Национальный исследовательский университет

“Высшая школа экономики”

Московский институт электроники и математики

“Прикладная математика”

Компьютерный практикум по математике

“Регрессия”

Работу выполнил

студент группы БПМ-171:

Михайлевский Станислав

Москва, 2019

Введение

В работе рассмотрены две простейшие статистические модели: регрессионная модель — линейная регрессия и модель классификации — логистическая регрессия.

Регрессия — математическое выражение, отражающее зависимость эндогенной переменной y от независимых переменных x.

Классификация — систематезированное распределение объектов на определенные классы на основании сходства и различия в признаках, описывающих объекты.

Теоретическое обоснование

Линейная регрессия

Пусть $D = \{(x_i, y_i) : \ X_i \in \mathbf{R}^n, y^i \in \mathbf{R}\}$ некоторый набор данных. Необходимо восстановить отношение между зависимой переменной y^i и независимой переменной $x_i^T = [x_1^i, \ldots, x_n^i]$.

В модели линейной регрессии отношение между X и y предполагается линейным, тогда *гипотеза*, отражающая линейную зависимость X и y может быть записана следующим образом:

$$\hat{y} = h_\theta(x) = \theta_0 + \sum_{j=1}^{n} x_j \theta_j,$$

где $\theta = [\theta_0, \theta_1, \ldots, \theta_n]^T$ — неизвестные параметры (или веса) модели. Параметр θ_0 отражает предположение о том, что данные изначально не центрированы, а сдвинуты на какую-то константу.

Если к признакам всех объектов добавить фиктивный признак равный единице, гипотезу можно переписать компактнее:

$$h_\theta(x) = \sum_{j=0}^{n} x_j \theta_j$$

или в векторном виде:

$$h_\theta(X) = X\theta,$$

где

$$X = \begin{bmatrix} 1, & x_1^1, & x_2^1, & ... & x_n^1 \\ 1 & x_1^2 & x_2^2 & ... & x_n^2 \\ ... & & ... & ... & ... \\ 1 & x_1^m & x_2^m & x_3^m & x_n^m \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ ... \\ \theta_n \end{bmatrix}$$

Теперь, чтобы регрессионная модель обучалась, необходимо определить функцию, измеряющую, насколько хорошо параметры θ восстанавливают имеющуюся зависимость. Для этого

введем среднеквадратичную функцию ошибки:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^i - y^i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$$

Приняв $\vec{y} = \left[y^1, y^2, \dots y^m\right]^T$, запишем функцию потерь в матричной форме:

$$J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

Функцию ошибок, очевидно, необходимо минимизировать. Исходя из этого получаем следующую задачу оптимизации:

$$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 \to \min_\theta$$

или

$$\frac{1}{2m}||X\theta - \vec{y}||^2 \to \min_\theta$$

Для решения оптимизационной задачи воспользуемся методом градиентного спуска. Будем итеративно обновлять веса модели, сдвигаясь в направлении антиградиента функции потерь с некоторым шагом:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i, \qquad j \in \{0, 1, \dots n\}$$

В матричной форме

$$\theta := -\alpha\nabla J(\theta) = \theta - \frac{\alpha}{m}X^T(X\theta - \vec{y})$$

Для предотвращения проблемы переобучения используется регуляризация. Введем "штраф"на большие веса модели. Теперь функция потерь будет иметь вид:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^i - y^i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

Необходимо заметить, что параметр θ_0 в последней сумме не участвует, то есть не штрафуется. Этот факт лишает возможности компактной записи функции потерь в матричном виде с сохранением введенных обозначений.

Выражения для обновления весов методом градиентного спуска с введенным регуляризатором примет вид:

$$\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_0^i$$

$$\theta_j := \theta_j - \alpha\Big[\Big(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i\Big) + \frac{\lambda}{m}\theta_j\Big], \qquad j \in \{1, 2, \dots, n\}$$

Логистическая регрессия

Рассмотрим задачу классификации. Данные теперь имеют вид $D = \{(x_i, y_i) : \ x_i \in \mathbf{R}^n, y^i \in \{0, 1\}\}$. Будем предсказывать вероятность принадлежности объекта x^i к классу y^i. То есть $h_\theta(x) = P(y = 1|x, \theta) = 1 - P(y = 0|x, \theta)$ Это, очевидно, накладывает условия на гипотезу, а именно:

$$0 \leqslant h_\theta(x) \leqslant 1$$

Воспользуемся для удовлетворения данного условия сигмоидой:

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$z = \theta^T x$$

$$h_\theta(x) = g(\theta^T x)$$

После этого необходимо установить решающую границу. Сделаем это следующим образом:

$$h_\theta(x) \geqslant 0.5 \ \rightarrow \ y = 1$$

$$h_\theta < 0.5 \ \rightarrow \ y = 0$$

Зададим для нашей гипотезы функцию потерь так, чтобы она имела единственный минимум:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^i), y^i)$$

$$\begin{cases} Cost(h_\theta(x), y) = -\log(h_\theta(x)), & y = 1 \\ Cost(h_\theta(x), y) = -\log(1 - h_\theta(x)), & y = 0 \end{cases}$$

Можем упростить запись функции $Cost(\cdot)$:

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

Окончательно функция потерь имеет вид:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log(h_\theta(x^i)) + (1 - y^i)\log(1 - h_\theta(x^i))\right]$$

В матричной форме:

$$J(\theta) = \frac{1}{m}\big(-y^T \log(g(X\theta)) - (1 - y)^T \log(1 - g(X\theta))\big)$$

Осталось лишь записать выражения для обновления весов модели:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i, \qquad j \in \{0, 1, \ldots, n\}$$

Или в матричном виде:

$$\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \vec{y})$$

Аналогично тому, как мы вводили регуляризацию для линейной регрессии, проделаем это для логистической регрессии. Начнем с функции потерь:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log(h_\theta(x^i)) + (1 - y^i)\log(1 - h_\theta(x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Выражения для градиентного спуска:

$$\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_0^i$$

$$\theta_j = \theta_j - \alpha\Big[\Big(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i\Big) + \frac{\lambda}{m}\theta_j\Big], \qquad j \in \{1, 2, \ldots n\}$$

Реализация

Модели были реализованы средствами языка *Python*. Они представлены в виде классов совместимых с функциями популярной библиотеки машинного обучения $scikit - learn$.

Линейная регрессия

Инициализация

```
def __init__(self, fit_intercept=True, max_iters=10000, gd_step=.00001,
                 penalty=True, penalty_coef=1.0)
```

fit_intercept: *булевая, опциональная, по умолчанию=Правда*

Обучать ли вес θ_0. Если выставлено как *Ложь*, к данным не будет добавлен признак из 1, и параметр θ_0 будет отсутствовать (например, данные уже отцентрированы).

max_iters: *натуральная, опциональная, по умолчанию=10000*

Максимальное количество итераций градиентного спуска.

gd_step: *вещественная, опциональная, по умолчанию=0.00001*

Шаг α градиентного спуска.

penalty: *булевая, опциональная, по умолчанию=Правда*

Использовать ли регуляризацию. Если выставлено как *Ложь*, веса не будут штрафоваться

за большие значения.

penalty_coef: *вещественная, опциональная, по умолчанию=1.0*

Коэффициент λ регуляризации.

Методы

```
def fit(self, X, y)
```

X: *многомерный массив, обязательная*

Матрица X объекты-признаки.

y: *массив, обязательная*

Вектор ответов $\vec{y}$

Возвращает вектор весов модели $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$

```
def predict(self, X)
```

X: *многомерный массив, обязательная*

Матрица X объекты-признаки.

Возвращает вектор прогнозируемых значений $\hat{y}$ для объектов из X

Логистическая регрессия

Инициализация

```
def __init__(self, fit_inercept=True, penalty='l2', penalty_coef=1.0,
                   max_iters=10000, gd_step=.00001)
```

---

fit_intercept: *булевая, опциональная, по умолчанию=Правда*

Обучать ли вес θ_0. Если выставлено как *Ложь*, к данным не будет добавлен признак из 1, и параметр θ_0 будет отсутствовать (например, данные уже отцентрированы).

penalty: *строковая: ('l2', 'l1', 'none'), опциональная, по умолчанию='l2'*

Использовать ли регуляризацию. Если выставлено как *'none'*, веса не будут штрафоваться за большие значения.

'l1' — лассо регуляризация.

'l2' — гребневая регуляризация.

penalty_coef: *вещественная, опциональная, по умолчанию=1.0*

Коэффициент λ регуляризации.

max_iters: *натуральная, опциональная, по умолчанию=10000*

Максимальное количество итераций градиентного спуска.

gd_step: *вещественная, опциональная, по умолчанию=0.00001*

Шаг α градиентного спуска.

Методы

```
def fit(self, X, y)
```

X: *многомерный массив, обязательная*

Матрица X объекты-признаки.

y: *массив, обязательная*

Вектор ответов $\vec{y}$.

Возвращает список весов модели $\theta = [\theta_1, \theta_2, \ldots, \theta_n]^T$.

```
def predict_proba(self, X)
```

X: *многомерный массив, обязательная*

Матрица X объекты-признаки.

Возвращает вероятности отнесения объектов из X к классу *1*

```
def predict(self, X)
```

X: *многомерный массив, обязательная*

Матрица X объекты-признаки.

Возвращает вектор прогнозируемых классов $\hat{y}$ для объектов из X.

Полный исходный код на github.com.