УДК 62 **Технические науки**

Петухов Дмитрий Евгеньевич, *студент-магистр, Калужский филиал ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»*

Белов Юрий Сергеевич, *к.ф. -м.н., доцент, Калужский филиал ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»*

ОБЗОР ЧАСТО ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ ПО ОПТИМИЗАЦИИ СТОХАСТИЧЕСКОГО ГРАДИЕНТНОГО СПУСКА

Аннотация: Стохастический градиентный спуск (СГС) - это итерационный метод оптимизации целевой функции с подходящими свойствами гладкости. Его можно рассматривать как стохастическую аппроксимацию оптимизации градиентного спуска. В данной статье проводится обзор пяти наиболее часто используемых алгоритмов по оптимизации стохастического градиентного спуска в архитектуре простой сверточной нейронной сети.

Ключевые слова: Оптимизатор импульса, ускоренный градиент Нестерова, адаптивный градиент, адаптивная дельта, адаптивная оценка момента.

Annotation: Stochastic Gradient Descent (SGD) is an iterative objective function optimization method with suitable smoothness properties. It can be thought of as a stochastic approximation of gradient descent optimization. This article provides an overview of the five most commonly used stochastic gradient descent optimization algorithms in a simple convolutional neural network architecture.

Keywords: Momentum, Nesterov accelerated gradient, Adagrad, Adadelta, Adam.

Введение. Стандартный градиентный спуск никак не обеспечивает оптимальной сходимости, но предоставляет ряд вопросов, на которые следует найти решение:

• Подбор подходящей скорости обучения является тяжелым. Очень низкая скорость обучения порождает чрезмерно медлительную сходимость, в таком случае как излишне высокая скорость обучения способна мешать сходимости и порождать колебания функции потерь около минимума, в том числе отклонения.

• Графики скорости обучения [1] стараются корректировать скорость обучения. Все же эти графики и пороговые значения должны устанавливаться предварительно и по этой причине не адаптируемы к характеристикам набора данных.

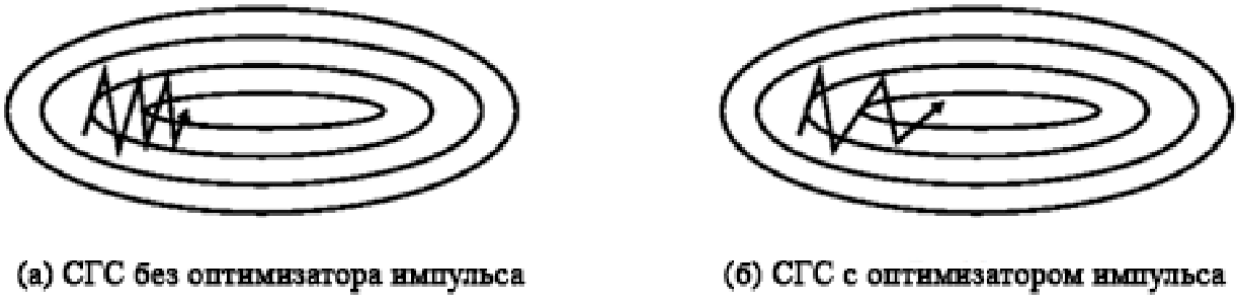
• Помимо этого, к абсолютно всем обновлениям характеристик применяется одна и та же скорость обучения. В случае если наши данные малочисленны и функции обладают весьма различной частотой.

• Еще одна основная цель минимизации невыпуклых функций ошибок, свойственных нейронным сетям, состоит в том, избежать попадания в ловушку их многочисленных субоптимальных локальных минимумов. Дофин и др. заявляют, что трудности появляются на самом деле не из локальных минимумов, а точек, где одно измерение наклоняется вверх, а другое-вниз [2].

