Сравнение методов визуализации данных

Карпова Анастасия (БЭК 165)

Омелюсик Владимир (БЭК 161)

14 мая 2020 г.

Содержание

Методы снижения размерности данных разрабатывались с тридцатых годов прошлого века. Первые алгоритмы, такие как PCA (Hotelling, 1933) и классический вариант MDS (Torgerson, 1952), были линейными. В их основу была положена идея, что точки, значительно удалённые друг от друга в исходном пространстве, должны оставаться на большом расстоянии в новом пространстве меньшей размерности [6]. Таким образом, линейные методы сохраняли глобальные особенности структуры данных.

Тем не менее, на практике исходное пространство далеко не всегда является линейным, а может представлять собой многообразие сложной формы. В этой связи часто возникает ситуация, когда важно сохранить, скорее, не глобальные, а локальные свойства данных. Например, в биологии молекулы могут кодироваться высокоразмерными векторами, отражающими их свойства, и при снижении размерности важно, чтобы близколежащие молекулы оставались локально различимыми в новом пространстве [9].

Так как линейные методы не способны справиться с этой задачей, в последние годы активно развиваются нелинейные методы снижения размерности.

Одной из задач, которую можно решить при помощи понижения размерности, является визуализация многомерных данных. Её суть заключается в проецировании исходных данных на пространство размерности не выше трёх и последующее изображение векторов в этом пространстве. Тем не менее, у задачи визуализации многомерных данных есть некоторые специфические особенности. Во-первых, качественная визуализация должна быть понятной и доносить до наблюдателя релевантную (чтобы это не значило) информацию об исходном пространстве. Во-вторых, не очень понятно, каким образом оценивать и измерять качество визуализации. Критерии «хорошей картинки» могут варироваться от задачи к задаче.

В этом обзоре будут рассмотрены некоторые методы понижения размерности и их применение для визуализации данных. Кроме того, будут приведены примеры метрик, используемых для сравнения их качества. В связи с популярностью нелинейных методов, большая часть обзора посвящена им. Также в работе приведен пример снижения размерности с помощью MDS — одного из наиболее популярных линейных методов после всем известного метода главных компонент (PCA).

1 Описание методов визуализации

1.1. MDS

Методы многомерного шкалирования (multidimensional scaling) для снижения размерности данных разрабатывались с пятидесятых годов прошлого века. В этом разделе будет представлен краткий обзор этих методов, а более полную информацию о них можно найти, например, в статье Buja et al. (2008) [2].

Основная идея всех методов MDS заключается в том, чтобы отобразить точки из исходного признакового пространства в пространство меньшей размерности таким образом, чтобы попарное расстояние между точками в новом пространстве было как можно более близко к расстоянию между ними в исходном пространстве. Формально, расстояние в исходном пространстве задано «матрицей расхождений» (matrix of dissimilarities) $D = \{d_{i,j}\}_{i,j=1}^{N}$, а цель алгоритма заключается в поиске векторов $x_1, \dots x_N \in \mathbb{R}^k$, таких что $d_{i,j}$ хорошо приближается значением $||x_i - x_j||$. Для целей визуализации k выбирается не больше трёх. В классической постановке векторы подбираются таким образом, чтобы минимизировать функцию стресса (stress function) [4].

$$\text{Stress}_D(x_1, \dots x_N) = \left(\sum_{i \neq j = 1 \dots N} (d_{i,j} - ||x_i - x_j||)^2 \right)^{1/2}.$$

Оптимизация происходит применением градиентного спуска в пространстве новых признаков. Матрица D обычно предполагается симметричной.

Методы MDS различаются, в основном, спецификацией функции потерь. Выделяются два ортогональных вектора различий:

1. Классическое шкалирование Торгерсона-Гоуэра, основанное на скалярных произведениях, против шкалирования Краскала-Шепарда, основанного на расстояниях. Первый вариант подразумевает преобразование элементов матрицы D в элементы матрицы B таким образом, чтобы для каждого элемента матрицы выполнялось равенство $d_{i,j}^2 = b_{ii} - 2b_{i,j} + b_{jj}$. Идея преобразования состоит в переходе от нормы к скалярным произведениям. Во втором варианте используются непосредственно $||x_i - x_j||$. Важное различие между методами заключается в том, что скалярные произведения зависят от выбора начала координат, а расстояния — нет. Чтобы уменьшить уровень произвольности обычно в классическом шкалировании используется центрирование по среднему.

2. Метрическое шкалирование против неметрического. В первом случае используется сами элементы матрицы расхожести, а во втором — только их ранги.

В случае классического шкалирования в качестве функции потерь используется функция напряжения [2]:

$$\text{Strain}_D(x_1, \dots x_N) = \left(\frac{\sum_{i,j} [b_{i,j} - \langle x_i, x_j \rangle]^2}{\sum_{i,j} b_{i,j}^2} \right)^{1/2}.$$

Следует отметить, что для финальной оптимизации функции стресса и напряжения модифицируются для повышения скорости работы, а потому приобретают достаточно сложный вид. Это можно увидеть в статье Buja et al. (2008).

Сложность методов MDS по времени составляет $O(N^2)$, а потому не рекомендуется использовать их для больших наборов данных (ориентироваться стоит на несколько тысяч наблюдений). Тем не менее, ввиду достаточно простой реализации и свободы в определении расхождений между объектами, MDS используются для визуализации данных во многих отраслях. Например, в социальных науках по данным опросов можно определить близость между респондентами на основе их предпочтений, а в археологии — близость двух мест раскопки на основании найденных артефактов. MDS также применяется для визуализации структур молекул, что является особой задачей, так как в этом случае расстояние между молекулами можно непосредственно измерить или рассчитать, а единственная разумная размерность нового пространства равна $k = 3$.

Вычислительная сложность: $O(N^2)$.

1.2. Isomap

Isomap (isometric feature mapping) — метод нелинейного снижения размерности, основанный на графовых методах. Он является расширением методов снижения размерности в линейных пространствах (PCA, MDS) до нелинейных многообразий. Метод был представлен в статье Tenenbaum et al. (2000) [12]. Алгоритм состоит из трех шагов:

1. На первом шаге строится граф окрестностей. Для каждой точки в многообразии находятся ее соседи. В качестве метода поиска соседей обычно используют k ближайших соседей (k-Isomap) либо отбирают все точки, находящиеся в пределах заданного радиуса ε (ε-Isomap). Затем все окрестно-

сти изображаются в виде взвешенного графа G, в котором вес ребра равен расстоянию между соседними точками $d_X(i,j)$. В качестве расстояние $d_X(i,j)$ можно использовать, например, расстояние Евклида.

2. На втором шаге между всеми парами точек в на многообразии подсчитывается геодезическое расстояние $d_M(i,j)$, то есть оценивается длина кратчайшего пути $d_G(i,j)$ в графе G. Сначала предполагают, что $d_G(i,j) = d_X(i,j)$, если вершины i, j соединены ребром, и $d_G(i,j) = \infty$ иначе. Затем для каждой вершины $k = 1, 2, \ldots, N$ значения $d_G(i,j)$ заменятся на $\min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$. Итоговая матрица $D_G = \{d_G(i,j)\}_{i,j}^{1,\ldots N}$ содержит длины всех кратчайших маршрутов между всеми парами точек в G. Использование геодезического расстояния вместо привычного евклидова позволяет учесть геометрию исходного многообразия.

3. На третьем шаге классический вариант MDS применяется к матрице расстояний $D_G = \{d_G(i,j)\}_{i,j}^{1,\ldots N}$, в результате чего получается представление исходных данных в d-мерном евклидовом пространстве. Координаты y_i в новом пространстве получается путем минимизации следующей функции:

$$E = ||\tau(D_G) - \tau(D_Y)||_{L^2},$$

где D_Y обозначает матрицу расстояний Евклида $\{d_Y(i,j) = ||y_i - y_j||\}$. Запись $||A||_{L^2}$ — норма матрицы $\sqrt{\sum\limits_{i,j} A_{i,j}^2}$. Функция τ переводит расстояния в скалярное произведение, которое все также является уникальной характеристикой геометрии данных, но в форме, более удобной для оптимизации.

Настраиваемые гиперпараметры:

- k-Isomap: k — число ближайших соседей;
- ε-Isomap: 'ε' — радиус.

Вычислительная сложность: $O(N^3)$.

1.3. LLE

Locally Linear Embedding (LLE) — это метод машинного обучения без учителя, предназначенный для нелинейного снижения размерности. LLE был предложен в статье Sam T. Roweis и Lawrence K. Saul (2000) [11]. Алгоритм способен находить глобальные нелинейные структуры в данных, сохраняя при этом локальные расстояния между точками.

