Урок 4

Матричные разложения

4.1. Матричные разложения

4.1.1. Постановка задачи матричного разложения

В задаче матричного разложения требуется приблизить матрицу произведением двух других(рисунок 4.1):

$$X_{l,n} \approx U_{l,k} \cdot V^T_{k,n},$$

где матрицы U и V задают новое признаковое описание объектов и изначальных признаков.

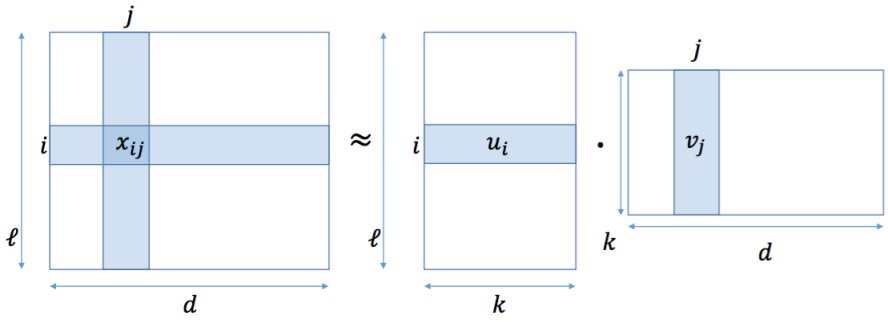

Рис. 4.1: Матрицы X, U, V

Формально задачу можно свести к поиску минимума:

$$\|X - U \cdot V^T\| \to \min.$$

Если использовать норму Фробениуса:

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2},$$

то задача записывается следующим образом:

$$\sum_{i,j} (x_{ij} - \langle u_i, v_j \rangle)^2 \to \min,$$

где u_i — новое описание объекта i, а v_j — новое описание признака j.

4.1.2. Сингулярное разложение

Сингулярное разложение — это способ представить некоторую исходную матрицу в виде произведения трех других:

$$X = U\Sigma V^T,$$

где U — ортогональная матрица, Σ — диагональная матрица, V — ортогональная матрица.

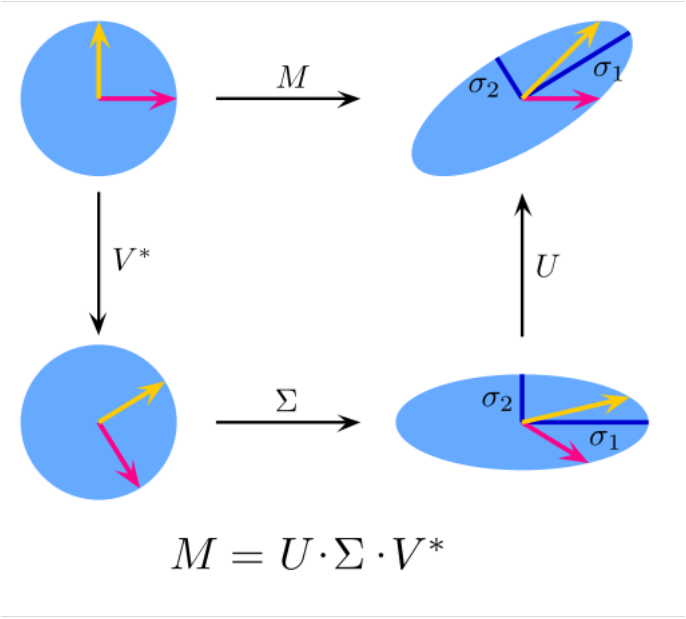

Рис. 4.2: Геометрическая интерпретация сингулярного разложения

Геометрически это можно проиллюстрировать так (рисунок 4.2): если рассматривать матрицу X как задающую некоторое отображение, то оно раскладывается на составляющие: сначала пространство поворачивают, потом растягивают вдоль осей координат, а затем снова поворачивают (говоря более точно, здесь могут быть не только повороты, но и вращения некоторых осей, но в упрощённом виде это выглядит так).

Сингулярное разложение матриц может быть полезно для рассматриваемой задачи. Можно рассмотреть сингулярное разложение матрицы X:

$$X = \tilde{U}\Sigma\tilde{V}^T.$$

Здесь приходится прибегнуть к обозначениям $\tilde{U}$ и $\tilde{V}$, чтобы не было пересечения с обозначениями для разложения на две матрицы U и V.

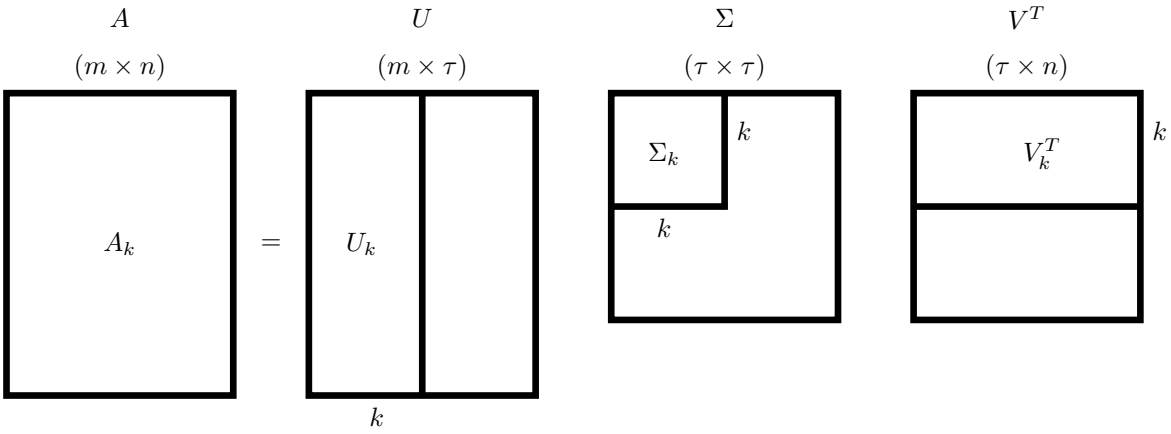

Рис. 4.3: Усечённые матрицы из SVD

Итак, с помощью сингулярного разложения получается представление исходной матрицы в виде произведения трех матриц. Теперь у каждой матрицы можно взять какую-то часть (рисунок 4.3): от матрицы $\tilde{U}$ — первые k столбцов, от матрицы Σ — квадрат размера $k \times k$, от матрицы $\tilde{V}^T$ — первые k строк. Таким образом определяются усеченные матрицы сингулярного разложения:

$$\tilde{U}_k, \quad \Sigma_k, \quad \tilde{V}_k.$$

Оказывается, если в задаче матричного разложения взять в качестве искомых матриц

$$U = \tilde{U}_k \Sigma_k, \quad V = \tilde{V}_k,$$

то это будет решением оптимизационной задачи, которое даст наилучшее приближение исходной матрицы по норме Фробениуса. При этом в качестве искомых матриц можно использовать другие:

$$U = \tilde{U}_k, \quad V = \tilde{V}_k \Sigma_k$$

или

$$U = \tilde{U}_k \sqrt{\Sigma_k}, \quad V = \tilde{V}_k \sqrt{\Sigma_k}.$$

Это указывает на неоднозначность решения задачи матричного разложения.

