Otimização e algoritmos
2020-2021 1º semestre
Projecto

Grupo n.º14

António Pereira – 90019

Bernardo Taveira - 90031

Miguel Amaral – 90150

Diogo Alves - 86980

Part 1 - Controlar um robô

Nesta parte, queremos controlar um robô de acordo com 4 desejos:

- **Transferência**:

$$x(0) = \begin{bmatrix} p_{inicial} \\ 0 \end{bmatrix} \Rightarrow \quad x(T) = \begin{bmatrix} p_{final} \\ 0 \end{bmatrix}$$

- **Controlo limitado:**

$$||u(t)||_2 \leq U_{max} \qquad \text{onde} \qquad (||u(t)||_2 = (u_1^2 + ... + u_t^2)^{\frac{1}{2}})$$

Part 1 - Controlar um robô

- **Pontos Intermédios:**

$$p(\tau_k) = w_k, k \in 1 \leq k \leq K$$

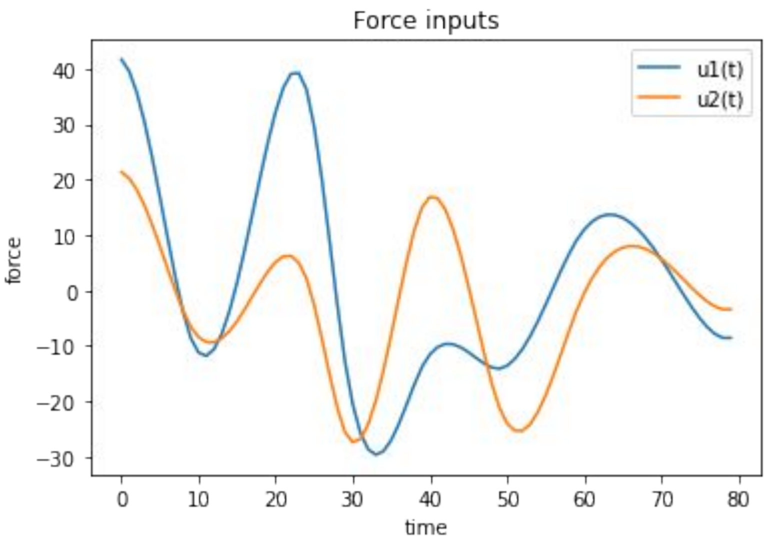
- **Controlo simples:**

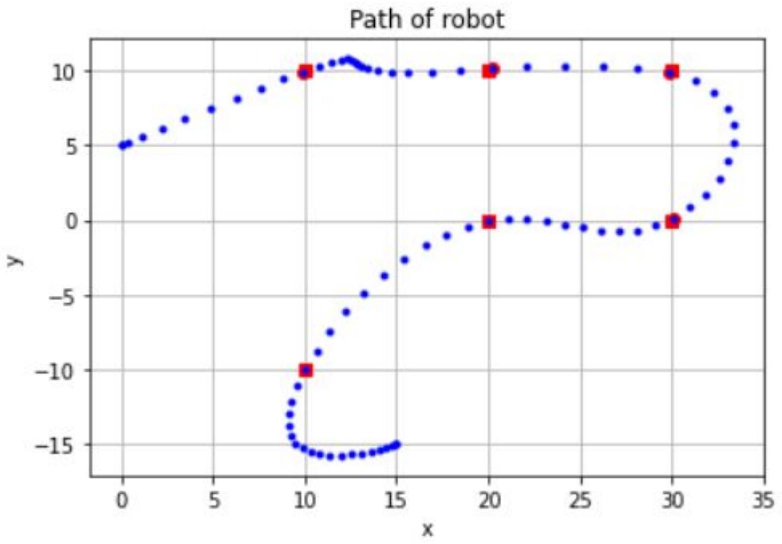
$$u(t) = u(t-1), for\ most\ t \in 1, .., T-1$$

Task 1 - Regularizador l_2^2

$$
\begin{aligned}
\underset{x,u}{\text{minimize}} \quad & \sum_{k=1}^{k} ||Ex(\tau_k) - w_k||^2 + \lambda \sum_{t=1}^{T-1} ||u(t) - u(t-1)||_2^2 \\
\text{subject to} \quad & x(0) = x_{initial} \\
& x(T) = x_{final} \\
& ||u(t)|| \leq U_{max} \\
& x(t+1) = Ax(t) + Bu(t)
\end{aligned}
$$

Task 1


Force inputs
u1(t)
u2(t)
force
time
40
30
20
10
0
−10
−20
−30
0 10 20 30 40 50 60 70 80

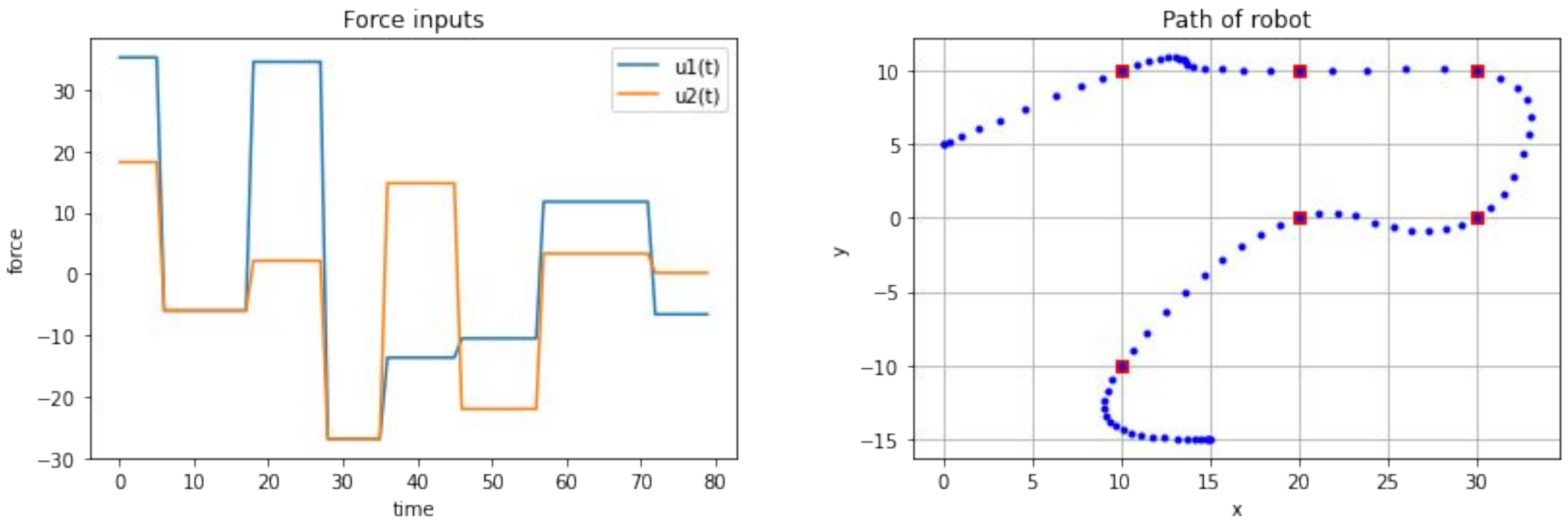

Path of robot
y
x
10
5
0
−5
−10
−15
0 5 10 15 20 25 30 35

Task 2 - Regularizador l_2

$$\underset{x,u}{\text{minimize}} \quad \sum_{k=1}^{k} ||Ex(\tau_k) - w_k||^2 + \lambda \sum_{t=1}^{T-1} ||u(t) - u(t-1)||_2$$

$$\begin{aligned} \text{subject to} \quad & x(0) = x_{initial} \\ & x(T) = x_{final} \\ & ||u(t)|| \leq U_{max} \\ & x(t+1) = Ax(t) + Bu(t) \end{aligned}$$

