# Gradiente Descendente Estocástico com Nesterov *momentum*

Humberto da Silva Neto

4 de maio de 2018

## 1 Introdução:

Sutskever et al. (2013) introduziu uma variante do algoritmo de *momentum* que foi inspirado pelo método de gradiente acelerado de Nesterov (Nesterov, 1983, 2004). As regras de atualização são dadas por:

$$v \leftarrow \alpha v - \epsilon \nabla_\theta \Big[ \frac{1}{m} \sum_{i=1}^{m} L\big(f(x^{(i)}; \theta + \alpha v, y^{(i)}\big) \Big],$$
$$\theta \leftarrow \theta + v$$

aonde os parâmtros $\alpha$ e $\epsilon$ desenpenham um papel semelhante ao método do *momentum* padrão. A diferença entre o método de Nesterov e o padrão é aonde o gradiente é avaliado. No método de Nesterov, o gradiente é avaliada após a velocidade atual ser aplicada. Assim, pode-se interpretar o Nesterov *momentum* como uma tentativa de adicionar um fator de correção ao método padrão.

O algoritmo completo de Nesterov é apresentado logo abaixo:

**Algoritmo 1.** Gradiente Descendente Estocástico com Nesterov *momentum*

**Requer:** - Taxa de aprendizado $\epsilon$ - Parâmetro *momentum* $\alpha$ - Parâmetro inicial $\theta$ - Velocidade inicial $v$

***while*** critério de parada não chegar, ***do:***

Obtenha um *minibatch* de $m$ exemplos dos dados de treinamento $\{x^{(1)}, \cdots, x^{(m)}\}$ com suas respectivas *labels* $y^{(i)}$.

Aplique a atualização temporária: $\theta^{'} \leftarrow \theta + \alpha v$ Calcule o gradiente: $g \leftarrow \frac{1}{m} \nabla_\theta^{'} \sum_i L(f(x^{(i)}; \theta^{'}), y^{(i)})$

Calcule a atualização da velocidade: $v \leftarrow \alpha v - \epsilon g$

Aplique a atualização: $\theta \leftarrow \theta + v$

Termina o ***while***

## 2 Revisão:

Uma técnica clássica para acelerar o gradiente descendente estocástico (SGD) é o método de *momentum*, que acumula um vetor de velocidades em direções de reduções persistentes no objetivo e (Plaut et al., 1986). Esta propriedade permite que o *momentum* acumule velocidades em direções de baixa curvatura que persistem em várias iterações, levando a um progresso mais acelerado em tais direções quando comparado com o gradiente descendente. O Gradiente acelerado de Nesterov (Nesterov, 1983) é um método de otimização de primeira ordem que provou ter uma melhor garantia de taxa de convergência do que o gradiente descendente para funções convexas gerais com derivadas contínuas de Lipshitz (*O(1/T2) versus O(1/T)*).

O gradiente acelerado de Nesterov é bastante similar ao *momentum* padrão, diferindo em dois aspectos:

1. Prescreve uma fórmula particular para a taxa de aprendizagem $\epsilon$ e a constante do *momentum* $\mu$, que são necessárias para alcançar suas garantis teóricas para funções convexas suaves. Contudo, eles se mostraram muitos agressivos para funções não convexas.
2. O mecanismo de atualização para o vetor de velocidade $v$.

O momento padrão é dado pelas seguintes equações:

$$v_{t+1} = \mu v_t - \epsilon \nabla f(\theta_t + \mu v_t) \tag{1}$$

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{2}$$

tal que $\epsilon > 0$ é a taxa de aprendizagem, $\mu \in [0, 1]$ é a constante de *momentum*, e $\nabla f(\theta_t)$ é a estimativa imparcial do gradiente em $\theta_t$. A constante de *momentum* $\mu$ controla o "decaimento" do vetor velocidade $v$, e valores mais próximos de 1 resultam em informações de gradientes que persistem por mais iterações, o que geralmente implica em velocidades mais altas.

