Optimization Project in Energy
ENT306

Elise Grosjean
Ensta-Paris

Ensta-Paris
Institut Polytechnique de Paris

Supervised classification

Training a machine learning model to assign labels (or classes) to data points based on their features.

- $(x_i)_{i=1,\ldots,d}$ features
- $y_i \in \{0, 1\}$ associated labels

Objective

Learn a relationship between x and y that allows the model to predict the label y for new unseen data points x.

Supervised classification applications

1. **Predictive maintenance of energy equipment**
   Application: Predicting whether a piece of equipment (e.g. a wind turbine) is likely to break ($y = 1$) or not ($y = 0$) based on measured parameters such as temperature, pressure...

   Interpretation:
   x: Feature vector representing measurements from the equipment (e.g., temperature, pressure, etc.).
   w: Weights indicating the relative importance of each feature in predicting equipment failure.
   y: Binary indicator of failure (1 for failure, 0 for normal operation).

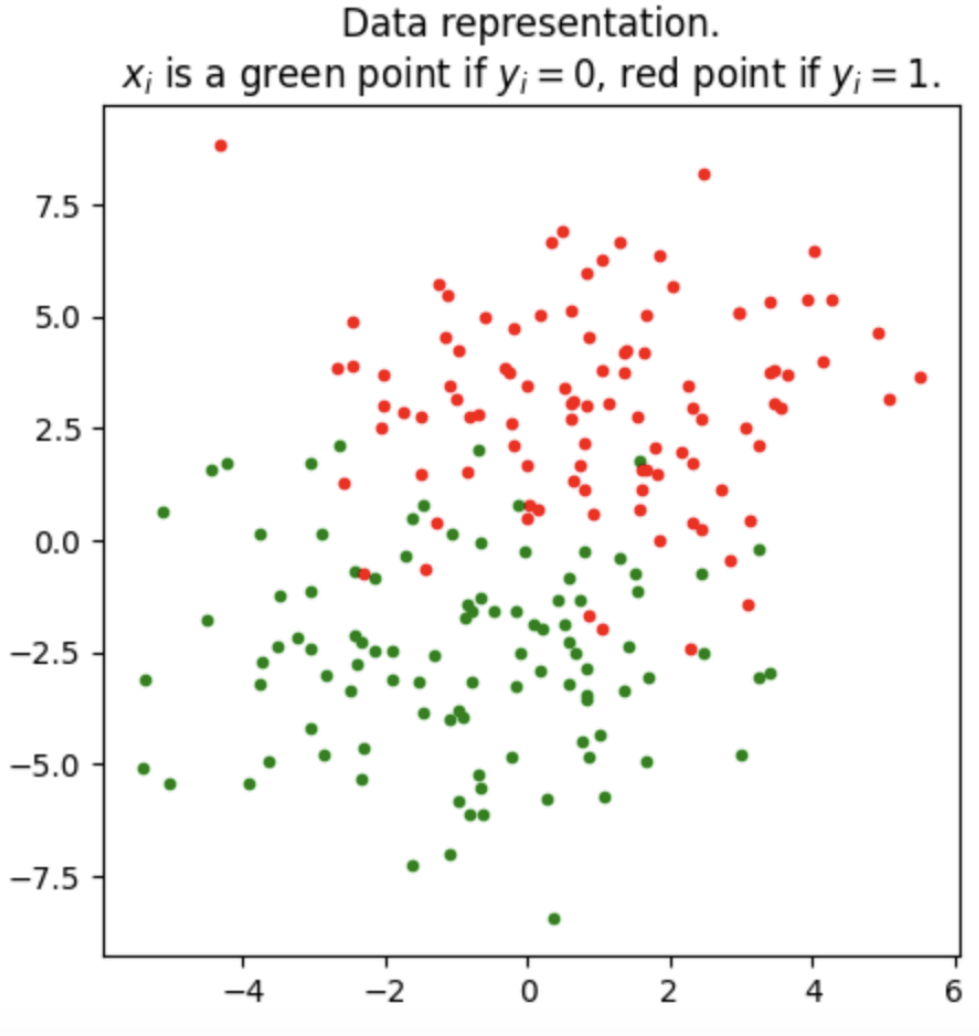
2. **Classification of buildings by energy performance**
   Application: Classifying buildings based on their energy efficiency (e.g., low-energy consumption buildings $y = 0$ versus energy-intensive buildings $y = 1$). Interpretation:

   x: Features describing the building (e.g., thermal insulation, heating type, surface area, etc.).
   w: Contributions of each feature to the likelihood of a building being energy-intensive.
   y: Energy performance classification (0 for efficient, 1 for energy-intensive).

Application example

1. **Predictive maintenance of energy equipment**
   Application: Predicting whether a piece of equipment (e.g. a wind turbine) is likely to break ($y = 1$) or not ($y = 0$) based on measured parameters such as temperature, pressure...