В основе метода лежит несложная геометрическая интуиция. Допустим, имеется набор данных, состоящий из N штук D-мерных векторов x_i . Требуется перейти в d-мерное пространство меньшей размерности ($d << D$) и при этом сохранить как можно больше геометрических структур, имеющихся в многообразии, откуда семплированы исходные данные.

Алгорит состоит из трех шагов:

1. Поиск k ближайших соседей для каждой точки. Несмотря на то, что исходное многообразие может быть очень сложным и искривленным, пусть верна предпосылка, что каждая точка и ее соседи в некоторой окрестности лежат в одной и той же плоскости.

2. Тогда каждую точку можно восстановить с помощью линейной комбинации ее k-ближайших соседей, взвешенных с некоторыми весами. Оптимальный набор весов подбирается путем минимизации следующей функции потерь:
$$\varepsilon(W) = \sum_i |x_i - \sum_{i,j} w_{ij} x_j|^2.$$
Вес w_{ij} можно интерпретировать как вклад j-ой точки в реконструкцию i-ой точки.

   Следует учитывать два ограничения на веса: во-первых, $w_{ij} = 0$ для всех точек j, не являющихся соседями точки i, во-вторых, $\sum_{i,j} w_{ij} = 1$. Получаемые в ходе оптимизации при заданных ограничениях веса имеют важное свойство: они устойчивы к повороту, масштабированию и переносу восстанавливаемой точки и ее соседей. Это значит, что возможно применение этих преобразований и их комбинаций для перехода в пространство новой размерности, сохраняя возможность использовать уже полученные веса для подсчета аппроксимации вектора x_i в новом признаковом пространстве.

3. На следующем шаге с помощью полученных весов подбирается представление вектора x_i в d-мерном пространстве:
$$\Phi(y) = \sum_i |y_i - \sum_j w_{ij} y_j|^2$$

Настраиваемые гиперпараметры:

- k — число ближайших соседей.

Вычислительная сложность: $O(dN^2)$, где d — размерность пространства представления.

1.4. t-SNE

Метод t-SNE был представлен в статье Maaten и Hinton (2008) как более удобня для оптимизации [6], [3] модификация метода SNE из работы Hinton и Roweis (2003). Идея SNE заключается в том, чтобы преобразовать расстояния между точками x_i и x_j в исходном пространстве большой размерности в условные вероятности $p_{j|i}$, которые отражали бы меру сходства между этими точками. Формально, $p_{j|i}$ — это вероятность того, что x_j является соседней точкой к x_i, если соседние точки выбираются пропорционально функции плотности нормального распределения $\mathcal{N}(x_i, \sigma_i)$:
$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}.$$

Так как моделируются попарные сходства, $p_{i|i}$ предполагают равным нулю.

Аналогично меру сходства можно определить и в пространстве меньшей размерности для точек y_i и y_j, соответствующим точкам x_i и x_j в исходном пространстве. Если обозначить соответствующие условные вероятности как $q_{j|i}$ и задать дисперсию нормального распределения $\mathcal{N}(y_i, \sigma_i)$ (в оригинальной статье используется значение $1/\sqrt{2}$), то

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2)}$$

Как и прежде, $q_{i|i} = 0$.

В такой постановке естественной идеей является выбор дивергенции Кульбака-Лейблера в качестве функции потерь:

$$\text{Loss} = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

SNE минимизирует сумму дивергенций по всем наблюдениям при помощи градиентного спуска. В оригинальной статье утверждается, что хотя метод и производит достаточно хорошие визуализации, он подвержен влиянию нескольких проблем, также хорошо описанных здесь. Во-первых, функцию потерь достаточно сложно оптимизировать из-за наличия в градиенте условных вероятностей, зависящих от экспонент, которые могут достаточно быстро взорваться. Во-вторых, наличие так называемой «проблемы столпотворения» (crowding problem), заключающейся в том, что невозможно отобразить равноудалённые точки из пространства высокой размерности (например, 10), в двух- или трёхмерное пространство так, чтобы в нём они остались на равном расстоянии друг от друга. SNE обходит эту проблему, отображая равноудалённые точки исходного пространства очень близко друг другу. Это приводит к тому, что если при проецировании важно сохранить небольшие расстояния между объектами, то точки, находящиеся на даже среднем расстоянии друг от друга в исходном пространстве, будут лежать очень далеко друг от друга в двух- или трёхмерной плоскости, что исказит изображение.

t-SNE предлагает решение обеих проблем с помощью использования распределения со значительно более тяжёлыми хвостами, чем нормальное, для задания условных вероятностей в пространстве меньшей размерности. В оригинальной статье используется распределение Коши (t-распределение с одной степенью свободы):

$$q_{i|j} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}.$$

Использование такого распределения позволяет изменить градиент функции потерь таким образом, что в результате оптимизации далёкие точки будут отображены на большом расстоянии друг от друга, а близкие — на близком, что в итоге позволит сохранить расстояния между равноудалёнными точками. Более того, градиент функции потерь содержит только одну условную вероятность с экспонентой, что способствует лучшей сходимости. Оптимизация происходит при помощи градиентного спуска, хотя в оригинальной статье авторы предлагают некоторые методы её улучшения.

Основным гиперпараметром алгоритма, непосредственно влияющим на итоговую визуализацию, является перплексия (perplexity). Она вычисляется по формуле:

$$Perp(P_i) = 2^{H(P_i)},$$

где $H(P_i)$ — этнропия Шеннона, измеряемая в битах:

$$H((P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

При заданной пользователем перплексии алгоритм при помощи бинарного поиска подбирает дисперсию нормального распределния σ_i таким образом, чтобы получить распределение P_i с соответсвующей перплексией. Таким образом гиперпараметр может рассматривать как своеобразная гладкая аппроксимация числа ближайших соседей каждой точки. Авторы оригинальной статьи утверждают, что алгоритм робастен к различным значениям гиперпараметра и рекомендуют выбирать его из диапазона от 5 до 50.

Сравнение авторами t-SNE с SNE, Isomap и LLE показало, что качество визуализации (на основе здравого смысла) первого превышает все прочие, при том что сложность по времени и памяти у этих алгоритмов схожа — $O(N^2)$. Нужно отметить, что из-за своей простоты и универсальности t-SNE может использоваться для визуализации в самых разных задачах.

Настраиваемые гиперпараметры:

- `perplexity` — перплексия.

Вычислительная сложность: $O(N^2)$.

Реализация всех перечисленных выше методов есть в `sklearn`. MDS реализован в варианте с расстояниями в метрической и неметрической версиях.

1.5. UMAP

UMAP (Uniform Approximation and Projection) был предложен в статье Leland McInnes, John Healy, James Melville в 2018 году [7]. Для математического описания работы алгоритма необходимо знание теории категорий и топологического анализа данных. В этом разделе будут рассмотрены только вычислительные особенности UMAP и интуиция метода. Все математические выкладки и подробные объяснения можно посмотреть в прекрасной оригинальной статье. Алгоритм состоит из двух шагов:

1. Пусть имеется набор данных $X = \{x_1, \dots, x_N\}$. Тогда для заданного гиперпараметра k, для каждого x_i найдем каким-нибудь алгоритмом поиска соседей с заданной метрикой расстояния d k ближайших соседей $\{x_{i_1}, \dots, x_{i_k}\}$.

   Для каждого x_i определяется функция расстояния ρ_i между этой точкой и его ближайщим соседом:

$$\rho_i = \min\{d(x_i, x_{i_j}) | 1 \leqslant j \leqslant k, d(x_i, x_{i_j}) > 0\},$$

и с помощью бинарного поиска определяется для каждой точки x_i диаметр окрестности σ_i так, чтобы выполнялось следующее равенство:

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

Далее задаётся взвешенный ориентированный граф $\hat{G} = (V, E, w)$. Вершинами этого графа будут точки входящего набора данных X. Ориентированные ребра задаются как $\{(x_i, x_{i_j}) | 1 \leqslant j \leqslant k, 1 \leqslant i \leqslant N\}$, а веса ребер рассчитываются по формуле

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

Пусть взвешенному ориентированному графу $\hat{G}$ соответствует матрица смежности A.

Следует отметить, что $w((x_i, x_{i_j}))$ — ассиметричная функция. Чтобы получить граф, более удобный для оптимизации, можно пересчитать веса по формуле

$$w\left(X_i, X_j\right) = w_i\left(X_i, X_j\right) + w_j\left(X_j, X_i\right) - w_i\left(X_i, X_j\right) w_j\left(X_j, X_i\right) :$$

и построить новый граф $\bar{G}$, с весами, принимающими значения от 0 до 1.

Матрица смежности B нового графа связана с матрицей A соотношением:

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

где символ $\circ$ обозначает поэлементное умножение.

