Машинное обучение

Лекция 3. Линейная регрессия

Автор: Рустам Азимов

Санкт-Петербург, 2023г.

Задача предсказания

- **Задача предсказания (prediction)** — есть множество объектов с известными значениями признаков $X = [X_1, X_2, \ldots, X_m]$ и целевого признака Y, требуется научиться предсказывать значения Y для новых объектов по значениям признаков X

Объект	Рост	Вес	Пол	Возраст
Петр	177	70	1	20
Иван	140	40	1	12
Мария	165	55	0	20

Объект	Рост	Вес	Пол	Возраст
Дарья	160	45	0	?

Задача предсказания

- Предсказание значения количественного признака Y называется **задачей регрессии**
- Предсказание значения номинального (категориального) признака Y называется **задачей классификации**
- Например, предсказание признака Возраст — это задача регрессии, а предсказание признака Пол — задача классификации

План решения задачи регрессии

- Из данных выделить 2 множества: **обучающую** выборку *Train* и **тестовую** выборку *Test*
- Модель предсказания строится по объектам из *Train*
- После построения модели оцениваем её качество по объектам из *Test*
- Для оценки можно считать различные показатели точности предсказания, т.е. сравнивать правильные значения y_i целевого признака Y и предсказанные нашей моделью y'_i
- Например, считать **среднюю абсолютную ошибку**

$$MAE(y, y') = \frac{1}{n} \sum_{i=1}^{n} |y_i - y'_i|$$

Линейная регрессия

- Модель регрессии называется **линейной**, если значение предсказываемого признака Y вычисляется как сумма известных признаков $X_1, X_2, \dots, X_m$, взятых с некоторыми коэффициентами:

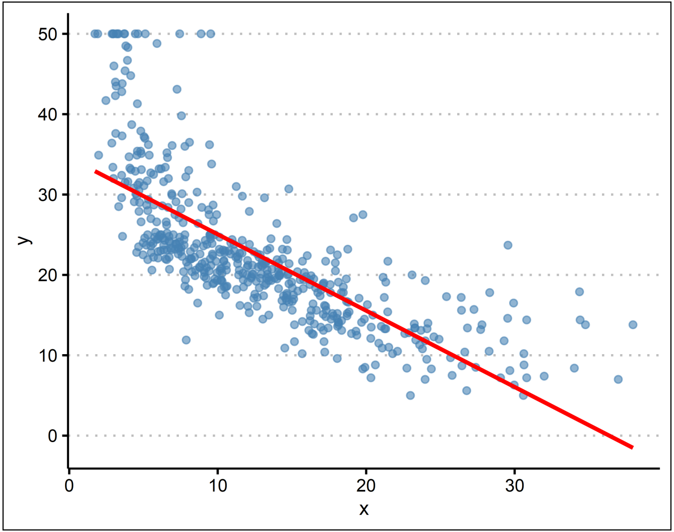
$$y' = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m + w_0$$

- Если добавить фиктивный признак $X_0 = 1$, то можно записать так: $y' = \sum_{i=0}^{m} w_i x_i$
- Матричная запись: $\vec{y'} = X\vec{w}$
- Задача заключается в нахождении оптимальных весов (коэффициентов) w_i

Линейная регрессия

- Для объектов обучающей выборки $Train$ нужно минимизировать отклонение предсказываемых значений от истинных значений признака Y
- За счёт простоты линейные модели быстро и легко обучаются, поэтому они популярны при работе с большими объёмами данных

Пример



Нахождение оптимальных весов

- Что означает оптимальность весов w_i?

