

Стохастический градиентный спуск и бета-регрессия

Олейник Михаил, группа 24.М22-мм

15 декабря 2024 г.

Весь код с комментариями и результатами можно найти в репозитории.

1 Теория

1.1 Задача линейной регрессии

Задача линейной регрессии заключается в нахождении линейной зависимости между переменной y , которая по предположению имеет некоторое нормальное распределение, и набором независимых переменных \mathbf{X} , тоже имеющим нормальное распределение. При этом мы хотим минимизировать разницу между наблюдаемыми значениями и теми, которые предсказывает построенная модель.

Постановка задачи в виде системы уравнений выглядит так:

$$y_i = \mathbf{X}_i^T \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, 2, \dots, n,$$

где y_i — значение зависимой переменной для i -го наблюдения, $\mathbf{X}_i = [1, x_{i1}, x_{i2}, \dots, x_{ip}]^T$ — вектор независимых (или почти независимых переменных, иначе вероятна ситуация сильного разброса значений коэффициентов модели при зависимых переменных) переменных, включая единичный столбец для константы, $\boldsymbol{\beta} = [\beta_0, \beta_1, \dots, \beta_p]^T$ — вектор коэффициентов модели; ε_i — случайная ошибка (шум), предполагается, что $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ и независимы между собой, n — число наблюдений.

Самый популярный метод решения этой задачи — это минимизация квадратичной ошибки. То есть находим $\boldsymbol{\beta}$, минимизируя сумму квадратов остатков:

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = \arg \min_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \mathbf{X}_i^T \boldsymbol{\beta})^2,$$

что аналогично максимизации функции правдоподобия (логарифма функции правдоподобия) для нормального распределения:

$$\ell(\boldsymbol{\beta}, \sigma^2 \mid \mathbf{y}, \mathbf{X}) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{X}_i^T \boldsymbol{\beta})^2.$$

Вообще говоря, задача зависит также от параметра σ , но обычно этот параметр считают фиксированным (понятно, что неизвестным) и просто максимизируют последнее слагаемое без множителя (константного в данных предположениях), до знака совпадающим с МНК.

1.2 Матричный вид

В таких условиях задача имеет точное решение. Модель можно записать в матричном виде:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

где $\mathbf{y} \in \mathbb{R}^n$ — вектор наблюдаемых значений, $\mathbf{X} = [\mathbf{1}^T, \mathbf{X}_1^T, \dots, \mathbf{X}_p^T]^T$ — матрица независимых признаков, $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ — вектор случайных ошибок.

Если $\mathbf{X}^T \mathbf{X}$ обратима (что гарантирует независимость признаков), то оптимальное значение $\boldsymbol{\beta}$ имеет аналитическое решение:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

И предсказание для новых данных \mathbf{X}_{new} :

$$\hat{\mathbf{y}}_{\text{new}} = \mathbf{X}_{\text{new}} \hat{\boldsymbol{\beta}}.$$

Однако данное решение может быть трудно применимым в реальности, когда мы имеем очень много наблюдений и количество строк матрицы \mathbf{X} очень велико. Тогда матричные операции будут стоить довольно много ресурсов.

1.3 Градиентный спуск

Когда нужно находить минимум некоего функционала, можно применить один из самых популярных и эффективных методов — градиентный спуск. Он находит локальный минимум функции, обновляя точку нахождения (в случае некоторой модели машинного обучения — её параметры) в направлении антиградиента функции (в МЛ — функции потерь). Далее будем говорить в терминах модели (пусть даже линейной регрессии)

Пусть дана функция $F(\boldsymbol{\theta})$, зависящая от параметров модели $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_n]^T$. Цель — найти такие $\boldsymbol{\theta}$, которые минимизируют $F(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}).$$

Первоначально задаётся стартовое значение параметров $\boldsymbol{\theta}^{(0)}$ (например, случайным образом или нулями).

Вычисляется градиент $\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, который содержит частные производные $F(\boldsymbol{\theta})$ по каждому параметру:

$$\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = \left[\frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2}, \dots, \frac{\partial F}{\partial \theta_n} \right]^T.$$

Градиент указывает направление наибольшего возрастания функции $F(\boldsymbol{\theta})$.

