Projeto Final

Inteligência Artificial

CK0031/CK0248

Anderson Gomes - 354038 , Karlos Ítalo Pedrosa Bastos - 371666

Parte I - Otimização numérica

1 Introdução

Utilizamos o método de busca linear de Newton, que utiliza a inversa da matriz Hessiana para calcular o vetor direção nas iterações do método. Para isso, utilizamos as bibliotecas numpy e matplotlib da linguagem Python. Como plataforma de desenvolvimento foi utilizado o jupyter notebook.

2 Características da Função

$$f(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 - 4(1 - x_2^2)x_2^2$$

2.1 Curvatura e Contorno

Podemos ver que a função é crescente à medida em que o valor absoluto dos eixos x e y cresce. Nos intervalos [-3, 3] do eixo x e [-2, 2] do eixo y, há algumas ondulações(onde os mínimos locais da função se encontram), mais acentuadas nos pontos próximos a (2, -1), (-2, 1), (0, -0.5) e (0, 0.5). Além disso, podemos observar que os mínimos locais são simétricos.

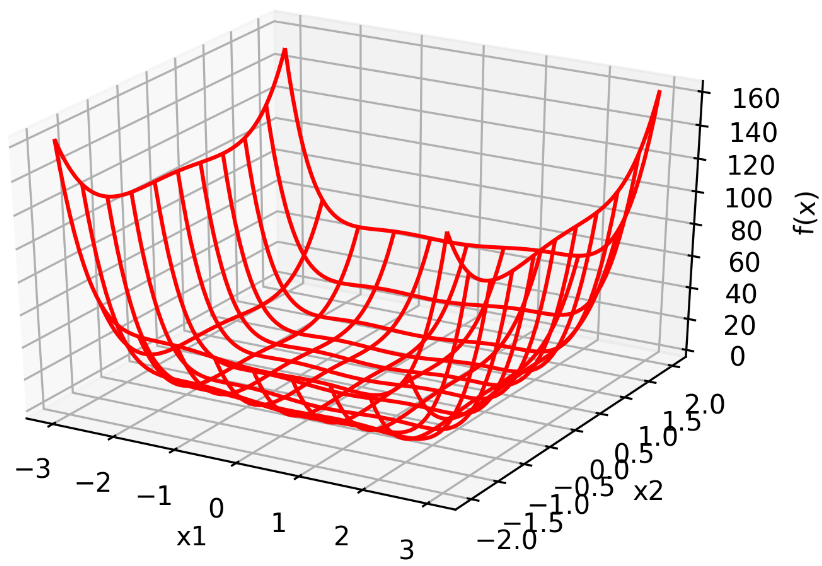


Figure 1: Curvatura de $f(x)$

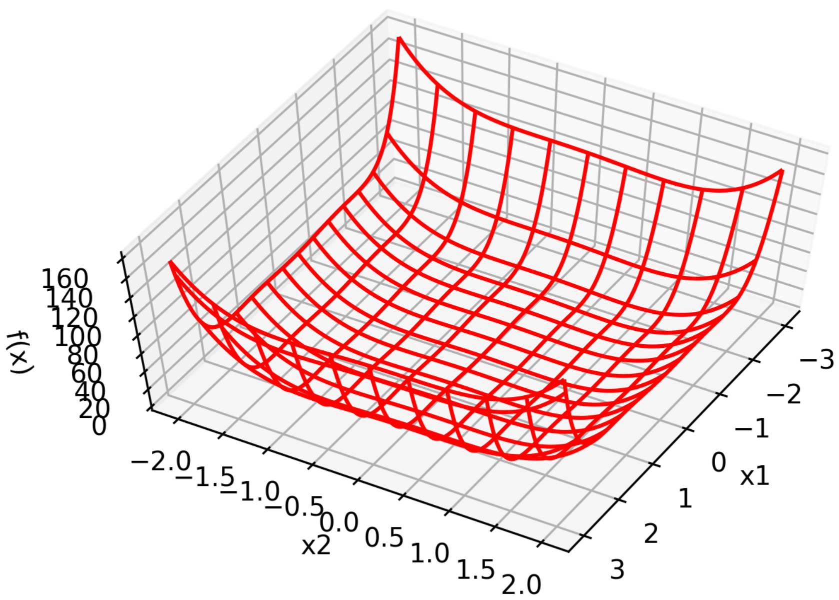


Figure 2: Curvatura de $f(x)$ em outro aspecto de visão

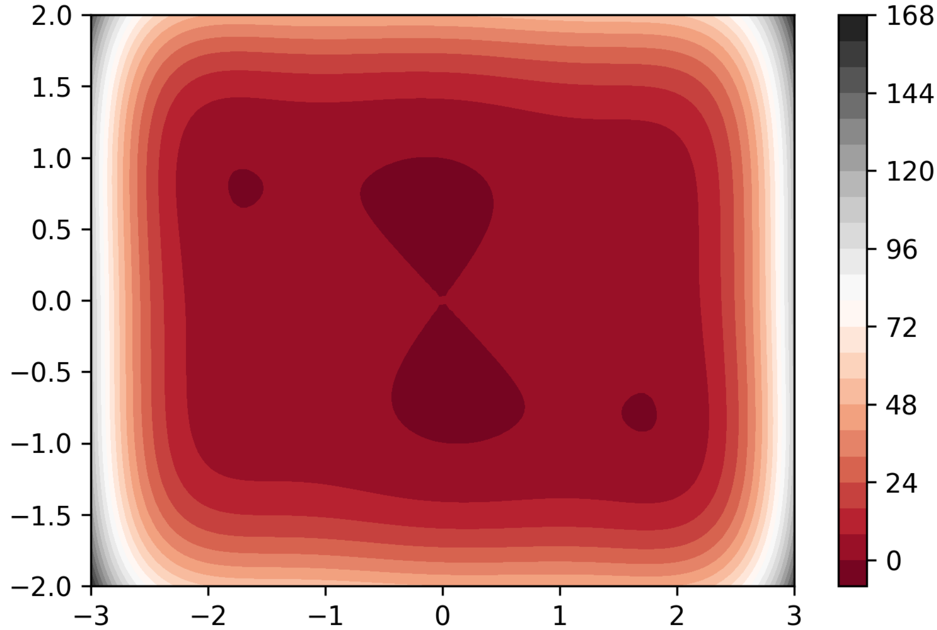


Figure 3: Contorno de $f(x)$

2.2 Vetor Gradiente e Matrix Hessiana

$$\bigtriangledown f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2(x_1^5 - 4.2x_1^3 + 4x_1 + 0.5x_2) \\ x_1 + 16x_2^3 - 8x_2 \end{bmatrix}$$

