

Занятие 12. Снижение размерности

Гирдюк Дмитрий Викторович

21 декабря 2024

СПбГУ, ПМ-ПУ, ДФС

1. Постановка задачи
2. Общая классификация алгоритмов
3. PCA
4. MDS
5. ISOMAP, LLE и Laplacian Eigenmaps
6. t-SNE
7. Обсуждение

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 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 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.

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

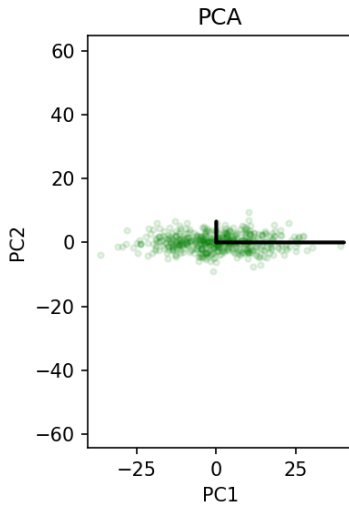
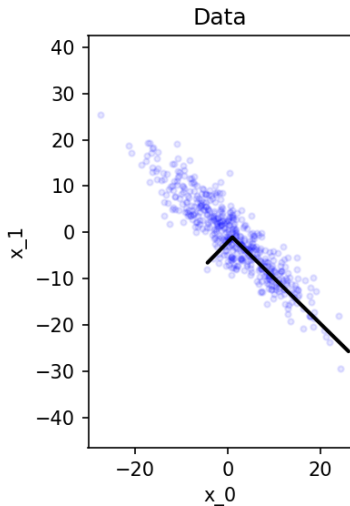
- Методы, оптимизирующие выпуклую целевую функцию без локальных минимумов: PCA, Kernel PCA, Multidimensional scaling, ISOMAP, MVU, Diffusion maps, Laplacian eigenmaps, LLE, Hessian LLE, LTSA.
- Методы, оптимизирующие невыпуклую целевую функцию с локальными минимумами: Sammon mapping, LLC, Manifold charting,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

PCA

Principal component analysis

- 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!
- Метод главных компонент (Principal component analysis, PCA) – метод снижения размерности, основанный на построении набора из первых k таких ортогональных векторов.

РСА: визуализация



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

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

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

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

- Теорема Пифагора позволяет преобразовать целевую функцию

$$\begin{aligned} \|x_i \perp v\|_2^2 + \langle x_i, v \rangle^2 &= \|x_i\|_2^2 \implies \\ \implies \frac{1}{m} \sum_{i=1}^m \langle x_i, v \rangle^2 &= \frac{1}{m} (Xv)^T (Xv) = \frac{1}{m} v^T X^T X v = \\ &= \frac{1}{m} v^T A v \longrightarrow \max_{\|v\|_2=1} \quad (1) \end{aligned}$$

- Симметричная матрица $\frac{1}{m} X^T X$ есть ничто иное как эмпирическая ковариационная матрица. Важное свойство: собственные числа такой матрицы неотрицательны.
- Целевая функция для случай $d > 1$, проекция x_i на векторное подпространство $V = \{v_1, \dots, v_d\}$

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

- Теперь рассмотрим разложение матрицы A

$$A = VDV^T$$

где матрица V есть ортогональная матрица, а D – диагональная.

- Заметим, что если матрица $A = \text{diag}(\lambda_1, \dots, \lambda_n)$ (пусть еще $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$), то максимум для (1) достигается для $v = e_1$

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

- Отсюда следует, что для произвольной матрицы A , положив $v_1 = V e_1$, получим

$$v_1^T A v_1 = v_1^T V D V^T v_1 = e_1^T V^T V D V^T V e_1 = e_1^T D e_1 = \lambda_1$$

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

$$A v_i = A V e_i = V D V^T V e_i = V D e_i = \lambda_i V e_i = \lambda_i v_i$$

- 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.

