

Programa para casa #6

Otimização multidimensional sem restrições
Aplicação de métodos diretos e métodos gradientes

5 de junho de 2025

1 Contextualização

Nessa seção adentraremos no universo de métodos numéricos voltados a encontrar pontos de mínimo ou máximo em funções contínuas de uma ou mais variáveis. Nesse contexto, assim como no caso da busca pelo zero de uma função também estaremos lidando com a busca de pontos específicos em um gráfico ou curva multidimensional. Entretanto, os pontos que buscamos estarão associados a diferentes características. A figura (1) ilustra bem essas diferenças. Note por exemplo que se tratamos o problema de determinação do zero não de uma função, mas de sua derivada, utilizando por exemplo um método de Newton-Raphson para o caso 1D, ainda assim não saberíamos dizer se esse ponto encontrado é um ponto de mínimo ou de máximo, a não ser que avaliemos por exemplo o sinal da segunda derivada. Essa é só uma das diversas particularidades que a solução numérica de problemas de otimização traz em relação ao caso mais simples do estudo de métodos numéricos voltados ao contexto de busca de raízes de equações.

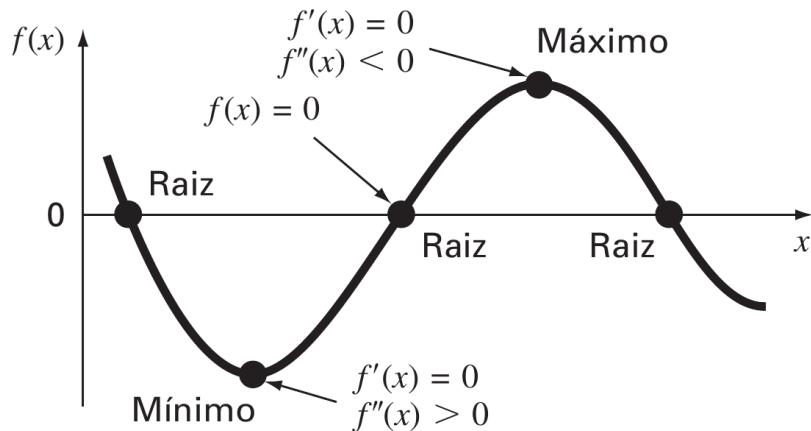


Figura 1

De toda sorte, o conhecimento do valor da derivada (para contextos unidimensionais) ou do gradiente (para cenários multidimensionais) da função que desejamos otimizar será importante em nosso estudo. Em contextos nos quais essa derivada ou gradiente não se encontra disponível analiticamente, teremos também que apelar para esquemas numéricos de avaliação dessa informação utilizando por exemplo aproximações por diferenças finitas. O mesmo racional aplicado para transformarmos o método de Newton-Raphson no método da secante se aplica aqui.

2 Perspectiva histórica

Antes de entrarmos nos processos de elaboração e construção de diferentes esquemas numéricos destinados à finalidade de encontrar esses pontos ótimos em diferentes tipos de curvas uni e multi-dimensionais, precisamos falar um pouco mais sobre aspectos históricos e trazer algumas definições iniciais. Essas definições serão muito importante para entendermos como se formula um problema de otimização antes de sairmos resolvendo qualquer coisa.

Do ponto de vista histórico, o estudo de otimização bebe de uma fonte maior que no campo da matemática é conhecida como cálculo variacional, cujos fundamentos foram assentados por nomes como Bernoulli, Euler e Lagrange. Enquanto nos problemas de otimização geralmente desejamos encontrar pontos dentro de uma curva que maximizam ou minimizam os valores da própria curva (função), no cálculo variacional geralmente queremos encontrar funções que minimizem um funcional. Em certo sentido podemos afirmar que o cálculo variacional leva a ideia de otimização para contextos mais gerais e extremos. Acontece que isso acaba sendo muito útil na física, pois a natureza acaba se organizando da melhor maneira que poderia se organizar. Quando pensamos em leis da física, estamos pensando em relações entre causa e efeito escritas em linguagem matemática em termos de funcionais, que são funções de funções. Nesse sentido, o cálculo variacional acaba sendo uma excelente ferramenta para a descrição de muitas leis da física.

O marco inicial do cálculo variacional é o trabalho de Bernoulli relacionado ao problema da braquistôcrona, nome dado a uma curva específica que faz com que uma esfera percorrendo-a sob ação da gravidade chegue no ponto B a partir de uma ponto A pelo menor tempo possível. Apesar de não ser o caminho mais curto entre dois pontos, acaba que essa curva fornece o caminho mais rápido quando a gravidade induz o movimento. Esse problema é considerado por muitos como uma espécie de ponto fundador do cálculo variacional. Em seguida, Euler sistematiza matematicamente o problema variacional de minimização de integrais e finalmente Lagrange consolida o cálculo variacional como uma poderosa ferramenta matemática para enunciação de princípios físicos por meio do meu método de multiplicadores de Lagrange, utilizado em problemas de otimização com restrições, como veremos mais para frente. Nos seus estudos, Lagrange simplesmente reformula toda a mecânica clássica, cujas bases haviam sido todas lançadas por Newton, sob o novo ponto de vista do princípio variacional da ação mínima, criando o que nós chamamos de mecânica Lagrangiana.

Apesar do cálculo variacional estar relacionado com a essência de problemas de otimização, os problemas geralmente abordados por cada um são diferentes. Como dissemos aqui, enquanto o cálculo variacional visa encontrar funções que otimizem funcionais, a otimização visa encontrar pontos que otimizem funções. Nesse sentido, a otimização acaba sofrendo uma revolução muito maior com o advento dos computadores do que o próprio cálculo variacional. Com a possibilidade de execução de rotinas de cálculo pré-programadas nessas novas máquinas, a área de otimização avança de forma substancial após a segunda guerra mundial.

É interessante notar que o estímulo para o uso de computadores visando resolver problemas de otimização também tem suas raízes históricas. O grande computador ENIAC foi projetado e construído entre 1943 e 1945, tendo sido lançado só após o fim da segunda guerra em 1946. Nessa mesma época uma demanda das grandes nações era justamente a determinação da distribuição de custo mínimo de suprimentos e produtos, que justifica os trabalhos de Koopsman na Inglaterra e Kantorovich na antiga União Soviética, ambos

relacionados à formalização e aplicação problemas de otimização linear com restrições à alocação ótima de recursos na economia. Em 1947, um aluno de Doutorado de Koopsman inventa o procedimento SIMPLEX, um algoritmo robusto para a solução de problemas de programação linear, que nós veremos em detalhe nesse capítulo.

3 Motivação inicial: otimização e aprendizado de máquina

Mais recentemente, motivado pelo avanço de algoritmos de aprendizado de máquina baseados em modelos de linguagem, temos visto cada vez mais pessoas falando sobre o tema. Entretanto, é comum ainda um certo ar de misticismo em torno do princípio central de funcionamento por trás da lógica de um ChatGPT, DeepSeek e afins.

O que podemos dizer é que quando você usa um algoritmo de IA baseado em modelos de linguagem você está usando sem saber algoritmos de otimização. Todo modelo de inteligência artificial precisa ser treinado para poder dar uma resposta a uma pergunta ou demanda que chega. Esse treinamento é baseado em dados que alimentam o modelo. Na fase de treinamento o que está acontecendo ali dentro é um processo de minimização de uma função, que em inglês é chamada de *loss function*. Essa função mede o quanto errado o modelo está ao prever a próxima palavra em uma frase. Se ele recebe a informação de que o céu é azul, ele pode se auto-treinar tentando adivinhar cada próxima palavra da frase. É como aquela conversa interna que temos na nossa cabeça. Você às vezes recebe uma informação e sua cabeça fica falando meio que por cima do que está chegando para você. É como se a fase de treinamento de um modelo desses fosse uma voz ativada ao mesmo tempo em que a frase está chegando de fora (dados) e tentando adivinhar a próxima palavra. Se a sua cabeça conclui errado que o céu é cinza, logo em seguida os dados confirmam que era azul e aí você aumenta o valor do erro que quer minimizar. Com esse feedback, alguns parâmetros do seu cérebro artificial devem ser ajustados. Esse cérebro artificial no caso de um modelo de linguagem pode ser um algoritmo de redes neurais, que utiliza funções com pesos ajustáveis para zerar resíduos associados à diferenças entre entradas e saídas.

Para a coisa não ficar muito abstrata vamos a um exemplo simples que nos ajudará a compreender muitos conceitos importantes. Esse exemplo visa abrir um pouco a caixa-preta dos algoritmos de inteligência artificial baseados em modelos de linguagem para que você entenda alguns conceitos centrais que habitam o coração desses algoritmos.

Suponha que queiramos construir um algoritmo capaz de prever a próxima palavra provável em uma lista de palavras a partir de uma palavra de entrada. No nosso caso a lista de palavras de entrada é [física, psicologia, poesia] e a lista de opções de palavras de saída é [arte, humanidades, ciência]. Suponha que cada lista possa ser representada em termos de um vetor, de tal sorte que se a palavra de entrada for física, podemos definir um vetor de entrada com valores $[1, 0, 0]$. Sabemos que física é uma ciência, logo o valor esperado de saída é $[0, 0, 1]$ para o nosso vetor de saída. Da mesma maneira, se a entrada é psicologia, nosso vetor de entrada seria $[0, 1, 0]$ e nossa saída seria humanidades, ou em termos vetoriais $[0, 1, 0]$.

A ideia de um algoritmo de IA baseado em modelo de linguagem é partir de um vetor de entrada e de um vetor de saída dado (valor esperado), criar um vetor de saída previsto de mesma dimensão do valor de saída dado, inicialmente povoado com números randômicos por meio de uma matriz de pesos e a partir do vetor de saída dado ir ajustando os pesos

da matriz para calibrar o modelo de tal sorte a prever um valor de saída próximo do valor dado na etapa de treinamento com base num processo de minimização de um erro.

