Глубокое обучение

Дмитрий Никулин

14 апреля 2021 г.

Лекция 4: 50 оттенков градиентного спуска

Agenda

- Разбираем способы улучшить SGD
- Экспериментируем с ними

50 оттенков градиентного спуска

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_w \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Градиент указывает направление максимального роста

$$\nabla L(w) = \left(\frac{\partial L(w)}{\partial w_0}, \frac{\partial L(w)}{\partial w_2}, \dots, \frac{\partial L(w)}{\partial w_k} \right)$$

Идём в противоположную сторону:

$$w^1 = w^0 - \lambda \cdot \nabla L(w^0)$$

скорость обучения

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_w \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Инициализация w_0

while True:

$g_t = \frac{1}{n} \sum_{i=1}^{n} \nabla L(w, x_i, y_i)$

$w_t = w_{t-1} - \lambda_t \cdot g_t$

if $||w_t - w_{t-1}|| < \varepsilon$:

break

(можно останавливаться и при превышении количества итераций / по метрикам на валидации / etc)

Стохастический градиентный спуск (SGD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_w \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Инициализация w_0

while True:

рандомно выбрали $b < n$ индексов $B = \{i_1, \dots, i_b\}$

$g_t = \frac{1}{b} \sum_{j \in B} \nabla L(w, x_j, y_j)$

$w_t = w_{t-1} - \lambda_t \cdot g_t$

if $||w_t - w_{t-1}|| < \varepsilon$:

break

Momentum SGD

Мы считали на каждом шаге градиент по формуле

$$g_t = \frac{1}{b}\sum_{i=1}^{b} \nabla L(w_{t-1}, x_i, y_i).$$

После шага мы забывали его. Давайте запоминать направление:

$$m_t = \alpha \cdot m_{t-1} + g_t$$
$$w_t = w_{t-1} - \lambda \cdot m_t$$

- Движение поддерживается в том же направлении, что и на предыдущем шаге
- Нет резких изменений направления движения.
- Обычно $\alpha \approx 0.9$.

Крутой интерактив для моментума: https://distill.pub/2017/momentum/

Momentum SGD

- Бежим с горки и всё больше ускоряемся в том направлении, в котором были направлены сразу несколько предыдущих градиентов, но при этом движемся медленно там, где градиент постоянно меняется

- Хотелось бы не просто бежать с горы, но и хотя бы на полшага смотреть себе под ноги, чтобы внезапно не споткнуться $\Rightarrow$ давайте смотреть на градиент в будущей точке

- В Momentum SGD $\alpha \cdot m_{t-1}$ точно будет использоваться при шаге, давайте искать $\nabla L(w_{t-1} - \lambda \cdot \alpha \cdot m_{t-1})$.

Nesterov Momentum SGD

- Мы теперь сначала прыгаем в том же направлении, в каком шли до этого, потом корректируем его
- Иллюстрация из статьи Хинтона 2013 года:

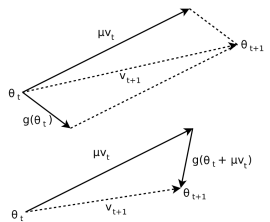


Figure 1. **(Top)** Classical Momentum **(Bottom)** Nesterov Accelerated Gradient

http://www.cs.toronto.edu/~hinton/absps/momentum.pdf

Nesterov Momentum SGD

- Мы теперь сначала прыгаем в том же направлении, в каком шли до этого, потом корректируем его

$$m_t = \alpha \cdot m_{t-1} + \nabla L(w_{t-1} - \alpha \cdot m_{t-1})$$
$$w_t = w_{t-1} - \lambda \cdot m_t$$

Разная скорость обучения

- Может сложиться, что некоторые веса уже близки к своим локальным минимумам, по этим координатам надо двигаться медленнее, а по другим быстрее $\Rightarrow$ адаптивные методы градиентного спуска

- Шаг изменения должен быть меньше у тех параметров, которые в большей степени варьируются в данных, и больше у тех, которые менее изменчивы

AdaGrad

$$
\begin{aligned}
v_0^j &= 0 \\
v_t^j &= v_{t-1}^j + g_{tj}^2 \\
w_t^j &= w_{t-1}^j - \frac{\lambda}{\sqrt{v_t^j + \varepsilon}} \cdot g_{tj}
\end{aligned}
$$

