= -0.5\mathbf{m}_i^\top \mathbf{S}^{-1}\mathbf{m}_i + \log \hat{P}(C_i)
\end{aligned}
$$

Linear discrimination with parametric methods



logit function

- Two-class parametric classification (C_1 and C_2)
  - Choose C_1 for $\mathbf{x}$ when $P(C_1|\mathbf{x}) > P(C_2|\mathbf{x})$ and C_2 otherwise
  - For two classes, $P(C_1|\mathbf{x}) + P(C_2|\mathbf{x}) = 1$, so $P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x})$.
  - Equivalent formulations, by setting $y \equiv P(C_1|\mathbf{x})$

$$
\begin{aligned}
P(C_1|\mathbf{x}) > P(C_2|\mathbf{x}) \quad &\Rightarrow \quad y > (1-y) \\
\frac{y}{1-y} > 1 \quad &\Rightarrow \quad \log \frac{y}{1-y} > 0
\end{aligned}
$$

- $\mathrm{f}_{logit}(y) = \log \frac{y}{1-y}$ is named *logit* function

Parametric classification and linear discriminant

- Two classes following multivariate normal distributions with shared covariance matrix: linear discriminant

$$
\begin{aligned}
f_{logit}(P(C_1|\mathbf{x})) &= \log \frac{P(C_1|\mathbf{x})}{1 - P(C_1|\mathbf{x})} = \log \frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})} \\
&= \log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)} \\
&= \log \frac{(2\pi)^{-0.5D}|\boldsymbol{\Sigma}|^{-0.5} \exp[-0.5(\mathbf{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)]}{(2\pi)^{-0.5D}|\boldsymbol{\Sigma}|^{-0.5} \exp[-0.5(\mathbf{x} - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_2)]} + \log \frac{P(C_1)}{P(C_2)} \\
&= \mathbf{w}^\top \mathbf{x} + w_0
\end{aligned}
$$

with:

$$
\begin{aligned}
\mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
w_0 &= -0.5(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + \log \frac{P(C_1)}{P(C_2)}
\end{aligned}
$$

Sigmoid function

- logit function

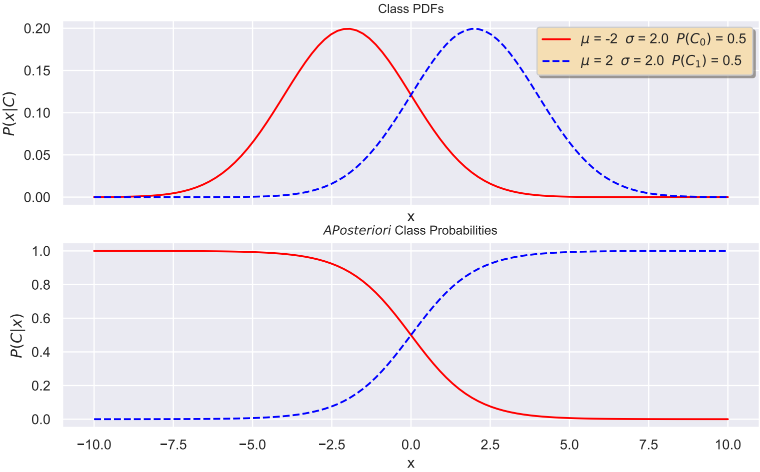
$$\mathrm{f}_{logit}(P(C_1|\mathbf{x})) = \log \frac{P(C_1|\mathbf{x})}{1 - P(C_1|\mathbf{x})} = \mathbf{w}^\top \mathbf{x} + w_0$$

- Inverse of the logit function: sigmoid function (also called logistic function)

$$\mathrm{f}_{logit}(y) = \log \frac{y}{1-y} = a \quad \Rightarrow \quad y = \mathrm{f}_{sig}(a) = \frac{1}{1+\exp(-a)}$$

$$P(C_1|\mathbf{x}) = \mathrm{f}_{sig}(\mathbf{w}^\top \mathbf{x} + w_0) \quad = \quad \frac{1}{1+\exp[-(\mathbf{w}^\top \mathbf{x} + w_0)]}$$

