Занятие 12. Снижение размерности

Гирдюк Дмитрий Викторович

9 декабря 2023

СПбГУ, ПМ–ПУ, ДФС

Содержание

Постановка задачи

Постановка задачи

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

Формализуя,

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

С какой целью используется

- Избавление от лишнего (шума) и мультиколлинеарности признаков.
- Меньшие временные затраты на обработку данных.
- Визуализация.

Общие проблемы

Все те же проблемы, что и с кластеризацией

- Масса подходов со своими целевыми функциями.
- Необходимо знание предметной области.
- Сложности в настройке гиперпараметров.
- Большинство методов непараметрические – отсутствует гибкость.

Общая классификация алгоритмов

Расширяем кругозор [1, 2] 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)
- Maximum Variance Unfolding (MVU, LandmarkMVU, FastMVU)

Расширяем кругозор [1, 2] 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)
- LargeVis
- Neighborhood Components Analysis (NCA)
- Maximally Collapsing Metric Learning (MCML)
- Large-Margin Nearest Neighbor (LMNN)
- UMAP
- Deep autoencoders

Общая классификация

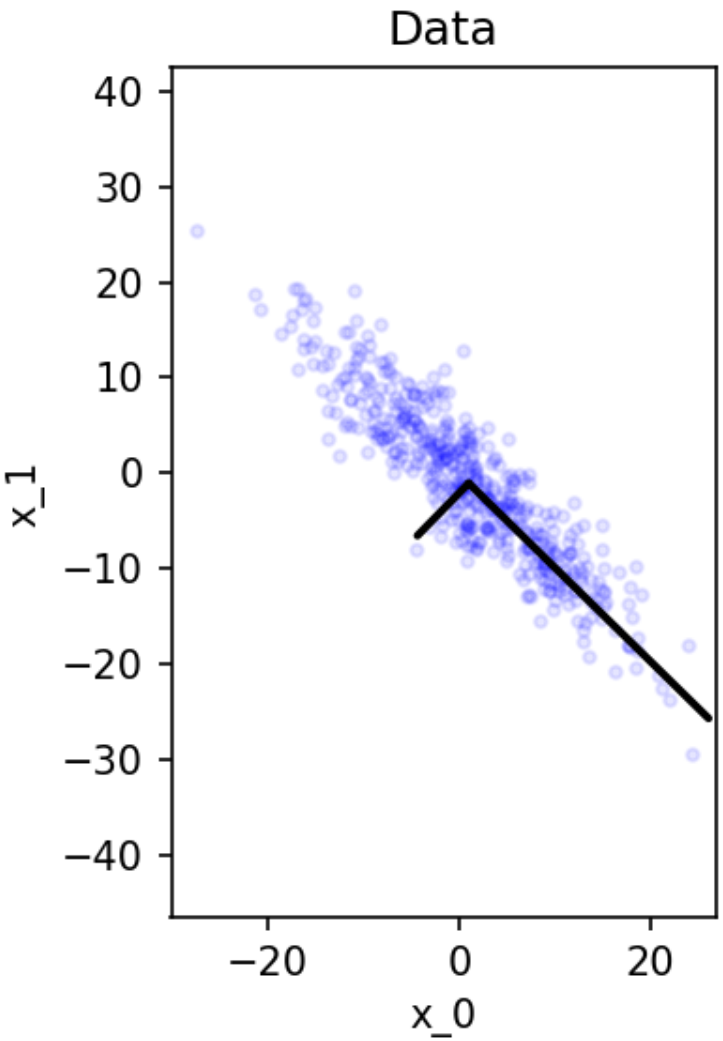
- Методы, оптимизирующие выпуклую целевую функцию без локальных минимумов: PCA, Kernel PCA, Multidimensional scaling, ISOMAP, MVU, Diffusion maps, Laplacian eigenmaps, LLE, Hessian LLE, LTSA.
- Методы, оптимизирующие невыпуклую целевую функцию с локальными минимумами: Sammon mapping, LLC, Manifold charting,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

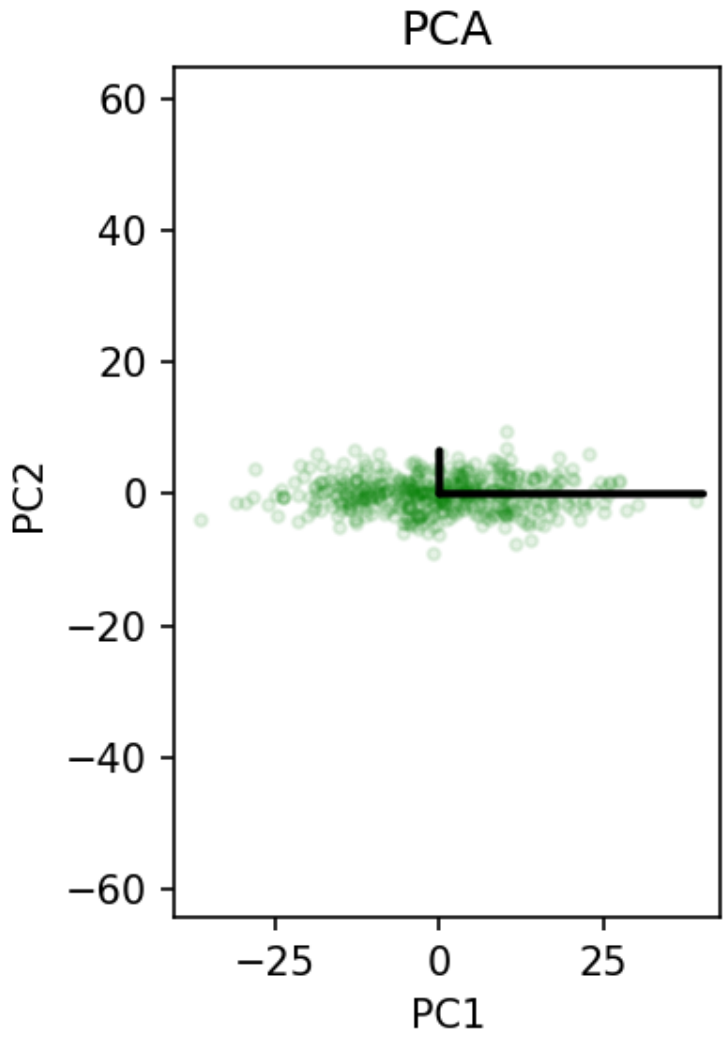
PCA

Pricnipal component analysis

- Главными компонентами некоторого набора данных $\boldsymbol{X}_{[m\times n]}$ 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!
- Метод главных компонент (Principal component analysis, PCA) – метод снижения размерности, основанный на построении набора из первых k таких ортогональных векторов.