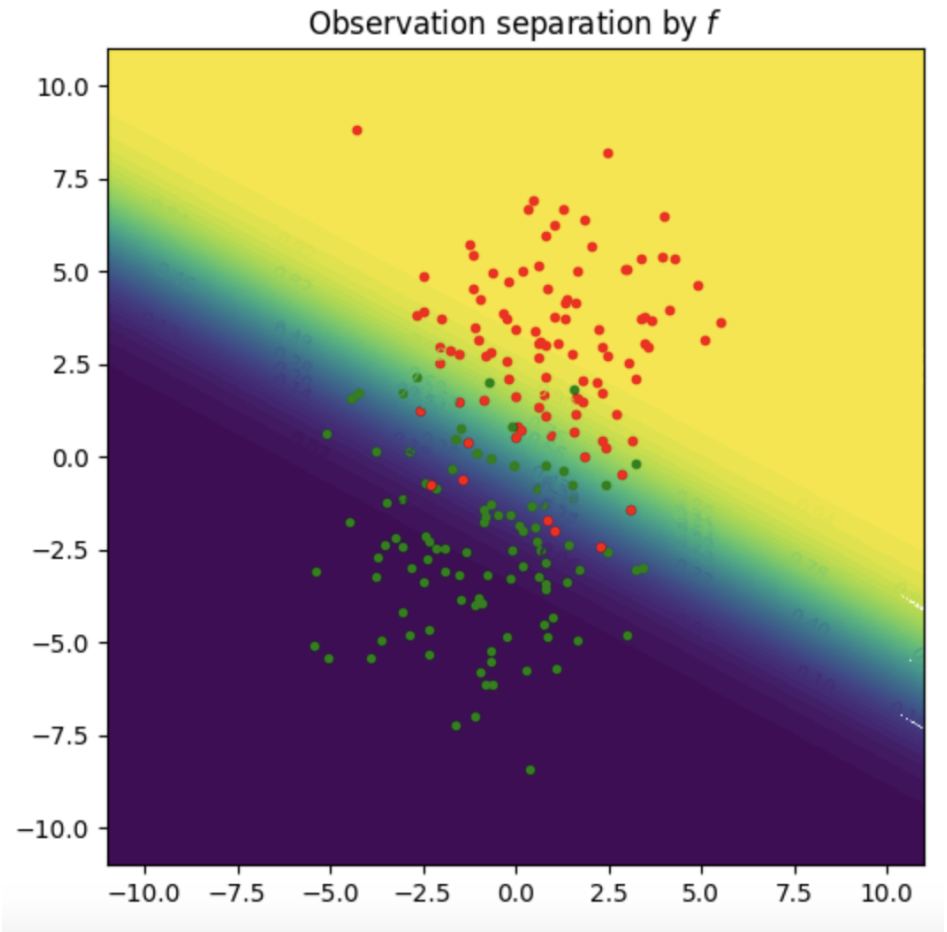
   Interpretation:
   x: Feature vector representing measurements from the equipment (e.g., temperature, pressure, etc.).
   w: Weights indicating the relative importance of each feature in predicting equipment failure.
   y: Binary indicator of failure (1 for failure, 0 for normal operation).



Application example

1. **Predictive maintenance of energy equipment**
   Application: Predicting whether a piece of equipment (e.g. a wind turbine) is likely to break ($y = 1$) or not ($y = 0$) based on measured parameters such as temperature, pressure...

   Interpretation:
   x: Feature vector representing measurements from the equipment (e.g., temperature, pressure, etc.).
   w: Weights indicating the relative importance of each feature in predicting equipment failure.
   y: Binary indicator of failure (1 for failure, 0 for normal operation).



Binary supervised classification

Goal

Find the separation line thanks to the training data.

In other words, find the optimal weights $w = (w_1, w_2) \in \mathbb{R}^2$ and $b \in \mathbb{R}$ such that

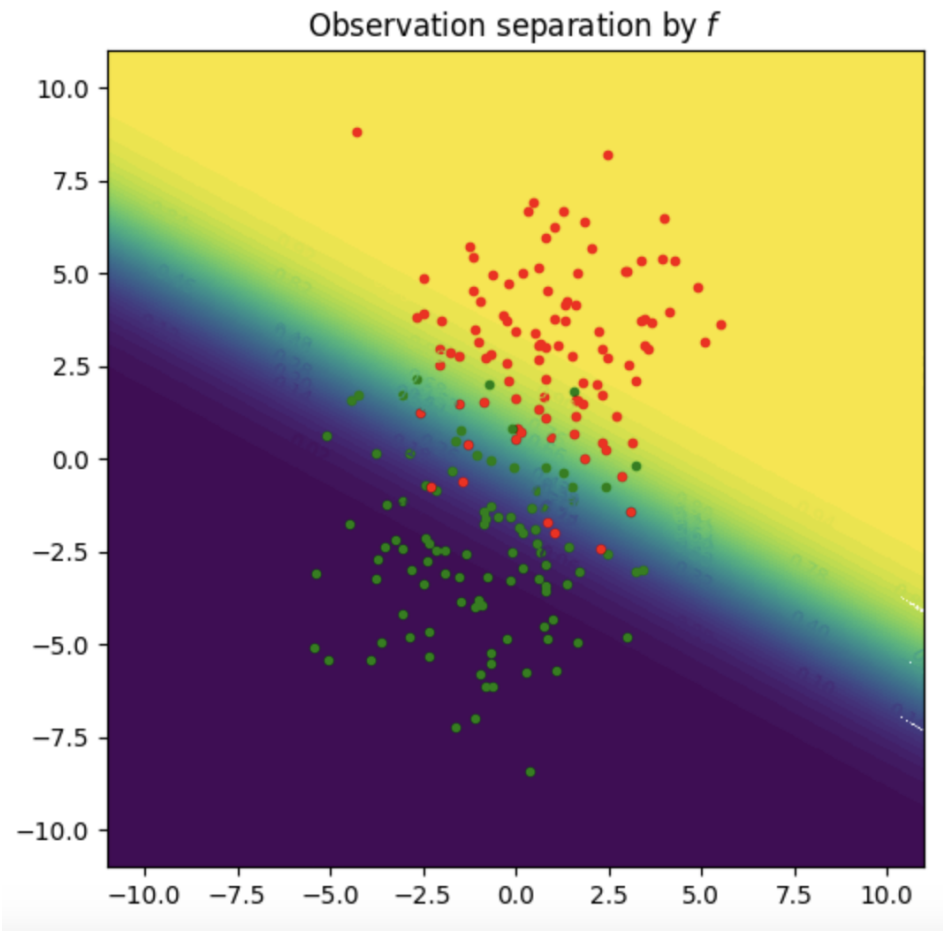
$$\langle w, x \rangle + b = w^T x + b = 0.$$

Interpretation:

x: Feature vector representing measurements from the equipment (e.g., temperature, pressure, etc.).

w: Weights indicating the relative importance of each feature in predicting equipment failure.

y: Binary indicator of failure (1 for failure, 0 for normal operation).



Binary supervised classification

How does it work?

