Глубинное обучение

Градиентный спуск. Алгоритм обратного распространения ошибки

Даниил Водолазский

ВШЭ

30 июня 2021 г.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Содержание

Как обучать нейросеть?

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

Градиентный спуск (GD)

Проблема оптимизации:

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

Градиентный спуск (GD)

Проблема оптимизации:

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

Градиент

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

указывает направление максимального роста в точке w.

Градиентный спуск (GD)

Проблема оптимизации:

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

Градиент

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

указывает направление максимального роста в точке w.
Идём в противоположную сторону:

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

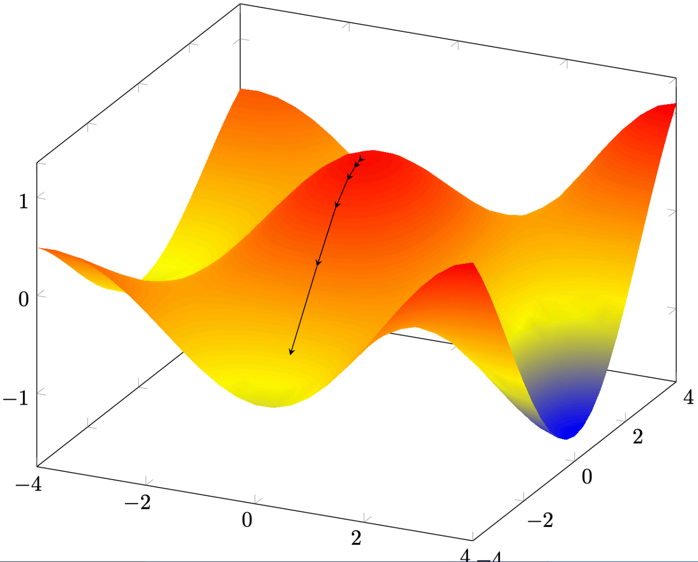
Градиентный спуск (GD)

Проблема оптимизации:

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

1. Проинициализировать $w^{(0)}$.
2. Пока $t < max\_iter$
    1. Вычислить градиент:
    $$g^{(t)} = \frac{1}{n}\sum_{i=1}^{n} \nabla L(w^{(t-1)}, x_i, y_i).$$
    2. Обновить веса:
    $$w^{(t)} = w^{(t-1)} - \eta \cdot g^{(t)}.$$
    3. Если $||w_t - w_{t-1}|| < \varepsilon$, стоп.

Градиентный спуск



Пример

Проблема оптимизации:

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

Градиент:

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

Идём в противоположную сторону:

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

Пример

Проблема оптимизации:

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

Градиент:

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

Идём в противоположную сторону:

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

Дорого постоянно считать такие суммы!

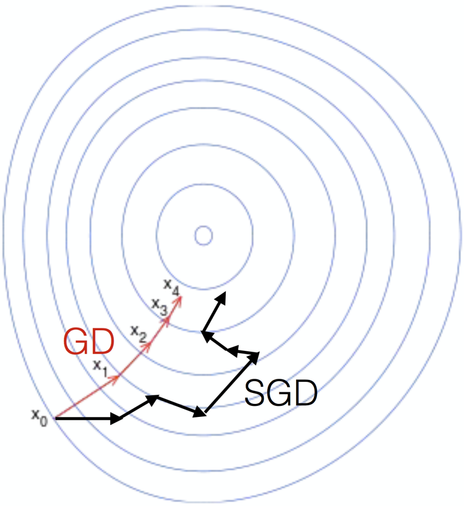
Стохастический градиентный спуск (SGD)

Проблема оптимизации:

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

1. Проинициализировать $w^{(0)}$.
2. Пока $t < max\_iter$
    1. Случайно выбрать i.
    2. Вычислить градиент:
    $$g^{(t)} = \nabla L(w^{(t-1)}, x_i, y_i).$$
    3. Обновить веса:
    $$w^{(t)} = w^{(t-1)} - \eta \cdot g^{(t)}.$$
    4. Если $||w_t - w_{t-1}|| < \varepsilon$, стоп.

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.
- Трудозатраты на одну итерацию: GD: $O(n)$, SGD: $O(1)$.
- На практике используют нечто среднее — минибатчи.