PCA: визуализация





PCA: Общая теория 1

- Хотим получить такой нормированный ортогональный набор векторов $\boldsymbol{v}_j \in R^n, j = 1, 2, \ldots, d$, которыми можно будет саппроксимировать исходные векторы $\boldsymbol{x}_i \in R^n, i = 1, 2, \ldots, m$

$$\boldsymbol{x}_i \approx \sum_{j=1}^{d} c_{ij} \boldsymbol{v}_j, \quad i = 1, 2, \ldots, m$$

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

$$\frac{1}{m} \sum_{i=1}^{m} ||\boldsymbol{x}_i \perp \boldsymbol{v}||_2^2 \longrightarrow \min_{||\boldsymbol{v}||_2 = 1}$$

PCA: Общая теория 2

- Теорема Пифагора позволяет преобразовать целевую цункцию

$$||\boldsymbol{x}_i \perp \boldsymbol{v}||_2^2 + \langle \boldsymbol{x}_i, \boldsymbol{v} \rangle^2 = ||\boldsymbol{x}_i||_2^2 \implies$$

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

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

- Симметричная матрица $\frac{1}{m}\boldsymbol{X}^T\boldsymbol{X}$ есть ничто иное как эмпирическая ковариационная матрица. Важное свойство: собственные числа такой матрицы неотрицательны.
- Целевая функция для случай $d > 1$, проекция $\boldsymbol{x}_i$ на векторное подпространство $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{d}\langle \boldsymbol{x}_i, \boldsymbol{v}_j \rangle^2 \longrightarrow \max_{\boldsymbol{V}:||\boldsymbol{v}_j||_2=1}$$

PCA: Общая теория 3

- Теперь рассмотрим разложение матрицы $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$$

где матрица $\boldsymbol{V}$ есть ортогональная матрица, а $\boldsymbol{D}$ – диагональная.

- Заметим, что если матрица $\boldsymbol{A} = \text{diag}(\lambda_1, \ldots, \lambda_n)$ (пусть еще $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n \geqslant 0$), то максимум для (1) достигается для $\boldsymbol{v} = \boldsymbol{e}_1$

$$\boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} = \sum_{j=1}^{n} \lambda_j \boldsymbol{v}_j^2$$

- Отсюда следует, что для произвольной матрицы $\boldsymbol{A}$, положив $\boldsymbol{v}_1 = \boldsymbol{V}\boldsymbol{e}_1$, получим

$$\boldsymbol{v}_1^T\boldsymbol{A}\boldsymbol{v}_1 = \boldsymbol{v}_1^T\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T\boldsymbol{v}_1 = \boldsymbol{e}_1^T\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{e}_1 = \boldsymbol{e}_1^T\boldsymbol{D}\boldsymbol{e}_1 = \lambda_1$$

PCA: Общая теория 4

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

$$\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{V}\boldsymbol{e}_i = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{e}_i = \boldsymbol{V}\boldsymbol{D}\boldsymbol{e}_i = \lambda_i\boldsymbol{V}\boldsymbol{e}_i = \lambda_i\boldsymbol{v}_i$$

- 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 главных компонент

- Основанный на сингулярном разложении (Singular Value Decomposition, SVD).
- Итерационный алгоритм (Power Iteration).

PCA: Сингулярное разложение (SVD)

- Сингулярное разложение

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$$

где $\boldsymbol{U}_{[m\times m]}$ и $\boldsymbol{V}_{[n\times n]}$ ортогональные матрицы, а $\boldsymbol{S}$ – диагональная матрица размерности $m \times n$, значения на диагонали которой отсортированы в убывающем порядке.

- Столбцы матриц $\boldsymbol{U}$ и $\boldsymbol{V}$ называются левыми и правыми сингулярными векторами матрицы $\boldsymbol{X}$ соответственно.

PCA: Сингулярное разложение (SVD)

- Левые сингулярные векторы матрицы $\boldsymbol{X}$ есть ничто иное как собственные векторы матрицы $\boldsymbol{X}\boldsymbol{X}^T$. Аналогично, правые сингулярные векторы – собственные векторы матрицы $\boldsymbol{X}^T\boldsymbol{X}$.
- В самом деле

$$\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{D}_1\boldsymbol{U}^T \implies$$
$$\implies \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{D}$$

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{S}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}_2\boldsymbol{V}^T \implies$$
$$\implies \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}$$

- Отсюда следует алгоритм нахождения главных компонент: с помощью библиотек линейной алгебры построить сингулярное разложение матрицы $\boldsymbol{X}$. Правые сингулярные векторы и квадраты сингулярных чисел определяют собственные векторы и собственные числа матрицы $\boldsymbol{X}^T\boldsymbol{X}$.

PCA: Итерационный алгоритм

Algorithm 1: Итерационный алгоритм поиска главной компоненты

input : Матрица $\boldsymbol{A} = \boldsymbol{X}^T\boldsymbol{X}$

Выбираем произвольный нормированный вектор $\boldsymbol{u}_0$;

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

  $\boldsymbol{u}_i = \boldsymbol{A}^i\boldsymbol{u}_0$;

  if $\boldsymbol{u}_i/||\boldsymbol{u}_i||_2 \approx \boldsymbol{u}_{i-1}/||\boldsymbol{u}_{i-1}||_2$ **then**

    Возвращаем $\boldsymbol{u}_i$;

- Важно отметить (доказывается по индукции): $\boldsymbol{A}^{i+1} = \boldsymbol{A}^i\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}^i\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{D}^{i+1}\boldsymbol{V}^T$
- Как только нашли первую компоненту, проектируем данные ортогонально найденной главной компоненте, т.е. полагаем $\boldsymbol{x}_i := \boldsymbol{x}_i - \langle \boldsymbol{x}_i, \boldsymbol{v}_1\rangle\boldsymbol{v}_1$ и запускаем итерационный алгоритм заново.

PCA: обсуждение итерационного алгоритма

Почему последовательное домножение на вектор $\boldsymbol{u}_0$ матрицы $\boldsymbol{A}$ приводит нас в конечном итоге к главной компоненте?

- $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$
- $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}$
- Учитывая то, что по столбцам у матрицы $\boldsymbol{V}$ стоят собственные векторы, формирующие ортогональный базис, то любой вектор в этом базисе может быть расписан как $\boldsymbol{u}_0 = \boldsymbol{c}_1\boldsymbol{v}_1 + \ldots + \boldsymbol{c}_n\boldsymbol{v}_n$.
- Тогда получаем

$$\boldsymbol{A}^i\boldsymbol{u}_0 = c_1\boldsymbol{A}^i\boldsymbol{v}_1 + \ldots + c_n\boldsymbol{A}^i\boldsymbol{v}_n = c_1\lambda_1^d\boldsymbol{v}_1 + \ldots + c_n\lambda_n^d\boldsymbol{v}_n =$$

$$= \lambda_1^d(c_1\boldsymbol{v}_1 + c_2\frac{\lambda_2^d}{\lambda_1^d}\boldsymbol{v_2} + \ldots + c_n\frac{\lambda_n^d}{\lambda_1^d}\boldsymbol{v}_n)$$

- Откуда следует, что при $i \to \infty$, учитывая $\lambda_1 \geqslant \ldots \geqslant \lambda_n$, $\boldsymbol{A}^i\boldsymbol{u}_0 \to C(i)\boldsymbol{v}_1$

PCA: Обсуждение i

- Простая интерпретация метода главных компонент: компоненты последовательно выбираются так, чтобы дисперсия проекции данных на нее была максимальной. Следует это из центрированности данных и вида целевой функции (1).
- Выбираем число главных компонент на основе объясненной дисперсии, равной кумулятивной сумме отнормированных собственных чисел, соответствующих собственным векторам (главным компонентам). Или используем правило Кайзера:

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

- Еще раз, нормализуем (и шкалируем) данные!
- Интерпретация главных компонент зачастую затруднительна.