4.1.3. Использование термина SVD в машинном обучении

Так или иначе, часто оказывается, что производить SVD (т.е. сингулярное) разложение — не очень быстрая операция, и, в принципе, можно мыслить о задаче получения матриц U и V как о задаче оптимизации

$$\sum_{i,j} (x_{ij} - \langle u_i, v_j \rangle)^2 \to \min,$$

и подбирать u_i и v_j каким-либо оптимизационным методом. В таком случае оказывается, что SVD уже не имеет отношения к задаче, она решается напрямую. Но другой стороны, известно, что задача связана с SVD, и решение этой задачи можно применить к вычислению усечённых матриц из SVD. Поэтому в машинном обучении термин SVD прижился как обозначение решения оптимизационной задачи, указанной выше, и встречая этот термин, нужно помнить, что речь не идёт об обычном сингулярном разложении.

4.1.4. Практическое применение матричного разложения

Матричные разложения часто используются в рекомендательных системах, например, при рекомендации фильмов, когда необходимо заполнить неизвестные значения в матрице произведением векторов, задающих интересы пользователя и параметры фильма. Похожий подход применяется в задаче анализа текстов.

4.1.5. Используемые обозначения

Для полной ясности, необходимо вернуться к обозначениям. Дело в том, что здесь встречается не просто представление матрицы

$$X = UV$$

а в виде

$$X = UV^T.$$

Возникает вопрос: зачем понадобилось транспонировать? Это сделано из желания сделать так, чтобы обе матрицы, U и V представляли собой матрицы размера количество каких-то сущностей (либо объектов, либо исходных признаков) на количество новых признаков k (рисунок 4.4).

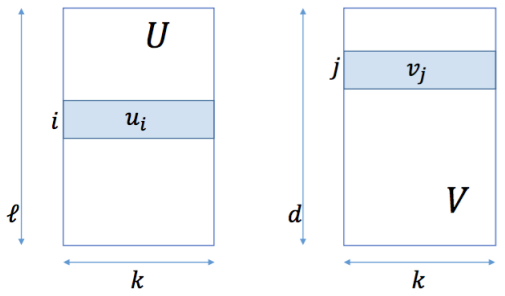

Рис. 4.4: Матрицы U, V

Помимо этого, можно вспомнить запись для x_{ij} в виде скалярного произведения векторов:

$$x_{ij} \approx \langle u_i, v_j \rangle$$

Обычно, когда речь идёт о скалярном произведении, имеется в виду скалярное произведение векторов, которые обычно представляются в виде столбцов. Из этих соображений становится понятно, почему можно встретить запись

$$x_{ij} = u_i^T v_j.$$

Это просто расписанное скалярное произведение, сумма покоординатных произведений векторов. Однако эта же запись может вызвать вопросы, потому что изначально u_i — это строка в матрице U, а не столбец, и v_j — это строка в матрице V (рисунок 4.4). Не стоит путаться из-за этого и нужно понимать, что имеется в виду именно покоординатное произведение u_i и v_j, которые были строчками в матрицах U и V.

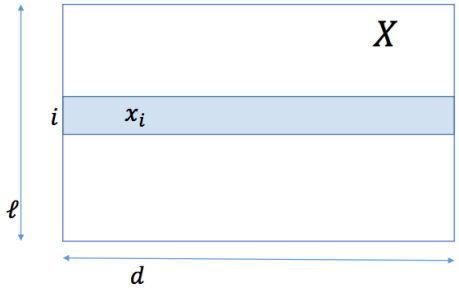

Рис. 4.5: Матрица X

Аналогичная ситуация встречалась, когда шла речь линейных моделях. В этой задаче также присутствовала матрица признаков X (рисунок 4.5), в которой строка i соответствовала объекту с номером i. В то же время встречалась следующая запись скалярного произведения:

$$\langle w, x_i \rangle = w^T x_i$$

Хотя в действительности x_i является строчкой, в этой записи подразумевается столбец.

Кроме того, в матричных разложениях встречается много различных обозначений в зависимости от того, для чего они применяются:

- $X \approx UV^T$ — часто встречаемое в рекомендациях обозначение;
- $X \approx PQ^T$ — это обозначение также часто можно встретить в рекомендациях;
- $X \approx WH$ — встречается в неотрицательных матричных разложениях и в анализе текстов;
- $X \approx \Phi\Theta$ — используется в вероятностных тематических моделях.

4.2. SGD и ALS

4.2.1. Применение стохастического градиентного спуска в задаче матричного разложения

Задача, описанная в предыдущей части, — это минимизация функционала

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \to \min_{u_i, v_j}.$$

Этот функционал можно минимизировать методом градиентного спуска. Для этого требуется выписать градиент по u_I:

$$\frac{\partial Q}{\partial u_i} = \sum_{i,j} \frac{\partial}{\partial u_i} (\langle u_i, v_j \rangle - x_{i,j})^2 = \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) \frac{\partial \langle u_i, v_j \rangle}{\partial u_i} == \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) v_j.$$

Из этого можно получить значение ошибки на x_{ij}:

$$\varepsilon_{ij} = (\langle u_i, v_j \rangle - x_{ij})$$

и формулу для обновления u_i на каждой итерации градиентного спуска:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \sum_j \varepsilon_{ij} v_j.$$

Аналогичную формулу можно получить для v_j:

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \sum_i \varepsilon_{ij} u_i$$

В этих формулах присутствует сумма большого количества слагаемых. Матрицы, на которых используется этот метод, могут быть очень велики, и считать такие суммы на каждом шаге градиента может быть ресурсозатратно. Именно поэтому для решения задачи неизбежно потребуется метод стохастического градиентного спуска, где на каждом шаге берутся случайные слагаемые, и используются только они:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \varepsilon_{ij} v_j,$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \varepsilon_{ij} u_i.$$

Плюсами стохастического градиентного спуска являются простота реализации и его сходимость. Однако, есть и минусы: алгоритм сходится медленно, и совершенно неясно, как выбирать шаг градиентного спуска (при константном шаге алгоритм сходится очень медленно).

