Глубокое обучение и вообще

Соловей Влад

17 ноября 2020 г.

Посиделка 2: 50 оттенков градиентного спуска

Agenda

- 50 оттенков градиентного спуска
- Немного поговорить о Backpropagation

50 оттенков градиентного спуска

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

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

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

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

Идём в противоположную сторону:

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

скорость обучения

Градиентный спуск (GD)

Проблема оптимизации:

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

Инициализация w_0

while True:

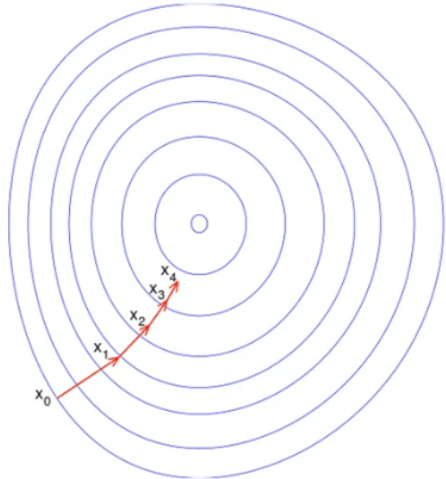
$g_t = \frac{1}{n} \sum_{i=1}^{n} \nabla L(w, x_i, y_i)$

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

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:

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

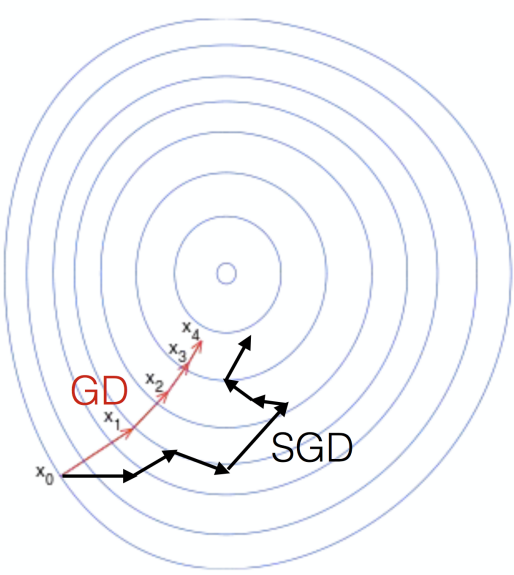
Градиент:

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

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

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

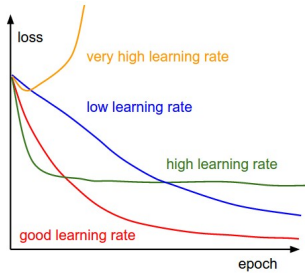
$w_t = w_{t-1} - \eta_t \cdot g_t$

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.

Risk
modelling.research

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 $\nabla L(w_{t-1} - \alpha \cdot h_{t-1})$.

Momentum SGD

Примечания:

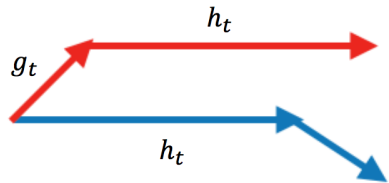
- Momentum использует экспоненциальное скользящее среднее (EMA), что позволяет уменьшить вычислительную сложность вычисления предыдущих значений весов.

- Хорошо подходит для ResNet архитектур, а именно для задачи классификации изображений.

https://arxiv.org/pdf/1607.01981.pdf

Nesterov Momentum SGD

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



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

Хорошо подходит для RNN сетей

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 $\Rightarrow$ RMSprop

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

Примечание:AdaGrad

Подходит для разреженных данных. Glove обучался с помощью AdaGrad, так для редко встречающихся слов требуется большие изменения, чем для часто встречающихся.

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

Примечания: RMSprop

Подходит для CNN, глубоких нейронных сетей. MobileNets был обучен с помощью RMSProp

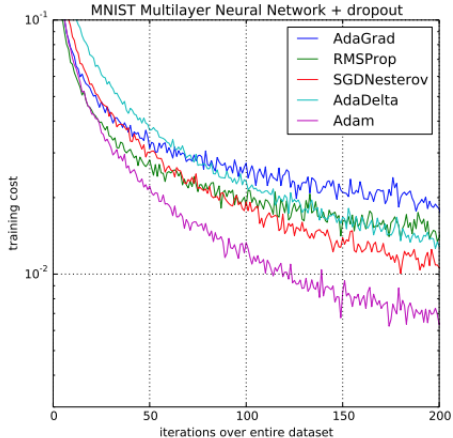
Adam (Adaptive Moment Estimation)

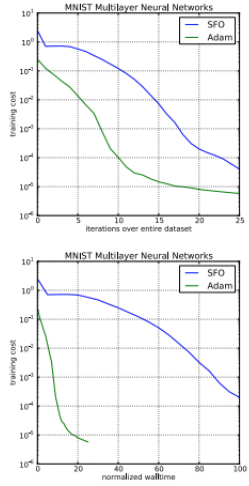
$$h_t^j = \beta_1 \cdot h_{t-1}^j + (1 - \beta_1) \cdot g_{tj}$$
$$G_t^j = \beta_2 \cdot G_{t-1}^j + (1 - \beta_2) \cdot g_{tj}^2$$
$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot h_t^j$$

- Комбинируем Momentum и индивидуальные скорости обучения
- Фактически h_t и G_t это оценки первого и второго моментов для стохастического градиента