2. На втором шаге A рассчитывается нормализованная версия лаплассиана $L = D^{1/2}(D - A)D^{1/2}$, где D — матрица, на главной диагонали которой расположены степени вершин графа G. Затем d первых собственных векторов матрицы L выбираются в качестве инициализации представления Y_{init} в новом признаковом пространстве.

3. На следующем шаге представление оптимизируется путем минимизации кросс-энтропии двух нечетких множеств. Как это относится ко всему, описанному выше, можно прочитать в таинственной второй секции упомянутой оригинальной статьи.

   Для простоты можно представлять себе это как минимизацию кросс-энтропии между значениями «отмасштабированных» весов w в исходном признаковом пространстве и весов w' в новом признаковом пространстве (вновь стоит упомянуть, что это лишь интуитивное и очень вольное объяснение)

   $$C\left(w, w'\right) = \sum_{i \sim j} w(i, j) \log\left(\frac{w(i, j)}{w'(i, j)}\right) + (1 - w(i, j)) \log\left(\frac{1 - w(i, j)}{1 - w'(i, j)}\right).$$

   Минимизация осуществляется с помощью стохастического градиентного спуска. Библиотеку `umap-learn` можно установить из `conda`.

Настраиваемые гиперпараметры:

- k — число соседей;
- d — размер пространства меньшей размерности;
- `min_dist` — желаемое расстояние между соседними точками в представлении;
- `n_epochs` — число эпох для оптимизации .

Вычислительная сложность: $O(kN)$.

1.6. TMAP

Метод Tree MAP (TMAP) был представлен в статье Probst и Reymond (2020) для визуализации больших наборов данных, в которых важно сохранить не только глобальные, но и локальные связи между признаками при снижении размерности [9]. Такие задачи возникают в лингвистике, биологии, медицине и ядерной физике. Оригинальная статья приводит в пример исследования по разработке лекарств. В этой области используются наборы данных (например, ChEMBL), в которых объектами являются результаты экспериментов над молекулами, а признаками — их химическое описание и свойства, представленные в виде бинарных или целочисленных векторов высокой размерности.

Качественная визуализация должна сохранить связи даже между ближайшими молекулами, так как это является важным при создании лекарства. Использование линейных методов, например, PCA, приведёт к потере этой информации. Нелинейные методы (t-SNE, UMAP, GTM, SOM) могут отразить требуемую структуру, но являются затратными по времени, так как их сложность на практике варьируется от $O(N^{1.14})$ до $O(N^5)$. Поэтому эти методы не могут применяться к большим наборам данных. TMAP, созданный для решения этой проблемы, способен обработать наборы, содержащие до 10^7 объектов и произвольное число признаков, с меньшими затратами по времени.

Алгоритм представляет собой комбинацию LSH и графовых методов и состоит из четырёх шагов.

1. Данные индексируются при помощи LSH Forest, что позволяет проводить поиск c-approximate kNN на следующей стадии за сублинейное по N время. Текстовые и бинарные переменные кодируются при помощи алгоритма MinHash, а числовые — с использованием разновидности этого же алгоритма, учитывающей веса. Опытным путём было установлено, что использование комбинации MinHash и LSH Forest хорошо работает при анализе молекул, потому что позволяет быстро рассчитывать расстояние Жаккара.
2. Из данных, индексированных LSH Forest, конструируется ненаправленный взвешенный c-approximate kNN граф (c-k-NNG). В оригинальной статье используется авторская модификация LSH Forest, повышающая эффективность работы, сложность которого по времени на данной стадии составляет $O(n(k \times k_c + \log n))$, где k — число искомых ближайших соседей, а k_c — авторский параметр LSH Forest [8]. Веса на рёбрах графа соответствуют расстояниям Жаккара между связанными вершинами. В зависимости от распределения данных и хеширования граф может получиться несвязным, если:

(a) в данных присутствуют выбросы, для которых расстояние Жаккара до всех прочих точек составит больше 1;

(b) образуются кластеры размера больше k, что приведёт к образованию компонент связности.

Тем не менее, даже если граф и получится несвязным, на дальнейшие шаги это никак не повлияет.

3. При помощи алгоритма Краскала на полученном c-k-NNG графе строится минимальное остовное дерево (MST), что позволяет убрать циклы и таким образом снизить вычислительную сложность. Примечательно, что UMAP и t-SNE для этой цели используют прунинг графов. Алгоритм Краскала сходится с использованием жадной стратегии, и его сложность по времени составляет $O(E + \log V)$, где E — число рёбер графа, V — число вершин. Такая сложность пренебрежимо мала по сравнению со сложностью на предыдущем шаге. В случае несвязного графа создаётся минимальный остовный лес.

4. Полученный граф строится на евклидовой плоскости. Для обработки миллионов вершин используется алгоритм со сложным названием[1] из библиотеки `OGDF`, написанной на C++.

Официальный сайт представляет инструкции и замечательные визуализации, полученные в результате работы алгоритма. Библиотеку `tmap` на Python можно установить из `conda` (на Windows устанавливается через WSL).

В оригинальной статье приводится сравнение TMAP, t-SNE и UMAP на данных о молекулах из ChEMBL (2^{32} признаков). TMAP выигрывает как по времени, так и по памяти, и лучше (визуально) сохраняет локальные связи между молекулами. Авторы также отмечают, что TMAP сам по себе может использоваться для поиска интересных особенностей в данных. В пример приводится результат работы алгоритма на корпусе Gutenberg, состоящим из произведений 142 авторов на английском языке, закодированных при помощи мешка слов (более миллиона двухсот тысяч признаков). На построенных деревьях работы разных авторов находятся на различных ветвях, как и различные по жанру произведения одного автора.

Настраиваемые гиперпараметры:

- k — число ближайших соседей для c-k-NNG.
- Размер векторов MinHash и число префиксных деревьев LSH Forest.

2 Метрики качества визуализации

На основе примеров, представленных в литературе, можно выделить два класса метрик качества визуализации в зависимости от поставленной задачи. Первый класс базируется на здравом смысле и возникает в случаях, когда исследователь обладает теоретическими знаниями о том, насколько адекватно полученное представленые данных. Типичные примеры: кластеризация осмысленных сущностей (MNIST, торго-

[1] Spring-electrical model layout algorithm with multilevel multipole-based force approximation.

вые точки на карте), а также всевозможные случаи из области науки (геодезия, изображение иероглифов) [10], [1].

Второй класс метрик основан на идее сделать процесс оценки качества более универсальным при помощи использования свойств данных, например, расстояний между точками. Ниже описываются некоторые такие метрики, реализованные в пакете `coranking`.

2.1. Коранговая матрица

Сначала необходимо ввести понятие коранговой матрицы Q, которая используется для расчета всех рассмотренных ниже метрик качества визуализации [5].

Пусть σ_{ij} обозначает расстояние между точками ξ_i и ξ_j в исходном многообразии, а ij — расстояние между точками x_i и x_j в пространстве меньшей размерности.

Пусть рангом точки ξ_i относительно точки ξ_j называется $\rho_{ij} = |k : \delta_{ik} < \delta_{ij}$ или $(\delta_{ik} = \delta_{ij}$ и $k < j)\,|$. Аналогично, рангом точки x_i относительно точки x_j называется $r_{ij} = |k : d_{ik} < d_{ij}$ или $(d_{ik} = d_{ij}$ и $k < j)\,|$.

Тогда коранговая матрица — это

$$Q = [q_{kl}]_{1 \leqslant k,l \leqslant N-1}, \text{ где } q_{kl} = |(i,j : p_{ij}|.$$

2.2. Достоверность и непрерывность

Достоверность (trustworthiness) и непрерывность (continuity) как метрики качества визуализации были представлены в работе Venna и Kaski (2006) [13]. Эти метрики основаны на простой идее: когда исследователь изучает свойства конкретной точки из данных, он обычно смотрит на точки в её окрестности. Поэтому качественная визуализация должна сохранять такие окрестности точек.