4.2.2. Метод ALS

Метод ALS (alternating least squares) помогает избежать проблем, возникающих при использовании стохастического градиентного спуска. Идея этого метода заключается в следующем: в точке, где функционал будет принимать минимальное значение, его производные по u_i и по v_j должны быть равны нулю. Тогда можно действовать следующим образом: на одной итерации обновляется значение u_i из выражения

$$\frac{\partial Q}{\partial u_i} = 0.$$

На следующей итерации находится новое значение v_j:

$$\frac{\partial Q}{\partial v_i} = 0.$$

Это продолжается до тех пор, пока значения u_i и v_j не стабилизируются.

Можно явно выписать шаги в ALS:

$$\frac{\partial Q}{\partial u_i} = \sum_j 2(\langle u_i, v_j \rangle - x_{i,j}) v_j = 0$$

$$\sum_j v_j \langle v_j, u_i \rangle = \sum_j x_{ij} v_j$$

В результате получаются достаточно простые выражения и, сделав несложные преобразования, можно получить задачу в виде решения системы линейных уравнений:

$$\sum_j v_j v_j^T u_i = \sum_j x_{ij} v_j$$

$$\left(\sum_j v_j v_j^T \right) u_i = \sum_j x_{ij} v_j.$$

В итоге необходимо до сходимость повторять решение систем по случайным i, j:

$$\left(\sum_j v_j v_j^T \right) u_i = \sum_j x_{ij} v_j$$

$$\left(\sum_i u_i u_i^T \right) v_j = \sum_i x_{ij} u_i$$

	Пила	Улица Вязов	Ванильное небо	1 + 1
Маша	5	4	1	2
Юля	5	5	2	
Вова			3	5
Коля	3	?	4	5
Петя				4
Ваня		5	3	3

Таблица 4.1: Пример матрицы пользователи/оценки

4.2.3. Использование регуляризации

Важно отметить, что эта задача поставлена не совсем корректно, потому что имеет много решений. В таких случаях для того чтобы получить адекватное решение, обычно используют регуляризацию, то есть добавляют некоторые штрафы за большие значения параметров (u_i, v_j). Можно применить, например, L2-регуляризатор:

$$Q = \sum_{i,j}(\langle u_i, v_j\rangle - x_{ij})^2 + \alpha\sum_i \|u_i\|^2 + \beta\sum_j \|v_j\|^2 \to \min_{u_i, v_j},$$

где α и β — небольшие положительные числа (0.001, 0.01, 0.5).

Удивительно, но добавление регуляризаторов очень сильно сказывается на качестве в этой задаче.

4.3. Прогнозирование неизвестных значений в матрице

4.3.1. Постановка задачи в рекомендациях

Одно из применений матричных разложений — это прогнозирование неизвестных значений матрицы, такая задача часто возникает в рекомендательных системах. Итак, в таких системах, как правило, имеется матрица пользователи / оценки:

В этой матрице стоят оценки, которые пользователи поставили тем объектам, которые видели (например, тем фильмам, которые они смотрели). Многие значения в этой матрице неизвестны, потому, как правило, не все пользователи видели все объекты. Можно посмотреть на эту задачу следующим образом: в известных значениях матрицы эти значения приближаются как скалярное произведение некоторого профиля пользователя на профиль объекта:

$$x_{ij} \approx \langle u_i, v_j\rangle.$$

В данном случае оптимизируется функционал, в котором суммы берутся только по известным значениям матрицы:

$$\sum_{i,j:x_{ij}\neq 0}(\langle u_i, v_j\rangle - x_{ij})^2 \to \min.$$

В дальнейшем, взяв скалярное произведение профиля пользователя и профиля фильма, можно получить прогноз оценки, которую пользователь поставил бы фильму, если бы видел его, для тех элементов матрицы, которые нам неизвестны.

4.3.2. Добавление дополнительных параметров

В этой задаче можно добавить учет дополнительных факторов. Например, если шкала подразумевает, что существует некоторая средняя оценка μ, от которой отталкиваются пользователи при оценке фильмов, то правильнее будет приближать оценки следующим образом:

$$x_{ij} \approx \mu + \langle u_i, v_j\rangle,$$

а параметр μ также настраивать, исходя из оптимизационной задачи:

$$\sum_{i,j}(\mu + \langle u_i, v_j\rangle - x_{ij})^2 \to \min$$

Есть и другое соображение: пользователи бывают разными не только в силу своих интересов, но и в силу строгости оценок, которые они ставят фильмам. Одни пользователи ставят оценки выше, другие — ниже. Поэтому можно добавить еще одно слагаемое: b_i^u. Это базовый предиктор пользователя i, который будет отражать предпочтения этого пользователя в среднем по оценкам. Но при этом не стоит считать, что это честное среднее оценок, которые он поставит фильму, потому что присутствует слагаемое μ. Ещё можно добавить в сумму b_j^v, базовый предиктор по фильму j. Это слагаемое в какой-то мере приближает рейтинг фильма самого по себе, без мнения конкретного пользователя о нём.

$$x_{ij} \approx \mu + b_i^u + b_j^v + \langle u_i, v_j \rangle$$

$$\sum_{i,j} (\mu + b_i^u + b_j^v + \langle u_i, v_j \rangle - x_{ij})^2 \to \min$$

И теперь в задаче производится оптимизация взаимодействия интересов пользователя и особенности фильма, без влияния качества фильма, строгости пользователя и без учета средней оценки, которую пользователь ставит фильмам.