PCA: Обсуждение ii

- Нелинейная структура в данных – используйте другой метод (рассмотрим позже в курсе).
- Например, Kernel PCA! Все отличие лишь в том, что находим собственные векторы не ковариационной матрицы $\boldsymbol{X}^T\boldsymbol{X}$, а ядровой матрицы $\boldsymbol{K}$: $k_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
- Самое главное: $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V}$. Но мы можем взять лишь первые d компонент для аппроксимации: $\boldsymbol{Y}_d = \boldsymbol{X}\boldsymbol{V}_d$.

MDS

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.
- Алгоритм работает не с объектами напрямую, а с матрицей расстояний между ними.
- Зачастую используется для визуализации.
- Существует масса вариаций этой общей идеи. Рассмотрим классический алгоритм и то, что сейчас используется в современных пакетах.

Классический MDS: общая теория

- Немного формализма: имеем матрицу $\boldsymbol{D}_{[m\times m]} = \{d_{ij}\}_{i,j=1}^{n}$, хотим найти такие $\boldsymbol{y}_j \in R^d, d < n, j = 1, 2, \ldots, m$, чтобы

$$d_{ij} \approx ||\boldsymbol{y}_i - \boldsymbol{y}_j||_2$$

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

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

Классический MDS: общая теория

- Теперь рассмотрим матрицу Грама $\boldsymbol{B} = \boldsymbol{Y}\boldsymbol{Y}^T$ (не путать с $\boldsymbol{Y}^T\boldsymbol{Y}$)

$$||\boldsymbol{y}_i - \boldsymbol{y}_j||_2^2 = \boldsymbol{y}_i^T\boldsymbol{y}_i + \boldsymbol{y}_j^T\boldsymbol{y}_j - 2\boldsymbol{y}_i^T\boldsymbol{y}_j \implies d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij} \quad (3)$$

- Кроме того из (2)

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

- Из (3)

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

Классический MDS: общая теория

- Наконец, из (3), (4) имеем

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

- А дальше для полученной матрицы $\boldsymbol{B}$, как и в PCA, находим разложение уже известного вида $\boldsymbol{B} = \boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$.
- Что дает нам $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, но, как и в PCA, можно взять лишь первые d собственных векторов $\boldsymbol{Y}_d = \boldsymbol{U}_d\boldsymbol{\Lambda}_d^{\frac{1}{2}}$.

Метрический MDS

- Метрический MDS. Исходная задача подменяется задачей минимизации функционала, называемого "стрессом"

$$\text{Stress}(\boldsymbol{Y}) = \sigma(\boldsymbol{Y}) = \sum_{i<j\leqslant m} w_{ij}\left(\delta_{ij} - \widehat{d}_{ij}(\boldsymbol{Y})\right)^2 \longrightarrow \min_{\boldsymbol{Y}} \qquad (5)$$

  где δ_{ij} есть расстояния между объектами i и j в исходном n-мерном пространстве.

- Чтобы найти представление данных в пространстве меньшей размерности с помощью метрического MDS, чаще всего используют итеративный алгоритм SMACOF (Scaling by MAjorizing a COmplicated Function). Он и реализован в sklearn.

Метрический MDS: итеративная мажоризация

Центральная идея алгоритма мажоризации заключается в итеративной замене исходной сложной функции $f(x)$ вспомогательной функцией $g(x, z)$, где z есть некоторое фиксированное значение. Функция g должна соответствовать следующим требованиям:

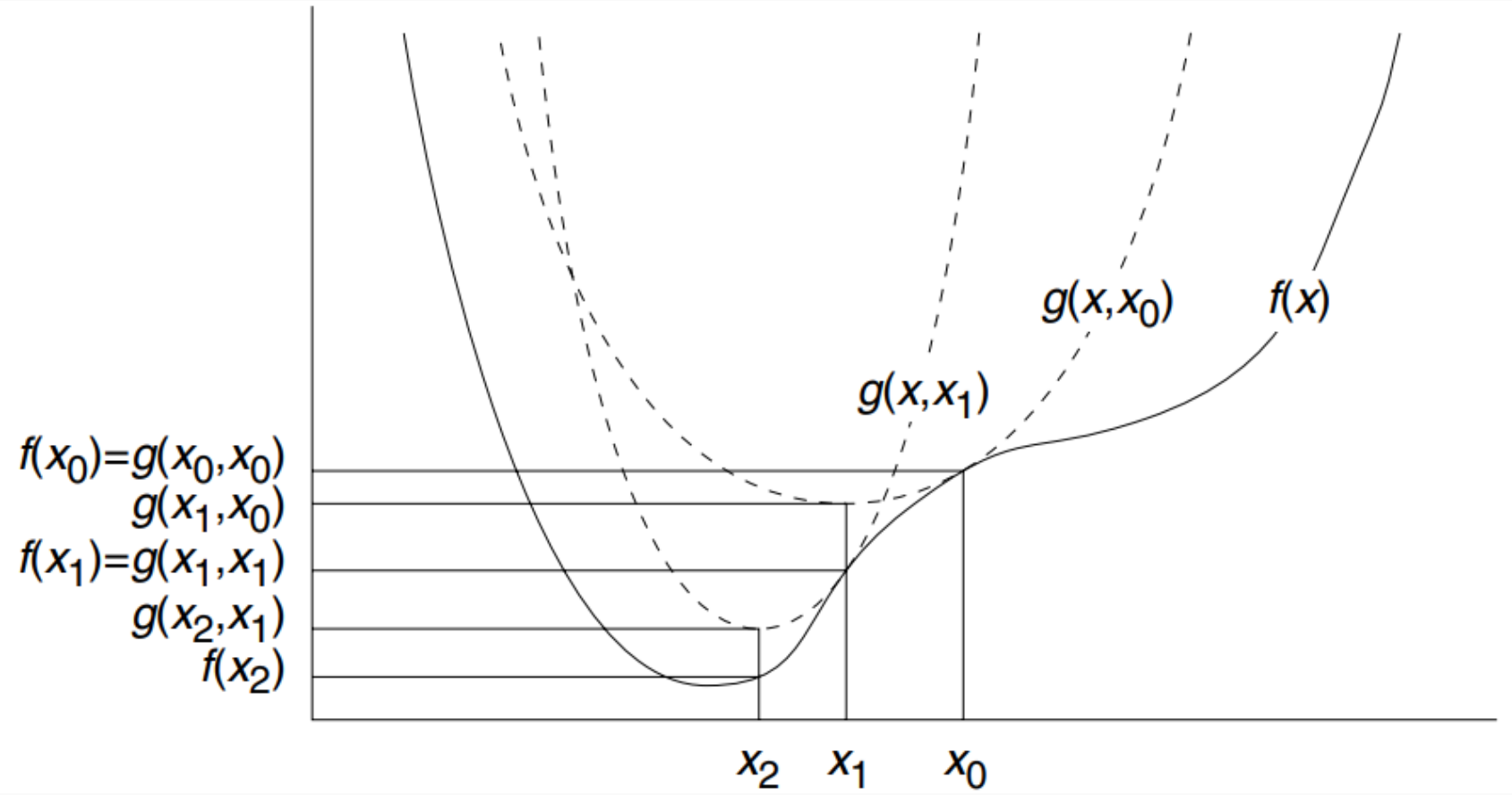
- Очевидно, функция $g(x, z)$ должна быть проще, чем $f(x)$. Квадратичной или даже линейной.
- $f(x) \leqslant g(x, z)$.
- Вспомогательная функция должна касаться функции в так называемом саппорте (supporting point): $f(z) = g(z, z)$.