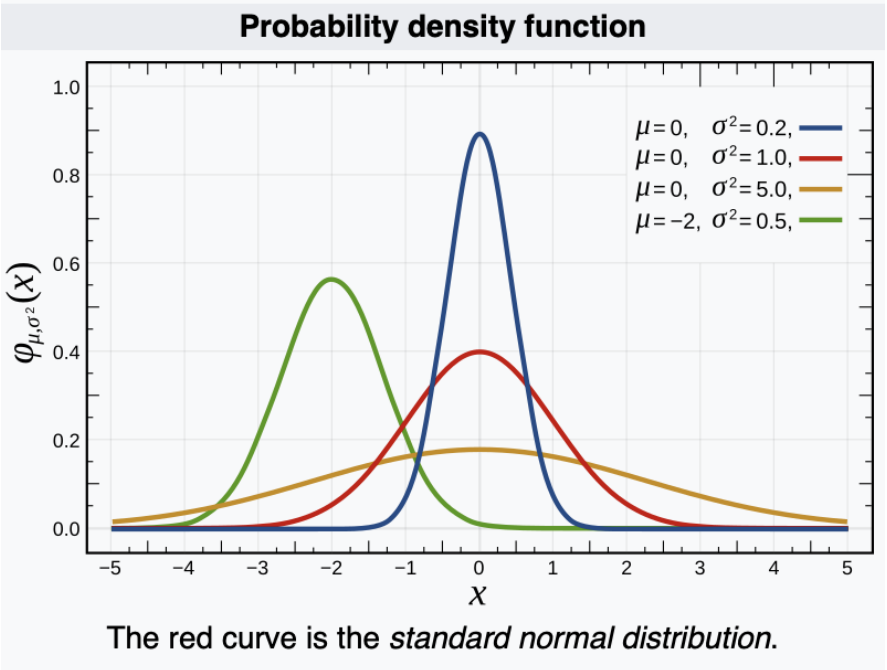
Binary supervised classification

To simplify, we will take centered data in 0, so we only need to find w such that

$$\langle w, x \rangle = w^T x = 0.$$

e.g. for $i = 1, \ldots, n$, x_i = random float vectors ($x \in \mathbb{R}^2$) sampled from a "normal" (Gaussian) distribution of mean 0 and variance 1: $\mathcal{N}(\mu, \sigma) = \mathcal{N}(0, 1)$
The form of the associated probability density function is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Python → np.random.randn(n)

- μ : mean of the distribution
- σ : variance of the distribution

Binary supervised classification

Binary supervised classification $\rightarrow$ logistic regression

regression = find a correlation between a binary variable and some observations thanks to an optimization problem

- Decision Trees
- K-Nearest Neighbors (k-NN)
- Probabilistic Models
- Neural Networks

Binary supervised classification

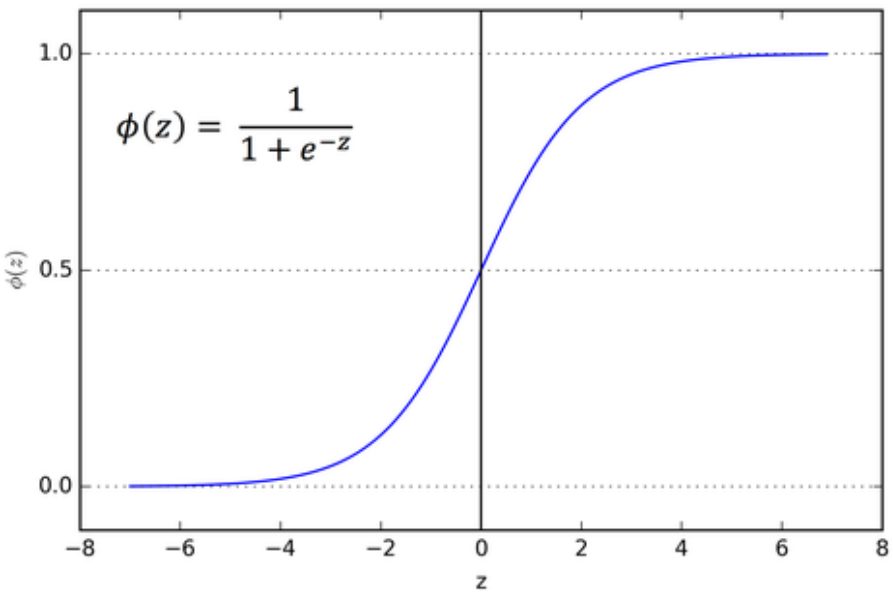
Logistic regression

$$\begin{array}{lcl} x_1 & \to & f(\langle w_1, x_1 \rangle) \\ \dots & \to & \dots \\ x_n & \to & f(\langle w_n, x_n \rangle) \end{array}$$

The sigmoid function σ is often used (for f) in logistic regression:

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$



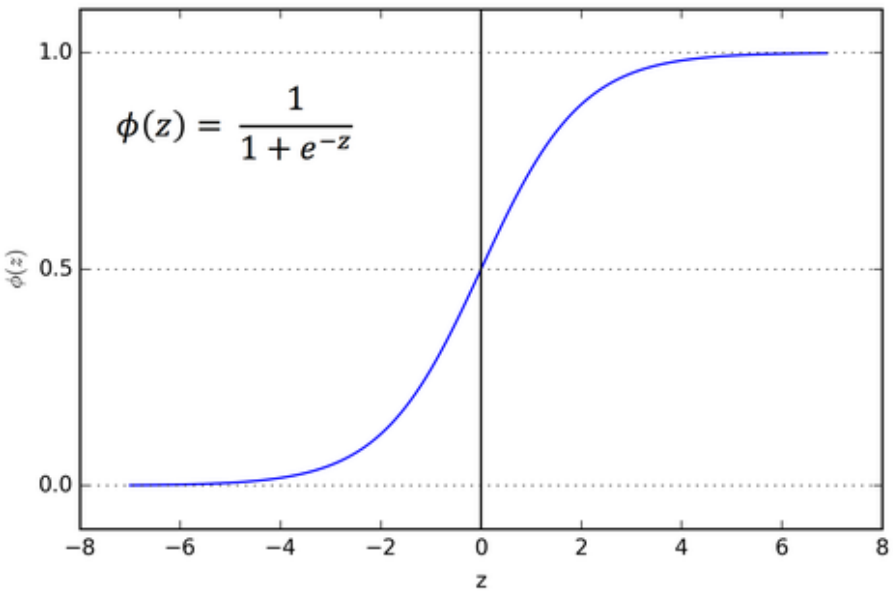
Binary supervised classification

Logistic regression

$$\begin{array}{ccc} x_1 & \rightarrow & f(\langle w_1, x_1 \rangle) \\ \dots & \rightarrow & \dots \\ x_n & \rightarrow & f(\langle w_n, x_n \rangle) \end{array}$$

