Глубокое обучение

Неделя 2: 50 оттенков градиентного спуска

Agenda

- 50 оттенков градиентного спуска или стратегии его реализации
- Несколько слов о Backpropagation
- Семинар с кодом разных градиентных спусков

Как обучать нейросеть?

- Нейросеть - сложная функция, зависящая от весов W
- «Тренировка» — поиск оптимальных W
- «Оптимальных» — минимизирующих какой-то функционал
- Какими бывают функционалы: MSE, MAE, logloss и многие другие
- Как оптимизировать: градиентный спуск!

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_{w}$$

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_{w}$$

Градиент указывает направление максимального роста

$$\nabla L(w) = \left(\frac{\partial L(w)}{\partial w_0}, \frac{\partial L(w)}{\partial w_1}, \ldots, \frac{\partial L(w)}{\partial w_k} \right)$$

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_{w}$$

Градиент указывает направление максимального роста

$$\nabla L(w) = \left(\frac{\partial L(w)}{\partial w_0}, \frac{\partial L(w)}{\partial w_1}, \ldots, \frac{\partial L(w)}{\partial w_k} \right)$$

Идём в противоположную сторону:

$$w^1 = w^0 - \underbrace{\eta}_{\text{скорость обучения}} \cdot \nabla L(w^0)$$

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_{w}$$

Инициализация w_0

while True:

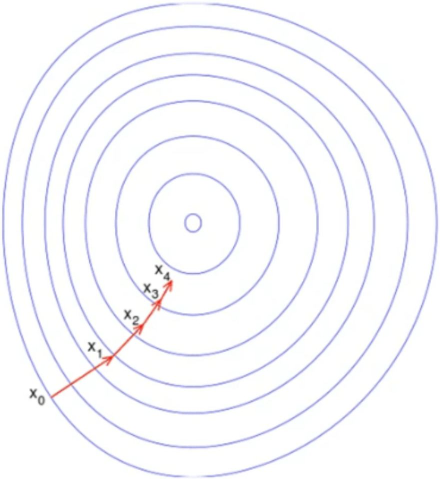
$g_t = \frac{1}{n} \sum_{i=1}^{n} \nabla L(w, x_i, y_i)$

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

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

break

Градиентный спуск



Пример:

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_w$$

Градиент:

$$\nabla L(w) = -2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i$$

Пример:

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_w$$

Градиент:

$$\nabla L(w) = -2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i$$

Дорого постоянно считать такие суммы!

Стохастический градиентный спуск (SGD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_{w}$$

Инициализация w_0

while True:

    рандомно выбрали i

    $g_t = \nabla L(w_{t-1}, x_i, y_i)$

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

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

        break

Пример:

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_w$$

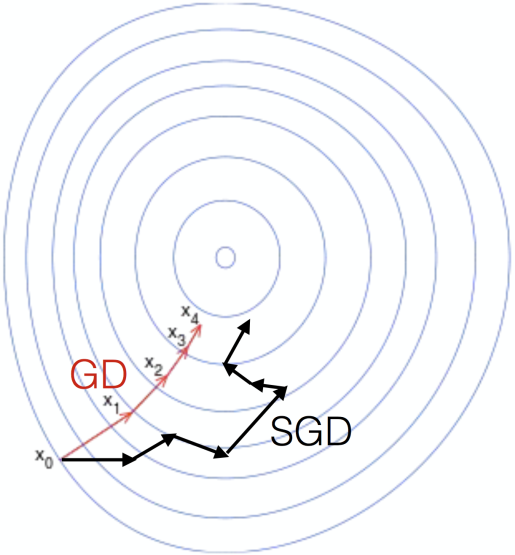
Градиент:

$$\nabla L(w) = -2 \cdot (y_i - x_i^T w) \cdot x_i$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot (y_i - x_i^T w) \cdot x_i$$

Стохастический градиентный спуск (SGD)



- И для GD и для SGD нет гарантий глобального минимума, сходимости
- SGD быстрее,на каждой итерации используется только одно наблюдение
- Для SGD спуск очень зашумлён
- GD: $O(n)$, SGD: $O(1)$

Mini-bath SGD

Проблема оптимизации:

$$L(w) = \sum_{i=1}^{n} L(w, x_i, y_i) \to \min_w$$

Инициализация w_0

while True:

рандомно выбрали $m < n$ индексов

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

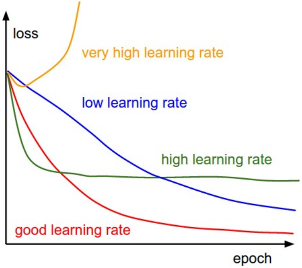
$w_t = w_{t-1} - \eta_t \cdot g_t$

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

break

Вызовы

- Скорость обучения η надо подбирать аккуратно, если она будет большой, мы можем скакать вокруг минимума, если маленькой - вечно ползти к нему.



