Estudo de caso usando algoritmos de otimização

Carlos Estevão Bastos Sousa, Ramiro Francisco Bezerra dos Santos
Instituto Federal de Educação Ciência e Tecnologia do Ceará

Resumo - Os estudos sobre Aprendizado de Máquina datam a partir da década de 1950. Estes estudos favoreceram o surgimento de vários algoritmos de otimização onde um dos principais objetivos destes algoritmos é justamente a minimização do valor esperado de certa função. Existe uma variedade de algoritmos que buscam resolver estes tipos de problemas e dentre os principais algoritmos usados em problemas de minimização foram avaliados neste trabalho os seguintes: Gradient Descent, Steepest Descent e Newton's Method. Todos foram submetidos a mesma função objetivo onde foram analisados do ponto de vista de tempo de execução, quantidade de iterações e desempenho.

Palavras-chave - Aprendizado de Máquina; Otimização; Programação Não-Linear.

I. INTRODUÇÃO

Este trabalho propõe-se a apresentar e analisar alguns dos algoritmos de otimização mais usados para minimização de funções complexas. O estudo foi organizado da seguinte forma: na seção II é descrito o Gradiente Descendente (Gradient Descent), na seção III é abordado o Descida mais íngreme (Steepest Descent), enquanto Método de Newton (Newton's Method) é apresentado na seção IV. Na seção V são apresentados os resultados obtidos para as diversas simulações computacionais realizadas e, por fim, serão apresentadas as considerações finais sobre o estudo de caso na seção VI.

II. GRADIENT DESCENT

O algoritmo do Gradiente Descendente (GD) é usado para minimização de funções. Dada uma função definida por um conjunto de parâmetros, o gradiente descendente começa com um conjunto inicial de valores de parâmetros e, de forma iterativa, avança em direção a um conjunto de valores de parâmetros que minimizam a função. Essa minimização iterativa é obtida usando a função descrita abaixo. (Boyd, 2004)

$$\Delta x = -\nabla f(x), \quad (1)$$

Cada passo da iteração pode ser visto como um problema de minimização. A direção neste método será sempre em busca ao valor negativo do gradiente da função.

III. STEEPEST DESCENT

O método de Descida mais íngreme (SD) tem como objetivo minimizar a função cuja a direção neste método segue a (2) onde a direção da descida depende diretamente da maior derivada direcional. (Boyd, 2004)

Diferente do GD, o SD pode asumir um valor não negativo de gradiente no caso de norma L1.

$$\Delta x_{sd} = \|\nabla f(x)\| * \Delta x_{nsd}\ , \quad (2)$$

$$\Delta x_{nsd} = \text{argmin}\{\nabla f(x)^T \text{ v} \mid \ ||\text{v}|| \leq 1\}\ , \quad (3)$$

IV. NEWTON'S METHOD

O método de Newton (NM) tem como ideia básica substituir a função não linear por sua aproximação linear. De forma iterativa e com base na função 4

$$\text{x}^{(\text{k})} = \text{x}^{(\text{k}-1)} - (\nabla^2 \text{f}(\text{x}^{(\text{k}-1)}))^{-1} \nabla \text{f}\left(\text{x}^{(\text{k}-1)}\right), \text{k} = 1, 2, 3, \ldots, (4)$$

No NM para se determinar a direção de busca é necessário o cálculo da derivada segunda da função objetivo e da resolução de um sistema de equações, o que requer alto custo computacional. (Ribeiro e Karas, 2013).

V. SIMULAÇÕES COMPUTACIONAIS

Para realização das simulações foi feito uso do software Matlab/Simulink 2017a. Todos os algoritmos foram submetidos a mesma função objetivo abaixo.

$$f(x1, x2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} - e^{-x_1-0.1}\ , \quad (5)$$

Os algoritmos foram iniciados com os mesmos parâmetros para possibilitar um analise equilibrada dos resultados. Como medida de desempenho dos algoritmos foi adotado o tempo transcorrido até o término da execução para cada instância testada. Apenas o tempo em que a CPU ficou efetivamente ocupada pelo processamento das operações dos algoritmos foram contabilizados. (i.e. CPU *time*) A regra de convergência usada foi a

$$\|g_k\|_\infty \leq \epsilon\ , \quad (6)$$

com ϵ assumindo valores no conjunto $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

No algoritmo GD o comportamento pode ser verificado na Fig. 1. Onde para este caso foi usado o método de busca linear.

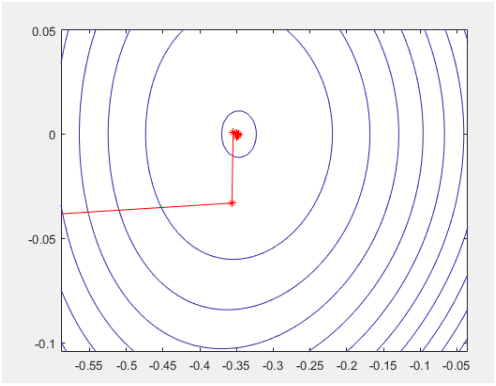

Fig. 1. Gradient Descent - Exact Line Search.