The sigmoid function σ is often used (for f) in logistic regression:

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$



Decision rule:

- if $\sigma(\langle w, x \rangle) > 0.5, y = 1$
- if $\sigma(\langle w, x \rangle) < 0.5, y = 0$
- if $\langle w, x \rangle >> 0, P(y = 1|x) \simeq 1$
- if $\langle w, x \rangle << 0, P(y = 1|x) \simeq 0$

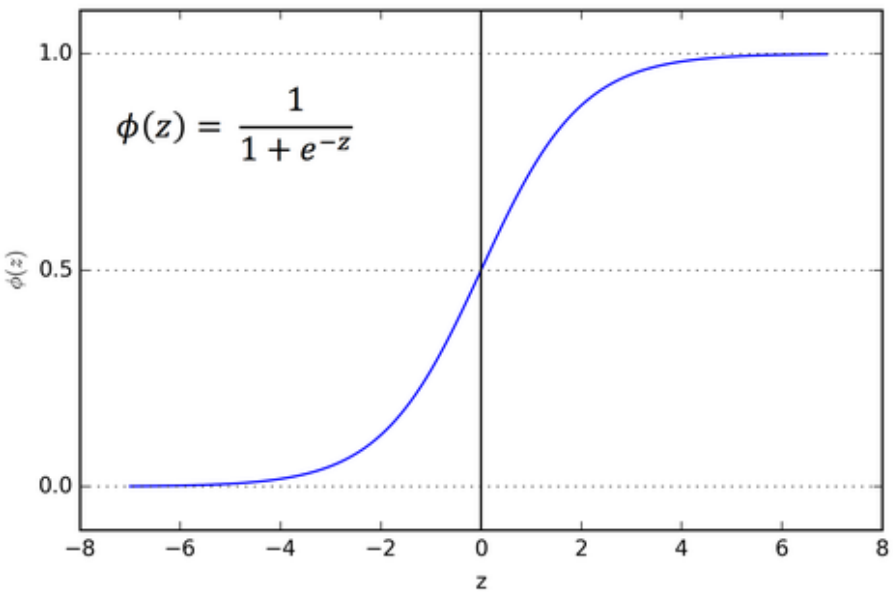
Binary supervised classification

Logistic regression

$$
\begin{array}{ccc}
x_1 & \rightarrow & f(\langle w_1, x_1 \rangle) \\
\dots & \rightarrow & \dots \\
x_n & \rightarrow & f(\langle w_n, x_n \rangle)
\end{array}
$$

The sigmoid function σ is often used (for f) in logistic regression:

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$



Decision rule:

- if $\sigma(\langle w, x \rangle) > 0.5, y = 1$
- if $\sigma(\langle w, x \rangle) < 0.5, y = 0$
- modeling uncertainty

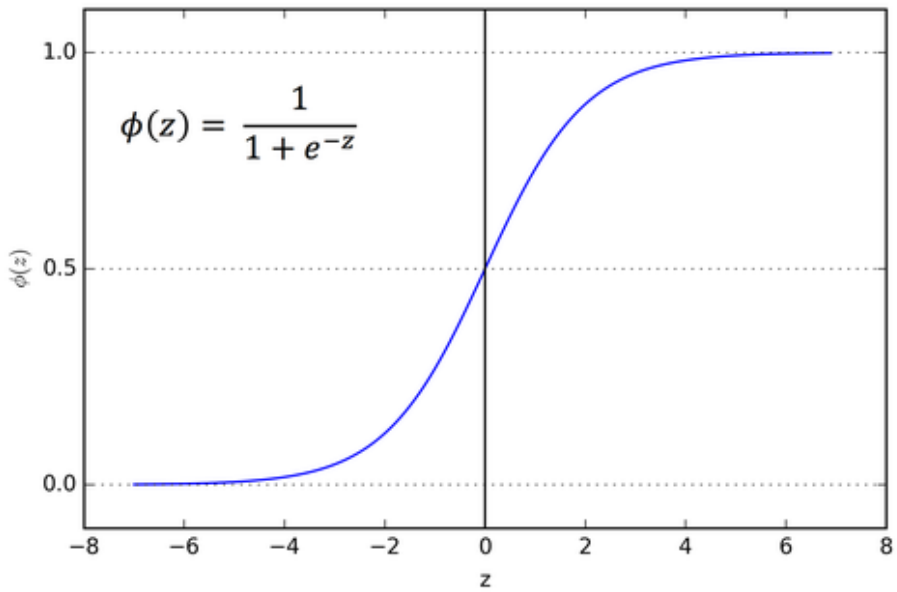
Binary supervised classification

Logistic regression

$$\begin{array}{ccc} x_1 & \rightarrow & f(\langle w_1, x_1 \rangle) \\ \dots & \rightarrow & \dots \\ x_n & \rightarrow & f(\langle w_n, x_n \rangle) \end{array}$$

The sigmoid function σ is often used (for f) in logistic regression:

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$



- How much more likely is $y = 1$ compared to $y = 0$?
  Linear since if the probability to have $y = 1$ increases then the probability of having 0 decreases proportionally.
  Likelihood function:
  $log\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = \langle w, x \rangle$