Normal density and a posteriori probabilities



5.5 Logistic regression

Logistic regression

- Logistic regression: estimate $P(C_1|\mathbf{x})$ by gradient descent

$$y = \hat{P}(C_1|\mathbf{x}) = \frac{1}{1 + \exp[-(\mathbf{w}^\top \mathbf{x} + w_0)]}$$

- Learning $\mathbf{w}$ and w_0 from $\mathcal{X} = \{\mathbf{x}^t, r^t\}$, with $r^t \in \{0, 1\}$
  - r^t for a certain $\mathbf{x}^t$ follows a Bernoulli distribution with a probability $y^t = P(C_1|\mathbf{x}^t)$

$$r^t|\mathbf{x}^t \sim \mathcal{B}(1, y^t)$$

  - Likelihood of sampling $\mathcal{X}$ based on $\mathbf{w}, w_0$

$$l(\mathbf{w}, w_0|\mathcal{X}) = \prod_t (y^t)^{(r^t)} (1 - y^t)^{(1-r^t)}$$

  - Error that maximizes log-likelihood

$$E_{entr}(\mathbf{w}, w_0|\mathcal{X}) = -\log l(\mathbf{w}, w_0|\mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t)$$

  - Error $E_{entr}(\mathbf{w}, w_0|\mathcal{X})$ also named *cross entropy*

Minimization of cross entropy

- Derivative of the sigmoid function $y = \mathrm{f}_{sig}(a) = \frac{1}{1+\exp(-a)}$

$$\begin{aligned} \frac{dy}{da} &= \frac{\exp(-a)}{[1+\exp(-a)]^2} = \frac{1}{1+\exp(-a)} \frac{\exp(-a)+1-1}{1+\exp(-a)} \\ &= \frac{1}{1+\exp(-a)} \left(1 - \frac{1}{1+\exp(-a)}\right) = y(1-y) \end{aligned}$$

- Minimizing cross entropy by gradient descent

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = -\eta \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1-r^t}{1-y^t} \right) y^t(1-y^t)x_j^t \\ &= \eta \sum_t (r^t - y^t)x_j^t \\ \Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = -\eta \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_0} = \eta \sum_t (r^t - y^t) \end{aligned}$$

Algorithm for discrimination by logistic regression

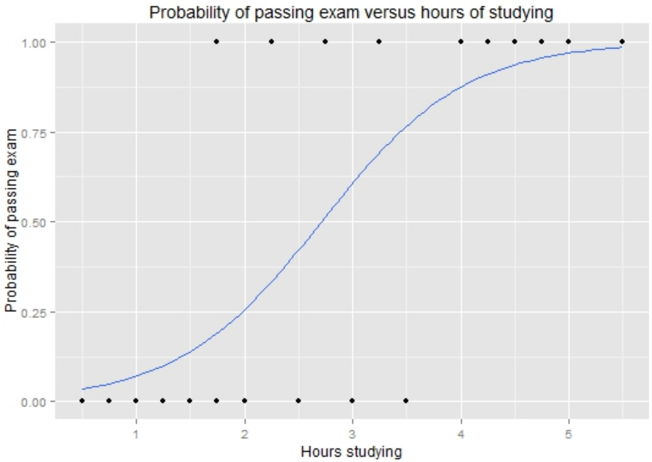
1. Randomly initialize weights (evenly distributed), $w_j \sim \mathcal{U}(-0{,}01, 0{,}01)$

$$w_j = \mathrm{rand}(-0{,}01, 0{,}01), \quad j = 0, \ldots, D$$

2. Repeat until convergence:

$$\begin{aligned} y^t &= \frac{1}{1 + \exp[-(\mathbf{w}^\top \mathbf{x}^t + w_0)]}, \quad t = 1, \ldots, N \\ w_j &= w_j + \eta \sum_t (r^t - y^t) x_j^t, \quad j = 1, \ldots, D \\ w_0 &= w_0 + \eta \sum_t (r^t - y^t) \end{aligned}$$

