Занятия 2 и 3. Снижение размерности. Матричные разложения

Анализ данных и машинное обучение

Гирдюк Дмитрий Викторович

19 марта 2021 г.

СПбГУ, ПМ–ПУ

Содержание

Снижение размерности: постановка задачи

Постановка задачи [1]

Задача *снижения размерности* состоит преобразование многомерных данных в осмысленное представление пониженной размерности с как можно более полным сохранением расстояний между наблюдениями

Формализуя,

- Имеем матрицу X размерности $m \times n$
- Предполагаем, что исходная размерность исследуемых данных равна k, и они представляют собой отображение в пространство большей размерности n, $k \ll n$
- Хотим получить исходное представление Y размерности $m \times k$
- Желательно иметь возможность без полного переобучения иметь возможность получить исходное представление для новых данных

С какой целью используется

- Избавление от лишнего (шума) и мультиколлинеарности фичей
- Меньше временные затраты на процессинг данных
- Визуализация

Общие проблемы

Все те же проблемы, что и с кластеризацией

- Масса подходов со своими целевыми функциями (No free lunch theorem)
- Необходимо знание предметной области
- Сложности в настройке гиперпараметров
- Хотя большинство методов непараметрические – отсутствует гибкость

Снижение размерности: общая классификация алгоритмов

Расширяем кругозор [2, 3] i

- Principal Component Analysis (PCA, Probabilistic PCA, Kernel PCA)
- Classical multidimensional scaling (MDS)
- Sammon mapping
- Linear Discriminant Analysis (LDA, Generalized DA)
- Factor Analysis (FA, Coordinated FA)
- Isometric feature mapping a.k.a ISOMAP (Landmark Isomap)
- Local Linear Embedding (LLE, Hessian LLE, Conformal Eigenmaps, Maximum Variance Unfolding)
- Laplacian Eigenmaps
- Local Tangent Space Alignment (LTSA, Linear LTSA)
- Landmark MVU (LandmarkMVU, FastMVU)

Расширяем кругозор [2, 3] ii

- Diffusion maps
- Neighborhood Preserving Embedding (NPE)
- Locality Preserving Projection (LPP)
- Stochastic Proximity Embedding (SPE)
- Local Linear Coordination (LLC)
- Manifold charting
- Gaussian Process Latent Variable Model (GPLVM)
- Stochastic Neighbor Embedding (SNE, Symmetric SNE, t-SNE)
- Neighborhood Components Analysis (NCA)
- Maximally Collapsing Metric Learning (MCML)
- Large-Margin Nearest Neighbor (LMNN)
- UMAP
- Deep autoencoders

Общая классификация

- Методы, оптимизирующие выпуклую целевую функцию без локальных минимумов
  - Полное спектральное разложение
    - PCA, Kernel PCA, Multidimensional scaling, ISOMAP, MVU, Diffusion maps
  - Частичное спектральное разложение
    - Laplacian eigenmaps, LLE, Hessian LLE, LTSA
- Методы, оптимизирующие невыпуклую целевую функцию с локальными минимумами:
  - Sammon mapping, LLC, Manifold charting, Deep autoencoders, t-SNE, UMAP

Снижение размерности: основные алгоритмы

Principal Component Analysis (PCA)

- Главными компонентами некоторого набора данных $X_{[m \times n]}$ является последовательность из n векторов, каждый из которых наилучшим образом (в смысле минимизации средних квадратов расстояний между наблюдениями и текущим вектором) подгоняется под данные, при этом каждый $i-$ый вектор ортогонален предыдущим $i-1$ векторам
- Новый ортогональный базис? Новый ортогональный базис!
- Principal component analysis (PCA, метод главных компонент) – метод снижения размерности, основанный на построении набора из первых k таких ортогональных векторов

