Лекция 13
Матричные разложения и рекомендательные системы

Е. А. Соколов
ФКН ВШЭ

18 декабря 2017 г.

1 Понижение размерности и метод главных компонент

В машинном обучении часто возникает задача уменьшения размерности признакового пространства. Для этого можно, например, удалять признаки, которые слабо коррелируют с целевой переменной; выбрасывать признаки по одному и проверять качество модели на тестовой выборке; перебирать случайные подмножества признаков в поисках лучших наборов. Ещё одним из подходов к решению задачи является поиск новых признаков, каждый из которых является линейной комбинацией исходных признаков. В случае использования квадратичной функции ошибки при поиске такого приближения получается *метод главных компонент* (principal component analysis, PCA), о котором и пойдет речь.

Пусть $X \in \mathbb{R}^{\ell \times D}$ — матрица «объекты-признаки», где ℓ — число объектов, а D — число признаков. Поставим задачу уменьшить размерность пространства до d. Будем считать, что данные являются центрированными — то есть среднее в каждом столбце матрицы X равно нулю.

Будем искать главные компоненты $u_1, \ldots, u_D \in \mathbb{R}^D$, которые удовлетворяют следующим требованиям:

1. Они ортогональны: $\langle u_i, u_j \rangle = 0$, $i \neq j$;
2. Они нормированы: $\|u_i\|^2 = 1$;
3. При проецировании выборки на компоненты $u_1, \ldots, u_d$ получается максимальная дисперсия среди всех возможных способов выбрать d компонент.

Чтобы понизить размерность выборки до d, мы будем проецировать её на первые d компонент — из последнего свойства следует, что это оптимальный способ снижения размерности.

Дисперсия проецированной выборки показывает, как много информации нам удалось сохранить после понижения размерности — и поэтому мы требуем максимальной дисперсии от проекций.

Проекция объекта x на компоненту u_i вычисляется как $\langle x, u_i \rangle$, а проекция всей выборки на эту компоненту — как Xu_i. Если за U_d обозначить матрицу, столбцы которой равны первым d компонентам, проекция выборки на них будет записываться как XU_d, а дисперсия проецированной выборки будет вычисляться как след ковариационной матрицы:

$$\operatorname{tr} U_d^T X^T X U_d = \sum_{i=1}^{d} \|Xu_i\|^2.$$

Начнём с первой компоненты. Сведём все требования к ней в оптимизационную задачу:

$$\begin{cases} \|Xu_1\|^2 \to \max_{u_1} \\ \|u_1\|^2 = 1 \end{cases}$$

Запишем лагранжиан:

$$L(u_1, \lambda) = \|Xu_1\|^2 + \lambda(\|u_1\|^2 - 1).$$

Продифференцируем его и приравняем нулю:

$$\frac{\partial L}{\partial u_1} = 2X^T X u_1 + 2\lambda u_1 = 0.$$

Отсюда получаем, что u_1 должен быть собственным вектором ковариационной матрицы $X^T X$. Учтём это и преобразуем функционал:

$$\|Xu_1\|^2 = u_1^T X^T X u_1 = \lambda u_1^T u_1 = \lambda \to \max_{u_1}$$

Значит, собственный вектор u_1 должен соответствовать максимальному собственному значению.

Для следующих компонент к оптимизационной задаче будут добавляться требования ортогональности предыдущим компонентам. Решая эти задачи, мы получим, что главная компонента u_i равна собственному вектору, соответствующему i-му собственному значению.

После того, как найдены главные компоненты, можно проецировать на них и новые данные. Если нам нужно работать с тестовой выборкой X', то её проекции вычисляются как $Z' = X'U_d$. Отметим также, что в методе главных компонент новые признаки вычисляются как линейные комбинации старых:

$$z'_{ij} = \sum_{k=1}^{D} x'_{ik} u_{kj}.$$

Альтернативные постановки. Существует несколько других постановок задачи понижения размерности, приводящих к методу главных компонент.

Первый способ основан на матричном разложении. Будем искать матрицу с новыми признаковыми описаниями $Z \in \mathbb{R}^{\ell \times d}$ и матрицу проецирования $U \in \mathbb{R}^{D \times d}$, произведение которых даёт лучшее приближение исходной матрицы X:

$$\|X - ZU^T\|^2 \to \min_{Z,U}$$

Решением данной задачи также являются собственные векторы ковариационной матрицы.

