Deep Learning. Оптимизация и настройка нейронных сетей

Урок 4

Егор Конягин

МФТИ & АО "ЦОСиВТ"

Содержание

Алгоритмы оптимизации

Постановка задачи

Оптимизация - это процесс подбора параметров функции с целью ее минимизации. Рассмотрим формальную постановку задачи оптимизации:

$$J = J(y, \hat{y}(w, b)) \tag{1}$$

$$w, b = \arg\min J(y, \hat{y})|_{y=const} \tag{2}$$

Проблема градиентного спуска

Рассмотрим функцию $z = x^2 - y^2$:

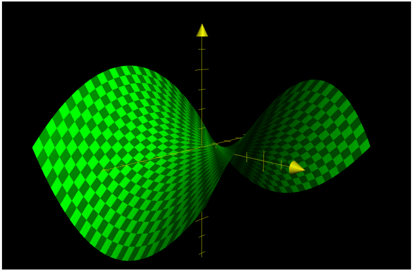

Рис. 1: График седловой функции вблизи точки (0,0)

Подсчитаем градиент данной функции:

$$\nabla z_{x,y} = (2x, 2y)^T \tag{3}$$

Видно, что в точке $(0, 0)$ $\nabla z_{(0,0)} = (0, 0)^T$. Таким образом, попав в эту точку, параметры перестанут обновляться!

Функция Розенброка

Функция Розенброка - это невыпуклая функция с одним глобальным минимумом. Она используется для оценки качества алгоритмов оптимизации. Формулой она задается следующим образом:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{4}$$

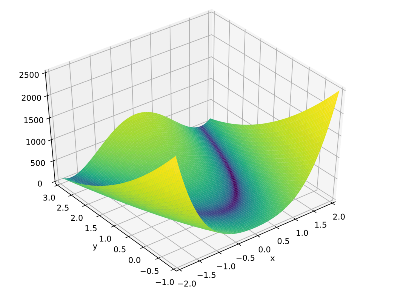

Рис. 2: График функции Розенброка

Методы оптимизации. Обзор

В задачах глубокого обучения используются градиентные методы оптимизации. Это значит, что функция потерь должна быть непрерывно дифференцируемой (на практике, их можно использовать и для кусочно-непрерывных функций). Чтобы справляться с седловыми точками, применяются различные модификации градиентного спуска с целью избегания попадания в эти точки. Сегодня мы рассмотрим следующие (наиболее популярные в современном DL) методы:

- SGD - stochastic gradient descent (стохастический градиентный спуск);
- Mini batch GD - градиентный спуск по подвыборке;
- Momentum GD - градиентный спуск с импульсом;
- RMSProp (root mean square propagation);
- AdaM - Adaptive momentum GD.

Stochastic gradient descent. Mini-batch GD

SGD. Mini-batch GD

SGD - это градиентный спуск, при котором градиент считается не по всей выборке данных, а только используя один объект, случайно выбранный:

$$J = \sum_{i=1}^{N} \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) \quad \rightarrow \quad \tilde{J} = \mathcal{L}(y^{(i^*)}, \hat{y}^{(i^*)}) \tag{5}$$

$$\frac{\partial J}{\partial w} \quad \rightarrow \quad \frac{\partial \tilde{J}}{\partial w} \tag{6}$$

$$\frac{\partial J}{\partial b} \quad \rightarrow \quad \frac{\partial \tilde{J}}{\partial b} \tag{7}$$

Mini-batch GD - это градиентный спуск, при котором градиент считается не по всей выборке данных, а по случайной подвыборке фиксированного размера (этот размер наз-ется batch size, а сама подвыборка - batch):

$$J = \sum_{i=1}^{N} \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) \quad \rightarrow \quad \tilde{J} = \sum_{j=1}^{batch\_size} \mathcal{L}(y^{(j)}, \hat{y}^{(j)}) \tag{8}$$

$$\frac{\partial J}{\partial w} \quad \rightarrow \quad \frac{\partial \tilde{J}}{\partial w} \tag{9}$$

$$\frac{\partial J}{\partial b} \quad \rightarrow \quad \frac{\partial \tilde{J}}{\partial b} \tag{10}$$