PCA: общая теория

- Хотим получить такой нормированный ортогональный набор векторов $v_j \in R^n, j = 1, 2, \ldots, k$, которыми можно будет саппроксимировать исходные векторы $x_i \in R^n, i = 1, 2, \ldots, m$

$$x_i \approx \sum_{j=1}^{k} a_{ij} v_j, \quad i = 1, 2, \ldots, m$$

- Нормируем (опционально шкалируем) данные!
- От простого к сложному: целевая функция (минимизация дисперсии/разброса проекции точек на главную компоненту) в случае $k = 1$

$$\frac{1}{m} \sum_{i=1}^{m} ||x_i \perp v||_2 \longrightarrow \min_{||v||_2 = 1}$$

PCA: общая теория

- Теорема Пифагора позволяет преобразовать целевую цункцию

$$||x_i \perp v||_2^2 + \langle x_i, v\rangle^2 = ||x_i||_2^2 \implies$$

$$\implies \frac{1}{m}\sum_{i=1}^{m}\langle x_i, v\rangle^2 = \frac{1}{m}(Xv)^T(Xv) = \frac{1}{m}v^T X^T X v =$$

$$= \frac{1}{m}v^T A v \longrightarrow \max_{||v||_2=1} \quad (1)$$

- Симметричная матрица $A = X^T X$ – штука известная. Ничто иное как ковариационная (учитывая шкалирование, еще и корреляционная) матрица
- Целевая функция для случай $k > 1$, проекция x_i на векторное подпространство $V = \{v_1, \ldots, v_k\}$

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\langle x_i, v_k\rangle^2 \longrightarrow \max_{V:||v_k||_2=1}$$

PCA: общая теория

- Теперь рассмотрим разложение матрицы A

$$A = VDV^T,$$

где матрица V есть ортогональная матрица, а D – диагональная

- Заметим, что если матрица $A = \text{diag}(\lambda_1, \ldots, \lambda_n)$ (пусть еще $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$), то максимум для (1) достигается для $v = e_1$

$$v^T Av = \sum_{i=1}^{n} \lambda_i v_i^2$$

- Отсюда следует, что для произвольной матрицы A, положив $v_1 = Ve_1$, получим

$$v_1^T Av_1 = v_1^T VDV^T v1 = e_1^T V^T VDV^T Ve_1 = e_1^T De_1 = \lambda_1$$

PCA: общая теория

- Более того, для любого другого вектора $\hat{v}$, $\hat{v}^T A v \leqslant \lambda_1$
- Для случая k > 1 все выводится аналогичным образом. Оптимальное решение – первые k столбцов матрицы V
- Вопрос: как эту матрицу V искать-то?
- На самом деле матрица V в разложении $A = VDV^T$ есть ничто иное как матрица, столбцы которой являются собственными векторами матрица $A = X^T X$

$$Av_i = AVe_i = VDV^T Ve_i = VDe_i = \lambda_i Ve_i = \lambda_i v_i$$

- Последнее, что тут стоит отметить, это уникальность решения: если все собственные числа уникальны, то и разложение уникально. Если же встречаются кратные, то образуется целое подпространство собственных векторов, решающих задачу

PCA: алгоритмы поиска компонент

- Основанный на Singular Value Decomposition (SVD, в другой раз)
- Итерационный алгоритм (Power Iteration)

PCA: итерационный алгоритм

Algorithm 1: Итерационный алгоритм поиска главной компоненты

input : Матрица $A = X^T X$

Выбираем произвольный нормированный вектор u_0;

for $i = 1, 2, \ldots$ **do**

    $u_i = A^i u_0$;

    if $u_i / ||u_i||_2 \approx u_{i-1} / ||u_{i-1}||_2$ **then**

        Возвращаем u_i;

- Важно отметить (доказывается по индукции):
  $A^{i+1} = A^i A = V D^i V^T V D V^T = V D^i V^T$
- Как только нашли первую компоненту, проектируем данные ортогонально найденной главной компоненте, т.е. полагаем $x_i := x_i - \langle x_i, v_i \rangle v_i$ и запускаем итерационный алгоритм заново