- К обновлению всех параметров применяется одна и та же скорость обучения. Возможно, что какие-то параметры приходят в оптимальную точку быстрее, и их не надо обновлять.

Momentum SGD

Мы считали на каждом шаге градиент по формуле

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

После шага мы забывали его. Давайте запоминать направление:

$$h_t = \alpha \cdot h_{t-1} + \eta \cdot g_t$$
$$w_t = w_{t-1} - h_t$$

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

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

Momentum SGD

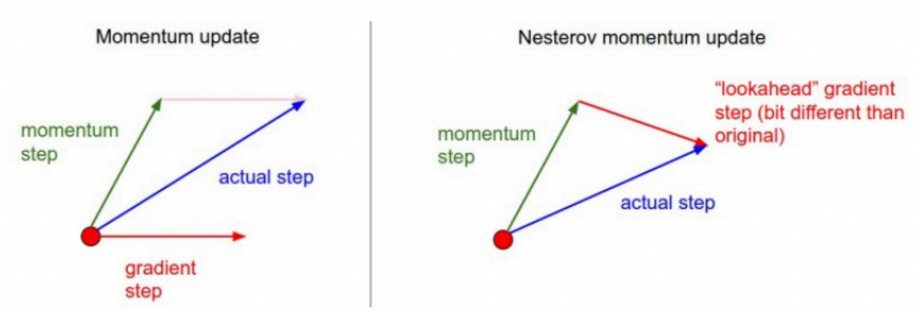
- Бежим с горки и всё больше ускоряемся в том направлении, в котором были направлены сразу несколько предыдущих градиентов, но при этом движемся медленно там, где градиент постоянно меняется

- 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

- Согласно методу моментов $\alpha \cdot h_{t-1}$ точно будет использоваться при шаге, давайте искать $\nabla L(w_{t-1} - \alpha \cdot h_{t-1})$.

Nesterov Momentum SGD

- Мы теперь сначала прыгаем в том же направлении, в каком шли до этого, потом корректируем его (голубая траектория).



$$h_t = \alpha \cdot h_{t-1} + \eta \cdot \nabla L(w_{t-1} - \alpha \cdot h_{t-1})$$
$$w_t = w_{t-1} - h_t$$

Рисунок взят из классчисеких лекций Стэнфорда https://cs231n.github.io/neural-networks-3/

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

$$G_t^j = G_{t-1}^j + g_{tj}^2$$
$$w_t^j = w_{t-1}^j - \frac{\eta}{\sqrt{G_t^j + \varepsilon}} \cdot g_t^j$$

- g_t^j — градиент по j—ому параметру
- своя скорость обучения для каждого параметра
- обычно $\eta = 0.01$, т.к. параметр не очень важен
- G_t^j всегда увеличивается, из-за этого обучение может рано останавливаться (изменение весов затухает) $\Rightarrow$ RMSprop

RMSprop

$$G_t^j = \alpha \cdot G_{t-1}^j + (1 - \alpha) \cdot g_{tj}^2$$
$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot g_t^j$$

- Обычно $\alpha = 0.9$
- Скорость обучения адаптируется к последнему сделанному шагу, бесконтрольного роста G_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)

$$h_t^j = \beta_1 \cdot h_{t-1}^j + (1 - \beta_1) \cdot g_{tj}$$

$$G_t^j = \beta_2 \cdot G_{t-1}^j + (1 - \beta_2) \cdot g_{tj}^2$$

$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot h_t^j$$

- $\beta_1 = 0.9$, $\beta_2 = 0.999$
- Комбинируем Momentum и индивидуальные скорости обучения
- Фактически h_t и G_t это оценки первого и второго моментов для стохастического градиента

Adam with bias correction

$$h_t^j = \beta_1 \cdot h_{t-1}^j + (1-\beta_1) \cdot g_{tj}$$

$$G_t^j = \beta_2 \cdot G_{t-1}^j + (1-\beta_2) \cdot g_{tj}^2$$

$$\widehat{h_t^j} = \frac{h_t^j}{1-\beta_1^t}, \ \widehat{G_t^j} = \frac{G_t^j}{1-\beta_2^t} \quad \text{(bias correction)}$$

$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{\widehat{G_t^j} + \varepsilon}} \cdot \widehat{h_t^j}$$

Зачем идейно нужен bias correction?