Так как были добавлены новые параметры, уместно обсудить вопрос регуляризации. Новых параметров в задаче значительно меньше, чем старых, потому что b_i^u и b_j^v — это просто числа, соответствующие каждому фильму и каждому пользователю. Для них тоже можно добавить l2-регуляризацию:

$$\sum_{i,j} (\mu + b_i^u + b_j^v + \langle u_i, v_j \rangle - x_{ij})^2 + \alpha \sum_i \|u_i\|^2 + \beta \sum_j \|v_j\| + \gamma \sum_i {b_i^u}^2 + \delta \sum_j {b_j^v}^2 \to \min .$$

4.4. Проблема отсутствия негативных примеров и implicit методы

4.4.1. Задача рекомендации товаров и её отличия от предыдущей

В отличие от случая рекомендаций фильмов, в случае рекомендации товаров имеется не матрица с оценками пользователей для объектов, а матрица каких-то событий, например, матрица покупок. Она будет устроена следующим образом: в каких-то ячейках матрицы будут нули, в каких-то — единички, реже — числа побольше.

	Вечернее платье	Поднос для писем	iPhone 6s	Шуба D&G
Маша	1		1	
Юля	1	1		1
Вова		1	1	
Коля	1	?	1	
Петя		1	1	
Ваня			1	1

Таблица 4.2: Пример матрицы покупок

Так или иначе, в матрице будут присутствовать только положительные примеры, то есть случаи, когда человек что-то купил: ему понравился товар, пользователь был готов потратить на это деньги, и в итоге потратил деньги. В то же время в матрице не будет примеров товаров, про которые точно известно, что человек никогда это не купит. Невозможно понять: человек не видел этот товар и поэтому не купил, или он его не купил, потому что он ему не нравится.

Пусть в матрице присутствуют только единицы и нули (такое может получиться, если никакой товар не покупался больше одного раза). Кажется, что можно применить все те же методы:

$$x_{ij} = 1 \approx \langle u_i, v_j \rangle$$

$$\sum_{i,j:x_{ij} \neq 0} (\langle u_i, v_j \rangle - x_{ij})^2 \to \min$$

Но если рассматривать задачу матричного разложения как прогнозирование неизвестных значений матрицы, то на такую матрицу, в которой все известные значения — это единицы, легко настроятся константные

профили для пользователей и объектов, которые, не будут нести никакой содержательной информации:

$$u_i = \frac{1}{\sqrt{d}}(1 \dots 1)$$

$$v_j = \frac{1}{\sqrt{d}}(1 \dots 1)$$

Это наводит на мысль, что в постановке задачи что-то не так. И действительно, как уже было сказано ранее, в данных есть только примеры, когда что-то человеку понравилось. Примеров, когда не понравилось, — нет. Поэтому модель в принципе нельзя научить понимать, что что-то кому-то нравится не будет. Конечно, такая модель неприменима на практике.

Таким образом, обратную связь от пользователя можно разделить на два типа:

- Explicit feedback: можно получить как положительные, так и отрицательные примеры (низкие и высокие оценки фильмов, лайки и дизлайки и т.д.)
- Implicit feedback: есть только положительные (лайки, покупки, просмотры) или отрицательные (дизлайки) примеры.

4.4.2. Implicit matrix factorization, implicit ALS

Это наводит на мысль, что нужно адаптировать методы матричных разложений для случая implicit feedback'a. Идея очень простая: можно заполнить неизвестные значения матрицы каким-то числом, например нулем, и настроить матричное разложение на всей матрице. Однако если поступить таким образом, то нулевые и ненулевые значения будут приближаться с одинаковой степенью неопределённости. Это не очень хорошо, потому что в ненулевых значениях есть уверенность, а в нулевых — нет. С этим нужно бороться. Можно решать оптимизационную задачу, но каждому слагаемому в этой задаче присвоить некоторый вес, который зависит от того, известно ли это значение в матрице. В итоге получается задача минимизации некоторого взвешенного расстояния между настоящими ответами и прогнозами:

$$\sum_{i,j} w_{ij}(\langle u_i, v_j \rangle - x_{ij})^2 \to \min$$

При этом для нулевых элементов берутся веса поменьше, чтобы не слишком настраиваться на нули, в которых нет уверенности. Получается, алгоритм склонен ошибаться в тех ячейках матрицы, где стоят нули, в большей степени, нежели в тех, где стоят ненулевые элементы. Это достаточно логично, если интерпретировать нулевые элементы как результат того, что человек на наблюдал данный объект, а не как то что этот объект ему не понравился. Для выбора весов часто используют следующую формулу:

$$w_{ij} = 1 + \alpha|x_{ij}|,$$

где α — это некоторый параметр, который нужно подбирать. Обычно подбирается порядок значения: $10, 100, 1000$, и т. д.

Этот метод можно использовать не только для случая implicit feedback'a, потому что в выражении стоит модуль, и можно считать, что значения, близкие к нулю, — это значения, в которых есть уверенность, а чем больше значение по модулю, тем меньше уверенности в нём.

Если ко всем описанному выше добавить оптимизацию с помощью метода ALS, то получится часто используемый на практике алгоритм implicit ALS.

4.5. Вероятностный взгляд на матричные разложения

4.5.1. Матричное разложение по норме Фробениуса и нормальное распределение

В случае построения матричного разложения по норме Фробениуса оптимизационная задача выглядит просто:

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \to \min_{u_i, v_j}$$

Она напоминает обобщение метода наименьших квадратов на случай матриц.

Полезно вспомнить, как выглядит нормальное распределение (рисунок 4.6):

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Нормальное распределение очень часто используется для моделирования погрешностей, шумов и других

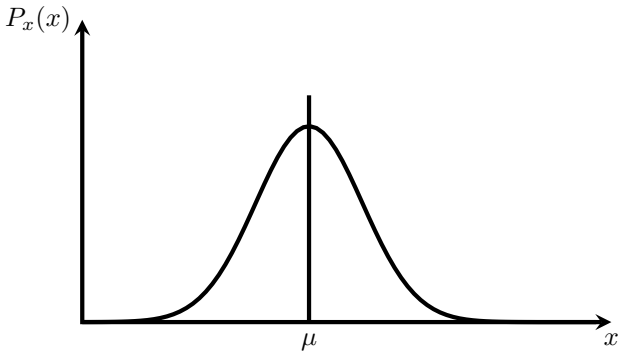

Рис. 4.6: Нормальное распределение

подобных отклонений. Поэтому логично представить, что значения в исходной матрице — это истинные значения (математическое ожидание) плюс нормальный шум:

$$x_{ij} = \langle u_i, v_j \rangle + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$$x_{ij} \sim \mathcal{N}(\langle u_i, v_j \rangle, \sigma^2)$$