Ниже будут описаны некоторые алгоритмы, которые широко используются сообществом глубокого обучения для решения вышеупомянутых проблем. Не будут обсуждаться алгоритмы, которые невозможно вычислить на практике для многомерных наборов данных, например методы второго порядка, такие как метод Ньютона

Оптимизатор импульса (ОИ). У СГС есть проблемы с навигацией по оврагам, то есть по участкам, где поверхность изгибается намного круче в одном измерении, чем в другом, что является обычным явлением вокруг

локальных оптимумов. В этих сценариях СГС колеблется по склонам оврага, при этом неуверенно продвигаясь по дну к локальному оптимуму (рис. 1а.)



(а) СГС без оптимизатора импульса (б) СГС с оптимизатором импульса

Рис. 1. Колебания СГС

Оптимизатор импульса - это метод, который помогает ускорить СГС в соответствующем направлении и гасит колебания (рис. 1б) [3]. Это достигается путем добавления части γ вектора обновления прошедшего временного шага к текущему вектору обновления.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

Член импульса γ обычно устанавливается равным 0.9 или подобное значение. По сути, используя импульс, толкаем мяч с холма. Мяч накапливает импульс по мере того, как катится вниз, становясь все быстрее и быстрее по пути (пока не достигнет своей конечной скорости, если есть сопротивление воздуха, то есть $\gamma < 1$). То же самое происходит с нашими обновлениями параметров: член импульса увеличивается для измерений, градиенты которых указывают в тех же направлениях, и уменьшает обновления для измерений, градиенты которых меняют направление. В результате мы получаем более быструю сходимость и уменьшение колебаний.

Ускоренный градиент Нестерова. Ускоренный градиент Нестерова (УгС) - это способ предоставить нашему термину импульс такого рода предчувствие [4]. Будем использовать наш импульсный член γv_{t-1}, чтобы переместить параметры θ. Вычисление $\theta - \gamma v_{t-1}$, таким образом, дает нам приближение следующего положения параметров (градиент отсутствует для полного обновления), приблизительное представление о том, где будут

находиться наши параметры. Теперь можем эффективно смотреть вперед, вычисляя градиент не относительно наших текущих параметров θ, а относительно приблизительного будущего положения наших параметров:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J\,(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

Устанавливаем импульс γ равным примерно 0,9. Тогда как оптимизатор импульса вначале вычисляет текущий градиент (небольшой синий вектор рис. 2.), а потом выполняет большой скачок в направлении обновленного накопленного градиента (большой синий вектор рис. 2.), УгС сначала делает большой скачок в направлении предыдущего накопленного градиента (коричневый вектор рис. 2.), измеряет градиент и затем корректирует (зеленый вектор рис. 2.). Данное предсказывающее обновление не дает возможности нам действовать слишком быстро и приводит к увеличению скорости отклика, что существенно повышает эффективность РНС для ряда задач.

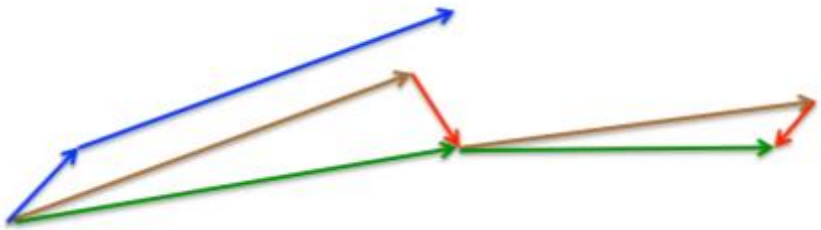


Рис. 2. Обновление Нестерова

Адаптивный градиент (АГ). АГ - это алгоритм для оптимизации на основе градиента, который выполняет именно это: он адаптирует скорость обучения к параметрам, выполняя большие обновления для редких и меньшие обновления для частых параметров [5]. Согласно данному фактору он хорошо подходит для работы с разреженными данными. Дин и др. обнаружили, что АГ значительно улучшил надежность СгС и использовал его для обучения крупномасштабных нейронных сетей в Google, которые, помимо прочего, научились распознавать кошек в видеороликах Youtube [6]. Более того, Пеннингтон и др. использовали АГ для обучения встраиванию слов в GloVe, поскольку редкие слова требуют гораздо больших обновлений, чем частые.