$$\bigtriangledown^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} \end{bmatrix} = \begin{bmatrix} 10x_1^4 - 25.2x_1^2 + 8 & 1 \\ 1 & 48x_2^2 - 8 \end{bmatrix}$$

3 Implementação

Como critério de parada do algoritmo, utilizamos a diferença entre a norma do vetor atual e a norma do novo vetor calculado na iteração. Se essa diferença for menor que a tolerância passada como argumento para o método, o algoritmo pára. Além desse critério, outro fator que força a parada do algoritmo é o limite de iterações ser ultrapassado.

```
def derivative_x1(x):
    return 2 * (x[0]**5 - 4.2 * x[0]**3 + 4 * x[0] + 0.5 * x[1])
```

```
def derivative_x2(x):
    return x[0] + 16 * x[1]**3 - 8 * x[1]
```

```
def gradientVector(x):
    return np.array([derivative_x1(x), derivative_x2(x)])
```

```
def hessian(x):

    matrix = np.array([[10 * x[0]**4 - 25.2 * x[0]**2 + 8, 1],
                       [1, 48 * x[1]**2 - 8]])

    return matrix
```

```
def inverseHessian(x):

    matrix = hessian(x)

    det = matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]
    invDet = 1 / det

    inverseHessian = invDet * np.array([[matrix[1][1], -1 * matrix[0][1]],
                                        [-1 * matrix[1][0], matrix[0][0]]])

    return inverseHessian
```

Figure 4: Código que computa derivadas parciais, vetor gradiente, hessiana e sua inversa, dados que serão usados no método de Newton.

```python
def direction(x):
    return - inverseHessian(x).dot(gradientVector(x))
```

```python
def norm(x):
    return np.linalg.norm(x)
```

Figure 5: Funções que computam a direção do método de Newton e a norma do vetor que liga a origem a um determinado ponto.

```python
def newton(x, err, iterMax):

    convergenceData = list()
    print('Ponto inicial: {0}'.format(x))
    print('N° máximo de iterações {0}'.format(iterMax))
    print('Tolerância: {0}'.format(err))
    convergenceData.append(x)
    i = 0
    while True:

        xNew = x + 0.1 * direction(x)
        convergenceData.append(xNew)
        #print('Iteração: {0}\nx: {1}\nf(x): {2}'.format(i, xNew, fun(x[0], x[1])))

        if (i > iterMax or abs(norm(x) - norm(xNew)) < err):
            break

        x = xNew
        i = i + 1

    print('Resultado: {0}'.format(xNew))
    print('f(x): {0}'.format(xNew))

    return xNew,np.asarray(convergenceData)
```

Figure 6: Função que executa o método de Newton. Nota: Alpha fixo em 0.1.

4 Exemplos de Execução

4.1 Exemplo 1

Nesse exemplo foi escolhido o ponto (3, -2) como chute inicial para verificar se o método converge para o mínimo local localizado próximo ao ponto (2, -1) da função. A tolerância foi configurada com o valor 10^{-5} e o número máximo de iterações com o valor 500.

Resultado:

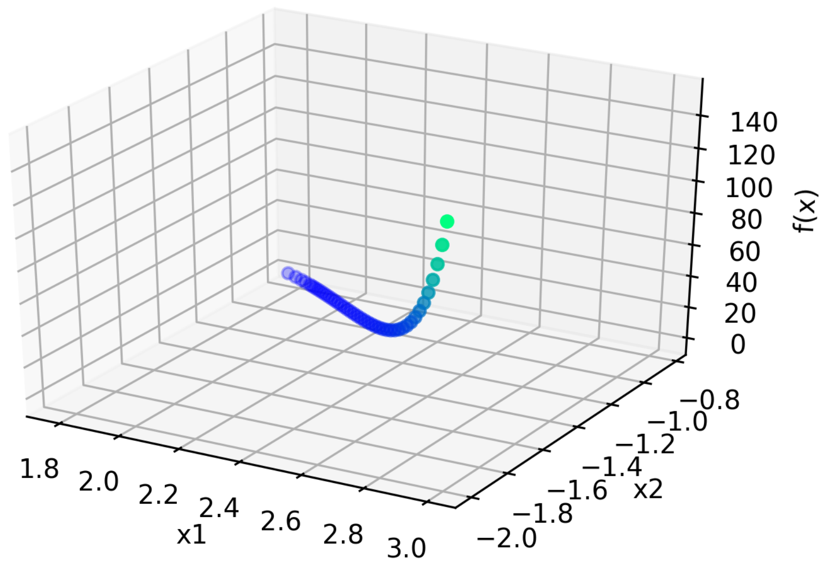


Figure 7: Iterações do processo de minimização a partir do ponto (3,-2). (Iterações iniciais são mais claras).

