Оптимальный Адаптивный Ускоренный Стохастический Градиентный Спуск

Сотников А.Д, Северилов П.А., Усманова А.А., Ивченков Я.П.

28 мая 2019 г.

Аннотация

Есть SGD. он нормас. если добавить акселерэйтед, то будет ваще бомба. аээээ, ено кроме того, сгд еще улучшает адаптив, не, адаптив улучшает сгд. и в статье решили такие: а мб бахнуть и то, и то одновременно?? ну и капсом ВУАЛЯ.не не. адаптив акселерейтед уже есть адам и ева (зачеркнутая) и амсград. в работе сравнивается реализация их с адамом и евой(зачеркнуто) и амсград. вот мотивация

1 Введение

Методы стохастического градиентного спуска SGD являются наиболее мощными инструментами оптимизации в обучении модели машинного обучения и глубокого обучения. Кроме того, методы тяжелого шарика и методы диагонального масштабирования (например, адаптивный градиент) являются двумя основными методами для улучшения SGD. Хотя эмпирические исследования продемонстрировали потенциальные преимущества сочетания этих двух методов, остается неизвестным, могут ли эти методы достичь оптимальной скорости сходимости для стохастической оптимизации. В этом проекте мы попытались реализовать новый класс адаптивных и ускоренных стохастических методов градиентного спуска и показать, что они демонстрируют оптимальную сложность выборки и итерации для стохастической оптимизации.

2 Постановка задачи

Рассмотрим следующую задачу выпуклой оптимизации – вместо градиента $\nabla f(x)$ оракул выдает его несмещённую оценку $\nabla_x f(x,\xi)$ с конечной дисперсией.

$$\mathbb{E}_\xi[\nabla_x f(x,\xi)] \equiv \nabla f(x),\ \mathbb{E}[\|\nabla_x f(x,\xi) - \nabla f(x)\|_2^2] < D \tag{1}$$

Конструкция минибатчинга в общем представлении:

$$\overset{r}{\nabla}_x f(x, \{\xi^l\}_{l=1}^r) = \frac{1}{r}\sum_{l=1}^{r} \nabla_x f(x,\xi^l) \tag{2}$$

Для выпуклых функций, обладающих липшецевым градиентом, т.е. таких, что $|\nabla f(x) - \nabla f(y)| \leq L\|x-y\|_2$, справедлива следующая оценка на количество обращений к оракулу при не малых D будет

$$N(\varepsilon) = O\left(\frac{DR^2}{\varepsilon^2}\right), \tag{3}$$

причём эта оценка остаётся верной и для ускоренных методов и не является улучшаемой.

Для сильно выпуклого случая данную оценку можно улучшить при помощи конструкции рестартов. В результате получим неулучшаемую оценку для сильно выпуклого случая:

$$N(\varepsilon) = O\left(\min\left\{\frac{DR^2}{\varepsilon^2}, \frac{D}{\mu\varepsilon}\right\}\right), \tag{4}$$

В случае, когда неизвестны значения L и D (считая при этом, что сделанные выше предположения выполнены), используется адаптивный метод подбора константы L.

В данной работе предлагается исследовать следующий адаптивный алгоритм (и его версию для реализации на практике):

Algorithm 1 Adaptive accelerated stochastic gradient (A2Grad) algorithm

Input: $x_0, \overline{x}_0, \gamma_k, \beta_k > 0$

1: **for** $k = 0,1,\ldots,K$ **do**
2: Update $x_k = (1-\alpha_k)\overline{x}_0 + \alpha_k x_k$
3: Sample ξ_k, compute $\underline{G}_k \in \nabla F(\underline{x}_k, \xi_k)$ and $\phi_k(\cdot)$, then update:
4: $x_{k+1} = \underset{x\in X}{argmin}\{\langle \underline{G}_k, x\rangle + \gamma_k D(x_k,x) + \beta_k D_{\phi_k(x_k,x)}\}$
5: $\overline{x}_{k+1} = (1-\alpha_k)\overline{x}_k + \alpha_k x_{k+1}$
6: **end for**

Output: $\overline{x}_{K+1}$

3 Описание методов

В статье предлагается изучение алгоритма 1, используя усовершенствования приведенные ниже:

Algorithm 2 A2Grad-inc: Adapdtive ASGD with incremental moving average (quadratic weight)

Input: $v_{-1} = 0$ and the rest of other parameters

The $k-$th step of Algorithm 1:

$v_k = k^2/(k+1)^2 v_{k-1} + \delta_k^2$

$h_k = \sqrt{v_k}$

Algorithm 3 A2Grad-uni: Adapdtive ASGD with uniform moving average

Input: $v_{-1} = 0$ and the rest of other parameters

The $k-$th step of Algorithm 1:

$v_k = v_{k-1} + \delta_k^2$

$h_k = \sqrt{v_k}$

Algorithm 4 A2Grad-exp: Adapdtive ASGD with exponential moving average

Input: $\tilde{v}_{-1} = 0$ and the rest of other parameters

The $k-$th step of Algorithm 1:

$$\tilde{v}_k = \begin{cases} \delta_k^2, & \text{if k=0} \\ \rho\tilde{v}_{k-1} + (1-\rho)\delta_k^2, & \text{otherwise} \end{cases}$$

$v_k = \max\{\tilde{v}_k, v_{k-1}\}$

$h_k = \sqrt{(k+1)v_k}$

Для практической реализации предложена эквивалентная запись алгоритма 1.

Algorithm 5 Adaptive ASGD rewritten

Input: the rest of other parameters

The $k-$th step of Algorithm 1:

$$x_{k+1} = x_k - \frac{1}{\gamma_k + \beta_k h_k} G_k$$

$$y_{k+1} = (1-\alpha_{k+1})y_k + \alpha_{k+1}x_{k+1} - \frac{(1-\alpha_{k+1})\alpha_k}{\gamma_k + \beta_k h_k} G_k$$

/*чето в статье там хреновенько написано, как практически реализовать прделгааемый алгоритм, в особенности непонятно, как подьирались параметры, включая L. поэтому были попытки сделать свою реализацию а2града. Также мы попробовали адасгд с нестеровым и сервером.но он тоже хуже был. без нестерова было круто, с ним не оч.*/

Параметры α_k, γ_k, δ_k подбирали по следующим формулам (учитывая леммы из статьи [6]):

$\alpha_k = 2/(k+2)$

$\gamma_k = 2L/(k+1)$

$\delta_k = G_k$ - $\frac{1}{k+1}\sum_{t=0}^{k} G_t$

4 Эксперименты

В данной работе мы сравниваем результаты работы оптимизаторов Adam, AMSGrad, accelerated SGD (описан в статье [5]) и adaptive SGD (Spokoiny's practical variant) с реализациями 3х оптимизаторов, предложенных в исходной статье - A2GradUni, A2GradInc, A2GradExp.

Тестирование:

Logistic Regression on MNIST

Two-layer neural network on MNIST

Deep neural network on CIFAR10

Выбор параматеров:

Параметры моделей подбирали как предложено в статье. В алгоритмах A2Grad-Uni, A2Grad-Inc, A2Grad-Exp наилучшее качество в наших экспериментах дали следующие значения: $\beta = 10$, $L = 10$, $\rho = 0.9$. Но все-таки A2Grad не дал лучший результат. Причиной тому может являться не очень подробный подбор параметров. Результаты экспериментов приведены на графиках.

Algorithm 6 Accelerated stochastic gradient descent

Input: Initial ω_0, short step δ, long step parameter $\kappa \geqslant 1$, statistical advantage parameter $\xi \leqslant \sqrt{\kappa}$

$\overline{\omega}_0 \longleftarrow \omega_0; t \longleftarrow 0$

$\alpha \longleftarrow 1 - \frac{0.49 \cdot \xi}{\kappa}$

3: while ω_t not converged do

$\overline{\omega}_{t+1} \longleftarrow \alpha \cdot \overline{\omega}_t + (1-\alpha) \cdot \left(\omega_t - \frac{\kappa \cdot \delta}{0.7} \cdot (\hat{\nabla}) f_t(\omega_t)\right.$

$\omega_{t+1} \longleftarrow \frac{0.7}{0.7+(1-\alpha)} \cdot \left(\omega_t - \delta \cdot \hat{\nabla} f_t(\omega_t)\right) + \frac{1-\alpha}{0.7+(1-\alpha)} \cdot \overline{\omega}_{t+1}$

6: $t \longleftarrow t+1$

Output: ω_t

Algorithm 7 AMSGrad

Input: $x_1 \in \digamma$, step size $\{\alpha_t\}_{t=1}^{T}$, $\{\beta_{1t}\}_{t=1}^{T}$, β_2

Set $m_0 = 0$, $v_0 = 0$ and $\hat{v_0} = 0$

for $t = 1,...,T$ **do**

$g_t = \nabla f_t(x_t)$

$m_t = \beta_{1m} m_{t-1} + (1 - \beta_{1t} g_t$

$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\hat{v_t} = max(\hat{v_{t-1}}, v_t)$ and $\hat{V_t} = diag(\hat{v_t})$

$x_{t+1} = \Pi_{\digamma, \sqrt{(\hat{V_t})}}(x_t - \alpha_t m_t / \sqrt{\hat{v_t}}$

end for

Algorithm 8 Adam

Input: step size α, decay rates β_1, β_2, $f(\Theta)$, Θ_0, $m_0 = 0$, $v_0 = 0$, $t = 0$

while θ_t not converged **do**

$t \longleftarrow t + 1$

$g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \longleftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$

$v_t \longleftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$

$\hat{m}_t \longleftarrow m_t / (1 - \beta_1^t)$

$\hat{v}_t \longleftarrow v_t / (1 - \beta_2^t)$

$\theta_t \longleftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$

end while

return θ_t

Algorithm 9 Adaptive Stochastic Gradient Method (Spokoiny's practical variant)

Input: lower estimate for the variance of the gradient $D_0 \leq D$,
accuracy $0 < \varepsilon < \frac{D_0}{L}$, starting point $x_0 \in Q$, initial guess $L_{-1} > 0$

for $k = 0,1,...$ **do**

Set $i_k = 0$. Set $r^k = \lceil \frac{2D_0}{L_{k-1}} \varepsilon \rceil$, generate i.i.d. $\xi_K^i,\ i = 1, \ldots, r^k$

repeat

Set $L_k = 2^{i_k - 1} L_{k-1}$

Calculate $\tilde{g}(x_k) = \frac{1}{r^k} \sum_{i=1}^{r^k} \nabla f(x_k, \xi_k^i)$.

Calculate $w_k = x_k - \frac{1}{2L_k} \tilde{g}(x_k)$.

Calculate $\tilde{f}(x_k) = \frac{1}{r_k} \sum_{i=1}^{r^k} f(x_k, \xi_k^i)$ and
$\tilde{f}(w_k) = \frac{1}{r^k} \sum_{i=1}^{r^k} f(w_k, \xi_k^i)$.

Set $i_k = i_k + 1$.

until
$\tilde{f}(w_k) \leq \tilde{f}(x_k) + \langle \tilde{g}(x_k), w_k - x_k \rangle + \frac{2L_k}{2} \|w_k - x_k\|_2^2 + \frac{\varepsilon}{10}$.

Set $x_{k+1} = w_k$, $k = k + 1$.