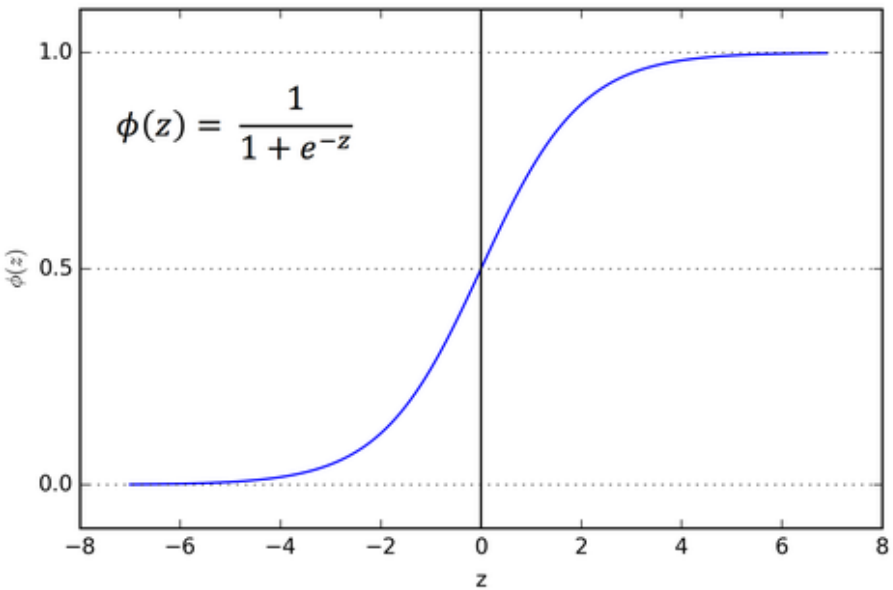
Binary supervised classification

Logistic regression

$$\begin{array}{ccc} x_1 & \to & f(\langle w_1, x_1 \rangle) \\ \dots & \to & \dots \\ x_n & \to & f(\langle w_n, x_n \rangle) \end{array}$$

The sigmoid function σ is often used (for f) in logistic regression:

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$



- Likelihood function: $log\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = \langle w, x \rangle$
- $P(y = 1|x) = \sigma(\langle w, x \rangle)$

Likelihood function

$$P(y=0|x) = 1 - P(y=1|x) \qquad (*)$$

$$\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = e^{\langle w,x\rangle}$$

$$\left(\frac{P(y=1|x)}{1-P(y=1|x)}\right) = e^{\langle w,x\rangle}$$

$$P(y=1|x) = e^{\langle w,x\rangle}(1 - P(y=1|x))$$

$$P(y=1|x)(1 + e^{\langle w,x\rangle}) = e^{\langle w,x\rangle}$$

$$P(y=1|x) = \frac{1}{1+e^{-\langle w,x\rangle}} := \sigma(\langle w,x\rangle)$$

$$P(y=0|x) = 1 - \frac{1}{1+e^{-\langle w,x\rangle}} := 1 - \sigma(\langle w,x\rangle)$$

Likelihood function

$$P(y=1|x) = \frac{1}{1+e^{-\langle w,x\rangle}} := \sigma(\langle w,x\rangle)$$
$$P(y=0|x) = 1 - \frac{1}{1+e^{-\langle w,x\rangle}} := 1-\sigma(\langle w,x\rangle)$$

Log-loss function:

$$f(w) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(\langle w,x_i\rangle)) + (1-y_i) log(1-\sigma(\langle w,x_i\rangle)) + \lambda\frac{1}{2}\|w\|^2$$

- y_i : true label (0 or 1),
- $\sigma(\langle w,x_i\rangle)$: probablity predicted by the model to get $y_i = 1$.

Likelihood optimization

MINIMIZE the log-loss function:

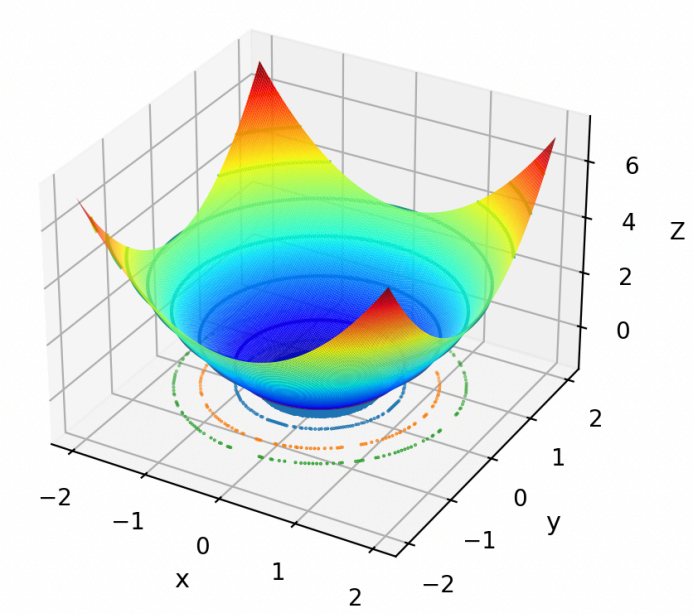
$$f(w) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\sigma(\langle w, x_i\rangle))+(1-y_i)log(1-\sigma(\langle w, x_i\rangle))+\lambda\frac{1}{2}\|w\|^2$$

$\rightarrow$ Gradient descent algorithm !!!

Exercise 1

Write a function `gradf(w)` which computes the gradient of the log-loss function.

Reminders: Gradient methods



Our goal: solving **numerically** the problem

$$\inf_{x \in \mathbb{R}^n} f(x). \qquad (P)$$

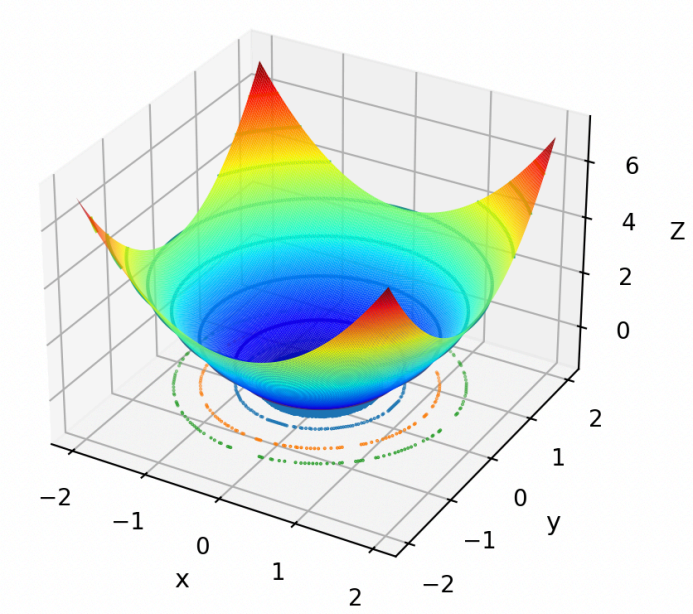
General idea: to compute a sequence $(x_k)_{k \in \mathbb{N}}$ such that

$$f(x_{k+1}) \leq f(x_k), \quad \forall k \in \mathbb{N},$$

the inequality being strict if $\nabla f(x_k) \neq 0$. $\rightarrow$ **Iterative** method.

How to compute x_{k+1}?

Reminders: Gradient methods



Our goal: solving **numerically** the problem

$$\inf_{x \in \mathbb{R}^n} f(x). \qquad (P)$$

General idea: to compute a sequence $(x_k)_{k \in \mathbb{N}}$ such that

$$f(x_{k+1}) \leq f(x_k), \quad \forall k \in \mathbb{N},$$

the inequality being strict if $\nabla f(x_k) \neq 0$. $\rightarrow$ **Iterative** method.
How to compute x_{k+1}?

Gradient methods

Main idea of gradient methods.

Let $x_k \in \mathbb{R}^n$. Let d_k be a descent direction at x_k. Let $\alpha > 0$. Then

$$f(x_k + \alpha d_k) = f(x_k) + \alpha \underbrace{\langle \nabla f(x_k), d_k \rangle}_{<0} + o(\alpha).$$

Therefore, if α is small enough,

$$f(x_k + \alpha d_k) < f(x_k).$$

We can set

$$x_{k+1} = x_k + \alpha d_k.$$

Gradient methods

Definition 1

Let $x \in \mathbb{R}^n$ and let $d \in \mathbb{R}^n$. The vector d is called **descent direction** if

$$\langle \nabla f(x), d \rangle < 0.$$

Remark. If $\nabla f(x) \neq 0$, then $d = -\nabla f(x)$ is a descent direction. Indeed,

$$\langle \nabla f(x), d \rangle = -\langle \nabla f(x), \nabla f(x) \rangle = -\|\nabla f(x)\|^2 < 0.$$

Gradient methods

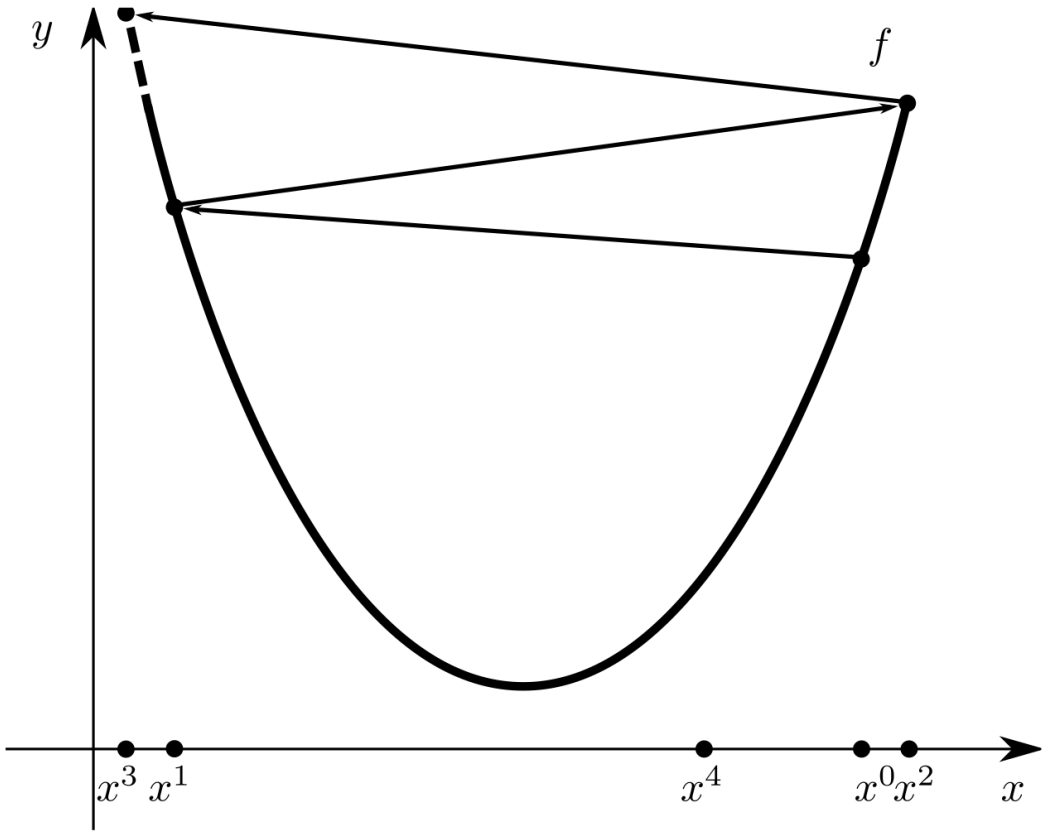
Gradient descent algorithm.

1. Input: $x_0 \in \mathbb{R}^n$, $\varepsilon > 0$.
2. Set $k = 0$.
3. While $\|\nabla f(x_k)\| \geq \varepsilon$, do
   (a) Find a descent direction d_k.
   (b) Find $\alpha_k > 0$ such that $f(x_k + \alpha_k d_k) < f(x_k)$.
   (c) Set $x_{k+1} = x_k + \alpha_k d_k$.
   (d) Set $k = k + 1$.
4. Output: x_k.

Remark. Step (b) is crucial; it is called **line search**.
The real α_k is called **stepsize**.