SGD. Mini-batch GD - III

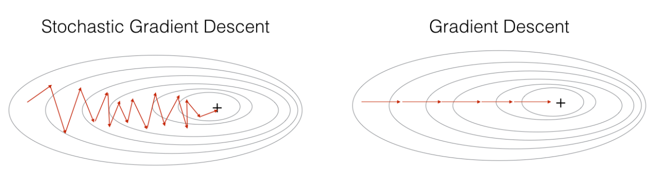

Рис. 3: SGD Vs. GD. Источник: Stanford University

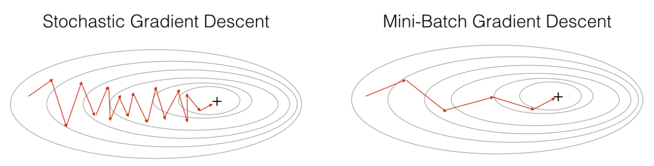

Рис. 4: Mini-batch GD Vs. GD. Источник: Stanford University

Понятие скользящего среднего

Скользящее среднее (exponentially weighted average) - это вариант аппроксимации среднего арифметического некой величины. Пусть есть последовательность $\theta = (\theta_1, \theta_2, \cdots \theta_n)$ неких измеренных величин. Тогда скользящее среднее рассчитывается по следующим формулам:

$$v_0 = 0 \tag{11}$$

$$v_1 = 0.9 \cdot v_0 + 0.1 \cdot \theta_1 \tag{12}$$

$$v_2 = 0.9 \cdot v_1 + 0.1 \cdot \theta_2 \tag{13}$$

$$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot \theta_t \tag{14}$$

Коррекция смещения: проблема смещения возникает при малых значениях t для v_t: пусть $v_0 = 0$, тогда

$$v_1 = 0 + 0.1 \cdot \theta_1 = 0.1 \cdot \theta_1 << \theta_1 \tag{15}$$

Решение:

$$v_t \quad \rightarrow \quad \frac{v_t}{1 - \beta^t} \tag{16}$$

Momentum GD

Momentum GD

Идея алгоритма: считать скользящие средние для градиентов, и использовать уже эти векторы, а не сами градиенты для обновления параметров. Суть его работы на картинке ниже:

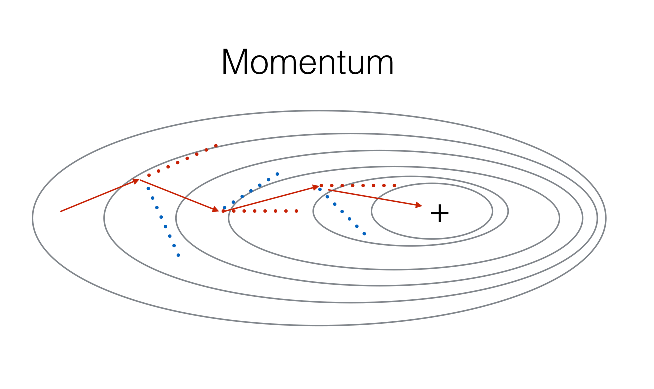

Рис. 5: Оптимизация функции с помощью MGD. Источник: Stanford University

Momentum GD. Математическое описание

Теперь рассмотрим уравнения, описывающие этот тип оптимизации. Реализация: считаем градиент на одном образце (SGD)...

$$dw = \dots \quad db = \dots$$

Далее считаем скользящее среднее:

$$V_{dw} = \beta \cdot V_{dw} + (1 - \beta) \cdot dw \tag{17}$$

$$V_{db} = \beta \cdot V_{db} + (1 - \beta) \cdot db \tag{18}$$

Далее происходит обновление весов:

$$w = w - \alpha \cdot V_{dw} \tag{19}$$

$$b = b - \alpha \cdot V_{db} \tag{20}$$

NB: β - это тоже гиперпараметр (характерные значения: 0.9)

RMSProp

RMSProp

Алгоритм RMSProp эффективно оптимизирует функции, в которых дисперсия значений по одной оси сильно отличается от дисперсии по другой.