Ранее обновление выполнялось сразу для всех параметров θ, так как каждый параметр θ_i использовал одинаковую скорость обучения η. Поскольку АГ пользуется разной скоростью обучения для каждого параметра θ_i на каждом временном шаге t, сначала демонстрируем обновление АГ для каждого параметра, которое затем векторизуем. Для краткости установим $g_{t,i}$ как градиент целевой функции относительно параметра θ_i на временном шаге t:

$$g_{t,i} = \nabla_{\theta t} J(\theta_{t,i})$$

Обновление СгС для каждого параметра θ_i на каждом временном шаге t тогда становится следующим:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

В своем правиле обновления АГ изменяет общую скорость обучения η на каждом временном шаге t для каждого параметра θ_i на основе прошлых градиентов, которые были вычислены для θ_i:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$G_t \in \mathbb{R}^{d \times d}$ здесь - диагональная матрица, где каждый диагональный элемент i, i представляет собой сумму квадратов градиентов относительно θi до временного шага t, а ϵ это сглаживающий член, который позволяет избежать деления на ноль (обычно порядка 1e - 8). Интересно, что без операции извлечения квадратного корня алгоритм работает намного хуже.

Поскольку G_t содержит сумму квадратов прошлых градиентов относительно ко всем параметрам θ по диагонали, то можем векторизовать нашу реализацию, выполнив поэлементное умножение матрицы на вектор $\odot$ между G_t и g_t:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

Одной из ключевых положительных сторон АГ является то, что он ликвидирует потребность вручную устанавливать скорость обучения. Большая часть реализаций применяют значение по умолчанию 0.01 и оставляют его как есть. Основным минусом АГ считается накапливание квадратов градиентов в

знаменателе: так как любой добавленный член является положительным, накопленная сумма продолжает увеличиваться во время обучения. Это приводит к уменьшению скорости обучения и, в окончательном результате, к бесконечно малой, после чего алгоритм более не способен извлекать вспомогательные знания. Следующие алгоритмы ориентированы на предотвращение данного минуса.

Адаптивная дельта (АД). АД - это расширение АГ, направленное на снижение его агрессивной, монотонно снижающейся скорости обучения [7]. Вместо того, чтобы накапливать все прошлые квадраты градиентов, АД ограничивает окно накопленных прошлых градиентов некоторым фиксированным размером w.

Вместо неэффективного сохранения w предыдущих квадратов градиентов, сумма градиентов рекурсивно определяется как убывающее среднее всех прошлых квадратов градиентов. Скользящее среднее $E[g^2]_t$ на временном шаге t тогда зависит (в виде доли γ аналогично члену ОИ) только от предыдущего среднего и текущего градиента:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)\, g_t^2$$

Устанавливаем γ равным значению, аналогичному значению импульса, около 0.9. Для ясности, теперь перепишем наше стандартное обновление СГС в терминах вектора обновления параметров $\Delta\theta_t$:

$$\Delta\theta_t = -\eta \cdot g_{t,i}$$

$$\theta_{t+1} = \theta t + \Delta\theta_t$$

Вектор обновления параметров АГ, который получился ранее. таким образом, принимает вид:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

Теперь просто заменим диагональную матрицу G_t на убывающее среднее значение по прошлым квадратам градиентов $E[g^2]_t$:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t$$

Поскольку знаменатель - это просто критерий среднеквадратичной ошибки градиента, заменим его сокращенным критерием:

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t}\, g_t$$

Единицы измерения в этом обновлении (а также в СГС, ОИ или АГ) не совпадают, то есть в обновлении должны быть те же гипотетические единицы, что и в параметре. Чтобы понять это, сначала определим другое экспоненциально убывающее среднее, на этот раз не квадратов градиентов, а квадратов обновлений параметров:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma)\ \Delta\theta_t^2$$

