AIMasters

Классификация заемщиков линейными моделями

Курс «Машинное обучение 1»

Формулировка задания

Данное задание направлено на ознакомление с линейными моделями и градиентными методами обучения. В задании необходимо:

1. Написать на языке Python собственную реализацию линейного классификатора с произвольной функцией потерь и реализацию функции и градиента функции потерь для логистической регрессии. Реализации можно частично проверить через юнит-тесты (запускаются командой `pytest tests.py`).
2. Вывести все необходимые формулы, привести выкладки в отчёте.
3. Провести описанные ниже эксперименты с модельными данными и приложенным датасетом.
4. Написать отчёт о проделанной работе (в формате jupyter notebook).

Теоретическая часть (1 балл)

Выведите формулу градиента функции потерь для задачи бинарной логистической регрессии. Запишите вывод градиента в отчёт.

$$L(a(x), y) = \log(1 + \exp(-ya(x))), \quad y \in \{-1, 1\}, \quad a(x) \in (-\infty, \infty)$$

Реализация алгоритмов (4 балла)

Прототипы функций должны строго соответствовать прототипам, описанным в спецификации и проходить все тесты. Задание, не проходящее все тесты, приравнивается к невыполненному. При написании необходимо пользоваться стандартными средствами языка Python, библиотеками numpy, scipy и matplotlib. Библиотекой scikit-learn для реализаций пользоваться запрещается, но разрешается использовать её в процессе экспериментов. Все подробности реализации алгоритмов подробно описаны в спецификации к заданию.

Список экспериментов (10 баллов)

В приложенном ноутбуке.

Линейная модель бинарной классификации

Рассмотрим задачу бинарной классификации. Пусть дана обучающая выборка $X = (x_i, y_i)_{i=1}^N$, где $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{Y} = \{1, -1\}$. Линейная модель классификации определяется следующим образом:

$$a(x) = \text{sign}\left(\langle w, x\rangle + w_0\right), \quad \text{где } w \in \mathbb{R}^d \text{ — вектор весов, } w_0 \text{ — сдвиг.}$$

$$\text{sign}(z) = \begin{cases} -1, & z \leqslant 0 \\ 1, & z > 0 \end{cases}$$

Далее будем считать, что среди признаков есть константа. Тогда классификатор задаётся как:

$$a(x) = \text{sign}\left(\langle w, x\rangle\right)$$

Заметим, что на практике часто порог бинаризации α отдельно подбирается перебором по всем возможным значениям порога на отложенной выборке. Так как классы часто несбалансированы, выбор порога по отложенной выборке может приводить к лучшим результатам.

$$a(x) = \text{sign}\left(\langle w, x\rangle - \alpha\right)$$

Процесс обучения заключается в настройке вектора w. Величина $M_i(w) = y_i\langle w, x_i\rangle$ называется отступом (margin) объекта x_i относительно алгоритма $a(x)$. Если $M_i(w) < 0$, алгоритм допускает ошибку на объекте x_i. Чем больше отступ $M_i(w)$, тем более надёжно и правильно алгоритм классифицирует объект x_i. Пусть $\mathcal{L}(M(w))$ — монотонно невозрастающая функция, такая что $\mathbb{I}[M(w) < 0] \leqslant \mathcal{L}(M(w))$. Будем настраивать вектор весов w, оптимизируя функционал $Q(X, w)$:

$$Q(X, w) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(M_i(w)) \to \min_w$$

Метод обучения, использующий логистическую функцию потерь $\log(1 + \exp(-M))$, называется логистической регрессией. Одно из основных свойств логистической регрессии — возможность корректного оценивания вероятности принадлежности объекта к каждому из классов:

$$p(y = 1|x) = \frac{1}{1 + \exp(-\langle w, x\rangle)} = \sigma(\langle w, x\rangle))$$

Нетрудно заметить, что максимизация логарифма правдоподобия выборки при такой параметризации вероятностей эквивалентна минимизации суммы логистических функций потерь.

Обычно к оптимизируемому функционалу добавляют второе слагаемое — регуляризатор, не зависящий от данных. Второе слагаемое ограничивает вектор параметров модели тем самым уменьшая переобучение. Один из примеров регуляризации — L_2 регуляризатор:

$$Q(X, w) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(M_i(w)) + \frac{\lambda}{2}\|w\|_2^2 \to \min_w$$

Стохастический градиентный спуск

Для минимизации функционала $Q(X, w)$ можно использовать метод градиентного спуска. В этом методе выбирается начальное приближение для вектора весов w, затем запускается итерационный процесс, на каждом шаге которого вектор w изменяется в направлении антиградиента функционала $Q(X, w)$:

$$w^{(k+1)} = w^{(k)} - \eta_k \nabla_w Q(X, w) = w^{(k)} - \frac{1}{N}\eta_k \sum_{i=1}^{N} \nabla_w \mathcal{L}(M_i(w))$$

Параметр $\eta_k > 0$ — темп обучения (learning rate), который может равняться константе, а может, например, монотонно уменьшаться с течением итераций. В задании вам предлагается использовать формулу:

$$\eta_k = \frac{\alpha}{k^\beta}, \quad \text{где } \alpha, \beta \text{ — заданные константы}$$

Остановку алгоритма предлагается производить при малом изменении вектора весов или после заданного числа итераций.

Функционал $Q(X, w)$ представляет собой сумму функций потерь на каждому объекте. Вычислять градиент на каждой итерации при большом числе объектов может быть очень трудоёмко. Можно оценить градиент суммы градиентом одного слагаемого, на каждой итерации слагаемое выбирается случайно:

$i \sim unif\{N\}$

$w^{(k+1)} = w^{(k)} - \eta_k \nabla_w \mathcal{L}(M_i(w))$

Такой метод называется методом стохастического градиентного спуска. На каждой итерации можно выбирать не один объект, а небольшое подмножество (mini-batch), что ускорит сходимость алгоритма. Преимуществом алгоритма помимо маленькой вычислительной сложности является то, что на каждом шаге необходимо держать в память лишь один объект из обучающей выборки. Обратите внимание, что в случае стохастического градиентного спуска, темп обучения следует изменять только раз за эпоху (пробег метода оптимизации по всем объектам выборки).

Практический трюк. Для каждой эпохи необходимо сгенерировать случайную перестановку индексов всех объектов. Далее, на каждой итерации выбирать следующее подмножество индексов. Такой трюк ускоряет работу алгоритма.

Разностная проверка градиента

При написании собственной реализации линейной модели возникает необходимость проверить правильность её работы. Проверить правильность реализации подсчета градиента можно с помощью конечных разностей:

$$[\bigtriangledown f(w)]_i \approx \frac{f(x + \varepsilon e_i) - f(x)}{\varepsilon}$$

e_i — базисный вектор, $e_i = [0, 0, \ldots, 0, 1, 0, \ldots, 0]$, ε — небольшое положительное число.