Task 2


Force inputs
u1(t)
u2(t)
force
time
30
20
10
0
−10
−20
−30
0
10
20
30
40
50
60
70
80
Path of robot
y
x
10
5
0
−5
−10
−15
0
5
10
15
20
25
30

Task 3 - Regularizador l_1

$$\underset{x,u}{\text{minimize}} \quad \sum_{k=1}^{k} ||Ex(\tau_k) - w_k||^2 + \lambda \sum_{t=1}^{T-1} ||u(t) - u(t-1)||_1$$

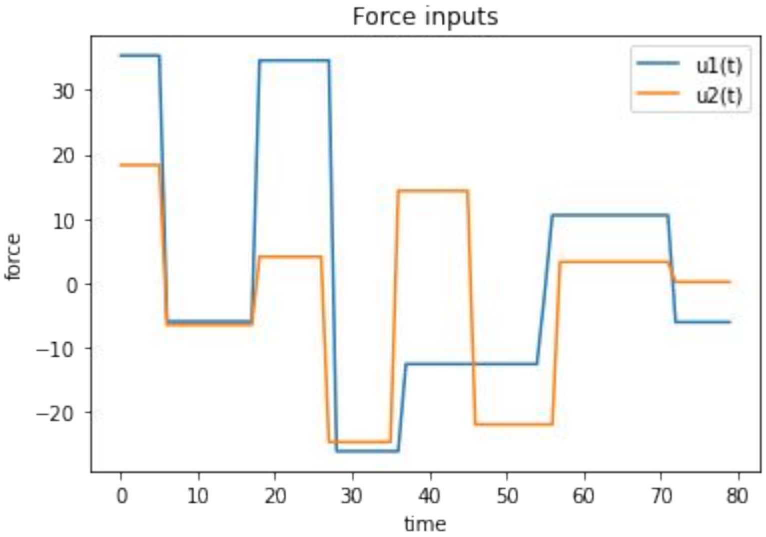
$$\text{subject to} \quad x(0) = x_{initial}$$

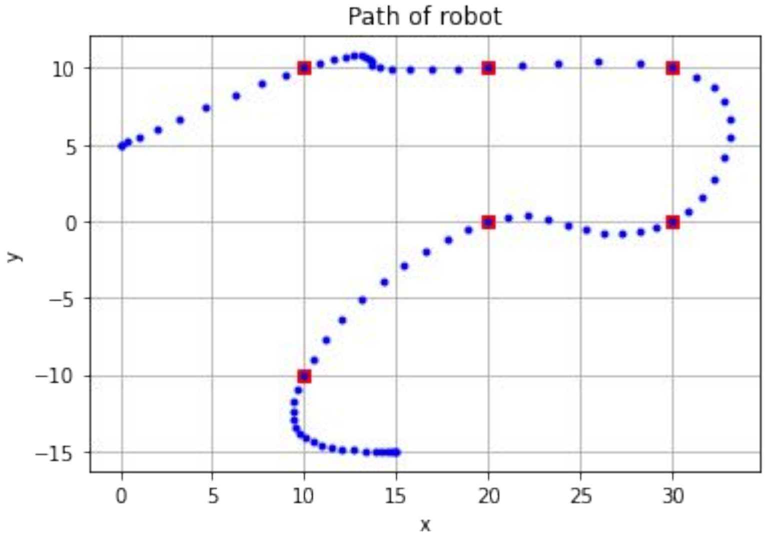
$$x(T) = x_{final}$$

$$||u(t)|| \leq U_{max}$$

$$x(t+1) = Ax(t) + Bu(t)$$

Task 3


Force inputs
u1(t)
u2(t)
force
time
30
20
10
0
−10
−20
0
10
20
30
40
50
60
70
80


Path of robot
y
x
10
5
0
−5
−10
−15
0
5
10
15
20
25
30

Task 4 - Comparação

1.

λ	Waypoint deviation	Optimal Control Signal Changes
10^{-3}	0.12653634833380814	79
10^{-2}	0.8291418701691656	79
10^{-1}	2.206581300320847	79
10^{0}	3.7140231509139032	79
10^{1}	5.627997634931546	79
10^{2}	11.029778990641217	79
10^{3}	15.394723280425408	79

2.

λ	Waypoint deviation	Optimal Control Signal Changes
10^{-3}	0.0075260515065971026	7
10^{-2}	0.07490042785104145	7
10^{-1}	0.7042847620936804	9
10^{0}	2.902559005393649	5
10^{1}	5.360819048216747	3
10^{2}	12.656066807343135	2
10^{3}	16.27485839271584	1

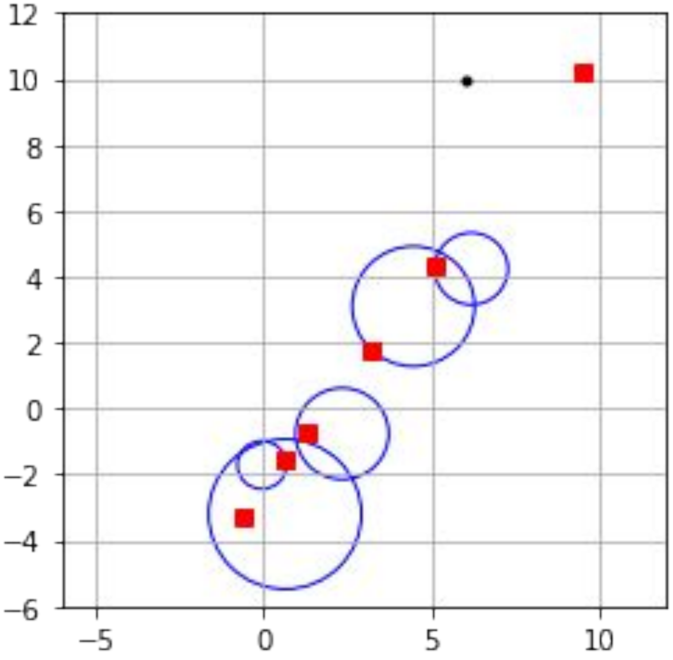
3.

λ	Waypoint deviation	Optimal Control Signal Changes
10^{-3}	0.010834711962522931	11
10^{-2}	0.10738083162217865	10
10^{-1}	0.8922241950430558	13
10^{0}	2.8940849000558297	9
10^{1}	5.421924524677606	5
10^{2}	13.094034792975402	2
10^{3}	16.075380063793776	2

- Os dois primeiros desejos são sempre cumpridos
- O 3º desejo depende do valor de λ
- O 4º desejo é cumprido pelas tasks 2 e 3

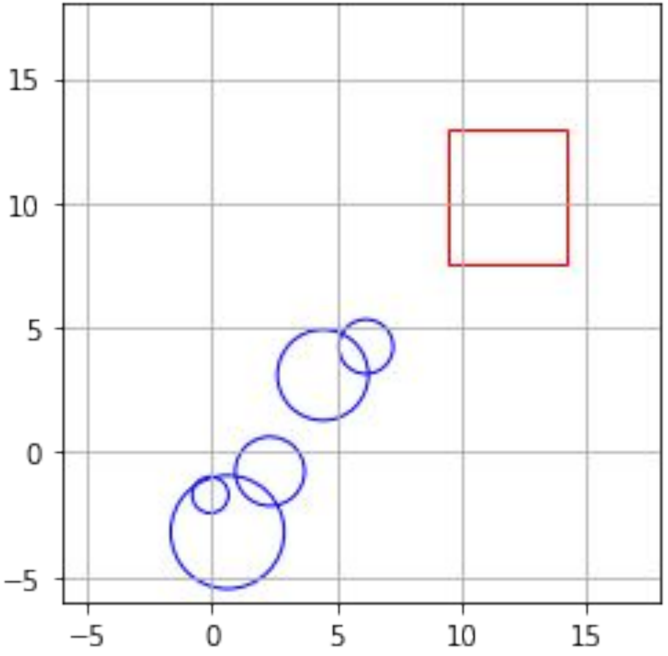
Task 5 - Localização de um alvo móvel

$$\begin{array}{ll} \underset{p_0, v}{\text{minimize}} & \|p_0 + t^* v - x^*\|^2 \\ \text{subject to} & R_k \geq \|p_0 + \tau_k v - c_k\| \end{array}$$