end for

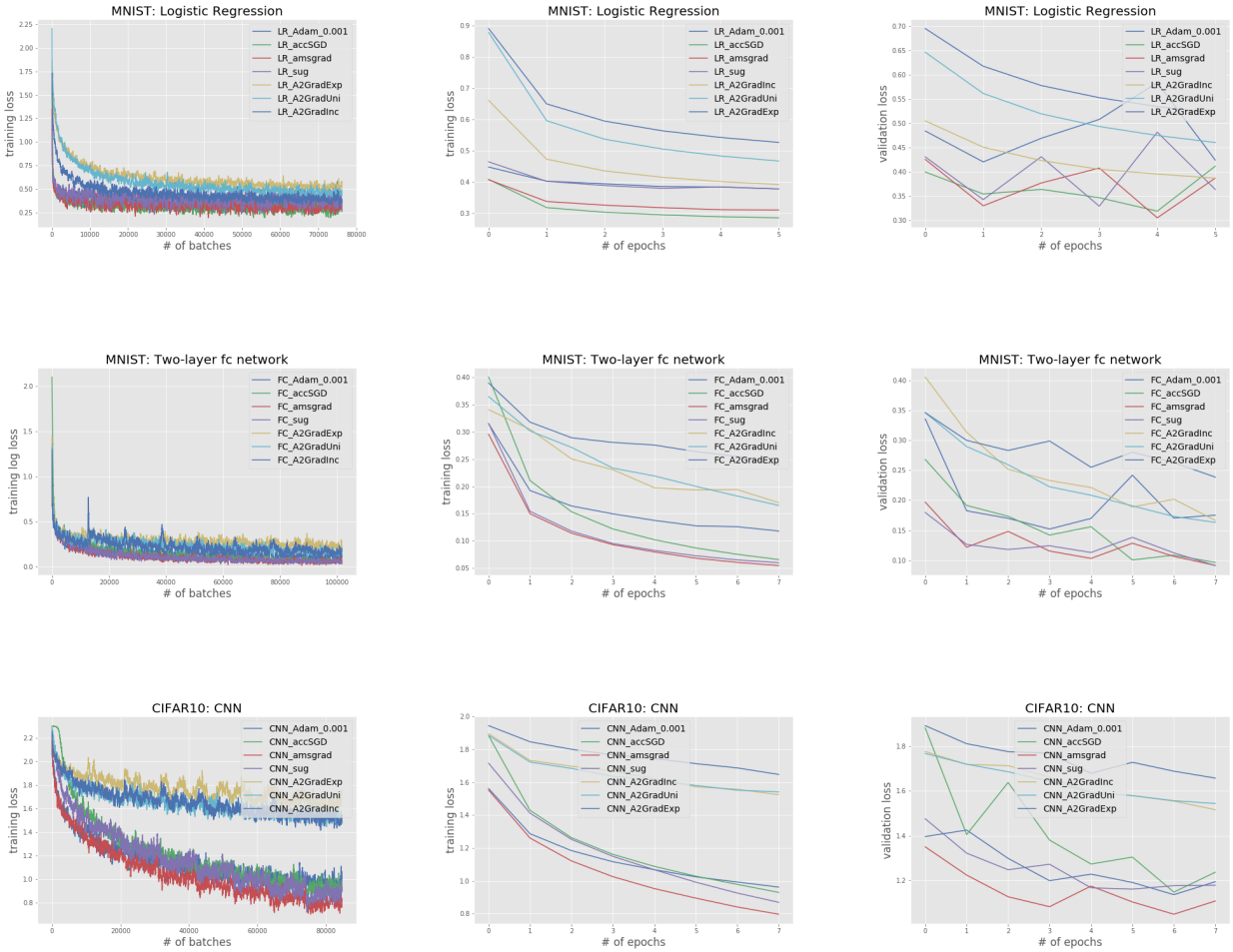

Рис. 1: Первый ряд графиков показывает результаты эксперимента для Logistic Regression на датасете MNIST. Второй - two-layer neural network тоже на MNIST. Последний ряд показывает результаты работы CNN на датасете CIFAR10.

5 Заключение

В данной работе был реализован алгоритм *A2Grad*. Анализ ошибки говорит о том, что этот алгоритм выигрывает в скорости сходимости у всех сравниваемых с ним алгоритмов, но, либо немного хуже, либо показывает такие же результаты в самой сходимости.

Список литературы

[1] *Гасников А.В.* Современные численные методы оптимизации. Метод универсального градиентного спуска

[2] *Камзолов Д.И.* Семинары 674 группы

[3] *Камзолов Д.И.* Лекции ФИВТ

[4] *Камзолов Д.И.* Презентации 674 группы

[5] *Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, Sham M. Kakade,* On the insufficiency of existing momentum schemes for Stochastic Optimization

[6] *Qi Deng, Yi Cheng, Guanghui Lan,* Optimal Adaptive and Accelerated Stochastic Gradient Descent