O mesmo algoritmo GD usando a busca backtracking pode ser verificado pela Fig. 2.

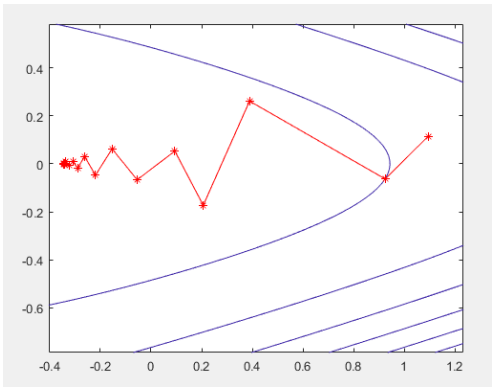

Fig. 2. Gradient Descent - Backtracking Search.

No algoritmo SD o comportamento pode ser verificado na Fig. 3. Onde para este caso foi usado o método de busca linear.

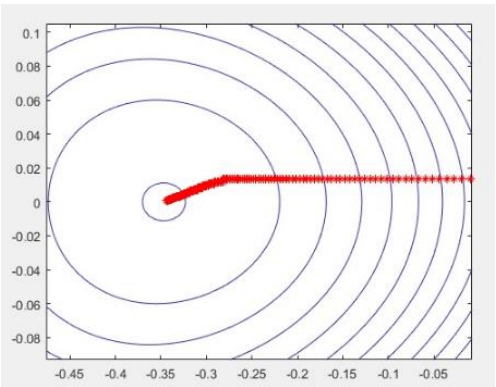

Fig. 3. Steepest Descent - Exact Line Search.

O mesmo algoritmo GD usando a busca backtracking pode ser verificado pela Fig. 4.

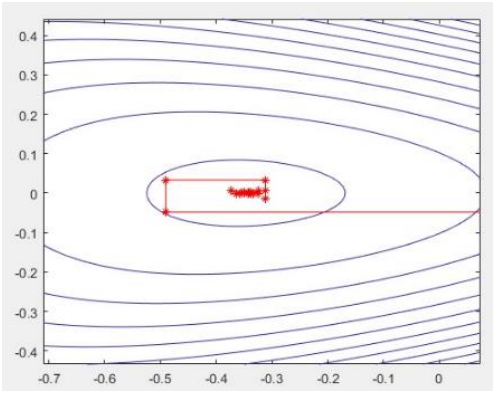

Fig. 4. Steepest Descent - Backtracking Search.

No algoritmo NM o comportamento pode ser verificado na Fig. 5. Onde para este caso foi usado o método de busca backtracking.

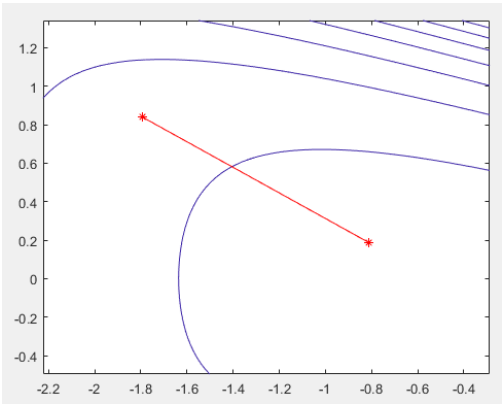

Fig. 5. Newton's Method - Backtracking Search.

Os resultados obtidos nas simulações realizadas com todos os algoritmos estão representados na Tabela I. É necessário salientar que foi escolhido como chute inicial x0 = [1,1]. Na tabela a primeira coluna representa cada algoritmo através de sua sigla. As colunas restantes representam a quantidade de iterações necessárias para a convergência por unidade de tempo em segundos.

TABELA I

ANALISE DOS RESULTADOS

Soma de Tempo(segundos)	Rót															
Rótulos de Linha	5	6	8	11	13	15	16	17	20	21	25	164	270	379	487	Total Geral
GD				4,13		3,75	1,89		4,8	2,67	9,23					26,4686
Backtracking				2,77		3,75			4,8		6,68					17,9957
10^-1				2,77												2,7721
10^-2						3,75										3,7464
10^-3									4,8							4,795
10^-4											6,68					6,6822
Linear				1,36			1,89			2,67	2,55					8,4729
10^-1				1,36												1,3602
10^-2							1,89									1,8939
10^-3										2,67						2,6735
10^-4											2,55					2,5453
NM	1,96	2,46														4,4215
Backtracking	1,96	2,46														4,4215
10^-1	0,968															0,9684
10^-2	0,989															0,9892
10^-3		1,09														1,0856
10^-4		1,38														1,3783
SD			1,8	2,66			3,24		3,9			20	33,5	47,5	60,8	173,342
Backtracking			1,8	2,66			3,24		3,9							11,5671
10^-1			1,77													1,767
10^-2				2,66												2,6592
10^-3							3,24									3,2425
10^-4									3,9							3,8994
Linear												20	33,5	47,5	60,8	161,7749
10^-1												20,04				20,0373
10^-2													33,5			33,5022
10^-3														47,48		47,4839
10^-4															60,75	60,7515
Total Geral	1,96	2,46	1,8	4,13	2,66	3,75	1,89	3,24	4,8	6,57	9,23	20	33,5	47,5	60,8	204,2321

Fonte: Autor.