Рассмотрим его реализацию:

- Инициализация: $S_{dw} = 0 \quad S_{db=0}$
- t-ый шаг

$$S_{dw} = \beta \cdot S_{dw} + (1 - \beta) \cdot (dw)^2 \tag{21}$$

$$S_{db} = \beta \cdot S_{db} + (1 - \beta) \cdot (db)^2 \tag{22}$$

AdaM

AdaM

AdaM - это алгоритм, комбинирующий и RMSProp, и Momentum GD. Рассмотрим его реализацию:

- Инициализация значений:

$$V_{dw} = 0, \quad S_{dw} = 0, \quad V_{db} = 0, \quad S_{db} = 0; \tag{23}$$

- t-ый шаг: $dw, db = ...,$

$$V_{dw} = \beta_1 \cdot V_{dw} + (1 - \beta_1) \cdot dw, \quad S_{dw} = \beta_2 \cdot S_{dw} + (1 - \beta_2) \cdot (dw)^2 \tag{24}$$

$$V_{db} = \beta_1 \cdot V_{db} + (1 - \beta_1) \cdot db, \quad S_{db} = \beta_2 \cdot S_{db} + (1 - \beta_2) \cdot (db)^2; \tag{25}$$

- теперь применим коррекцию смещения:

$$V_{dw}^{correct} = \frac{V_{dw}}{1 - \beta_1^t}, \quad S_{dw} = \frac{S_{dw}}{1 - \beta_2^t}$$

$$V_{db}^{correct} = \frac{V_{db}}{1 - \beta_1^t}, \quad S_{db} = \frac{S_{db}}{1 - \beta_2^t}$$

AdaM - II

Далее производим обновление параметров:

$$w = w - \alpha \frac{V_{dw}^{correct}}{\sqrt{S_{dw}^{correct} + \varepsilon}} \tag{26}$$

$$b = b - \alpha \frac{V_{db}^{correct}}{\sqrt{S_{db}^{correct} + \varepsilon}} \tag{27}$$

Как видно из описания реализации, этот алгоритм содержит много гиперпараметров: $\alpha, \beta_1, \beta_2, \varepsilon$. Типичные значения для β_1- это 0.9, для β_2- это 0.999.

Проблема переобучения. Регуляризация

Проблема переобучения

Переобучение - это чрезмерное постраивание под данные обучающей выборки, при котором ухудшается качество работы модели.

Рис. 6: Проблема переобучения

Переобучение или недообучение

Важно понимать, насколько хорошо данная модель обучена в текущей момент. Возможны три состояния:

1. оптимальное обучение;
2. недообучение (underfitting);
3. переобучение (overfitting).

Underfitting заключается в следующем: мы имеем похожее качество на train и на test - выборке, однако это качество нас не устраивает.

Overfitting характеризуется тем, что качество на train-выборке очень высокое (вплоть до 100%), в то время как на test-выборке оно может быть существенно ниже

Кривая обучения

Кривая обучения - это график зависимости функции потерь от номера итерации обучения.

