Algoritmos de Otimização com Taxa de Aprendizado Adaptativa

João Pedro Daher Aranha

Histórico

Otimização é um ramo muito importante da matemática. Isso pode ser percebido por tantos grandes cientistas como Fermat, Newton, Euler, Gauss, Lagrange e Cauchy terem se dedicado a esse estudo nos séculos XVII à XIX.

Um dos primeiros métodos de otimização que se estuda no contexto do aprendizado de máquina é justamente um que foi inicialmente proposto por Cauchy em *Compte Rendu à l'Académie des Sciences of October 18, 1847 - Méthode générale pour la résolution des systèmes d'équations simultanées*: o gradiente descendente.

Desde essa época, muito se avançou nessa área e houve o advento dos computadores, que propiciou a busca de soluções para problemas antes inviáveis. Mesmo assim, alguns problemas continuam sendo tão custosos computacionalmente que motivam uma constante busca de algoritmos mais eficientes.

No campo de Machine Learning

Em aprendizado de máquinas, um parâmetro muito importante do modelo é a taxa de aprendizado (learning rate). No Gradiente Descendente é um número ajustado empiricamente observando as curvas de aprendizado conforme os testes são efetuados. Tanto no clássico como na versão estocástica são números pré-definidos. A diferença no caso estocástico é ser fornecida ao modelo uma sequência finita de taxas de aprendizado para serem utilizadas conforme o algoritmo é executado. Normalmente costuma-se usar uma taxa de aprendizado linear decrescente.

Além do Gradiente Descendente Estocástico estão, nessa categoria de algoritmos básicos, conforme classificação do livro Deep Learning Book (Goodfellow et al.), o Momentum (Polyak, 1964) e o Nesterov Momentum (Sutskever et al.). Esses dois últimos propõem considerar no cálculo o acúmulo dos gradientes passados por meio de uma média exponencial móvel e adicionando outro hiperparâmetro.

Revolução nos algoritmos de otimização

Uma característica marcante dos algoritmos mais recentes é poder calcular dinamicamente a taxa de aprendizado. Dessa forma, o modelo torna-se mais preciso em problemas de dimensionalidade mais alta, pois as diferentes variações em cada dimensão tem essa característica refletida em sua taxa de aprendizado. Isso representa uma enorme vantagem em relação ao paradigma anterior de uma única taxa de variação para todas as variáveis. O algoritmo **delta-bar-delta** (Jacobs,1988) fez um procedimento nesse sentido bem antes dos demais a serem citados.

AdaGrad

É um dos algoritmos de gradiente adaptativos mais famosos. Foi publicado em 2011 (Duchi et al.) e se propõe a obter informações sobre a geometria do problema desde etapas mais iniciais e fazer com que o cálculo dos gradientes leve em conta quais são as variáveis mais relevantes para obter a predição.

Como é feito o cálculo:

Necessário estabelecer:

- Parâmetro de aprendizado global ϵ
- Parâmetro inicial θ
- Constante pequena □, sugerido 10^{-7}
- Variável de acumulação **r** = 0

Enquanto não for satisfeito critério de parada:

Amostrar m exemplos do conjunto de teste $\{x^{(1)},x^{(2)},x^{(3)},...,x^{(m)}\}$ e seus $y^{(i)}$ correspondentes.

Calcular o gradiente: $g \leftarrow \frac{1}{m}\nabla\theta\sum L(f(x^{(i)};\theta), y^{(i)})$

Acumular em r o quadrado do gradiente: $r \leftarrow r + g\odot g$
(Obs: ⊙ é o operador elemento a elemento)

Obter a variação de θ: $\Delta\theta \leftarrow -\frac{\epsilon}{\sqrt{\delta+r}}\odot g$

Atualizar θ:$\theta \leftarrow \theta + \Delta\theta$

Fim

Fonte: Deep Learning Book - Goodfellow et al. 2016

Observa-se que se soma a cada etapa valores em r pode representar um problema. Com o crescimento não controlado do parâmetro r pode-se acabar tendo termo tendendo a zero limitando a aprendizagem do modelo.

