Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации

Отчет по лабораторной работе №2

⟨Собрано 22 апреля 2023 г.⟩

Работу выполнили:
Бактурин Савелий Филиппович M32331
Вереня Андрей Тарасович M32331
Сотников Максим Владимирович M32331

Преподаватель:
TBA

Стохастический градиентный спуск

Исследование с разными размерами батча

Реализуйте стохастический градиентный спуск для решения линейной регрессии. Исследуйте сходимость с разным размером батча (1 – SGD, 2, ..., $n-1$ – Minibatch GD, n – GD из предыдущей работы).

Стохастический градиентный спуск

Стохастический градиентный спуск — модификация к основному методу итерационного поиска минимума через антиградиент дифференцируемой функции в рассматриваемой плоскости $\mathbb{R}^n$. Идея: пусть есть множество M – какой-то полный набор данных вычисленных оценок при спуске к минимуму, тогда в рассматриваемой версии мы будем брать случайное значение из выбранного $M' \subset M$. Как правило, такой подход преуменьшает вычислительные ресурсы, в особенности, следует помнить, что `float` считается крайне медленно, и ускоряет итерацию по количествам эпохам, но при этом мы теряем точность сходимости.

Пусть $x_i = \{x_i^0, x_i^1, \ldots, x_i^{n-1}\}$ – координата в $\mathbb{R}^n$ и задана функция $f(x_i) : \mathbb{R}^n \to \mathbb{R}$. Мы хотим нашу задачу свести к исследованию на некоторых специальных образцах заданной функции, для каждой точки из которых мы будем минимизировать ошибку для дальнейшего нахождения приближенного минимума рассматриваемой функции $f(x_i)$. Мы хотим найти линейную регрессию, представляющая из себя полином 1-ой степени от n переменных. Для начала мы найдем функцию ошибки S по следующей формуле:

$$S(f) = (X'^{\mathrm{T}} \times X')^{-1} \times X'^{\mathrm{T}} \times Y \times \vec{x},$$

где Y – матрица значений при множестве X (определение), X' – это матрица X, но в 1-ой колонке забитый единицами. По другому мы можем записать данную формулу следующим образом:

$$S(f) = \sum_{i=1}^{N} (y_i - x_i \cdot w_i)^2,$$

где $y_i \in D(f(x_i))$ (образ функции), $w_i \in W$ – сгенерированные веса, коэффициенты при линейной функции.

Рассмотрим идею алгоритма. При итерации, пока мы не превысили максимальное количество шагов или не сведем нашу функцию потери до некоторого ε, мы будем обобщать экспериментальные данные в виде случайных точек в некоторую многомерную линию. На каждом шаге мы будем изменять функцию потерь от измененной w.

Напишем идейный псевдокод. Скажем, что $x_i = \{x_i^0,\ x_i^1,\ \ldots, x_i^{n-1}\}$ – координата в n-мерном пространстве $\mathbb{R}^n$, $y_i = \{y_i^0,\ y_i^1,\ \ldots, y_i^{n-1}\}$ – образ функции $f(x_i)$.

```
function S(x, y, w):
    return ∑_{i=1}^{N} (y_i − x_i · w_i)^2

function stochastic_descent(x, y):
    w ← [w_i ∈ ℝ] * n
```

```
    prev ← INIT, предыдущее значение функции потери
    next ← S(x, y, w), текущее значение функции потери
    α ← const
    while |prev − next| > ε:
        prev ← next
        i ← x ∈ [0, |Y|]
        T ← [0] * n
        ∀j ∈ |w| do
            T_j ← (y_i − x_i × w) · x_i^j
        w ← w + α · T
        next ← S(x, y, w)
    return w
```

Minibatch градиентный спуск

Модификация *Minibatch* обобщает вариант стохастического градиентного спуска, тем, что во время итерации мы будем брать не одну случайную точку из посчитанных на предыдущем шаге и изменять функцию потери как бы относительно её, а теперь возьмем выборку $M' \subset M$, причем, обязательно, чтобы $|M'| > 1$ и $|M'| < |M|$.

Тогда псевдокод от предыдущего рассмотренного варианта почти ничем не отличается. Скажем также, что $x_i = \{x_i^0,\ x_i^1,\ \dots, x_i^{n-1}\}$ – координата в n-мерном пространстве $\mathbb{R}^n$, $y_i = \{y_i^0,\ y_i^1,\ \dots, y_i^{n-1}\}$ – образ функции $f(x_i)$; значение m – выступает в роли мощности подмножества M'.

```
function S(x, y, w):
    return Σ_{i=1}^{N} (y_i − x_i · w_i)^2

function stochastic_descent(x, y, m):
    w ← [w_i ∈ ℝ] * n
    prev ← INIT, предыдущее значение функции потери
    next ← S(x, y, w), текущее значение функции потери
    α ← const
    while |prev − next| > ε:
        prev ← next
        ∀i ∈ [0, m] do
            T ← [0] * n
            ∀j ∈ |w| do
                T_j ← (y_i − x_i × w) · x_i^j
            w ← w + α · T
        next ← S(x, y, w)
    return w
```

Градиентный спуск

Наконец, самый общий случай и являющийся самым быстроходным среди двух рассмотренных ранее, благодаря тому, что мы учитываем все точки $M' \equiv M$. Данный метод работает крайне медленно, но является одним из самых быстрых в сходимости к приближенной точке минимума.

Пусть m – есть мощность множества M, тогда идейным псевдокодом-решением задачи будет являться тот же код, что и для предыдущего варианта, то есть

```
function S(x, y, w):
    return \sum_{i=1}^{N} (y_i - x_i \cdot w_i)^2

function stochastic_descent(x, y, m):
    w ← [w_i ∈ ℝ] * n
    prev ← INIT, предыдущее значение функции потери
    next ← S(x, y, w), текущее значение функции потери
    α ← const
    while |prev − next| > ε:
        prev ← next
        ∀i ∈ [0, m] do
            T ← [0] * n
            ∀j ∈ |w| do
                T_j ← (y_i − x_i × w) · x_i^j
        w ← w + α · T
    next ← S(x, y, w)
    return w
```

Learning rate scheduling

Исследование различных модификаций

Nesterov

Momentum

AdaGrad

RMSProp

Adam

Сравнение модификаций

Сходимость

Траектории

Полиномиальная регрессия

Введение

Исследование различных модификация

L1

L2

Elastic регуляризация