- Adam вычисляет скользящее среднее $WMA_t = \sum_{i=0}^{n-1} w_{t-i} \cdot f_{t-i}$ градиентов и называет его h_t

- Причина: мы не хотим сильно полагаться на отдельно взятый градиент g_t. Нам хочется использовать знание об изменениях градиента с течением времени

- Один из способов достичь наши цели - это рассчитывать мат. ожидание $E[g_t] = E[\nabla L(w_t)]$

- Тем не менее в Adam рассчитывается не $E[\nabla L(w_t)]$, а h_t, причем методом скользящего среднего

- В идеальном мире нам хочется: $E[h_t] = E[\nabla L(w_t)]$. Поэтому мы делаем bias correction

Откуда берётся bias correction?

CS7015: Lec 5.9 (Part-2)

$$
\begin{aligned}
h_t &= \beta_1 \cdot h_{t-1} + (1-\beta_1) \cdot g_t \\
h_0 &= 0 \\
h_1 &= \beta h_0 + (1-\beta) g_1 \\
&= (1-\beta) g_1 \\
h_2 &= \beta h_1 + (1-\beta) g_2 \\
&= \beta(1-\beta) g_1 + (1-\beta) g_2 \\
h_3 &= \beta h_2 + (1-\beta) g_3 \\
&= \beta\left(\beta(1-\beta) g_1 + (1-\beta) g_2\right) + (1-\beta) g_3 \\
&= \beta^2 (1-\beta) g_1 + \beta(1-\beta) g_2 + (1-\beta) g_3 \\
&= (1-\beta) \sum_{i=1}^{3} \beta^{3-i} g_i \Rightarrow \boxed{h_t = (1-\beta) \sum_{i=1}^{t} \beta^{t-i} g_i}
\end{aligned}
$$

Откуда берётся bias correction?

Возьмем мат. ожидание с обеих сторон

$$E\left[h_t\right] = E\left[(1-\beta)\sum_{i=1}^{t}\beta^{t-i}g_i\right]$$

$$E\left[h_t\right] = (1-\beta)E\left[\sum_{i=1}^{t}\beta^{t-i}g_i\right]$$

$$E\left[h_t\right] = (1-\beta)\sum_{i=1}^{t}E\left[\beta^{t-i}g_i\right]$$

$$= (1-\beta)\sum_{i=1}^{t}\beta^{t-i}E\left[g_i\right]$$

Откуда берётся bias correction?

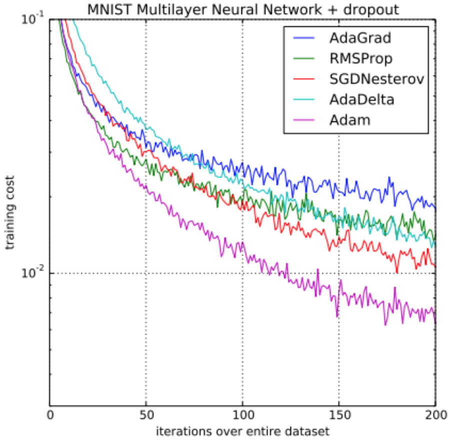
$$
\begin{aligned}
E\left[h_t\right] &= (1-\beta)\sum_{i=1}^{t}(\beta)^{t-i}E[g] \\
&= E[g](1-\beta)\sum_{i=1}^{t}(\beta)^{t-i} \\
&= E[g](1-\beta)\left(\beta^{t-1}+\beta^{t-2}+\dots+\beta^{0}\right) \\
&= E[g](1-\beta)\frac{1-\beta^t}{1-\beta}
\end{aligned}
$$

$$E\left[h_t\right] = E[g]\left(1-\beta^t\right)$$

$$E\left[\frac{h_t}{1-\beta^t}\right] = E[g]$$

$$E\left[\hat{h}_t\right] = E[g]\left(\text{т.к.}\frac{h_t}{1-\beta^t} = \hat{h}_t\right)$$