Из этих правил следует

$$f(\widehat{x}) \leqslant g(\widehat{x}, z) \leqslant g(z, z) = f(z)$$

где $\widehat{x}$ – минимум $g(x, z)$.

Метрический MDS: визуализация итеративной мажоризации [3]



Метрический MDS: SMACOF

- Функция Stress может быть ограничена сверху

$$\sigma(\boldsymbol{Y}) = \sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}\hat{d}_{ij}^2(Y) - 2\sum_{i<j} w_{ij}\delta_{ij}\hat{d}_{ij}(Y) =$$

$$= \text{Const} + \text{tr}\left[\boldsymbol{Y}^T\widehat{\boldsymbol{W}}\boldsymbol{Y}\right] - 2\text{tr}\left[\boldsymbol{Y}^T B(\boldsymbol{Y})\boldsymbol{Y}\right] \leqslant$$

$$\leqslant \text{Const} + \text{tr}\left[\boldsymbol{Y}^T\widehat{\boldsymbol{W}}\boldsymbol{Y}\right] - 2\text{tr}\left[\boldsymbol{Y}^T B(\boldsymbol{Z})\boldsymbol{Z}\right] = \tau(\boldsymbol{Y}, \boldsymbol{Z}) \quad (6)$$

где $\boldsymbol{Z}$ – саппорт и $B(Z)$ определяется следующим образом

$$b_{ij} = -\frac{w_{ij}\delta_{ij}}{\hat{d}_{ij}(Z)} \text{ при } \hat{d}_{ij}(Z) \neq 0, i \neq j$$

$$b_{ij} = \epsilon \text{ при } \hat{d}_{ij}(\boldsymbol{Z}) = 0, i \neq j$$

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

Метрический MDS: SMACOF

Algorithm 2: SMACOF

Произвольно инициализируем матрицу $\boldsymbol{Y}_0$ размерности $m \times d$;

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

  $\boldsymbol{Z} = \boldsymbol{Y}_{i-1}$;

  $\boldsymbol{Y}_i = \arg\min_{\boldsymbol{Y}} \tau(\boldsymbol{Y}, \boldsymbol{Z})$ (6);

  if $\sigma(\boldsymbol{Y}_{i-1}) - \sigma(\boldsymbol{Y}_i) < \varepsilon$ **then**

    Возвращаем $\boldsymbol{Y}_i$;

Порядковый MDS

- Существует также порядковая версия MDS, где метрика $\rho(\boldsymbol{y}_i, \boldsymbol{y}_j)$ неевклидова, лишь монотонна и сохраняет отношение порядка. Чаще всего вместо исходных объектов используются их ранги.
- А задача нахождения эмбеддингов распространяется и на аппроксимацию $f(\boldsymbol{y}_i, \boldsymbol{y}_j)$. Для тех, кому интересно, обратите внимание на книгу [3].

MDS: обсуждение

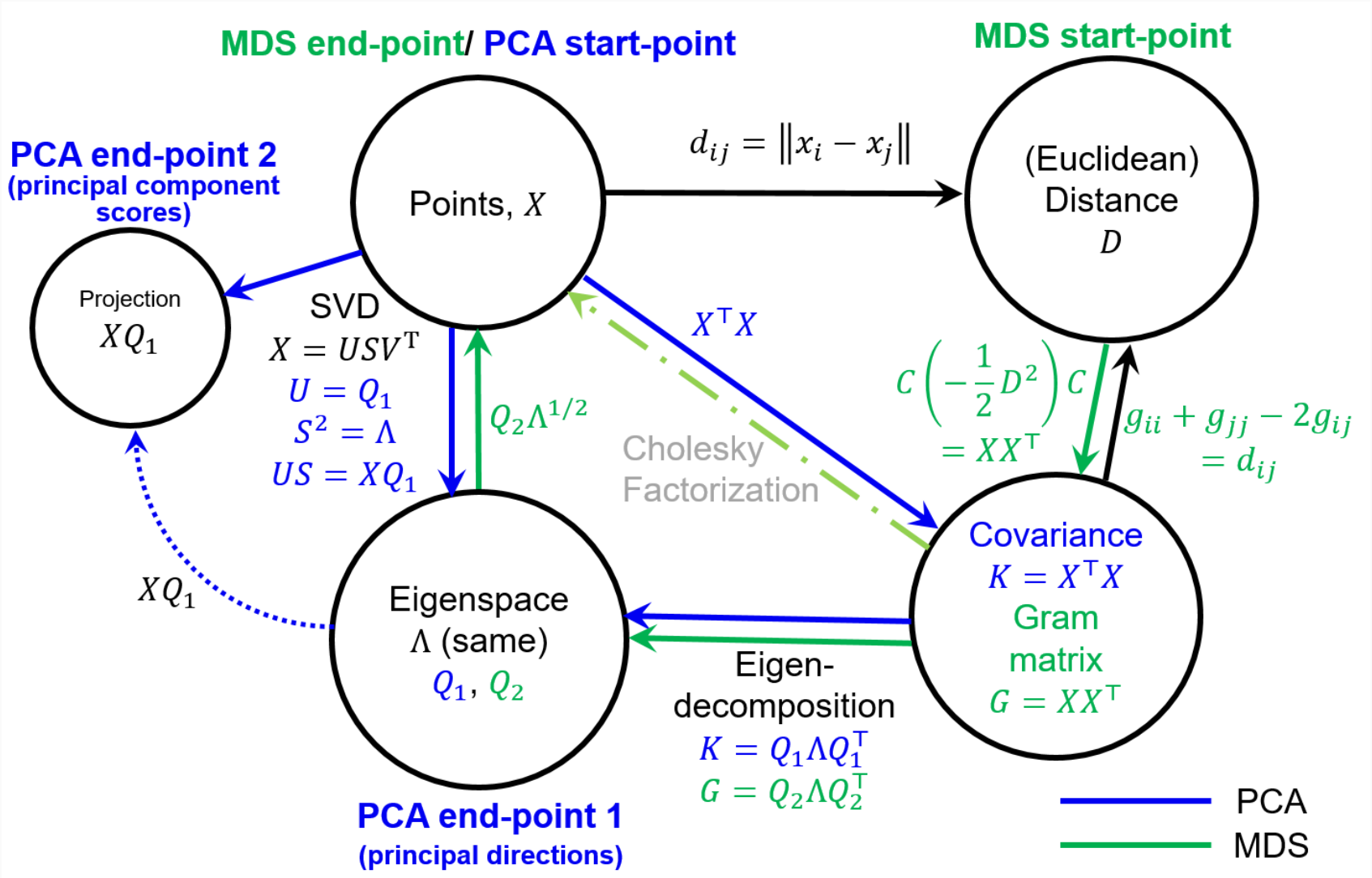
- Чаще всего используется все же именно для визуализации.
- Устоявшийся и все еще актуальный алгоритм (набор алгоритмов) с долгой историей.
- Все те же проблемы с существенной нелинейностью многообразия как и у PCA.