Для такой постановки задачи можно записать принцип максимума правдоподобия:

$$\prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(x_{ij} - \langle u_i, v_j \rangle\right)^2}{2\sigma^2}} \to \max,$$

и из него получить всё тот же функционал, который напоминает метод наименьших квадратов, который также связан с нормальным распределением:

$$\sum_{i,j} \frac{(x_{ij} - \langle u_i, v_j \rangle)^2}{2\sigma^2} - \frac{1}{2} \ln 2\pi\sigma^2 \to \min$$

$$\sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \to \min$$

4.5.2. Распределение Пуассона и неотрицательное матричное разложение

С другой стороны, нормальное распределение оказывается не очень удачным, если значения матрицы — это целые числа, или если в матрице часто встречаются нули. Такая ситуация часто встречается в задачах анализа текстов, когда в качестве признаков выступают частоты слов. Возможно, в данной ситуации больше подошло бы другое распределение. Если речь идёт о признаках-счётчиках, то для решения задачи лучше всего подходит распределение Пуассона (рисунок 4.7):

$$X \sim Poiss(\lambda)$$

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \qquad \mathbb{E}X = \lambda$$

	database	SQL	index	regression	likelihood	linear
d1	24	21	9	0	0	3
d2	32	10	5	0	3	0
d3	12	16	5	0	0	0
d4	6	7	2	0	0	0
d5	43	32	20	0	3	0
d6	2	0	0	18	7	16
d7	0	0	1	32	12	0
d8	3	0	0	22	4	2
d9	1	0	0	34	27	25
d10	6	0	0	17	4	23

Таблица 4.3: Таблица частот слов в текстах

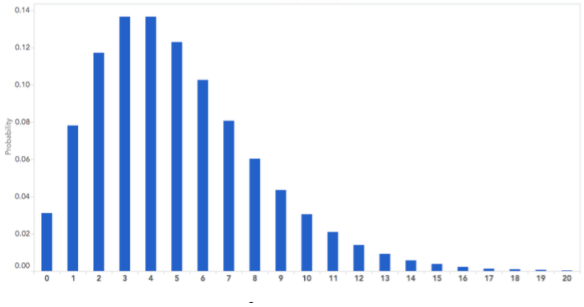

Рис. 4.7: Распределение Пуассона

Распределение Пуассона — это дискретное распределение; значения, которые может принимать случайная величина — целочисленные, поэтому оно действительно подходит для решения данной задачи. В этом случае также можно записать принцип максимизации правдоподобия:

$$P\left(x_{ij}\right) = \frac{\langle u_i, v_j\rangle^{x_{ij}}}{x_{ij}!} e^{-\langle u_i, v_j\rangle}$$

$$\prod_{i,j} \frac{\langle u_i, v_j\rangle^{x_{ij}}}{x_{ij}!} e^{-\langle u_i, v_j\rangle} \to \max$$

Тогда после некоторых преобразований получится выражение, которое нужно оптимизировать, чтобы подобрать профили u_i и v_j, исходя из предположения, что значения в исходной матрице были распределены по Пуассону:

$$\sum_{i,j} \langle u_i, v_j\rangle - x_{ij} \ln\langle u_i, v_j\rangle \ln x_{ij}! \to \min$$

$$\sum_{i,j} \langle u_i, v_j\rangle - x_{ij} \ln\langle u_i, v_j\rangle \to \min$$

В этом случае получается разложение исходной матрицы на некоторые матрицы с неотрицательными элементами. Такие разложения называются неотрицательными матричными разложениями, и их гораздо легче интерпретировать, чем разложения с отрицательными элементами. Каждая компонента может означать степень представленности термина, а некоторыми нормировками все эти числа можно привести к вероятностям.

В случае неотрицательного матричного разложения, использующего распределение Пуассона, можно точно так же выписать метод стохастического градиентного спуска и получить простые правила обновления профилей u_i и v_j:

$$Q = \sum_{i,j} \langle u_i, v_j\rangle - x_{ij} \ln\langle u_i, v_j\rangle \to \min$$

$$\frac{\partial Q}{\partial u_i} = \sum_j v_j - \frac{x_{ij}}{\langle u_i, v_j\rangle} v_j = \sum_j \frac{\langle u_i, v_j\rangle - x_{ij}}{\langle u_i, v_j\rangle} v_j \to \min$$

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \tilde{\varepsilon}_{ij} v_j$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \tilde{\varepsilon}_{ij} u_i$$

Но в то же время использовать распределение Пуассона — это не единственный способ получить неотрицательные матрицы. Можно по-прежнему использовать норму Фробениуса, но при этом добавлять дополнительные ограничения на значения элементов матрицы:

$$Q = \sum_{i,j} \left(\langle u_i, v_j \rangle - x_{ij} \right)^2 \to \min_{\substack{u_i, v_j: \\ u_{ik} \geq 0 \\ v_{jk} \geq 0}}$$

Подробнее об этом будет рассказано далее.

4.6. Неотрицательные матричные разложения: постановка и решение

4.6.1. Постановка задачи

Прежде чем начать изучать неотрицательные матричные разложения, полезно снова вспомнить постановку задачи факторизации матриц — разложения матрицы на произведение двух матриц меньшего ранга, — но уже в других обозначениях (рисунок 4.8):

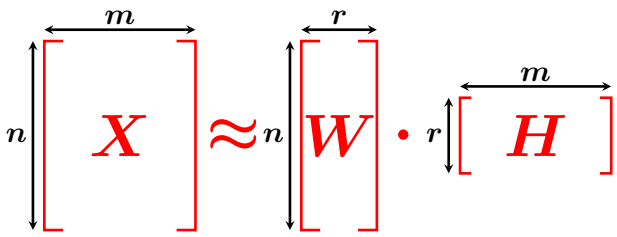

Рис. 4.8: Матричное разложение

$$r < \min(n, m)$$

Задача поиска таких матриц может быть представлена, как оптимизационная:

$$(W^*, H^*) = \underset{W \geq 0, H \geq 0}{\operatorname{argmin}} D(X, WH).$$

Функционал $D(X, WH)$ (функция потерь) показывает насколько точно найденное произведение приближает исходную матрицу X.

