Implementando Algoritmo de Descida de Gradiente em C++

Danilo Sanchez Tuzita
(danilo_st@hotmail.com)

I. Resumo

Este é um trabalho que tem como objetivo implementar os algoritmos de Descida de Gradiente em *C++*. E analisar como se comportam duas funções distintas.

II. Introdução

A Descida do Gradiente é um dos algorítimos mais utilizados para otimizar redes neurais [1], pois é capaz de encontrar um mínimo de uma função qualquer.

III. Teoria

Para o entendimento desse trabalho é necessário conhecimentos básico de cálculo.

IV. Proposta e Implementação

Nesse trabalho o algorítimo foi desenvolvido de forma que ele aceite qualquer função e qualquer função de validação[1].

O pseudo-código 1 descreve de forma simples o algorítimo Descida de Gradiente. Seja x um valor inicial qualquer e seja f a função que será otimizada, calcula-se o $f'(x)$. Enquanto $f'(x)$ não for igual ou próximo de zero, x será $x - \beta \times f'(x)$, onde β é a taxa de aprendizado.

A ideia do algorítimo é usar a derivada da função f em certo ponto x para chutar um novo valor de x para que $f'(x)$ seja o mais próximo de zero possível, pois quando $f'(x) = 0$ a função está mudando de direção, indicando que foi encontrado um vale na função.

Por exemplo, seja $f(x) = x^3 - 2x^2 + 2$, $x = 1$ e $\beta = 1$, observando a figura 1 nota-se que $f'(1) = -1$, com isso podemos calcular o x da próxima iteração do algorítimo. $x = 1 - (1 \times -1) = 2$, também pode se observar que a tangente de $f(x)$ retorna o próximo x para o $\beta = 1$.

Algorithm 1 Pseudo-Código de Descida de Gradiente

```
let x_0 = valorInicial
let f_0 = f(x_0)
while f'_n ≠ 0 do
    let s_i = f'(x_i)
    x_{i+1} = x_i − β × s_i
    f'_{i+1} = f'(x_{i+1})
end while
```

Ao continuar esse *loop*, o x fará com que a função tenda ao vale mais próximo de x_0.

[1]Código no GitHub: https://github.com/danilotuzita/programacao-cientifica/blob/master/Aula4/Aula4/main.cpp.

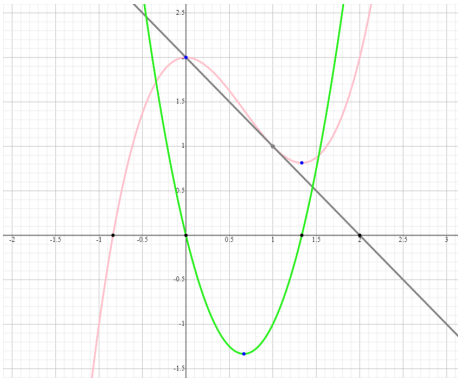

Fig. 1. Rosa: $f(x)$; Verde: $f'(x)$; Cinza: $tan(f(1))$

V. Resultados

Para esse trabalho, foi pedido para aplicar o algorítimo de Descida de Gradiente nas duas funções: $f(x) = x^2$, com $x_0 = 2$ e $f(x) = x^3 - 2x^2 + 2$, com $x_0 = 2$. Também foi pedido para se avaliar como a variação do parâmetro β entre 0.1 e 1 afeta o algorítimo. Para os testes, foi limitado a no máximo 1000 iterações por teste e uma taxa de erro de 0.0001, ou seja, se $|f'(x)| \leq 0.0001$ o algorítimo sairá do *loop* e retornará o x.

A. *função* $f(x) = x^2$

Pode-se observar na figura 2 que para um β muito próximo de 1 a quantidade de iterações é alto se comparado com os demais β, até estourando a quantidade máxima de iterações para o $\beta = 1$. Isso se deve ao fato do novo valor de x sempre ser baseado em 100% na derivada de $f(x)$, fazendo o novo x *overshootar* o objetivo.

Ao diminuir o β, a quantidade de iterações cai consideravelmente sendo seu vale $\beta = 0.5$ com apenas duas iterações para encontrar o x objetivo. Ao diminuir mais o seu valor, a quantidade de iterações começa a aumentar sendo com $\beta = 0.1$ aproximadamente 100 iterações. Isso ocorre pois o novo x receberá uma porção muito pequena de $f'(x)$ fazendo o x andar muito lentamente em direção ao objetivo.

B. *função* $f(x) = x^3 - 2x^2 + 2$

Observando-se a figura 3, nota-se uma curva atípica. Com exceção de um pico no $\beta \approx 0.48$ a quantidade de iterações se mantém abaixo de 50 iterações na maioria dos casos. Esse pico foi originado de um *sweet spot* que faz com que o x tenda ao vale de $x = \frac{4}{3}$.

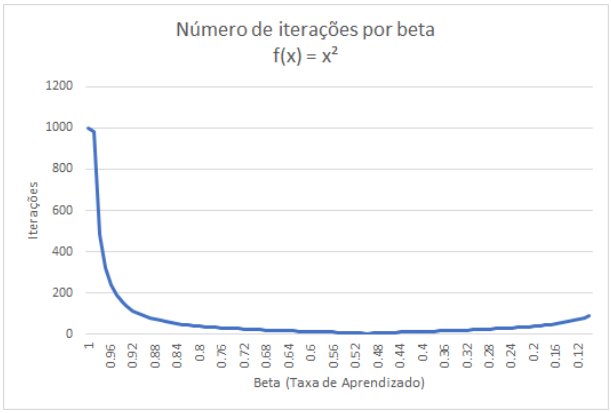

Fig. 2. Performance de β para $f(x) = x^2$

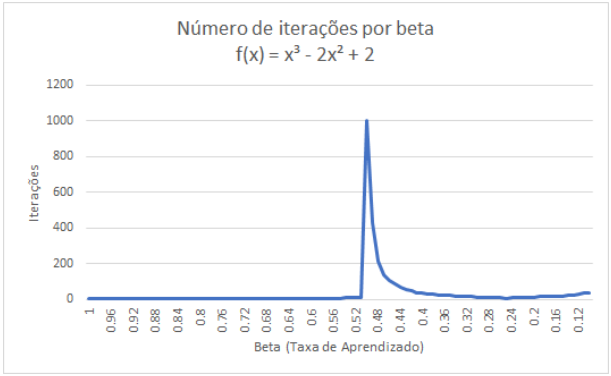

Fig. 3. Performance de β para $f(x) = x^3 - 2x^2 + 2$

Pode-se notar que, similarmente à função $f(x) = x^2$, com um β próximo de zero, o algorítimo começa a ter mais iterações para encontrar o x, pelo mesmo motivo da função anterior.

VI. Conclusão

O algorítimo Descida de Gradiente é um algorítimo muito relevante na atualidade pelo seu uso em redes neurais e outros algorítimos de inteligência artificial. Esse trabalho demonstrou a importância do parâmetro β ou taxa de aprendizado para a eficiência do algorítimo e sua assertividade.

Nesse trabalho para cada iteração do algorítimo o β se manteve estático. Para trabalhos futuros, pode ser estudado formas de alterar dinâmicamente esse parâmetro.

References

[1] S. Ruder, "An overview of gradient descent optimization algorithms," *ArXiv*, vol. abs/1609.04747, 2016.