Классический MDS эквивалентен PCA

Кстати, классический MDS эквивалентен PCA.

- Пусть $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{Q}_1\boldsymbol{\Lambda}_1\boldsymbol{Q}_1^T$ and $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{Q}_2\boldsymbol{\Lambda}_2\boldsymbol{Q}_2^T$.
- Пусть также $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ и, помня про ортогональность $\boldsymbol{U}$ и $\boldsymbol{V}$, получаем $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{V}^T$. Но при этом $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{U}^T$. Таким образом, $\boldsymbol{Q}_1 = \boldsymbol{V}$, $\boldsymbol{Q}_2 = \boldsymbol{U}$, и т.к. $\boldsymbol{S}$ диагональна, то $\boldsymbol{\Lambda}_{1[n\times n]} \equiv \boldsymbol{\Lambda}_{2[m\times m]}$.
- $\implies \boldsymbol{X}_{[m\times n]}\boldsymbol{V}_{[n\times n]} = \boldsymbol{U}_{[m\times m]}\boldsymbol{S}_{[m\times n]}$.

Классический MDS эквивалентен PCA [4]

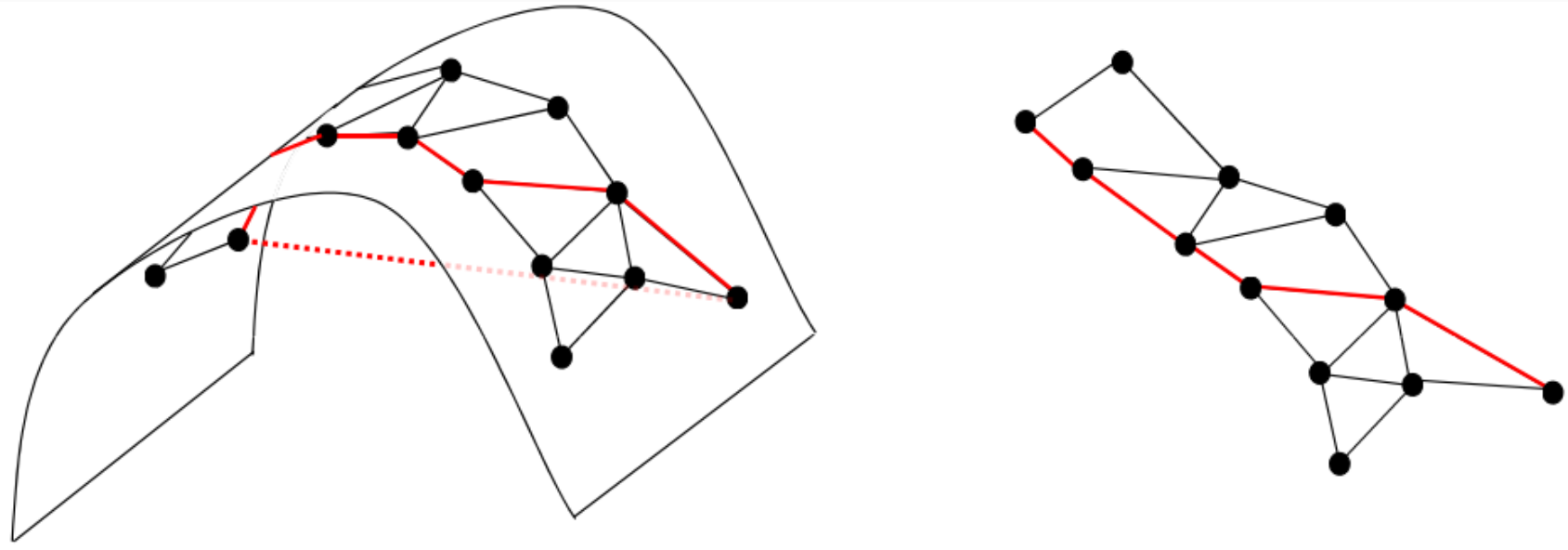


ISOMAP, LLE и Laplacian Eigenmaps

ISOMAP (Isometric feature mapping) [5]

- 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 $\boldsymbol{X}$.
  - Строим матрицу попарных расстояний для наблюдений $\boldsymbol{x}_i, \boldsymbol{x}_j$ путем применения какого-либо алгоритма поиска кратчайших путей на графе (Дийкстра, Флойд–Уоршелл).
  - Полученную матрицу подаем на вход в классический MDS.

ISOMAP: визуализация [6]



ISOMAP: обсуждение

- В sklearn'е вместо MDS используется PCA, который, как уже было отмечено, эквивалентен классическому MDS.
- В целом, удачное обобщение, тем не менее страдающее от некоторых недостатков [1]
  - Топологическая нестабильность – проблема с замкнутыми циклами может существенно ухудшить производительность метода.
  - Проблемы с дырами в многообразии.
  - Проблемы с невыпуклыми многообразиями.

Local Linear Embedding (LLE) [7]

- 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

- Используя граф соседей, построим матрицу $\boldsymbol{W}$, такую что

$$\mathcal{E}(\boldsymbol{W}) = \sum_{i=1}^{m} \left| \boldsymbol{x}_i - \sum_{j:\boldsymbol{x}_j \in Nb(\boldsymbol{x}_i)} w_{ij}\boldsymbol{x}_j \right|^2 \longrightarrow \min_{\boldsymbol{W}},$$

  причем, $w_{ij} = 0$, если $\boldsymbol{x}_i$ и $\boldsymbol{x}_j$ 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 следующие соображения

$$\varepsilon_i = \left| \boldsymbol{x}_i - \sum_j w_{ij} \boldsymbol{\eta}_j \right|^2 = \left| \sum_j w_{ij} (\boldsymbol{x}_i - \boldsymbol{\eta}_j) \right|^2 = \\ = \sum_{j,p} w_{ij} w_{ip} \langle \boldsymbol{x}_i - \boldsymbol{\eta}_j, \boldsymbol{x}_i - \boldsymbol{\eta}_p \rangle$$

- Ну а дальше задача условной оптимизации методом множителей Лагранжа.