Проекция точки называется достоверной, если её K ближайших соседей в новом пространстве также являются её ближайшими соседями в исходном пространстве. Формально, пусть N обозначает число объектов в наборе данных. Для каждой точки i в исходном пространстве все остальные точки ранжируются в соответствии с расстоянием от них до точки i. Пусть $r(i,j)$ обозначает порядок точки j в таком ранжировании. Также пусть $U_K(i)$ обозначает множество точек, находящихся в K-окрестности точки i в новом, но не в исходном, пространстве. Тогда достоверность определяется как

$$T(K) = 1 - \frac{2}{NK(2N-3K-1)} \sum_{i=1}^{N} \sum_{j \in U_K(i)} (r(i,j) - K) = 1 - \frac{2}{NK(2N-3K-1)} \sum_{(k,l) \in LL_K} (k-K) q_{kl},$$

где q_{kl} - элементы нижнего левого LL_K блока коранговой матрицы Q

Непрерывность строится на схожей идее для нового пространства. Пусть $V_K(i)$ обозначает множество точек, которые содержатся в K-окрестности точки i в исходном, но не в новом пространстве. Пусть $\hat{r}(i,j)$

обозначает порядок точки j в ранжировании всех прочих точек по расстоянию до точки i в новом пространстве. Тогда непрерывность определяется как

$$C(K) = 1 - \frac{2}{NK(2N - 3K - 1)} \sum_{i=1}^{N} \sum_{j \in V_K(i)} (\hat{r}(i, j) - K) = 1 - \frac{2}{NK(2N - 3K - 1)} \sum_{(k,l) \in UR_K} (k - K) q_{kl},$$

где q_{kl} - элементы верхнего правого UR_K блока кoранговой матрицы Q

В оригинальной статье предлагается рассчитывать метрики для разного числа соседей K при сравнении алгоритмов. Особое внимание следует обращать на значения метрик при небольших K, так как они является показателем сохранения локальных особенностей данных. Лучшим признаётся алгоритм с наибольшим значением достоверности и непрерывности.

2.3. LCMC

LCMC (local continuity meta-criterion), как и достоверность и непрерывность, основана на идее анализа того, как меняются K-окрестности точек в новом пространстве меньшей размерности при разных значениях параметра K.

LCMC рассчитывается по формуле:

$$U_{LC}(K) = \frac{K}{1 - N} + \frac{1}{NK} \sum_{(k,l) \in UL_k} q_{kl},$$

где q_{kl} - элементы верхнего левого UL_K блока кoранговой матрицы Q.

Большее значение критерия соответствует лучшему качеству работы алгоритма.

3 Сравнение методов визуализации

В этом разделе представлены результаты визуализаций на наборах данных Swissroll (геометрическая проекция), FASHION MNIST (изображения одежды и обуви), News Aggregator Dataset (заголовки новостей) и 20 Newsgroups (посты в группах), полученные описанными выше методами. Качество визуализаций оценивалось на основе здравого смысла и при помощи универсальных метрик. Следует отметить, что метрики невозможно рассчитать для визуализаций TMAP ввиду специфичности их построения, поэтому результаты работы этого алгоритма оценивались только на основе их адекватности.

3.1. Swissroll

Набор данных представляет собой точки из нормального распределения на двумерной плоскости, переведённые в трёхмерное пространство некоторым гладким преобразованием. Трёхмерное изображение набора приведено на рисунке 1.

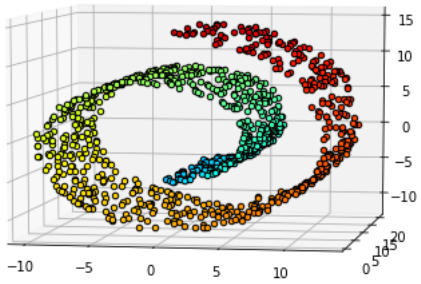

Рис. 1: Набор данных Swissroll

Задача визуализации состоит в том, чтобы перевести объекты обратно на двумерную плоскость. Выборка для эксперимента состоит из 1000 точек. Результаты работы методов MDS, Isomap, LLE, t-SNE и UMAP с подобранными[2] параметрами изображены на рисунке 2.

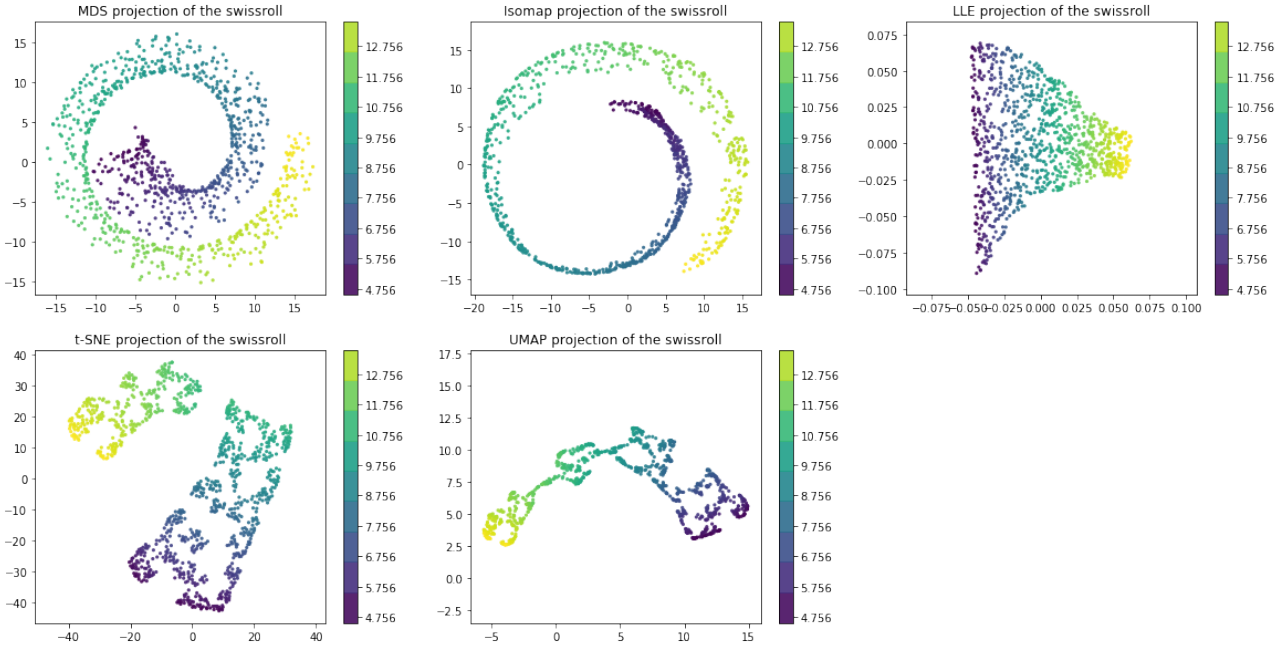

Рис. 2: Результаты работы методов на наборе данных Swissroll

Визуально понятно, что более простые методы (MDS и Isomap) при проецировании сохранили геометрическую форму объекта из трёхмерного пространства. Этот результат соответствует принципу их работы: при проецировании на плоскость, параллельную оси Oz на рисунке 1 попарные расстояния между точками остаются наиболее близкими. При этом, границы кластеров чисел разной величины (дискретизированные по colorbar) визуально различимы, и их можно разделить нелинейными кривыми. Проекция LLE практически идеально разделила точки на кластеры в зависимости от их величины, и разделяющие

[2] Здесь и далее: параметры различных методов подбирались так, чтобы достичь адекватного качества визуализации. В некоторых случаях устанавливались дефолтные параметры. Подробнее о влиянии гиперпараметров на качество визуализации смотрите раздел 4.

поверхности имеют линейный вид. То же относится и к t-SNE и UMAP, проекции которых, помимо этого, имеют форму полумесяца, то есть эти методы, как и MDS и Isomap, «раскрутили» изначальную спираль данных на двумерной плоскости. Результаты работы TMAP представлены на рисунке 3.