Второй способ состоит в поиске такого линейного подпространства, что расстояние от исходных объектов до их проекций на это подпространство будет минимальным. В этом случае задача оказывается эквивалентной задаче максимизации дисперсии проекций.

2 Рекомендательные системы

Рекомендательные системы используются в интернет-магазинах, музыкальных сервисах, социальных сетях; с их помощью каждому пользователю можно подобрать наиболее интересный товар или, например, фильм. В этой лекции мы поговорим об основных подходах к построению рекомендательных систем (на основе коллаборативной фильтрации и на основе контента), обсудим методы оценивания их качества и некоторые проблемы.

Мы будем рассуждать в терминах пользователей (users, U) и товаров (items, I), но все методы подходят для рекомендаций любых объектов. Будем считать, что для некоторых пар пользователей $u \in U$ и товаров $i \in I$ известны оценки r_{ui}, которые отражают степень заинтересованности пользователя в товаре. Вычисление таких оценок — отдельная тема. Например, в интернет-магазине заинтересованность может складываться из покупок товара и просмотров его страницы, причём покупки должны учитываться с большим весом. В социальной сети заинтресованность в материале может складываться из времени просмотра, кликов и явного отклика (лайки, репосты); это всё тоже должно суммироваться с различными весами. Не будем сейчас останавливаться на этом вопросе, а перейдём к основной задаче.

Требуется по известным рейтингам r_{ui} научиться строить для каждого пользователя u набор из k товаров $I(u)$, наиболее подходящих данному пользователю — то есть таких, для которых рейтинг r_{ui} окажется максимальным.

Самый распространённый подход в данном случае — сформировать признаки, характеризующие пользователя, товар и их взаимодействия, и обучить модель, которая по данным признакам будет предсказывать рейтинг. Это может быть ранжирующая модель, которая сортирует все товары для данного пользователя; может быть и обычная поточечная модель. Ниже мы рассмотрим некоторые простые методы рекомендаций, оценки которых, как правило, используются в качестве признаков для итоговой модели.

§2.1 Коллаборативная фильтрация

Методы коллаборативной фильтрации строят рекомендации для пользователя на основе похожестей между пользователями и товарами. Мы рассмотрим два подхода к определению сходства.

2.1.1 Memory-based

Два пользователя похожи, если они ставят товарам одинаковые оценки. Рассмотрим двух пользователей u и v Обозначим через I_{uv} множество товаров i, для

которых известны оценки обоих пользователей:

$$I_{uv} = \{i \in I \mid \exists r_{ui} \ \& \ \exists r_{vi}\}.$$

Тогда сходство двух данных пользователей можно вычислить через корреляцию Пирсона:

$$w_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}},$$

где $\bar{r}_u$ и $\bar{r}_v$ — средние рейтинги пользователей по множеству товаров I_{uv}.

Чтобы вычислять сходства между товарами i и j, введём множество пользователей U_{ij}, для которых известны рейтинги этих товаров:

$$U_{ij} = \{u \in U \mid \exists r_{ui} \ \& \ \exists r_{uj}\}.$$

Тогда сходство двух данных товаров можно вычислить через корреляцию Пирсона:

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}},$$

где $\bar{r}_i$ и $\bar{r}_j$ — средние рейтинги товаров по множеству пользователей U_{ij}. Отметим, что существуют и другие способы вычисления похожестей — например, можно вычислять скалярные произведения между векторами рейтингов двух товаров.

Мы научились вычислять сходства товаров и пользователей — разберём теперь несколько способов определения товаров, которые стоит рекомендовать пользователю u_0. В подходе на основе сходств пользователей (user-based collaborative filtering) определяется множество $U(u_0)$ пользователей, похожих на данного:

$$U(u_0) = \{v \in U \mid w_{u_0 v} > \alpha\}.$$

После этого для каждого товара вычисляется, как часто он покупался пользователями из $U(u_0)$:

$$p_i = \frac{|\{u \in U(u_0) \mid \exists r_{ui}\}|}{|U(u_0)|}.$$

Пользователю рекомендуются k товаров с наибольшими значениями p_i. Данный подход позволяет строить рекомендации, если для данного пользователя найдутся похожие. Если же пользователь является нетипичным, то подобрать что-либо не получится.

