Implementação e Análise do Algoritmo Descida de Gradiente

Lucas Fontes Buzuti
Departamento de Engenharia Elétrica
Centro Universitário FEI
São Bernardo do Campo-SP, Brasil
lucas.buzuti@outlook.com

***Resumo*—Esse artigo tem uma finalidade acadêmica na compreensão e implementação do algoritmo descida de gradiente. O algoritmos foi analisado e comparado a partir de duas funções adotadas e variando a taxa de aprendizado, no intuito de encontrar o mínimo local. Na implementação foi utilizada a programação orientada a objeto na linguagem *C++*, tendo em foco a otimização e a velocidade na execução do algoritmo.**

***Index Terms*—algoritmo descida de gradiente, otimização, programação orientada a objeto, C++**

I. Introdução

Esse artigo tem em seu objetivo a compreensão e implementação do algoritmo descida de gradiente (gradient descent). Além da implementação, uma análise comparativa do algoritmo é proposta. A implementação tem como alvo a utilização da linguagem *C++*, pois é uma linguagem focada na otimização e na velocidade da execução de algoritmos.

A descida de gradiente é um método numérico de otimização iterativa de primeiro ordem, no qual visa encontrar o mínimo de uma função seguindo a direção do gradiente e é necessário que a função seja computável e diferenciável.

II. Teoria

Diversos algoritmos, tais como algoritmos de aprendizado de máquina (machine learning) e algoritmos de aprendizado profundo (deep learning), utilizam-se de alguma otimização. A tarefa de otimização refere-se em uma minimização ou maximização de alguma função $f(x)$ alterando x. Como a descida de gradiente busca o mínimo da função, então, a otimização será na minimização nessa função. Correlacionando essa função com algoritmos de aprendizado de máquina e aprendizado profundo, podem ser chamadas de função de custo, função de perda ou função de erro (cost function, loss function ou error function) [1].

Geralmente para representar o valor mínimo ou máximo de uma função, usa-se x^*. Por exemplo, pode-se dizer $x^* = arg\ min\ f(x)$.

Dado uma função $y = f(x)$, no qual x e $y \in \mathbb{R}$. A derivação dessa função é representada como: $f^{'}(x)$ ou $\frac{dy}{dx}$, em que $f^{'}(x)$ fornece a inclinação de $f(x)$ dado um ponto x. Em outras palavras, especifica como dimensionar uma pequena alteração na entrada para obter a alteração correspondente na saída. A derivada é, portanto, útil para minimizar uma função, porque ela diz como alterar x para fazer uma pequena melhoria em y. Sendo assim, pode-se reduzir $f(x)$ movendo x em pequenos passos com o sinal oposto da derivada, assim chamando de gradiente descendente, proposto em 1847 pelo Cauchy [1].

Por definição um mínimo local é um ponto, em que $f(x)$ é menor do que todos os pontos vizinhos, portanto, não é mais possível diminuir $f(x)$ fazendo passos finitos. Portanto, para determinar o valor mínimo da função, segue-se a direção do gradiente:

$$x_{n+1} = x_n - \beta \nabla f(x_n), \tag{1}$$

no qual β é a taxa de aprendizado e é necessário que a função seja computável e diferenciável em x_n, portanto, se for possível computar $f(x)$ e $\frac{df}{dx}$ para qualquer valor de x, pode-se sempre seguir o gradiente na direção em que o valor se aproxima de zero ou seja zero.

III. Proposta e Implementação

Esse artigo propõe utilizar a linguagem *C++* para a implementação[1] do algoritmo descida de gradiente. Além das implementações, uma análise comparativa entre algumas taxas de aprendizado são propostas. Esta implementação foi feita em um computador com sistema operacional *Linux* e com o compilador *GCC*, a versão da linguagem utilizada foi a *C++11*. Para desenvolver o algoritmo, foi utilizada a programação orientada a objeto (POO), onde visa a construção de classes e métodos.

Foi codificada uma classe denominada *GradDesc* do Algoritmo 1. Para avaliar a classe, computou-se duas funções: x^2 e $x^3 - 2x^2 + 2$, sendo $x_0 = 2$ em ambas as funções, e verificou-se o acontecimento ao mudar a taxa de aprendizado.

Algorithm 1 Gradient Descent Algorithm

$x_0 = initial\ value$
$f_0 = f(x_0)$ → evaluate f on x

while $f_n^{'} \neq 0$ **do**
  $s_i = \frac{df}{dx}(x_i)$ → slope compute
  $x_{i+1} = x_i - \beta s_i$ → move in x
  $f_{i+1}^{'} = f^{'}(x_{i+1})$ → evaluate f on x_{i+1}
end while