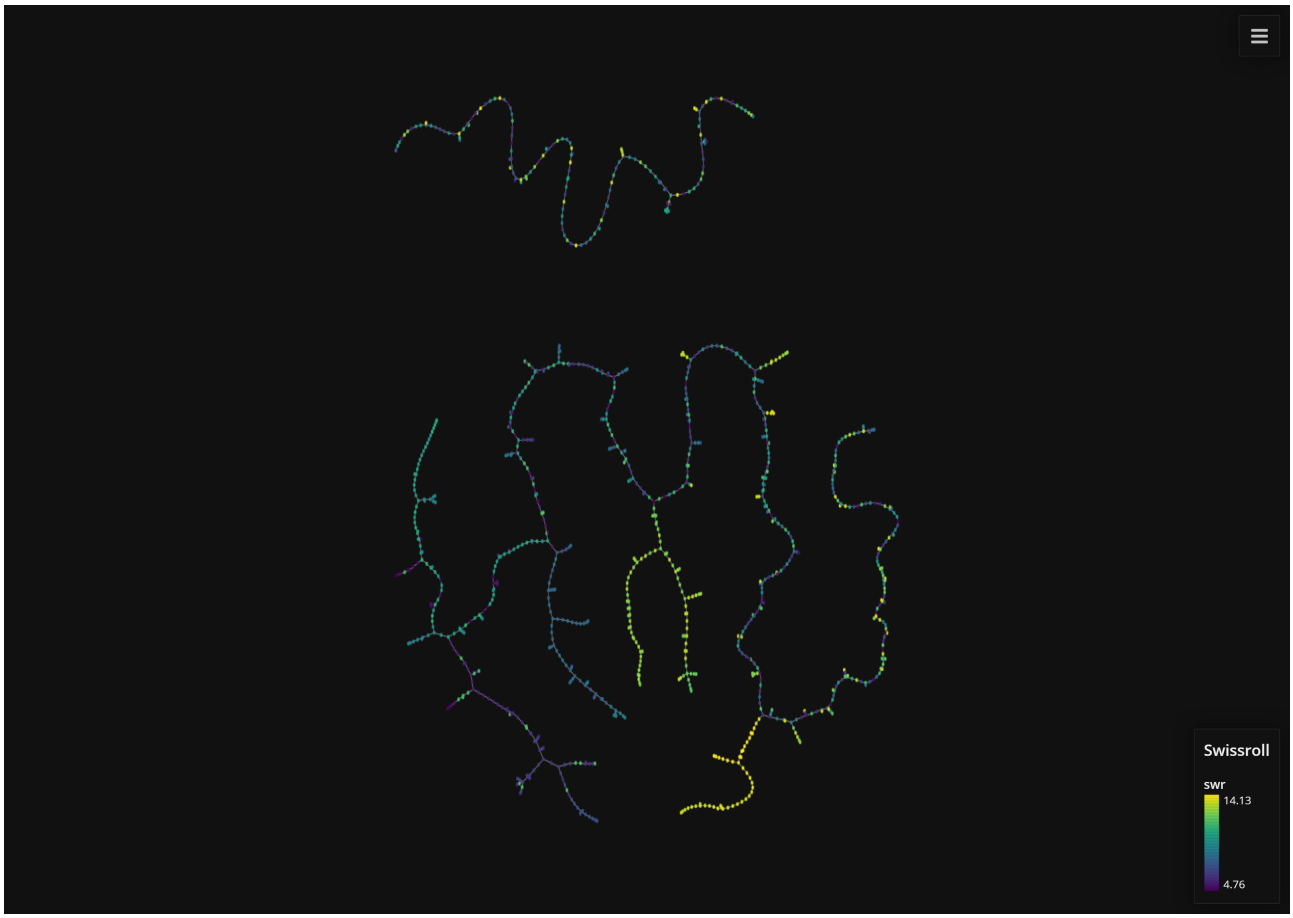

Рис. 3: Результаты работы TMAP на наборе данных Swissroll

Видно, что в нижней части построенного графа можно выделить кластеры объектов схожей величины, причём близкие по величине кластеры находятся недалеко друг от друга. Например, в левой нижней части графа тёмно-синий кластер переходит в зелёный, как и должно быть. Тем не менее, в правой части графа много неопределённых объектов: жёлтый, синий и зелёный цвета кластеров чередуются в хаотичном порядке. Верхняя компонента графа также состоит из таких неопределённых объектов.

Таким образом, с точки зрения здравого смысла лучший алгоритм зависит от задачи. Если требуется получить двумерную проекцию трёхмерной фигуры так, чтобы эти фигуры были как можно более похожи, то следует выбрать MDS или Isomap. Если же требуется линейно разделить точки, то лучшими вариантами будут LLE, t-SNE и UMAP. TMAP показал худшие результаты[3], так как не был создан для задач такого типа.

Затраченное время (здесь и далее — на компьютере с AMD A9 Radeon R5, 3.10 GHZ, 8 Gb RAM, Win10 WSL) представлено в таблице 1.

LLE показал лучший результат по времени работы. MDS продемонстрировал худший результат, обраба-

[3]Хотя и изобразил замечательного кальмара!

Алгоритм	Время работы (сек.)
MDS	59
Isomap	1
LLE	0.5
t-SNE	8
UMAP	15
TMAP	2

Таблица 1: Время работы использованных методов на наборе данных Swissroll

тывая данные в несколько раз дольше, чем все прочие алгоритмы.

Описанные выше метрики изображены на рисунке 4.

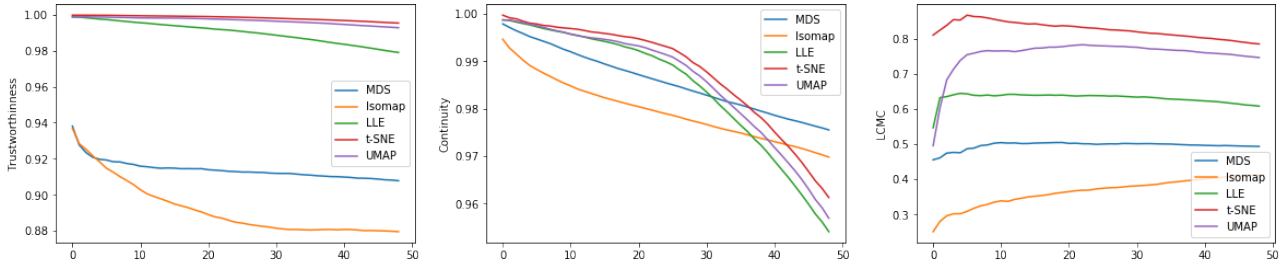

Рис. 4: Достоверность, непрерывность и LCMC для методов на наборе данных Swissroll

При небольшом числе соседей t-SNE занимает лидирующую позицию по всем метрикам, UMAP — второе место, что означает, что эти алгоритмы лучше всего сохранили локальные особенности данных. В глобальном масштабе они также оказались лучшими по достоверности и LCMC, в то время как по непрерывности лучшими стали MDS и Isomap. Такой результат означает, что глобально проекции t-SNE и UMAP не склонны группировать далеко расположенные в исходном пространстве точки (высокая достоверность), но при этом склонны располагать на значительно расстоянии близкие точки (низкая непрерывность). MDS и Isomap стремятся сохранить расстояния в целом, поэтому в глобальном масштабе показывают более хороший результат.

Таким образом, в совокупности исследованных критериев, лучший алгоритм для проецирования геометрического объекта зависит от задачи. Если требуется сохранить геометрическую форму, следует использовать Isomap. Если требутеся сохранить локальные расстояния между объектами и добиться их линейного разделения по величине, стоит применить LLE, t-SNE или UMAP. В последнем случае, если особо важна скорость работы, LLE является лучшим кандидатом.

3.2. FASHION MNIST

Набор данных состоит из чёрно-белых изображений предметов одежды одного из десяти классов. Изображения имеют размер 28×28, то есть общее число признаков равно 784. Задача визуализации — пред-

ставить набор на двумерной плоскости так, чтобы изображения одного предмета одежды попали в собственный кластер. Набор данных содержит 70 000 картинок. Это очень утомительная (или непосильная) задача для большинства приведенных алгоритмов, поэтому в работе для получения визуализаций использовалась случайная подвыборка из 3 500 наблюдений. Стоит отметить, что на обработку полного набора данных у UMAP уходит около 5 минут, то есть этот алгоритм, в отличие от многих, пригоден для эффективной работы с большими наборами данных.

Результаты работы методов MDS, Isomap, LLE, t-SNE и UMAP с дефолтными параметрами изображены на рисунке 5.