Task 6 - Menor retângulo envolvente

$$a_1 = \min_{p_0, v} \quad p_0^{(1)} + t^* v^{(1)}$$
$$\text{subject to} \quad R_k \geq ||p_0 + \tau_k v - c_k||$$
$$a_2 = \max_{p_0, v} \quad p_0^{(1)} + t^* v^{(1)}$$
$$\text{subject to} \quad R_k \geq ||p_0 + \tau_k v - c_k||$$
$$b_1 = \min_{p_0, v} \quad p_0^{(2)} + t^* v^{(2)}$$
$$\text{subject to} \quad R_k \geq ||p_0 + \tau_k v - c_k||$$
$$b_2 = \max_{p_0, v} \quad p_0^{(2)} + t^* v^{(2)}$$
$$\text{subject to} \quad R_k \geq ||p_0 + \tau_k v - c_k||$$

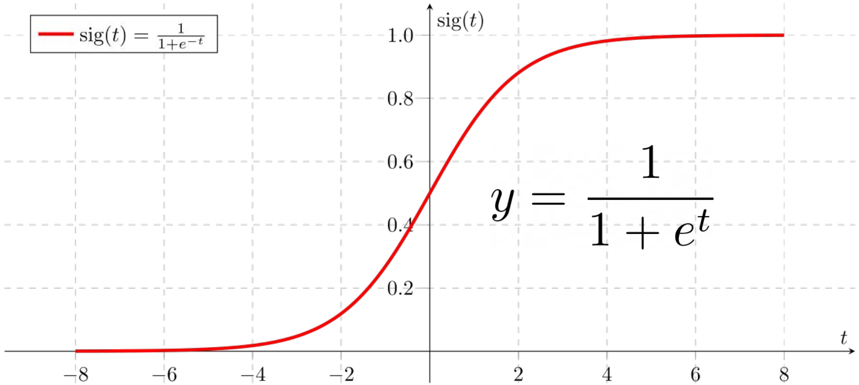


Parte 2

- Objectivo: Adaptar um modelo probabilístico a dados

- Regressão logística para classificação binária

Regressão logística



$$y = \frac{1}{1 + e^{t}}$$

Sigmoid para expressar a probabilidade de ser 0

$$t = s^T x - r$$

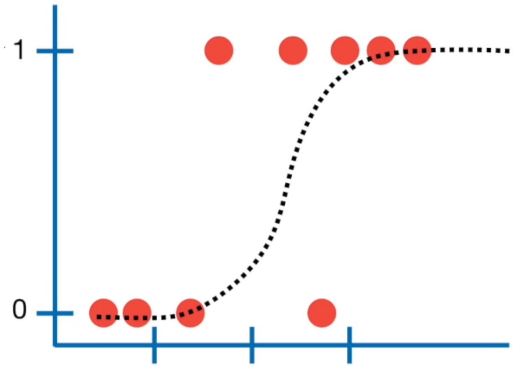
Função linear como input da sigmoid

$$y = \frac{1}{1 + e^{s^T x - r}}$$

O vetor *s* tem de ser obtido. Uma boa forma de o obter é fazer *fitting* aos dados que temos

Função de custo

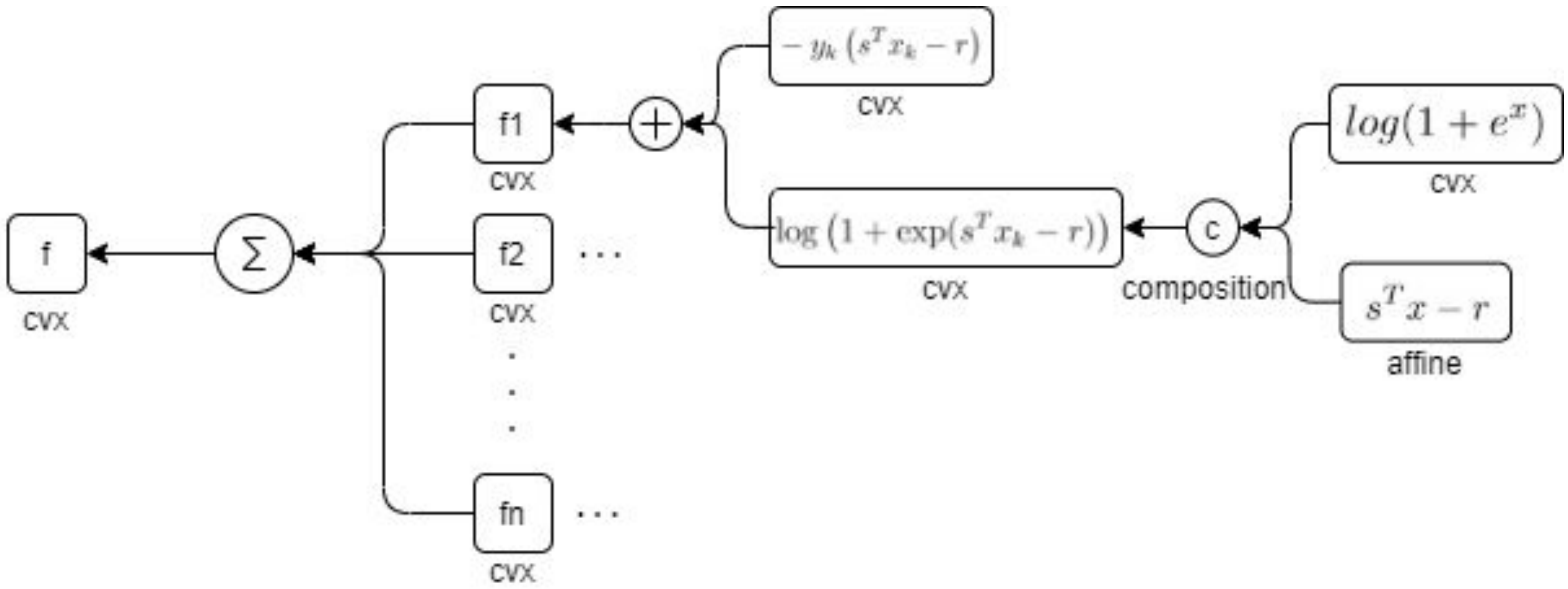
$$\underset{(s,r)\in\mathbf{R}^n\times\mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{K}\sum_{k=1}^{K}\left(\log\left(1+\exp(s^T x_k - r)\right) - y_k\left(s^T x_k - r\right)\right)}_{f(s,r)}. \qquad (1)$$



