4 chapter
Обучение моделей

Линейная модель

Как правило, **линейная модель** вырабатывает прогноз, просто вычисляя взвешенную сумму входных признаков плюс константы под названием **член смещения (Bias term)**, также называемой **свободным членом (intercept term)**, как показано в уравнении 4.1.

Уравнение 4.1. Прогноз линейной регрессионной модели

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- $\hat{y}$ — спрогнозированное значение.
- n — количество признаков.
- x_i — значение i-того признака.
- θ_j — j-тый параметр модели (включая член смещения θ_0 и веса признаков $\theta_1, \theta_2, \ldots, \theta_n$).

Уравнение 4.4. Нормальное уравнение

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- $\hat{\theta}$ — значение θ, которое сводит к минимуму функцию издержек.
- $\mathbf{y}$ — вектор целевых значений, содержащий от $y^{(1)}$ до $y^{(m)}$.

Градиентный спуск

Градиентный спуск представляет собой самый общий алгоритм оптимизации, способный находить оптимальные решения широкого диапазона задач. Основная идея градиентного спуска заключается в том, чтобы итеративно подстраивать параметры для сведения к минимуму функции издержек.

Когда используется градиентный спуск, вы должны обеспечить наличие у всех признаков похожего масштаба (например, с применением класса StandardScaler из Scikit-Learn), иначе он потребует гораздо большего времени на схождение

Пакетный градиентный спуск

Уравнение 4.5. Частные производные функции издержек

$$\frac{\partial}{\partial \theta_j}\mathrm{MSE}(\theta) = \frac{2}{m}\sum_{i=1}^{m}\left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)x_j^{(i)}$$

Уравнение 4.6. Вектор-градиент функции издержек

$$\nabla_\theta \mathrm{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0}\mathrm{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1}\mathrm{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathrm{MSE}(\theta) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Уравнение 4.7. Шаг градиентного спуска

$$\theta^{(\text{следующий шаг})} = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$$

Стохастический градиентный спуск

Главная проблема с пакетным градиентным спуском - тот факт, что он использует полный обучающий набор для вычисления градиентов на каждом шаге, который делает его очень медленным в случае крупного обучающего набора. Как противоположная крайность, *стохастический градиентный спуск (Stochastic Gradient Descent - SGD)* на каждом шаге просто выбирает из обучающего набора случайный образец и вычисляет градиенты на основе только этого единственного образца. Очевидно, алгоритм становится гораздо быстрее, т.к. на каждой операции ему приходится манипулировать совсем малым объемом данных.

С другой стороны, из-за своей стохастической (т.е. случайной) природы этот алгоритм гораздо менее нормален, чем пакетный градиентный спуск: вместо умеренного понижения вплоть до достижения минимума функция издержек будет скачками изменяться вверх и вниз, понижаясь только в среднем.

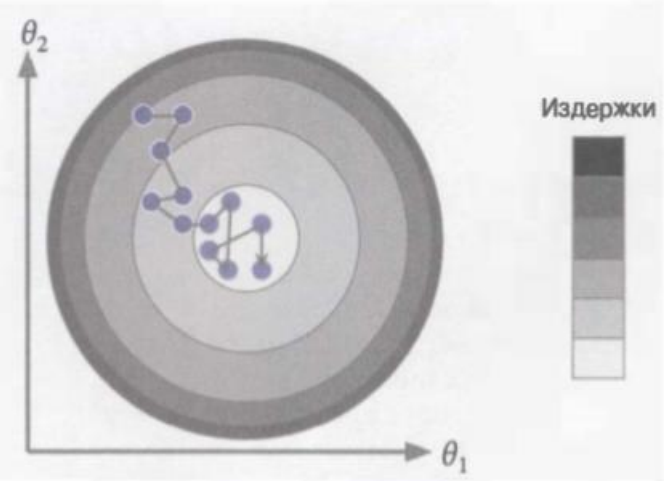

Рис. 4.9. Стохастический градиентный спуск

Мини-пакетный градиентный спуск

На каждом шаге вместо вычисления градиентов на основе полного обучающего набора (как в пакетном градиентном спуске) или только одного образца (как в стохастическом градиентном спуске) мини-пакетный градиентный спуск вычисляет градиенты на небольших случайных наборах образцов, которые называются *мини-пакетами (mini-batch).*