Na Fig.6 é possível verificar a diferença de acordo com o tipo de busca escolhido para cada algoritmo. A busca Linear exige mais iterações até a convergência. Contudo a busca Backtracking além de exigir menor número de iterações nos permite concluir que o NM obteve a melhor performance.

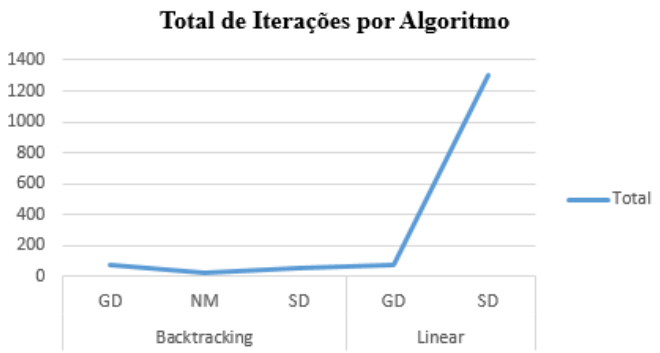

Fig. 6. Gráfico de Iterações.

É possível verificar através da Fig. 7 que entre os métodos que usam busca linear o GD apresentou o menor número de iterações necessárias para a convergência.

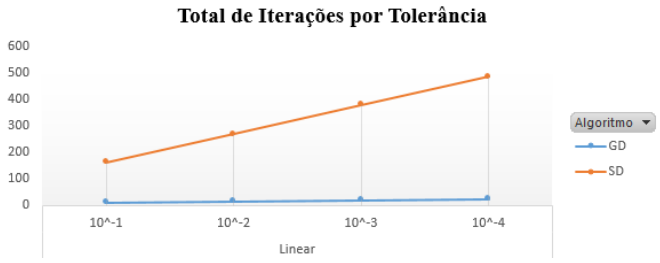

Fig. 7. Gráfico de Tolerância com busca Linear.

É possível concluir através da Fig. 8 que entre os métodos que usam busca backtracking o NM independente do grau de tolerância. Já entre os algoritmos que usam Gradiente o SD apresentou uma melhor performance com um menor número de iterações.

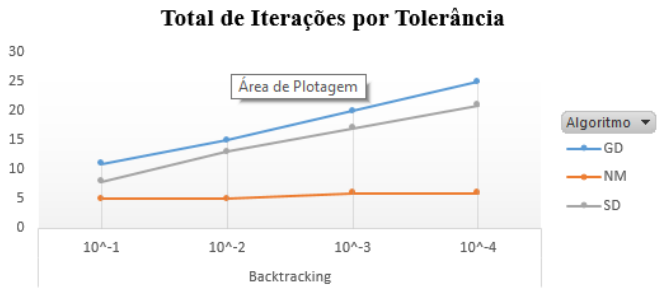

Fig. 8. Gráfico de Tolerância com busca Backtracking.

Com relação ao tempo de execução de cada algoritmo podemos verificar através da Fig. 9 que novamente o NM se destaca tendo o menor tempo. Seguido por GD com busca Linear, SD com Backtracking, GD com backtracking e por último o SD com busca Linear.

Fig. 9. Gráfico de Tempo de Execução por Algoritmo

VI. CONSIDERAÇÕES FINAIS

Este estudo de caso se propôs, como objetivo geral, analisar os algoritmos de minimização Gradiente Descendente, Descida mais íngreme e o Método de Newton com relação a performance e custo de processamento. Para que o trabalho não se limitasse à teoria, buscou, realizar diversas simulações com cada algoritmo.

Pode-se chegar, assim, a algumas conclusões: o NM apresentou como sendo o mais performático, porém é necessário salientar que para este algoritmo se requer um alto custo computacional. Em segundo lugar o SD apresentou uma boa escolha se usado com método de busca backtracking e em último, mas não menos importante ficou o GD com busca backtracking que pelo menor grau de complexidade pode atender bem com relação aos requisitos de processamento e tempo de execução.

REFERÊNCIAS

[1] Boyd, S.; Vandenberghe, L. Convex Optimization. Cambridge University Press, 2004.

[2] Chong, E. K. P., Zak, S. H., An introduction to optimization. Canada, Joh n Wiley & Sons, 2001.

[3] Mateus, G. R. & Luna, H. P. L. "Programação Nãolinear", Livro da V Escola de Computação, Belo Horizonte, 1986.

[3] Ribeiro, A. A.; Karas, E. W. Otimização Contínua: Aspectos teóricos e computacionais. São Paulo: Cengage Learning, 2013.