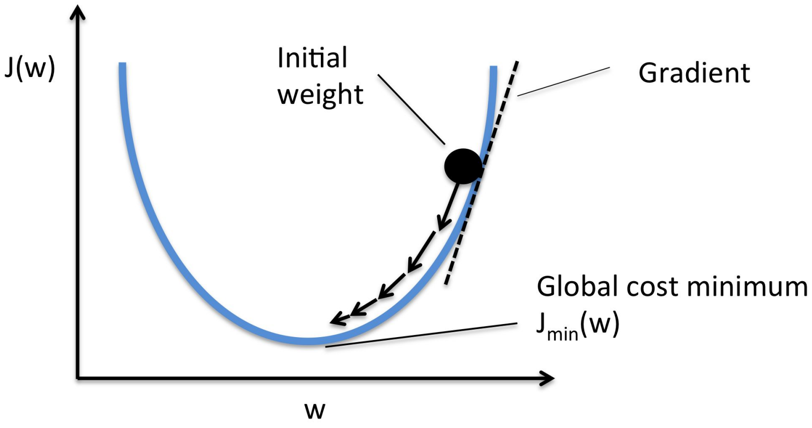
Adaptar os vectores s e r a conjuntos de dados

Task 1 - Provar Convexidade

$$\underset{(s,r)\in\mathbf{R}^n\times\mathbf{R}}{\text{minimize}} \quad \underbrace{\frac{1}{K}\sum_{k=1}^{K}\left(\log\left(1+\exp(s^T x_k - r)\right) - y_k\left(s^T x_k - r\right)\right)}_{f(s,r)}.$$



Gradient Descent Method



Quanto mais próximo da solução, menor será o seu avanço

Task 2

Resolver o problema de otimização (1) com o Gradient Descent Method expresso na seguinte imagem:

Gradient descent algorithm

choose $x_0 \in \mathbf{R}^n$ and tolerance $\epsilon > 0$
set $k = 0$
loop
    compute $g_k = \nabla f(x_k)$
    check stopping criterion: if $\|g_k\| < \epsilon$ stop
    set $d_k = -g_k$
    find $\alpha_k > 0$ with the backtracking subroutine
    update $x_{k+1} = x_k + \alpha_k d_k$
    $k \leftarrow k + 1$
end loop

Parâmetros de inicialização e paragem

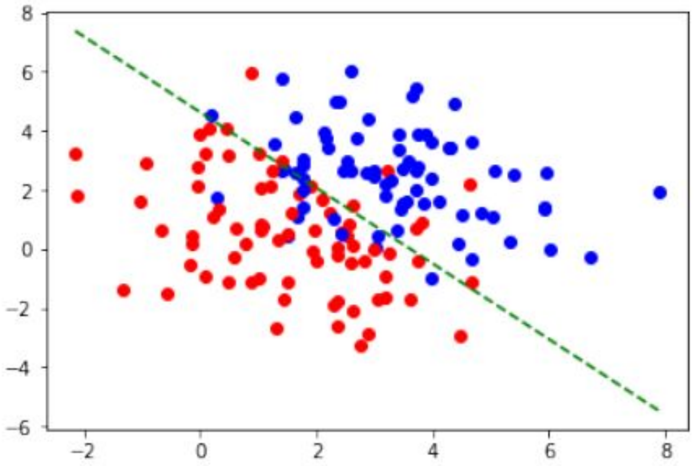
- $s_0 = (-1, -1)$
- $r_0 = 0$
- $\varepsilon = 10^{-6}$

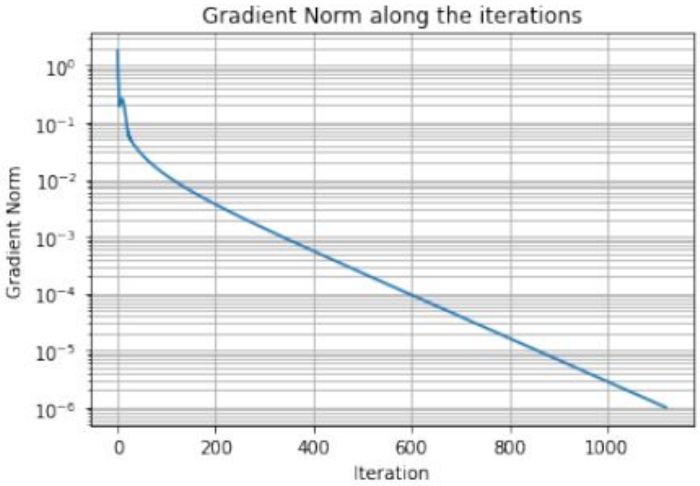
Parâmetros de backtracking

$$\widehat{\alpha} = 1, \quad \gamma = 10^{-4}, \quad \beta = \frac{1}{2}$$

Task 2

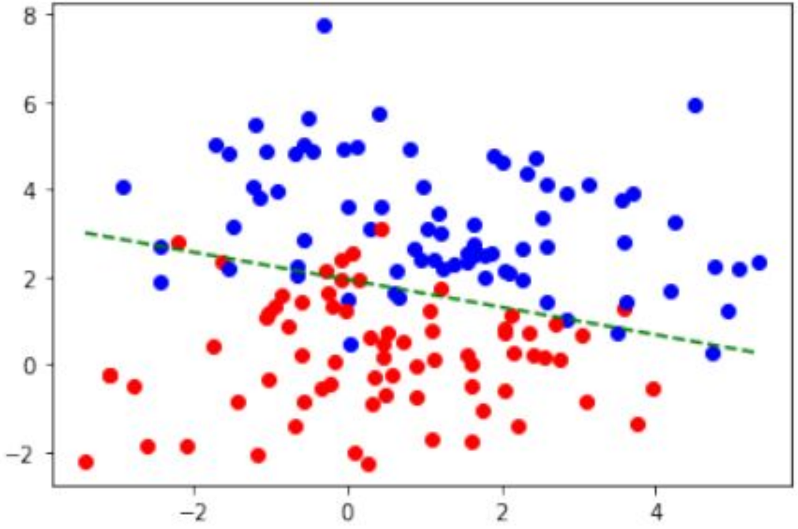
- s = [1.34953921 1.05397136]
- r = 4.881542559495996

