Занятие 4. Матричные разложения

Анализ данных и машинное обучение

Гирдюк Дмитрий Викторович

3 апреля 2021 г.

СПбГУ, ПМ–ПУ

Постановка задачи матричного разложения [1]

- Имеем матрицу X размерности $m \times n$
- Хотим получить ее разложение вида

$$X \approx UV^T,$$

где матрицы U и V размерности $m \times k$ и $n \times k$ соответственно, а $k \ll \min(m, n)$

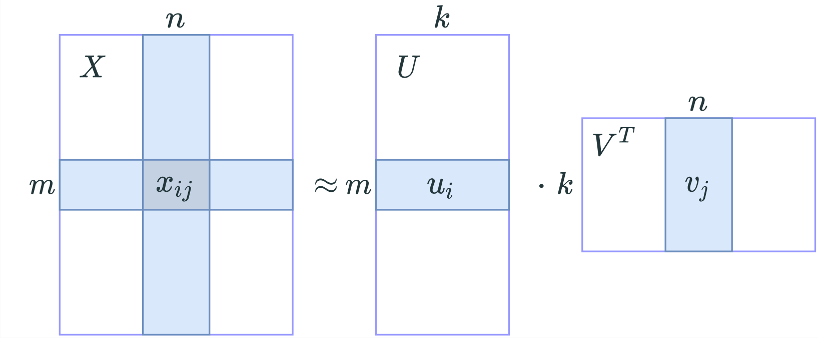
- Формализуя, хотим найти решение задачи

$$L(U, V) = ||X - UV^T||_F^2 = \sum_{i,j} (x_{ij} - \langle u_i, v_j \rangle)^2 \longrightarrow \min_{U,V}, \quad (1)$$

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

Постановка задачи матричного разложения

- Матрица $X_k = UV^T$ называется k-рановой аппроксимацией исходной матрицы X



Неотрицательное матричное разложение

- Часто на разложение накладывают дополнительное ограничение. Матрицы W и H в разложении $X = WH$ (такие обозначения приняты в этой области)
- Для чего нужно? Обычно все упирается в интерпретируемость разложения. В особенности в случае рекомендательных систем, где задача состоит в разложении матрицы X, по строкам которой, обычно, профили пользователей, а по столбцам – рекомендуемые объекты (товары, кино, etc)
- В качестве функционала качества разложения также наиболее популярна норма Фробениуса, к которой докидывают L_2 регуляризацию на столбцы матрицы W и строки матрицы H

Неотрицательное матричное разложение

- Регуляризация в том числе нужна для того, что решение задачи неединственно. В самом деле, существует множество таких матриц Y, что
$$\widehat{W} = WY, \quad \widehat{H} = Y^{-1}H$$
также является решением задачи. Конечно, при условии, что преобразования сохраняют неотрицательность матриц $\widehat{W}$ и $\widehat{H}$

Комментарии по терминологии

- В общем смысле матричные разложения – подраздел линейной алгебры, занимающийся исследованием различных способов разложения матрицы на составляющие
- Различных видов/типов матричных разложений достаточно много. Чаще всего используются для построения эффективных вычислительных алгоритмов. Например, для решения СЛАУ (LU-разложение, QR-разложение)
- Матричные разложения, обсуждаемые на этом занятии, скорее относятся к так называемой ранговой факторизации матрицы (на две матрицы меньшего ранга)

С какой целью используется

- Аппроксимация пропущенных значений матрицы. В простейшем случае заменяем пропущенные значения каким-либо образом (0, средние/медианы по столбцам/строкам/всей матрице, методы коллаборативной фильтрации в случае рекомендательной системы), после чего находим факторизованное представление ранга k
- Сжатие данных. Понятно, что если m и n большие, то $mn \gg k(m+n)$
- Избавление от шума. Если исходная матрица зашумленная, то ее факторизация позволит исключить существенную долю шума и небольшую долю «истинного» сигнала, давая в результате существенно более информативную матрицу (в теории)