Example of logistic regression



Performance criteria

- Different methods give different parameterizations $(\mathbf{w}, w_0)$
  - Perceptron
$$E_{percp}(\mathbf{w}, w_0 | \mathcal{X}) = -\sum_{\mathbf{x}^t \in \mathcal{Y}} r^t \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)$$
    - $\mathcal{Y}$ are the data from $\mathcal{X}$ which were misclassified by $\mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)$
  - Least squares error
$$E_{quad}(\mathbf{w}, w_0 | \mathcal{X}) = \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0))^2$$
  - Cross entropy (logistic regression)
$$\begin{aligned} y &= \frac{1}{1 + \exp[-\mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)]} \\ E_{entr}(\mathbf{w}, w_0 | \mathcal{X}) &= -\sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t) \end{aligned}$$

Other performance criteria

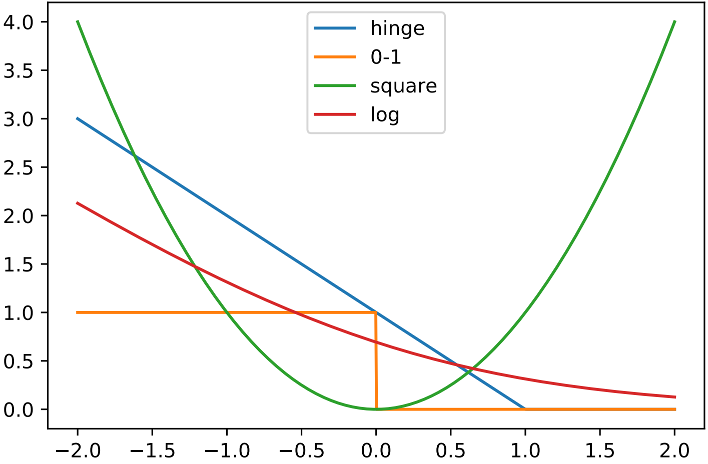
- Linear discriminant analysis: maximize $J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$
  - Separate class averages m_i while reducing the variance of each class s_i^2
- Error function *hinge*

$$E_{hinge}(\mathbf{w}, w_0 | \mathcal{X}) = \sum_{\mathbf{x}^t \in \mathcal{Y}} [1 - r^t \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0)]$$

  - $\mathcal{Y} = \{\mathbf{x}^t \in \mathcal{X} \mid r^t \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0) \leq 1\}$
  - $\mathcal{Y}$ are the data from $\mathcal{X}$ which are in the *margin*
  - Used in the SVM (presented next week)
- Log loss function

$$E_{log}(\mathbf{w}, w_0 | \mathcal{X}) = \sum_{\mathbf{x}^t \in \mathcal{X}} \log \left[1 + \exp(-r^t \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0))\right]$$

Comparison of different error criteria



5.6 Normalization and regularization

Weight normalization

- Under certain circumstances, the values of the weights $\mathbf{w}$ and w_0 can explode (or implode)
  - Several weight values give the same discriminant, only relative values of $\mathbf{w}$ and w_0 count for the classification (sign of $\mathrm{h}(\mathbf{x})$)
  - Repeated corrections continuously add value to the weights
  - According to the criterion, a lower error is obtained with low weights
  - May exceed values that can be represented on a computer (*overflow* or *underflow*)
- Possible solutions
  - Normalize the weights at each iteration

$$\mathbf{w}' = \frac{\mathbf{w}}{\|[\mathbf{w}\ \ w_0]^\top\|}, \quad w_0' = \frac{w_0}{\|[\mathbf{w}\ \ w_0]^\top\|}$$

  - Standardization in error criteria

$$E'(\mathbf{w}, w_0|\mathbf{x}^t) = \frac{E(\mathbf{w}, w_0|\mathbf{x}^t)}{\|\mathbf{x}^t\|^2}$$

Ridge regression (regularization l_2)