Особенность задачи неотрицательных матричных разложений (NMF), заключается в том, что решение W и H ищут в множестве матриц с неотрицательными элементами. Такая постановка часто имеет смысл, если матрицы W и H по итогам разложения нужно интерпретировать. Например, эти матрицы могут представлять собой вероятности появления слов в каких-то текстах, документах, или концентрации веществ в растворах и т.д.

Пусть

$$\hat{X} = WH$$

Интерес представляют функции потерь D, которые распадаются на суммы функции потерь между компонентами матриц X и $\hat{X}$:

$$D\left(X, \hat{X}\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} d(x_{ij}, \hat{x}_{ij}).$$

Такие функции потерь называются сепарабельными.

Чаще всего в задаче неотрицательного матричного разложения в качестве функции потерь используют норму Фробениуса:

$$d_F(x, \hat{x}) = (x - \hat{x})^2,$$

$$D_F\left(X, \hat{X}\right) = \sum_{i=1}^{m}\sum_{j=1}^{n}(x_{ij} - \hat{x}_{ij})^2 \equiv \|X - WH\|_F^2$$

4.6.2. Проблемы задачи поиска неотрицательного матричного разложения

В задаче получения неотрицательных матричных разложений есть несколько проблем. Во-первых, она некорректно поставлена. Если имеется какое-то решение этой задачи W_0, H_0, то существует большое количество матриц Y, таких, что матрицы

$$W = W_0 Y, \qquad H = Y^{-1} H_0$$

также будут решением этой задачи, если существует Y^{-1} и преобразование с помощью Y сохраняет неотрицательность элементов матриц W_0, H_0. Таким образом, решение никогда не определяется однозначно.

Кроме того, функция потерь D и норма Фробениуса, которая используется чаще всего, не выпукла по совокупности аргументов W и H, поэтому нельзя использовать методы, которые находят глобальный минимум функции потерь, и приходится искать другие методы.

Чаще всего используется блочно-покоординатная минимизация (f и g - некоторые функции):

Вход: $W^0 \geq 0, H^0 \geq 0$
Цикл
$H \leftarrow f(X, W, H)$;
$W \leftarrow g(X, W, H)$.

Цикл повторяется до тех пор, пока значения H и W не сойдутся. Поскольку эта задача симметричная, функция f должна быть очень похожа на функцию g, в частности:

$$f(X, W, H) = g^T\left(X^T, H^T, W^T\right)$$

4.6.3. Применение градиентных методов

Итак, постановка задачи с использованием нормы Фробениуса выглядит следующим образом:

$$(W^*, H^*) = \underset{W\geq 0,\, H\geq 0}{\operatorname{argmin}} \|X - WH\|_F^2.$$

Если бы не было ограничения на неотрицательность, решение можно было бы легко получить, используя сингулярное разложение. Но поскольку присутствует ограничение, можно воспользоваться градиентными методами. Базовым методом будет для решения задачи поочередный градиентный спуск:

$$h_{kj} \leftarrow h_{kj} - \nu_{kj}\frac{\partial D_F}{\partial h_{kj}}, w_{ik} \leftarrow w_{ik} - \eta_{ik}\frac{\partial D_F}{\partial w_{ik}}. k = 1, \ldots, r, ; \quad i = 1, \ldots, m, ; \quad j = 1, \ldots, n$$

Проблема в том, что градиентный спуск не учитывает ограничение неотрицательности. Существует несколько способов это исправить. Во-первых, можно выбрать шаг градиентного спуска так, что обновления матриц станут мультипликативными, и тогда знак будет автоматически сохраняться:

$$\frac{\partial D_F}{\partial h_{kj}} = \sum_{i=1}^{m} w_{ik}\hat{x}_{ij} - \sum_{i=1}^{m} w_{ik}x_{ij}, \quad \nu_{kj} = \frac{h_{kj}}{\sum\limits_{i=1}^{m} w_{ik}\hat{x}_{ij}}, h_{kj} \leftarrow h_{kj} - \frac{h_{kj}}{\sum\limits_{i=1}^{m} w_{ik}\hat{x}_{ij}}\left(\sum_{i=1}^{m} w_{ik}\hat{x}_{ij} - \sum_{i=1}^{m} w_{ik}x_{ij}\right) = = h_{kj}\frac{\sum\limits_{i=1}^{m} w_{ik}x_{ij}}{\sum\limits_{i=1}^{m} w_{ik}\hat{x}_{ij}}.$$

В итоге старый элемент матрицы каждый раз умножается на положительное число. Таким образом, если матрица H инициализирована удачно, без отрицательных компонент, то в ходе итерационного процесса они не могут появиться. В матричном виде такие мультипликативные обновления можно записать следующим образом:

$$H \leftarrow H \otimes \left(W^T X\right) \oslash \left(W^T \hat{X}\right),$$

где $\otimes$ — поэлементное умножение матриц, $\oslash$ — поэлементное деление. Можно показать, что в алгоритме с мультипликативными обновлениями функция потерь D на каждом шаге монотонно не возрастает.

Методы, использующие мультипликативные обновления, очень популярны. Во-первых, они просты в реализации. Во-вторых, они хорошо масштабируются и легко приспосабливаются к работе с разреженными матрицами. Кроме того, исторически эти методы были предложены в самой первой статье по неотрицательным матричным разложениям. Однако скорость сходимости таких методов не очень большая. Впрочем, ее можно увеличить, если обновлять матрицы W и H не по одному разу в цикле, а по несколько раз подряд.

4.6.4. Метод попеременных наименьших квадратов

Задача неотрицательного матричного разложения с нормой Фробениуса может решаться еще несколькими эффективными способами. Если фиксировать одну из двух матриц внутри нормы Фробениуса, получается задача минимизации наименьших квадратов, которую можно решать аналитически. Для этого можно использовать метода попеременных наименьших квадратов (ALS):

$$H \leftarrow \max\left(\underset{H}{\operatorname{argmin}} \|X - WH\|_F^2, 0\right) = \\ = \max\left(\left(W^T W\right)^{-1} W^T X, 0\right).$$

На каждом шаге находится точное решение задачи наименьших квадратов по одной из компонент, но, к сожалению, это решение может давать и отрицательные значения в матрицах. Их можно заменять нулями, то есть проецировать матрицу на неотрицательную область. Этот метод быстрый и грубый: итерационный процесс может не сходиться при таких обновлениях. Функция потерь не монотонна, то есть она может увеличиваться, что является результатом проецирования. Метод ALS можно использовать для инициализации более сложных методов.