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

Сравнение на MNIST





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

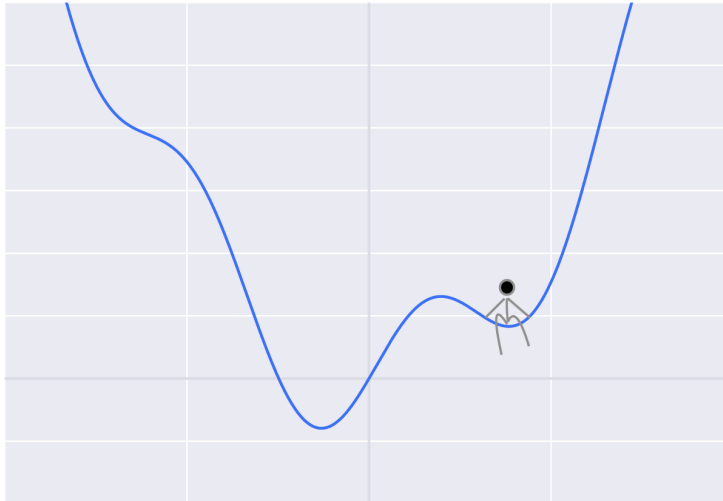
Risk
modelling.research

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!

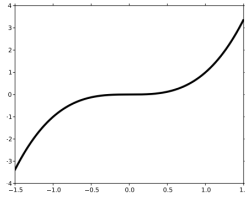
Risk
modelling.research

Боб чилит в локальном минимуме

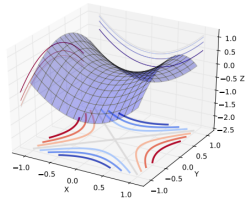


https://hackernoon.com/life-is-gradient-descent-880c60ac1be8

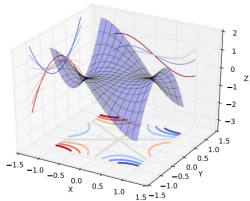
Седловые точки



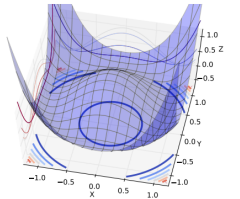
(a)



(b)



(c)

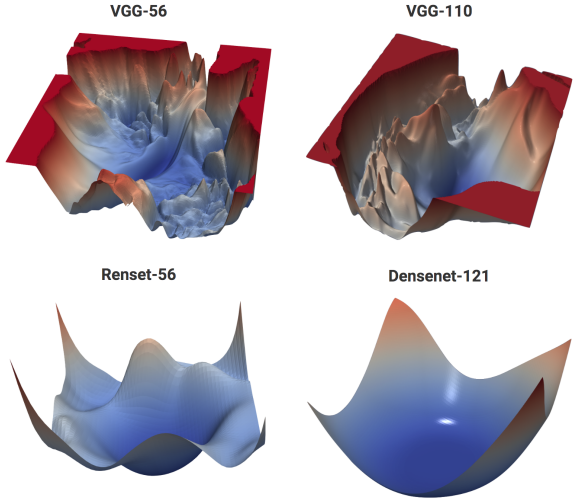


(d)

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

Risk
modelling.research

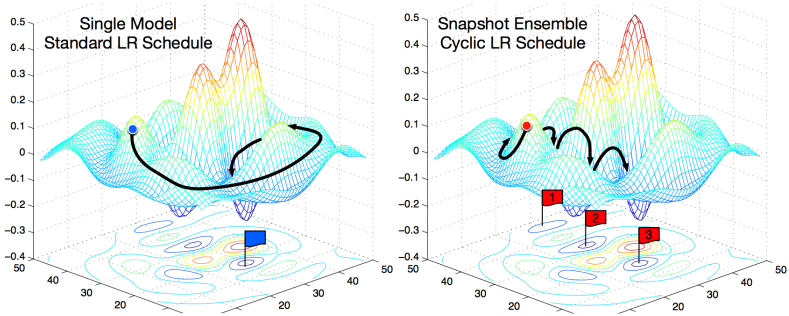
Визуализация потерь



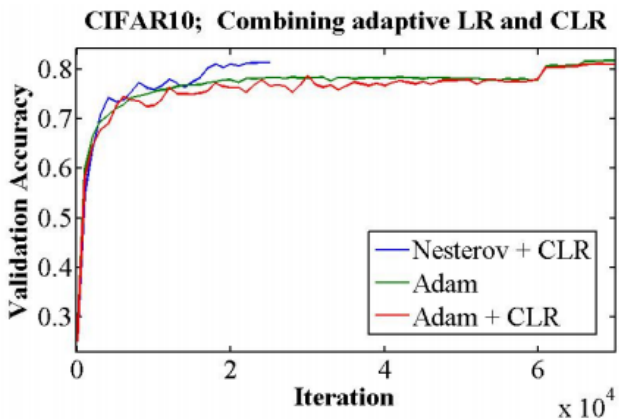
https://arxiv.org/pdf/1712.09913.pdf
https://github.com/tomgoldstein/loss-landscape

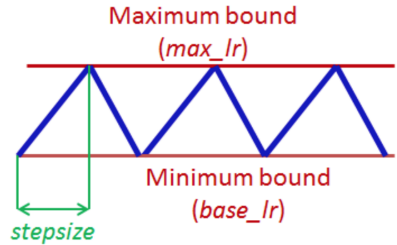
Циклическая скорость обучения (CLR)

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



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

Risk modelling.research

Так как же обучить нейросетку?

Ты необучаем!

Нейросеть — сложная функция

- Прямое распространение ошибки (forward propagation):

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

- 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.

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

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

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

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.

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

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

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

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

Risk
modelling.research

Back-propagation