- g_{tj} — градиент по $j-$ому параметру
- своя скорость обучения для каждого параметра
- обычно $\lambda = 0.01$, т.к. параметр не очень важен
- v_t^j всегда увеличивается, из-за этого обучение может рано останавливаться $\Rightarrow$ RMSprop

RMSprop

$$
\begin{aligned}
v_0^j &= 0 \\
v_t^j &= \alpha \cdot v_{t-1}^j + (1 - \alpha) \cdot g_{tj}^2 \\
w_t^j &= w_{t-1}^j - \frac{\lambda_t}{\sqrt{v_t^j + \varepsilon}} \cdot g_{tj}
\end{aligned}
$$

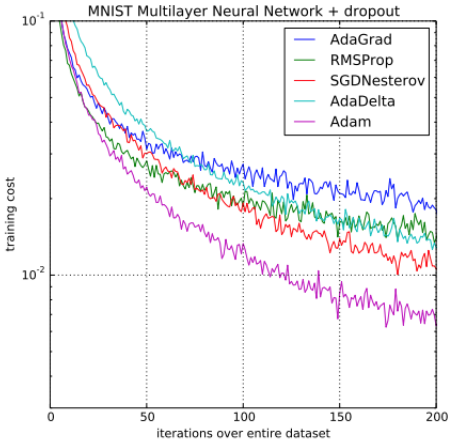
- Обычно $\alpha = 0.99$
- Скорость обучения адаптируется к последнему сделанному шагу, бесконтрольного роста v_t^j больше не происходит
- RMSprop нигде не был опубликован, Хинтон просто привёл его в своей лекции, сказав, что это норм тема

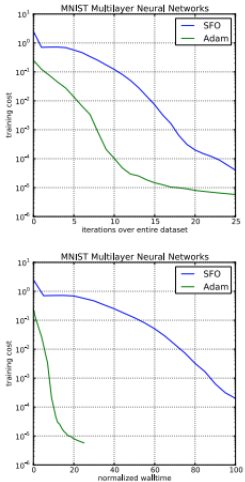
Adam (Adaptive Moment Estimation)

$$m_t^j = \beta_1 \cdot m_{t-1}^j + (1 - \beta_1) \cdot g_{tj}$$
$$v_t^j = \beta_2 \cdot v_{t-1}^j + (1 - \beta_2) \cdot g_{tj}^2$$
$$\hat{m}_t^j = \frac{m_t^j}{1 - \beta_1^t} \quad \hat{v}_t^j = \frac{v_t^j}{1 - \beta_2^t}$$
$$w_t^j = w_{t-1}^j - \frac{\lambda_t}{\sqrt{\hat{v}_t^j} + \varepsilon} \cdot \hat{m}_t^j$$

- Комбинируем Momentum и индивидуальные скорости обучения
- Фактически, $\hat{m}_t$ и $\hat{v}_t$ — это оценки первого и второго моментов для стохастического градиента
- Деление на $1 - \beta_i^t$ нужно, чтобы оценки были несмещёнными

Сравнение на MNIST (из статьи про Adam)





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

Лучший оптимизатор?..

https://towardsdatascience.com/optimization-algorithms-in-deep-learning-191bfc2737a4

… или нет?

A. List of optimizers and schedules considered

Table 2: List of optimizers we considered for our benchmark. Note, that this is far from being a complete list of all existing optimization methods applicable to deep learning, but only a subset, comprising of some of the most popular choices.

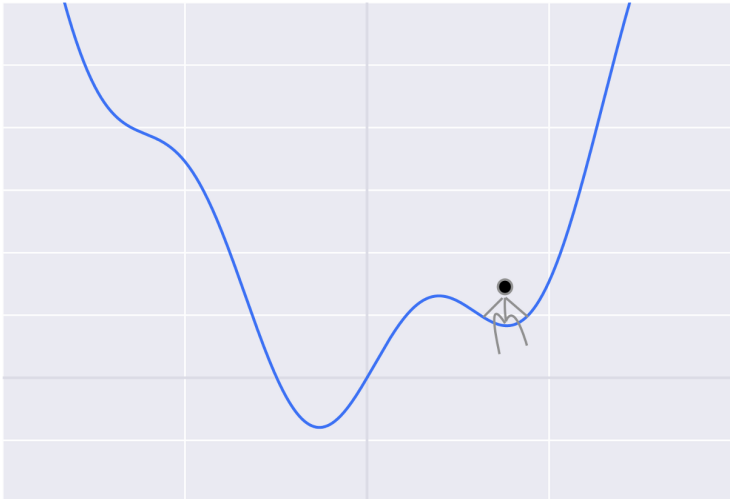
Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KFAC	(Martens & Grosse, 2015)
AdaBatch	(Devarakonda et al., 2017)	KFLR/KFRA	(Botev et al., 2017)
AdaBayes/AdaBayes-SS	(Aitchison, 2020)	L4Adam/L4Momentum	(Rolínek & Martius, 2018)
AdaBelief	(Zhuang et al., 2020)	LAMB	(You et al., 2020)
AdaBlock	(Yun et al., 2019)	LaProp	(Ziyin et al., 2020)
AdaBound	(Luo et al., 2019)	LARS	(You et al., 2017)
AdaComp	(Chen et al., 2018)	LookAhead	(Zhang et al., 2019)
Adadelta	(Zeiler, 2012)	M-SVAG	(Balles & Hennig, 2018)
Adafactor	(Shazeer & Stern, 2018)	MAS	(Landro et al., 2020)
AdaFix	(Bae et al., 2019)	MEKA	(Chen et al., 2020b)
AdaFom	(Chen et al., 2019a)	MTAdam	(Malkiel & Wolf, 2020)
AdaFTRL	(Orabona & Pál, 2015)	MVRC-1/MVRC-2	(Chen & Zhou, 2020)
Adagrad	(Duchi et al., 2011)	Nadam	(Dozat, 2016)
ADAHESSIAN	(Yao et al., 2020)	NAMSB/NAMSG	(Chen et al., 2019b)
Adai	(Xie et al., 2020)	ND-Adam	(Zhang et al., 2017)
AdaLoss	(Teixeira et al., 2019)	Nesterov	(Nesterov, 1983)
Adam	(Kingma & Ba, 2015)	Noisy Adam/Noisy K-FAC	(Zhang et al., 2018)
Adam^{+}	(Liu et al., 2020b)	NosAdam	(Huang et al., 2019)
AdamAL	(Tao et al., 2019)	Novograd	(Ginsburg et al., 2019)
AdaMax	(Kingma & Ba, 2015)	Padam	(Chen et al., 2020a)
AdamBS	(Liu et al., 2020c)	PAGE	(Li et al., 2020b)
AdamNC	(Reddi et al., 2018)	PAL	(Mutschler & Zell, 2020)
AdaMod	(Ding et al., 2019)	PolyAdam	(Orvieto et al., 2019)
AdamP	(Heo et al., 2020)	Polyak	(Polyak, 1964)
AdamT	(Zhou et al., 2020)	PowerSGD/PowerSGDM	(Vogels et al., 2019)
AdamW	(Loshchilov & Hutter, 2019)	ProbLS	(Mahsereci & Hennig, 2017)
AdamX	(Tran & Phong, 2019)	PStorm	(Xu, 2020)
ADAS	(Eliyahu, 2020)	QHAdam/QHM	(Ma & Yarats, 2019)
AdaS	(Hosseini & Plataniotis, 2020)	RAdam	(Liu et al., 2020a)
AdaScale	(Johnson et al., 2020)	Ranger	(Wright, 2020b)
AdaSGD	(Wang & Wiens, 2020)	RangerLars	(Grankin, 2020)
AdaShift	(Zhou et al., 2019)	RMSProp	(Tieleman & Hinton, 2012)
AdaSqrt	(Hu et al., 2019)	RMSterov	(Choi et al., 2019)
Adathm	(Sun et al., 2019)	S-SGD	(Sung et al., 2020)
AdaX/AdaX-W	(Li et al., 2020a)	SAdam	(Wang et al., 2020b)
AEGD	(Liu & Tian, 2020)	Sadam/SAMSGrad	(Tong et al., 2019)
ALI-G	(Berrada et al., 2020)	SALR	(Yue et al., 2020)
AMSBound	(Luo et al., 2019)	SC-Adagrad/SC-RMSProp	(Mukkamala & Hein, 2017)
AMSGrad	(Reddi et al., 2018)	SDProp	(Ida et al., 2017)
ArmijoLS	(Vaswani et al., 2019)	SGD	(Robbins & Monro, 1951)
ARSG	(Chen et al., 2019b)	SGD-BB	(Tan et al., 2016)
AvaGrad	(Savarese et al., 2019)	SGD-G2	(Ayadi & Turinici, 2020)
BAdam	(Salas et al., 2018)	SGDM	(Liu & Luo, 2020)
BGAdam	(Bai & Zhang, 2019)	SGDP	(Heo et al., 2020)
BRMSProp	(Aitchison, 2020)	SGDR	(Loshchilov & Hutter, 2017)
BSGD	(Hu et al., 2020)	SHAdagrad	(Huang et al., 2020)
C-ADAM	(Tutunov et al., 2020)	Shampoo	(Anil et al., 2020; Gupta et al., 2018)
CADA	(Chen et al., 2020c)	SignAdam++	(Wang et al., 2019a)
Cool Momentum	(Borysenko & Byshkin, 2020)	SignSGD	(Bernstein et al., 2018)
CProp	(Preechakul & Kijsirikul, 2019)	SKQN/S4QN	(Yang et al., 2020)
Curveball	(Henriques et al., 2019)	SM3	(Anil et al., 2019)
Dadam	(Nazari et al., 2019)	SMG	(Tran et al., 2020)
DeepMemory	(Wright, 2020a)	SNGM	(Zhao et al., 2020)
DiffGrad	(Dubey et al., 2020)	SoftAdam	(Fetterman et al., 2019)
EAdam	(Yuan & Gao, 2020)	SRSGD	(Wang et al., 2020a)
EKFAC	(George et al., 2018)	SWATS	(Keskar & Socher, 2017)
Eve	(Hayashi et al., 2018)	SWNTS	(Chen et al., 2019c)
Expectigrad	(Daley & Amato, 2020)	TAdam	(Ilboudo et al., 2020)
FRSGD	(Wang & Ye, 2020)	TEKFAC	(Gao et al., 2020)
GADAM	(Zhang & Gouza, 2018)	VAdam	(Khan et al., 2018)
Gadam	(Granziol et al., 2020)	VR-SGD	(Shang et al., 2020)
GOLS-I	(Kafka & Wilke, 2019)	vSGD-b/vSGD-g/vSGD-l	(Schaul et al., 2013)
Grad-Avg	(Purkayastha & Purkayastha, 2020)	vSGD-fd	(Schaul & LeCun, 2013)
Gravilon	(Kelterborn et al., 2020)	WNGrad	(Wu et al., 2018)
Gravity	(Bahrami & Zadeh, 2021)	YellowFin	(Zhang & Mitliagkas, 2019)
HAdam	(Jiang et al., 2019)	Yogi	(Zaheer et al., 2018)

https://arxiv.org/pdf/2007.01547.pdf

Резюме по методам градиентного спуска

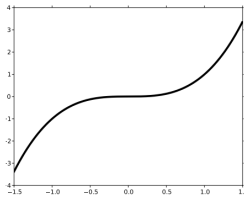
- Momentum SGD сохраняет направление шага и позволяет добиваться более быстрой сходимости
- Адаптивные методы позволяют находить индивидуальную скорость обучения для каждого параметра
- Adam комбинирует в себе оба подхода
- Давайте посмотрим визуализацию 1 и визуализацию 2
- Но это же не все вызовы!

Боб чилит в локальном минимуме

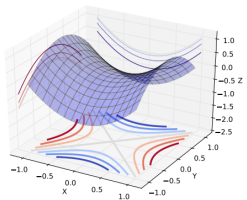


https://hackernoon.com/life-is-gradient-descent-880c60ac1be8

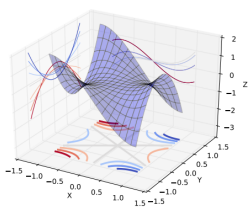
Седловые точки



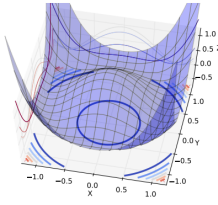
(a)



(b)



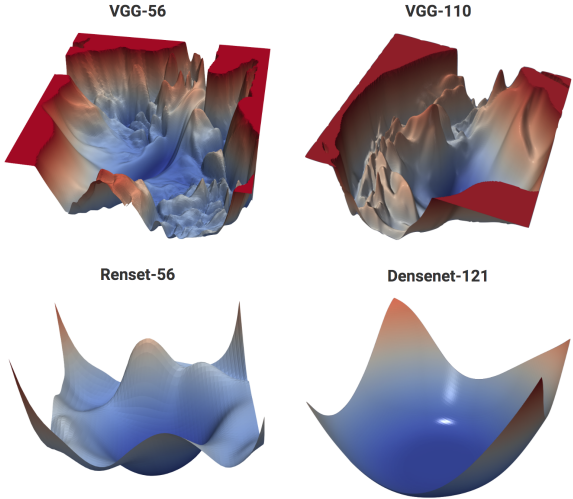
(c)



(d)

https://arxiv.org/pdf/1406.2572.pdf

Визуализация потерь



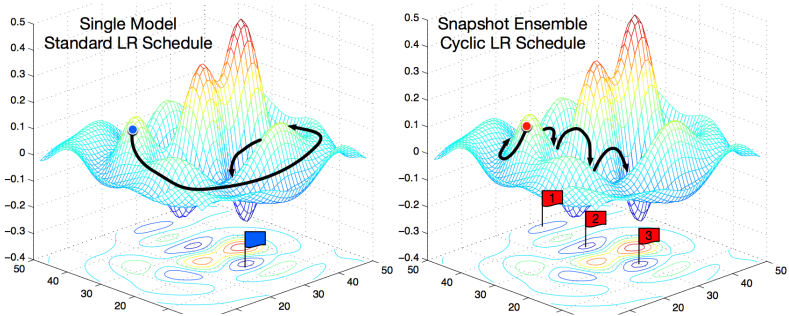
https://arxiv.org/pdf/1712.09913.pdf
https://github.com/tomgoldstein/loss-landscape

Переменный learning rate

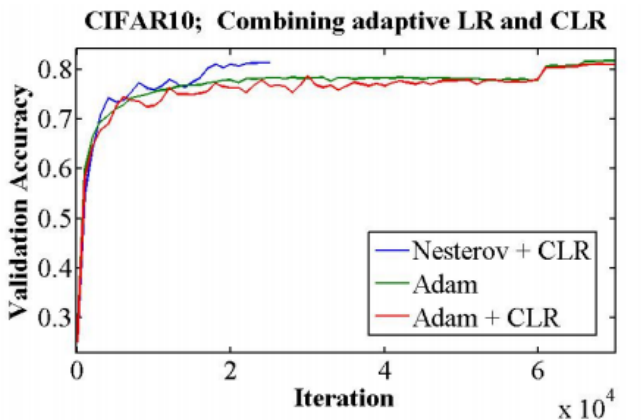
- Когда качество на валидации перестаёт улучшаться, уменьшаем learning rate в несколько раз

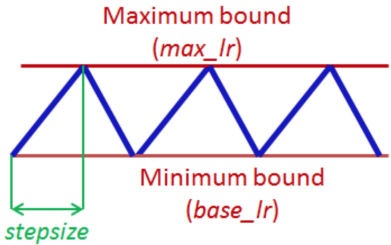
Циклическая скорость обучения (CLR)

- Хочется, чтобы был шанс вылезти из локального минимума, а также шанс сползти с седла $\Rightarrow$ давайте менять глобальную скорость обучения циклически



Циклическая скорость обучения (CLR)





Нестеров с CLR отработал быстрее и лучше Adam

Нет одного правильного алгоритма на все случаи!

Всегда надо экспериментировать

https://arxiv.org/pdf/1506.01186.pdf
https://openreview.net/pdf?id=BJYwwY9ll

А что люди сейчас делают с оптимизаторами?

- Нейросеть пытается сама от слоя к слою выстраивать новые, более сложные фичи
- Самостоятельно фичи больше придумывать не надо
- При этом оптимизаторы какие-то слишком олдскульные и ручные
- Круто было бы, если бы оптимизаторы тоже сами обучались
- Такие работы в последнее время активно появляются, но какого-то суперпрогресса пока нет

https://arxiv.org/pdf/2009.11243.pdf

Экспериментируем!