Среднеквадратичная ошибка обновления параметров, таким образом, равна:

$$RMS[\Delta\theta]_t = \sqrt{[\Delta\theta^2]_t + \epsilon}$$

Поскольку RMS $[\Delta\theta]_t$ неизвестно, аппроксимируем его с помощью RMS обновлений параметров до предыдущего временного шага. Замена скорости обучения η в предыдущем правиле обновления на RMS $[\Delta\theta]_{t-1}$, наконец, дает правило обновления АД:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$\theta_{t+1} = \theta t\ +\ \Delta\theta_t$$

В АД нам даже не нужно определять скорость обучения по умолчанию, так как она исключена из правила обновления.

Адаптивная оценка момента (АОМ). АОМ - еще один метод, который вычисляет скорость адаптивного обучения для каждого параметра [8]. В добавок к хранению экспоненциально убывающего среднего прошлых квадратов градиентов v_t, АОМ также хранит экспоненциально убывающее среднее прошлых градиентов m_t, аналогично импульсу:

$$m_t = \beta_1 m_{t-1}\ +\ (1\ -\ \beta_1)\, g_t$$

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

m_t и v_t являются оценками первого момента (среднего) и второго момента (нецентрированная дисперсия) градиентов соответственно, отсюда и название метода. Поскольку m_t и v_t инициализируются как векторы нулей, авторы AOM отмечают, что они смещены в сторону нуля, особенно на начальных временных шагах, и особенно когда скорости затухания невелики (т.е. β1 и β2 близки к 1). Они противодействуют этим смещениям, вычисляя скорректированные смещения оценок первого и второго моментов:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Затем они используют их для обновления параметров, что приводит к правилу обновления AOM:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\widehat{m}_t$$

Авторы предлагают значения 0.9 для β1, 0.999 для β2 и 10^{-8} для ϵ. Они эмпирически показывают, что AOM хорошо функционирует на практике и выигрышно выделяется от остальных алгоритмов адаптивного метода обучения.

Выводы: В этой статье были подробно рассмотрены алгоритмы, наиболее часто используемые для оптимизации СГС такие как: оптимизатор импульса, ускоренный градиент Нестерова, адаптивный градиент, адаптивная дельта, адаптивная оценка момента

Библиографический список:

1. D. Chunlin, Y. Liu. Sample Average Approximation Method for Chance Constrained Stochastic Programming in Transportation Model of Emergency Management. Systems Engineering Procedia. // [Электронный ресурс]. URL: https://doi.org/10.1016/j.sepro.2012.04.022.

2. Y. Dauphin, R. Pascanu, C. Gulcehre, C. Cho K., S. Ganguli, Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex

optimization. // Advances in Neural Information Processing Systems. 2014, Vol. 27, P. 2933-2941.

3. I. Sutskever, J. Martens, G. Dahl, G. Hinton. On the importance of initialization and momentum in deep learning. // Proceedings of the 30th International Conference on Machine Learning. 2013, Vol. 28, №3, P. 1139-1147.

4. G. Qu, N. Li. Accelerated Distributed Nesterov Gradient Descent for smooth and strongly convex functions. // IEEE Transactions on Automatic Control. 2020, Vol. 65 № 6, P. 2566-2581.

5. J. Duchi, E. Hazan, Y.Singer. Adaptive SubgradientMethods for Online Learning and Stochastic Optimization. // Journal of Machine Learning Research. 2011, Vol. 12, P. 2121-2159.

6. J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, A. Ng. Large Scale Distributed Deep Networks. // Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012, Vol. 1, P. 1223–1231.

7. M. Zeiler. ADADELTA: An Adaptive Learning Rate Method. // [Электронный ресурс]. URL: https://arxiv.org/abs/1212.5701.

8. Kingma D.P., Ba J. A Method for Stochastic Optimization. // [Электронный ресурс]. URL: https://arxiv.org/abs/1412.6980.