Также существует подход на основе сходств товаров (item-based collaborative filtering). В нём определяется множество товаров, похожих на те, которые интересовали данного пользователя:

$$I(u_0) = \{i \in I \mid \exists r_{u_0 i_0}, w_{i_0 i} > \alpha\}.$$

Затем для каждого товара из этого множества вычисляется его сходство с пользователем:

$$p_i = \max_{i_0 : \exists r_{u_0 i_0}} w_{i_0 i}.$$

Пользователю рекомендуются k товаров с наибольшими значениями p_i. Даже если пользователь нетипичный, то данный подход может найти товары, похожие на интересные ему — и для этого необязательно иметь пользователя со схожими интересами.

2.1.2 Модели со скрытыми переменными

Все описанные выше подходы требуют хранения разреженной матрицы $R = \{r_{ui}\}$, которая может быть достаточно большой. Более того, они весьма эвристичны и зависят от выбора способа вычисления сходства, способа генерации товаров-кандидатов, способа их ранжирования. Альтернативой являются подходы на основе моделей со скрытыми переменными (latent factor models).

Мы будем пытаться построить для каждого пользователя u и товара i векторы $p_u \in \mathbb{R}^d$ и $q_i \in \mathbb{R}^d$, которые будут характеризовать «категории интересов». Например, каждую компоненту такого вектора можно интерпретировать как степень принадлежности данного товара к определённой категории или степень заинтересованности данного пользователя в этой категории. Разумеется, никак не будет гарантироваться, что эти компоненты соответствуют каким-то осмысленным категориям, если только мы специально не потребуем этого от модели. По сути, векторы пользователей и товаров являются представлениями (embeddings), позволяющими свести эти сущности в одно векторное пространство.

Сходство пользователя и товара будем вычислять через скалярное произведение их представлений:

$$r_{ui} \approx \langle p_u, q_i \rangle.$$

Также через скалярное произведение можно вычислять сходство двух товаров или двух пользователей.

Мы можем записать функционал ошибки, исходя из способа вычисления сходства:

$$\sum_{(u,i)\in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \to \min_{P,Q} \tag{2.1}$$

Суммирование здесь ведётся по всем парам пользователей и товаров, для которых известен рейтинг r_{ui}. Заметим, что если R' — матрица R с центрированными строками и столбцами, то данная задача сводится к низкоранговому матричному разложению:

$$\|R' - P^T Q\|^2 \to \min_{P,Q}$$

Здесь представления пользователей и товаров записаны в столбцах матриц P и Q. Существуют модификации, в которых к скалярным произведениям добавляется масштабирующий множитель $\alpha \in \mathbb{R}$:

$$\|R' - \alpha P^T Q\|^2 \to \min_{P,Q,\alpha}$$

Данный функционал можно регуляризовать:

$$\sum_{(u,i)\in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_{u\in U} \|p_u\|^2 + \mu \sum_{i\in I} \|q_i\|^2 \to \min_{P,Q} \tag{2.2}$$

Описанная модель носит название Latent Factor Model (LFM).

Отметим, что использование среднеквадратичной ошибки не всегда имеет смысл — в рекомендациях требуется выдать более высокие предсказания для товаров, которые более интересны пользователю, но вовсе не требуется точно предсказывать рейтинги. Впрочем, среднеквадратичную ошибку удобно оптимизировать; более

того, именно она использовалась в качестве функционала в конкурсе Netflix Prize, который во многом определил развитие рекомендательных систем и в котором было предложено много популярных сейчас методов.