Обновляем параметры $\boldsymbol{\theta}$ на каждом шаге, двигаясь в направлении антиградиента (уменьшая значение функции потерь):

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta}^{(t)}),$$

где t — номер текущей итерации, $\alpha > 0$ — шаг обучения, определяет величину изменения параметров на каждом шаге.

Можно придумать несколько условий для остановки алгоритма, но мною были реализованы два варианта:

1. достигается максимальное число итераций (например, `max_step = 1000`);
2. норма разницы между значениями функции потерь в точке на последней и предпоследней итерации меньше определённого числа (например, `epsilon = 1e-5`):

$$\|\boldsymbol{\theta}^{(t)} - \boldsymbol{\theta}^{(t-1)}\| < \varepsilon.$$

1.4 Стохастический градиентный спуск

Для задачи регрессии, когда мы вычисляем градиент функции потерь, то получаем достаточно большую сумму, если n исчисляется сотнями тысяч и больше:

$$\nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^n \nabla_{\boldsymbol{\beta}} \ell(\mathbf{X}_i, y_i; \boldsymbol{\beta}),$$

где ℓ — локальная функция потерь для МНК:

$$\ell(\mathbf{X}_i, y_i; \boldsymbol{\beta}) = (y_i - \mathbf{X}_i^{\top} \boldsymbol{\beta})^2$$

Решением данной проблемы является использование модификации под названием стохастический градиентный спуск (Stochastic Gradient Descent — SGD):

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha \nabla_{\boldsymbol{\beta}} \ell(\mathbf{x}_i, y_i; \boldsymbol{\beta}),$$

где для каждой итерации вычисляется градиент только для одной локальной функции потерь для отдельного индивида.

Но этот метод не очень устойчив. Поэтому применяют Mini-batch Gradient Descent:

$$\nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) = \frac{1}{m} \sum_{i \in \tau_t} \nabla_{\boldsymbol{\beta}} \ell(\mathbf{x}_i, y_i; \boldsymbol{\beta}),$$

где $m \ll n$ — размер батча, $\tau_t = \{\tau_t^{[1]}, \dots, \tau_t^{[m]}\}$ — случайная выборка из $\{1, \dots, N\}$.

1.5 RMSProp

Данный алгоритм является сочетанием методов экспоненциального сглаживания:

$$V_0 = -\alpha \nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}), \quad V_t = \eta V_{t-1} - (1 - \eta) \alpha \nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}^{(t)}),$$

где $\eta \in (0, 1)$ — параметр сглаживания, а следующая точка вычисляется, как $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + V_t$, и адаптивного градиента:

$$\mathbb{F}_t = \sum_{i=1}^t \nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}^{(i)}) \cdot (\nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}^{(i)}))^T,$$

возьмём $\text{diag}(\mathbb{F}_t)$ и подставим вместо $-\alpha \nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta})$:

$$-\nabla_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) \odot (\text{diag}(\mathbb{F}_t) + (\varepsilon, \dots, \varepsilon)^T)^{-1/2},$$

где в качестве ε обычно берут корень из машинного нуля (10^{-8}).

Так вот, сочетание заключается в том, что вместо \mathbb{F}_t берут:

$$\mathbb{G}_t = \eta \mathbb{G}_{t-1} + (1 - \eta) \mathbb{F}_t.$$

1.6 Задача бета-регрессии

Существуют задачи предсказания, когда зависимая переменная имеет ограниченные границы значения (положительные вещественные числа, $[0, 1]$ или $(0, 1)$), тогда применяется метод обобщенных линейных моделей (GLM).

