Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Matheus Felipe Ramos Borges

Relatório do Laboratório 3 - Otimização com Métodos de Busca Local

1 Breve Explicação em Alto Nível da Implementação

O objetivo deste experimento é aplicar três algoritmos clássicos de otimização para estimar os parâmetros que melhor descrevem o movimento de uma bola sobre uma superfície rugosa. O modelo físico adotado considera uma desaceleração linear devido ao atrito, representada pela equação:

$$v = v_o - ft \tag{1}$$

A tarefa consiste em determinar os valores ótimos de v_0 (velocidade inicial) e f (coeficiente de desaceleração devido ao atrito) que minimizam o erro entre o modelo teórico e os dados experimentais obtidos a partir do movimento real de uma bola durante uma partida da competição de robótica VSS (Very Small Size Soccer).

Para resolver esse problema de otimização, foram implementados três métodos distintos: Descida do Gradiente, *Hill Climbing* e *Simulated Annealing*, cada um com suas particularidades em termos de exploração do espaço de busca e convergência para mínimos locais ou globais. A ideia dos algoritmos é sempre minimizar a função de custo, que é dada por:

$$J(x) = \frac{1}{2m} \sum_i (f(x_i) - y_i)^2 \tag{2}$$

1.1 Descida do Gradiente

A Descida do Gradiente é um algoritmo iterativo de otimização utilizado para minimizar funções diferenciáveis. A ideia central baseia-se no fato de que o gradiente de uma função em um ponto indica a direção de maior crescimento da função. Assim, ao mover-se no sentido oposto ao gradiente, é possível reduzir progressivamente o valor da função objetivo. No contexto da minimização de uma função de custo $J(\boldsymbol{\theta})$, a atualização dos parâmetros é dada por:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3}$$

A cada iteração, os parâmetros são ajustados de modo a reduzir o valor de $J(\boldsymbol{\theta})$, aproximando-se de um mínimo local (ou global, dependendo da função).

O pseudo-código correspondente ao algoritmo da descida do gradiente pode ser representado da seguinte forma:

```
def gradient_descent(dJ, theta, alpha):
    while not check_stopping_condition():
        theta = theta - alpha * dJ(theta)
    return theta
```

1.2 *Hill Climbing*

O algoritmo *Hill Climbing* é uma técnica de otimização heurística utilizada para problemas de maximização (embora possa ser facilmente adaptado para tarefas de minimização, como no contexto do lab). Sua lógica central consiste em avaliar os estados vizinhos do ponto atual no espaço de busca e mover-se iterativamente para o vizinho que apresenta o melhor valor da função objetivo.

Mais especificamente, a cada iteração, o algoritmo identifica entre os vizinhos o ponto com maior valor da função de custo $J(\boldsymbol{\theta})$ e atualiza o nó atual. No contexto do lab, para cada estado atual $\boldsymbol{\theta}$, são considerados oito vizinhos igualmente espaçados em torno do ponto, simulando uma varredura local no espaço de soluções.

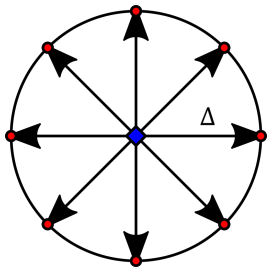

Figura 1: seleção dos vizinhos no algoritmo *Hill Climbing*.

O pseudo-código correspondente ao algoritmo da descida do gradiente pode ser representado da seguinte forma:

```
# Assuming maximization
def hill_climbing(J, theta):
    while not check_stopping_condition():
        best = None  # J(None) is assumed to be -inf
        for neighbor in neighbors(theta):
            if J(neighbor) > J(best):
                best = neighbor
        if J(best) < J(theta):
            return theta
        theta = best
    return theta
```

1.3 *Simulated Annealing*

O algoritmo *Simulated Annealing* é uma técnica de maximização (que pode ser usada também como minimização) que simula o processo de têmpera da metalurgia (aquecimento seguido de resfriamento lento para tratamento de um metal). De forma análoga ao *Hill Climbing*, o *Simulated Annealing* explora os vizinhos do estado atual no espaço de busca. No entanto, seu diferencial está na aceitação probabilística de soluções piores, o que permite escapar de ótimos locais e ampliar a exploração do espaço de soluções.

