Математика для Data Science. Математический анализ. Шпаргалка

Содержание

Пятая неделя. Модификации градиентного спуска

Градиентный спуск

Для поиска минимума функции $f : \mathbb{R}^n \to \mathbb{R}$ мы можем использовать следующий **Алгоритм градиентного спуска:**

1. Выберем какую-нибудь начальную точку $r_0 \in \mathbb{R}^n$.
2. Обозначим за i номер шага градиентного спуска. Сейчас $i = 1$.
3. Вычислим градиент $\nabla f(r_i)$.
4. Если $\nabla f(r_i) = 0$, то алгоритм останавливается. Иначе выбираем $\delta > 0$ и сдвигаемся на δ в направлении $(-\nabla f(r_i))$. Называем точку, в которую мы попадаем, r_{i+1}.
5. Заменяем i на $i+1$ и повторяем шаги 3, 4, 5.

Если в градиентом спуске мы делаем шаг на $-\lambda \nabla f(r_i)$ для некоторого положительного числа $\lambda > 0$, то такое λ называется *learning rate* или *скоростью обучения*. В таком случае в 4 пункте алгоритма $r_{i+1} = r_i - \lambda \nabla f(r_i)$.

Линейная регрессия и градиентный спуск

В задаче линейной регрессии мы представляем объекты в виде набора признаков, каждый из которых является некоторым числом. А затем пробуем найти для каждого признака коэффициент такой, чтобы при сложении признаков с данными коэффициентами мы получали что-то близкое к нашей целевой функции.

Обозначения в задаче линейной регрессии

1. Объект это x, объект номер i из обучающей выборки это $x^{(i)}$.
2. Каждый объект задаётся n признаками: $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}) \in \mathbb{R}^n$. Итого, $x_j^{(i)}$ это j-ый признак i-ого объекта.
3. Значение целевой функции на объекте $x^{(i)}$ обозначается за $y^{(i)} \in \mathbb{R}$.
4. Обучающая выборка это m пар: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$.
5. Предсказание нашей модели на объекте $x^{(i)}$ обозначается за $\hat{y}^{(i)}$.
6. Значение квадратичной функции потерь на i-ом объекте это $L(y^{(i)}, \hat{y}^{(i)}) := (\hat{y}^{(i)} - y^{(i)})^2$.
7. Значение функции потерь для всей выборки это $\frac{1}{2} \sum\limits_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$.
8. Модель задаётся набором параметров $\theta = (\theta_0, \theta_1, \ldots, \theta_n) \in \mathbb{R}^{n+1}$.
9. Для каждого объекта вводим фиктивный признак x_0, который всегда равен 1. Тогда $\hat{y} = h_\theta(x) := \sum\limits_{j=0}^{n} \theta_j x_j$.
10. Функция потерь $J(\theta) := \frac{1}{2} \sum\limits_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2} \sum\limits_{i=1}^{m} \left(\sum\limits_{j=0}^{n} \theta_j x_j^{(i)} - y^{(i)} \right)^2$. Её-то мы и будем минимизировать.

Чтобы сделать шаг градиентного спуска, нам нужно найти градиент, то есть

$$\nabla J(\theta) := \left(\frac{\partial J}{\partial \theta_0}(\theta), \frac{\partial J}{\partial \theta_1}(\theta), \ldots, \frac{\partial J}{\partial \theta_n}(\theta) \right).$$

Тогда из точки $(\theta_0, \theta_1, \ldots, \theta_n)$ после одного шага градиентного спуска с learning rate α мы попадём в точку

$$\left(\theta_0 - \alpha \frac{\partial J}{\partial \theta_0}(\theta), \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}(\theta), \ldots, \theta_n - \alpha \frac{\partial J}{\partial \theta_n}(\theta) \right).$$

Стохастический градиентный спуск

Один шаг градиентного спуска задаётся следующим преобразованием координат вектора $(\theta_0, \theta_1, \dots, \theta_n)$:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J}{\partial \theta_j}(\theta) = \theta_j - \alpha \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot x_j^{(i)} \text{ для всех } j \in \{0, 1, \dots, n\}$$

Итак, в алгортиме градиентного спуска мы на каждом шаге должны считать значения функции потерь для всех m объектов выборки.

А алгоритм *стохастического градиентного спуска (или SGD)* же выглядит следующим образом.

Мы перемешиваем наши m объектов из обучающей выборки. Делаем m шагов — по одному для каждого объекта $x^{(i)}$. На одном шаге координаты вектора θ меняются так:

$$\theta_j \leftarrow \theta_j - \alpha \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot x_j^{(i)} \text{ для всех } j \in \{0, 1, \dots, n\}$$

Потом повторяем эту процедуру k раз. Каждая такая итерация из m шагов называется одной *эпохой*. А число k соответственно называется *количеством эпох*.

Разберём теперь, как работает *mini-batch gradient descent*.

Мы выбираем число, которое будет размером *mini-batch*, например 16. Перемешиваем нашу обучающую выборку из m элементов. Разбиваем её на $\frac{m}{16}$ частей по 16 объектов. Каждая такая часть называется mini-batch. Делаем $\frac{m}{16}$ шагов — по одному шагу для каждого mini-batch. Одна такая процедура называется одной эпохой. Так же, как и в SGD мы можем сделать несколько эпох.

Градиентный спуск с моментом

Опишем алгоритм *градиентного спуска с моментом*.

Пусть $\alpha > 0$ — learning rate, а $0 < \beta < 1$ — коэффициент момента. Пусть также $V_0 = 0$.

Пусть после $(t-1)$-ого шага градиентного спуска с моментом мы находимся в какой-то точке. Обозначим за S_t градиент функции потерь в этой точке. Положим

$$V_t := \beta V_{t-1} + (1-\beta) S_t$$

или, эквивалентно,

$$V_t = (1-\beta)(\beta^{t-1} S_1 + \beta^{t-2} S_2 + \dots + \beta^2 S_{t-2} + \beta^1 S_{t-1} + S_t)$$

и сделаем шаг на $(-\alpha)V_t$.

Ещё можно делать шаг не на $(-\alpha V_t)$, а на $\frac{-\alpha}{1-\beta^t} V_t$. Величина $\frac{1}{1-\beta^t} V_t$ при этом называется *экспоненциально взвешенным средним* от $S_1, S_2, \dots, S_t$.

RMSprop

Как и раньше, обозначим через S_t градиент функции потерь на t-ом шаге. Кроме того, пусть $0 < \beta_2 < 1$ и ε — очень маленькая константа, обычно её берут равной 10^{-8}.

Пусть $s_{t,j}$ это j-ая координата вектора S_t. То есть $S_t = (s_{t,1}, s_{t,2}, s_{t,3}, \dots)$.

Обозначим $u_{t,j} := (1-\beta_2)(\beta_2^{t-1} s_{1,j}^2 + \beta_2^{t-2} s_{2,j}^2 + \dots + \beta_2^2 s_{t-2,j}^2 + \beta_2^1 s_{t-1,j}^2 + s_{t,j}^2)$.

Разделим j-ую координату S_t на $\sqrt{\frac{1}{1-\beta_2^t} u_{t,j}} + \varepsilon$. Мы делаем такую операцию с каждой координатой вектора S_t. Умножаем полученный вектор на $(-\alpha)$ и на такой вектор делаем шаг градиентного спуска.