PCA: обсуждение

- Выбираем число главных компонент на основе объясненной дисперсии равной кумулятивной сумме отнормированных собственных чисел, соответствующих собственным векторам (главным компонентам). Или используйте правило Кайзера:

$$\lambda_i > \frac{1}{n}\text{trace A}$$

- Еще раз, нормализуем (и шкалируем) данные!
- Интерпретация главных компонент зачастую затруднительна
- Нелинейная структура – увы. Используйте другой метод
- Например, Kernel PCA! Все отличие лишь в том, что находим собственные векторы не ковариационной матрицы $X^T X$, а ядровой матрицы $K : k_{ij} = \kappa(x_i, x_j)$
- Самое главное: $Y_k = XV_k$

Multidimensional Scaling

- Multidimensional scaling – семейство техник для снижения размерности (чаще всего для визуализации), цель которых состоит в сохранении расстояний между наблюдениями в пространстве меньшей размерности
- Из интересного – алгоритм работает не с точками напрямую, а с матрицей расстояний между ними
- Существует масса вариаций этой общей идеи. Рассмотрим классический алгоритм и то, что сейчас используется в соответствующих пакетах как sota

Classical Multidimensional Scaling: общая теория

- Немного формализма: имеем матрицу $D_{[n\times n]} = \{d_{ij}\}$, хотим найти такие $y_j \in R^k, j = 1, 2, \ldots, m, k < n$, чтобы

$$d_{ij} \approx ||y_i - y_j||_2$$

- Стандартизируем!
- Понятно, что решение не единственно: $Y + const$ также будет удовлетворять основному требованию. Потому вводится дополнительное ограничение

$$\sum_{i=1}^{m} y_{ip} = 0, \quad p = 1, \ldots, k \tag{2}$$

Classical Multidimensional Scaling: общая теория

- Теперь рассмотрим матрицу Грама $B = YY^T$ (не путать с Y^TY)

$$||y_i - y_j||_2 = y_i^T y_i + y_j^T y_j - 2y_i^T y_j \implies d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij} \quad (3)$$

- Кроме того из (2)

$$\sum_{i=1}^{m} b_{ij} = \sum_{i=1}^{m}\sum_{p=1}^{k} y_{ip}y_{jp} = \sum_{p=1}^{k} y_{jp}\sum_{i=1}^{m} y_{ip} = 0, \quad j = 1, \ldots, m$$

- Из (3)

$$\sum_{i=1}^{m} d_{ij}^2 = \mathrm{tr}B + mb_{jj}, \quad \sum_{j=1}^{m} d_{ij}^2 = \mathrm{tr}B + mb_{ii}, \quad \sum_{j=1}^{m}\sum_{i=1}^{m} d_{ij}^2 = 2m\mathrm{tr}B, \quad (4)$$

Classical Multidimensional Scaling: общая теория

- Наконец, из (3), (4) имеем

$$b_{ij} = -1/2(d_{ij}^2 - d_{.j}^2 - d_{i.}^2 + d_{..}^2)$$

- Ну а дальше для полученной матрицы B как и в PCA находим разложение уже известного вида $B = VDV^T$
- Что дает нам $Y = V_k D_k^{1/2}$
- Точное решение для случая евклидовой нормы. Впрочем, может быть использован и для неевклидовых метрик

Metric/Nonmetric Multidimensional Scaling

- Есть вариант, называющийся Metric MDS. Задача переформулируется в виде минимизации функционала (называемого stress'ом)

$$Stress(Y) = \left(\sum_{i,j=1|i\neq j}^{m} (d_{ij} - ||y_i - y_j||_2)^2 \right)^{\frac{1}{2}} \longrightarrow \min_Y \qquad (5)$$