Um algoritmo simples para realizar essa tarefa usando um script Python é apresentado a seguir. Aqui destrincharemos todos os detalhes desse algoritmo. As primeiras linhas são destinadas à importação da biblioteca numpy e criação dos vetores 1x3 referentes aos dados de entrada e saída esperada. Definir o vetor de entrada como $[1, 0, 0]$ e um vetor de saída como $[0, 0, 1]$ no nosso contexto equivale a ensinar para o algoritmo que física é uma ciência. O próximo passo consiste na criação de uma matriz 3×3 contendo pesos que serão ajustados posteriormente na etapa de calibração (ou tunagem da nossa caixinha de processamento). Essa matriz $[W]$ é criada inicialmente de maneira randômica. A próxima etapa consiste na criação de uma função chamada *softmax*. E esse é o primeiro passo interessante do nosso algoritmo em termos conceituais. O objetivo dessa função aqui é transformar um vetor qualquer de pontuações x chamado no jargão de *logits* em uma distribuição de probabilidade. Esse vetor de pontuações é um vetor no nosso caso 1×3 que contém informações relacionadas ao produto matricial entre a matriz $[W]$ e o vetor de entrada. Essas informações serão usadas para construção de um erro que deverá ser minimizado. De toda sorte, o objetivo da função *softmax* é transformar os *logits* em uma função distribuição de probabilidade.

```

1 import numpy as np
2
3 # Vetor de entrada
4 entrada = np.array([1, 0, 0]) # fisica, psicologia, poesia
5 saida_esperada = np.array([0, 0, 1]) # arte, humanidades, ciencia
6
7 # Matriz de pesos (3x3) - inicialmente randomica
8 pesos = np.random.rand(3, 3)
9
10 # Funcao softmax
11 def softmax(x):
12     e_x = np.exp(x - np.max(x))
13     return e_x / e_x.sum()
14
15 # Funcao de perda (entropia cruzada)
16 def cross_entropy(pred, real):
17     return -np.sum(real * np.log(pred + 1e-10))
18
19 # Taxa de aprendizado
20 alpha = 0.1
21
22 # Treinamento
23 for epoca in range(100):
24     logits = np.dot(entrada, pesos)
25     saida_predita = softmax(logits)
26     perda = cross_entropy(saida_predita, saida_esperada)
27     gradiente = np.outer(entrada, saida_predita - saida_esperada)
28     pesos -= alpha * gradiente
29     if epoca % 10 == 0:
30         print(f"Epoca {epoca}, Perda: {perda:.4f}")
31
32 print("Saida final predita:", softmax(np.dot(entrada, pesos)))

```

Listing 1: Código simplificado para treinamento de modelo linear com softmax e entropia cruzada

Aqui a função *softmax* no algoritmo é definida como uma função que não altera a ordem matricial de suas entradas, ou seja, se recebe um vetor de dimensão 1×3 retorna na saída um vetor diferente com a mesma dimensão. Se nosso vetor de *logits* é um vetor $z \in \mathbb{R}^m$, a função *softmax* transforma esse vetor z em um vetor $y \in \mathbb{R}^m$, dado por

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^m e^{z_j}}, \quad \text{para } i = 1, 2, \dots, m. \quad (1)$$

O que essa função faz é transformar todos os números z dos logits em números positivos que variam entre $[0, 1]$ por meio do uso da função exponencial. Isso é feito aqui para conferir uma característica de função distribuição de probabilidade a esses dados. Considere o nosso caso em que o vetor de entrada é dado por $x = [1, 0, 0]$ e uma matriz de pesos W tal que $z = xW$ gera os logits $z = [2.0, 1.0, 0.1]$. Aplicando a função softmax sobre os logits temos como saída:

$$\hat{y}_1 = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} \approx \frac{7.39}{7.39 + 2.71 + 1.11} \approx 0.659 \quad (2)$$

$$\hat{y}_2 = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} \approx \frac{2.72}{7.39 + 2.71 + 1.11} \approx 0.242 \quad (3)$$

$$\hat{y}_3 = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} \approx \frac{1.11}{7.39 + 2.71 + 1.11} \approx 0.099 \quad (4)$$

De tal sorte que o resultado dessa operação para o nosso caso seria o vetor de saída $\hat{y} \approx [0.659, 0.242, 0.099]$. Note que a função *softmax* pega valores de entrada não necessariamente limitados ao intervalo $[0, 1]$ e por meio do uso da função exponencial de cada termo, dividida pela soma das exponenciais dos termos restantes transforma cada termo em um termo limitado no intervalo $[0, 1]$. Além disso, se você leu o script em Python com atenção, deve ter notado a aplicação não apenas da exponencial sobre cada termo, mas de uma função do tipo $\exp[x - \max(x)]$. Essa subtração do valor máximo de x é uma artimanha numérica para evitar que valores muito grandes de x estourem a saída da função. Como a função *softmax* retorna a divisão dessa exponencial pela soma de outras exponenciais também subtraídas do valor máximo de x , isso não afeta o resultado final e resolve o problema numérico. Talvez fique mais fácil de ver com a equação (5).

$$\frac{e^{x_1 - x_{\max}}}{e^{x_1 - x_{\max}} + e^{x_2 - x_{\max}} + e^{x_3 - x_{\max}}} = \frac{e^{-x_{\max}} e^{x_1}}{e^{-x_{\max}} (e^{x_1} + e^{x_2} + e^{x_3})} = \frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}} \quad (5)$$

O próximo passo do nosso algoritmo é a construção de uma função responsável por calcular uma propriedade chamada de *entropia cruzada*. Essa quantidade é uma métrica que será utilizada no nosso algoritmo para computar o quanto boa ou ruim é a predição do modelo. Aqui a entropia cruzada é a função de perda (*loss function*) que deverá ser minimizada no processo de treinamento. Essa propriedade mede a distância entre a saída prevista \hat{y} e a saída esperada y , que no nosso caso é dada por :

$$\mathcal{L}(\hat{y}, y) = - \sum_{i=1}^m y_i \log(\hat{y}_i). \quad (6)$$

Ainda no nosso exemplo, temos que a saída esperada é $[0, 0, 1]$. Se nosso vetor referente à saída esperada é y , então temos $y_1 = 0$, $y_2 = 0$ e $y_3 = 1$, de tal sorte que

$$\mathcal{L} = -\log(\hat{y}_3) = -\log(0.099) \approx 2.31. \quad (7)$$

Note que quando menor a probabilidade atribuída à classe correta, maior a penalização da função perda. No exemplo anterior, a aplicação da função *softmax* ao vetor de *logits* gerou como resultado um vetor de probabilidades $\hat{y} \approx [0.659, 0.242, 0.099]$. A baixa probabilidade ao termo \hat{y}_3 penaliza-os perante os demais, gerando um valor mais alto da entropia cruzada. Note também que no script Python fornecido considera-se dentro da função log a soma de um fator 10^{-10} para evitar a singularidade provocada pelo logaritmo de uma entrada nula.

Finalmente, precisamos destrinchar o laço do nosso algoritmo. É ali que de fato iremos utilizar um algoritmo de minimização para ajustar os pesos da matriz $[W]$ a fim de minimizar a entropia cruzada entre a saída prevista \hat{y} e a saída esperada y . Mas antes de adentrarmos em como isso é feito, segue abaixo uma lista que resume o significado das variáveis utilizadas nesse exemplo.

- $x \in \mathbb{R}^n$: vetor de entrada
- $W \in \mathbb{R}^{n \times m}$: matriz de pesos
- $z = W^T x \in \mathbb{R}^m$: logits
- $\hat{y} = \text{softmax}(z) \in \mathbb{R}^m$: saída prevista
- $y \in \mathbb{R}^m$: vetor contendo a classe correta (saída esperada)
- $\mathcal{L} = -\sum_{i=1}^m y_i \log(\hat{y}_i)$: entropia cruzada

Chegamos agora na parte do nosso script no qual adentramos um laço. Esse laço faz a variável *epoca* variar incrementalmente de 1 em 1 enquanto chama sequencialmente as funções definidas anteriormente numa ordem específica. A primeira chamada diz respeito ao cálculo dos *logits*, dado pelo produto escalar entre o vetor de entrada e a matriz peso. Esse cálculo é feito aqui pela biblioteca numpy por meio da função implícita `np.dot(entrada, pesos)`. Em seguida, esses *logits* são transformados num vetor distribuição de probabilidade pela função *softmax*. Essa transformação é armazenada na saída predita pelo modelo. É importante mencionar que na primeira passagem pelo laço, os valores previstos ainda estarão longe dos valores de referência, uma vez que a matriz peso é inicializada com números randômicos. A ideia dos próximos passos dentro desse laço é justamente usar a entropia cruzada entre a saída predita pelo modelo e a saída esperada para ajustar a matriz peso a fim de diminuir as diferenças entre essas duas informações. Dessa forma, o passo seguinte consiste em calcular essa entropia cruzada e atribuí-la à variável perda.

O passo subsequente do laço é o coração de onde queremos chegar com essa motivação para o estudo de métodos de otimização. A ideia aqui agora é calcular o gradiente da entropia cruzada com relação à matriz peso. Esse gradiente é o equivalente ao nosso conceito mais simples de derivada, porém num espaço multidimensional, uma vez que cada um dos 9 termos da matriz peso possuirá impacto na entropia cruzada. A ideia aqui de calcular esse gradiente é a de justamente caminhar em seu sentido contrário a fim de encontrar valores dos coeficientes da matriz peso que minimizem a entropia cruzada. Lembre-se do cálculo de várias variáveis que se temos uma função escalar de duas variáveis $f(x, y)$, podemos plotar essa função num plano xy para obter uma curva tridimensional que representa o comportamento da função. O cálculo do gradiente de $f \rightarrow \nabla f$ gera um vetor que aponta no sentido de crescimento da função. Aqui a ideia é caminharmos no sentido inverso a fim de encontrarmos valores da matriz peso W que minimizem a nossa *loss function*.

Como o cálculo do gradiente é um ponto crítico do algoritmo, vamos olhar para ele com atenção. No nosso contexto o que queremos calcular é

$$\nabla_W \mathcal{L} = \frac{\partial \mathcal{L}}{\partial W_{ij}} = \frac{\partial \mathcal{L}}{\partial z_k} \frac{\partial z_k}{\partial W_{ij}}. \quad (8)$$

Na equação (8) estamos usando a regra da cadeia. O interessante é que o uso da função *softmax* aplicada ao logits z gerando como saída \hat{y} leva como consequência uma expressão final razoavelmente simples para a determinação desse gradiente que estamos querendo calcular. Mas vamos fazer isso passo a passo. Note que

$$\frac{\partial \mathcal{L}}{\partial z_k} = \frac{\partial \mathcal{L}}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_k}, \quad (9)$$

mas

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i} \left[- \sum_i y_i \log(\hat{y}_i) \right] = -y_i \sum_i \frac{1}{\hat{y}_i} \quad (10)$$

e

$$\frac{\partial \hat{y}_i}{\partial z_k} = \frac{\partial}{\partial z_k} \left(\frac{e^{z_i}}{\sum_j e^{z_j}} \right) = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & \text{se } i = k \\ -\hat{y}_i \hat{y}_k & \text{se } i \neq k. \end{cases} \quad (11)$$

Substituindo (10) em (9), obtém-se

$$\frac{\partial \mathcal{L}}{\partial z_k} = - \sum_i \frac{y_i}{\hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_k}. \quad (12)$$

Para a substituição de (11) em (12) precisamos avaliar os dois casos ($i = k$ e $i \neq k$). Primeiramente, para $i = k$, temos

$$\frac{\partial \mathcal{L}}{\partial z_k} = - \frac{y_k}{\hat{y}_k} \hat{y}_k (1 - \hat{y}_k) = -y_k (1 - \hat{y}_k), \quad (13)$$

já para $i \neq k$, temos

$$\frac{\partial \mathcal{L}}{\partial z_k} = \sum_{i \neq k} \frac{y_i}{\hat{y}_i} \hat{y}_i \hat{y}_k = \sum_{i \neq k} y_i \hat{y}_k \quad (14)$$

No nosso caso, nossos vetores de saída possuem apenas um termo não nulo com valor igual a 1 e os outros termos nulos. Esses vetores são chamados na terminologia dos algoritmos de aprendizado baseado em modelos de linguagem de vetores *one-hot*. Como y é one-hot, essa derivada de \mathcal{L} com relação aos logits é simplificada. Na equação (13) para o caso de y ser um vetor *one-hot* temos simplesmente $y_k = 1$ e da equação (14) o somatório em y_i dá $1 + 0 + 0 = 1$. Em outras palavras

- se $k = p$: $\frac{\partial \mathcal{L}}{\partial z_k} = \hat{y}_k - 1$;
- se $k \neq p$: $\frac{\partial \mathcal{L}}{\partial z_k} = \hat{y}_k$.