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

PCA: Сингулярное разложение (SVD)

- Сингулярное разложение

$$X = USV^T$$

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

- Столбцы матриц U и V называются левыми и правыми сингулярными векторами матрицы X соответственно.

PCA: Сингулярное разложение (SVD)

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

$$\begin{aligned} XX^T &= USV^T V S^T U^T = USS^T U^T = UD_1 U^T \Rightarrow \\ &\Rightarrow XX^T U = UD \end{aligned}$$

$$\begin{aligned} X^T X &= VS^T U^T U S V^T = VS^T S V = VD_2 V^T \Rightarrow \\ &\Rightarrow X^T X V = VD \end{aligned}$$

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

PCA: Итерационный алгоритм

Algorithm 1: Итерационный алгоритм поиска главной компоненты

input : Матрица $A = X^T X$

Выбираем произвольный нормированный вектор u_0 ;

for $i = 1, 2, \dots$ **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):
$$A^{i+1} = A^i A = V D^i V^T V D V^T = V D^{i+1} V^T$$
- Как только нашли первую компоненту, проектируем данные ортогонально найденной главной компоненте, т.е. полагаем $x_i := x_i - \langle x_i, v_1 \rangle v_1$ и запускаем итерационный алгоритм заново.

PCA: обсуждение итерационного алгоритма

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

- $\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T$
- $\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{D}$
- Учитывая то, что по столбцам у матрицы \mathbf{V} стоят собственные векторы, формирующие ортогональный базис, то любой вектор в этом базисе может быть расписан как $\mathbf{u}_0 = c_1\mathbf{v}_1 + \dots + c_n\mathbf{v}_n$.
- Тогда получаем

$$\begin{aligned}\mathbf{A}^i\mathbf{u}_0 &= c_1\mathbf{A}^i\mathbf{v}_1 + \dots + c_n\mathbf{A}^i\mathbf{v}_n = c_1\lambda_1^d\mathbf{v}_1 + \dots + c_n\lambda_n^d\mathbf{v}_n = \\ &= \lambda_1^d(c_1\mathbf{v}_1 + c_2\frac{\lambda_2^d}{\lambda_1^d}\mathbf{v}_2 + \dots + c_n\frac{\lambda_n^d}{\lambda_1^d}\mathbf{v}_n)\end{aligned}$$

- Откуда следует, что при $i \rightarrow \infty$, учитывая $\lambda_1 \geq \dots \geq \lambda_n$,
 $\mathbf{A}^i\mathbf{u}_0 \rightarrow C(i)\mathbf{v}_1$

- 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:

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

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

- 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 $X^T X$, а ядровой матрицы $K : k_{ij} = \kappa(x_i, x_j)$.
- Самое главное: $Y = XV$. Но мы можем взять лишь первые d компонент для аппроксимации: $Y_d = XV_d$.

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 $D_{[m \times m]} = \{d_{ij}\}_{i,j=1}^n$, хотим найти такие $\mathbf{y}_j \in R^d, d < n, j = 1, 2, \dots, m$, чтобы

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

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

$$\sum_{i=1}^m y_{ip} = 0, \quad p = 1, \dots, d \quad (2)$$

Классический MDS: общая теория

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

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

- Кроме того из (2)

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

- Из (3)

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

- Наконец, из (3), (4) имеем

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

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

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

$$\text{Stress}(\mathbf{Y}) = \sigma(\mathbf{Y}) = \sum_{i < j \leq m} w_{ij} \left(\delta_{ij} - \hat{d}_{ij}(\mathbf{Y}) \right)^2 \longrightarrow \min_{\mathbf{Y}} \quad (5)$$

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:

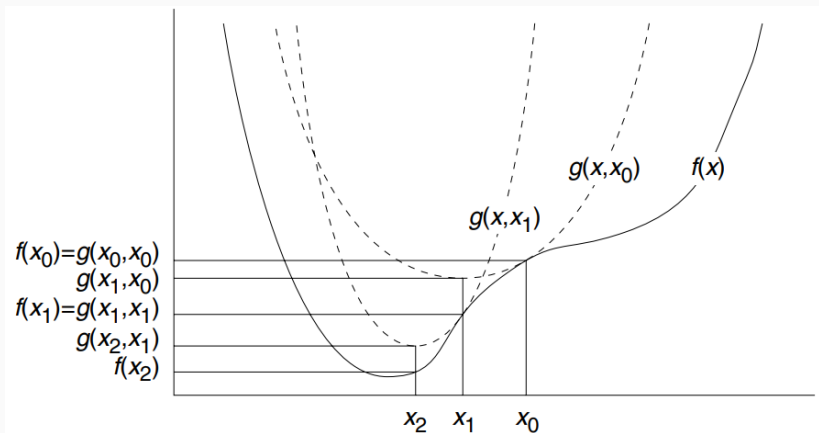
- Очевидно, функция $g(x, z)$ должна быть проще, чем $f(x)$. Квадратичной или даже линейной.
- $f(x) \leq g(x, z)$.
- Вспомогательная функция должна касаться функции в так называемом саппорте (supporting point): $f(z) = g(z, z)$.

Из этих правил следует

$$f(\hat{x}) \leq g(\hat{x}, z) \leq g(z, z) = f(z)$$

где \hat{x} – минимум $g(x, z)$.

Метрический MDS: визуализация итеративной мажоризации [3]



- Функция Stress может быть ограничена сверху

$$\begin{aligned}\sigma(\mathbf{Y}) &= \sum_{i < j} w_{ij} \delta_{ij}^2 + \sum_{i < j} w_{ij} \hat{d}_{ij}^2(Y) - 2 \sum_{i < j} w_{ij} \delta_{ij} \hat{d}_{ij}(Y) = \\ &= \text{Const} + \text{tr} [\mathbf{Y}^T \widehat{\mathbf{W}} \mathbf{Y}] - 2 \text{tr} [\mathbf{Y}^T B(\mathbf{Y}) \mathbf{Y}] \leqslant \\ &\leqslant \text{Const} + \text{tr} [\mathbf{Y}^T \widehat{\mathbf{W}} \mathbf{Y}] - 2 \text{tr} [\mathbf{Y}^T B(\mathbf{Z}) \mathbf{Z}] = \tau(\mathbf{Y}, \mathbf{Z}) \quad (6)\end{aligned}$$

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

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

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

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

Algorithm 2: SMACOF

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

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

$Z = Y_{i-1}$;

$Y_i = \arg \min_Y \tau(Y, Z)$ (6);

if $\sigma(Y_{i-1}) - \sigma(Y_i) < \varepsilon$ **then**

 Возвращаем Y_i ;

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

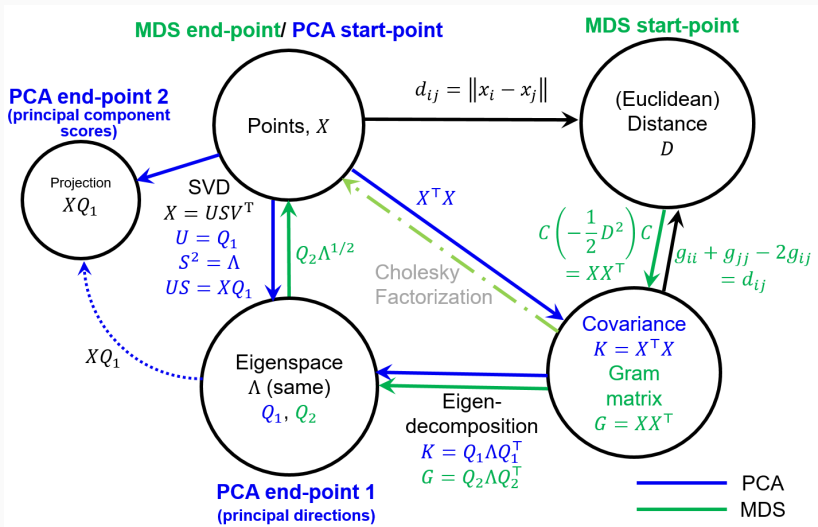
- Чаще всего используется все же именно для визуализации.
- Устоявшийся и все еще актуальный алгоритм (набор алгоритмов) с долгой историей.
- Все те же проблемы с существенной нелинейностью многообразия, как и у PCA.