Сравнение на MNIST

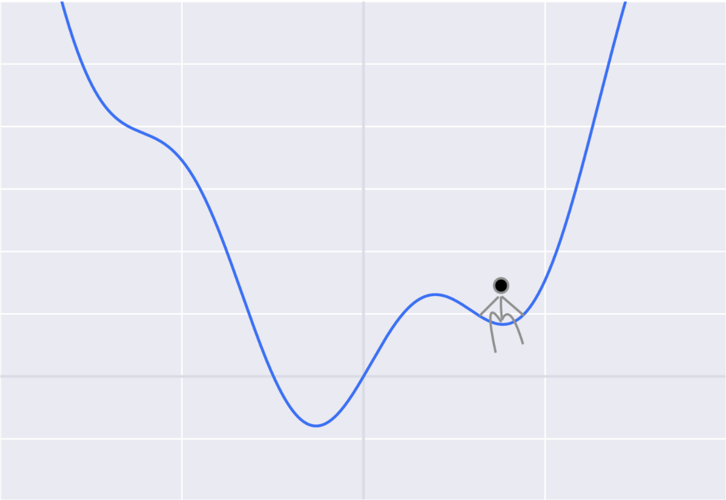


https://arxiv.org/pdf/1412.6980.pdf

Резюме по методам градиентного спуска

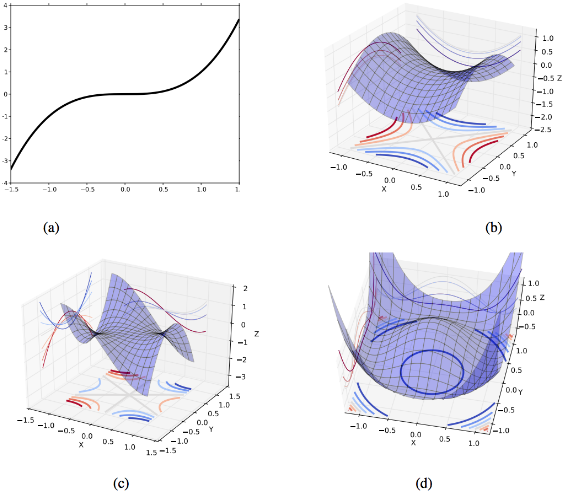
- Momentum SGD сохраняет направление шага и позволяет добиваться более быстрой сходимости
- Адаптивные методы позволяют находить индивидуальную скорость обучения для каждого параметра
- Adam комбинирует в себе оба подхода
- Давайте посмотрим визуализацию 1 и визуализацию 2
- Но это же не все вызовы!

Боб чилит в локальном минимуме



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

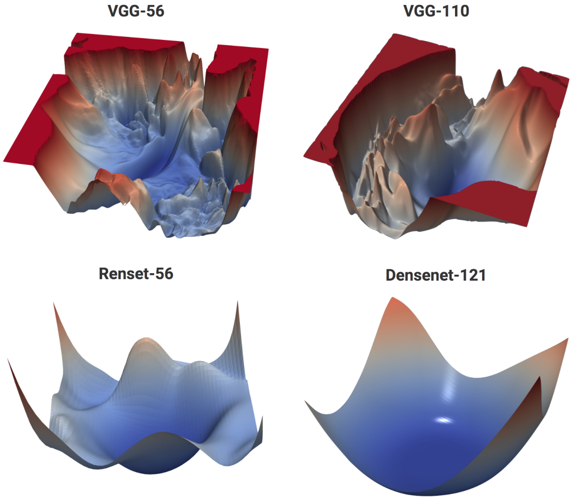
Седловые точки



(a) (b) (c) (d)