Существует два основных подхода к решению задачи (2.1). Первый — стохастический градиентный спуск, который на каждом шаге случайно выбирает пару $(u, i) \in R$:

$$p_{uk} := p_{uk} + \eta q_{ik} \left(r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle\right),$$
$$q_{ik} := q_{ik} + \eta p_{uk} \left(r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle\right).$$

Второй подход основан на особенностях функционала (2.1) и называется ALS (alternating least squares). Можно показать, что этот функционал не является выпуклым в совокупности по P и Q, но при это становится выпуклым, если зафиксировать либо P, либо Q. Более того, оптимальное значение P при фиксированном Q (и наоборот) можно выписать аналитически, — но оно будет содержать обращение матрицы:

$$p_u = \left(\sum_{i:\exists r_{ui}} q_i q_i^T\right)^{-1} \sum_{i:\exists r_{ui}} r_{ui} q_i;$$
$$q_i = \left(\sum_{u:\exists r_{ui}} p_u p_u^T\right)^{-1} \sum_{u:\exists r_{ui}} r_{ui} p_u;$$

(здесь через p_u и q_i мы обозначили столбцы матриц P и Q).

Чтобы избежать сложной операции обращения, будем фиксировать всё, кроме одной строки p_k матрицы P или одной строки q_k матрицы Q. В этом случае можно найти оптимальное значение для p_k и q_k:

$$p_k = \frac{q_k (R - \sum_{s \neq k} p_s q_s^T)^T}{q_k q_k^T},$$
$$q_k = \frac{p_k (R - \sum_{s \neq k} p_s q_s^T)}{p_k p_k^T}.$$

Данный подход носит название Hierarchical alternating least squares (HALS) [**?**].

2.1.3 Учёт неявной информации

Выше мы обсуждали, что интерес пользователя к товару может выражаться по-разному. Это может быть как явный (выставление рейтинга или лайк, написание рецензии с оценкой), так и неявный (просмотр видео, посещение страницы) сигнал. Неявным сигналам нельзя доверять слишком сильно — пользователь мог по многим причинам смотреть страницу товара. При этом неявной информации гораздо больше, и поэтому имеет смысл использовать её при обучении моделей.

Один из способов учёта неявной информации предлагается в методе Implicit ALS (iALS) [**?**]. Введём показатель неявного интереса пользователя к товару:

$$s_{ui} = \begin{cases} 1, & \exists r_{ui}, \\ 0, & \text{иначе}. \end{cases}$$

Здесь мы считаем, что даже если пользователь поставил низкую оценку товару, то это всё равно лучше ситуации, в которой пользователь совсем не поставил оценку. Это не очень сильные рассуждения — пользователь мог просто не найти товар, и в таком случае неправильно судить об отсутствии интереса. Поэтому введём веса c_{ui}, характеризующие уверенность в показателе интереса s_{ui}:

$$c_{ui} = 1 + \alpha r_{ui}.$$

Коэффициент α позволяет регулировать влияние явного рейтинга на уверенность в интересе.

Теперь мы можем задать функционал:

$$\sum_{(u,i)\in D} c_{ui} \left(s_{ui} - \bar{s}_u - \bar{s}_i - \langle p_u, q_i \rangle\right)^2 + \lambda \sum_u \|p_u\|^2 + \mu \sum_i \|q_i\|^2 \to \min_{P,Q}$$

Как и раньше, обучать его можно с помощью стохастического градиентного спуска, ALS или HALS. Предложенные способы вычисления s_{ui} и c_{ui} могут изменяться в зависимости от специфики задачи.

§2.2 Контентные модели

В коллаборативной фильтрации используется информация о предпочтении пользователей и об их сходствах, но при этом никак не используются свойства самих пользователей или товаров. При этом мы можем обладать дополнительными данными — например, текстовыми описаниями или категориями товаров, данными из профиля пользователя. Из этих данных можно сформировать признаковое описание пары (пользователь, товар) и пытаться предсказывать рейтинг по этим признакам с помощью каких-либо моделей (линейных, композиций деревьев и т.д.).

Список литературы

[1] *Bishop, C.M.* Pattern Recognition and Machine Learning. // Springer, 2006.

[2] *Shawe-Taylor, J., Cristianini, N.* Kernel Methods for Pattern Analysis. // Cambridge University Press, 2004.

[3] *Sholkopf, B.A., Smola, A.J.* Learning with kernels. // MIT Press, 2002.

[4] *Micchelli, C.A.* Algebraic aspects of interpolation. // Proceedings of Symposia in Applied Mathematics, 36:81-102, 1986.