### 2.1 Cálculos:

O gradiente acelerado de Nesterov é um algoritmo iterativo originalmente derivado de gradientes não estocásticos. Ele é inicializado pela configuração de $k = 0, a_0 = 1, y_0 = \theta_{-1}, y_0$ para um ajuste de parâmetro arbitrário, $z$ para um ajuste de parâmetro arbitrário, e $\epsilon_{-1} = \|y_0 - z\| / \|\nabla f(y_0) - \nabla f(z)\|$. Em seguida, ele atualiza repetidamente seus parâmetros com as seguintes equações:

$$\epsilon_t = 2^{-i} \epsilon_{t-1}{}^* \tag{3}$$

$$\theta_t = y_t - \epsilon_t \nabla f(y_t) \tag{4}$$

$$a_{t+1} = \frac{(1 + \sqrt{4a_t^2 + 1})}{2} \tag{5}$$

$$y_{t+1} = \frac{\theta_t + (a_t - 1)(\theta_t - \theta_{t-1})}{a_t + 1} \tag{6}$$

* $i$ é o menor numero natural para o qual $f(y_t) - f(y_t - 2^{-i}\epsilon_{t-1}\nabla f(y_t)) \geq 2^{-i}\epsilon_{t-1}\frac{\|\nabla f(y_t)\|^2}{2}$

A taxa de aprendizado $\epsilon_t$ é adaptada para sempre ser menor que o coeficiente de Lipshitz $\nabla f$ na trajetória de otimização. Contudo, se o coeficiente de Lipshitz da derivada é conhecida como igual a $L$, então fazendo $\epsilon_t = 1/L$ para todo $t$ é suficiente para obter as mesmas garantias teóricas. Esse método para escolher a taxa de aprendizado assume que $f$ não é ruidoso e resultará em taxas de aprendizado muito grandes se o objetivo for estocástico.

Para entender a sequência $a_t$, nota-se que a função $x \to \left(1 + \sqrt{4x^2 + 1}\right)/2$ se aproxima rapidamente para $x \to x + 0.5$ por baixo quando $x \to \infty$, então $a_t \approx (t + 4)/2$ para $t$ grandes, e assim $(a_t - 1)/a_{t+1}$ (da equação 6) se comporta como $1 - 3/(t + 5)$.

Dessa forma, Sutskever (2013) define as seguintes equações:

$$v_t \equiv \theta_t - \theta_{t-1} \tag{7}$$

$$\mu_t \equiv (a_t - 1)/a_{t+1} \tag{8}$$

Combinando as equações 6 e 8, teremos:

$$y_t = \theta_{t-1} + \mu_{t-1}v_{t-1} \tag{9}$$

do qual podemos usar para reescrever a equação 4 como:

$$\theta_t = \theta_{t-1} + \mu_{t-1}v_{t-1} - \epsilon_{t-1}\nabla f(\theta_{t-1} + \mu_{t-1}v_{t-1}) \tag{10}$$

$$v_t = \mu_{t-1}v_{t-1} - \epsilon_{t-1}\nabla f(\theta_{t-1} + \mu_{t-1}v_{t-1}) \tag{11}$$

tal que a equação 11 é consequência da equação 7. Alternativamente, temos:

$$\theta_t = \theta_{t-1} + v_t \tag{12}$$

em que $\mu_t \approx 1 - 3/(t + 5)$. Nesterov(1983) mostra que se $f$ é uma função convexa com uma derivada de $L$-Lipshitz contínua, então o método acima satisfaz o seguinte:

$$f(\theta_t) - f(\theta_t^*) \leq \frac{4L\|\theta_{-1} - \theta^*\|}{(t + 2)^2} \tag{13}$$