Классический MDS эквивалентен PCA

Кстати, классический MDS эквивалентен PCA.

- Пусть $X^T X = Q_1 \Lambda_1 Q_1^T$ and $XX^T = Q_2 \Lambda_2 Q_2^T$.
- Пусть также $X = USV^T$ и, помня про ортогональность U и V , получаем $X^T X = VS^T SV^T$. Но при этом $XX^T = USS^T U^T$. Таким образом, $Q_1 = V$, $Q_2 = U$, и т.к. S диагональна, то $\Lambda_{1[n \times n]} \equiv \Lambda_{2[m \times m]}$.
- $\implies X_{[m \times n]} V_{[n \times n]} = U_{[m \times m]} S_{[m \times n]}$.

Классический MDS эквивалентен PCA [4]

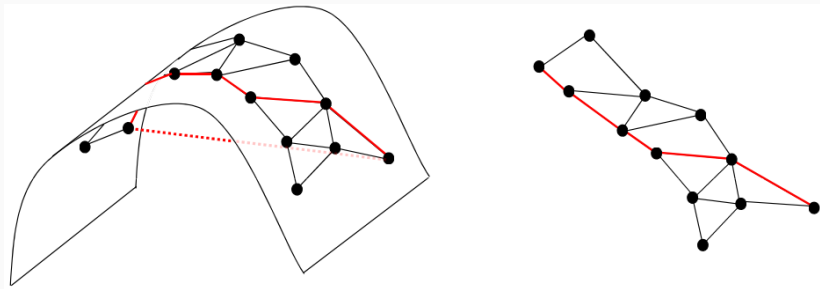


ISOMAP, LLE и Laplacian Eigenmaps

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 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 подаем на вход в классический MDS.

ISOMAP: визуализация [6]



- 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).

- Используя граф соседей, построим матрицу W , такую что

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

причем, $w_{ij} = 0$, если \mathbf{x}_i и \mathbf{x}_j не являются соседями, а также $\sum_j w_{ij} = 1$.

- Веса, удовлетворяющие ограничениям, обладают важным свойством: при поворотах, шкалировании или сдвигах осей они остаются неизменными.

- Для нахождения весов применяются следующие соображения

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

- Ну а дальше задача условной оптимизации методом множителей Лагранжа.

- Найдя веса, рассматриваем аналогичную оптимизационную задачу

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

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

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

- А также ограничение на равенство ковариационной матрицы единичной матрице

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

- Отметим, что целевая функция является функцией квадратичной

$$\begin{aligned}\Phi(Y) &= \sum_{i=1}^m \|Y^T \mathbf{1}_i - Y^T \mathbf{w}_i\|_2^2 = \|Y^T (E - W)^T\|_F^2 = \\ &= \text{tr}(Y^T (E - W)^T (E - W) Y) = \text{tr}(Y^T M Y) \quad (7)\end{aligned}$$

- 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).

- 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 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 -ближайшими соседями.
- С точки зрения практики алгоритм прост: строим лапласиан графа, находим его собственные векторы, выбираем первые d .

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

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

t-SNE

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

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

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

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2m} \quad (9)$$

- Из (9) следует, что

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

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

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

- Среднеквадратическое отклонение подбирается на основе гиперпараметра, называемого перплексией

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

где $H(P_i)$ есть энтропия Шеннона 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 эмбедингов $\mathbf{y}_i, i = 1, \dots, m$ происходит на основе оптимизации КЛ-дивергенции между распределениями P и Q , являющимися совместными распределениями на все расстояния между парами наблюдений

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

- Градиент дивергенции найти не сложно

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

- t-распределение позволяет улучшить ситуацию 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.