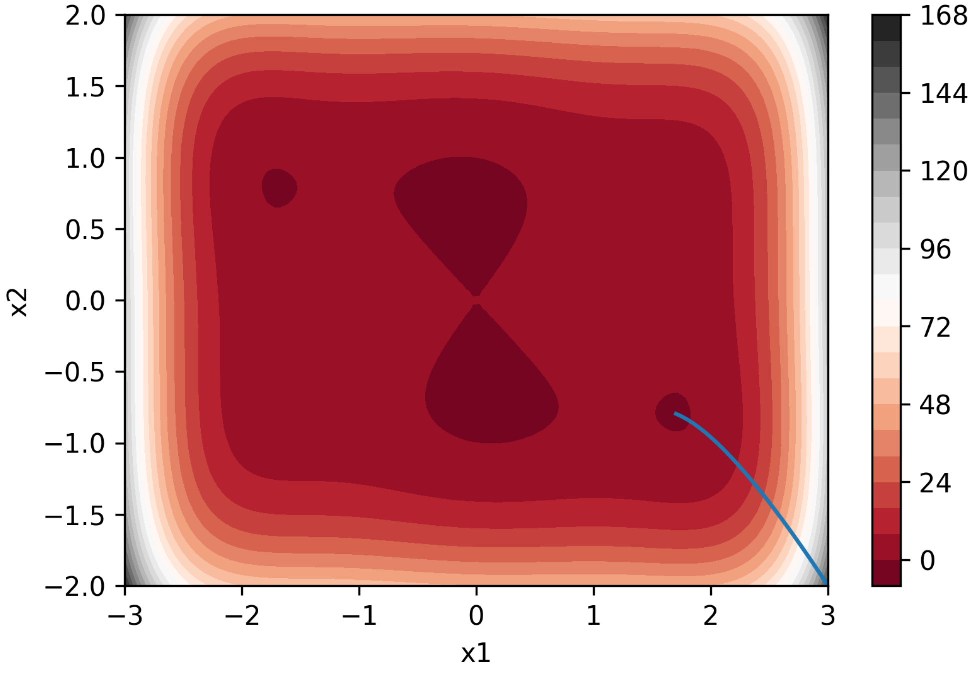


Figure 8: Processo de minimização a partir do ponto (3,-2) sobre o gráfico de contorno da função.

4.2 Exemplo 2

Nesse exemplo foi escolhido o ponto (-3, 2) como chute inicial para verificar se o método converge para o mínimo local localizado próximo ao ponto (-2, 1) da função. A tolerância foi configurada com o valor 10^{-5} e o número máximo de iterações com o valor 500.

Resultado:

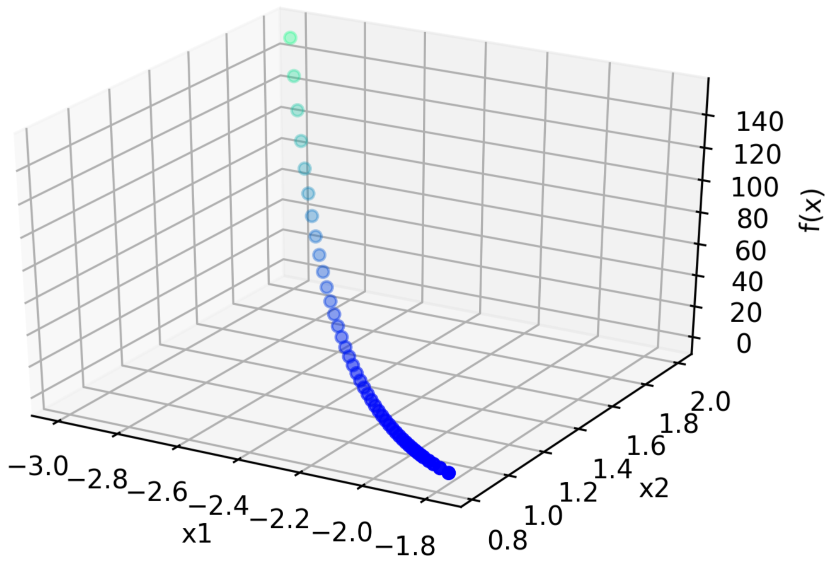


Figure 9: Iterações do processo de minimização a partir do ponto (-3,2). (Iterações iniciais são mais claras).