Нахождение оптимальных весов

- Что означает оптимальность весов w_i?
- Для всех объектов обучающей выборки $Train$ необходимо минимизировать некоторую функцию отклонения $Q(y, y')$ предсказания y' целевого признака от его истинного значения y
- Чем плоха функция отклонения MAE?

$$Q(y, y') = MAE(y, y') = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|$$

Нахождение оптимальных весов

- Что означает оптимальность весов w_i?
- Для всех объектов обучающей выборки $Train$ необходимо минимизировать некоторую функцию отклонения $Q(y, y')$ предсказания y' целевого признака от его истинного значения y
- Чем плоха функция отклонения MAE?

$$Q(y, y') = MAE(y, y') = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|$$

- Поиск минимума осложняется взятием производной у модуля
- Поэтому на практике чаще используются **среднеквадратичное отклонение** (mean squared error) **MSE**:

$$Q(y, y') = MSE(y, y') = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$$

$RMSE$ и R^2

- MSE плохо интерпретируется, так как не сохраняет единицы измерения
- Используют корень из MSE (root mean squared error) $RMSE$:

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

$RMSE$ и R^2

- MSE плохо интерпретируется, так как не сохраняет единицы измерения
- Используют корень из MSE (root mean squared error) $RMSE$:

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

- Такие ошибки помогают контролировать процесс обучения или сравнивать качество двух моделей
- Но сами по себе среднеквадратичные ошибки не позволяют сделать выводы о модели, так как не используют интервал значений признака Y
- Удобно использовать **коэффицент детерминизации** R^2:

$$Q(y, y') = R^2(y, y') = 1 - \frac{\sum_{i=1}^{n}(y_i - y'_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Коэффицент детерминизации

- R^2 — это нормированная среднеквадратичная ошибка
- Если она близка к 1, то модель хорошо объясняет данные
- Если она близка к 0, то наши предсказания по качеству сопоставимы с константным предсказанием

Аналитическое решение

- Мы будем использовать *MSE*
- Минимизируем путём дифференцирования по вектору w, приравнивания к нулю и решения системы уравнений
- Явная формула с псевдообратной матрицей:

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

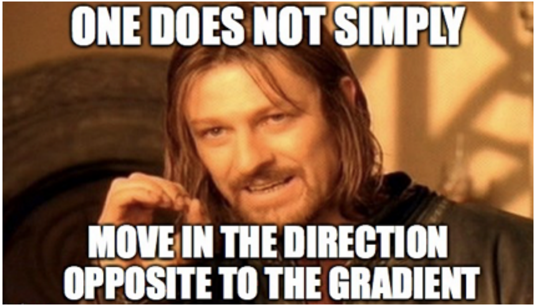
- Обращение матрицы за куб от количества признаков — дорого
- Матрица $X^T X$ может иметь нулевой или близкий к нему определитель, что приведёт к неустойчивым результатам

Проблемы

- Такие проблемы встречаются, когда между нецелевыми признаками существует сильная корреляция (проблема **мультиколлинеарности**)
- Также есть риск переобучения, если получаем большие веса
- Но аналитические решения редки в *ML*
- Рассмотрим более общий подход к задаче линейной регрессии
- Он работает не только с *MSE*, но и с другими дифференцируемыми функциями ошибок

Градиентный спуск (gradient descent)

- Оптимизационную задачу можно решать итерационно с помощью градиентных методов
- **Градиент** — вектор частных производных и направление наискорейшего роста функции в конкретной точке
- **Антиградиент** — противоположное направление наискорейшего убывания



Итерации градиентного спуска

- Стартуем с какой-то точки со значениями весов $w^{(0)}$, считаем антиградиент и сдвигаемся в его направлении
- Пересчитываем антиградиент в новой точке и снова сдвигаемся, и т.д.

$$w^{(k)} = w^{(k-1)} - \lambda_k \nabla Q(w^{(k-1)})$$

- Здесь $Q(w)$ — значение фукнции ошибки для вектора весов w, а λ_k — длина шага, используемая для контроля скорости движения
- Если шаги λ_k слишком большие, то есть вероятность перепрыгивать точку минимума
- Если шаги λ_k слишком маленькие, то движение к минимуму может занять слишком много итераций и времени
- Полезно уменьшать шаг по мере движения, например