Mini-bath SGD

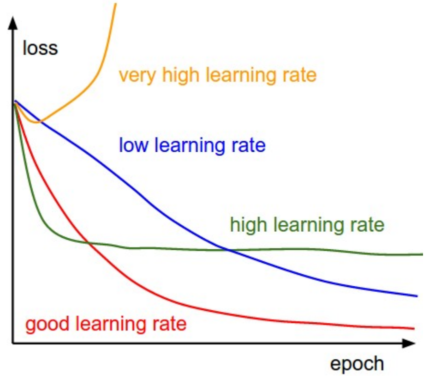
Проблема оптимизации:

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

1. Проинициализировать $w^{(0)}$.
2. Пока $t < max\_iter$
   1. Случайно выбрать $m < n$ индексов $B = \{i_1, \dots, i_m\}$.
   2. Вычислить градиент:
   $$g^{(t)} = \frac{1}{m}\sum_{i \in B} \nabla L(w^{(t-1)}, x_i, y_i).$$
   3. Обновить веса:
   $$w^{(t)} = w^{(t-1)} - \eta \cdot g^{(t)}.$$
   4. Если $||w_t - w_{t-1}|| < \varepsilon$, стоп.

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

$$g^{(t)} = \frac{1}{m} \sum_{i \in B} \nabla L(w^{(t-1)}, x_i, y_i).$$

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 $\alpha = 0.9$.

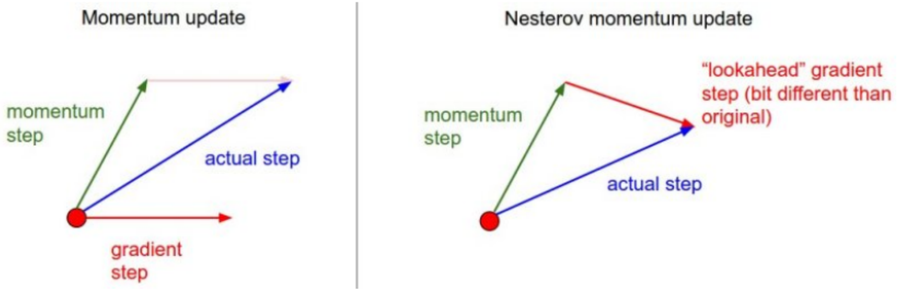
Крутой интерактив для моментума: https://distill.pub/2017/momentum/

Momentum SGD (1986)

- 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.



$$
\begin{aligned}
h^{(t)} &= \alpha \cdot h^{(t-1)} + \eta \cdot \nabla L(w^{(t-1)} - \alpha \cdot h^{(t-1)}), \\
w^{(t)} &= w^{(t-1)} - h^{(t)}.
\end{aligned}
$$

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 (2011)

AdaGrad — **Ada**ptive **Grad**ient.

$$G_j^{(t)} = G_j^{(t-1)} + (g_j^{(t)})^2,$$
$$w_j^{(t)} = w_j^{(t-1)} - \frac{\eta}{\sqrt{G_j^{(t)} + \varepsilon}} \cdot g_j^{(t)}.$$

- $g_j^{(t)}$ — градиент по j-му параметру.
- Своя скорость обучения для каждого параметра.
- Обычно $\eta = 0.01$, т. к. параметр не очень важен.
- $G_j^{(t)}$ всегда увеличивается, из-за этого обучение может рано останавливаться $\Rightarrow$ RMSprop.

RMSprop

RMSprop — **R**oot **M**ean **S**quare **Prop**agation.

$$G_j^{(t)} = \alpha \cdot G_j^{(t-1)} + (1-\alpha) \cdot (g_j^{(t)})^2$$
$$w_j^{(t)} = w_j^{(t-1)} - \frac{\eta}{\sqrt{G_j^{(t)} + \varepsilon}} \cdot g_j^{(t)}.$$

- Скорость обучения адаптируется к последнему сделанному шагу, бесконтрольного роста $G_j^{(t)}$ больше не происходит.
- RMSprop нигде не был опубликован, Хинтон просто привёл его в своей лекции, сказав, что это норм тема.
- Обычно $\alpha = 0.9$.

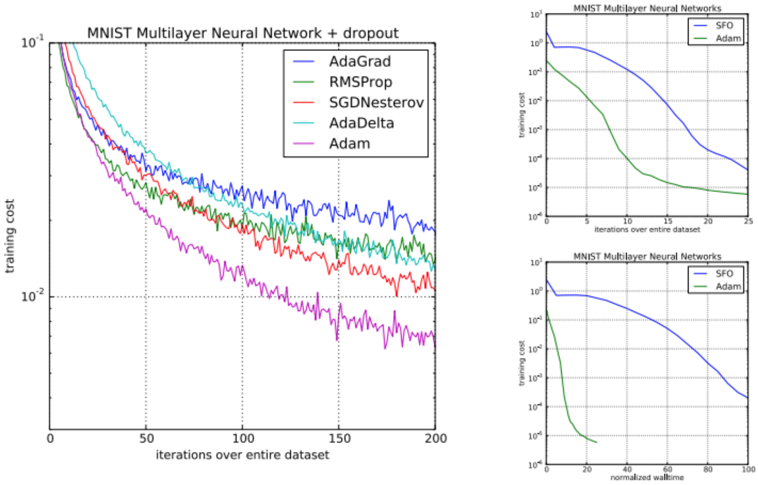
Adam (2014)

Adam — **Ada**ptive **M**oment Estimation.

$$h_j^{(t)} = \beta_1 \cdot h_j^{(t-1)} + (1 - \beta_1) \cdot g_j^{(t)}$$
$$G_j^{(t)} = \beta_2 \cdot G_j^{(t-1)} + (1 - \beta_2) \cdot (g_j^{(t)})^2$$
$$w_j^{(t)} = w_j^{(t-1)} - \frac{\eta_t}{\sqrt{G_j^{(t)} + \varepsilon}} \cdot h_j^{(t)}$$

- Комбинируем Momentum и индивидуальные скорости обучения.
- Фактически $h^{(t)}$ и $G^{(t)}$ — это оценки первого и второго моментов для стохастического градиента.
- Обычно $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Резюме по оптимизаторам. Сравнение на MNIST

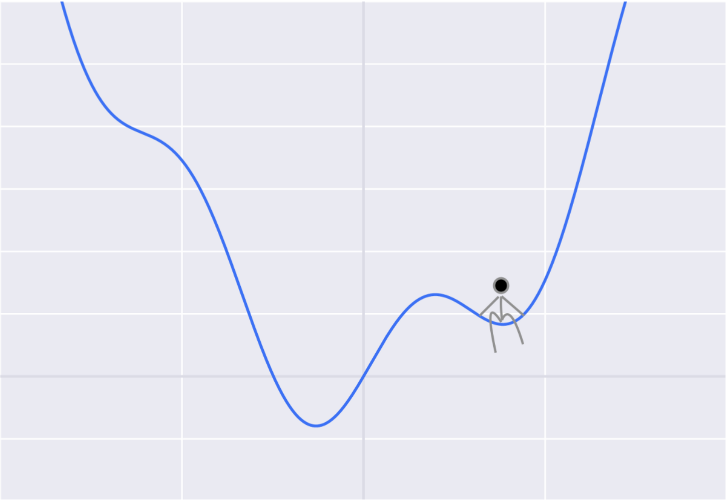


Резюме по оптимизаторам

- **GD** слишком трудозатратный, поэтому чаще пользуются **SGD**.
- **Momentum SGD** сохраняет направление шага и позволяет добиваться более быстрой сходимости.
- Адаптивные методы (**AdaGrad**, **RMSProp**) позволяют находить индивидуальную скорость обучения для каждого параметра.
- **Adam** комбинирует в себе оба подхода.
- Давайте посмотрим визуализацию 1[1] и визуализацию 2[2].
- Но это же не все вызовы!

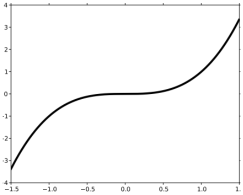
[1] http://ruder.io/content/images/2016/09/contours_evaluation_optimizers.gif
[2] http://ruder.io/content/images/2016/09/saddle_point_evaluation_optimizers.gif

Новые вызовы. Локальные минимумы

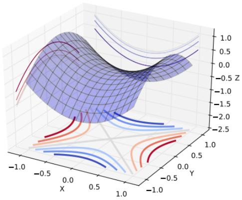


https://hackernoon.com/life-is-gradient-descent-880c60ac1be8

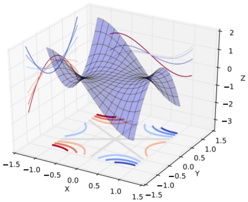
Новые вызовы. Седловые точки



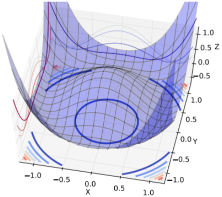
(a)



(b)



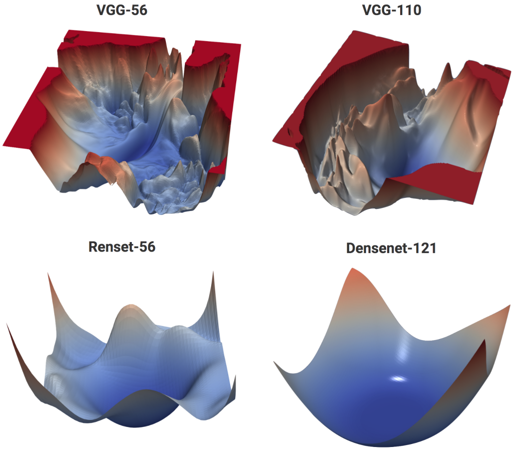
(c)



(d)