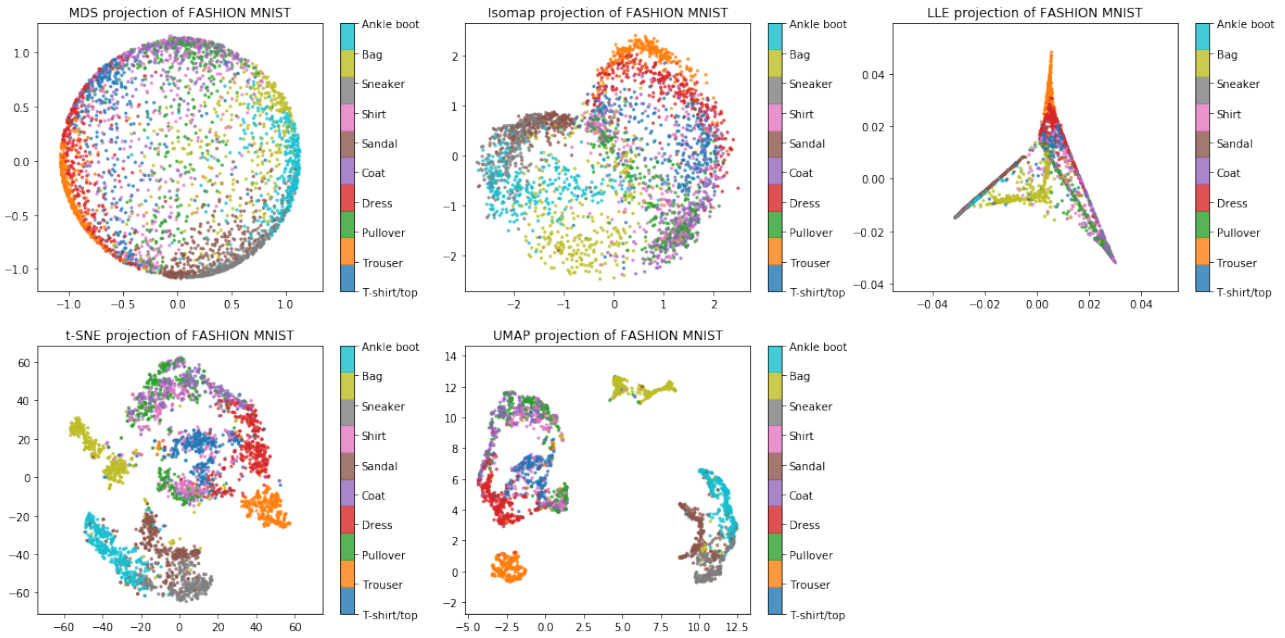

Рис. 5: Результаты работы методов на наборе данных FASHION MNIST

Видно, что MDS, Isomap и LLE достаточно плохо справились с задачей[4]: на проекциях визуально различимы некоторые кластеры (брюки и платья), однако границы сильно размыты, и разделяющая поверхность имеет сложный нелинейный вид. Есть и кластеры (сумки для MDS, рубашки для Isomap, ботильоны для LLE), оказавшиеся неотделимыми. t-SNE показал более хороший результат: с небольшой ошибкой можно отделить ботильоны, сандалии, сникеры, брюки, футболки, сумки и платья. Свитеры и плащи оказались неразделимыми, что является логически приемлемым. UMAP также хорошо отделил сумки, ботильоны, брюки, платья и футболки, но свитеры и плащи, а также сникеры и сандалии оказались неразделимыми. Результаты работы TMAP представлены на рисунке 6.

Полученный граф достаточно хорошо решает задачу: отчётливо разделимы брюки, ботильоны и сумки. Как и в случае UMAP, сандалии и сникеры и свитеры и плащи оказались смешаны. В целом, граф достаточно похож на проекцию UMAP в плане расположения кластеров, например, платья и футболки лежат близко к плащам и свитерам, а последние находятся далеко от брюк.

Затраченное время представлено в таблице 2.

[4] Тем не менее, мыльный пузырь и сердце получились восхитительными!

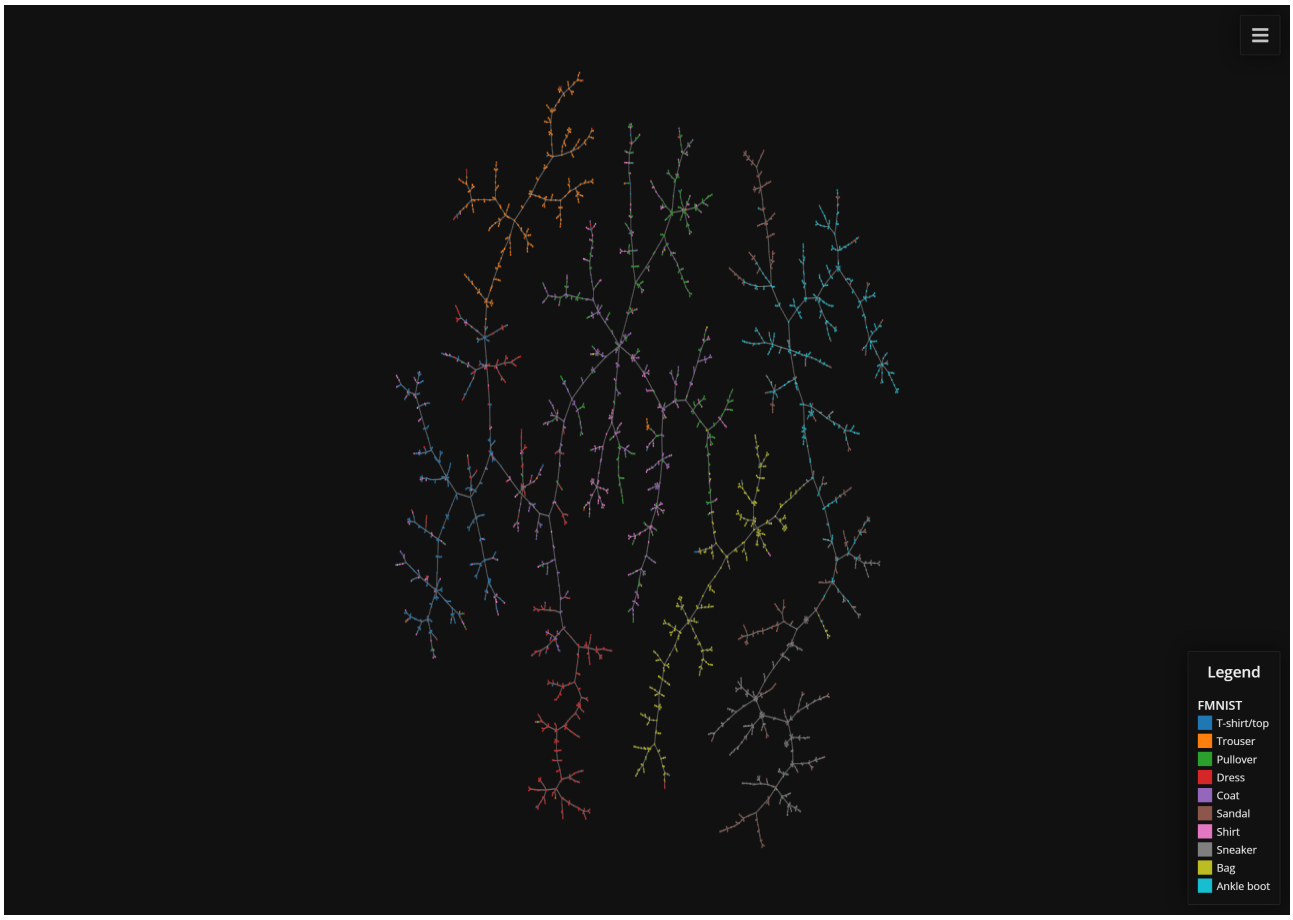

Рис. 6: Результаты работы TMAP на наборе данных FASHION MNIST

Алгоритм	Время работы (сек.)
MDS	1002
Isomap	43
LLE	22
t-SNE	52
UMAP	25
TMAP	25

Таблица 2: Время работы использованных методов на наборе данных FASHION MNIST

LLE показал лучший результат по времени. Примечательно, что t-SNE, проекция которого лучше всего разделяет классы, компенсирует качество визулизации временем работы, более чем в два раза превышающем прочие методы, кроме MDS. Последний выделяется: если все прочие алгоритмы работают в пределах минуты, то MDS выполняется за 15-20 минут.

Метрики изображены на рисунке 7.

В данном случае по всем метрикам и локальные, и глобальные связи лучше всего сохраняет t-SNE, на втором месте — UMAP.

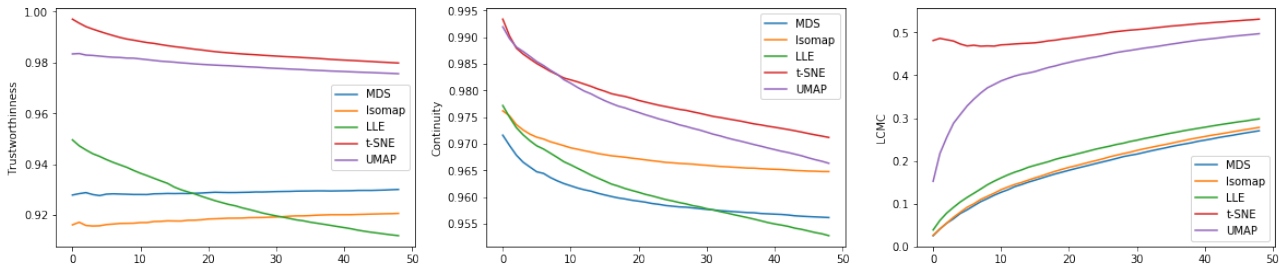

Рис. 7: Достоверность, непрерывность и LCMC для методов на наборе данных FASHION MNIST

Таким образом, в задаче визуализации изображений для последующей кластеризации лучшее качество показал t-SNE как на основе здравого смысла, так и по метрикам. Тем не менее, время его работы оказалось в два раза больше, чем у UMAP, который справился с задачей лишь немного хуже. Поэтому если в выборе между временем работы и точностью важно первое, следует использовать UMAP, если второе, то t-SNE. Отдельно стоит упомянуть, что TMAP также показал хорошие результаты и сопоставим с UMAP как по качеству визуализации, так и по времени работы.

3.3. News Aggregator Dataset

Набор данных содержит заголовки новостей, собранные сетевым аггрегатором с марта по август 2014 года. Каждый заголовок является предложением, которое относится к одной из четырёх категорий: бизнес, науки и технологии, развлечения и здоровье. Задача визуализации состоит в том, чтобы представить эти предложения на двумерной плоскости так, чтобы рядом оказались заголовки из одной категории.

Перед применением алгоритмов визуализации, предложения были лемматизированы, очищены от стоп-слов и векторизированы при помощи TF-IDF. Стоит отметить, что в оригинальном наборе данных содержится более 422 тыс. предложений, но для эксперимента была взята случайная подвыборка из 1000 наблюдений, так как обработка даже 5 тыс. объектов оказалась очень времязатратной для MDS и t-SNE. Результаты работы методов MDS, Isomap, LLE, t-SNE и UMAP с дефолтными параметрами изображены на рисунке 8.

Все алгоритмы отработали плохо: ни один не позволяет разделить предложения по категориям. TMAP, показал сходие результаты, которые представлены на рисунке 9. Видно, что алгоритм также не справился с задачей: примерно отделимы только предложения, относящиеся к категории здоровья. Все прочие категории оказались неразделимы. Затраченное время представлено в таблице 3. Лучшим по времени оказался UMAP. Примечательно, что в отличие от других наборов, время работы MDS сопоставимо с другими алгоритмами.

Метрики изображены на рисунке 10. На основании метрик лучшее качество показали t-SNE и UMAP как в локальном, так и в глобальном масштабах. Тем не менее, в данном случае разумным кажется полагаться на здравый смысл, а не на метрики: все алгоримы не справились с поставленной задачей.

Таким образом, ни один алгоритм не справился с визуализацией на текстах. Чтобы устранить эффект

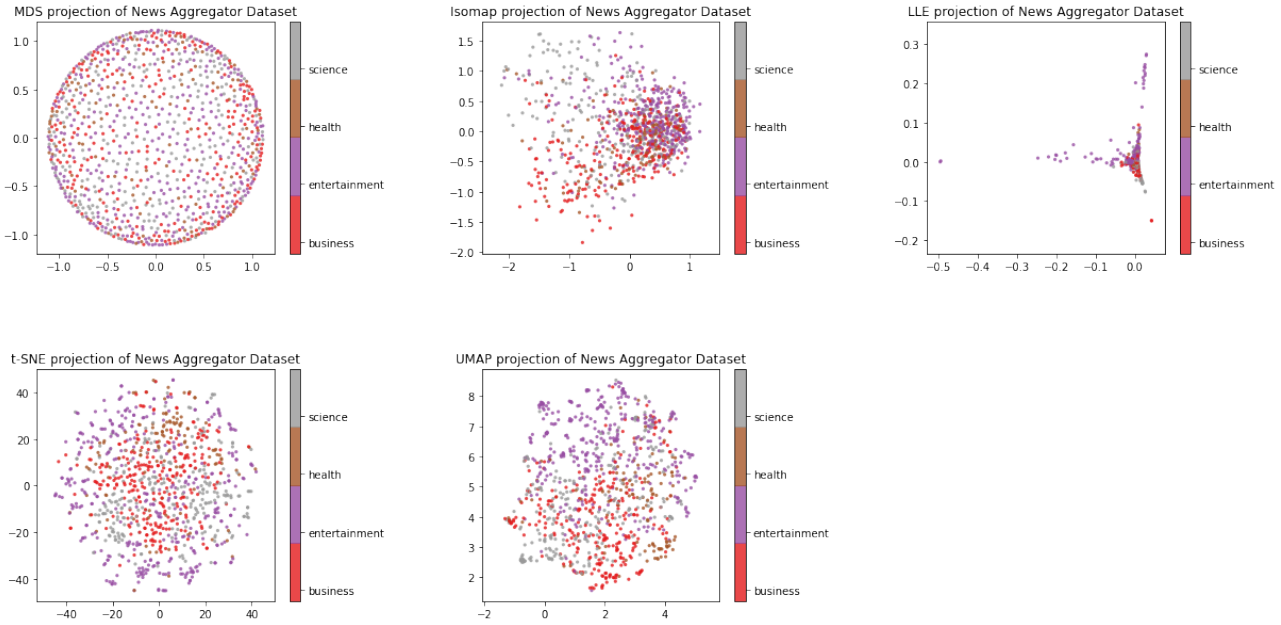

Рис. 8: Результаты работы методов на наборе данных News Aggregator Dataset

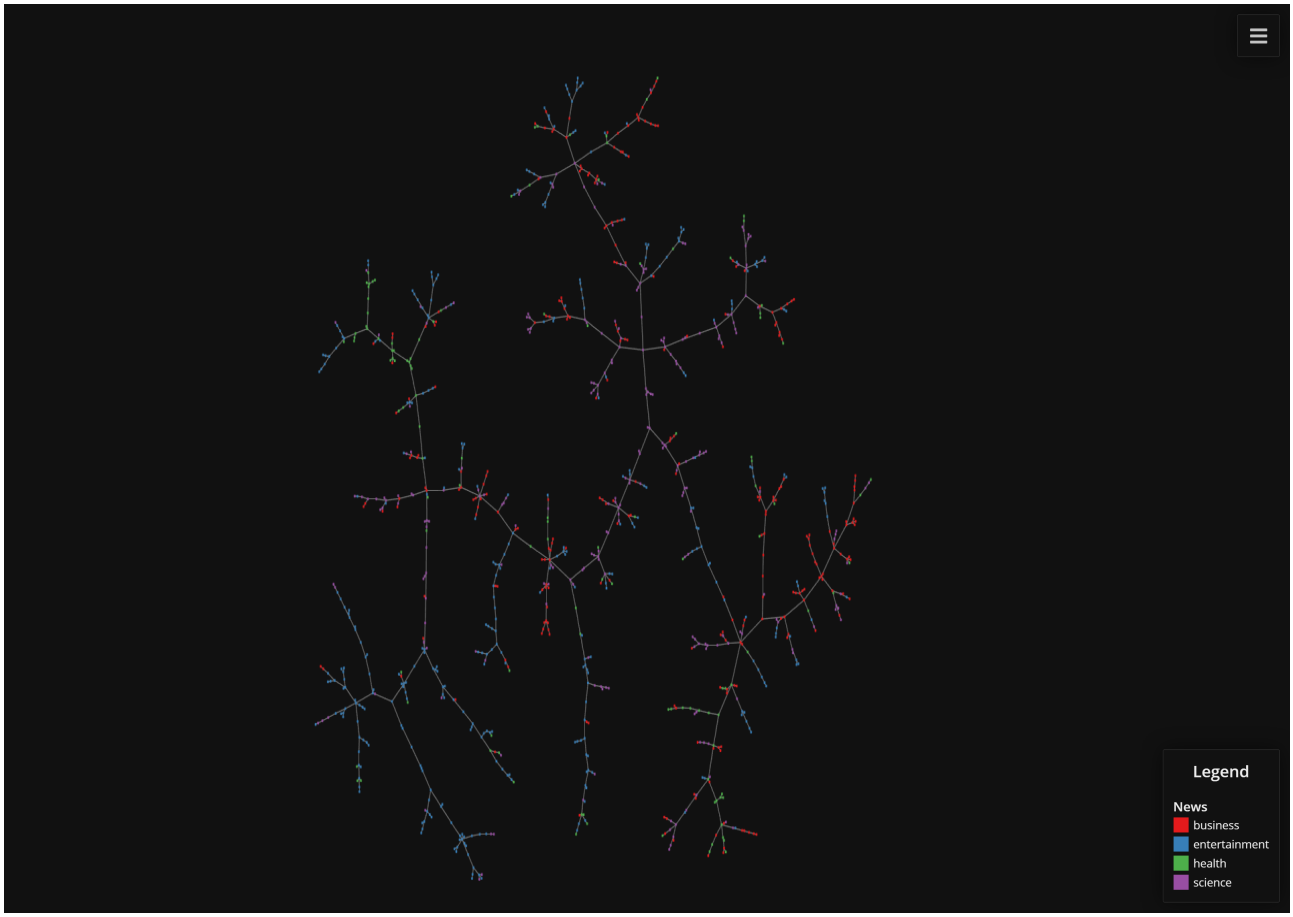

Рис. 9: Результаты работы TMAP на наборе данных News Aggregator

влияния качества набора данных, эксперимент был повторён на стандартном корпусе текстов из `sklearn` 20 Newsgroups. Набор содержит 20 000 постов из 20 новостных групп, соответствующих разным темам.