Sigular-Value Decomposition (SVD) [2]

- Сингулярное разложение – специальное разложение матрицы, обобщающее спектральное разложение квадратной матрицы на случай прямоугольной матрицы
- Имеет широкое применение при решении многих прикладных задач. В частности, один из способов построить главные компоненты в PCA

SVD: общая теория

- Сингулярное разложение матрицы X размерности $m \times n$ – разложение матрицы на три составляющие

$$X = USV^T,$$

где U и V являются ортогональными матрицами размерности $m \times m$ и $n \times n$ соответственно, а матрица S есть диагональная матрица размерности $m \times n$ с отсортированными в порядке убывания элементами на диагонали

- Столбцы матрицы U называются левыми сингулярными векторами матрицы X, столбцы матрицы V – правыми сингулярными векторами матрицы X

SVD: общая теория

- Сингулярные числа неотрицательны!

$$\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_{\min(m,n)} \geqslant 0$$

- Левые сингулярные векторы матрицы X есть ничто иное как собственные векторы матрицы XX^T. Аналогично, правые сингулярные векторы – собственные векторы матрицы X^TX
- В самом деле

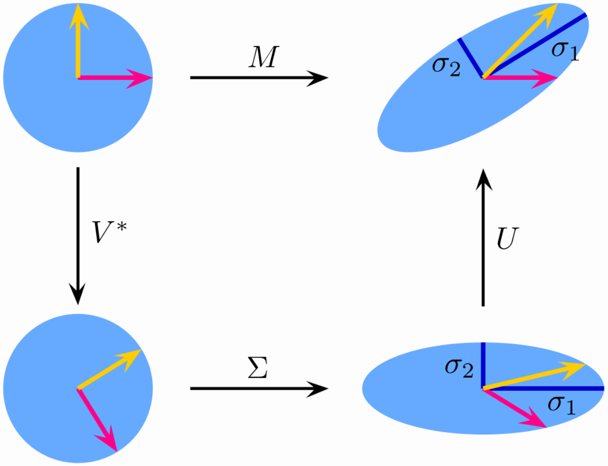
$$XX^T = USV^TVS^TU^T = USS^TU^T = US^2U^T \implies$$
$$\implies XX^TU = US^2$$

- Отсюда явно вытекает алгоритм построения разложения – используем библиотеки линейной алгебры для построения спектрального разложения матрицы XX^T и X^TX

SVD: общая теория

- Любая матрица X имеет SVD!
- Сингулярные значения матрицы уникальны. Если кратность какого-то из чисел больше 1, то соответствующие подпространства, порожденные левыми и правыми сингулярными векторами, определяются однозначно, но для каждого можно выбрать произвольный ортогональный базис
- Геометрически, соответствующий матрице X линейный оператор представляет собой последовательность из линейных операторов вращений и растяжения

SVD: графическая интерпретация



$$M = U \cdot \Sigma \cdot V^*$$

SVD: ранговая факторизация

- Итак, что мы имеем

$$A = \sum_{i=1}^{\min(m,n)} s_i \cdot u_i v_i^T$$

- Естественным образом аппроксимируем

$$A \approx A_k = \sum_{i=1}^{k} s_i \cdot u_i v_i^T$$

- Более того, подобное разложение оптимально с позиции нормы Фробениуса

$$||A - A_k||_F \leqslant ||A - B_k||_F,$$

  где матрица B_k есть произвольная матрица ранга k

SVD: флэшбеки с PCA

- Ранее, при обсуждении PCA, было указано, что SVD является одним из способов построения главных компонент
- Действительно

$$X^T X = VS^T U^T USV^T = VS^T SV = VDV^T$$

- Так что эффективнее: SVD или Power Iteration? Все упирается в k. Если достаточно лишь небольшого числа главных компонент, то второй метод выглядит наиболее адекватным выбором. Если k нужно достаточно больше, а то и вовсе требуется построить все n главных компонент, то следует использовать SVD

SVD: обсуждение i

- Как выбирать k? Аналогично тому, как это делается в PCA: кумулятивная сумма сингулярных значений
- Насколько эффективно? Понятно, что разложение для матрицы размером 1000000×1000000 придется ждать долго
- Не забываем про вставку пропущенных значений. Отсюда, вообще говоря, вытекают проблемы с точностью аппроксимации, переобучением и т.п.