Таблица 4.1. Сравнение алгоритмов для линейной регрессии

Алгоритм	Большое m	Поддерживает ли внешнее обучение	Большое n	Гиперпараметры	Требуется ли масштабирование	Scikit-Learn
Нормальное уравнение	Быстрый	Нет	Медленный	0	Нет	LinearRegression
Пакетный градиентный спуск	Медленный	Нет	Быстрый	2	Да	–
Стохастический градиентный спуск	Быстрый	Да	Быстрый	≥2	Да	SGDRegressor
Мини-пакетный градиентный спуск	Быстрый	Да	Быстрый	≥2	Да	SGDRegressor

Полиномиальная регрессия

Что, если ваши данные в действительности сложнее обычной прямой линии? Удивительно, но вы на самом деле можете применять линейную модель для подгонки к нелинейным данным. Простой способ предполагает добавление степеней каждого признака в виде новых признаков и последующее обучение линейной модели на таком расширенном наборе признаков. Этот прием называется **полиномиальной регрессией (polynomial regression).**

Компромисс между смещением и дисперсией

Важным теоретическим результатом статистики и машинного обучения является тот факт, что ошибка обобщения модели может быть выражена в виде суммы трех очень разных ошибок.

Смещение. Эта часть ошибки обобщения связана с неверными предположениями, такими как допущение того, что данные линейные, когда они на самом деле квадратичные. Модель с высоким смещением, скорее всего, недообучится на обучаемых данных [10].

Дисперсия. Эта часть объясняется чрезмерной чувствительностью модели к небольшим изменениям в обучающих данных. Модель со многими степенями свободы (такая как полиномиальная модель высокой степени), вероятно, будет иметь высокую дисперсию и потому переобучаться обучающими данными.

Неустранимая погрешность. Эта часть появляется вследствие зашумленности самих данных. Единственный способ сократить неустранимую погрешность в ошибке предусматривает очистку данных (например, приведение в порядок источников данных, таких как неисправные датчики, или выявление и устранение выбросов).

Возрастание сложности модели обычно увеличивает ее дисперсию и уменьшает смещение. И наоборот, сокращение сложности модели увеличивает ее смещение и уменьшает дисперсию. Вот почему это называется компромиссом.

Если согласно метрикам перекрестной проверки модель хорошо выполняется на обучающих данных, но плохо обобщается, то модель переобучена. Если модель плохо выполняется в обоих случаях, тогда она недообучена. Так выглядит один из способов сказать, что модель слишком проста или чрезмерно сложна

Регуляризированные линейные модели

Как было показано в главах 1 и 2, хороший способ сократить переобучение заключается в том, чтобы регуляризировать модель (т.е. ограничить ее): чем меньше степеней свободы она имеет, тем труднее ее будет переобучить данными. Например, простой метод регуляризации полиномиальной модели предполагает понижение количества полиномиальных степеней.

Гребневая регрессия (ridge regression)

```
from sklearn.linear_model import Ridge
```

Гребневая регрессия (также называемая регуляризацией Тихонова) является регуляризированной версией линейной регрессии: к функции издержек добавляется **член регуляризации (regularization term)**, равный: $\alpha\sum_{i=1}^{n}\theta_i^2$.

Гиперпараметр α управляет тем, насколько необходимо регуляризировать модель. Когда $\alpha = 0$, гребневая регрессия оказывается просто линейной регрессией. При очень большом значении α все веса становятся крайне близкими к нулю, и результатом будет ровная линия, проходящая через середину данных.

Уравнение 4.8. Функция издержек для гребневой регрессии

$$J(\theta) = \text{MSE}(\theta) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

Перед выполнением гребневой регрессии важно масштабировать данные (скажем, посредством ***StandardScaler***), т.к. она чувствительна к масштабу входных признаков. Это справедливо для большинства регуляризированных моделей.

Лассо-регрессия

```
from sklearn.linear_model import Lasso
```

Регрессия методом наименьшего абсолютного сокращения и выбора (least absolute shrinkage and selection operator (lasso) regression)

Уравнение 4.10. Функция издержек для лассо-регрессии

$$J(\theta) = \text{MSE}(\theta) + \alpha\sum_{i=1}^{n}|\theta_i|$$

Важной характеристикой лассо-регрессии является то, что она стремится полностью исключить веса наименее важных признаков (т.е. устанавливает их в ноль).