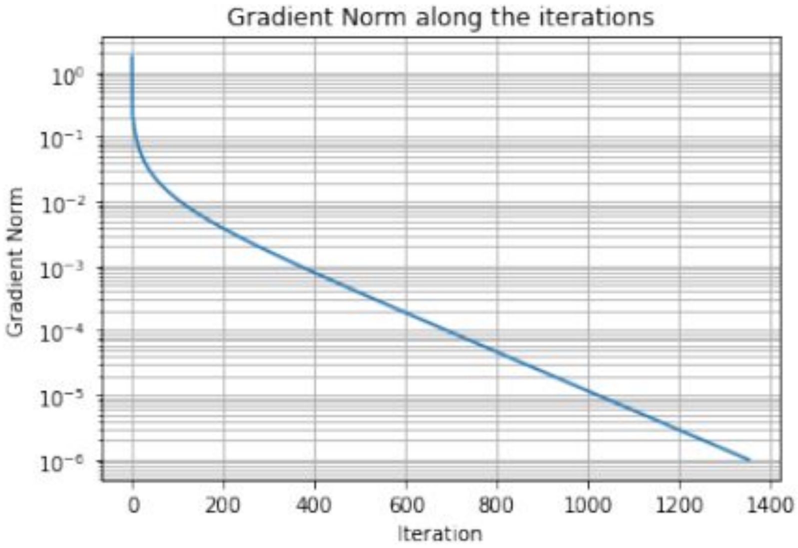




Task 3 - Refazer task 2 para dataset diferente

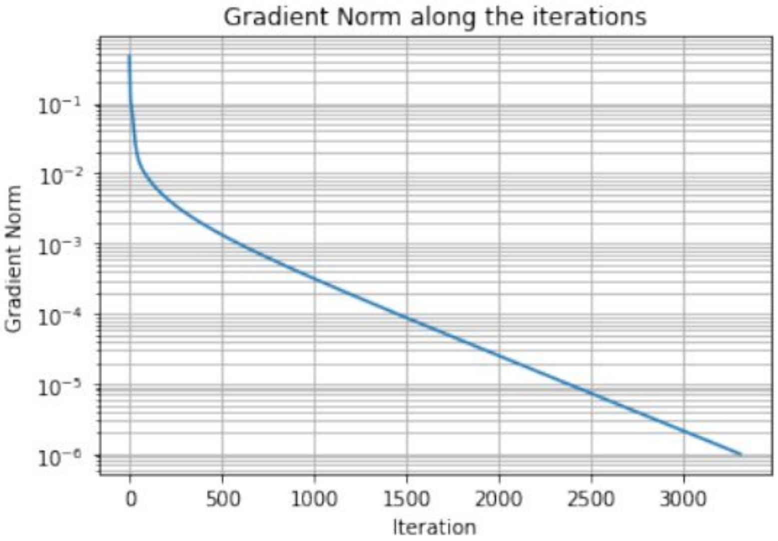
- s = [0.74021451 2.35765419]
- r = 4.555298359498353





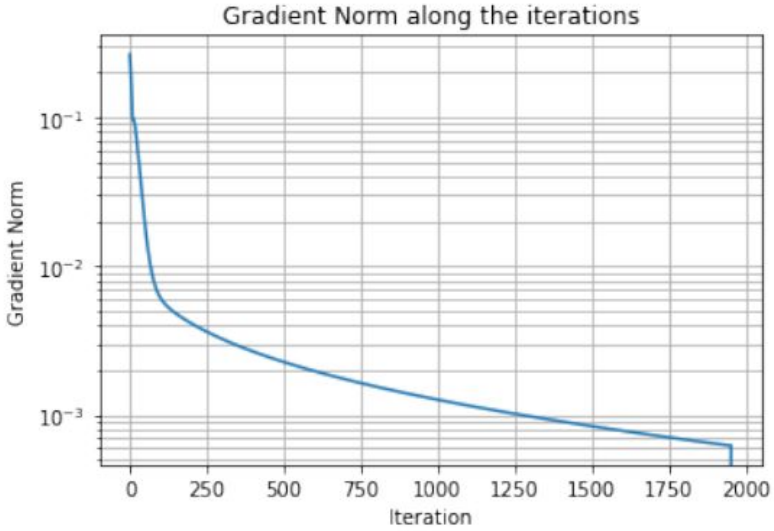
Task 4 - Refazer task 2 para dataset diferente

As dimensões do dataset fazem que este não seja divisível com uma só linha



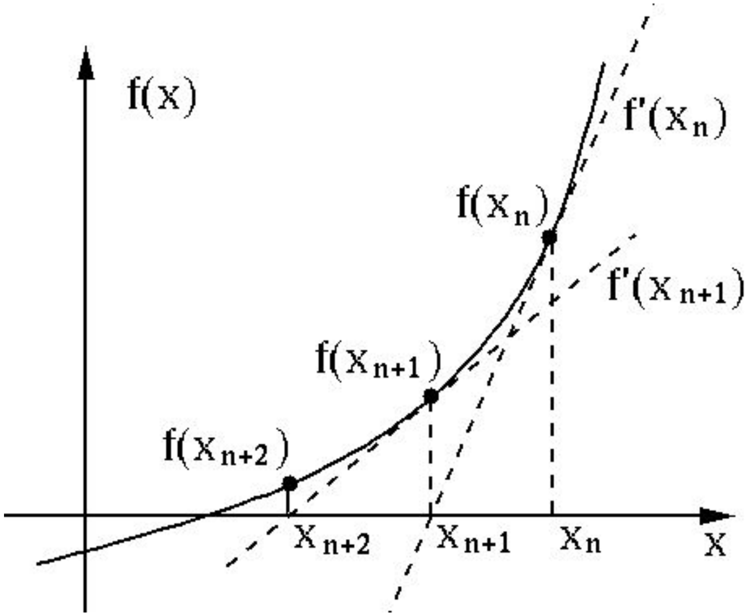
Data3.mat

Task 4 - Refazer task 2 para dataset diferente



Data4.mat

Método de Newton



O método de Newton é um algoritmo para encontrar raízes

Método de Newton

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x)$$

- if $x^\star$ is a minimum then

$$\begin{cases} \dfrac{\partial f}{\partial x_1}(x^*) & = & 0 \\ \dfrac{\partial f}{\partial x_2}(x^*) & = & 0 \\ & \vdots & \\ \dfrac{\partial f}{\partial x_n}(x^*) & = & 0 \end{cases}$$

Para transformar este algoritmo de procura de raízes num algoritmo de otimização, podemos encontrar os zeros da derivada da função de custo:

$$\nabla f(x^\star) = 0$$

Método de Newton

choose $x_0 \in \mathbf{R}^n$ and tolerance $\epsilon > 0$
set $k = 0$
loop
  compute $g_k = \nabla f(x_k)$
  check stopping criterion: if $\|g_k\| < \epsilon$ stop
  set $d_k = -\left(\nabla^2 f(x_k)\right)^{-1} g_k$
  find $\alpha_k > 0$ with the backtracking subroutine
  update $x_{k+1} = x_k + \alpha_k d_k$
  $k \leftarrow k + 1$
end loop

Task 5 - Encontrar o gradiente e a hessiana de p(x)

Dado:

$$p(x) = \sum_{k=1}^{K} \phi(a_k^T x)$$

O seu gradiente é dado por:

$$\nabla p(x) = \sum_{k=1}^{K} a_k \phi'(a_k^T x)$$

E a hessiana de p é dado por:

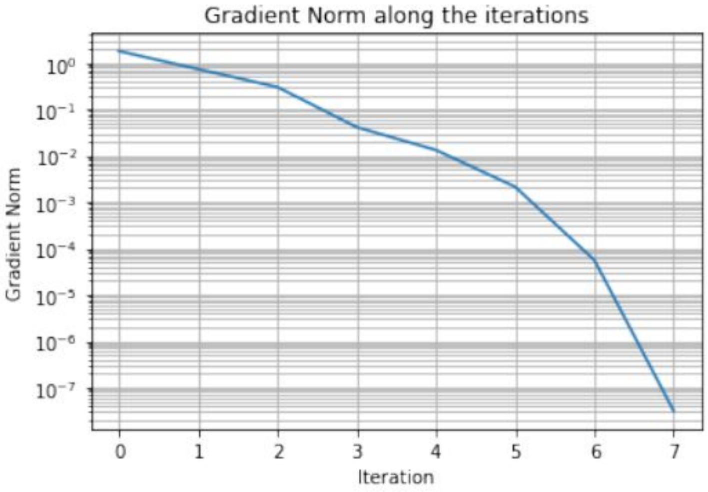
$$\nabla^2 p(x) = \sum_{k=1}^{K} a_k^2 \phi''(a_k^T x)$$