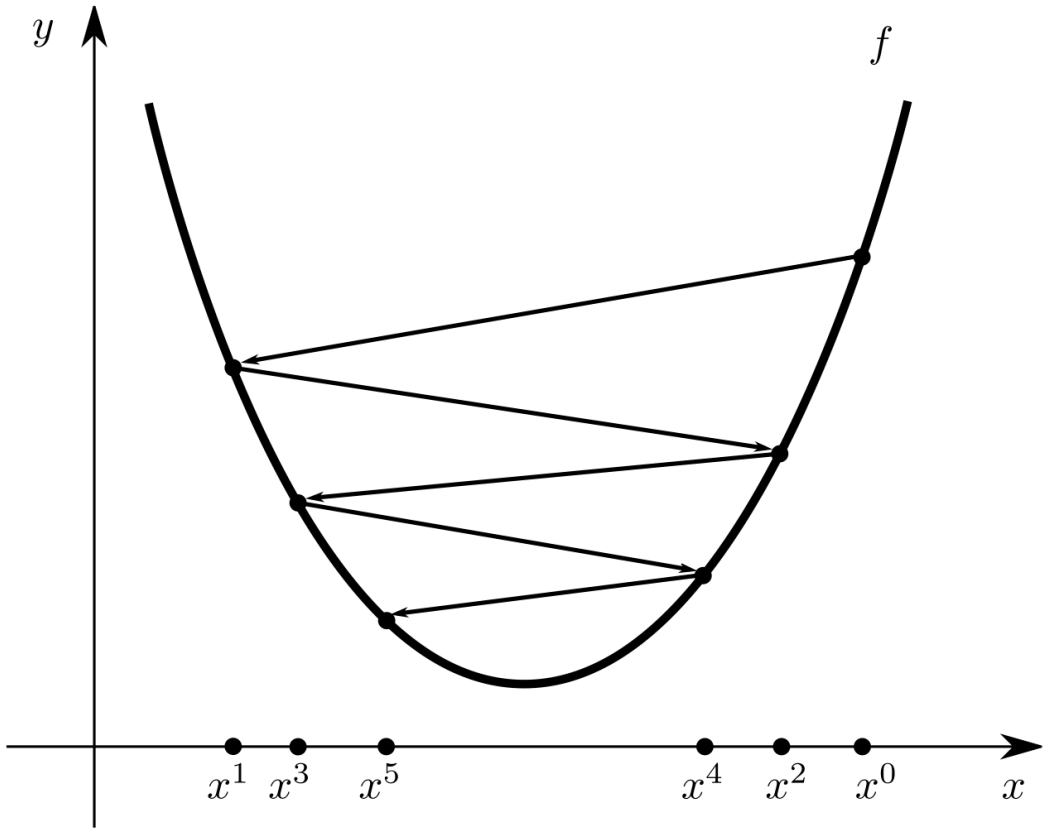
Gradient methods

On the choice of α_k.



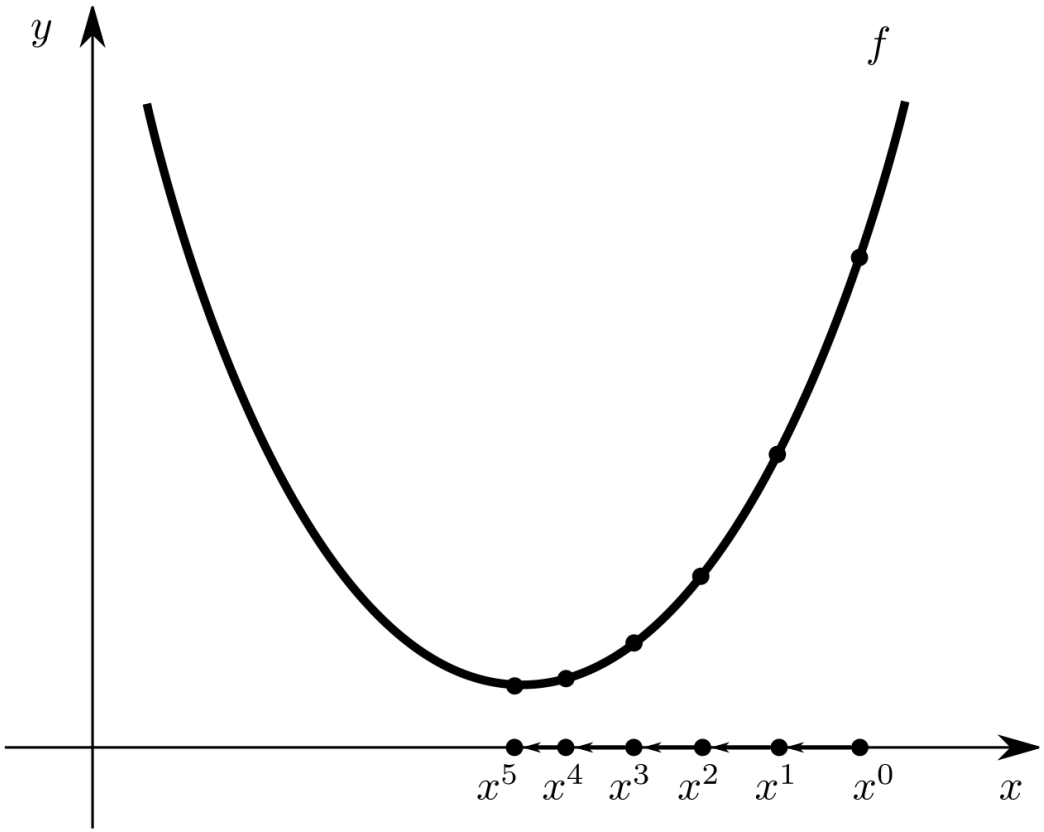
Gradient methods

On the choice of α_k.



Gradient methods

On the choice of α_k.



Gradient methods

On the choice of α_k.

Let us fix $x_k \in \mathbb{R}^n$. Let us define

$$\phi_k : \alpha \in \mathbb{R} \mapsto f(x_k + \alpha d_k).$$

The condition $f(x_k + \alpha_k d_k) < f(x_k)$ is equivalent to

$$\phi_k(\alpha_k) < \phi_k(0).$$

A natural idea: define α_k as a solution to

$$\inf_{\alpha \geq 0} \phi_k(\alpha).$$

Minimizing ϕ_k would take too much time! A **compromise** must be found between simplicity of computation and quality of α.

Observation. Recall that $\phi_k(\alpha) = f(x_k + \alpha d_k)$. We have

$$\phi_k'(\alpha) = \langle \nabla f(x_k + \alpha d_k), d_k \rangle.$$

In particular, since d_k is a descent direction,

$$\phi_k'(0) = \langle \nabla f(x_k), d_k \rangle < 0.$$

Definition 2

Let us fix $0 < c_1 < 1$. We say that α satisfies **Armijo's rule** if

$$\phi_k(\alpha) \leq \phi_k(0) + c_1 \phi_k'(0)\alpha.$$

Gradient methods

Backstepping algorithm for Armijo's rule

1. Input: $c_1 \in (0,1)$, $\beta > 0$, and $\gamma \in (0,1)$.
2. Set $\alpha = \beta$.
3. While α does not satisfy Armijo's rule,
   - Set $\alpha = \gamma\alpha$.
4. Output α.

