Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT-213
Aluno:

Relatório do Laboratório 3 - Otimização com Métodos de Busca Local

1. Breve Explicação em Alto Nível da Implementação

1.1. Descida do Gradiente

Primeiramente deve-se implementar a função gradiente da função de custo. Para tal , basta aplicar o operador gradiente na função custo, dada como a soma dos quadrados dos erros, sendo x e y caso v e t e os elementos de theta os parâmetros.

Para implementar a descida de gradiente, basta que, para cada função gradiente de theta, que se subtraia de cada elemento do vetor theta um múltiplo alfa de cada elemento do vetor gradiente, atualizando o theta, que será utilizado na próxima iteração. Faz-se isso enquanto o número máximo de iterações não é atingido e enquanto a função custo de de theta é maior que epsilon, pois caso seja menor que epsilon, já estamos satisfeitos com o funcionamento da função.

1.2. Hill Climbing

Primeiramente, deve-se implementar uma função que retorne uma lista dos vizinhos de cada theta, que nesse caso foram escolhidos pela estratégia 8 conectada num raio delta.

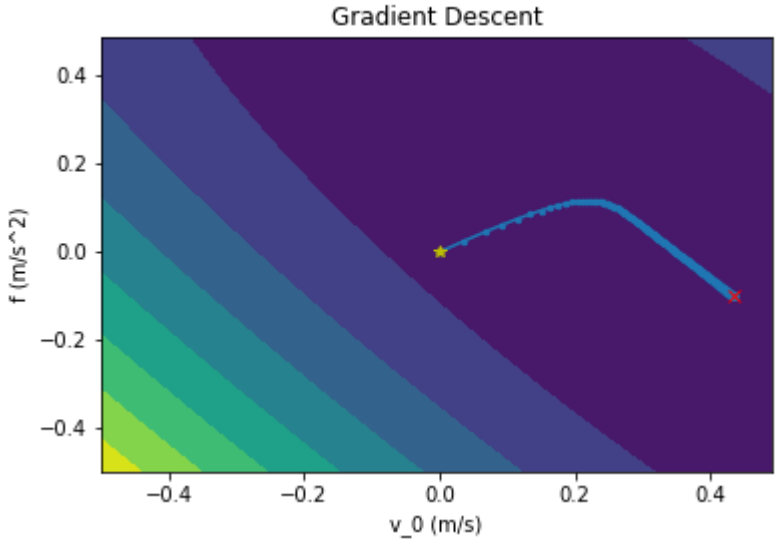
Para implementar a função hill_climbing basta que para cada theta se analise seus vizinhos e se escolha o de menor custo. Caso esse de menor custo tenha custo maior que o custo de theta, retorna-se o theta e finaliza-se o hill_climbing. Caso, porém, o custo seja menor, atualiza-se o theta, que passa a ser seu melhor vizinho. Assim, repete-se o processo até que se o número máximo de iterações seja atingido, ou até quando o custo do theta menor que o valor que o valor do mínimo epsilon, pois o custo já nos está suficientemente ajustado, ou ainda, como já foi dito, até que o custo do melhor vizinho seja maior que o custo de theta.

1.3. Simulated Annealing

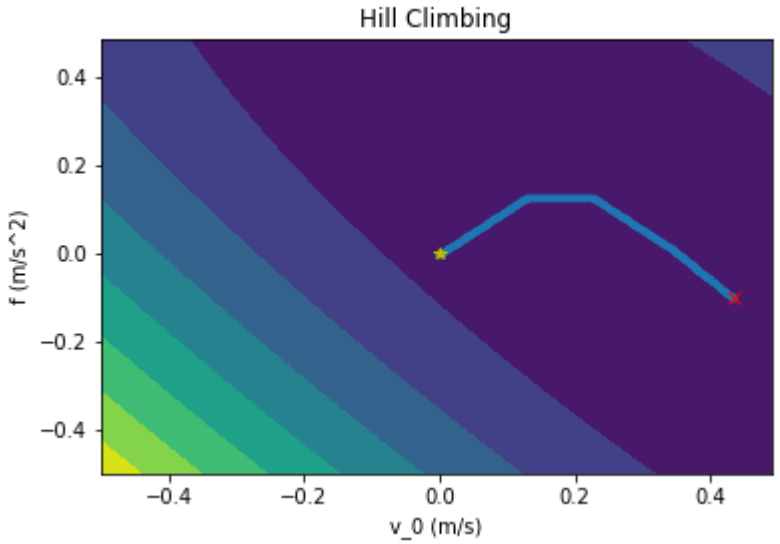
O objetivo do Simulated_Annealing é justamente incentivar a exploração na otimização e evitar mínimos locais, buscando escolher um vizinho aleatoriamente a uma distância delta de theta para que o theta seja atualizado. No nosso caso, buscamos diminuir a função custo, ou seja, queremos esquentar uma placa de temperatura negativa e buscar um vizinho aleatório. Caso o vizinho tenha um custo menor que o theta, atualiza-se o theta para o vizinho. Caso contrário, o normal seria não atualizarmos o theta, mas como queremos exploração e evitar mínimos, damos a chance para theta ser atualizada para o vizinho caso o número aleatório r seja menor ou igual a exp(-deltaE/T), onde deltaE é o custo do vizinho menos o custo do theta. Dessa forma, quanto menor o deltaE, ou seja, quando se está aproximando do custo desejado, mais a exponencial que tem valor entre 0 e 1 fica próxima do 1, fazendo com que seja menos provável que haja a atualização, pois o valor da exponencial fical limitado. Mas se o deltaE for grande, é mais provável que haja a troca. Repete-se esse processo enquanto o número máximo de iterações não for atingido e o custo de theta for maior que o valor mínimo epsilon, onde o custo já suficientemente pequeno.

2. Figuras Comprovando Funcionamento do Código

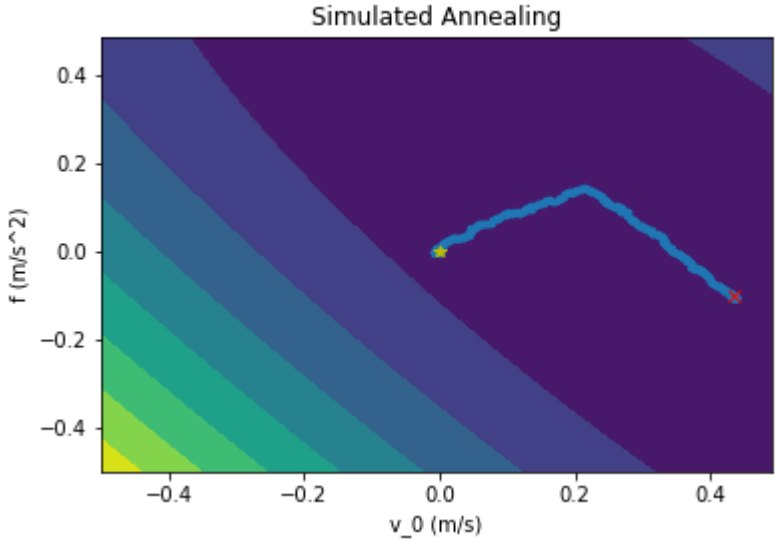
2.1. Descida do Gradiente



2.2. Hill Climbing

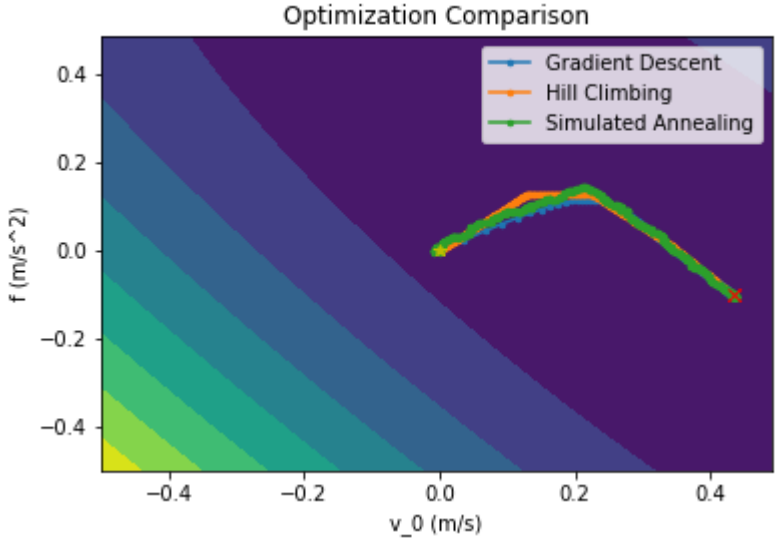


2.3. Simulated Annealing



3. Comparação entre os Métodos

Basta preencher a tabela e colocar as figuras da trajetória de otimização e das curvas da regressão linear.



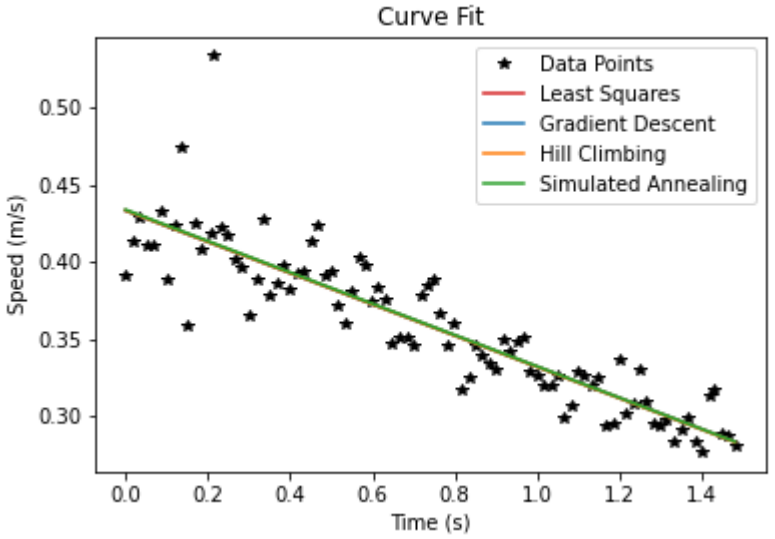


Tabela 1 com a comparação dos parâmetros da regressão linear obtidos pelos métodos de otimização.

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

	v0	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