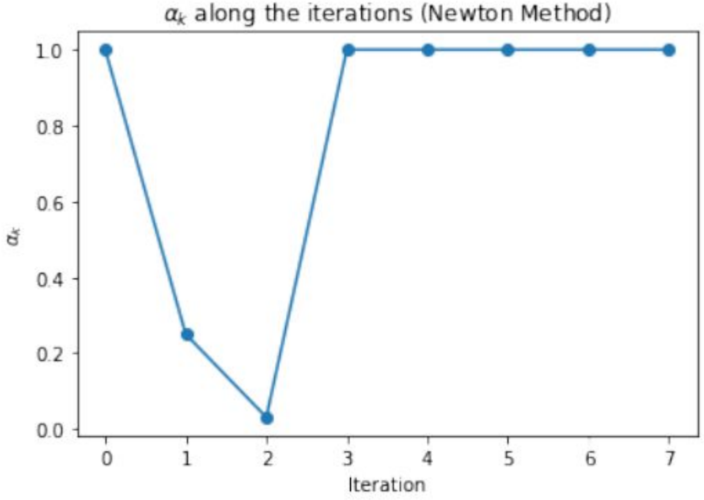
Task 6

Resolver problema de optimização (1) usando o método de Newton em todos os datasets, com:

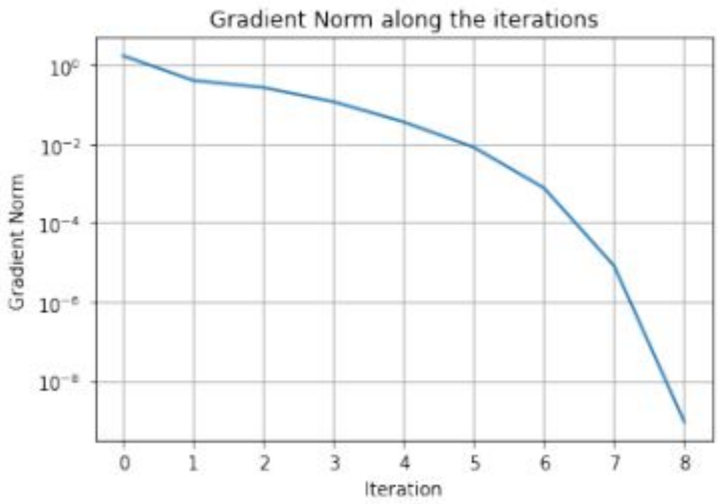
- $s_0 = (-1, -1, . .., -1)$
- $r_0 = 0$
- $\varepsilon = 10^{-6}$
- São usados os mesmo parâmetros de backtracking