LLE: общая теория

- Найдя веса, рассматриваем аналогичную оптимизационную задачу

$$\Phi(\boldsymbol{Y}) = \sum_{i=1}^{m} \left| \boldsymbol{y}_i - \sum_j w_{ij} \boldsymbol{y}_j \right|^2 \longrightarrow \min_Y$$

- Как и при рассмотрении MDS, накладываем ограничение на $\boldsymbol{Y}$

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

- А также ограничение на равенство ковариационной матрицы единичной матрице

$$\frac{1}{m} \sum_{i=1}^{m} \boldsymbol{y}_i \boldsymbol{y}_i^T = \boldsymbol{E}$$

LLE: общая теория

- Отметим, что целевая функция является функцией квадратичной

$$\Phi(Y) = \sum_{i=1}^{m} ||\boldsymbol{Y}^T \mathbf{1}_i - \boldsymbol{Y}^T \boldsymbol{w}_i||_2^2 = ||\boldsymbol{Y}^T (\boldsymbol{E} - \boldsymbol{W})^T||_F^2 =$$

$$= \text{tr}\left(\boldsymbol{Y}^T (\boldsymbol{E} - \boldsymbol{W})^T (\boldsymbol{E} - \boldsymbol{W}) Y\right) = \text{tr}\left(\boldsymbol{Y}^T \boldsymbol{M} \boldsymbol{Y}\right) \quad (7)$$

- Было показано [7], что решение данной задачи состоит в нахождении набора из $d+1$ собственных векторов матрицы M, которые соответствуют $d+1$ наименьшим собственным числам (все собственные числа неотрицательны).

LLE: обсуждение

- Простой и интуитивно понятный способ снижения размерности.
- Первый собственный вектор представляет собой вектор, состоящий из единиц (соответствующее собственное значение равно 0, лапласиан $E - W$) и должен быть исключен.
- По духу очень близок с ISOMAP, но сама идея на первый взгляд выглядит куда более «правильной».
- Есть куча расширений и обобщений, например 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.
- С точки зрения практики алгоритм прост: строим лапласиан графа, находим его собственные векторы, выбираем первые d.

Laplacian Eigenmaps: лапласиан графа

Лапласианом графа называется матрица $L = D - W$ с набором интересных свойств

- $z^\intercal L z = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(z_i - z_j)^2$ для любого $z \in R^n$.
- L симметрична и положительно полуопределена.
- У L n неотрицательных собственных чисел: $0 = \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$.
- Наименьшее собственное число λ_1 всегда равно 0. Соответствующий ему собственный вектор состоит из 1, если граф связный. Если у графа есть p компонент связанности, то кратность λ_1 равна p, а собственные векторы представляют собой индикаторные векторы.

t-SNE

t-SNE

- t-distributed Stohastic Neighbour Embedding (t-SNE) [8] – нелинейный алгоритм снижения размерности, разработанный для визуализации в (обычно) двух/трехмерном пространстве эмбеддингов.
- Он основанн на построении вероятностного распределения расстояний между парами наблюдений в исходном пространстве и пространстве эмбеддингов, и вычислении их [эмбеддингов] путем оптимизации дивергенции Кульбака–Лейбера между этими двумя распределениями.

t-SNE: общая теория i

- Распределение вероятностей на расстояния между парами наблюдений i и j в исходном пространстве

$$p_{j|i} = \frac{\exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_k)^2/2\sigma_i^2)}, \quad p_{i|i} = 0 \tag{8}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2m} \tag{9}$$

- Из (9) следует, что

$$\sum_{j=1}^{m} p_{ij} > \frac{1}{2m}$$

- Распределение вероятностей (на основе распределения Стьюдента) на расстояния в пространстве эмбеддингов

$$q_{ij} = \frac{\left(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2\right)^{-1}}, \quad q_{i|i} = 0. \tag{10}$$

t-SNE: общая теория ii

- Среднеквадратическое отклонение подбирается на основе гиперпараметра, называемого перплексией

$$u = Perp(P_i) = 2^{H(P_i)} = 2^{-\sum_j p_{j|i} \log p_{j|i}},$$

где $H(P_i)$ есть энтропия Шэннона от случайной величины P_i

- Чаще всего запускается бинарный поиск значения σ_i, ввиду того что монотонное увеличение среднеквадратической ошибки приводит к монотонному увеличению перплексии.
- Вычисление эмбеддингов $\boldsymbol{y}_i, i = 1, \ldots, m$ происходит на основе оптимизации КЛ-дивергенции между распределениями P и Q, являющимися совместными распределениями на все расстояния между парами наблюдений

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE: общая теория iii

- Градиент дивергенции найти не сложно

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2)^{-1} \tag{11}$$

- t-распределение позволяет улучшить ситуация с так называемой «проблемой скученности» (crowding problem), состоящей в том, что наблюдения, находящиеся на средней дистанции от рассматриваемого наблюдения в исходном многомерном пространстве, в, скажем, двухмерном пространстве, должны будут находиться куда дальше относительно того, как будут находиться друг относительно друга рассматриваемое наблюдение и наблюдения, находящиеся очень близко от него.

t-SNE: общая теория iv

- Следовательно, если мы пытаемся точно моделировать небольшие расстояния, большинство наблюдений со средними попарными расстояниями будут расположены слишком далеко друг от друга. Но тогда в ходе оптимизации все такие наблюдения будут сжиматься к центру в пространстве меньшей размерности (большая часть p_{ij} становится равной константе).

t-SNE: замечания об оригинальном SNE i

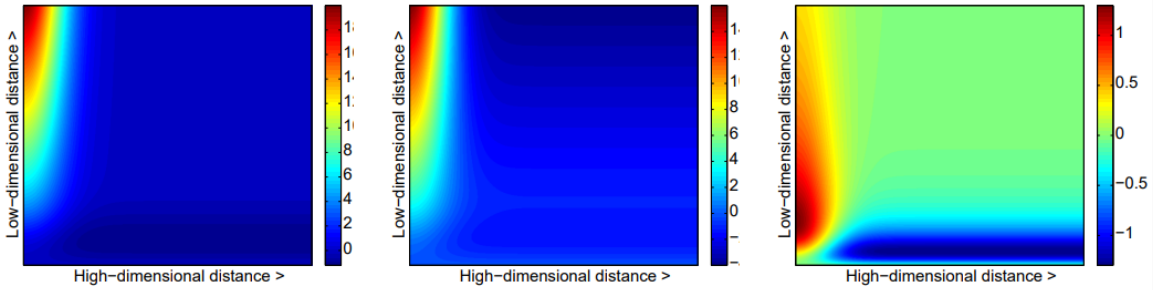
- В изначальном алгоритме SNE вместо распределения Стьюдента использовалось нормальное распределение с $\sigma_i = 1$.
- SNE минимизирует значение КЛ-дивергенции между условными вероятностями $p_{j|i}$ и $q_{j|i}$, но эта метрика не симметрична. Следовательно, различные ошибки в попарных расстояниях в пространстве меньшей размерности имеют разный вес.
- Например, больший вес для представления объектов, что находятся далеко друго от друга в пространстве эмбеддингов, но близко в исходном пространстве, чем когда объекты в пространстве эмбеддингов находятся близко, а в исходном пространстве далеко друг от друга.
- Следовательно, сохраняется только локальная структура данных (как в LLE).