Esses dois cenários podem ser resumidos numa única equação geral dada por

$$\frac{\partial \mathcal{L}}{\partial z_k} = \hat{y}_k - y_k. \quad (15)$$

Note que na equação (15) o resultado será $\hat{y}_k - 1$ ou \hat{y}_k simplesmente, dependendo se y_k é zero ou um. Finalmente, para terminarmos o cálculo de $\nabla_W \mathcal{L}$, que é o que estamos

buscando, precisamos ainda estimar a derivada dos *logits* com relação à matriz peso W . Esse cálculo é mais simples, temos

$$z_k = W_{ik}x_i \quad \Rightarrow \quad \frac{\partial z_k}{\partial W_{ik}} = x_i. \quad (16)$$

Substituindo (16) e (15) em (8), temos

$$\nabla_W \mathcal{L} = \frac{\partial \mathcal{L}}{\partial W_{ij}} = (\hat{y}_j - y_j)x_i. \quad (17)$$

A equação (17) representa um produto diático ou produto tensorial entre o vetor $\hat{y}_j - y_j$ e o vetor x_i . Esse produto muitas vezes é representado pelo símbolo \otimes e em contextos de álgebra linear é chamado em inglês de outer product. Na biblioteca numpy ele é calculado pela função np.outer(entrada, saída_preditiva - saída_esperada) do nosso algoritmo.

O último passo do algoritmo é ajustar os pesos da matriz W a uma certa taxa caminhando no sentido oposto ao do gradiente calculado a fim de que possamos calibrar essa matriz para que seus pesos minimizem a entropia cruzada entre a saída esperada e a saída predita pelo modelo.

4 Como formular um problema de otimização?

Para iniciarmos nossos estudos formais no campo da otimização precisamos construir algumas definições preliminares. A primeira delas diz respeito a formulação de um problema de otimização. Em muitos casos formular um problema pode ser até mais difícil do que resolvê-lo. Mais ainda, um problema mal formulado pode acabar demandando um esforço muito maior em sua solução justamente pela falta de clareza na formulação. Eu costumo brincar com meus alunos de Engenharia que diferentemente do treinamento recebido na faculdade, onde os problemas já chegam bem enunciados na forma de provas, listas de exercícios e trabalhos, na vida real o engenheiro deve ser capaz de olhar uma situação concreta, formular o problema a ser resolvido e depois resolvê-lo. Nesse sentido, antes de entrarmos nos métodos, algoritmos e esquemas numéricos destinados à solução de problemas de otimização, precisamos aprender a formular estes problemas.

Um problema típico de otimização é geralmente enunciado da seguinte forma: “*Seja $f(\mathbf{x})$ uma função objetivo e \mathbf{x} um vetor de projeto n -dimensional, desejamos encontrar \mathbf{x} que minimiza ou maximiza $f(\mathbf{x})$ sujeito à $d_i(\mathbf{x}) \leq a_i$ para $i = 1, 2, \dots, m$ e $e_i(\mathbf{x}) = b_i$ para $i = 1, 2, \dots, p$, em que $d_i(\mathbf{x})$ representa um conjunto de restrições expressas por desigualdades, $e_i(\mathbf{x})$ representa um conjunto de restrições expressas por igualdades e a_i e b_i são constantes.*

No contexto em que $f(\mathbf{x})$, $d_i(\mathbf{x})$ e $e_i(\mathbf{x})$ são funções lineares, o problema de otimização cai na categoria do que chamamos de *programação linear*. Aqui a palavra *programação* possui um sentido mais de planejamento do que propriamente de algoritmo. No contexto em que a função objetivo é quadrática e as restrições são lineares, o problema de otimização é categorizado como um problema de *programação quadrática*. Finalmente, quanto a função objetivo for não linear ou quadrática e as restrições forem não lineares, caímos no campo da *programação não-linear*.

Em muitos contextos os problemas de otimização não estão associados necessariamente à restrições. O exemplo que demos no início dessa seção, associado ao processo de aprendizado de algoritmos de aprendizado de máquina baseado em modelos de linguagem, é

um desses tipos de problema. Note que no nosso processo de minimização da entropia cruzada entre os dados de saída fornecidos na etapa de aprendizagem e os dados previstos pelo modelo, não precisamos impor nenhuma restrição ou vínculo à função objetivo.

Dependendo da situação a aplicação de restrições pode complicar muito a solução do problema de otimização. Mas em alguns casos, restrições podem ser facilmente implementadas. Se no nosso problema de otimização da função custo no exemplo dado na seção anterior tivéssemos implementado um condicional para finalização da etapa de treinamento do modelo baseado numa tolerância a ser atingida, essa não deixaria de ser uma restrição, uma vez que essa tolerância certamente esterá relacionada aos pesos da matriz W . Nesse caso, a implementação de uma restrição não afetaria a complexidade do algoritmo. Mas em muitos cenários esse não é o caso.

Em contextos de problemas com restrições o ideal é que o número de restrições seja menor ou igual à dimensão do nosso vetor de projeto, ou seja $p + m \leq n$. Quando o problema é formulado de tal sorte que $p + m > n$ dizemos que o problema é super restrito e isso pode ser um problema.

Um outro aspecto interessante de problemas de otimização sem restrições é que podemos apelar à métodos gráficos para compreender melhor o desenvolvimento de métodos numéricos voltados a essa finalidade. A figura (2) ilustra bem como um problema de otimização unidimensional pode ser compreendido tanto como um problema de minimização de $f(x)$ quanto de maximização de $-f(x)$ e como em ambos os casos a derivada primeira e segunda podem ser utilizadas como informações úteis para encontrarmos o vetor de projeto unidimensional responsável pelo processo de otimização. Já a figura (2 b) ilustra como curvas de nível podem ser utilizadas para encontrarmos o ponto ótimo de uma função de duas variáveis. Aqui a analogia de caminharmos numa montanha buscando vales e picos será muito útil no desenvolvimento de métodos de otimização multidimensional sem restrições por meio de esquemas baseados no cálculo do gradiente da função objetivo.

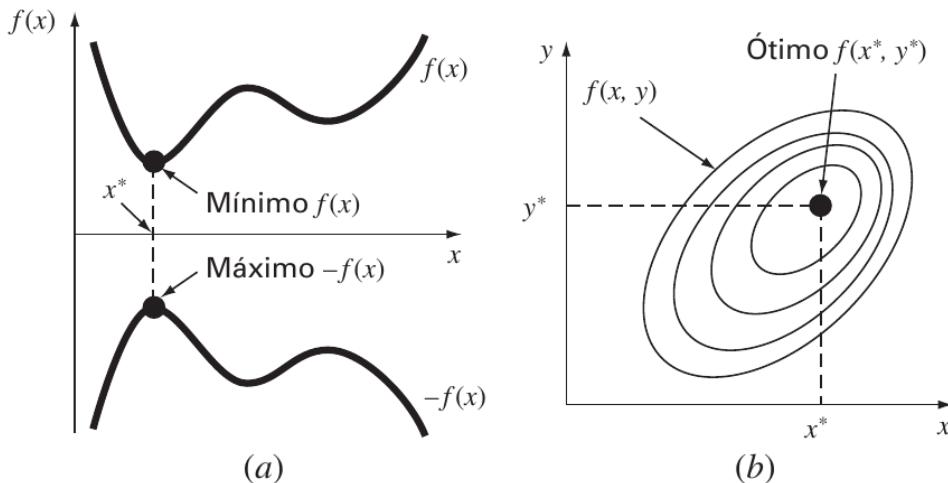


Figura 2: Representação gráfica de um problema de otimização unidimensional (a) e de um problema de otimização bidimensional (b).

A figura (2b) mostra um ponto de ótimo no plano xy associado a um máximo ou mínimo local da função objetivo $f(x, y)$. Uma representação tridimensional um pouco mais clara desse caminho de busca, que leva em consideração aspectos topológicos da superfície que desejamos otimizar é ilustrado na figura (3). Nessa figura, o caminho azul no plano xy representa a busca do ponto ótimo da função a partir de um ponto inicial

que leva em conta as características de $f(x, y)$ para ajustar a direção de busca afim de encontrar o vetor bidimensional de projeto que otimize nossa função objetivo. Nesse caso, as características da função objetivo que irão nortear o caminho de busca consistem no gradiente da função, ou seja ∇f e em sua Hessiana, cuja definição veremos mais para frente no momento em que formos apresentar métodos baseados no gradiente.

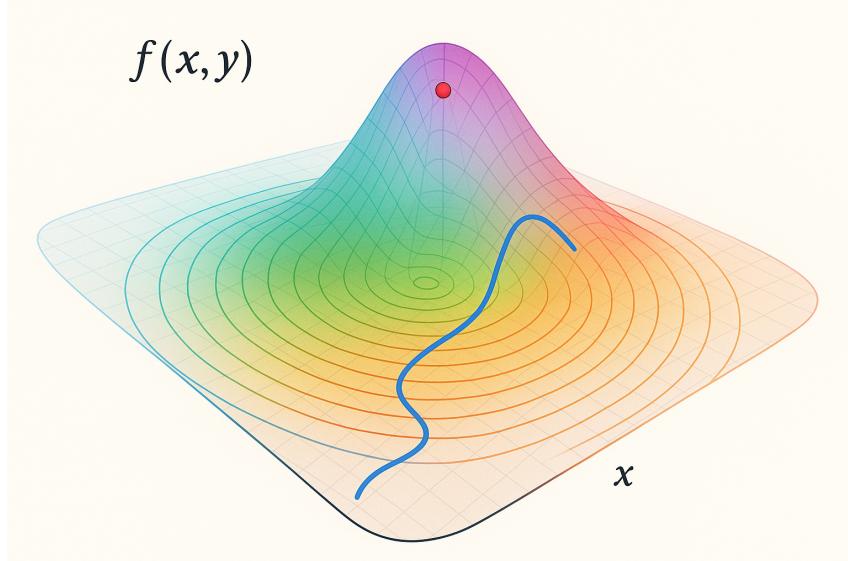


Figura 3: Representação gráfica de uma superfície tridimensional vinculada a um função objetivo bidimensional e do caminho de busca no plano xy do ponto ótimo.

5 Razão áurea e otimização 1D

Um dos primeiros métodos de otimização que veremos aqui é o método da razão áurea, muito interessante para otimização unidimensional sem restrições. Esse método é análogo ao método da bissecção utilizado para identificação de raízes de funções de uma única variável. Ou seja, é um método intervalar de busca. Entretanto o que se busca aqui não é um ponto x que zere a função objetivo, mas que a otimize.

Diferentemente do método da bissecção em que utilizamos dois valores da função para definirmos um intervalo de busca inicial, aqui utilizaremos dois intervalos iniciais de busca. Isso implica na escolha inicial de três valores da função para a definição de dois intervalos iniciais ℓ_1 e ℓ_2 , ambos definidos dentro do intervalo maior ℓ_0 , de tal sorte que $\ell_0 = \ell_1 + \ell_2$. A ideia aqui é checar se o máximo ou mínimo ocorreu no intervalo ℓ_1 ou no intervalo ℓ_2 e em função disso ir diminuindo progressivamente esse intervalo.

