Instituto Tecnológico de Aeronáutica
CT-213: Inteligência Artificial para Robótica Móvel
Lab 03 - Otimização com Métodos de Busca Local

Bruno Benjamim Bertucci - Turma 23.2

1 Introdução

O processo de determinar os valores máximos ou mínimos de uma função é denominado otimização. A otimização é usada para encontrar parâmetros para alcançar o melhor resultado em uma determinada tarefa, porém, existem funções para as quais não existe uma forma analítica de realizar a otimização. Nesse caso, pode-se utilizar um algorítmo de otimização numérica para obter um resultado aproximado.

Tomando como exemplo o problema de ajustar uma função linear (Equação 1) para se aproximar o máximo possível da tendência de um conjunto de pontos, temos como parâmetros para serem otimizados os valores de θ_0 e θ_1. O par ordenado formado por θ_0 e θ_1 será chamado de θ, de tal forma que $\theta = (\theta_0, \theta_1)$

$$f(x) = \theta_0 + \theta_1 x \tag{1}$$

Essa otimização pode ser feita através da minimização da chamada função de custo, dada pela Equação 2.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_i (f(x_i) - y_i)^2 = \frac{1}{2m} \sum_i (\theta_0 + \theta_1 x_i - y_i)^2 \tag{2}$$

Três exemplos de algorítmos de otimização numérica são: Descida de Gradiente, *Hill Climbing* e *Simulated Annealing*. Nesse relatório, será abordada a implementação dos três algorítmos, bem como o desempenho deles em um problema de ajuste linear usando medidas reais de velocidade de uma bola em deslocamento sujeita a atrito.

2 Descida do Gradiente (*Gradient Descent*)

O algorítmo de Descida do Gradiente consiste em tomar um valor inicial estimado de θ e ajustá-lo iterativamente usando o gradiente da função de custo para o valor de θ atual (θ_i) (Eq. 3 e 4) e um parâmetro chamado taxa de aprendizagem, denotado por α, de tal forma que o novo valor de θ (θ_f) é dado pela Equação 5. Esse equacionamento permite que o ajuste feito a cada iteração fique menor quanto mais próximo θ estiver de um mínimo local da função, o que significa também que esse algorítmo tende a ficar preso em mínimos locais.

$$\frac{\partial J}{\partial \theta_0}(\theta_0, \theta_1) = \frac{1}{m} \sum_i (\theta_0 + \theta_1 x_i - y_i) \tag{3}$$

$$\frac{\partial J}{\partial \theta_1}(\theta_0, \theta_1) = \frac{1}{m} \sum_i (\theta_0 + \theta_1 x_i - y_i) x_i \tag{4}$$

$$\theta_f = \theta_i - \alpha \nabla J(\theta_i) \tag{5}$$

Esse ajuste é repetido até atingir as condições de parada utilizadas, que nesse caso são um número máximo de iterações e a obtenção de uma função de custo suficientemente baixa. O funcionamento do algorítmo de Descida do Gradiente encontra-se ilustrado na Figura 1.

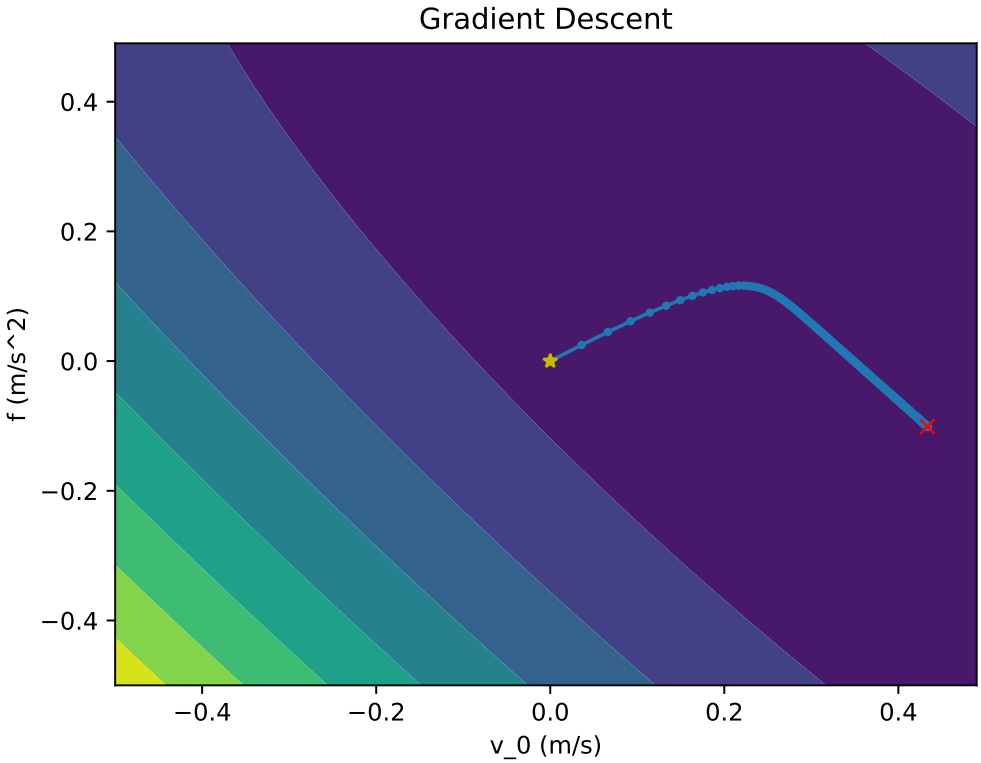

Figura 1: Algorítmo de Descida do Gradiente.