4.6.5. Метод попеременных неотрицательных наименьших квадратов

На другом полюсе методов неотрицательного матричного разложения с нормой Фробениуса находится метод попеременных неотрицательных наименьших квадратов (ANLS). Он действует сначала точно так же, как метод попеременных наименьших квадратов, но минимум функции потерь ищется только в неотрицательной области:

$$H \leftarrow \underset{H \geq 0}{\operatorname{argmin}} \|X - WH\|_F^2.$$

Решение такой задачи найти сложнее, его нельзя записать аналитически, поэтому этот метод медленнее, но он точнее. Каждая итерация метода требует существенных вычислительных затрат, поэтому им можно пользоваться для уточнения решения, которое найдено более простыми методами.

4.6.6. Метод иерархических попеременных наименьших квадратов

Наконец, один из самых эффективных и популярных методов неотрицательного матричного разложения с нормой Фробениуса — это метод иерархических попеременных наименьших квадратов (HALS). Идея заключается в том, чтобы найти аналитически минимум нормы Фробениуса не только по матрице W, но и по отдельному ее столбцу, или по строке матрицы H, а затем этот аналитический минимум проецировать на неотрицательную область:

$$h_k \leftarrow \underset{h_k \geq 0}{\operatorname{argmin}} \|X - WH\|_F^2 = \max\left(0, \frac{w_k^T X - \sum_{l \neq k} w_k^T w_l h_l}{w_k^T w_k}\right).$$

Проекция в этом случае оказывает менее разрушительное действие, чем в методе попеременных наименьших квадратов, поскольку одновременно обновляются только небольшие куски матриц. Такой метод вычислительно эффективен и сходится быстрее, чем метод мультипликативных обновлений, однако он более чувствителен к начальному приближению.

4.7. Неотрицательные матричные разложения: функционалы и инициализация

4.7.1. Функции потерь

Существует огромное количество функций потерь, единственное накладываемое ограничение в данной задаче — это сепарабельность, то есть функция потерь должна распадаться на сумму поэлементных расстояний между матрицами X и $\hat{X}$

$$(W^*, H^*) = \underset{W \geq 0, H \geq 0}{\operatorname{argmin}} D(X, WH).$$

Кроме того, это расстояние должно быть всегда неотрицательным:

$$D\left(X, \hat{X}\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} d(x_{ij}, \hat{x}_{ij}), d(x, \hat{x}) \geq 0, d(x, \hat{x}) = 0 \Leftrightarrow x = \hat{x}$$

Этим условиям удовлетворяют функции потерь, указанные в таблице ниже.

Название	$d(x, \hat{x})$
норма l_1	$d_1(x, \hat{x}) = \lvert x - \hat{x} \rvert$
норма Фробениуса	$d_F(x, \hat{x}) = (x - \hat{x})^2$
обобщённая дивергенция Кульбака-Лейблера	$d_{KL}(x, \hat{x}) = x \ln \frac{x}{\hat{x}} - x + \hat{x}$
дивергенция Итакура-Саито	$d_{IS}(x, \hat{x}) = \ln \frac{\hat{x}}{x} + \frac{x}{\hat{x}} - 1$
расстояние Хеллингера	$d_H(x, \hat{x}) = \left(\sqrt{x} - \sqrt{\hat{x}}\right)^2$

Таблица 4.4: Функции потерь, подходящие для решения задачи неотрицательного матричного разложения

4.7.2. Связь различных функций потерь с правдоподобием

В различных прикладных областях используются разные функции потерь. Во многих биологических приложениях используется норма Фробениуса, в анализе аудиозаписи — дивергенция Итакура-Саито, в задачах моделирования текстов чаще всего используется обобщённая дивергенция Кульбака-Лейблера. Почему какие то функции являются оптимальными для своих классов задач? Дело в том, что функции потерь часто представляют собой замаскированные правдоподобия. То есть существуют такие плотности $p\left(X|\hat{X}\right)$, что

$$D\left(X, \hat{X}\right) \propto -\ln p\left(X|\hat{X}\right).$$

Таким образом, при минимизация функции потерь происходит максимизация логарифма правдоподобия.

Функция потерь	Порождающая модель
Фробениуса	аддитивная гауссовская
Кульбака-Лейблера	пуассоновская
Итакура-Саито	мультипликативная гамма

Таблица 4.5: Порождающие модели для функций потерь

4.7.3. Использование дивергенции Кульбака-Лейблера в качестве функционала потерь

Дивергенция Кульбака-Лейблера находится на втором месте по популярности среди всех функционалов потерь для решения задач неотрицательного матричного разложения. Для неё можно точно так же как и для

нормы Фробениуса записать блочно-покоординатный спуск с мультипликативными обновлениями:

$$h_{kj} \leftarrow h_{kj} \frac{\sum\limits_{i=1}^{m} w_{ik} \frac{x_{ij}}{\hat{x}_{ij}}}{\sum\limits_{i=1}^{m} w_{ik}}.$$

И так же можно показать, что функция потерь будет монотонно не возрастать при таких обновлениях. Ещё один алгоритм, который решает самую задачу минимизации дивергенции Кульбака-Лейблера, но немного иначе, — это метод PLSA, о нём будет рассказано позже.

4.7.4. Инициализация искомых матриц

Инициализация в задачах неотрицательного матричного разложения играет очень большую роль, поскольку у минимизируемого функционала очень много локальных минимумов, и сходимость метода тоже локальная. В зависимости от того, насколько точно начальное приближение, получаются разные по качеству решения. Какими же могут быть начальные приближения?

- Случайная инициализация: матрицы W и H можно заполнить случайными неотрицательными числами.
- Кластеризация: можно разделить столбцы матрицы X на r кластеров (например, методом K-means), затем инициализировать столбцы матрицы W центроидами этих кластеров, а строки матрицы H инициализировать в соответствии с матрицей смежности.
- SVD: получить сингулярное разложение матрицы X, отобрать r собственных троек, относящихся к наибольшим собственным числам, и затем из соответствующих собственных векторов сконструировать неотрицательные матрицы W и H.
- ALS: на любом из предыдущих начальных приближений можно произвести несколько итераций метода попеременных наименьших квадратов, это улучшает качество приближения.
- Мультистарт: генерируются $10 - 20$ случайных матриц W и H. Каждое начальное приближение улучшается методом ALS, затем провести $10 - 20$ итераций основного метода (например, метода с мультипликативными обновлениями). Из полученных $10 - 20$ пар выбирается пара с наименьшим значением целевого функционала, это и будет начальное приближение.