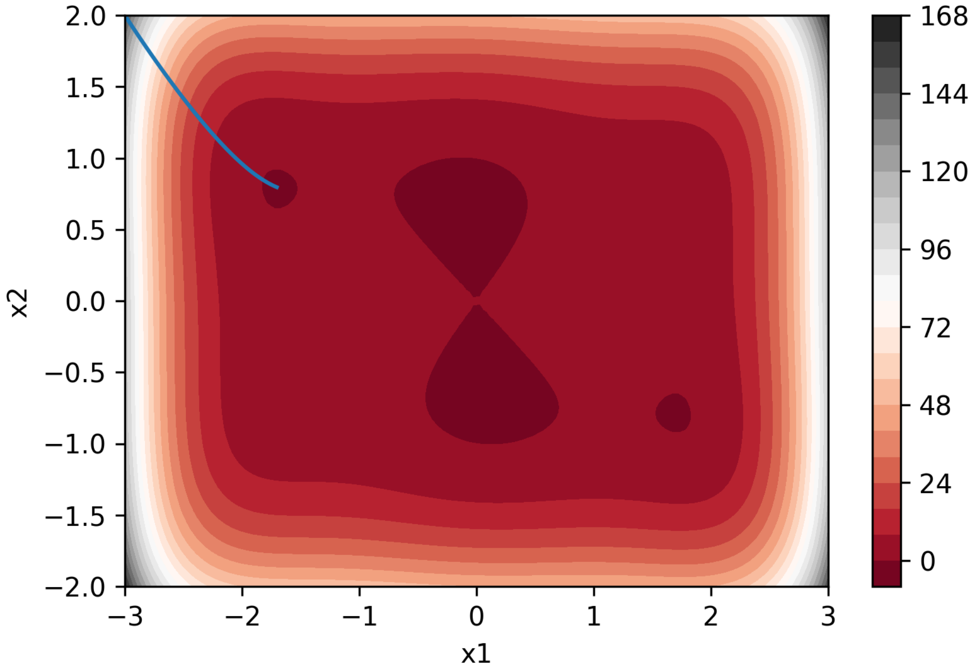


Figure 10: Processo de minimização a partir do ponto (-3,2) sobre o gráfico de contorno da função.

4.3 Exemplo 3

Nesse exemplo foi escolhido o ponto (0, -1.5) como chute inicial para verificar se o método converge para o mínimo local localizado próximo ao ponto (0.1, -0.7) da função. A tolerância foi configurada com o valor 10^{-5} e o número máximo de iterações com o valor 500.

Resultado:

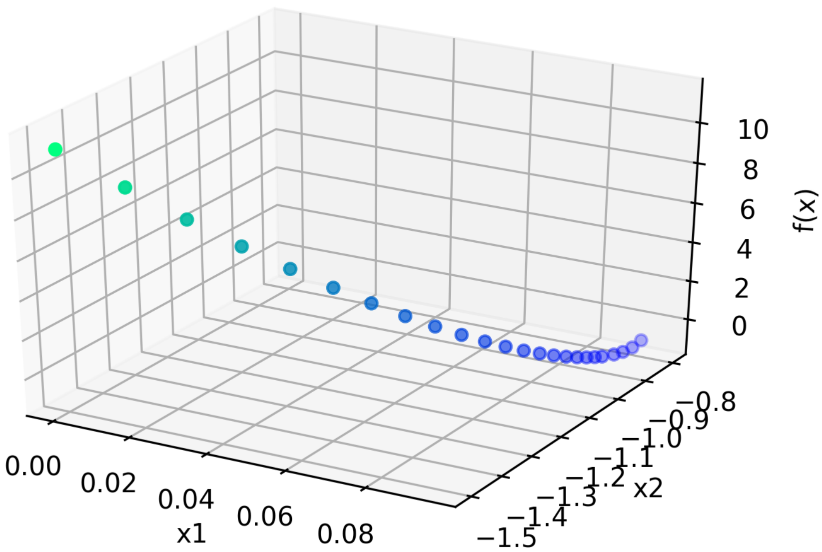


Figure 11: Iterações do processo de minimização a partir do ponto (0,-1.5). (Iterações iniciais são mais claras).

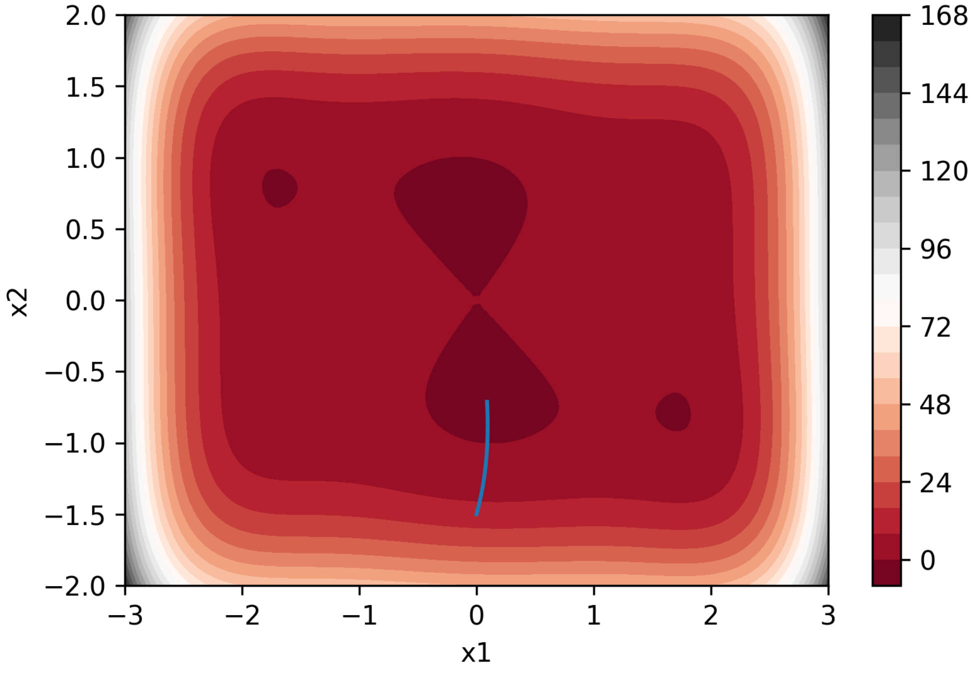


Figure 12: Processo de minimização a partir do ponto (0,-1.5) sobre o gráfico de contorno da função.

4.4 Exemplo 4

Nesse exemplo foi escolhido o ponto (0, 1.5) como chute inicial para verificar se o método converge para o mínimo local localizado próximo ao ponto (-0.1, 0.7) da função. A tolerância foi configurada com o valor 10^{-5} e o número máximo de iterações com o valor 500.

Resultado:

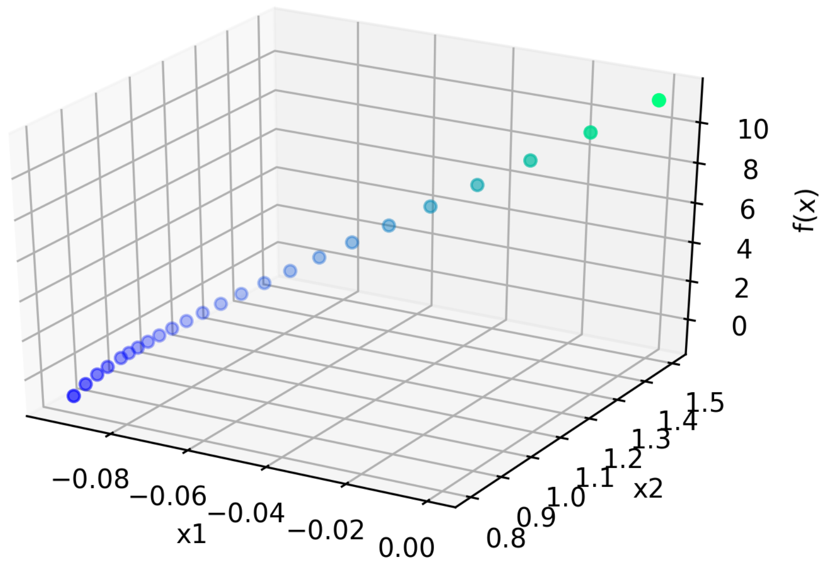


Figure 13: Iterações do processo de minimização a partir do ponto (0,1.5). (Iterações iniciais são mais claras).

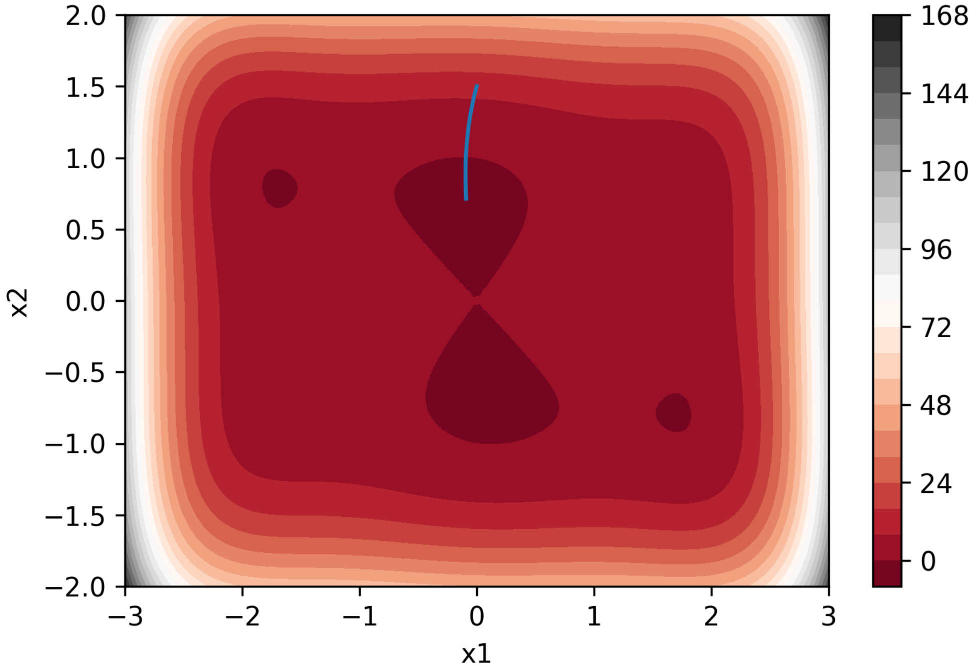


Figure 14: Processo de minimização a partir do ponto (0,1.5) sobre o gráfico de contorno da função.

Parte II - Raciocínio Probabilístico

5 Resolução dos problemas propostos

Problema 1 What is the probability that Sally's house is getting burg larised, given the initial evidence (the neighbour's call)?