- Ridge regression
  - Limiting weight values during optimization

$$\text{minimize} \quad E_{quad}(\mathbf{w}, w_0 | \mathcal{X}) = \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0))^2 \qquad \text{subject to} \quad \left(\sum_{i=1}^{D} w_i^2 \right) \leq \gamma$$

  - Equivalent formulation (Tychonoff regularization)

$$E_{ridge} = \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0))^2 + \lambda \sum_{i=1}^{D} w_i^2$$

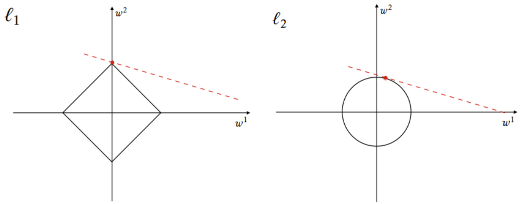
- Guides the search towards simpler models
  - Correlation between variables: positive value of w_i can be cancelled by negative value of w_j.
- Requires data to be standardized and centred at the origin
- Can be combined with criteria other than the least squares error

LASSO (regularization l_1)

- LASSO: use a l_1 regulation instead of l_2.

$$E_{LASSO} = \frac{1}{2} \sum_{\mathbf{x}^t \in \mathcal{X}} (r^t - \mathrm{h}(\mathbf{x}^t | \mathbf{w}, w_0))^2 + \lambda \sum_{i=1}^{D} |w_i|$$

  - Cannot be resolved by partial derivatives, requires methods such as quadratic programming
  - Favours elimination of variables (feature selection)



5.7 Multi-class models

Multi-class models

- With K classes, various strategies are possible
  - Approach *one versus all* (OvA)
  - Approach *one versus one* (OvO)
  - Other approaches (e.g. error correction code)
- One against all
  - One discriminating function per class, $\mathrm{h}_i(\mathbf{x}|\mathbf{w}_i, w_{i,0})$, $i = 1, \ldots, K$
  - Option 1: maximum value, $\mathrm{h}(\mathbf{x}) = \underset{C_i=C_1}{\overset{C_K}{\operatorname{argmax}}}\, \mathrm{h}_i(\mathbf{x})$
  - Option 2: positive value only

$$\mathrm{h}(\mathbf{x}) = \begin{cases} C_i & \text{if } \mathrm{h}_i(\mathbf{x}) \geq 0 \text{ and } \mathrm{h}_j(\mathbf{x}) < 0,\ \forall i \neq j \\ \text{ambiguity} & \text{otherwise} \end{cases}$$

