Регрессия и стохастический градиентный спуск

Необходимые сведения

Иванов И.Е., Петюшко А.А.

МГУ

10 марта 2021 г.

План

1. Коэффициент детерминации
2. Стохастический градиентный спуск
3. Численная производная
4. Итераторы

Коэффициент детерминации

- Пусть $Y = \{y_1, \ldots, y_N\}$ — множество правильных ответов
- $Y_{pred} = \{y_{pred1}, \ldots, y_{predN}\}$ — множество предсказанных ответов

Коэффициент детерминации

R^2-коэффициент (определяющий качество предсказания):

$$R^2 = 1 - \frac{\sum\limits_i (y_i - y_{predi})^2}{\sum\limits_i (y_i - \frac{1}{N}\sum\limits_j y_j)^2}$$

Замечание. $-\infty < R^2 \leq 1$. Идеальное предсказание дает $R^2 = 1$.

Классический градиентный спуск

- **Функция потерь** для линейной регрессии (без регуляризации):
  $L(w, X_{train}) = \frac{1}{N}\sum_i (w^T \cdot x^{(i)} - y_i)^2 = \frac{1}{N}\sum_i L(w, x^{(i)}) = \frac{1}{N}\sum_i L_i(w)$
- **Задача**: минимизировать $L(w, X_{train})$ путем обучения весов w: $L(w, X_{train}) \to \min_w$

Численная оптимизация методом градиентного спуска

- **Начальное приближение**: $w^{(0)} := 0$
- **Итерация алгоритма**: $w^{(t+1)} := w^{(t)} - \eta \cdot \nabla_w L(w^{(t)}, X_{train})$
- **Градиентный шаг**: η

Проблема: сложно считать в условиях большого количества объектов в обучающей выборке.

Стохастический градиентный спуск

Алгоритм стохастического градиентного спуска

- Инициализация весов $w^{(0)}$
- Инициализация функции потерь $L(w^{(0)}, X_{train}) := \frac{1}{N} \sum_i L_i(w^{(0)})$

Итерации

- Выбор объекта $x_i \in X^m$ (например, случайным образом)
- Вычисление ошибки на данном объекте: $L_i(w^{(t)})$
- Шаг градиентного спуска: $w^{(t+1)} := w^{(t)} - \eta \cdot \nabla_w L_i(w^{(t)})$

Вариативность SGD

Инициализация

- $w_j = 0$
- $w_j = rand(-\frac{1}{2n}, \frac{1}{2n})$

Пакетный SGD

Идея: на каждом шаге использовать более надежную оценку градиента не на одном примере, а на нескольких

Итерации

- Выбор подмножества объектов мощности $1 < k < N$: $J = \{i_1, \ldots, i_k\}$
- Вычисление ошибки на этих объектах: $L_{i_1}(w^{(t)}), \ldots, L_{i_k}(w^{(t)})$
- Шаг градиентного спуска: $w^{(t+1)} := w^{(t)} - \eta \cdot \frac{1}{k} \sum_{j=1}^{k} \nabla_w L_{i_j}(w^{(t)})$

Численный градиент

Численная производная

- Пусть $x \in \mathbb{R}$
- Используем разложение Тейлора до первого порядка: $f(x+\delta) \approx f(x) + \delta f'(x)$
- Производная с помощью конечных разностей первого порядка: $f'(x) \approx \frac{f(x+\delta)-f(x)}{\delta}$
- Для более надежной оценки производной: $\delta \to 0, \delta > 0$

Численный градиент

- Пусть $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- Тогда градиентом $\nabla f(x)$ называется вектор $\nabla f(x) = (\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n})$
- Частная производная: $\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+\delta e_i)-f(x)}{\delta}$, где $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ — единичный базисный вектор с 1 на месте i

Итераторы в Python

- Нужны для упрощения навигации по элементам объекта (некоторая коллекция)
- Применяются в цикле “for i in iterator:”
- Имеют достаточно строгий синтаксис
- В задачах подразумевается, что мы будем ходить по обучающей выборке некоторое количество полных раз по кругу, после чего завершаем работу

Итераторы в Python

```
class MyIterator:
        def __iter__(self):
                return self
        def __init__(self, limit):
                self.limit = limit
                self.counter = 0
        def __next__(self):
                if self.counter < self.limit:
                        self.counter += 1
                        return 1
                else:
                        raise StopIteration
iterate = MyIterator(3)
for i in iterate:
        print(i)
```

Удачи в решении задач!