Занятия 2 и 3. Снижение размерности

Анализ данных и машинное обучение

Гирдюк Дмитрий Викторович

29 мая 2022 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 (отдельными) 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 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 процессинг 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

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)
- Maximum Variance Unfolding (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)
- 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
  - 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, LarveVis, 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 $k = 1$

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

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. Важное свойство: собственные числа такой матрицы неотрицательны
- Целевая функция для случай $k > 1$, проекция x_i на векторное подпространство $V = \{v_1, \ldots, v_k\}$

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

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 $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n \geqslant 0$), то максимум для (1) достигается для $v = e_1$

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

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

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

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):
  $A^{i+1} = A^i A = V D^i V^T V D V^T = V D^{i+1} V^T$
- 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 алгоритма

Почему последовательное домножение на вектор u_0 матрицы A приводит нас в конечном итоге к главной компоненте?

- $A = QDQ^T$
- $AQ = QD$
- Учитывая то, что по столбцам у матрицы Q стоят собственные векторы, формирующие ортогональный базис, то любой вектор в этом базисе может быть расписан как $u_0 = c_1 v_1 + \ldots + c_n v_n$
- Тогда получаем

$$A^k u_0 = c_1 A^k v_1 + \ldots + c_n A^k v_n = c_1 \lambda_1^k v_1 + \ldots + c_n \lambda_n^k v_n = \\ = \lambda_1^k (c_1 v_1 + c_2 \frac{\lambda_2^k}{\lambda_1^k} v_2 + \ldots + c_n \frac{\lambda_n^k}{\lambda_1^k} v_n)$$

- Откуда следует, что при $k \to \infty$, учитывая $\lambda_1 \geqslant \ldots \geqslant \lambda_n$, $A^k u_0 \to v_1$

PCA: обсуждение i

- Простая интерпретация PCA: 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 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

PCA: обсуждение ii

- 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 $y_j \in R^k, k < n, j = 1, 2, \ldots, m$, 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

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

- 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)

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

- 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 низкоуровневого представления 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

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

Metric Multidimensional Scaling: SMACOF

- Ее ограничивают сверху

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

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 алгоритм (набор алгоритмов) 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 PCA, который, как уже было отмечено, эквивалентен classical 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

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

- 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

t-SNE

- t-distributed Stohastic Neighbour Embedding (t-SNE) [6] – нелинейный алгоритм снижения размерности, разработанный для визуализации в (обычно) двух/трехмерном пространстве эмбеддингов, основанный 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

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

$$p_{j|i} = \frac{\exp(-d(x_i, x_j)^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-d(x_i, x_k)^2 / 2\sigma_i^2)}, \quad p_{i|i} = 0 \tag{8}$$

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

- 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

t-SNE: общая теория

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

$$q_{ij} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||y_k - y_l||^2\right)^{-1}}, \quad q_{i|i} = 0. \tag{10}$$

- 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
- Подробнее о том, почему было взято t-распределение (конечно, за его длинные широкие хвосты), читайте в статье [6]. В изначальном SNE вместо распределения Стьюдента использовалось нормальное распределение с $\sigma_i = 1$.

t-SNE: общая теория

- Вычисление эмбеддингов $y_i, i = 1, \ldots, m$ происходит на основе оптимизации КЛ-дивергенции между распределениями P и Q, являющимися совместными распределениями на все расстояния между парами наблюдений

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

- Градиент дивергенции найти не сложно

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_k - y_l||^2)^{-1} \qquad (11)$$

t-SNE: общая теория

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

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

t-SNE: алгоритм

Algorithm 3: Классический t-SNE

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

Вычисляем p_{ij} по формуле (9);

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

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

- Вычисляем q_{ij} по формуле (10);
- Вычисляем градиент $\frac{\partial C}{\partial y_i}$ по формуле (11);
- Обновляем значения Y^i 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 [7]

- 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 [8] 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

$$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_k - y_l||^2)^{-1} =$$

$$= 4\sum_j (p_{ij} - q_{ij})q_{ij}Z(y_i - y_j) =$$

$$= 4\left(\sum_j p_{ij}q_{ij}Z(y_i - y_j) - \sum_j q_{ij}^2 Z(y_i - y_j)\right) \quad (13)$$

  где $Z = \sum_{k \neq l} \left(1 + ||y_k - y_l||^2\right)^{-1}$
- Первое слагаемое считается за $O(un)$ с учетом первой идеи по аппроксимации P и ввиду того, что $q_{ij}Z = (1 + ||y_i - y_j||^2)^{-1}$
- 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