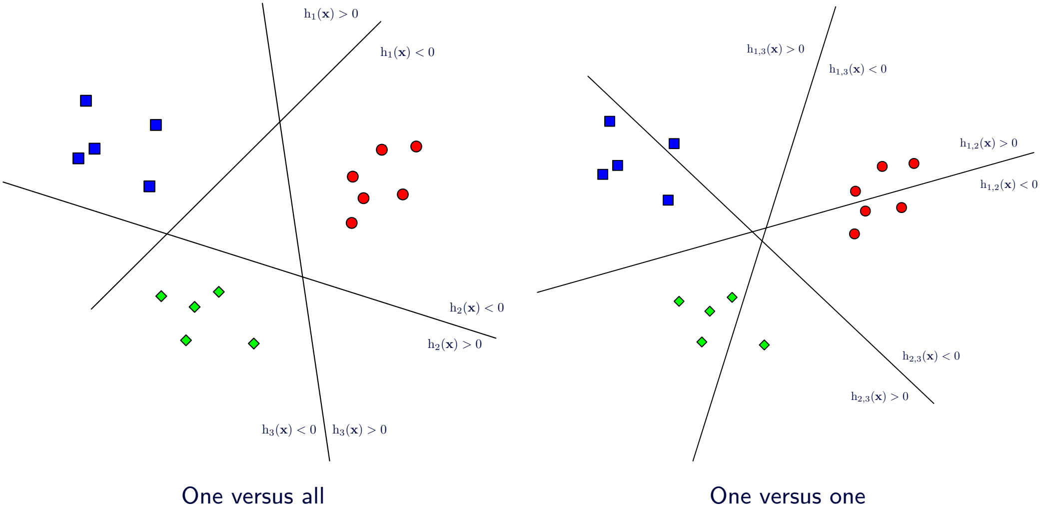
Multi-class models

- One versus one
    - One linear discriminator per each pair of classes, $\mathrm{h}_{i,j}(\mathbf{x}|\mathbf{w}_{i,j},w_{i,j,0})$, $i = 1,\ldots,K-1$, $j = i+1,\ldots,K$
    - Symmetrical discriminant $\mathrm{h}_{j,i}(\mathbf{x}) = -\mathrm{h}_{i,j}(\mathbf{x})$, $j = 2,\ldots,K$, $i = 1,\ldots,j-1$
    - Discriminators trained only on data of class C_i and C_j

$$\mathrm{h}_{i,j}(\mathbf{x}) = \begin{cases} \geq 0 & \text{si } \mathbf{x} \in C_i \\ < 0 & \text{si } \mathbf{x} \in C_j \\ \text{ignored} & \text{otherwise} \end{cases} \quad \text{with } i = 1,\ldots,K,\ j = i,\ldots,K$$

-
    - Data evaluation: choose C_i if $\forall j \neq i$, $\mathrm{h}_{i,j} > 0$
    - Possible relaxation: $\mathrm{h}_i(\mathbf{x}) = \sum_{j \neq i} \mathrm{h}_{i,j}(\mathbf{x})$

Multi-class decision boundaries



5.8 Online learning

Online and batch learning

- Batch learning
  - Weight correction once at each iteration, calculating the error for the whole dataset
  - Relatively stable learning
- Online learning
  - Weight correction for each data presentation, so N weight corrections per iteration
  - Guided by the error on each observation ($E(\mathbf{w}, w_0 | \mathbf{x}^t)$)
  - Requires permutation of the processing order at each iteration to avoid bad sequences
  - Online learning is faster than batch learning, but with the risk of greater instabilities

Stochastic gradient descent

- Stochastic gradient descent
  - Going further than online learning: random sampling of the training dataset
  - Typical algorithm:
    1. Randomly (uniformly) sample one observation $\mathbf{x}^t$ in $\mathcal{X}$, $t \sim \mathcal{U}(1,N)$
    2. Determine the value of the learning rate, typically $\eta^l = 1/l$ where l is the index of the current data in its order of processing
    3. Correct weights by gradient descent

$$\Delta w_j = -\eta^l \frac{\partial E(\mathbf{w}, w_0 | \mathbf{x}^t)}{\partial w_j}, \quad j = 0, \ldots, D$$

    4. Repeat until convergence or depletion of resources
- Requires a decreasing adjustment of the learning rate for each data
- Interesting for processing very large datasets in one pass
- Also allows to stop the learning at any time
- Can also be adapted to the processing of data streams

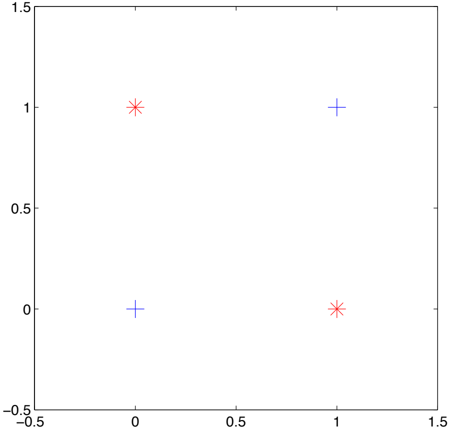
5.9 Basis functions

The XOR problem

- The XOR problem

$$
\begin{aligned}
\mathbf{x}_1 &= [0\ 0]^\top & r_1 &= 0\\
\mathbf{x}_2 &= [0\ 1]^\top & r_2 &= 1\\
\mathbf{x}_3 &= [1\ 0]^\top & r_3 &= 1\\
\mathbf{x}_4 &= [1\ 1]^\top & r_4 &= 0
\end{aligned}
$$