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

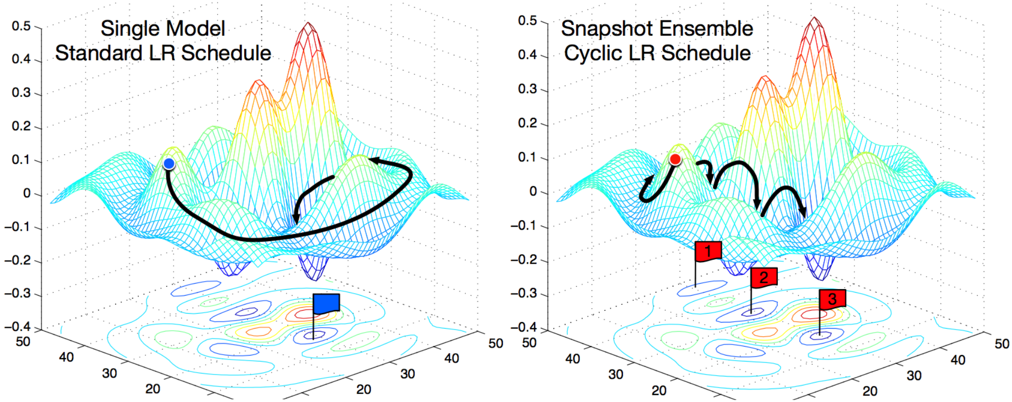
Новые вызовы. Сложные функции потерь



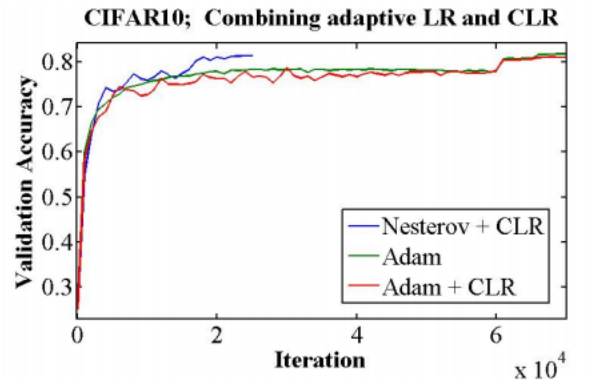
https://arxiv.org/pdf/1712.09913.pdf
https://github.com/tomgoldstein/loss-landscape

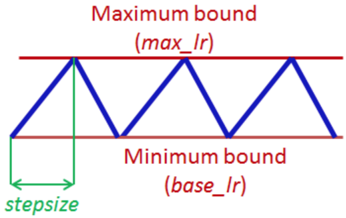
Циклическая скорость обучения (CLR)

- 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, чем Adam.
- Нет одного правильного алгоритма на все случаи!
- Всегда надо экспериментировать.

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

Содержание

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:

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

- Для обучения нужно использовать градиентный спуск.

Нейросеть — сложная функция

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

Секрет успеха в умении брать производную и градиентном спуске.

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

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

Нейросеть — сложная функция

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

Секрет успеха в умении брать производную и градиентном спуске.

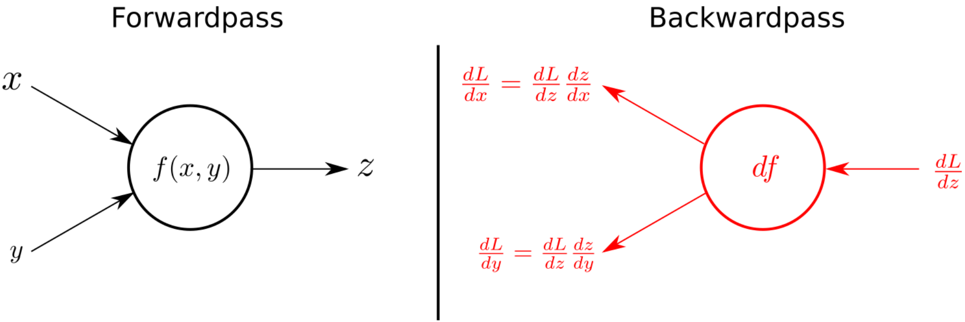
$$f(g(x))' = f'(g(x)) \cdot g'(x)$$

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

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

Дважды ищем одно и то же $\Rightarrow$ оптимизация поиска производных даст нам алгоритм обратного распространения ошибки (**backpropagation**)!

Обратное распространение ошибки



Содержание

Что узнали

- Нейронные сети обучаются градиентным спуском, причем градиент берется от функционала потерь по весам (параметрам модели).
- Существует множество модификаций GD, у каждой есть свои плюсы и минусы.
- Для борьбы с застреванием в локальных минимумах и седловых точках используется learning rate scheduler, например циклический.
- Эффективное обучение сетей возможно благодаря методу обратного распространения ошибки.