[1] https://github.com/buzutilucas/scientific-programming/tree/master/Ex04

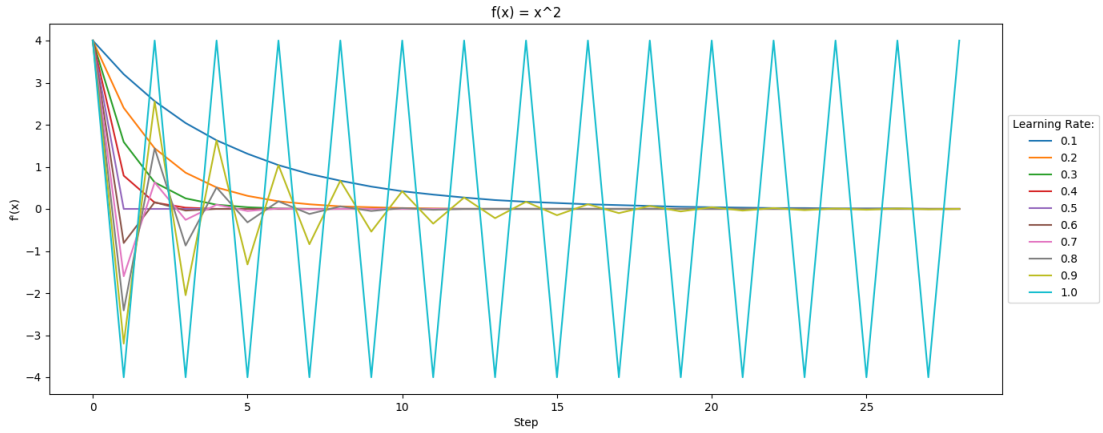

Figura 1. Gráfico da taxa de aprendizado aplicado a função $f(x) = x^2$ e $x_0 = 2$.

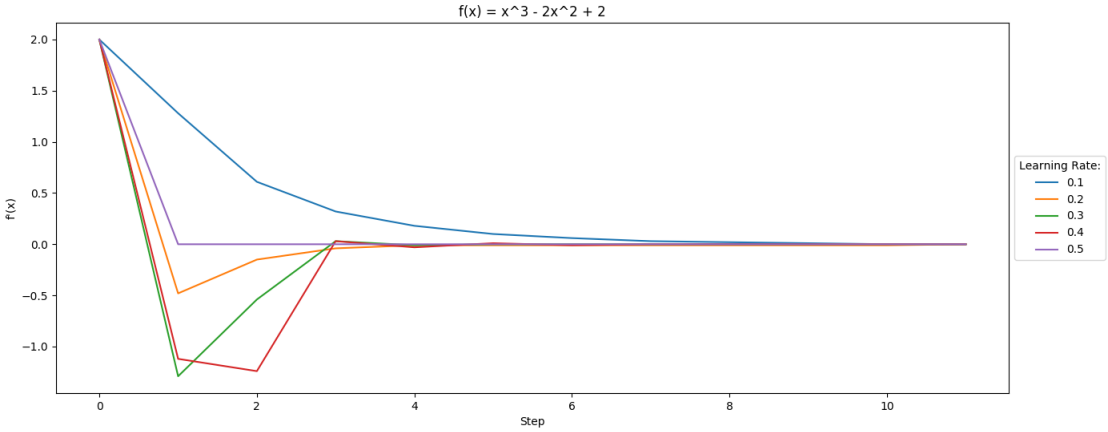

Figura 2. Gráfico da taxa de aprendizado aplicado a função $f(x) = x^3 - 2x^2 + 2$ e $x_0 = 2$.

IV. Experimentos e Resultado

Para analisar o algoritmo descida de gradiente, foi computado duas funções: x^2 e $x^3 - 2x^2 + 2$, sendo $x_0 = 2$ em ambas as funções, e modificou-se a taxa de aprendizado de 0.1 até 1.0 para verificar o acontecimento dessas variações. Nas Figuras 1 e 2 são demostradas as variações das taxas de aprendizado aplicadas nas duas funções propostas.

Analisando a Figura 1 em relação as taxas de aprendizado utilizadas, certifica-se que ao utilizar uma taxa 1.0 o gradiente não consegue convergir para um mínimo local, pode-se correlacionar esse fato com a movimentação do x, no qual x sempre estará nos extremos na função em relação a x_0 e não importando o seu valor, mas $x_0 > 0$ e $\in \mathbb{R}$. Em contrapartida, as outras taxas de aprendizado converge para um mínimo local, porém, a taxa que converge rapidamente a esse mínimo local é a taxa 0.5. Visualiza-se essa mesma convergência da taxa 0.5 na Figura 2. As taxas 0.6, 0.7, 0.8, 0.9 e 1.0 não convergem para uma mínimo local da função na Figura 2, levando o gradiente para os extremos da função, no qual esses extremos são $(-\infty, +\infty)$, por razões desses extremos fica impossível de visualizar graficamente o comportamento, mas se assemelha ao comportamento da função da Figura 1 com a taxa de aprendizado 1.0.

V. Trabalhos Correlatos

Diversos algoritmos, tais como algoritmos de aprendizado de máquina e algoritmos de aprendizado profundo, utilizam-se da descida de gradiente para minimizar a função de erro. Algoritmos de otimização avançado para aprendizado profundo tem em sua base a descida de gradiente, tais como Adadelta [2], RMSprop [4], Adam [3] e outros.

VI. Conclusão

Nesse artigo pode-se compreender e analisar o algoritmo descida de gradiente para determinar o mínimo de duas funções com taxas de aprendizado de 0.1 até 1.0. Conclui-se que, quanto menor for a taxa de aprendizado o algoritmo encontrará um mínimo local, porém, maior será o tempo para encontrar o mesmo. Em contrapartida, se a taxa de aprendizado

for muito maior, o algoritmo ficará em um loop entre as extremidades da função.

REFERÊNCIAS

[1] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning http://www.deeplearningbook.org. MIT Press, Cambridge, MA, 2016.

[2] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701, 2012.

[3] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, pages 1–13, 2015.

[4] TIELEMAN, Tijmen; HINTON, Geoffrey. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. University of Toronto, Technical Report, 2012.