- Example of non-linearly separable data



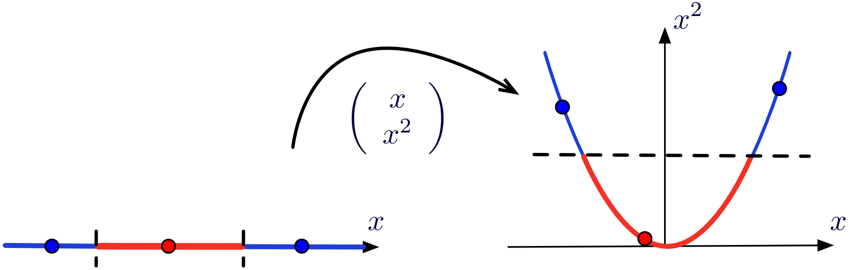
Basis functions

- Discriminant with basis function
  - Non-linear transformation $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^K$ processed in a linear form

$$\mathrm{h}_i(\mathbf{x}) = \sum_{j=1}^{K} w_j \phi_{i,j}(\mathbf{x}) + w_0$$

  - Example of basis functions
    - $\phi_{i,j}(\mathbf{x}) = x_j$
    - $\phi_{i,j}(\mathbf{x}) = x_1^{j-1}$
    - $\phi_{i,j}(\mathbf{x}) = \exp(-(x_2 - m_j)^2/c)$
    - $\phi_{i,j}(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{m}_j\|^2/c)$
    - $\phi_{i,j}(\mathbf{x}) = \mathrm{sgn}(x_j - c_j)$

Projection with a basis function



- In 1D: non-linearly separable
- With 2D projection: linearly separable

Basis functions

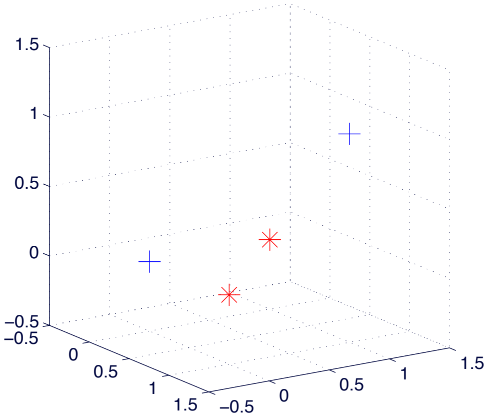
- XOR resolution with basis function $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$\phi(\mathbf{x}) = [x_1 \; x_2 \; (x_1 x_2)]^\top$$

- Transformation results

$$\begin{aligned} \mathbf{z}_1 &= [0 \; 0 \; 0]^\top & r_1 &= 0 \\ \mathbf{z}_2 &= [0 \; 1 \; 0]^\top & r_2 &= 1 \\ \mathbf{z}_3 &= [1 \; 0 \; 0]^\top & r_3 &= 1 \\ \mathbf{z}_4 &= [1 \; 1 \; 1]^\top & r_4 &= 0 \end{aligned}$$

- Linearly separable data in the new space!



5.10 Linear discriminants in scikit-learn

Scikit-learn: linear models

- `discriminant_analysis.LinearDiscriminantAnalysis`: parametric methods to generate linear discriminants
- `linear_model.LinearRegression`: least squares linear regression
- `linear_model.Ridge`: ridge regression (least squares + l_2 regularization)
- `linear_model.Lasso`: LASSO (least squares + l_1 regularization)
- `linear_model.LogisticRegression`: logistic regression
- `linear_model.Perceptron`: linear discriminant trained with the perceptron rule
- `linear_model.SGDClassifier` and `linear_model.SGDRegressor`: linear models trained by stochastic gradient descent

Scikit-learn: multi-class management

- Scikit-learn classifiers can manage multiple classes *out-of-the-box*
- Models for specific multi-class management
  - `multiclass.OneVsRestClassifier`: one versus all approach
  - `multiclass.OneVsOneClassifier`: one versus one approach
  - `multiclass.OutputCodeClassifier`: Error correction codes (to be seen in presentation on *Ensemble methods* in the second half of the semester)