A figura (4) ilustra como esses intervalos se relacionam com o processo de busca. Aqui começamos com um intervalo ℓ_0 para o qual assumimos a existência de um único ponto de máximo ou mínimo. Esse intervalo ℓ_0 é chamado de intervalo unimodal. Aqui para a definição do intervalo ℓ_0 precisamos apenas dos pontos x_ℓ e x_u , associados respectivamente ao limite inferior e superior do intervalo. Em seguida, dentro desse intervalo, dividimos ℓ_0 em dois intervalos ℓ_1 e ℓ_2 de tal sorte que $\ell_0 = \ell_1 + \ell_2$. Se identificarmos que o ponto de máximo ocorreu no intervalo ℓ_1 , como ilustra a figura (4) por exemplo, o próximo passo consiste em dividir ℓ_1 em dois intervalos ℓ_2 e ℓ_3 , de tal sorte que $\ell_1 = \ell_2 + \ell_3$. Uma segunda

regra que aplicaremos aqui para a divisão dos intervalos é que

$$\frac{\ell_1}{\ell_0} = \frac{\ell_2}{\ell_1} = \frac{\ell_3}{\ell_2} = \frac{\ell_4}{\ell_3} = \dots = \frac{\ell_n}{\ell_{n-1}} = R. \quad (18)$$

Note que se $\ell_0 = \ell_1 + \ell_2$ e $\ell_1/\ell_0 = \ell_2/\ell_1$, então

$$\frac{\ell_1}{\ell_1 + \ell_2} = \frac{\ell_1}{\ell_2} \Rightarrow 1 + \frac{\ell_2}{\ell_1} = \frac{\ell_1}{\ell_2} \Rightarrow 1 + R = \frac{1}{R} \Rightarrow R^2 + R - 1 = 0. \quad (19)$$

Note que a raiz positiva da equação quadrática expressa em (19) é dada por

$$R = \frac{-1 + \sqrt{5}}{2} = \varphi = 0.61803\dots \quad (20)$$

O número φ que surge no processo é um importante número irracional, presente no mundo natural, em projetos arquitetônicos do período clássico e em estruturas matemáticas como a própria sequência de Fibonacci. Esse número define uma proporção que os gregos antigos chamavam de razão áurea, por acreditarem revelar uma proporção esteticamente perfeita. Vários templos gregos foram projetados e construídos utilizando a razão áurea como parâmetro de projeto. Por esse motivo, o método aqui descrito é conhecido como método da razão áurea.

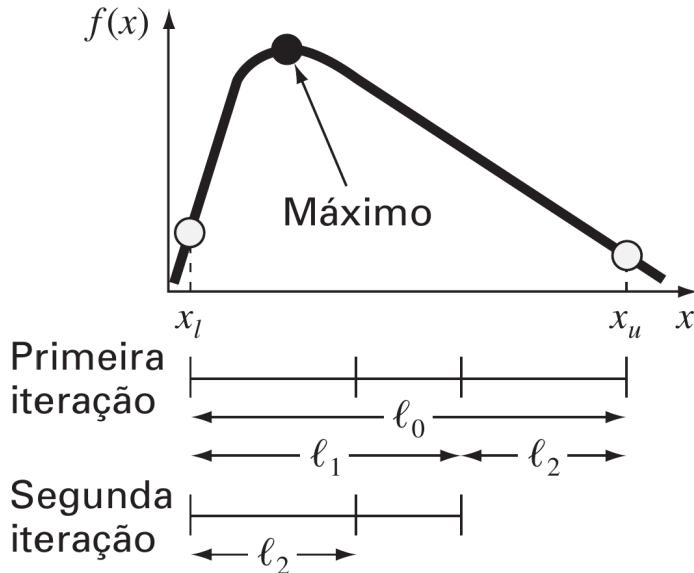


Figura 4: Intervalos de busca de pontos de ótimo utilizados no método da razão áurea

Se pegarmos um quadrado inicial e utilizarmos a razão áurea para dividirmos esse quadrado em sub-retângulos internos, ao fazermos isso indefinidamente vamos formar uma espiral cuja razão dos comprimentos dos círculos internos vai diminuindo numa proporção que surge em diversos organismos, como plantas e animais. A figura (5) ilustra bem esse conceito. É importante notar que no nosso contexto de otimização unidimensional sem restrições a razão áurea surge como uma consequência natural dos critérios estabelecidos para a definição das divisões do intervalo inicial de busca em sub-intervalos. Afinal de contas, se vamos dividir esses intervalos de busca, precisamos de um critério objetivo para realizar a divisão. A definição desse critério por meio das equações expressas anteriormente leva naturalmente ao surgimento da razão áurea como a razão entre os intervalos de busca que vão se reduzindo à medida que avançamos no processo.

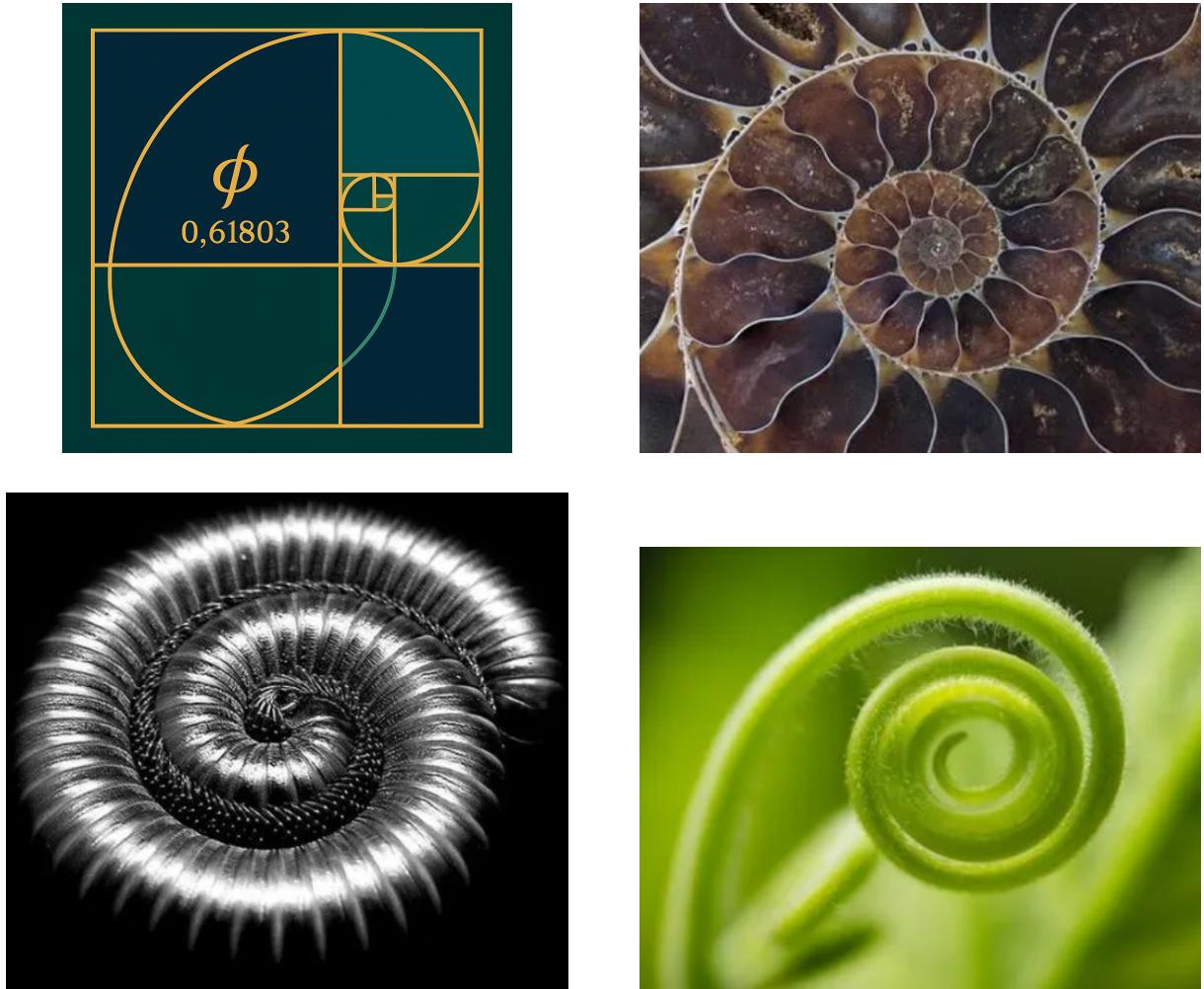


Figura 5: Ilustração do surgimento da razão áurea na natureza

5.1 Exemplo de otimização 1D pelo método da razão áurea

Agora que apresentamos conceitualmente o método da razão áurea, vamos a um exemplo numérico para entendermos como ele se aplica à solução de um problema real de otimização 1D. Para isso, considere que a função objetivo que desejamos otimizar seja a função $f(x) = 2 \sin(x) - x^2/10$. Para aplicarmos o método da razão áurea, precisamos definir um intervalo inicial de busca para compor o primeiro intervalo unimodal ℓ_0 . Nesse caso considere $x_\ell = 0$ e $x_u = 4$.

- O primeiro passo aqui consiste em utilizar a razão áurea para estimativa do comprimento do subintervalo ℓ_1 :

$$\ell_1 = \ell_0 R \Rightarrow \ell_1 = \frac{\sqrt{5} - 1}{2} \times 4 = 2.472. \quad (21)$$

- O passo seguinte consiste em calcular dois pontos x_1 e x_2 internos a esse intervalo como:

$$x_1 = x_\ell + \ell_1 = 2.472 \quad \text{e} \quad x_2 = x_u - \ell_1 = 1.528. \quad (22)$$

- Agora, calculamos $f(x_1)$ e $f(x_2)$. Esse cálculo fornece $f(x_1) = 0.63$ e $f(x_2) = 1.765$. Como $x_1 > x_2$ e $f(x_1) < f(x_2)$ sabemos já que o máximo não se encontra no intervalo

$[x_1, x_u]$. Podemos então descartar esse intervalo e reduzir nosso novo intervalo de busca para $[x_\ell, x_1 = \ell_1]$. Em outras palavras, precisamos agora dividir o intervalo de busca ℓ_1 em sub-intervalos.

4. Para estimativa do novo sub-intervalo de busca fazemos

$$\ell_2 = \ell_1 R \Rightarrow \ell_2 = \frac{\sqrt{5} - 1}{2} \times 2.472 = 1.528. \quad (23)$$

5. Usamos agora ℓ_2 para calcular novos valores internos de x_1 e x_2 no subintervalo ℓ_1 como

$$x_1 = x_\ell + \ell_2 = 0.944 \quad \text{e} \quad x_2 = x_u - \ell_2 = 1.528. \quad (24)$$

Note que agora $x_u = 2.472$ ao invés de 4.