SVD: обсуждение ii

- Более того, начинается путаница с понятиями. В рек. системах под SVD вообще понимается оптимизационная задача вида (1), которую обычно решают градиентными методами. Связано это с тем, что задача (1) напрямую связана с вычислением усеченных матриц из SVD.
- Отсюда, при чтении статей по использованию матричных разложений в рек. системах, будет множество различных модификаций аля Regularized SVD (Funk SVD), Improved Regularized SVD, SVD++ и других [3], напрямую к модификациям классического SVD отношения не имеющих

SGD и ALS

- Рассмотрим функционал (1) с регуляризацией и учетом лишь имеющихся данных в таблице (пусть на месте пропущенных значений стоят нули)

$$L(U,V) = \sum_{(i,j)\in \text{obs}} \left[(x_{ij} - \langle u_i, v_j \rangle)^2 + \lambda_U ||u_i||_2^2 + \lambda_V ||v_j||_2^2 \right] \quad (2)$$

- Способов регуляризаций достаточно. Встречаются и такие слагаемые

$$+\lambda \sum_{(i,j)\notin \text{obs}} \langle u_i, v_j \rangle$$

- В любом случае, наиболее популярными способами поиска матриц U и V являются стохастический градиентный спуск (Stochastic Gradient Descent, SGD) и метод попеременных квадратов (Alternating Least Squares, ALS)

SGD: общая теория

- Стохастический градиентный спуск применяется напрямую

$$\frac{\partial L}{\partial u_i} = \sum_{j|(i,j)\in\mathsf{obs}} 2\left(\langle u_i, v_j\rangle - x_{ij}\right) v_j + 2\lambda_U u_i$$

- Введем обозначение

$$\epsilon_{ij} = \left(\langle u_i, v_j\rangle - x_{ij}\right)$$

- Тогда итерация стохастического градиентного спуска будет состоять в последовательном обновлении столбцов искомых матриц

$$u_i^{(t+1)} := u_i^{(t)} - \alpha_t \left(\epsilon_{ij} v_j^{(t)} + \lambda_U u_i^{(t)}\right), \tag{3}$$

$$v_j^{(t+1)} := v_j^{(t)} - \beta_t \left(\epsilon_{ij} u_i^{(t)} + \lambda_V v_j^{(t)}\right) \tag{4}$$

SGD: псевдокод алгоритма

Algorithm 1: SGD для матричного разложения

input : Матрица X, размерность k, λ_U, λ_V, $iternum$

Инициализируем матрицы U и V;

for $t = 1, 2, \ldots, iternum$ **do**

  foreach $(i, j) \in$ *obs* **do**

    Обновляем u_i и v_j, используя формулы (3), (4);

SGD: обсуждение

Сходу возникает масса вопросов:

- Как инициализировать матрицы U и V, а также выбирать их размерность k?
- Как выбирать параметры регуляризации λ_U и λ_V?
- Что делать с лернинг рейтами $\alpha(t)$ и $\beta(t)$?
- А что если разложение нужно неотрицательное?

SGD: обсуждение

- Стратегий инициализации хватает. Самое популярное – произвольные значения из нормального/равномерного распределения (или пуассоновского для случая неотрицательного разложения)
- Остальные параметры существенно зависят от конкретной задачи. Особое внимание лернинг рейтам. Вообще говоря, они не должны быть константами
- Что до последнего, то тут могут выручить лернинг рейты. Подобрать их таким образом, чтобы обновление градиента было мультипликативным, причем предыдущее значение вектора домножалось на положительный коэффициент