Resposta

$$P(B|C) = \frac{P(B,C)}{P(C)}$$

$$P(B,C) = \sum_A \sum_E \sum_R P(B,C,A,E,R)$$

$$= \sum_A \sum_E \sum_R P(B)P(E)P(A|B,E)P(C|A)P(R|E)$$

$$= P(B) \sum_E P(E) \sum_R P(R|E) \sum_A P(A|B,E)P(C|A)$$

$$= 0.001 \times (0.999 \times (1 \times (0.00999 \times 0 + 0.99001 \times 1) + 0 \times (0.00999 \times 0 + 0.99001 \times 1)) +$$
$$0.001 \times (1 \times (0.00999 \times 0 + 0.99001 \times 1) + 0 \times (0.00999 \times 0 + 0.99001 \times 1)))$$

$$= 0.00099001$$

$$P(C) = \sum_B \sum_A \sum_E \sum_R P(B,C,A,E,R)$$

$$= P(B=0)\sum_E P(E)\sum_R P(R|E)\sum_A P(A|B,E)P(C|A)+$$
$$P(B=1)\sum_E P(E)\sum_R P(R|E)\sum_A P(A|B,E)P(C|A)$$

$$\begin{aligned}= 0.999 \times (0.999 \times (1 \times (0.999 \times 0 + 0.001 \times 1)+\\ 0 \times (0.999 \times 0 + 0.001 \times 1))+\\ 0.001 \times (1 \times (0.999 \times 0 + 0.001 \times 1)+\\ 0 \times (0.999 \times 0 + 0.001 \times 1))+\\ 0.00099001\end{aligned}$$

$$= 0.00198901$$

$$P(B|C) = \frac{P(B,C)}{P(C)} = 0.497740081$$

Problema 2 What is the probability that Sally's house is getting burglarised, given the initial and extra evidence (the news on the radio)?

Resposta

$$P(B|C,R) = \frac{P(B,C,R)}{P(C,R)}$$
$$P(B,C,R) = \sum_A\sum_E P(B,C,A,E,R)$$
$$= \sum_A\sum_E P(B)P(E)P(A|B,E)P(C|A)P(R|E)$$
$$= P(B)\sum_E P(E)P(R|E)\sum_A P(A|B,E)P(C|A)$$

$$\begin{aligned}= 0.001 \times (0.999 \times 0 \times (0.00999 \times 0 + 0.99001 \times 1)+\\ 0.001 \times 1 \times (0.0098901 \times 0 + 0.9901099 \times 1))\end{aligned}$$

$$= 0.0000009901099$$

$$P(C,R) = \sum_B\sum_A\sum_E P(B,C,A,E,R)$$

$$= P(B=0)\sum_E P(E)P(R|E)\sum_A P(A|B,E)P(C|A)+$$
$$P(B=1)\sum_E P(E)P(R|E)\sum_A P(A|B,E)P(C|A)$$

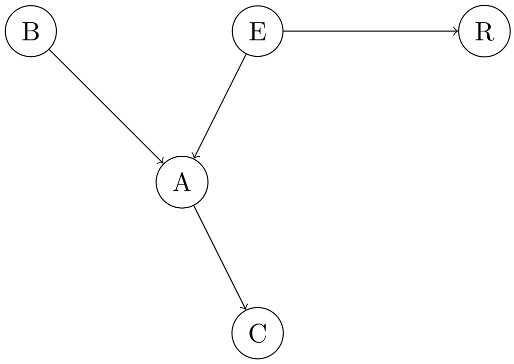
$$\begin{aligned}= 0.999 \times (0.999 \times 0 \times (0.999 \times 0 + 0.001 \times 1)+\\ 0.001 \times 1 \times (0.98901 \times 0 + 0.01099 \times 1))+\\ 0.0000009901099\end{aligned}$$

$$= 0.0000119691199$$

$$P(B|C,R) = \frac{P(B,C,R)}{P(C,R)} = 0.08272203$$

Problema 3 Draw the belief network for the problem.

Resposta



Problema 4 Indicate the Markov blanket of each of the variables.

Resposta

MB(A) = {B, E, C}
MB(B) = {A, E}
MB(C) = {A}
MB(R) = {E}
MB(E) = {A, B, R}