6. Agora, calculamos novos valores para $f(x_1)$ e $f(x_2)$. Esse cálculo fornece $f(x_1) = 1.531$ e $f(x_2) = 1.765$. Como agora $x_1 < x_2$ e $f(x_2) > f(x_1)$ sabemos já que o máximo não se encontra no intervalo $[x_\ell, x_1]$. Podemos então descartar esse intervalo e reduzir nosso novo intervalo de busca para $[x_1, x_u]$.
7. A continuidade desse processo de redução do intervalo de busca por meio do uso da razão áurea indo até a décima iteração leva nossa solução ao ponto ótimo $x_o = 1.4276$. Teste isso em casa para compreender melhor o método.

6 O método da interpolação quadrática

Um próximo método interessante no campo de esquemas de otimização de funções unidimensionais sem restrições é o método da interpolação quadrática. Esse método se baseia num conceito muito simples: toda parábola tem um ponto de máximo ou mínimo. De tal sorte que próximo a um mínimo ou máximo local de uma função contínua sempre poderemos ajustar uma parábola por perto, cujo máximo (ou mínimo) estará próximo do máximo (ou mínimo) local da função objetivo.

A figura (6) ilustra bem a essência do método. Nesse caso, concebemos uma parábola que possui três pontos comuns com a função objetivo $f(x)$. Esses pontos comuns são x_0, x_1, x_2 . A ideia é encontrar os coeficientes a, b, c que montam a parábola $g(x) = ax^2 + bx + c$, de tal sorte que:

$$f(x_0) = g(x_0), \quad f(x_1) = g(x_1) \quad \text{e} \quad f(x_2) = g(x_2). \quad (25)$$

Uma vez que tenhamos os valores dos coeficientes da parábola, derivamos a equação quadrática e igualamos à zero para isolar o valor x_3 do máximo ou mínimo local. Você pode tentar fazer o procedimento em casa para obter:

$$x_3 = \frac{f(x_0)(x_1^2 - x_2^2) + f(x_1)(x_2^2 - x_0^2) + f(x_2)(x_0^2 - x_1^2)}{2f(x_0)(x_1 - x_2) + 2f(x_1)(x_2 - x_0) + 2f(x_2)(x_0 - x_1)}. \quad (26)$$

Uma vez que saibamos o valor de x_3 a partir da equação (26), utilizamos os calculamos o de $f(x_3)$ e o comparamos com os valores de $f(x_0), f(x_1)$ e $f(x_2)$ para atualizarmos um novo intervalo no qual novos pontos serão definidos para construção de uma próxima parábola, ainda mais próxima do ponto de máximo ou mínimo local real de $f(x)$.

Para a ideia não ficar muito abstrata, vamos a um exemplo numérico. Considere o uso da interpolação quadrática para achar o ponto de máximo da função $f(x) = 2 \sin(x) - x^2/10$, abordada no problema anterior pelo método da razão áurea. Nesse caso, vamos partir dos seguintes valores iniciais $x_0 = 0$, $x_1 = 1$ e $x_2 = 4$. A substituição desses valores na equação (26) fornece $x_3 = 1.5055$. Para esses valores de x_0, x_1, x_2, x_3 calculamos $f(x_0) = 0$, $f(x_1) = 1.5829$, $f(x_2) = -3.1136$ e $f(x_3) = 1.7691$. Nesse caso temos que $f(x_1) > f(x_0)$ e $f(x_3) > f(x_1)$. Consequentemente o máximo não pode estar no intervalo $[x_0, x_1]$. Dessa forma, excluímos esse intervalo e definimos os novos três pontos da próxima parábola como $x_0 = 1$, $x_1 = 1.5055$ e $x_2 = 4$. Para esses novos valores, calculamos o novo valor de x_3 por meio da equação (26) como $x_3 = 1.4903$. E assim seguimos. É possível ver que apenas com duas iterações já estamos nos aproximando do ponto de máximo exato de $x_o = 1.4275$.

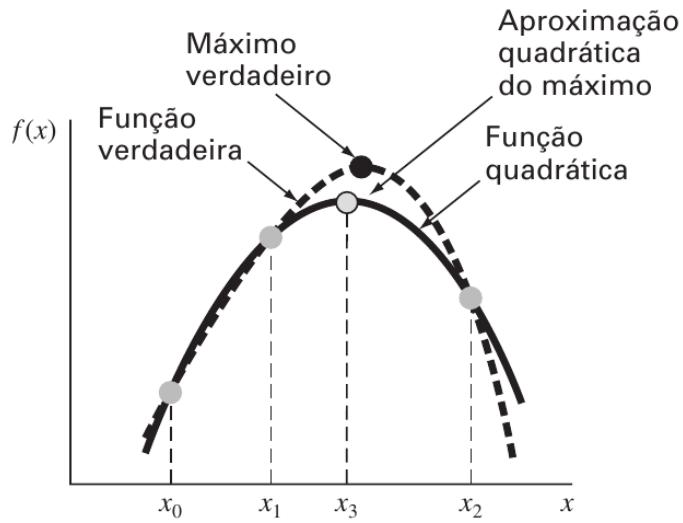


Figura 6: Representação visual dos conceitos por trás do método da interpolação quadrática

7 O método de Newton

No capítulo referente à métodos voltados para obtenção de raízes de equações de uma variável, vemos o método de Newton-Raphson. Naquela ocasião, a raiz de uma função $f(x)$ era dada por

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad (27)$$

se definirmos $g(x) = f'(x)$ podemos usar essa ideia para encontrar o ponto ótimo de uma função aplicando o método de Newton-Raphson para encontrar o zero não de $f(x)$, mas de $g(x)$, de tal sorte que esse ponto passa a ser dado por

$$x_{i+1} = x_i - \frac{f''(x_i)}{f'(x_i)}. \quad (28)$$

Note que agora para podermos utilizar o método de Newton para obtenção do ponto ótimo de uma função de uma variável num problema sem restrições, precisamos conhecer

analiticamente tanto o valor da primeira derivada da função quanto o valor de sua segunda derivada. Novamente, em contextos nos quais não temos a expressão analítica da função $f(x)$ podemos recorrer a esquemas numéricos de diferenciação baseados por exemplo no método das diferenças finitas para tal finalidade.

Para fins de ilustração desse método, consideremos ainda nosso exemplo no qual $f(x) = 2 \sin(x) - x^2/10$. Para esse caso, temos $f'(x) = 2 \cos(x) - x/5$ e $f''(x) = -2 \sin(x) - 1/5$, de tal sorte que a aplicação da equação (28) nos leva a

$$x_{i+1} = x_i - \left[\frac{-2 \cos(x_i) - x_i/5}{-2 \sin(x_i) - 1/5} \right]. \quad (29)$$

O processo iterativo associado à equação (29) é ilustrado na tabela (1).

i	x_i	$f(x_i)$	$f'(x_i)$	$f''(x_i)$
1	2.5	0.57194	-2.10229	-1.39694
2	0.99508	1.57859	0.88985	-1.87761
3	1.46901	1.77385	-0.09058	-2.18965
4	1.42764	1.77573	-0.00020	-2.17954
5	1.42755	1.77573	0.00000	-2.17952

Tabela 1: Valores numéricos de $x, f(x), f'(x), f''(x)$ para o processo iterativo de solução do problema-exemplo de otimização 1D sem restrições pelo método de Newton

Note que o método apresenta rápida convergência nesse caso, convergindo para o valor correto após apenas 5 iterações. É importante falar que nesse caso, como em qualquer esquema iterativo, dependendo das características da função objetivo e do ponto de partida, o método pode divergir.

8 Otimização multidimensional sem restrições

No contexto em que a função objetivo é uma função de mais de uma variável, classificamos o problema de otimização como um problema multidimensional. Nessa seção iremos explorar ideias e conceitos orientados à esquemas de otimização multidimensionais sem restrições. Nesse campo, existem duas categorias gerais de métodos no campo da otimização clássica. Os métodos diretos e os métodos gradientes. A diferença principal entre essas duas categorias diz respeito ao uso que se faz da função objetivo no processo de busca por um ponto ótimo. Enquanto métodos diretos não dependem do cálculo das derivadas da função objetivo com relação ao vetor de projeto, os métodos gradientes se baseiam justamente em cálculo de derivadas multidimensionais da função objetivo para fins de otimização.

Um exemplo fácil de entendermos de um método direto é o método da busca aleatória, que se baseia na comparação do valor da função objetivo em pontos aleatórios do domínio de busca. Nesse método espalhamos pontos de forma randômica no domínio de busca, olhamos para o valor da função objetivo em cada um desses pontos, pegamos o valor máximo e vamos restringindo o domínio de busca para as imediações desse ponto ótimo. Fazendo sucessivas reduções do espaço de busca em torno de cada ponto ótimo da etapa anterior podemos encontrar pontos de máximo ou mínimo em domínios multidimensionais mesmo para funções descontínuas e de difícil diferenciação analítica.

Já os métodos gradientes utilizam de maneira inteligente a topologia da função objetivo a fim de encontrar pontos críticos mais rapidamente. Exemplos de métodos gradientes são o método do gradiente descendente ou a clive máximo, o método dos gradientes conjugados, método de Newton e de Marquardt. Veremos todos esses esquemas nesse capítulo.

Cada uma dessas abordagens possui vantagens e desvantagens. Enquanto o método da busca aleatória consegue encontrar pontos ótimos para funções objetivos extremamente complicadas e mal comportadas, o mesmo pode demorar muito tempo para encontrar esses pontos críticos. Uma vez que os métodos gradientes se baseiam no conhecimento matemático da topologia da função objetivo e usam essas informações para caminhar no espaço de busca procurando por um ponto ótimo, é de se esperar que esses métodos sejam otimizados para encontrar rapidamente esses pontos de interesse.

Tanto os métodos diretos quanto os métodos gradientes estão associados ao que chamamos de otimização clássica. Mais recentemente, uma nova classe de métodos de otimização baseados na natureza, como os algoritmos genéticos, as redes neurais e o método do enxame de partículas, se enquadram numa classe que chamamos de métodos heurísticos ou meta-heurísticos. Esses métodos podem ser aplicados a funções descontínuas e são capazes de explorar grandes regiões de busca, mas podem ser computacionalmente bem caros.

Além das abordagens aqui citadas, existem esquemas híbridos que misturam meta-heurística com métodos gradientes. Um método recente que se vale dessa abordagem híbrida são as redes neurais informadas por física (PINN) que resolvem equações diferenciais parciais por meio do treinamento de uma grande rede neural que busca zerar localmente uma função erro associada a uma equação diferencial em pontos aleatórios de um domínio de solução. Essa técnica é interessante, pois viabiliza a realização de simulações computacionais sem a necessidade de malhas de cálculo. Nesse curso iremos focar nossa atenção nos métodos clássicos de otimização baseados em gradientes.