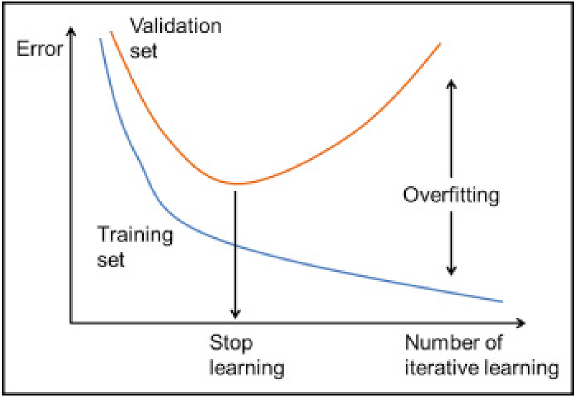

Рис. 7: Кривая обучения

Методы регуляризации

Регуляризация. обзор

Регуляризация - это наложение неких ограничений на параметры w, b с тем, чтобы избежать переобучение. Сегодня мы рассмотрим следующие алгоритмы регуляризации:

1. L2 - регуляризация (ridge regularization);
2. L1 - регуляризация (Tikhonov, lasso regularization);
3. Dropout;

L2 и L1 - регуляризация

При проведенной нормализации входных данных (т.е. приведение их к виду $[-1, 1]$) одним из симптомов переобучения являются большие по абсолютному значению веса. Идея L2 и L1 - регуляризации: увеличивать функцию потерь, если значения весов велики.

L2-регуляризация:

$$J(y, \hat{y}) = \sum \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2} \sum_{l} ||w^{[l]}||_F^2 \tag{28}$$

L1-регуляризация:

$$J(y, \hat{y}) = \sum \mathcal{L}(y^{(i)}, \hat{y}^{(i)}) + \lambda \sum_{l} \sum_{i,j} ||w_i^{[l]j}||_F \tag{29}$$

Важно учитывать, что теперь градиент функции ошибки по параметрам изменится: в него войдет слагаемое, получаемое дифференцированием второй суммы.

$$\frac{\partial J}{\partial w^{[l]}}^{*} = \frac{\partial J}{\partial w^{[l]}} + \frac{\lambda}{m} w^{[l]} \tag{30}$$

Dropout

Dropout - это техника случайного обнуления заданного количества весов с тем, чтобы увеличить стабильность работы нейросети.

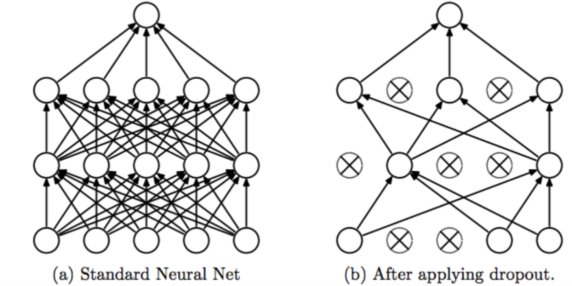

Рис. 8: Применение dropout. Источник: Stanford University

Dropout. Forward propagation

Рассмотрим, как же можно в формульном виде записать Dropout. Раньше forward propagation описывался так:

$$a^{[L]} = \sigma(W^{[L]}a^{[L-1]} + b^{[L]}) \tag{31}$$

Теперь надо выход каждого слоя поэлементно умножать на вектор, состоящий только из 1 и 0:

$$a^{[L]*} = a^{[L]} * D^{[L]}. \tag{32}$$

В итоге, в полученном выходе какие-то координаты будут равны нулю, а какие-то останутся без изменения.

Dropout характеризуется одним параметром - вероятностью.

Dropout. Backward propagation

Здесь также необходимо внести изменения: при подсчете $dA^{[L]}$ необходимо в конце этот вектор поэлементно умножить на маску $D^{[L]}$. Для этого надо не забыть сохранить ее значение, полученное в ходе forward propagation.

Помимо этого, вектор градиента надо разделить на параметр dropout-а, чтобы среднее значение выхода этого слоя оставалось неизменным.

ВАЖНО!!! Dropout применятся ТОЛЬКО на стадии обучения модели. Если модель уже обучена, dropout применять не нужно.

Summary

Сегодня мы рассмотрели следующие вещи:

- алгоритмы оптимизации:
    1. SGD;
    2. Mini-batch GD;
    3. Momentum GD;
    4. RMSProp;
    5. AdaM;
- алгоритмы регуляризации:
    1. L2-регуляризация;
    2. L1-регуляризация;
    3. Dropout.