В таком случае остаётся линейная взаимосвязь между матрицей независимых признаков \mathbf{X} и векторов коэффициентов $\boldsymbol{\beta}$, но с помощью некоторой «хорошей» (дифференцируемой и легко вычисляемой) функции мы можем изменить область значений данной линейной комбинации:

$$g(\mu_i) = \mathbf{X}_i^\top \boldsymbol{\beta}$$

или

$$\mathbb{E}(\mathbf{y}|\mathbf{X}) = g^{-1}(\mathbf{X}\boldsymbol{\beta}),$$

где $g : A \rightarrow \mathbb{R}$, а μ_i среднее ожидаемое значение для каждого наблюдения (предсказанное значение). Для использования возьмём сигмоиду (или логит-функцию):

$$g(\mu_i) = \log \frac{\mu_i}{1 - \mu_i}$$

Для более удобных вычислений перепараметризуем модель $\mathbf{B}(\alpha, \beta)$: $\mu_i = \frac{\alpha}{\alpha + \beta}$ — среднее бета-распределения, $\phi = \alpha + \beta$ — дисперсионный параметр. Тогда функция плотности бета-распределения записывается так:

$$f(y_i | \mu_i, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu_i\phi)\Gamma((1 - \mu_i)\phi)} y_i^{\mu_i\phi-1} (1 - y_i)^{(1-\mu_i)\phi-1},$$

где $\Gamma(\cdot)$ — гамма-функция.

Будем минимизировать функцию максимального правдоподобия, поэтому нужно определить производную функции потерь, причём сразу запишем логарифмическую функцию правдоподобия:

$$L(\mu, \phi | y) = \sum_{i=1}^n [\log \Gamma(\phi) - \log \Gamma(\mu_i\phi) - \log \Gamma((1 - \mu_i)\phi) + (\mu_i\phi - 1) \log y_i + ((1 - \mu_i)\phi - 1) \log(1 - y_i)].$$

Будем дифференцировать и минимизировать и по параметрам $\boldsymbol{\beta}$, и по параметру ϕ , так как дисперсия ошибок уже не постоянна и взятие дисперсионного параметра константным может сильно увеличить ошибку.

Производная по μ_i :

$$\frac{\partial L}{\partial \mu_i} = \sum_{i=1}^n \left[-\phi\psi(\mu_i\phi) + \phi\psi((1 - \mu_i)\phi) + \phi \log y_i - \phi \log(1 - y_i) \right],$$

где $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ — дигамма-функция.

Производная по ϕ :

$$\frac{\partial L}{\partial \phi} = \sum_{i=1}^n \left[\psi(\phi) - \mu_i\psi(\mu_i\phi) - (1 - \mu_i)\psi((1 - \mu_i)\phi) + \mu_i \log y_i + (1 - \mu_i) \log(1 - y_i) \right].$$

Производная по β выражается через цепное правило:

$$\frac{\partial L}{\partial \beta} = \sum_{i=1}^n \frac{\partial \ell}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial \beta},$$

что для логит-функции:

$$\frac{\partial \mu_i}{\partial \beta} = \mu_i(1 - \mu_i)\mathbf{X}_i.$$

Таким образом:

$$\frac{\partial L}{\partial \beta} = \sum_{i=1}^n \left[-\phi\psi(\mu_i\phi) + \phi\psi((1 - \mu_i)\phi) + \phi \log y_i - \phi \log(1 - y_i) \right] \cdot \mu_i(1 - \mu_i)\mathbf{X}_i.$$

Остаётся только применить один из алгоритмов градиентного спуска к этим параметрам и градиенту, и мы получим бета-регрессию.

2 Практика

2.1 Линейная регрессия и смоделированные данные

Возьмём нормально распределённые независимые переменные, истинные коэффициенты $\beta = [1, 2, -1, 6, 0.3, -2]$ и составим из них линейно зависимую переменную, добавив гауссовского шума. Далее применим алгоритм линейной регрессии с модификацией градиентного спуска RMSProp, а также линейную регрессию из пакета `sklearn`. Оценивать будем по RMSE, R^2_{adj} и BIC:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

где \hat{y}_i — предсказание алгоритма. То есть это корень суммы квадратичных ошибок на предсказаниях.

$$R^2_{\text{adj}} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \cdot \frac{n - 1}{n - p - 1},$$

где p — количество параметров модели. Если обычный R^2 показывает насколько предсказания модели лучше предсказания среднего, то поправленный R^2 учитывает ещё и количество параметров модели.

$$\text{BIC} = k \cdot \log(n) - 2 \cdot \ell(\hat{\beta}),$$

где ℓ — логарифм функции правдоподобия.