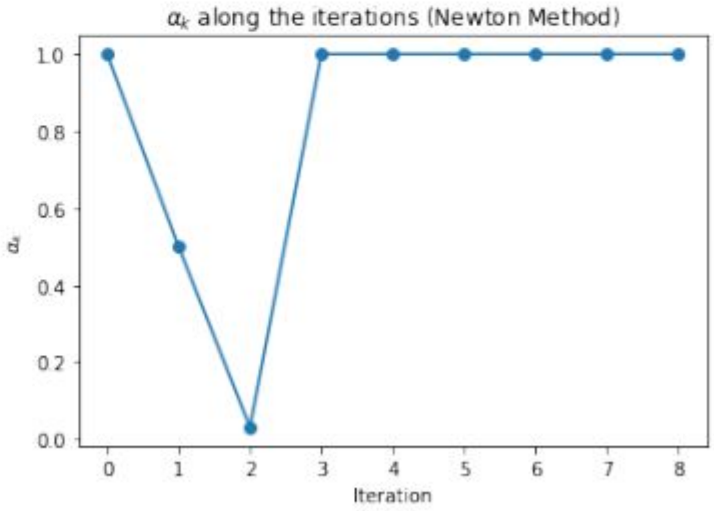
Task 6 data1.mat



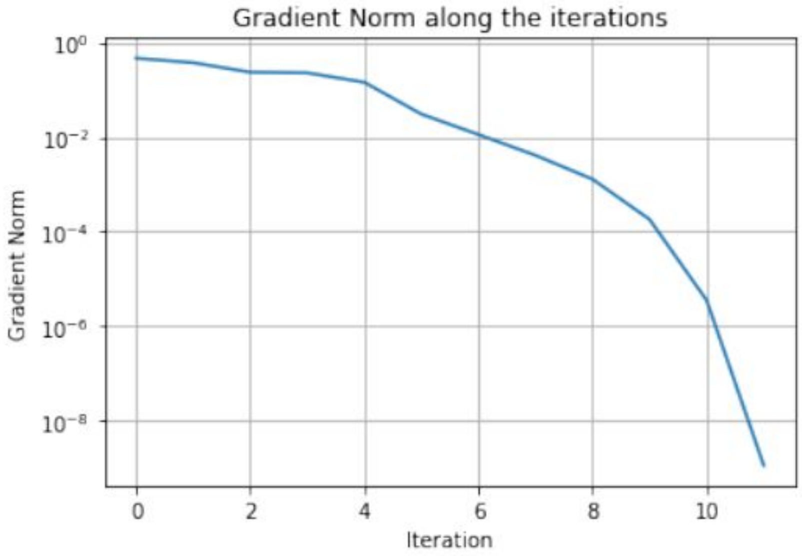


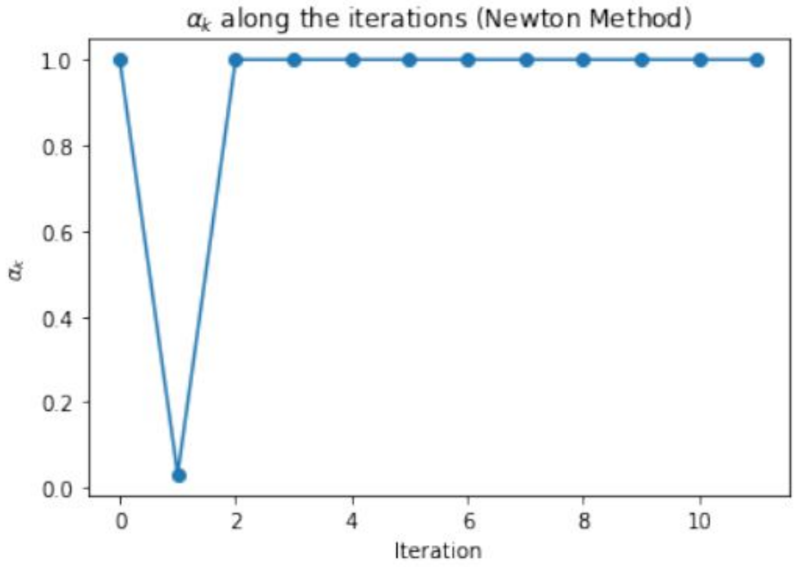
Task 6 data2.mat



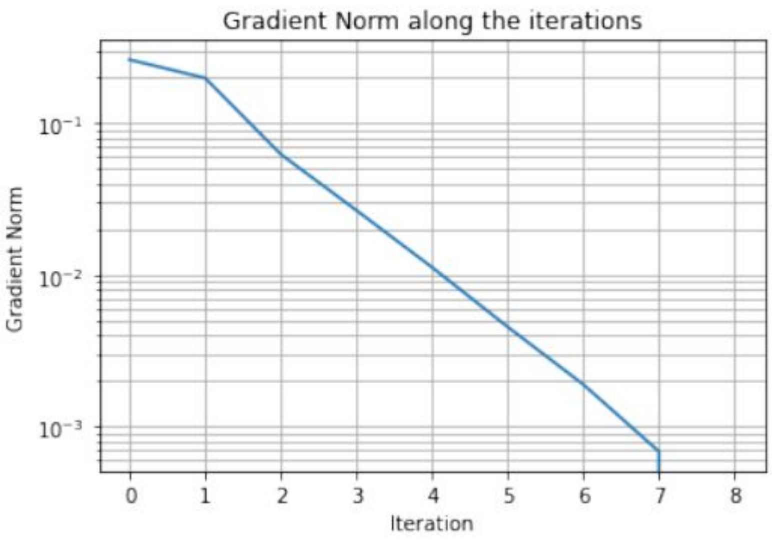


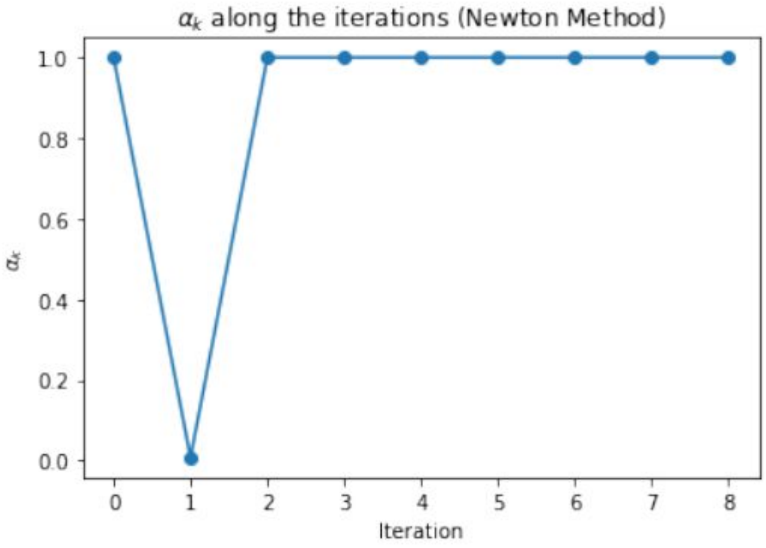
Task 6 data3.mat





Task 6 data4.mat





Task 7

- Comentar sobre os resultados obtidos
- Comparação entre Gradient Descent Method e Newton Method

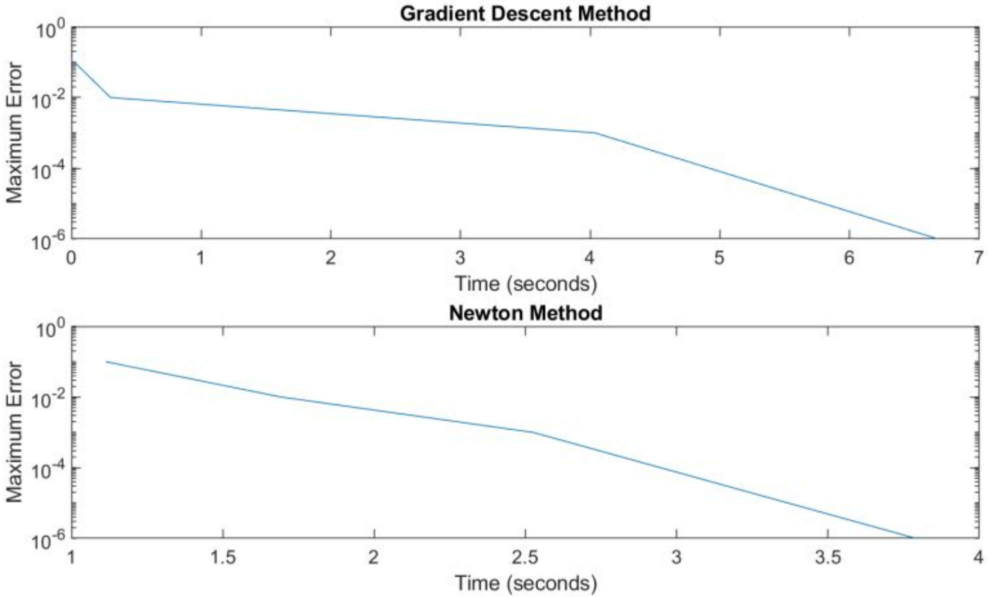
Task 7

File	n	k	Iterations GDM	Execution time GDM	Iterations NM	Execution Time NM
data1.mat	2	150	1121	0.24806	8	0.011002
data2.mat	2	150	1353	0.201756	9	0.007001
data3.mat	30	500	3309	0.64412	12	0.03249
data4.mat	100	8000	1955	6.77544	9	4.29659
data3_mod.mat	30	150	-	-	21	0.02699
data4_mod.mat	100	150	285	0.071908	18	0.105

Task 7

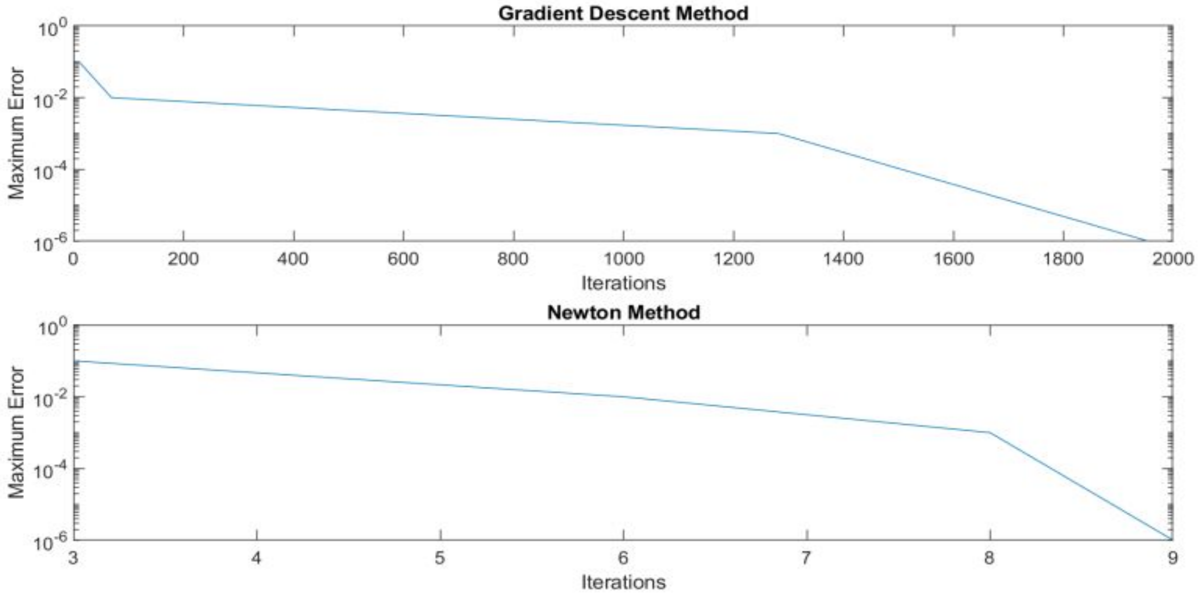
File	Max Error	Iterations GDM	Time GDM	Iterations NM	Time NM
data4.mat	10^{-1}	10	0.023036	3	1.11299
data4.mat	10^{-2}	69	0.30073	6	1.69218
data4.mat	10^{-3}	1282	4.03788	8	2.525
data4.mat	10^{-6}	1955	6.67895	9	3.79

Task 7



Gradient Descent Method
Maximum Error
10^0
10^{-2}
10^{-4}
10^{-6}
0 1 2 3 4 5 6 7
Time (seconds)
Newton Method
Maximum Error
10^0
10^{-2}
10^{-4}
10^{-6}
1 1.5 2 2.5 3 3.5 4
Time (seconds)

Task 7



Gradient Descent Method
Maximum Error
10^0
10^{-2}
10^{-4}
10^{-6}
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Iterations
Newton Method
Maximum Error
10^0
10^{-2}
10^{-4}
10^{-6}
3
4
5
6
7
8
9
Iterations

Parte 3

- Objetivo: Reduzir número de features de um dataset

- Multidimensional Scaling (MDS) - Preservar distâncias euclidianas entre vetores do dataset