t-SNE: замечания об оригинальном SNE ii

- Более того, процесс оптимизации очень нестабилен и требует множества ухищрений (добавление гауссова шума, изменение дисперсии этого шума во время процесса, добавление momentum и т.д.).

t-SNE: замечания о распределении Стъюдента i

- Так как t-SNE сопоставляет вероятности, существует естественный способ устранения проблемы скученности: используем нормальное распределение в исходном пространстве и распределение Стъюдента в пространстве эмбеддингов.
- Почему степень свободы равна 1? Упрощает формулу для расчета, а расстояния (почти) соответствуют обратной квадратичной зависимости для больших попарных расстояний.



(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

Figure 1: Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation.

t-SNE: замечания о распределении Стъюдента ii

- t-SNE фокусируется на (1) моделировании непохожих наблюдений посредством больших попарных расстояния и (2) моделировании близких наблюдений посредством небольших попарных расстояний.

t-SNE: оптимизация

Что до оптимизации, то

- инициализируют $\boldsymbol{y}_i$ обычно либо значениями из нормального распределения с нулевым матожиданием и очень маленькой дисперсией, либо на основе PCA.
- при градиентном спуске кроме слагаемого с лернинг рейтом и антиградиентом докидывают экспоненциально убывающую разность между предыдущим и предпредыдущим значениями градиентов

$$\boldsymbol{Y}^t = \boldsymbol{Y}^{t-1} - \eta \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{y}} + \alpha(t)(\boldsymbol{Y}^{t-1} - \boldsymbol{Y}^{t-2}) \tag{12}$$

t-SNE: алгоритм

Algorithm 3: Классический t-SNE

input : $u, \eta, \alpha(t)$

Вычисляем p_{ij} по формуле (9);

Произвольно инициализируем матрицу $\boldsymbol{Y}_0$ размерности $m \times d$;

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

- Вычисляем q_{ij} по формуле (10);
- Вычисляем градиент $\frac{\partial C}{\partial \boldsymbol{y}_i}$ по формуле (11);
- Обновляем значения $\boldsymbol{Y}^t$ по формуле (12) ;

t-SNE: еще пара слов об оптимизации в алгоритме

- На начальном этапе работы домножают p_{ij} на коэффициент, называемый «early exaggregation». Это позволяет в пространстве меньшей размерности работать с более кучными и хорошо разделенными кластерами относительно кластеров в исходных данных.
- Рекомендованные значения на перплексию лежат в пределах от 5 до 50. Впрочем, для больших датасетов это значение можно увеличить. Сама по себе перплексия может быть рассмотрена как эффективное количество соседей рассматриваемого наблюдения.

How to Use t-SNE Effectively [9]

- Гиперпараметры имеют принципиальное значение (особое внимание, разумеется, перплексии).
- Размеры кластеров в пространстве эмбеддингов не имеют значения.
- Расстояние между кластерами может также не иметь значения.
- Случайный шум порой может образовывать что-то «необычное».
- Порой могут появляться интересные формы кластеров в пространстве эмбеддингов, не свойственные исходным данным.
- Чтобы убедиться в наличие специфической топологии данных, необходимо строить графики с различными значениями перплексии.

t-SNE: оптимизация алгоритма

- Трудоемкость t-SNE во многом связана с квадратичной сложностью при вычислении p_{ij} и градиента (11).
- Первая идея, позволяющая улучшить положение дел, состоит в построении для каждого наблюдения $\lfloor 3u \rfloor$-ближайших соседей и вычисление p_{ij_k}. Остальные значения полагаются равными нулю. Соседей в статье предлагается искать, строя VP-дерево (vantage point). Вычислительная сложность $O(un \log n)$.
- Модификация алгоритма на основе алгоритма Барнеса-Хата [10] позволяет аппроксимировать значения градиентов путем построения дерева специального вида для хранения y_i, снижая вычислительную сложность этого шага до $O(n \log n)$.

t-SNE: алгоритм Барнеса–Хата

- Распишем градиент (11) следующим образом

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4\sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2)^{-1} =$$

$$= 4\sum_j (p_{ij} - q_{ij}) q_{ij} Z(\boldsymbol{y}_i - \boldsymbol{y}_j) =$$

$$= 4\left(\sum_j p_{ij} q_{ij} Z(\boldsymbol{y}_i - \boldsymbol{y}_j) - \sum_j q_{ij}^2 Z(\boldsymbol{y}_i - \boldsymbol{y}_j)\right) \quad (13)$$

  где $Z = \sum_{k \neq l} \left(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2\right)^{-1}$.
- Первое слагаемое считается за $O(un)$ с учетом первой идеи по аппроксимации P и ввиду того, что $q_{ij}Z = (1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2)^{-1}$.
- Второе слагаемое может быть саппроксимировано за $O(n \log n)$ путем построения дерева квадрантов (quadtree; для двумерного эмбеддинга). Для трехмерного случая можно использовать октодерево (octtree).