Definition 3

Let $0 < c_1 < c_2 < 1$. We say that $\alpha > 0$ satisfies **Wolfe's rule** if

$$\phi_k(\alpha) < \phi_k(0) + c_1\phi_k'(0)\alpha \quad \text{and} \quad \phi_k'(\alpha) \geq c_2\phi_k'(0).$$

Armijo condition implies that the function must decrease. The condition $\phi_k'(\alpha) \geq c_2\phi_k'(0)$ implies that the directional derivative must increase sufficiently to approach the local minimum.

Gradient methods

Bisection method for Wolfe's rule

1. Input: $c_1 \in (0,1)$, $c_2 \in (c_1,1)$, $\beta > 0$, $\alpha_{min}, \alpha_{max}$.
2. Set $\alpha = \beta$.

While Wolfe's rule not satified:

1. if α does not satisfy Armijo's rule :
   - Set $\alpha_{max} = \alpha$
   - $\alpha = 0.5(\alpha_{min} + \alpha_{max})$
2. if α satisfies Armijo's rule and $\phi_k'(\alpha) < c_2\phi_k'(0)$, do
   - Set $\alpha_{min} = \alpha$
   - $\alpha = 0.5(\alpha_{min} + \alpha_{max})$
3. Output: α.

Exercise 2

Code the gradient_descent (with a fixed step, with Armijo and Armijo_Wolfe rules). You can use your previous TP on gradient_descent (ENT305).

- with a fixed step: `gradient_descent_fixedstep`
- with Armijo rule: `gradient_descent_armijostep`
- with Wolfe rule: `gradient_descent_wolfestep`

Newton's method

Gradient descent:

$$f(x_k + \alpha d_k) = f(x_k) + \alpha \underbrace{\langle \nabla f(x_k), d_k \rangle}_{<0} + o(\alpha).$$

Main idea. Originally, Newton's method aims at solving non-linear equations of the form

$$F(x) = 0,$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a given continuously differentiable function. Here $F(x) = \nabla f(x)$, we want to solve $\nabla f(x) = 0$.

$$0 = F(x_{k+1}) = F(x_k + \alpha d_k) \simeq F(x_k) + \alpha \langle \nabla F(x_k), d_k \rangle,$$

where $\alpha d_k = x_{k+1} - x_k$.

Newton's method

$$\alpha d_k = x_{k+1} - x_k.$$

$$\begin{aligned} 0 = F(x_{k+1}) &\simeq F(x_k) + \langle \nabla F(x_k), x_{k+1} - x_k \rangle. \\ &= F(x_k) + DF(x_k)(x_{k+1} - x_k). \end{aligned}$$

$$0 = F(x_k) + DF(x_k)(x_{k+1} - x_k),$$

$$-F(x_k) = DF(x_k)(x_{k+1} - x_k),$$

$$-DF(x_k)^{-1} F(x_k) = x_{k+1} - x_k$$

$$x_{k+1} = x_k - D^2 f(x_k)^{-1} \nabla f(x_k).$$

Newton's method

$$x_{k+1} = x_k - D^2 f(x_k)^{-1} \nabla f(x_k).$$

Remarks.

- If there exists $\bar{x}$ such that $F(\bar{x}) = 0$ and $DF(\bar{x})$ is regular, then for x_0 close enough to $\bar{x}$, the sequence $(x_k)_{k \in \mathbb{N}}$ is well-posed and converges "**quickly**" to $\bar{x}$.
- On the other hand, if x_0 is far away from $\bar{x}$, there is **no guaranty** of convergence.

Newton's method

Optimization with Newton's method.

- Newton's formula can be written in the form:

$$x_{k+1} = x_k + \alpha_k d_k,$$

where

$$\alpha_k = 1 \quad \text{and} \quad d_k = -D^2 f(x_k)^{-1} \nabla f(x_k).$$

- If $D^2 f(x_k)$ is positive definite (and $\nabla f(x_k) \neq 0$), then $D^2 f(x_k)^{-1}$ is also positive definite, and therefore d_k is descent direction:

$$\langle \nabla f(x_k), d_k \rangle = -\langle \nabla f(x_k), D^2 f(x_k)^{-1} \nabla f(x_k) \rangle < 0.$$

Newton's method

Globalised Newton's method.

1. Input: $x_0 \in \mathbb{R}^n$, $\varepsilon > 0$, a linesearch rule (Armijo, Wolfe,...).
2. Set $k = 0$.
3. While $\|\nabla f(x_k)\| \geq \varepsilon$, do
   (a) If $-D^2 f(x_k)^{-1} \nabla f(x_k)$ is computable and is a descent direction, set $d_k = -D^2 f(x_k)^{-1} \nabla f(x_k)$, otherwise set $d_k = -\nabla f(x_k)$.
   (b) If $\alpha = 1$ satisfies the linesearch rule, then set $\alpha_k = 1$. Otherwise, find α_k with an appropriate method.
   (c) Set $x_{k+1} = x_k + \alpha_k d_k$.
   (d) Set $k = k + 1$.
4. Output: x_k.

Newton's method

Comments.

- Convergence is fast.
- The numerical computation of $D^2 f(x_k)$ may be **very time consuming** and may generate storage issues because of n^2 figures in general).
- **Quasi-Newton** methods construct a sequence of positive definite matrices H_k such that $H_k \approx D^2 f(x_k)^{-1}$. The matrix H_k can be stored efficiently (with $O(n)$ figures). Then $d_k = -H_k \nabla f(x_k)$ is a descent direction. Good speed of convergence is achieved. $\rightarrow$ **The ideal compromise!**

Newton's method

Exercise 3

Code the Newton algorithm. You first need to compute the hessian matrix.
Compare your result with $\alpha = 1$ and α given by Armijo/Wolfe step.

Stochastic gradient method

Instead of computing full gradient, we compute the partial derivatives for some i
$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$.
$d_k = -\nabla f_{i_k}(x_k)$ where i_k are random uniform variables in $\{1, \ldots, n\}$.
learning rate decay:
Instead of $x = x + \alpha d$, we take

$$\alpha_k = \frac{\alpha_0}{(k+1)^\kappa},$$

where $\kappa \in [0.5, 1]$

Exercise 4

Code the Stochastic gradient algorithm.