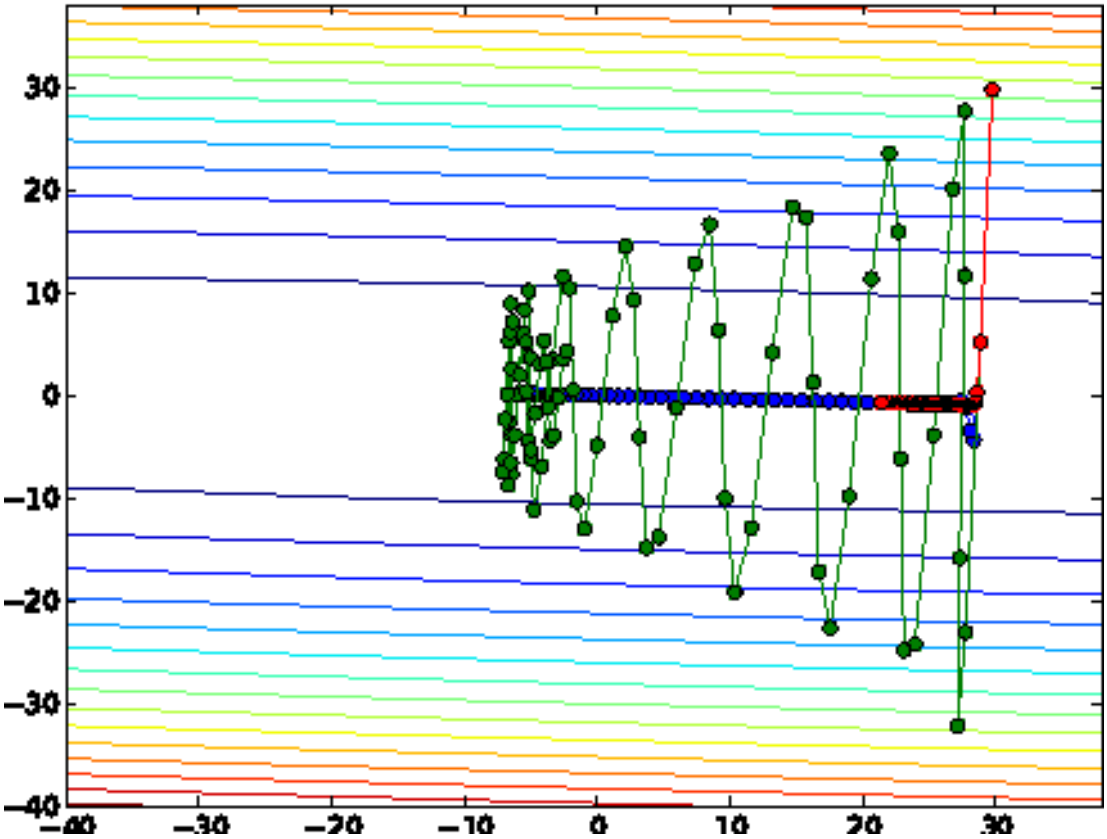
A dependência quadrática do número de iterações pode ser intuitivamente entendida considerando-se uma função linear unidimensional, em que a magnitude do $i$-ésimo passo seria proporcional a $i$. Assim, somente $O(\sqrt{N})$ dessas etapas são necessárias para percorrer $N$ unidades de distância.

## 3 Conclusão:

Quando o gradiente acelerado de Nesterov é apresentado nas equações 11 e 12, a semelhança com equações 1 e 2 que definem o momento padrão se torna aparente. Se ignorarmos o cronograma de momento proposto $\mu_t$ e o cronograma de taxa de aprendizado adaptável, então a principal diferença entre o momento e o gradiente acelerado de Nesterov é que o momento calcula o gradiente antes de aplicar a velocidade, enquanto o gradiente acelerado de Nesterov calcula o gradiente.

Essa diferença permite que o gradiente acelerado de Nesterov mude $v$ de uma maneira mais rápida e mais responsiva, permitindo que ele seja mais estável que o *momentum* em muitas situações, principalmente para altos valores de $\mu$. De fato, considere a situação em que a adição de $\mu v_t$ para $\theta_t$ resulta em um aumento indesejável no objetivo $f$. No caso do gradiente acelerado de Nesterov, a correção do gradiente para a velocidade $v_t$ é computada no ponto $\theta_t + \mu v_t$, e se $\mu v_t$ é de fato uma atualização fraca, $\nabla f(\theta_t + \mu v_t)$ apontará de volta para $\theta_t$ mais fortemente que o gradiente computado em $\theta_t$ (que é usado pelo momento), fornecendo assim uma correção maior e mais rápida para $v$. Por fim, a figura abaixo compara o desempenho entre o gradiente descendente, *momentum* e o gradiente acelerado de Nesterov.

**Figura 1.** Comparação entre Nesterov e *Momentum*




A trajetória do GD (vermelho), *momentum* (verde) e o gradiente acelerado de Nesterov (azul). O mínimo global é no centro da figura.

## Referências


Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O(1/sqr(k)). Soviet Mathematics Doklady, 27:372–376.

Nesterov, Y. (2004). Introductory lectures on convex optimization : a basic course. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London. 300

Plaut, D., Nowlan, S., and Hinton, G. E. (1986). Experiments on learning by back propagation. Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Sutskever, I. (2013). Training Recurrent Neural Networks. Ph.D. thesis, Department of computer science, University of Toronto.