4.8. Обработка пропусков

4.8.1. Постановка задачи

Пусть имеется матрица $X \in \mathbb{R}^{\ell \times d}$, строки этой матрицы — это пользователи сайта «КиноПоиск», столбцы — это фильмы, а элементы матрицы — это оценки, которые пользователи выставляют фильмам. Большая часть людей смотрела малую долю фильмов, которые есть в базе «КиноПоиска». Поэтому в матрице X будет огромное количество неизвестных значений. Можно ли пропуски в такой матрице как-то заполнить?

Ещё один пример. Пусть матрица X — это матрица объекты-признаки, дальше на этих данных нужно выполнить задачу классификации или регрессии. Как это правильно делать?

Ключевое предположение всех методов работы с пропусками — случайность расположения пропусков в матрице. Это предположение очень сильное и в каждой задаче нужно проверять, действительно ли оно выполняется. Вот пример задачи, в которой оно заведомо не выполняется: если в опросе есть вопрос о годовом доходе респондента, многие могут отказаться на него отвечать. Более того, может оказаться так, что респонденты с большим годовым доходом чаще будут отказываться отвечать на этот вопрос и тогда в итоговой матрице, составленной по результатам опроса, будут неслучайные пропуски. Еще один классический пример взят из практики венгерского статистика Абрахама Вальда, который работал в США во время второй мировой войны и анализировал повреждения самолетов, которые возвращались с бомбардировок. На основании этого анализа принималось решение о том, какие части самолетов нужно укрепить. Нельзя укрепить весь самолет целиком, потому что от этого он будет плохо летать, поэтому необходимо выбрать, какие части самолетов важнее всего. Для того чтобы это понять, собиралась статистика о количестве пулевых отверстий в разных частях самолетов. Это задача, в которой пропуски не просто неслучайны, а несут в себе самую важную информацию о признаке, потому что именно те самолеты, которые не возвращаются с боевых действий,

получили наиболее критические повреждения. Поэтому Вальд принял решение о том, что укреплять нужно те части самолетов, в которых было меньше всего повреждений на вернувшихся самолётах. Эти примеры показывают, что нужно всегда следить за выполнением предположения о случайности пропусков.

4.8.2. Метода обработки пропусков

Что же можно сделать с матрицей X в случае, когда данное предположение выполняется?. Самый простой способ — выбрасывать те объекты, на которых значение хотя бы одного из признаков пропущено. Перед этим стоит избавиться от признаков, у которых очень много пропусков, потому что иначе можно остаться совсем без выборки. Этот способ очень прост, но он не очень хорош, потому что исключаются объекты, для которых какие-то признаки известны, информация теряется.

Другая крайность работы с пропусками — это методы, которые пытаются заполнить пропуски. Это можно делать большим количеством разных способов. Например, для каждой строки x_{i1}, в которой есть хотя бы один пропуск, находят самую похожую (по функции потерь) на нее строку x_{i2}, и заменяют пропущенное значение в первой строке на аналогичное во второй.

Другой способ — это заполнение пропусков средними или медианами по столбцу. Таким образом, не используется никакой информация об объекте, а только среднее значение признака во всей выборке.

Эти две крайности сочетает друг с другом EM-алгоритм. EM-алгоритм предполагает, что полные данные каким-то образом распределены (например, имеют совместное многомерное нормальное распределение). По имеющимся данным оценивается среднее и ковариационная матрица признаков у этого распределения. Затем пропуски заполняются наиболее вероятными значениями в соответствии с полученной оценкой распределения. Этот процесс повторяется несколько раз: после заполнения пропусков, параметры распределения переоцениваются, после получения новых параметров пропуски заполняются. И так, пока процесс не сойдется.

Еще один способ заполнения пропусков — это матричное разложение. Можно представить имеющуюся большую разреженную матрицу в виде произведения двух плотных матриц меньшего ранга, а затем пропуски в исходной матрице представить, как значение матрицы $\hat{X} = WH$.

Все эти методы вызывают подозрение, поскольку непонятно, откуда в них берётся информация. В некоторых задачах (например, линейная регрессия или метод главных компонент) можно обойтись вовсе без заполнения пропусков. В таких задачах часто матрица объекты-признаки X используется только для подсчета величин вида $\frac{1}{\ell}X^TX$ и $\frac{1}{\ell}X^Ty$. Значение таких матриц можно вычислять только по полным парам, если оба необходимых элемента не пропущены:

$$\frac{1}{\ell}\left(X^TX\right)_{jk} = \frac{1}{\ell}\sum_{i=1}^{\ell} x_{ij}x_{ik} \approx \approx \frac{1}{\ell_{jk}}\sum_{i=1}^{\ell} x_{ij}x_{ik}\left[x_{ij} \neq NA, x_{ik} \neq NA\right],$$

где ℓ_{jk} — число полных пар. При использовании таких методов информация никуда не исчезает, и не появляется ниоткуда. Кажется, что когда этот метод применим, он оптимален.

4.8.3. Важные детали

Стоит обсудить еще несколько деталей в задаче заполнения пропусков. Если пропущены значения некоторой категориальной переменной, то удобно закодировать их в виде новой категории. Этот метод работает, в том числе и для ситуации, когда пропуски неслучайны. При работе с деревьями или лесами, если пропущены значения в каких-то непрерывных признаках, их можно заполнить значением, которое сильно отличается от всех типичных значений признака (например, очень большим отрицательным значением). Если так поступить, эти значения будут попадать в отдельный лист, и будут обрабатываться отдельно.

Заполнение пропусков — это творческая задача, поэтому к ней нужно подходить вдумчиво в каждой конкретной задаче анализа данных. Если используется какой-либо метод обучения с учителем, и в матрице объекты-признаки есть пропуски, нужно следить за тем, какой метод обработки пропусков используется по умолчанию в используемом методе. Например, в большей части методов, реализующих линейную регрессию, по умолчанию используется отбрасывание объектов, на которых есть пропуски, — не очень хороший метод. В методе xgboost по умолчанию используется достаточно сложный метод обработки пропусков и, этот алгоритм можно использовать менее вдумчиво.