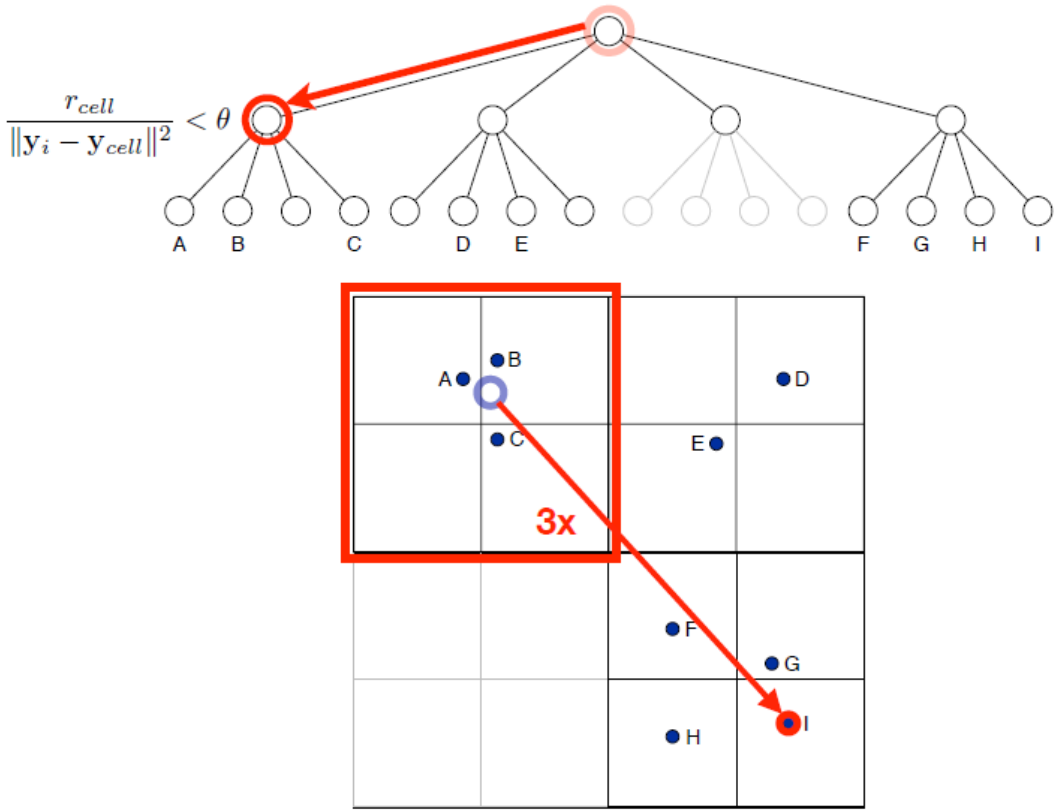

Figure 2: Illustration of the Barnes-Hut approximation. To evaluate the t-SNE gradient for point I, the Barnes-Hut algorithm performs a depth-first search on the embedding quadtree, checking at every node whether or not the node may be used as a "summary". In the illustration, the cell containing points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

$$-q_{ij}^2 Z(y_i - y_j) \approx -n_{cell} q_{i,cell} Z(y_i - y_{cell}),$$

где n_{cell} – числоточек в квадранте в квадранте, y_{cell} – центр масс точек в квадранте

- Небольшой хак: находим аппроксимацию для значения $-q_{ij}^2 Z^2(y_i - y_j)$ 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

$$\frac{r_{cell}}{||y_i - y_{cell}||^2} < \theta,$$

где r_{cell} – длина диагонали квадранта

- θ – тот самый параметр angle в sklearn'е

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 [7]
- 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

UMAP

- Uniform Manifold Approximation and Projection, UMAP [9] – относительно недавно появившийся (2018 г.) нелинейный алгоритм снижения размерности
- Разработан на основе результатов из римановой геометрии, алгебраической топологии, элементов нечеткой топологии
- Хотя для общего понимания алгоритма глубокое понимание не обязательно: метод относится к классу алгоритмов, строящих граф k-ближайших соседей, на манер Laplacian eigenmaps, ISOMAP или t-SNE

UMAP: предположения

Что точно стоит упомянуть из теории, так это основные предположения о данных

- Существует многообразие, на котором наблюдения распределены равномерно
- Данное многообразие является локально связанным пространством
- Основная цель алгоритма состоит в сохранении топологической структуры многообразия