$$\lambda_k = \frac{1}{k}$$

Остановка градиентного спуска

- Можно останавливать градиентный спуск при близости градиента к нулю
- Или при малом изменении весов $w^{(k)}$ по сравнению с $w^{(k-1)}$
- Если $Q(w)$ — выпуклая и гладкая, а также имеет минимум w^*, то имеет место следующая оценка сходимости

$$Q(w^{(k)}) - Q(w^*) = O(\frac{1}{k})$$

Полный градиент

- Функция ошибки Q — это сумма ошибок q_i на каждом из объектов обучающей выборки *Train*:

$$Q(w) = \sum_{i=1}^{n} q_i(w)$$

- Проблема в том, что на каждом шаге градиентного спуска необходимо вычислять градиент всей этой суммы
- Очень трудоёмко при больших размерах обучающей выборки
- Но на самом деле, мы можем допускать неточности в направлениях антиградиентов, так как двигаемся небольшими шагами

Полный VS Стохастический градиентный спуск

- Оценить градиент суммы поможет метод **стохастического градиентного спуска (stochastic gradient descent) SGD**:

$$w^{(k)} = w^{(k-1)} - \lambda_k \nabla q_{i_k}(w^{(k-1)})$$

- Здесь i_k — случайный выбранный номер объекта из обучающей выборки *Train*
- Для выпуклой и гладкой функции ошибки Q имеет место следующая оценка сходимости:

$$\mathbb{E}\left[Q(w^{(k)}) - Q(w^*)\right] = O(\frac{1}{\sqrt{k}})$$

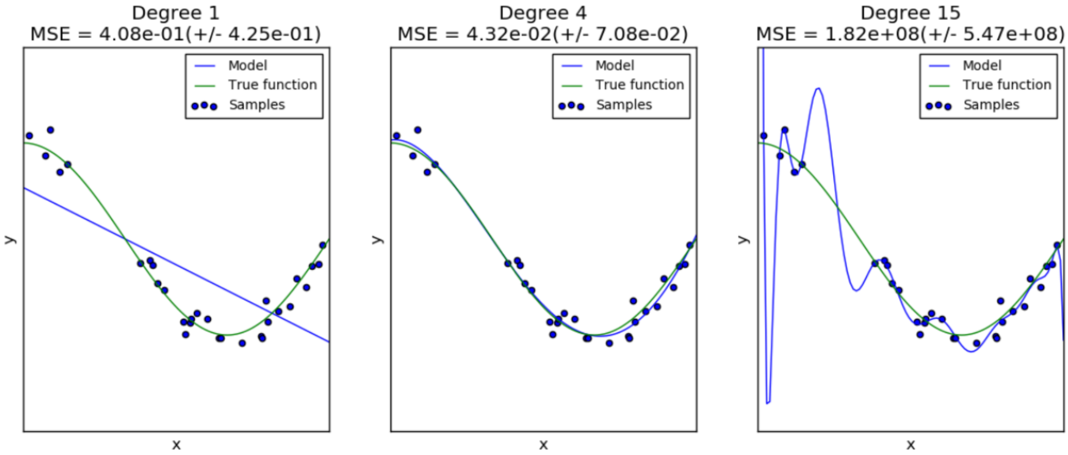
SGD

- Имеет менее трудоёмкие итерации, но и существенно меньшую скорость сходимости по сравнению с обычным (полным) градиентым спуском
- Важно, что при SGD можем держать в памяти только один объект из выборки
- Для очень больших выборок можно считывать объекты по одному и по каждому делать шаг метода SGD
- Компромисс — метод среднего стохастического градиентна (stochastic average gradient), мини пакетный градиентный спуск (mini-batch gradient descent)

Переобучение (overfitting)

- Мы можем слишком хорошо запомнить данные из обучающей выборки *Train*
- При этом на новых объектах показывать плохой результат
- Необходимо уметь оценивать обобщающую способность обученных моделей и штрафовать за излишнюю сложность
- Обычно штрафуется норма вектора весов w, чтобы уменьшить веса
- Такой подход называется **регуляризацией**