Другими словами, лассо-регрессия автоматически выполняет выбор признаков и выпускает разреженную модель (т.е. с незначительным числом ненулевых весов признаков).

Эластичная сеть

```
from sklearn.linear_model import ElasticNet
```

Эластичная сеть -это серединная точка между гребневой регрессией и лассо-регрессией.

Член регуляризации представляет собой просто смесь членов регуляризации гребневой регрессии и лассо-регрессии, к тому же можно управлять отношением смеси *r*. При *r = 0* эластичная сеть эквивалентна гребневой регрессии, а при *r = 1* она эквивалентна лассо-регрессии.

Уравнение 4.12. Функция издержек эластичной сети

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^{n} \theta_i^2$$

Почти всегда предпочтительнее иметь хотя бы немного регуляризации, поэтому в целом вам следует избегать использования обыкновенной линейной регрессии. Гребневая регрессия - хороший вариант по умолчанию, но если вы полагаете, что только несколько признаков будут фактически полезными, то должны отдавать предпочтение лассо-регрессии или эластичной сети, поскольку, как уже обсуждалось, они имеют тенденцию понижать веса бесполезных признаков до нуля. В общем случае эластичная сеть предпочтительнее лассо-регрессии, т.к. лассо-регрессия может работать с перебоями, когда количество признаков больше числа обучающих образцов или некоторые признаки сильно связаны.

Логистическая регрессия (Logistic regression)

Логистическая регрессия (также называемая *логит-регрессией (logit regression))* обычно используется для оценки вероятности того, что образец принадлежит к определенному классу (например, какова вероятность того, что заданное почтовое сообщение является спамом?). Если оценочная вероятность больше 50%, тогда модель прогнозирует, что образец принадлежит к данному классу (называемому по "1") (положительным классом, помеченным , а иначе - что не принадлежит т.е. относится к отрицательному классу, помеченному "0"). Это делает ее двоичным классификатором.

Итак, каким образом все работает? Подобно линейной регрессионной модели логистическая регрессионная модель подсчитывает взвешенные суммы входных признаков (плюс член смещения), но взамен выдачи результата напрямую, как делает линейная регрессионная модель, он выдает логистику (logistic) результата (уравнение 4.13).

Уравнение 4.13. Логистическая регрессионная модель оценивает вероятность (векторизованная форма)

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

Логистика, также называемая ***логитом (logit)*** и обозначаемая $\sigma(\cdot)$, представляет собой сигмоидальную (т.е. S-образной формы) функцию, которая выдает число между 0 и 1. Она определена, как показано в уравнении 4.14 и на рис. 4.21

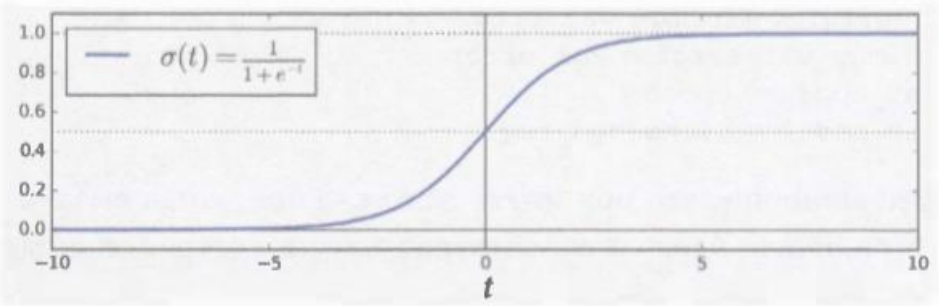

Рис. 4.21. Логистическая функция

Уравнение 4.14. Логистическая функция

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

Уравнение 4.15. Прогноз логистической регрессионной модели

$$\hat{y} = \begin{cases} 0, \text{ если } \hat{p} < 0.5, \\ 1, \text{ если } \hat{p} \geq 0.5. \end{cases}$$

```
from sklearn.linear_model import LogisticRegression
```