SGD: обсуждение

- Метод весьма гибок: много гиперпараметров для настройки, работает с произвольными функциями потерь
- Несмотря на свою природу, а именно последовательное обновлений градиентов, алгоритм научились эффективно распаралелливать. Тема весьма обширная, предлагаю начать вот отсюда [4]
- Тем не менее, весьма медленно сходящийся. Особенно если облажаться с лернинг рейтами

ALS: общая теория

- Функционал (2) невыпуклый, однако при фиксации либо U, либо W, он становится квадратичным
- ALS использует это наблюдение для минимизации функционала путем решения последовательностей СЛАУ методом наименьших квадратов (МНК)
- Идея такая: в точке, где функционал достигает минимального значения, его производные по u_i и v_j будут равны нулю

$$\frac{\partial L}{\partial u_i} = \sum_{j|(i,j)\in\text{obs}} 2\left(\langle u_i, v_j\rangle - x_{ij}\right) v_j + 2\lambda_U u_i = 0 \implies$$

$$\implies \sum_{j|(i,j)\in\text{obs}} v_j \langle v_j, u_i\rangle + \lambda_U u_i = \sum_{j|(i,j)\in\text{obs}} x_{ij} v_j \implies$$

$$\implies \left(\sum_{j|(i,j)\in\text{obs}} v_j v_j^T + \lambda_U E\right) u_i = \sum_{j|(i,j)\in\text{obs}} x_{ij} v_j$$

ALS: общая теория

- Таким образом, суть алгоритма состоит в последовательном обновлении столбцов матриц U и V решая СЛАУ методом наименьших квадратов

$$\left(\sum_{j|(i,j)\in\mathsf{obs}} v_j v_j^T + \lambda_U E\right) u_i = \sum_{j|(i,j)\in\mathsf{obs}} x_{ij} v_j, \tag{5}$$

$$\left(\sum_{i|(i,j)\in\mathsf{obs}} u_i u_i^T + \lambda_V E\right) v_j = \sum_{i|(i,j)\in\mathsf{obs}} x_{ij} u_i \tag{6}$$

- Напомню

$$Ax = b \implies A^T A x = A^T b \implies x = \left(A^T A\right)^{-1} A^T b$$

ALS: псевдокод алгоритма

Algorithm 2: ALS

input : Матрица X, размерность k, λ_U, λ_V, $iternum$

Инициализируем матрицы U и V;

for $t = 1, 2, \ldots, iternum$ **do**

  foreach $i = 1, \ldots, m$ **do**

    Обновляем u_i на основе МНК для СЛАУ (5);

  foreach $j = 1, \ldots, n$ **do**

    Обновляем v_j на основе МНК для СЛАУ (6);

ALS: обсуждение

- Большинство проблем SGD всплывают и у ALS
- Рекомендации те же
- Что до плюсов/минусов
    - Распараллеливается
    - Сходимость быстрее, чем у SGD
    - Но опирается именно на квадрат ошибок
- Что до имплементаций: ALS в Apache Spark [5], ALS для implicit данных [6]. В sklearn'е есть класс NMF, реализующий иерархический ALS для случая неотрицательного разложения

1. *Гирдюк Д.* Репозиторий с материалами для занятий. URL: https://github.com/dmgirdyuk/PtW_ML_Unsupervised_learning.
2. *Roughgarden T.*, *Valiant G.* CS168: The Modern Algorithmic Toolbox Lecture №9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations. URL: https://web.stanford.edu/class/cs168/l/l9.pdf.
3. *Mehta R.*, *Rana K.* A review on matrix factorization techniques in recommender systems. //. 04.2017. С. 269—274. DOI: 10.1109/CSCITA.2017.8066567.
4. *Aktulum Ö. F.* Matrix Factorization With Stochastic Gradient Descent For Recommender Systems. URL: http://repository.bilkent.edu.tr/handle/11693/50632.

Использованные источники ii

5. Collaborative Filtering in Apache Spark. URL: https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html.
6. Fast Python Collaborative Filtering for Implicit Datasets. URL: https://implicit.readthedocs.io/en/latest/index.html.