Заданные параметры: $\varepsilon = 10^{-5}$, $\alpha = 10^{-2}$, batch size = 64. Алгоритм не сошёлся за 1000 итераций, которые были заданы в качестве верхней границы. Получаем следующую таблицу 1 с метриками, вычисленными на тестовой выборке (здесь и далее тестовая выборка составляет 25% от всей выборки).

Таблица 1: Сравнение RMSProp и линейной регрессии из sklearn

	RMSE	R^2_{adj}	BIC
RMSProp	0.987	0.975	843.8
sklearn	0.986	0.975	834.6

2.2 Бета-регрессия и смоделированные данные

Сначала проверим бета-регрессию на смоделированных данных. Для чего возьмём ту же матрицу \mathbf{X} и $\boldsymbol{\beta} = [0.17, 0.19, 0.46, 0.29, 0.25, 0.65]$, а \mathbf{y} смоделируем, как случайные величины, которые распределены по бета и имеют среднее $\mu_i = g^{-1}(\mathbf{X}_i\boldsymbol{\beta})$ и дисперсионный параметр $\phi = 3$.

Заметим, что сравнение будет происходить между бета-регрессией, которая была написана мной и минимизирует функцию правдоподобия бета распределения, и бета-регрессией, которую предоставляет пакет **statsmodels**. Также используемые метрики — BIC на основе соответствующей функции правдоподобия, а также взвешенная RMSE, где каждый член суммы нормируется на дисперсию соответствующего наблюдения. Получаем таблицу 2 с метриками на тестовой выборке.

Таблица 2: Сравнение RMSProp Beta и бета-регрессии из statsmodels

	WRMSE	BIC
RMSProp Beta	0.236	-106.4
statsmodels	0.237	-102.6

Кажется, что наша модель показывает себя даже лучше библиотечной, но не значительно. Можем ещё посмотреть на коэффициенты в таблице 3.

Таблица 3: Сравнение коэффициентов RMSProp Beta и бета-регрессии из statsmodels

	β_5	β_4	β_3	β_2	β_1	β_0	ϕ
True	0.17	0.19	0.46	0.29	0.25	0.65	3
RMSProp Beta	0.135	0.122	0.472	0.279	0.258	0.666	2.90
statsmodels	0.136	0.154	0.488	0.302	0.251	0.588	2.92

По ним видно, что наша модель где-то ближе к правде, чем библиотечная, где-то нет, то есть примерно равный результат.

2.3 Реальные данные

Таблица 4: Сравнение на реальных данных по метрикам

	WRMSE	BIC
RMSProp Lin	0.199	33800
sklearn	0.197	32405
RMSProp Beta	0.1908	-3454
statsmodels	0.1915	-3311

Таблица 5: Сравнение на реальных данных по коэффициентам

	β_3	β_2	β_1	β_0	ϕ
RMSProp Lin	-0.019	-0.01	-0.062	0.138	—
sklearn	2.17e-07	2.09e-08	-6.67e-05	0.178	—
RMSProp Beta	0.019	-3.2e-4	-0.249	-1.69	3.24
statsmodels	5.77e-07	-3.9e-08	-0.0003	-1.53	3.17

Теперь возьмём реальные данные из `observations.csv`. Преобразуем зависимую переменную в интервал $(0, 1)$ и будем предсказывать по трём первым столбцам.

Возьмём все четыре рассматриваемые выше регрессии и сравним по WRMSE и BIC. Сравнение по метрикам представлено в таблице 4, а по коэффициентам в таблице 5.

По обоим метрикам видно, что бета-регрессия действительно лучше, чем линейная регрессия, хотя если посмотреть по всем коэффициентам можно предположить, что модели предсказывают примерно некоторое среднее значение (почти все коэффициенты кроме того, который относится к константе близки к нулю).