t-SNE: алгоритм Барнеса–Хата

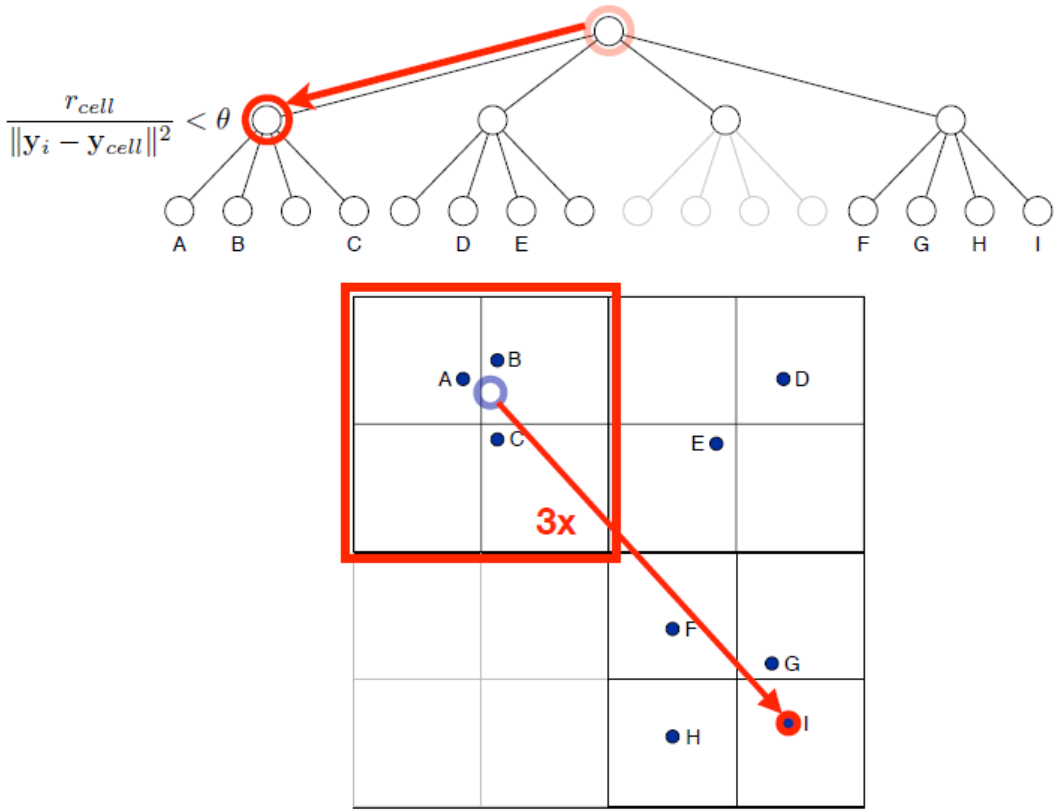


Figure 2: Illustration of the Barnes-Hut approximation. To evaluate the t-SNE gradient for point I, the Barnes-Hut algorithm performs a depth-first search on the embedding quadtree, checking at every node whether or not the node may be used as a "summary". In the illustration, the cell containing points A, B, and C satisfies the summary-condition: the force between the center-of-mass of the three points (which is stored in the quadtree node) and point I is computed, multiplied by the number of points in the cell (*i.e.*, by three), and added to the gradient for point I. All children of the summary node are pruned from the depth-first search.

t-SNE: алгоритм Барнеса–Хата

- Используя построенное дерево, можем саппроксимировать второе слагаемое следующим образом

$$-q_{ij}^2 Z(\boldsymbol{y}_i - \boldsymbol{y}_j) \approx q_{i,cell}^2 Z(\boldsymbol{y}_i - \boldsymbol{y}_{cell}),$$

  где $\boldsymbol{y}_{cell}$ – центр масс точек в квадранте.
- Небольшой хак: находим аппроксимацию для значения $-q_{ij}^2 Z^2(\boldsymbol{y}_i - \boldsymbol{y}_j)$ и делим на Z (приблизительное значение которого также подсчитываем в процессе) ввиду того, что из себя представляет $q_{ij}Z$.
- Критерий останова для квадранта следующий

$$\frac{r_{cell}}{||\boldsymbol{y}_i - \boldsymbol{y}_{cell}||^2} < \theta,$$

  где r_{cell} – длина диагонали квадранта.
- θ – это параметр angle в sklearn.

t-SNE: обсуждение

- Пожалуй, самый популярный нелинейный метод визуализации многомерных данных в (обычно) двухмерном/трехмерном пространстве.
- Метод гибок, количество параметров больше, чем у среднестатистического нелинейного алгоритма снижения размерности.
- Тем не менее, обычно достаточно посмотреть на различные значения перплексии (и попробовать проварьировать лернинг рэйт при оптимизации КЛ-дивергенции).

t-SNE: обсуждение

Основные недостатки:

- Трудности в интерпретации [9].
- Трудоемок (в классической имплементации алгоритма вычислительная сложность равна $O(n^2)$).
- Модификация на основе VP-дерева и алгоритма Барнеса–Хата (дефолт в sklearn'e) снижает это число до $O(un \log n)$, но ограничивает применение для эмбеддингов размерности 2 или 3.

Обсуждение

Обсуждение i

- Классификация алгоритмов разнится. Кроме указанной в начале презентации (из работы van der Maaten'a [1]), можно выделить следующую:
  - алгоритмы, которые стремятся сохранить попарную структуру расстояний среди всех наблюдений (например, PCA, MDS);
  - и те, которые поддерживают сохранение локальных расстояний по сравнению с глобальным расстоянием (ISOMAP, LLE, t-SNE).
- PCA полностью идентичен классическому MDS.
- Применение MDS для геодезических расстояний эквивалентно применению ISOMAP.

Обсуждение ii

- Общий паттерн алгоритмов [для удобного запоминания], основанных на построении графа k-ближайших соседей (ISOMAP, LLE, Laplacian eigenmaps, даже t-SNE), выглядит следующим образом:
  1. Построение взвешенного графа k-ближайших соседей и применение некоторой трансформации над расстояниями между вершинами, чтобы отразить локальную структуру данных (не забыть о симметрии).
  2. Подобрать специальную целевую функцию (отражающую желаемые свойства), на основе оптимизации которой построить эмбеддинг в пространстве меньшей размерности.

Обсуждение iii

- Расхождений между теорией и имплементацией алгоритмов в sklearn обнаружено не было. Однако есть тонкости, связанные с различными оптимизационными трюками (t-SNE и алгоритм Барнеса–Хата), построением Лапласиана графа (еще его называют матрицей Кирхгофа), etc.

Использованные источники i

1. *Van Der Maaten L.*, *Postma E.*, *Van den Herik J.* **Dimensionality reduction: a comparative review.** // J Mach Learn Res. 2009. T. 10. C. 66—71.
2. *Van der Maaten L.* **Dimensionality reduction techniques list.** URL: https://lvdmaaten.github.io/drtoolbox/.
3. *Borg I.*, *Groenen P. J. F.* **Modern Multidimensional Scaling Theory and Applications.** New York : Springer, 2005. ISBN 038728981X. DOI: 10.1007/0-387-28981-X.
4. *Ng Y. K.* **cMDS equivalence to PCA.** URL: https://www.quora.com/Whats-the-difference-between-MDS-and-PCA.

Использованные источники ii

5. *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.
6. *Wilson P. R. C.* **Similarities, Distances and Manifold Learning.** URL: http://simbad-fp7.eu/images/tutorial/02-ECCV2012Tutorial.pdf.
7. *Saul L.*, *Roweis S.* **An introduction to locally linear embedding.** // Journal of Machine Learning Research. 2001. Янв. Т. 7.
8. *van der Maaten L.*, *Hinton G.* **Visualizing High-Dimensional Data Using t-SNE.** // Journal of Machine Learning Research. 2008. Т. 9. С. 2579—2605.

Использованные источники iii

9. *Wattenberg M.*, *Viegas F.*, *Johnson I.* **How to use t-SNE Effectively.** URL: https://distill.pub/2016/misread-tsne/.

10. *Maaten L. van der.* **Accelerating t-SNE using Tree-Based Algorithms.** // Journal of Machine Learning Research. 2014. Т. 15, № 93. С. 3221—3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.