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

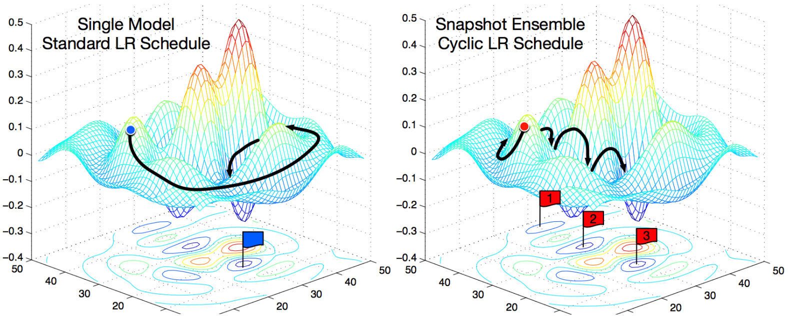
Визуализация потерь



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

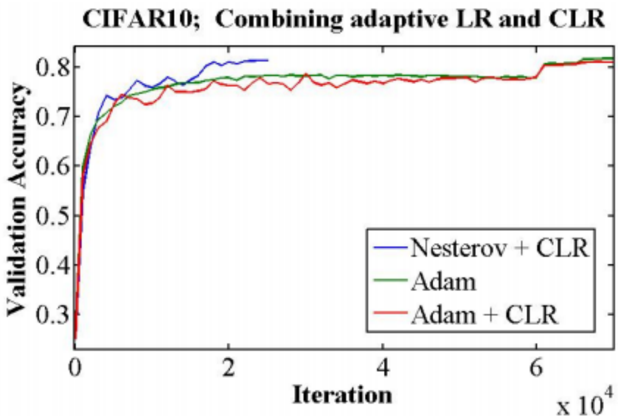
Циклическая скорость обучения (CLR)

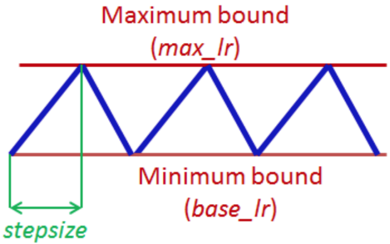
- Хочется, чтобы был шанс вылезти из локального минимума, а также шанс сползти с седла ⇒ давайте менять глобальную скорость обучения циклически



https://openreview.net/pdf?id=BJYwwY9llhttps://arxiv.org/pdf/1506.01186.pdf
https://openreview.net/pdf?id=BJYwwY9ll

Циклическая скорость обучения (CLR)



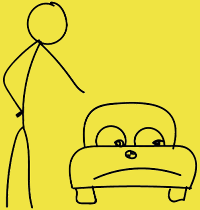


Нестеров с CLR отработал быстрее и лучше Adam

Нет одного правильного алгоритма на все случаи!

Всегда надо экспериментировать

Так как же обучить нейросетку?



Ты необучаем!

Нейросеть — сложная функция

- Прямое распространение ошибки (forward propagation):

$$X \Rightarrow X \cdot W_1 \Rightarrow f(X \cdot W_1) \Rightarrow f(X \cdot W_1) \cdot W_2 \Rightarrow \ldots \Rightarrow \hat{y}$$

- Считаем потери:

$$Loss = \frac{1}{2}(y - \hat{y})^2$$

- Для обучения нужно использовать градиентный спуск

Как обучить нейросеть?

$$L(W_1, W_2) = \frac{1}{2} \cdot (y - f(X \cdot W_1) \cdot W_2)^2$$

Секрет успеха в умении брать производную и градиентном спуске.

$$f(g(x))' = f'(g(x)) \cdot g'(x)$$

$$\frac{\partial L}{\partial W_2} = -(y - f(X \cdot W_1) \cdot W_2) \cdot f(X \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = -(y - f(X \cdot W_1) \cdot W_2) \cdot W_2 f'(X \cdot W_1) \cdot X$$

Как обучить нейросеть?

$$L(W_1, W_2) = \frac{1}{2} \cdot (y - f(X \cdot W_1) \cdot W_2)^2$$

Секрет успеха в умении брать производную и градиентном спуске.

$$f(g(x))' = f'(g(x)) \cdot g'(x)$$

$$\frac{\partial L}{\partial W_2} = -(y - f(X \cdot W_1) \cdot W_2) \cdot f(X \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = -(y - f(X \cdot W_1) \cdot W_2) \cdot W_2 f'(X \cdot W_1) \cdot X$$

Дважды ищем одно и то же $\Rightarrow$ оптимизация поиска производных даст нам алгоритм обратного распространения ошибки (back-propagation)

Back-propagation