3 *Hill Climbing*

A abordagem do algorítmo *Hill Climbing* para encontrar o mínimo da função de custo é: partir de um θ inicial, e em seguida analisar uma quantidade constante de vizinhos (nessa implementação foram considerados 8 vizinhos), igualmente espaçados a uma distância definida Δ deste θ. O vizinho com a menor função de custo será escolhido para se tornar o novo valor de θ. Esse processo é repetido até que, para algum θ, não sejam encontrados vizinhos com função de custo menor que ele, o que significa que um mínimo local foi atingido, ou até que alguma outra condição de parada seja atingida.

Nessa implementação, as demais condições de parada são um número máximo de iterações e a obtenção de uma função de custo abaixo de um determinado valor. Esse algorítmo, de forma semelhante á Descida do Gradiente, tende a ficar preso em mínimos locais. Um exemplo do funcionamento do algorítmo *Hill Climbing* está ilustrado na Figura 2.

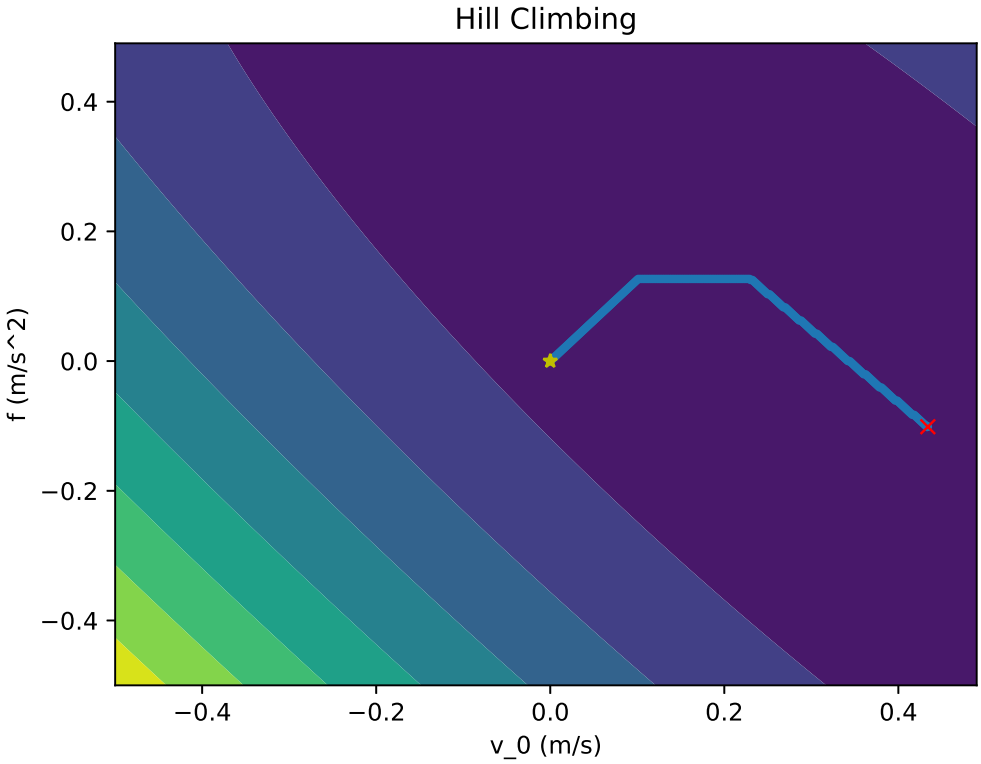

Figura 2: Algorítmo de *Hill Climbing*.

4 *Simulated Annealing*

O algorítmo de *Simulated Annealing* utiliza uma analogia de temperatura e energia. Partindo de um θ inicial, escolhe-se um vizinho dele distante de um valor constante Δ, e com um ângulo aleatório entre $[-\pi, \pi)$.

Define-se a variaçao de energia ΔE como sendo a diferença entre a função de custo do vizinho e do θ. O conceito de temperatura será utilizado para permitir a exploração de vizinhos com funçao de custo maior, na tentativa de escapar de mínimos locais. Essa temperatura deverá ser equacionada de tal forma que, quando maior ela for, maior será a chance de um vizinho com custo maior ser escolhido. Para que essa exploração possa acontecer sem comprometer o resultado final, a temperatura deverá ser maior no início da busca, e diminuir gradativamente, para que no fim da busca a escolha de um vizinho com maior custo seja improvável.