RMS Prop

Em 2012, o artigo de Tieleman & Hinton propõe alterações no AdaGrad mudando o acúmulo dos gradientes em uma média móvel exponencial, assim aproximando-se em abordagem de algoritmos anteriores baseados em Momentum.

Como é feito o cálculo:

Necessário estabelecer:

- Parâmetro de aprendizado global ε, taxa de decaimento ρ
- Parâmetro inicial θ
- Constante pequena □, sugerido 10^{-6}
- Variável de acumulação **r** = 0

Enquanto não for satisfeito critério de parada:

Amostrar m exemplos do conjunto de teste $\{x^{(1)},x^{(2)},x^{(3)},...,x^{(m)}\}$ e seus $y^{(i)}$ correspondentes.

Calcular o gradiente: $g \leftarrow \frac{1}{m}\nabla\theta\sum L(f(x^{(i)};\theta),y^{(i)})$

Acumular em r o quadrado do gradiente: $r \leftarrow \rho r + (1-\rho)g\odot g$

Obter a variação de θ: $\Delta\theta \leftarrow -\frac{\epsilon}{\sqrt{\delta+r}}\odot g$

Atualizar θ:$\theta \leftarrow \theta + \Delta\theta$

Fim

Fonte: Deep Learning Book - Goodfellow et al. 2016

Adam

Unindo as vantagens de otimizadores anteriores o artigo ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION (Kingma & Ba, 2014) apresenta o algoritmo Adam (Adaptative Moment Estimation). Ele é visto por alguns como a combinação do RMSProp e do Momentum com algumas alterações.

Como apontado pelo livro do Ian Goodfellow, o Adam incorpora o conceito de momentum aplicando momentum aos gradientes escalados. Outra característica é utilizar um corretor de tendência nos momentos de primeira ordem. Esses conceitos normalmente ficam mais claros observando a implementação:

Como é feito o cálculo:

Necessário estabelecer:

- Tamanho de "passo" ϵ (sugestão padrão 0.001)
- Taxas de decaimento exponencial para os estimadores de momento ρ_1e ρ_2(0.9 e 0.999 respectivamente)
- Constante pequena □, sugerido 10^{-8}
- Parâmetro inicial θ
- Iniciar variáveis de momento **r** = 0 e **s** = 0.

Enquanto não for satisfeito critério de parada:

Amostrar m exemplos do conjunto de teste $\{x^{(1)},x^{(2)},x^{(3)},...,x^{(m)}\}$ e seus $y^{(i)}$ correspondentes.

Calcular o gradiente: $g \leftarrow \frac{1}{m}\nabla\theta\sum L(f(x^{(i)};\theta),y^{(i)})$

$t \leftarrow t + 1$

Atualizar a estimativa do primeiro momento: $s \leftarrow \rho_1 + (1 - p_1)g\odot g$

Atualizar a estimativa do segundo momento: : $s \leftarrow \rho_2 + (1 - p_2)g\odot g$

Corrigir a tendência no primeiro momento: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$

Corrigir a tendência no segundo momento: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$

Obter a variação de θ: $\Delta\theta \leftarrow -\epsilon\frac{\hat{s}}{\sqrt{\hat{r}+\square}}$

Atualizar θ:$\theta \leftarrow \theta + \Delta\theta$

Fim

Fonte: Deep Learning Book - Goodfellow et al. 2016

Variações desses métodos citados já existem como o Adadelta, variação mais robusta do AdaGrad, o AdaMax apresentado no mesmo artigo do Adam, o Nadam que é o Adam utilizando um outro algoritmo de Momentum: o Nesterov Momentum, o AMSGrad que procurou solucionar um problema do Adam, entre outros.

Com essa breve análise comparativa, podemos perceber como a área de otimização é vasta. Não se chegou a um algoritmo com performance superior em todos os casos. Portanto, cabe ao usuário experimentar, conhecer e saber escolher aquele que atende melhor à sua necessidade.

Fontes:

https://www.deeplearningbook.org/contents/optimization.html

https://algorithmia.com/blog/introduction-to-optimizers

https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827

https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be

https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

https://arxiv.org/pdf/1412.6980.pdf

https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf