Глубокое обучение

Дмитрий Никулин

7 апреля 2021 г.

Лекция 3: градиентный спуск и backprop

Agenda

- Повторяем градиентный спуск
- Разбираемся с алгоритмом обратного распространения ошибки

Градиентный спуск

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:

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

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

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

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:

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

Градиентный спуск (GD)

Проблема оптимизации:

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

Инициализация w_0

while True:

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

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

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

break

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:

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

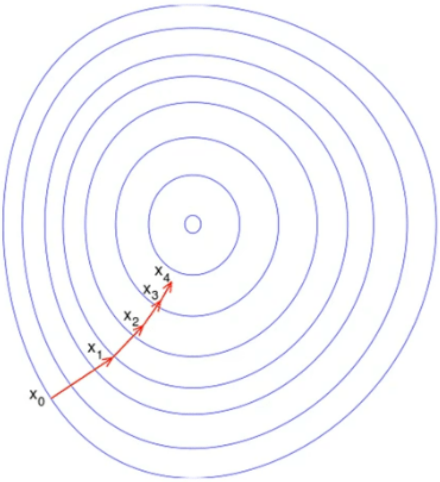
$w_t = w_{t-1} - \eta_t \cdot g_t$

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

break

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

Градиентный спуск



Пример: линейная регрессия

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_w \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^n$$

Градиент:

$$\nabla L(w) = -2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^n$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^n$$

Пример: линейная регрессия

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_w \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Градиент:

$$\nabla L(w) = -2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w) \cdot x_i \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

Дорого постоянно считать такие суммы!

Стохастический градиентный спуск (SGD)

Проблема оптимизации:

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

Инициализация w_0

while True:

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

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

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

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

    break

Пример: линейная регрессия

Проблема оптимизации:

$$L(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - x_i^T w)^2 \to \min_{w} \quad \text{по датасету } \{(x_i, y_i)\}_{i=1}^{n}$$

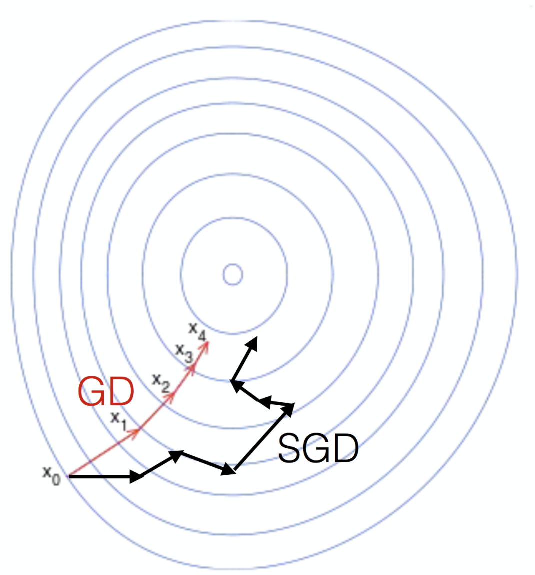
Градиент:

$$\nabla L(w) = -2 \cdot \frac{1}{b} \sum_{j \in B} (y_j - x_j^T w) \cdot x_j \quad \text{по батчу } \{(x_j, y_j)\}_{j \in B}$$

Идём в противоположную сторону:

$$w^1 = w^0 + 0.001 \cdot 2 \cdot \frac{1}{b} \sum_{j \in B} (y_j - x_j^T w_0) \cdot x_j \quad \text{по батчу } \{(x_j, y_j)\}_{j \in B}$$

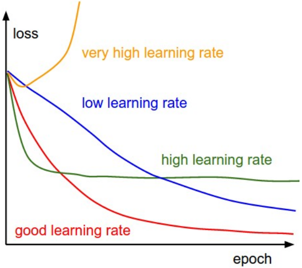
Стохастический градиентный спуск (SGD)



- И для GD и для SGD нет гарантий глобального минимума, сходимости
- SGD быстрее, на каждой итерации используется только часть выборки
- Для SGD спуск более зашумлён
- GD: $O(n)$, SGD: $O(b)$
- Шум в оценке градиента помогает выпрыгивать из локальных оптимумов

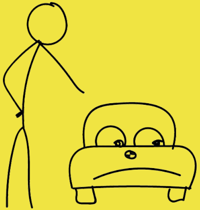
Вызовы

- 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.

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.

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

$$\frac{\partial L}{\partial W_2} = [-(y - f(X \cdot W_1) \cdot W_2)] \cdot f(X \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = [-(y - f(X \cdot W_1) \cdot W_2)] \cdot W_2^T \cdot f'(X \cdot W_1) \cdot X$$

Как обучить нейросеть?

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

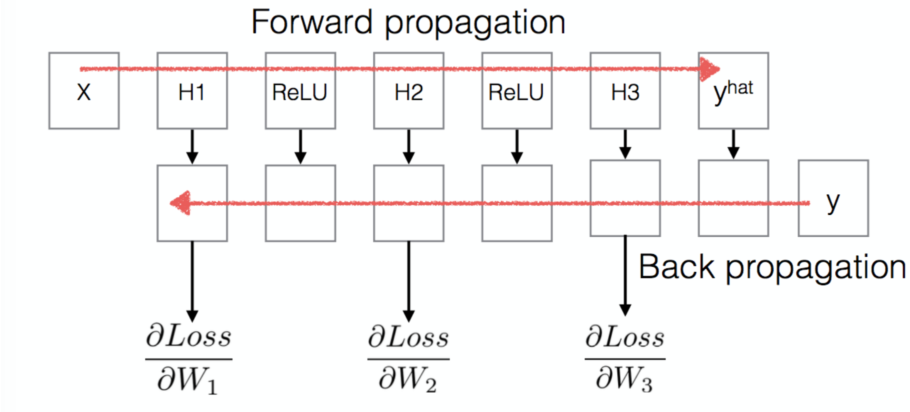
Секрет успеха в умении брать производную и градиентном спуске.

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

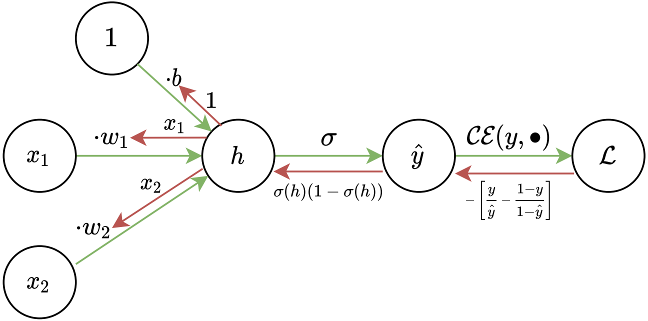
$$\frac{\partial L}{\partial W_2} = [-(y - f(X \cdot W_1) \cdot W_2)] \cdot f(X \cdot W_1)$$
$$\frac{\partial L}{\partial W_1} = [-(y - f(X \cdot W_1) \cdot W_2)] \cdot W_2^T \cdot f'(X \cdot W_1) \cdot X$$

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

Back-propagation



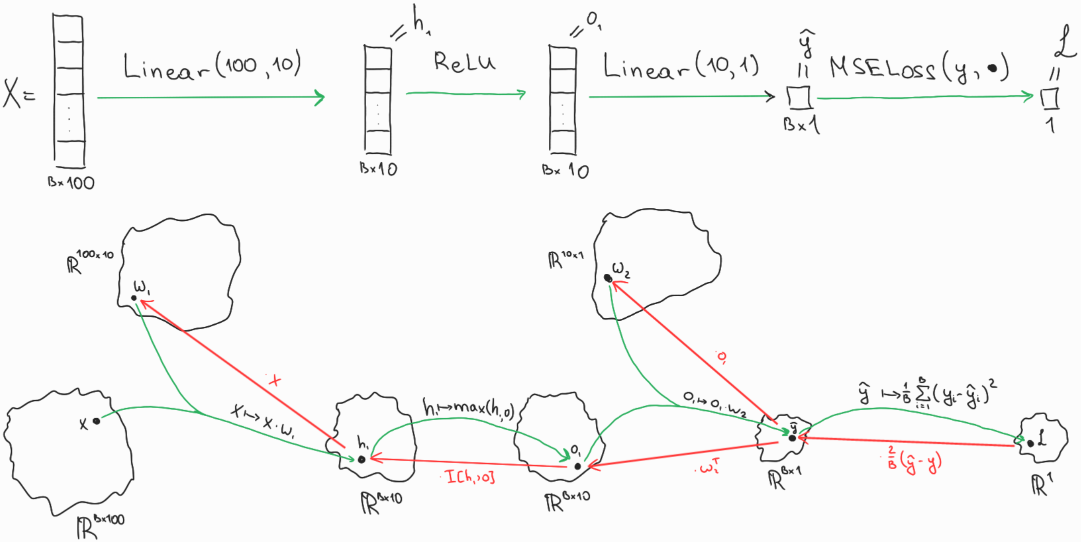
Пример: логистическая регрессия



$$\sigma(h) = \frac{1}{1+e^{-h}}$$
$$\sigma'(h) = \sigma(h) \cdot (1 - \sigma(h))$$

$$\mathcal{CE}(y, \hat{y}) = -\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$
$$\frac{\partial \mathcal{CE}}{\partial \hat{y}} = -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right]$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] \cdot \sigma(h)(1-\sigma(h))$$
$$\frac{\partial \mathcal{L}}{\partial w_1} = -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] \cdot \sigma(h)(1-\sigma(h)) \cdot x_1$$
$$\frac{\partial \mathcal{L}}{\partial w_2} = -\left[\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right] \cdot \sigma(h)(1-\sigma(h)) \cdot x_2$$



$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{2}{B}(\hat{y} - y) \cdot o_1$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{2}{B}(\hat{y} - y) \cdot w_2^T \cdot I[h_1 > 0] \cdot x$$