Task 1

- Construir matriz *D* - distâncias euclidianas entre cada par de vetores

$$D_{mn} = \|x_m - x_n\|_2$$

$x_i \in \mathbf{R}^p$ (p – número total de features)

- Dataset *data_opt.csv* (p=10, N=200):
  - D[2][3] = 5.8749
  - D[4][5] = 24.3769
  - Distância máxima: D[33][134] = 83.0030

Task 2

- Objetivo: Resolver problema de otimização usando método LM

$$\underset{y}{\text{minimize}} \quad \underbrace{\sum_{m}^{N} \sum_{n>m} \left(\|y_m - y_n\| - D_{mn} \right)^2}_{f(y)}$$

$$y_i \in \mathbf{R}^k \ (k \text{ – número de features no espaço reduzido})$$

- Decompor função *f* em soma de quadrados:

$$f(y) = \sum_{m=1}^{N} \sum_{n>m} (f_{nm}(y))^2$$

$$f_{nm}(y) = ||y_m - y_n|| - D_{mn} = \sqrt{y_m^T y_m - 2y_m^T y_n + y_n^T y_n} - D_{mn}$$

Task 2

→ Calcular gradiente de f:

$$\nabla f(y) = \sum_{m=1}^{N} \sum_{n>m} 2\, f_{nm}(y) \nabla f_{nm}(y)$$

$$\nabla f_{nm}(y) = \begin{array}{c} \begin{bmatrix} 0 \\ \vdots \\ \dfrac{y_m - y_n}{||y_m - y_n||} \\ \vdots \\ \dfrac{y_n - y_m}{||y_m - y_n||} \\ \vdots \\ 0 \end{bmatrix} \end{array} \begin{array}{c} [1] \\ \vdots \\ [m] \\ \vdots \\ [n] \\ \vdots \\ [N] \end{array}$$

Task 2

- Converter este problema de otimização num problema de mínimos quadrados:

$$\underset{y}{\text{minimize}} \sum_{m=1}^{N} \sum_{n>m} \left(f_{mn}(y_k) + \nabla f_{mn}(y_k)^T (y - y_k)\right)^2 + \lambda_k \left|\left|y - y_k\right|\right|^2$$

- Da seguinte forma: $\underset{y}{\text{minimize}} \ \left|\left|Ay - b\right|\right|^2$

$$A = \begin{bmatrix} \nabla f_{12}(y_k)^T \\ \nabla f_{13}(y_k)^T \\ \vdots \\ \nabla f_{1N}(y_k)^T \\ \nabla f_{23}(y_k)^T \\ \vdots \\ \sqrt{\lambda_k} I \end{bmatrix} \qquad b = \begin{bmatrix} \nabla f_{12}(y_k)^T y_k - f_{12}(y_k) \\ \nabla f_{13}(y_k)^T y_k - f_{13}(y_k) \\ \vdots \\ \nabla f_{1N}(y_k)^T y_k - f_{1N}(y_k) \\ \nabla f_{23}(y_k)^T y_k - f_{23}(y_k) \\ \vdots \\ \sqrt{\lambda_k} y_k \end{bmatrix}$$

- Trivial de resolver em *python*:

```
y = np.linalg.lstsq(A,b)
```

Task 2

LM algorithm (for nonlinear least-squares)

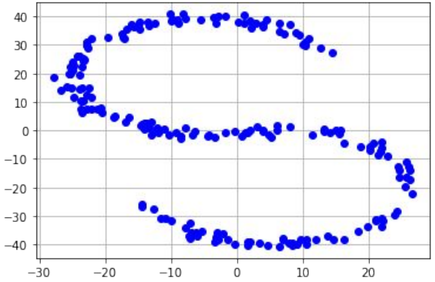
choose $x_0 \in \mathbf{R}^n$, $\lambda_0 > 0$, and tolerance $\epsilon > 0$
set $k = 0$
loop
  compute $g_k = \nabla f(x_k)$
  check stopping criterion: if $\|g_k\| < \epsilon$ stop
  solve

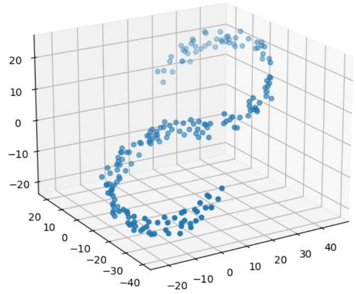
$$\widehat{x}_{k+1} = \underset{x \in \mathbf{R}^n}{\operatorname{argmin}} \sum_{p=1}^{P} \left(f_p(x_k) + \nabla f_p(x_k)^T (x - x_k)\right)^2 + \lambda_k \|x - x_k\|^2$$

  if $f\left(\widehat{x}_{k+1}\right) < f(x_k)$ [valid step]
    $x_{k+1} = \widehat{x}_{k+1}$
    $\lambda_{k+1} = 0.7\lambda_k$
   else [null step]
    $x_{k+1} = x_k$
    $\lambda_{k+1} = 2\lambda_k$
  $k \leftarrow k + 1$
end loop

Task 3

- Algoritmo LM para dataset *data_opt.csv*, para k=2 e k=3:

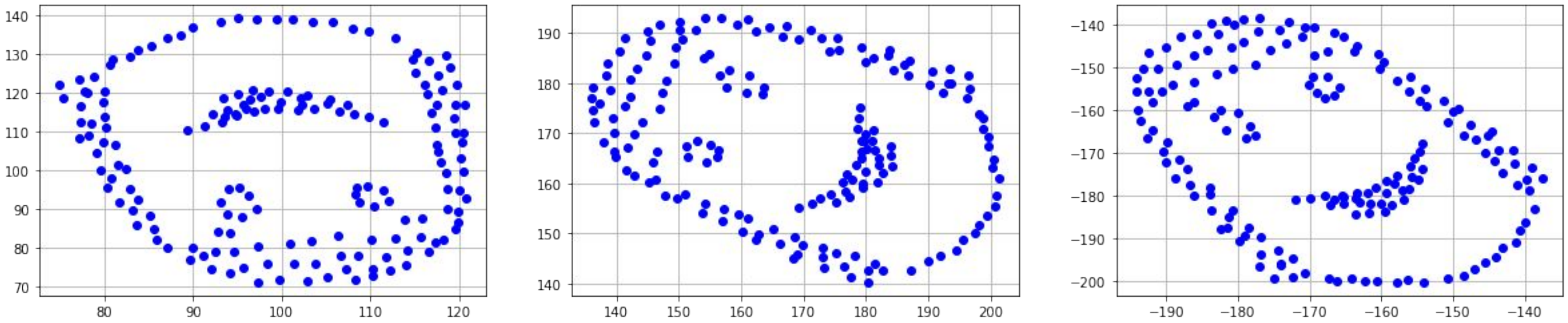




- Ambas as soluções apresentam a mesma forma – um “S”
- Nesta forma as posições relativas entre os vetores y_i fazem com que a função *f(y)* seja mínima.

Task 4

- Objetivo: Resolver problema de otimização com diferentes inicializações (para dataset *dataProj.csv*)



Task 4

- Soluções são rotações e translações umas das outras

$$f(y) = \sum_{m=1}^{N} \sum_{n>m} \left(||y_m - y_n|| - D_{mn}\right)^2$$

- Função *f* não depende do valor absoluto de y_i, apenas da posição relativa entre cada par de vetores (y_m, y_n)
- Uma rotação/translação deste conjunto de pontos irá resultar no mesmo valor para a função *f*.
- Assim, existem vários minimizantes para o problema de optimização e por essa razão o resultado é diferente consoante a inicialização.