UMAP: основная теория

Первый этап – построение взвешенного графа по наблюдениям

- Находим k-ближайших соседей для каждого $x_i \in X$
- Для каждого x_i находим ρ_i и σ_i

$$\rho_i = \min\left\{d(x_i, x_{i_j}) | 1 \leqslant j \leqslant k\right\}$$

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2 k$$

UMAP: основная теория

- Что есть p_i? Расстояние до ближайшего сосед. Для чего нужно? Чтобы точно иметь хотя бы одно ребро в графе с весом 1. А это, в свою очередь, позволяет быть уверенным в том, что общее нечеткое симплициальное множество (fuzzy simplical set, которое мы зададим графом) локально связано в точке x_i
- Для чего нужно σ_i? Для задания собственной Римановой метрики вблизи x_i (хак для приближения к выполнимости первой аксиомы о равномерности распределения данных на многообразии). Находится бинарным поиском

UMAP: основная теория

Строим сам граф $G = (V, E, w)$

- Вершины $V = X$
- Ребра E – связи между k-ближайшими соседями
- Весовая функция w определяются следующим образом

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

- Ну а дальше строим симметричную матрицу смежности B на основании несимметричной матрицы смежности A, состоящей из элементов $w((x_i, x_{i_j}))$

$$B = A + A^T - A \odot A^T,$$

где операция $\odot$ – покомпонентное произведение матриц

UMAP: основная теория

- Элементы матрицы A, т.е. веса $w((x_i, x_{i_j}))$, можно понимать как вероятности существования ребра между наблюдениями. Очевидно, что они несимметричны
- Тогда элементы симметричной матрицы смежности B можно понимать как вероятность того, что хотя бы одно из ребер между x_i и x_j существует
- По итогу имеем взвешенный ненаправленный граф
- Что до теоретических оснований, почему мы все это объединяем в таком ключе (различные локальные топологичские структуры с вообще говоря различными метриками) – придется разбираться с теорией нечетких симплициальных множеств [9] (удачи)

UMAP: основная теория

Второй этап – построение эмбеддинга

- Строго говоря, построенный граф G есть представление нечеткого множества A с функцией принадлежности $\mu(a), a \in A$
- Хотим получить граф в пространстве меньшей размерности, соответствующей нечеткому множеству A с функцией принадлежности $\nu(a), a \in A$
- Кросс-энтропия для нечетких множеств

$$
\begin{aligned}
C((A,\mu),(A,\nu)) = \\
\sum_{a\in A} \mu(a)\log\left(\frac{\mu(a)}{\nu(a)}\right) + (1-\mu(a))\log\left(\frac{1-\mu(a)}{1-\nu(a)}\right) = \\
\sum_{a\in A} (\mu(a)\log(\mu(a)) + (1-\mu(a))\log(1-\mu(a))) - \\
- \sum_{a\in A} (\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a)))
\end{aligned} \tag{14}
$$

UMAP:основная теория

- Ввиду того, что первая часть суммы зависит только от известной $\mu(a)$, ее можно отбросить, т.е. оптимизировать лишь

$$-\sum_{a \in A} \left(\mu(a) \log\left(\nu(a)\right) + \left(1 - \mu(a)\right) \log\left(1 - \nu(a)\right)\right)$$

- Для оптимизации предлагается использовать вариацию стохастического градиентного спуска, используя сэмплирование ребер из графов для обоих слагаемых под суммой (в случае второго слагаемого, будет использоваться так называемый negative sampling)

UMAP:основная теория

- Для этого надо определиться, что делать с $\nu(a)$. А именно найти дифференцируемую аппроксимацию этой функции. Авторы делают это на манер t-SNE

$$\Phi(y_i, y_j) = \left(1 + a\left(||y_i - y_j||_2^2\right)^b\right)^{-1}$$

- Причем параметры подбирают на основе МНК, подгоняясь под

$$\Psi(y_i, y_j) = \begin{cases} 1, & ||y_i - y_j||_2 \leqslant mindist \\ \exp(-(||y_i - y_j||_2 - mindist)), & \text{иначе} \end{cases}$$

UMAP: псевдокод всего алгоритма

Algorithm 1 UMAP algorithm

function UMAP(X, n, d, min-dist, n-epochs)

 # Construct the relevant weighted graph

for all $x \in X$ **do**

fs-set$[x] \leftarrow$ LocalFuzzySimplicialSet(X, x, n)

top-rep $\leftarrow \bigcup_{x \in X}$ fs-set$[x]$ *# We recommend the probabilistic t-conorm*

 # Perform optimization of the graph layout

$Y \leftarrow$ SpectralEmbedding(top-rep, d)

$Y \leftarrow$ OptimizeEmbedding(top-rep, Y, min-dist, n-epochs)

return Y

UMAP: псевдокод алгоритма построения графа

Algorithm 2 Constructing a local fuzzy simplicial set

function LocalFuzzySimplicialSet(X, x, n)

knn, knn-dists $\leftarrow$ ApproxNearestNeighbors(X, x, n)