8.1 Métodos gradientes

Antes de apresentarmos alguns dos métodos de otimização clássica baseados no cálculo dos gradientes da função objetivo, vamos recapitular alguns conceitos do nosso curso de Cálculo 3. Para isso, considere uma função de duas variáveis independentes $f(x, y)$, como representada na figura (7). Considere agora um par de coordenadas (a, b) representando

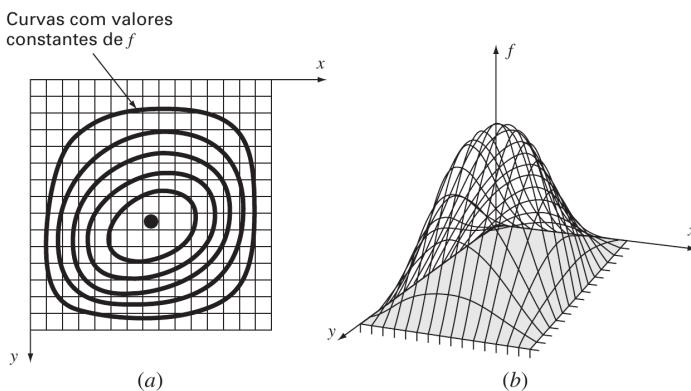


Figura 7: Ilustração de curvas de nível (a) de uma função $f(x, y)$ que pode ser imaginada como uma montanha (b)

um ponto nessa curva e um eixo h que forma um ângulo θ com relação ao eixo x a partir

do ponto (a, b) como ilustrado na figura (8). Esse eixo h representará para nós uma direção inicial de busca de um ponto crítico da função $f(x, y)$. Podemos imaginar aqui

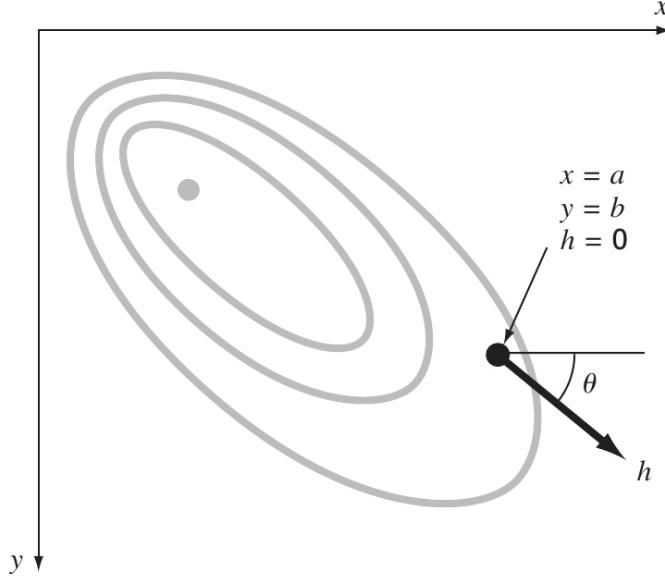


Figura 8: Definição de um ponto (a, b) no plano xy e de uma direção de busca h que forma um ângulo θ com o eixo x .

a função objetivo como uma grande montanha e o eixo h representará um corte nessa montanha para caminharmos ao longo dessa direção de busca. Dessa forma, podemos imaginar que a altura da montanha na direção de busca h é uma nova função $g(h)$ que estará relacionada com a função $f(x, y)$. A inclinação dessa montanha 1D $g(h)$ a partir do ponto (a, b) é representada então por $g'(0)$. A relação entre $g'(0)$ e as derivadas parciais de $f(x, y)$ avaliadas no ponto (a, b) é dada por:

$$g'(0) = \left(\frac{\partial f}{\partial x} \right)_{a,b} \cos(\theta) + \left(\frac{\partial f}{\partial y} \right)_{a,b} \sin(\theta). \quad (30)$$

Temos também que $\hat{e}_h = \cos(\theta)\hat{e}_x + \sin(\theta)\hat{e}_y$. Definindo o vetor gradiente ou operador ∇ como

$$\nabla = \frac{\partial}{\partial x} \hat{e}_x + \frac{\partial}{\partial y} \hat{e}_y + \frac{\partial}{\partial z} \hat{e}_z, \quad (31)$$

temos para o caso bidimensional que

$$g'(0) = \nabla f \cdot \hat{e}_h. \quad (32)$$

Além disso, temos que a direção de acidente máximo em um dado ponto é dada pelo próprio gradiente de f . Dessa forma, para um campo escalar f , o gradiente de f será um campo vetorial que localmente apontará na direção de maior inclinação a partir daquele ponto. Sabemos também que no caso 1D, um ponto de máximo ou mínimo é aquele para o qual a derivada da função é nula. Mais ainda, utilizamos o sinal da segunda derivada da função para determinar se um determinado ponto crítico é um máximo ou um mínimo local. No campo das funções multidimensionais a lógica é semelhante, mas a análise pode ficar um pouco mais complicada.

Sabemos por exemplo que um ponto de máximo ou mínimo para uma função de mais de uma variável é um ponto onde o gradiente da função objetivo é zero. Entretanto,

para sabermos se esse ponto é um ponto de máximo ou mínimo local, precisamos avaliar uma quantidade associada às derivadas segundas da função objetivo chamada de matriz Hessiana de f . Mais ainda, no caso multidimensional um mesmo ponto pode ser um ponto de máximo num certo sentido de busca e de mínimo no outro sentido, como ilustra a figura (9). Nesse cenário dizemos que o ponto crítico é um ponto de sela. Ele é tanto um máximo quanto um mínimo local.

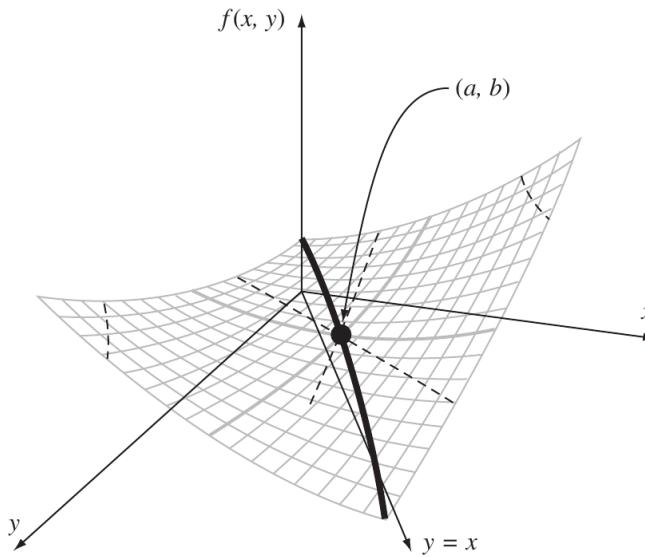


Figura 9: Exemplo de uma situação em que o ponto crítico é ao mesmo tempo um máximo e um mínimo local (ponto de sela)

8.2 A matriz Hessiana: um critério para avaliação de pontos críticos

Suponha que após nossa caminhada na montanha chegamos em um ponto crítico, ou seja, encontramos as coordenadas x, y para as quais $\nabla f = \mathbf{0}$. A pergunta que queremos responder aqui é: podemos construir um critério matemático objetivo, baseado nas características topológicas de $f(x, y)$ para determinar se esse ponto crítico é um máximo local, um mínimo local ou um ponto de sela? Para respondermos essa pergunta vamos precisar definir algumas coisas. Considere no caso bidimensional a seguinte estrutura matricial

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \Rightarrow \det H = |H| = \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2. \quad (33)$$

A partir dessa estrutura, conhecida como matriz Hessiana de f , podemos com base numa avaliação do sinal do determinante de H e das derivadas segundas da função objetivo com relação à x determinar a natureza do ponto crítico em questão. O critério aqui para o caso bidimensional é definido como:

- Se $|H| > 0$ e $\frac{\partial^2 f}{\partial x^2} > 0$ no ponto crítico, então esse ponto é um mínimo local;
- Se $|H| > 0$ e $\frac{\partial^2 f}{\partial x^2} < 0$ no ponto crítico, então esse ponto é um máximo local;

- Se $|H| < 0$ no ponto crítico, então esse ponto é um ponto de sela;

Para checar a validade desse critério para o caso bidimensional, vamos ao exemplo ilustrado pela figura (10). Essa figura mostra três funções objetivos simples, cada uma vinculada a um ponto crítico de natureza distinta.

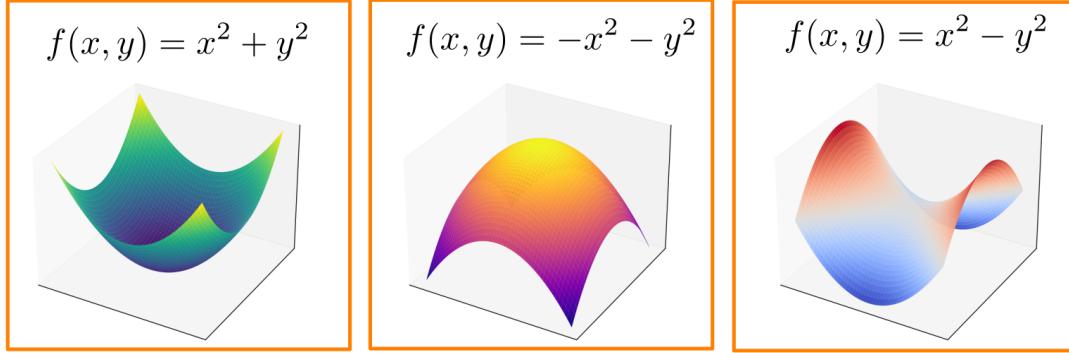


Figura 10: Três cenários possíveis para a natureza do ponto crítico. A figura da esquerda ilustra um mínimo local, a do meio um máximo local e a da direita representa um ponto de sela.

Para o exemplo da figura (10) é fácil ver que os pontos críticos para as três funções são ocorrem no mesmo local. As três apresentam pontos críticos em $(x, y) = (0, 0)$. Verifique você mesmo agora os sinais de $|H|$ e da segunda derivada da função objetivo com relação a x e perceba a validade dos critérios estabelecidos acima.

Uma questão relevante que pode surgir é: como estender esse critério para um contexto de busca multidimensional em um espaço de dimensão $n > 2$? Nesse contexto, a representação visual de uma montanha tridimensional não é mais possível. Apesar disso, ainda podemos utilizar o critério baseado no determinante da matriz Hessiana para identificação da natureza do nosso ponto crítico. Entretanto, é mais conveniente reescrever esse critério utilizando ideias mais gerais de álgebra linear.

Considere agora então o caso mais geral possível em que a função objetivo é uma função de n variáveis definida por $f(x_1, x_2, x_3, \dots, x_n)$. Nesse caso, a matriz Hessiana de f é dada por

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (34)$$

Pelo teorema de Schwarz, temos que as derivadas mistas são simétricas, ou seja:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i} \quad (35)$$

Assim, pela própria definição, a Hessiana é sempre uma matriz simétrica real. Essa simetria de H nos permite representar essa matriz em termos de uma matriz diagonal quando H é escrita num sistema de base formado por seus autovetores. Esse tipo de representação diagonal de uma matriz simétrica em sua base de autovetores é chamada de representação espectral de matrizes simétricas. Caso queira entender por que isso é possível, recomendamos que assista essa aula gratuita no YouTube clicando [aqui](#). Mais

ainda, quando representamos uma matriz simétrica em sua base de autovetores utilizando a representação espectral, os termos da diagonal são os autovalores da própria matriz. Esses autovalores são calculados por meio da solução de um polinômio em λ construído com base na relação

$$\det(H - \lambda I) = 0 \Rightarrow \lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n. \quad (36)$$

Nesse caso mais geral, os sinais dos autovalores vão representar as curvaturas principais de f em cada direção e servirão para identificar o tipo de ponto crítico em questão. Por mais que essas informações possam parecer um pouco abstratas quando a vemos pela primeira vez, elas ilustram propriedades e características fundamentais de matrizes que são utilizadas para descrever o comportamento de meios contínuos. Em outras palavras, o estudo formal das estruturas matemáticas por trás da descrição do comportamento de sólidos e fluidos, utiliza esses conceitos para representar por exemplo o comportamento das tensões principais e das direções de deformação principal de sólidos elásticos e outros materiais de interesse de Engenharia. Recomenda-se que aqueles que se interessarem pelo tema assistam o curso de Mecânica dos Meios Contínuos disponibilizado de forma gratuita no canal do YouTube, Ciência e Brisa.