Сложность модели и переобучение



Оценка качества модели

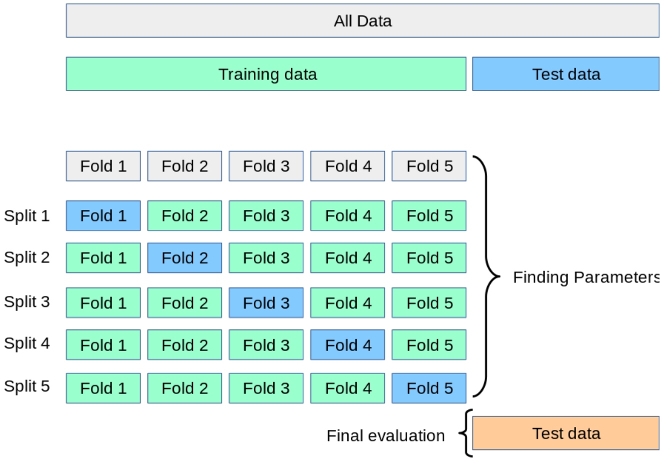
- Как раз для оценки качества модели мы и разбивали данные на две подвыборки *Train* и *Test*
- Оценивать качество будем на *Test*
- Но результат сильно зависит от разбиения

Кросс-валидация

- **Cross-validation (CV)** помогает решить проблему с разбиением на *Train* и *Test*
- Разбиваем данные на k (обычно 5) блоков $D_1, \dots, D_k$ примерно одинакового размера
- Обучаем k моделей и получаем вектора весов $w_1, \dots, w_k$, где для i-ой модели обучение происходит на всех блоках, кроме D_i
- Затем качество каждой модели оценивается на блоке, который не учавствовал в её обучении
- Результаты усредняются по формуле

$$CV = \frac{1}{k} \sum_{i=1}^{k} Q(w_i, D_i)$$

Кросс-валидация



Регуляризация

- Чтобы не переобучаться, добавим регуляризатор $4R(w)$, который штрафует за слишком большие веса

$$Q_\alpha(w) = Q(w) + \alpha R(w)$$

- Наиболее распространены L_2 и L_1-регуляризаторы
- $R(w) = ||w||_2 = \sum_{i=1}^{m} w_i^2$ (сжимает веса)
- $R(w) = ||w||_1 = \sum_{i=1}^{m} |w_i|$ (может занулить некоторые неважные признаки)
- Параметр α контролирует баланс между переобучением на обучающей выборке и штрафом за излишнюю сложность
- Нужно подбирать под каждую задачу
- Не регуляризуйте вес w_0 — он соответсвует фиктивному признаку
- $MSE + L_2$-регуляризатор — всегда решение линейной регрессии единственно

One-hot encoding

- В линейных моделях мы работаем с числовыми признаками
- Можно заменить категориальные признаки $C(x) \in \{C_1, \dots, C_m\}$ на m бинарных признаков $b_1(x), \dots b_m(x)$ (**one-hot encoding**)
- Но они линейно зависимы $(b_1(x) + \dots + b_m(x) = 1)$
- Чтобы избавится от зависимости можно, например, выбросить один из бинарных признаков

Нелинейные признаки

- Если зависимость в данных более сложная, чем линейная, то можно попробовать перейти к полиномиальным закономерностям
- Добавляем признаки соотвествующих степеней $x_1^2, x_2^2, x_1 x_2$
- Возможны и другие преобразования

Масштабирование

- До обучения полезно масштабировать признаки
- Без этого, например, могут быть проблемы с градиентным спуском
- Признаки следует масштабировать, например, путём стандартизации:

$$x_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sigma_j}$$

- Здесь μ_j и σ_j — мат. ожидание и дисперсия значений признака X_j в обучающей выборке
- Или можно масштабировать признаки на отрезок [0, 1]:

$$x_{i,j} \leftarrow \frac{x_{i,j} - \min_i x_{i,j}}{\max_i x_{i,j} - \min_i x_{i,j}}$$

Дополнительные источники

- machinelearning.ru
- scikit-learn.org
- kaggle