$\rho \leftarrow$ knn-dists[1] # *Distance to nearest neighbor*

$\sigma \leftarrow$ SmoothKNNDist(knn-dists, n, ρ) # *Smooth approximator to knn-distance*

fs-set$_0 \leftarrow X$

fs-set$_1 \leftarrow \{([x, y], 0) \mid y \in X\}$

for all $y \in$ knn **do**

$d_{x,y} \leftarrow \max\{0, \text{dist}(x, y) - \rho\}/\sigma$

fs-set$_1 \leftarrow$ fs-set$_1 \cup ([x, y], \exp(-d_{x,y}))$

return fs-set

UMAP: псевдокод алгоритмов инициализации параметров σ_i и спэктрального эмбеддинга

Algorithm 3 Compute the normalizing factor for distances σ

function SmoothKNNDist(knn-dists, n, ρ)

Binary search for σ such that $\sum_{i=1}^{n} \exp(-(\text{knn-dists}_i - \rho)/\sigma) = \log_2(n)$

return σ

Algorithm 4 Spectral embedding for initialization

function SpectralEmbedding(top-rep, d)

$A \leftarrow$ 1-skeleton of top-rep expressed as a weighted adjacency matrix

$D \leftarrow$ degree matrix for the graph A

$L \leftarrow D^{1/2}(D - A)D^{1/2}$

evec $\leftarrow$ Eigenvectors of L (sorted)

$Y \leftarrow \text{evec}[1..d+1]$ *# 0-base indexing assumed*

return Y

UMAP: псевдокод алгоритма стохастической оптимизации

Algorithm 5 Optimizing the embedding

function OPTIMIZEEMBEDDING(top-rep, Y, min-dist, n-epochs)

$\alpha \leftarrow 1.0$

Fit Φ from Ψ defined by min-dist

for $e \leftarrow 1, \ldots,$ n-epochs **do**

for all $([a, b], p) \in$ top-rep$_1$ **do**

if RANDOM( ) $\leq p$ **then** *# Sample simplex with probability p*

$y_a \leftarrow y_a + \alpha \cdot \nabla(\log(\Phi))(y_a, y_b)$

for $i \leftarrow 1, \ldots,$ n-neg-samples **do**

$c \leftarrow$ random sample from Y

$y_a \leftarrow y_a + \alpha \cdot \nabla(\log(1 - \Phi))(y_a, y_c)$

$\alpha \leftarrow 1.0 - e/$n-epochs

return Y

UMAP: гиперпараметры

- Число компонент k
- Число соседей k_{nn}. Чем меньше значение, тем точнее передается локальная топология многообразия. Чем выше, тем точнее передается глобальная структура
- $mindist$ – задает уровень разделения между точками в пространстве эмбеддингов. Чем меньше, тем более плотно упакованными будут локальные области с эмбеддингами. Чем больше, тем точки сильнее будут разбросаны по пространству меньшей размерности

UMAP: обсуждение

Из плюсов

- Серьезная теория под капотом
- Вычислительно эффективный ($O(N^{1.14})$)
- Умеет в произвольный размер эмбеддингов

UMAP: обсуждение

Основные недостатки

- Все те же проблемы с интерпретацией
- И нахождением структур в шуме (одна из аксиом гласит о существовании в данных многообразия)
- Ориентируется на локалььную структуру, потому в случае, когда нужно точнее передать общую структуру данных, рекомендуется применять другой алгоритм
- Ну и разного рода аппроксимации (поиск соседей, аппроксимация функции принадлежности в пространстве меньшей размерности) могут портить общую картину работы алгоритма. Особенно в случае небольшой выборки

Снижение размерности: обсуждение

Обсуждение

- Классификация алгоритмов разнится. Кроме указанной в начале презентации (из работы van der Maaten'a [2]), 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, UMAP).

Обсуждение

Немного о взаимосвязях

- 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 и UMAP), 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

- 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
- UMAP на бумаге выглядит очень многообещающе. И хотя есть замечательно написанная статья, а также подробная документация, разобраться в репе с кодом достаточно сложно. Ну и результаты экспериментов на своих собственных датасетах весьма спорные

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.
6. *van der Maaten L.*, *Hinton G.* Visualizing High-Dimensional Data Using t-SNE. // Journal of Machine Learning Research. 2008. Т. 9. С. 2579—2605.
7. *Wattenberg M.*, *Viegas F.*, *Johnson I.* How to use t-SNE Effectively. URL: https://distill.pub/2016/misread-tsne/.
8. *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.
9. *McInnes L.*, *Healy J.* UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. // ArXiv. 2018. Т. abs/1802.03426.