A tabela (2) resume os critérios de avaliação da natureza do ponto crítico de uma função objetivo multidimensional em termos da análise dos sinais dos autovalores de f .

Item	Significado
Autovalores da Hessiana	Curvaturas principais de f em cada direção
Sinais dos autovalores	Classificam o tipo de ponto crítico
Todos positivos	Mínimo local
Todos negativos	Máximo local
Sinais mistos	Ponto de sela
Algum autovalor nulo	Teste inconclusivo

Tabela 2: Critérios de avaliação da natureza de um ponto crítico para uma função objetivo multidimensional em termos dos sinais dos autovalores da matriz Hessiana de f

Para o assunto não ficar muito abstrato, vamos a um exemplo de aplicação. Considere então a função:

$$f(x, y, z) = x^2 + y^2 - z^2$$

Nesse caso, o **gradiente** de f é dado por:

$$\nabla f(x, y, z) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \\ -2z \end{bmatrix}$$

É fácil perceber que aqui o ponto crítico ocorre em $(x, y, z) = (0, 0, 0)$. Dessa forma, a **matriz Hessiana** de f no ponto crítico é dada por

$$H_f(x, y, z) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

Faça o teste em casa e mostre que para esse exemplo os **autovalores** da Hessiana são dados por

$$\lambda_1 = 2, \quad \lambda_2 = 2, \quad \lambda_3 = -2.$$

De acordo com os critérios estabelecidos na tabela (2), como a matriz Hessiana possui dois autovalores positivos e um negativo, o ponto crítico é um **ponto de sela**. A tabela (3) apresenta um resumo da análise realizada para este exemplo. Como no nosso exemplo

Item	Descrição
Função	$f(x, y, z) = x^2 + y^2 - z^2$
Ponto crítico	$(0, 0, 0)$
Gradiente no ponto	$\nabla f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
Hessiana no ponto	$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -2 \end{bmatrix}$
Autovalores	$\lambda_1 = 2, \lambda_2 = 2, \lambda_3 = -2$
Classificação	Ponto de sela (Hessiana indefinida)

Tabela 3: Resumo da análise da natureza do ponto crítico para a função $f(x, y, z) = x^2 + y^2 - z^2$

escolhemos uma função de 3 variáveis, não conseguimos representá-la em termo de uma curva tridimensional. Mas podemos fazer alguns cortes em planos específicos para avaliar o comportamento dessa função. A figura (11) mostra que essa função apresenta o comportamento de um parabolóide elíptico no plano z e de pontos de sela nos planos x e y .

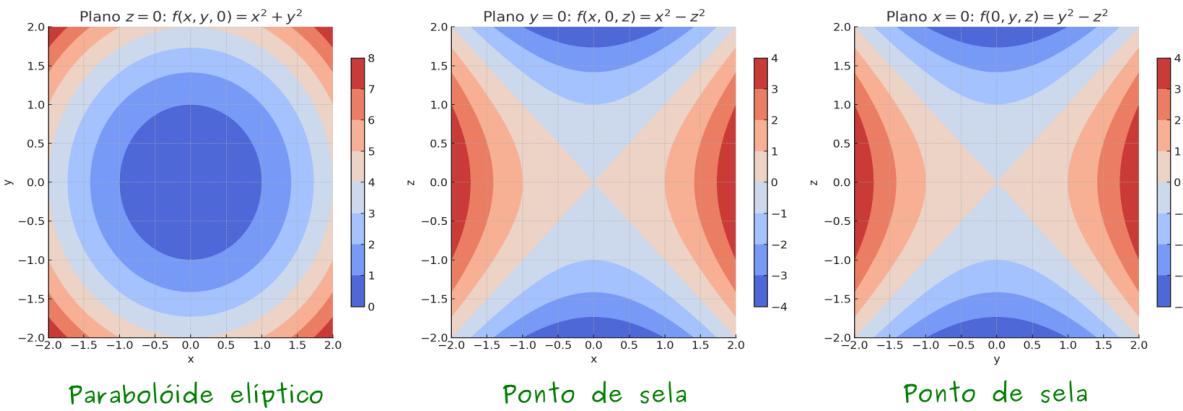


Figura 11: Comportamento nos planos z, y, x da função $f(x, y, z) = x^2 + y^2 - z^2$

8.3 O método do aclive máximo

O primeiro método baseado no cálculo do gradiente que iremos apresentar aqui é o método do aclive máximo. A ideia desse método é caminhar uma certa distância h^* ao longo da direção do gradiente a partir de um ponto inicial. Em seguida, para-se e faz-se uma nova avaliação de direção para ajustar a caminhada na direção do gradiente a partir do novo ponto de parada. O processo é realizado algumas vezes até atingirmos o ponto ótimo. Para entendermos a essência do método do aclive máximo façamos um exemplo.

Considere então a função objetivo $f(x, y) = 2xy + 2x - x^2 - 2y^2$ e um ponto inicial $(x_0, y_0) = (-1, 1)$. O primeiro passo do processo é tentar definir uma função $g(h)$ que seja idêntica à função $f(x, y)$ na direção do gradiente a partir de (x_0, y_0) . Esse procedimento é chamado de parametrização e consiste na aplicação das seguintes relações

$$x = x_0 + h \left(\frac{\partial f}{\partial x} \right)_{x_0, y_0} \quad \text{e} \quad y = y_0 + h \left(\frac{\partial f}{\partial y} \right)_{x_0, y_0}. \quad (37)$$

Para nossa função objetivo deste exemplo temos que $(\partial f / \partial x)_{x_0, y_0} = 6$ e $(\partial f / \partial y)_{x_0, y_0} = -6$, de tal sorte que nossas relações paramétricas são dadas por $x = -1 + 6h$ e $y = 1 - 6h$. Essas relações são então substituídas na função $f(x, y)$:

$$f(x, y) = 2(-1 + 6h)(1 - 6h) + 2(-1 + 6h) - (-1 + 6h)^2 - 2(1 - 6h)^2 = g(h). \quad (38)$$

Aplicando a distributiva entre todos os termos em parênteses e reorganizando em potência de h temos o seguinte polinômio para $g(h)$:

$$g(h) = -180h^2 + 72h - 7. \quad (39)$$

O próximo passo é definir a distância h^* que devemos percorrer no sentido \hat{e}_h ao longo do corte da nossa montanha $f(x, y)$ representado por $g(h)$. No método do aclive máximo a escolha consiste em caminhar até um máximo local em $g(h)$. Para isso, precisamos encontrar o valor h^* que faça com que $g'(h^*) = 0$. Derivando a equação (39), igualando à zero e isolando o valor de h , obtemos $h^* = 0.2$. Nesse momento voltamos às equações paramétricas e atualizamos o valor do próximo ponto x, y a partir do ponto inicial x_0, y_0 :

$$x = -1 + 0.2 \left(\frac{\partial f}{\partial x} \right)_{x_0, y_0} = 0.2 \quad \text{e} \quad y = 1 + 0.2 \left(\frac{\partial f}{\partial y} \right)_{x_0, y_0} = -0.2. \quad (40)$$

A figura (12) mostra esse primeiro passo da caminhada. Note que já estamos chegando mais perto do ponto crítico. O próximo passo consiste em descobrir a nova curva $g(h)$ a partir desse ponto de parada e repetir o processo. Faça isso em casa e veja como a caminhada evolui na direção do ponto crítico.

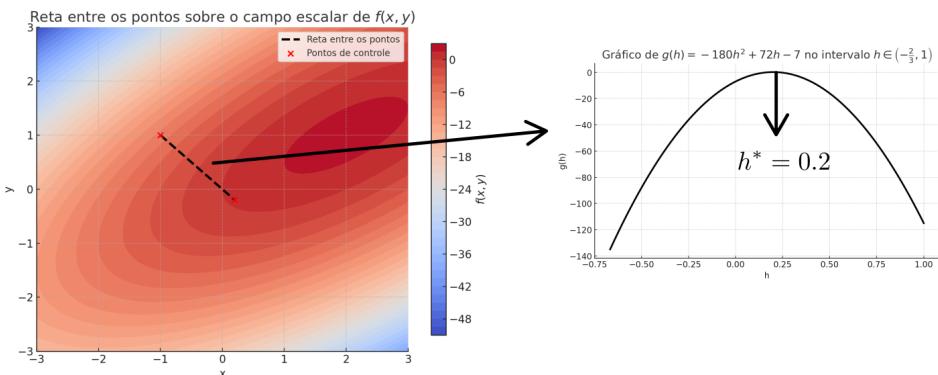


Figura 12: Primeiro passo no exemplo do método do aclive máximo para a função objetivo $f(x, y) = 2xy + 2x - x^2 - 2y^2$ com o detalhe da curva $g(h)$ ao lado.

8.4 O método dos gradientes conjugados

Se você fez o passo seguinte do exemplo dado na seção anterior para otimização da função $f(x, y) = 2xy + 2x - x^2 - 2y^2$ pelo método do aclive máximo e chegou a marcar no mapa essa nova coordenada, deve ter percebido que ela parece fazer um ângulo de 90 graus com relação à caminhada do passo anterior. De fato, quando utilizamos o método do aclive máximo, teremos sempre um ângulo reto sendo formado por dois passos sucessivos no processo de busca, o que pode levar a um processo lento de encontro do ponto ótimo da função objetivo. Esse processo de busca em zigue-zague pode demorar muito para encontrar o ponto crítico dependendo das características da função que desejamos otimizar. Por conta disso, alguns métodos mais robustos podem ser aplicados ao processo de otimização, como é o caso do método dos gradientes conjugados.