- 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).

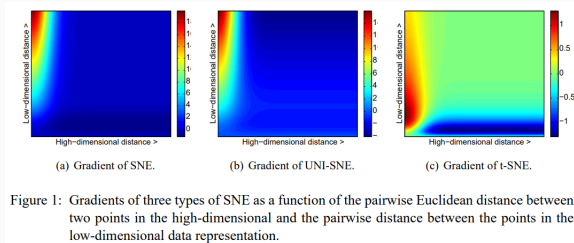
t-SNE: замечания об оригинальном SNE i

- 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, большой вес для представления объектов, что находятся далеко друг от друга в пространстве эмбедингов, но близко в исходном пространстве, чем когда объекты в пространстве эмбедингов находятся близко, а в исходном пространстве далеко друг от друга.
- Следовательно, сохраняется только локальная структура данных (как в LLE).

- 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 Стьюдента i

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



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

Что до оптимизации, то

- инициализируют 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

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

Algorithm 3: Классический t-SNE

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

Вычисляем p_{ij} по формуле (9);

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

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

 Вычисляем q_{ij} по формуле (10);

 Вычисляем градиент $\frac{\partial C}{\partial \mathbf{y}_i}$ по формуле (11);

 Обновляем значения \mathbf{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 exaggeration». 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 эмбедингов 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 эмбедингов, 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 $O(n \log n)$.

- Распишем градиент (11) следующим образом

$$\begin{aligned}\frac{\partial C}{\partial \mathbf{y}_i} &= 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1} = \\ &= 4 \sum_j (p_{ij} - q_{ij})q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j) = \\ &= 4 \left(\sum_j p_{ij}q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_j q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j) \right) \quad (13)\end{aligned}$$

где $Z = \sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}$.

- Первое слагаемое считается за $O(un)$ с учетом первой идеи по аппроксимации P и ввиду того, что $q_{ij}Z = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$.
- Второе слагаемое может быть аппроксимировано за $O(n \log n)$ путем построения дерева квадрантов (quadtree; для двумерного эмбединга). Для трехмерного случая можно использовать октодерево (octtree).

t-SNE: алгоритм Барнеса–Хата

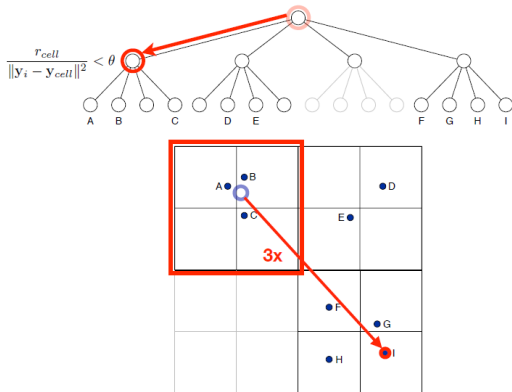


Figure 2: Illustration of the Barnes-Hut approximation. To evaluate the t-SNE gradient for point I, the Barnes-Hut algorithm performs a depth-first search on the embedding quadtree, checking at every node whether or not the node may be used as a “summary”. In the illustration, the cell containing points 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 аппроксимировать второе слагаемое следующим образом

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

где \mathbf{y}_{cell} – центр масс точек в квадранте.

- Небольшой хак: находим аппроксимацию для значения $-q_{ij}^2 Z^2(\mathbf{y}_i - \mathbf{y}_j)$ 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

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

где r_{cell} – длина диагонали квадранта.

- θ – это параметр `angle` в `sklearn`.

- 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).

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`'е) снижает это число до $O(un \log n)$, но ограничивает применение для эмбедингов размерности 2 или 3.

Обсуждение

- 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.

- 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 эмбединг в пространстве меньшей размерности.

- 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.

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>.

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.

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. T. 15, № 93. C. 3221—3245. URL: <http://jmlr.org/papers/v15/vandermaaten14a.html>.