Mais especificamente, a cada iteração, o custo do vizinho $J(\boldsymbol{\theta})$ é calculado e caso seja maior que o do nó atual, move-se para esse nó, caso contrário, há uma probabilidade de partir para o nó de menor custo que depende da temperatura. A função de temperatura utilizada no lab foi:

$$T = \frac{T_0}{1 + \beta i^2} \tag{4}$$

À medida que as iterações avançam, a temperatura é reduzida, diminuindo a probabilidade de aceitar soluções piores. Isso garante uma transição gradual de uma busca exploratória para uma busca mais refinada e convergente. O pseudo-código correspondente ao algoritmo da descida do gradiente pode ser representado da seguinte forma:

```
# Assuming maximization
def simulated_annealing(J, theta):
    while not check_stopping_condition():
        T = temperature_schedule(i)
        if T < 0.0:
            return theta
        neighbor = random_neighbor(theta)
        deltaE = J(neighbor) - J(theta)
        if deltaE > 0:
            theta = neighbor
        else:
            r = random_uniform(0.0, 1.0)  # Draws random number w/ uniform dist.
            if r <= exp(deltaE / T):
                theta = neighbor
    return theta
```

2 Figuras Comprovando Funcionamento do Código

2.1 Descida do Gradiente

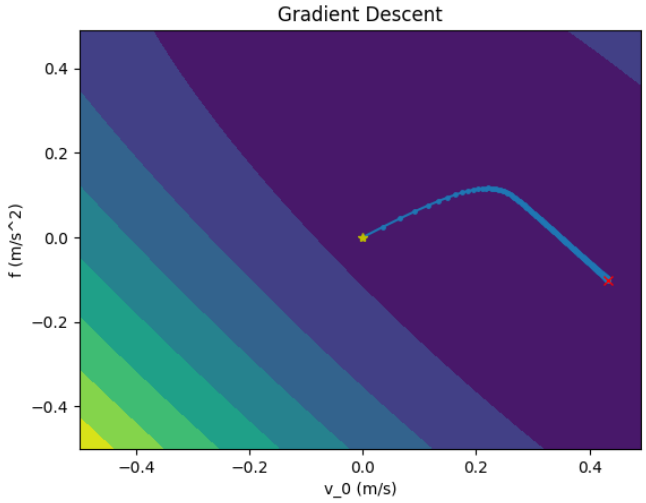

Figura 2: processo de otimização com a Descida do Gradiente

2.2 *Hill Climbing*

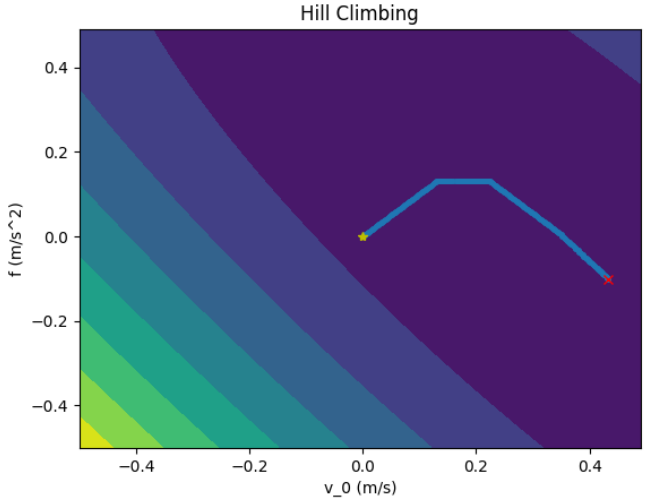

Figura 3: processo de otimização com o *Hill Climbing*

2.3 *Simulated Annealing*

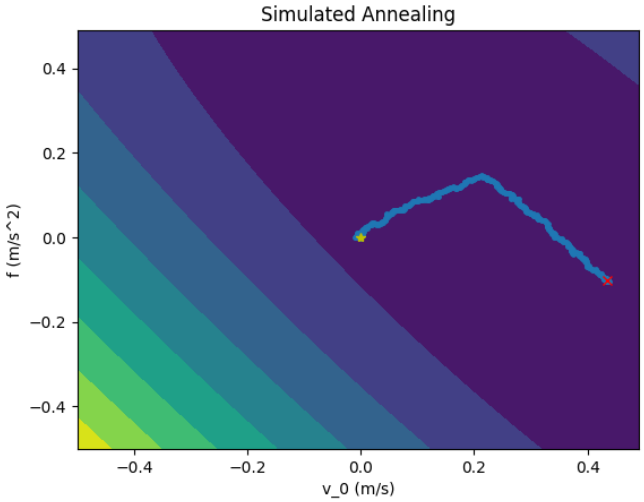

Figura 4: processo de otimização com o *Simulated Annealing*

3 Comparação entre os métodos

Tabela 1 com a comparação dos parâmetros da regressão linear obtidos pelos métodos de otimização.

Tabela 1: parâmetros da regressão linear obtidos pelos métodos de otimização.

Método	v_0	f
MMQ	0.433373	-0.101021
Descida do gradiente	0.433371	-0.101018
Hill climbing	0.433411	-0.101196
Simulated annealing	0.433977	-0.101345

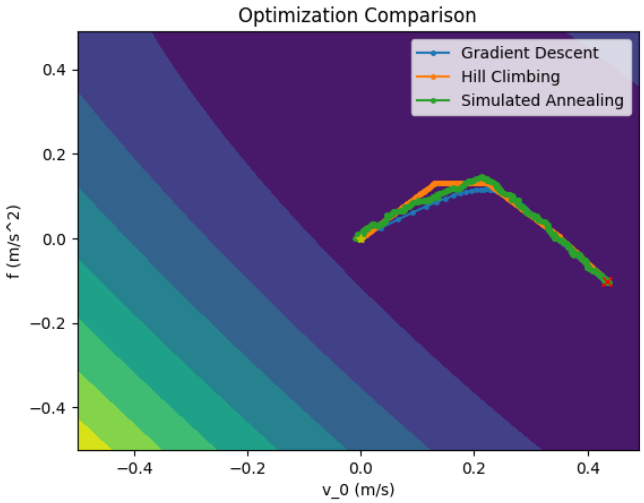

Figura 5: comparação entre os métodos de otimização.

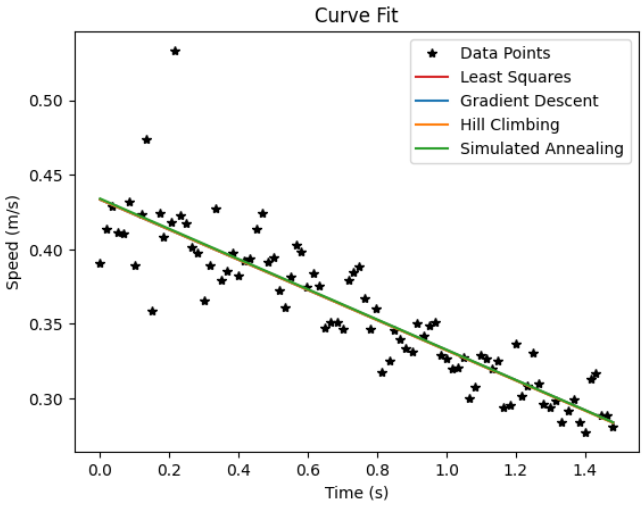

Figura 6: fit da curva com os dados experimentais utilizados.