Nessa implementação, a temperatura é representada por uma função que depende do número de iterações já feitas (i), explicitada pela Equação 6, onde T_0 e β são constantes.

$$T(i) = \frac{T_0}{1 + \beta i^2} \tag{6}$$

Sendo assim, para que um vizinho se torna o novo valor de θ, é necessário que seu $\Delta E < 0$, ou que o valor de $e^{\frac{-\Delta E}{T(i)}}$ seja maior ou igual a um número $r \in [0, 1]$, gerado aleatoriamente segundo distribuição uniforme.

Conforme o número de iterações aumenta, a temperatura diminui, portanto, a probabilidade da segunda condição ser atendida também diminui, lembrando que, quando essa condição for considerada, a primeira condição é necessariamente falsa, o que significa que $\Delta E \geq 0$, portanto, $e^{\frac{-\Delta E}{T(i)}} \in (0, 1]$.

Com esse mecanismo, o algorítmo evita de ficar preso em mínimos locais. A iteração desses passos continuará até que as condições de parada consideradas sejam atingidas. Novamente, para essa implementação foram utilizados como condições o número máximo de iterações e um

valor de custo suficientemente baixo. Um exemplo de caminho seguido pelo algorítmo *Simulated Annealing* encontra-se na Figura 3.

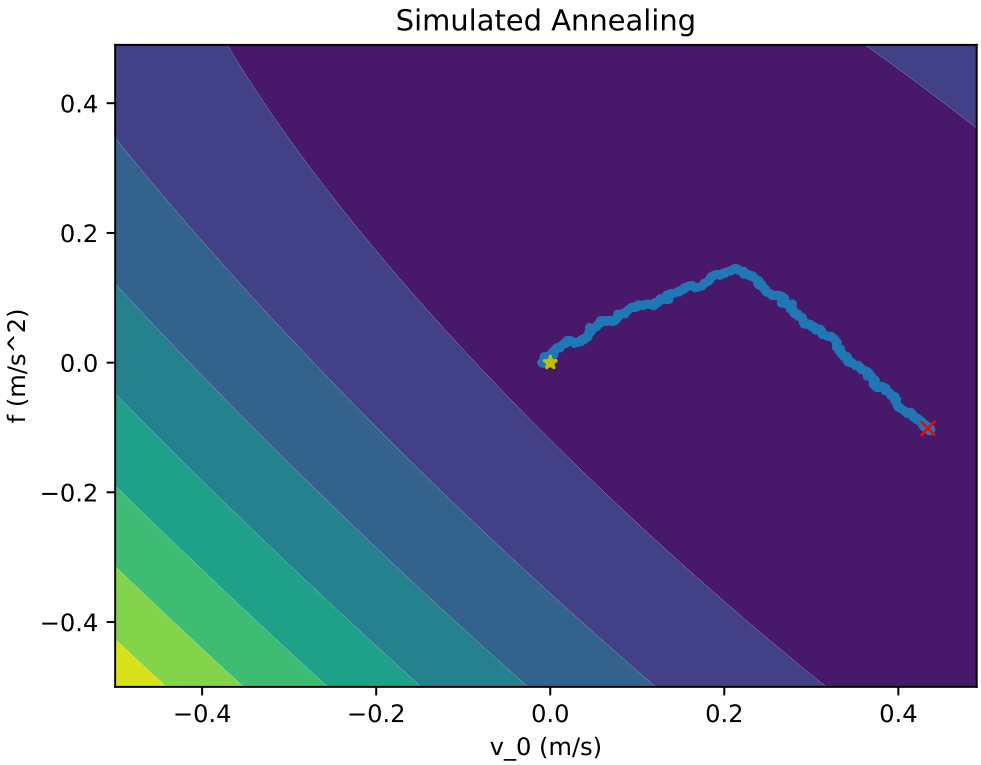

Figura 3: Algorítmo de *Simulated Annealing*.

5 Resultados

Os três algorítmos foram aplicados aos dados de posição (x e y) e tempo medidos para uma bola rolando, tendo iniciado seu movimento a uma velocidade v_0, e sujeita a uma força de arrasto f. A otimização foi realizada, a fim de obter-se os valores otimizados de v_0 e f segundo a Equação 7. Os resultados obtidos pelos algorítmos foram comparados com a solução analítica do problema feita pelo Método dos Mínimos Quadrados (MMQ).

$$v(t) = v_0 + ft \tag{7}$$

Os parâmetros otimizados pelos algorítmos encontram-se na Tabela 1. Para esse problema específico, o algorítmo de Descida do Gradiente fornece os resultados mais próximos da solução por MMQ.

Algorítmo	$v_0(m/s)$	$f(m/s^2)$
MMQ	0,43337277	-0,10102096
Descida do Gradiente	0,43337067	-0,10101846
Hill Climbing	0,43386631	-0,1012519
Simulated Annealing	0,43397656	-0,10134529

Tabela 1: Resultados dos três algorítmos e da solução usando MMQ.