- Тут уже используются всевозможные оптимизационные методы. В свое время Крускал предлагал вариацию градиентного спуска
- Есть и Nonmeric MDS, где метрика расстояния $f(y_i, y_j)$ неевклидова, вообще говоря, лишь монотонна и сохраняет отношение порядка (чаще всего там работа с рангами наблюдений)

Metric Multidimensional Scaling: SMACOF

- На данный момент для нахождения многообразия с помощью Metric MDS чаще всего используется итеративный алгоритм Scaling by MAjorizing a COmplicated Function, SMACOF (в том числе в sklearn)
- В основе всего следующая стресс-функция

$$\sigma(Y) = \sum_{i<j\geq m} w_{ij} \left(\widehat{d}_{ij}(Y) - d_{ij}\right)^2 \longrightarrow \min_{Y} \tag{6}$$

Metric Multidimensional Scaling: SMACOF

- Ее ограничивают сверху

$$\sigma(Y) = \sum_{i<j} w_{ij} d_{ij}^2 + \sum_{i<j} w_{ij} \widehat{d}_{ij}(Y)^2 - 2 \sum_{i<j} w_{ij} d_{ij}^2 d_{ij}^2(Y) =$$
$$= \text{Const} + \text{tr}\left[Y^T V Y\right] - 2\text{tr}\left[Y^T B(Y) Y\right] \leqslant$$
$$\leqslant \text{Const} + \text{tr}\left[Y^T V Y\right] - 2\text{tr}\left[Y^T B(Z) Z\right] = \tau(Y, Z), \quad (7)$$

где $B(Z)$ определяется следующим образом

$$b_{ij} = -\frac{w_{ij} d_{ij}}{\widehat{d}_{ij}(Z)} \text{ for } \widehat{d}_{ij}(Z) \neq 0, i \neq j$$

$$b_{ij} = \epsilon \text{ for } \widehat{d}_{ij}(Z) = 0, i \neq j$$

$$b_{ij} = -\sum_{j=1|j\neq i}^{m} b_{ij}$$

Metric Multidimensional Scaling: SMACOF

Algorithm 2: SMACOF

Произвольно инициализируем матрицу Y_0 размерности $m \times k$;

for $i = 1, 2, \ldots$ **do**

  $Z = Y_{k-1}$;

  $Y_k = \arg\min_Y \tau(Y, Z)$ (7);

  if $\sigma(Y_{k-1}) - \sigma(Y_k) < \varepsilon$ **then**

    Возвращаем Y_k;

Multidimensional Scaling: обсуждение

- Classical MDS с евклидовыми расстояниями практически полностью анологичен классическому PCA
- Чаще всего используется все же именно для визуализации
- Устоявшийся и все еще актуальный (набор) алгоритм с долгой историей
- Все те же проблемы с существенной нелинейностью многообразия как и у PCA

ISOMAP (Isometric feature mapping) [4]

- ISOMAP – пример нелинейного алгоритма снижения размерности, фактически обобщающим MDS путем работы не с евклидовыми расстояниями между наблюдениями, а геодезическими расстояниями на основе взвешенного графа, подогнанного к наблюдениям на основе метода k-ближайших соседей.
- Проще говоря
  - Строим граф на основе k-ближайших соседей для матрицы X
  - Строим матрицу попарных расстояний для наблюдений x_i, x_j путем применения какого-либо алгоритма поиска кратчайших путей на графе (Дийкстра, Флойд–Уоршелл)
  - Полученную матрицу загоняем в классический MDS

ISOMAP: обсуждение

- В sklearn'е вместо MDS используется Kernel PCA, который, как уже было отмечено, эквивалентен metric MDS
- В целом, удачное обобщение, тем не менее страдающее от некоторых недостатков [2]
  - Топологическая нестабильность – проблема с замкнутыми циклами может существенно ухудшить производительность метода
  - Проблемы с дырами в многообразии
  - Проблемы с невыпуклыми многообразиями

Local Linear Embedding (LLE) [5]