Mas antes de adentrarmos na ideia do método dos gradientes conjugados, vamos demonstrar aqui a ortogonalidade entre direções sucessivas de busca associada à aplicação do método do aclive máximo. Para isso, considere que a nossa função objetivo é uma função quadrática, do tipo

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \mathbf{x}, \quad (41)$$

em que $A \in \mathbb{R}^{n \times n}$ é uma matriz simétrica definida positiva e $\mathbf{b} \in \mathbb{R}^n$. A ideia de utilizarmos aqui uma função objetivo quadrática se deve ao fato de que parábolas são ótimas candidatas a funções aproximadoras do comportamento de funções nas vizinhanças de pontos ótimos. Para esse cenário, se tomamos o gradiente da função f , podemos mostrar que

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = 0, \quad (42)$$

ou seja, o problema de otimização de uma função objetivo quadrática é equivalente ao problema de resolvemos um sistema linear. Por conta disso, o método dos gradientes conjugados é não só um método para solução de problemas de otimização multidimensional sem restrições, como também é um método aplicado à solução de sistemas lineares. Podemos então afirmar que o ponto \mathbf{x}^* que minimiza f é a solução do sistema linear:

$$A\mathbf{x} = \mathbf{b}. \quad (43)$$

Agora, dado um ponto inicial \mathbf{x}_0 , o método do aclive máximo caminha no sentido de busca do ponto ótimo por meio da relação:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \quad (44)$$

em que α_k equivale ao nosso h^* no passo k e pode ser dado por

$$\alpha_k = \arg \min_{\alpha} f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)). \quad (45)$$

Mas nós sabemos que

$$\nabla f(\mathbf{x}_k) = A\mathbf{x}_k - \mathbf{b}, \quad \text{e} \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \quad (46)$$

de tal sorte que

$$\nabla f(\mathbf{x}_{k+1}) = A\mathbf{x}_{k+1} - \mathbf{b} = A(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)) - \mathbf{b} = \nabla f(\mathbf{x}_k) - \alpha_k A \nabla f(\mathbf{x}_k). \quad (47)$$

Para mostrarmos a ortogonalidade entre $\nabla f(\mathbf{x}_k)$ e $\nabla f(\mathbf{x}_{k+1})$ precisamos calcular o seguinte produto interno:

$$\begin{aligned} \nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_k) &= (\nabla f(\mathbf{x}_k) - \alpha_k A \nabla f(\mathbf{x}_k))^T \nabla f(\mathbf{x}_k) \\ &= \|\nabla f(\mathbf{x}_k)\|^2 - \alpha_k \nabla f(\mathbf{x}_k)^T A \nabla f(\mathbf{x}_k) \end{aligned} \quad (48)$$

Para encontrarmos o valor ótimo de α_k substituimos a expressão para \mathbf{x}_{k+1} obtida em (44) na expressão para a função objetivo quadrática, definida em (41), derivamos o resultado com relação à α_k , igualamos à zero e descobrimos o valor de α_k que minimiza o gradiente no caminho da direção de busca. Esse valor é dado por

$$\alpha_k = \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T A \nabla f(\mathbf{x}_k)}. \quad (49)$$

Substituindo a equação (49) em (48) obtemos:

$$\nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_k) = \|\nabla f(\mathbf{x}_k)\|^2 - \frac{(\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k))^2}{\nabla f(\mathbf{x}_k)^T A \nabla f(\mathbf{x}_k)} = 0, \quad (50)$$

provando assim a ortogonalidade entre direções sucessivas de busca quando utilizamos o método do acente máximo. A fim de resolver esse problema de busca em zigue-zague, o método dos gradientes conjugados propõe que duas direções sucessivas de busca \mathbf{p}_i e \mathbf{p}_j devam ser mutuamente conjugadas entre si para fugir desse problema. Dizemos então que essas duas direções são chamadas de **conjugadas com respeito à matriz A** se

$$\mathbf{p}_i^T A \mathbf{p}_j = 0 \quad \text{para } i \neq j \quad (51)$$

Essa propriedade é mais geral do que o conceito de ortogonalidade euclidiana e está relacionada à curvatura da função objetivo quadrática

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \cdot \mathbf{x}$$

Mas como essa imposição de conjugação mútua entre direções sucessivas de busca leva a uma busca mais retilínea em direção ao ponto ótimo do que aquela em zigue-zague observada no método do acente máximo? Para responder a essa pergunta, precisamos definir aqui como o método dos gradientes conjugados é construído para além da restrição de conjugação mútua. Nesse método, a direção de busca é atualizada pela relação:

$$\mathbf{p}_{k+1} = -\nabla f_{k+1} + \beta_k \mathbf{p}_k \quad (52)$$

com β_k sendo um parâmetro minimizador. Aqui entra uma característica importante do método dos gradientes conjugados. Podemos escolher um parâmetro minimizador de diferentes maneiras, ou seja, existem variantes do método dos gradientes conjugados, como por exemplo, o algoritmo de Fletcher-Reeves, no qual

$$\beta_k = \frac{\|\nabla f_{k+1}\|^2}{\|\nabla f_k\|^2}.$$

Vamos verificar agora então como essa nova forma de caminhar rumo a um ponto crítico impacta na ortogonalidade entre direções sucessivas de busca. Para isso precisamos checar se

$$\mathbf{p}_{k+1}^T \mathbf{p}_k \neq 0.$$

Essa verificação é razoavelmente simples, para isso basta substituirmos a expressão das direções de busca do método dos gradientes conjugados para obter:

$$\mathbf{p}_{k+1}^T \mathbf{p}_k = (-\nabla f_{k+1} + \beta_k \mathbf{p}_k)^T \mathbf{p}_k = -\nabla f_{k+1}^T \mathbf{p}_k + \beta_k \|\mathbf{p}_k\|^2$$

Para a ortogonalidade ser verdadeira, precisaríamos que

$$\beta_k = \frac{\nabla f_{k+1}^T \mathbf{p}_k}{\|\mathbf{p}_k\|^2}, \quad (53)$$

o que significa que qualquer valor de β_k definido de maneira diferente ao que expressa a equação (53) implicaria na não-ortogonalidade entre duas direções sucessivas de busca. É fácil verificar por exemplo que a variante de Fletcher-Reeves para o parâmetro minimizador β_k leva a um caminhar não ortogonal entre direções sucessivas ao longo da busca do ponto ótimo. Mas uma pergunta continua em aberto: como escolher um valor de β_k que atenda à exigência de conjugação mútua entre direções sucessivas de busca com relação à matriz A ? Para isso, precisamos testar a conjugação mútua com base nas definições do método:

$$\mathbf{p}_{k+1}^T A \mathbf{p}_k = (-\nabla f_{k+1} + \beta_k \mathbf{p}_k)^T A \mathbf{p}_k = -\nabla f_{k+1}^T A \mathbf{p}_k + \beta_k \mathbf{p}_k^T A \mathbf{p}_k. \quad (54)$$

A partir da equação (54) é possível descobrir o valor de β_k que garante conjugação mútua, dado por:

$$\beta_k = \frac{\nabla f_{k+1}^T A \mathbf{p}_k}{\mathbf{p}_k^T A \mathbf{p}_k}. \quad (55)$$

Neste caso, ao utilizarmos o valor de β_k definido em (55) obtemos

$$\mathbf{p}_{k+1}^T A \mathbf{p}_k = 0 \Rightarrow \text{direções são } A\text{-conjugadas.}$$

O problema de utilizarmos esse valor de β_k dentro de um código numérico voltado à otimização de uma função multidimensional num contexto sem restrições é que a cada passo precisamos realizar o produto da matriz dos coeficientes pela direção de busca e esse procedimento pode se tornar computacionalmente caro quando as dimensões do espaço de busca aumentam. Por conta disso, alguns autores desenvolveram fórmulas alternativas para a determinação do parâmetro minimizador que são mais baratas computacionalmente e satisfazem de forma aproximada a condição de conjugação mútua, como é o caso do algoritmo de Fletcher-Reeves.

No contexto da variante de Fletcher-Reeves do método dos gradientes conjugados temos que as direções de busca evoluem de acordo com a seguinte expressão

$$\mathbf{p}_{k+1} = -\nabla f_{k+1} + \beta_k \mathbf{p}_k \quad \text{e} \quad \beta_k = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}. \quad (56)$$

Para checarmos se as direções de busca são mutuamente conjugadas nesse método, precisamos avaliar se $\mathbf{p}_k^T A \mathbf{p}_{k+1} = 0$. Para isso, façamos

$$\mathbf{p}_k^T A \mathbf{p}_{k+1} = -\mathbf{p}_k^T A \nabla f_{k+1} + \beta_k \mathbf{p}_k^T A \mathbf{p}_k \quad \text{com} \quad \nabla f_{k+1} = \nabla f_k + \alpha_k A \mathbf{p}_k \quad (57)$$

logo

$$\mathbf{p}_k^T A \mathbf{p}_{k+1} = \mathbf{p}_k^T A \mathbf{p}_k - \alpha_k \mathbf{p}_k^T A^2 \mathbf{p}_k + \beta_k \mathbf{p}_k^T A \mathbf{p}_k, \quad (58)$$

com

$$\alpha_k = \frac{\nabla f_k^T \nabla f_k}{\mathbf{p}_k^T A \mathbf{p}_k}, \quad \beta_k = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}. \quad (59)$$

Se assumirmos que a função objetivo é quadrática e que as operações numéricas envolvendo a matriz A são computadas com aritmética exata (sem erros de arredondamento) é possível provar nesse caso que $\mathbf{p}_k^T A \mathbf{p}_{k+1} = 0$. Isso significa que para funções objetivo

não quadráticas ou em contextos nos quais o espaço de busca possui um número elevado de dimensões de tal sorte que erros numéricos de arredondamento possam se propagar, o método de Fletcher-Reeves pode não garantir a conjugação mútua entre direções sucessivas de busca. Esse fenômeno é chamado de degradação da conjugação mútua e pode levar a uma eventual busca em zigue-zague do nosso ponto crítico.

8.5 Programa para casa

Sua tarefa consiste em escrever um programa de computador que encontre o ponto crítico da seguinte função objetivo:

$$f(x, y) = 4x + 2y + x^2 - 2x^4 + 2xy - 3y^2. \quad (60)$$

Entretanto, a ideia aqui é implementar, validar e comparar a performance de diferentes métodos. Para isso, você deverá implementar os seguintes métodos:

1. Método da busca aleatória com espaço de busca igual a $[-3, 3]$;
2. Método do aclave máximo com aproximações iniciais $(x_0, y_0) = (0, 0)$;
3. Método dos gradientes conjugados (variante de Fletcher-Reeves) com aproximações iniciais $(x_0, y_0) = (0, 0)$;

Para o método de busca aleatória faça as seguintes análises:

- Varie o número n_p de pontos no domínio de busca dentro dos seguintes valores $n_p = [50, 200, 350, 500, 700, 1000]$. Para cada quantidade de n_p distribua esses pontos de forma regular/ordenada (num arranjo de malha de pontos) e de forma randômica. Compare o valor do ponto crítico encontrado para cada uma das formas de distribuição de pontos na forma de uma tabela;
- Avalie para cada forma de distribuição de pontos de busca como o erro relativo em relação ao cenário anterior (com menos pontos) varia em função de n_p ;
- Como o custo computacional responde a essa variação?

Para os métodos do aclave máximo e dos gradientes conjugados, plote no plano xy , juntamente com as curvas de nível o caminho de busca do ponto ótimo traçado para cada método. Você identifica a ortogonalidade dos passos sucessivos de busca para o método do aclave máximo? A aplicação do método dos gradientes conjugados quebra esse padrão em zigue-zague do caminho de busca? Quantos passos foram necessários para encontrar o ponto ótimo para cada método a partir do mesmo ponto inicial?

Finalmente, uma vez encontrado o ponto ótimo, use os sinais dos autovalores da matriz Hessiana da função objetivo no ponto crítico para determinar a natureza desse ponto. Esse é um ponto de máximo, mínimo, ponto de sela ou indeterminado? A sua análise corresponde com o aspecto gráfico da função $f(x, y)$?

9 Referências bibliográficas

1. S. C. Chapra, R. P. Canale. “Métodos Numéricos para Engenharia.” McGrawHill, 5a edição (2008): 1-825.