- Local Linear Embedding – нелинейная техника снижения размерности, основанная на построении графа k-ближайших соседей и попыткой поиска локальных весов для восстановления исходных наблюдений по их ближайшим соседям
- Основная идея состоит в том, что веса, позволяющие восстановить наблюдение по его соседям, могут быть использованы с той же целью в пространстве меньшей размерности (ввиду предположения о том, что наблюдение и его ближайшие соседи лежат на линейном многообразии)

LLE: общая теория

- Имея на руках граф соседей, построим матрицу W, такую что

$$\mathcal{E}(W) = \sum_{i=1}^{m} \left| x_i - \sum_{j : x_j \in Nb(x_i)} w_{ij} x_j \right|^2 \longrightarrow \min_W,$$

  причем, $w_{ij} = 0$, если x_i и x_j не являются соседями, а также $\sum_j w_{ij} = 1$
- Веса, удовлетворяющие ограничениям, обладают важным свойствам: при поворотах, шкалировании или сдвигах осей они остаются неизменны

LLE: общая теория

- Для нахождения весов применяются следующее соображение

$$\varepsilon_i = \left| x_i - \sum_j w_{ij}\eta_j \right|^2 = \left| \sum_j w_{ij}(x_i - \eta_j) \right|^2 = \sum_{j,p} w_{ij}w_{ip}C_{ijp},$$

  где $C_{ijp} = \langle x_i - \eta_j, x_i - \eta_p \rangle$

- Ну а дальше задача условной оптимизации методом множителей Лагранжа

LLE: общая теория

- Найдя веса, рассматриваем аналогичную оптимизационную задачу

$$\Phi(Y) = \sum_{i=1}^{m} \left| y_i - \sum_j w_{ij} y_j \right|^2$$

- Как и при рассмотрении MDS, накладываем ограничение на Y

$$\sum_{i=1}^{m} y_i p = 0, p = 1, \ldots, k,$$

- Решается путем построения собственных векторов матрицы

$$M = (E - W)^T (E - W),$$

которые и являются искомыми эмбеддингами

LLE: обсуждение

- Простой и интуитивно понятный способ снижения размерности
- По духу очень близок с ISOMAP, но сама идея на первый взгляд выглядит куда более «правильной»
- Есть куча расширений и обобщений, аля Hessian LLE, LTSA и другие

Laplacian eigenmaps

- Laplacian eigenmaps во многом схож с LLE: поиск низкоуровневого представления данных с попыткой сохранить локальные свойства искомого многообразия.
- В данном случае под локальными свойствами подразумеваются расстояния между наблюдением и его k-ближайшеми соседями
- Фактически, изучили, когда обсуждали спектральную кластеризацию

Продолжение следует

Снижение размерности: обсуждение

Матричные разложения: постановка задачи

Матричные разложения: общая классификация алгоритмов

Матричные разложения: основные алгоритмы

Матричные разложения: обсуждение

Использованные источники i

1. *Гирдюк Д.* Репозиторий с материалами для занятий. URL: https://github.com/dmgirdyuk/PtW_ML_Unsupervised_learning.
2. *Van Der Maaten L.*, *Postma E.*, *Van den Herik J.* Dimensionality reduction: a comparative review. // J Mach Learn Res. 2009. Т. 10. С. 66—71.
3. *Van der Maaten L.* Подробный список классических методов снижения размерности. URL: https://lvdmaaten.github.io/drtoolbox/.
4. *Tenenbaum J. B.*, *Silva V. d.*, *Langford J. C.* A Global Geometric Framework for Nonlinear Dimensionality Reduction. // Science. 2000. Т. 290, № 5500. С. 2319—2323. ISSN 0036-8075. DOI: 10.1126/science.290.5500.2319. URL: https://science.sciencemag.org/content/290/5500/2319.

Использованные источники ii

5. *Saul L.*, *Roweis S.* An introduction to locally linear embedding. // Journal of Machine Learning Research. 2001. Янв. Т. 7.