Алгоритм	Время работы (сек.)
MDS	22
Isomap	13
LLE	13
t-SNE	22
UMAP	6
TMAP	31

Таблица 3: Время работы использованных методов на наборе данных News Aggregator Dataset

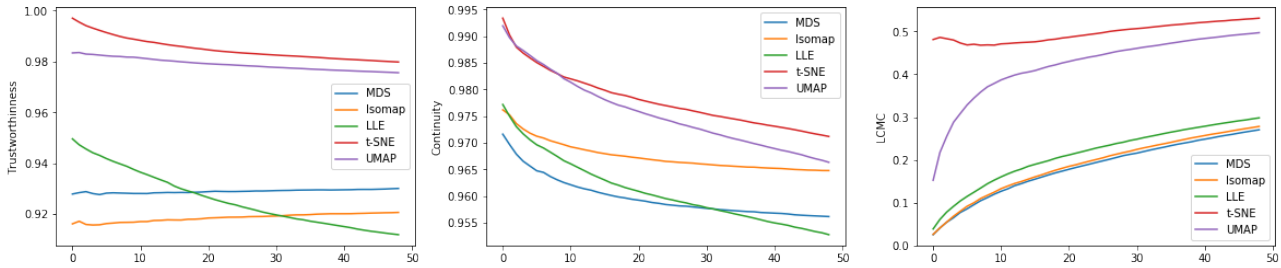

Рис. 10: Достоверность, непрерывность и LCMC для методов на наборе данных News Aggregator Dataset

На этом наборе были протестированы методы, показавшие себя лучшими в предыдущих задачах: t-SNE и UMAP. Результаты приведены на рисунке 11.

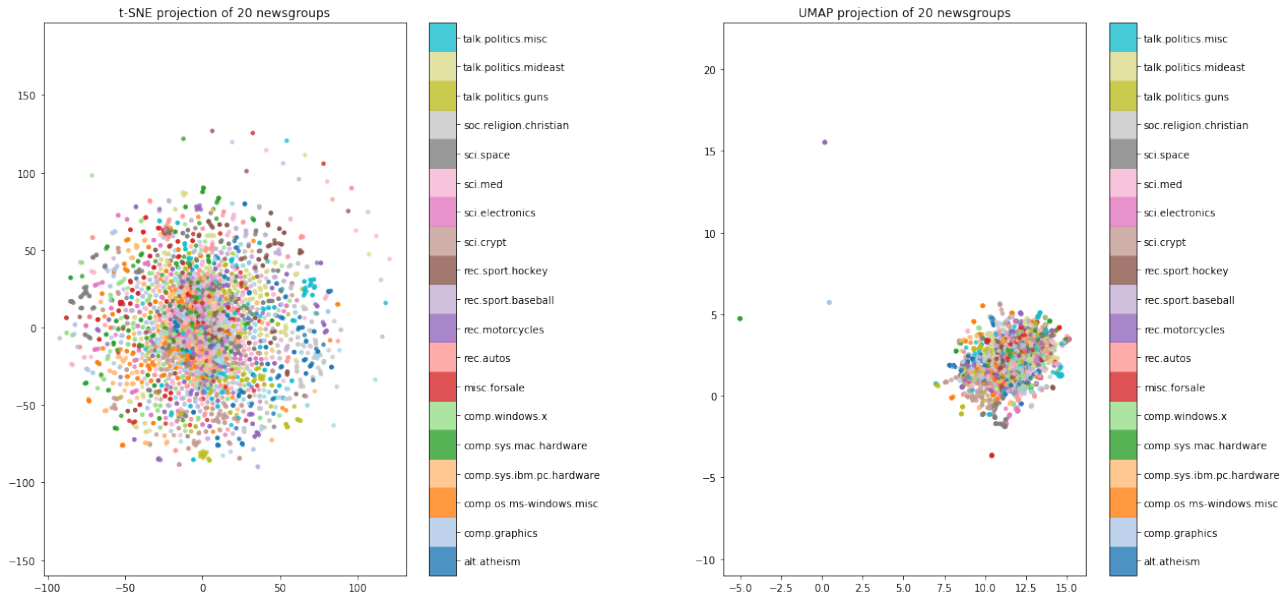

Рис. 11: Результаты работы t-SNE и UMAP на наборе данных 20 Newsgroup

К сожалению, эксперимент провалился: алгоритму не удалось выделить классы новостей по тематикам. Можно сделать вывод о том, что рассмотренные методы визуализации плохо работают на текстовых данных, во всяком случае при использовании выборок небольших размеров.

4 Настройка гиперпараметров

В этом разделе проиллюстрировано, как меняется качество визуализации описанных алгоритмов в зависимости от значений гиперпараметров. Все эксперименты были проведены на наборе данных FASHION MNIST. Влияние гиперпараметров на качество визуализации MDS не рассматривалось, так как стоит признать, что этот алгоритм является бесспорным аутсайдером[5]. При этом время его работы даже на небольшой выборке велико. Для экономии места изображения для тестирования TMAP вынесены в код. Для этого алгоритма изменялись размер векторов MinHash и число префиксных деревьев в LSH Forest. Было получено, что в целом, качество визуализации остаётся примерно одинаковым: меняется струкутра деревьев и расположение в них классов, однако чёткость их выделения и логика расположения сохраняются.

Для всех остальных методов рассмотривается параметр `n_neighbors` (для t-SNE — `perplexity`), регулирующий число соседей в окрестности многообразия для каждой точки. Как правило, этот параметр оказывает наибольшее влияние на визуализацию, так как устанавливает баланс между сохранением локальных и глобальных особенностей исходного признакового пространства.

На рисунках 12 — 15 представлены визуализации работы алгоритмов при разных значениях гиперпараметров.

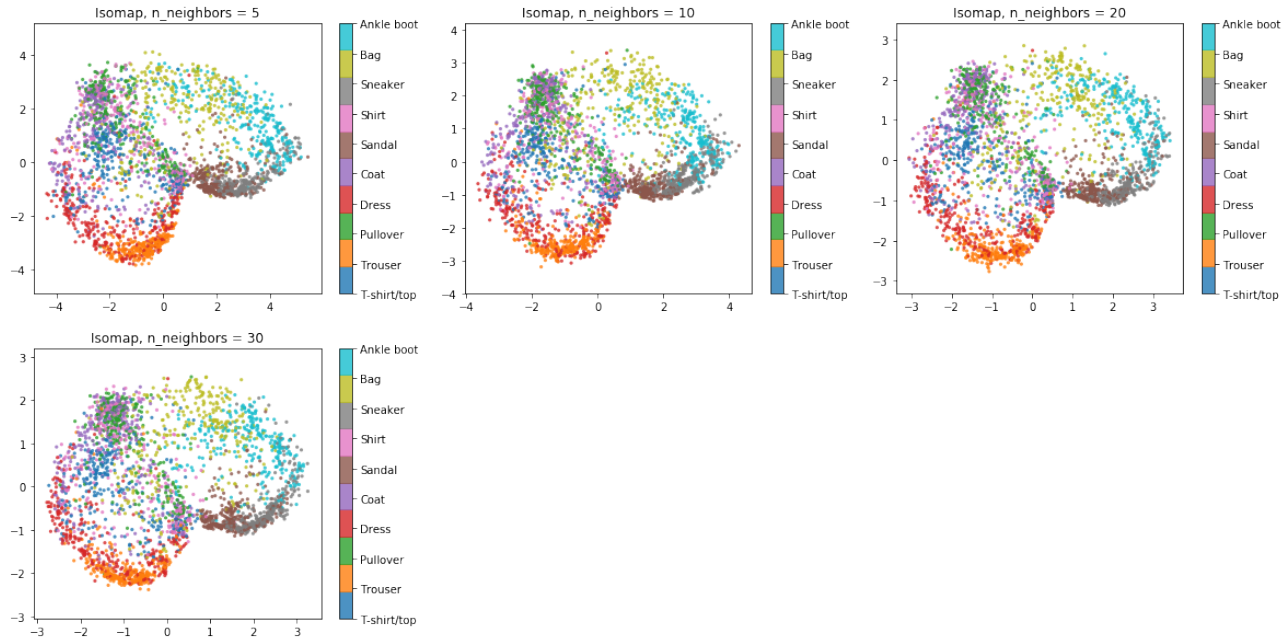

Рис. 12: Isomap-визуализация FMNIST

Визуализация с помощью Isomap (рисунок 12) практически не меняется при различных значениях параметра. Изменения можно заметить, только наблюдая за отдельными точками. Художественные возможности алгоритма поражают, однако границы классов остаются размытыми, а некоторые отдельные классы сильно перемешаны, например, платья (красные точки) и брюки (оранжевые точки).

В случае LLE (рисунок 13) качество визуализации явно растет при больших значениях гиперпараметра.

[5] :)

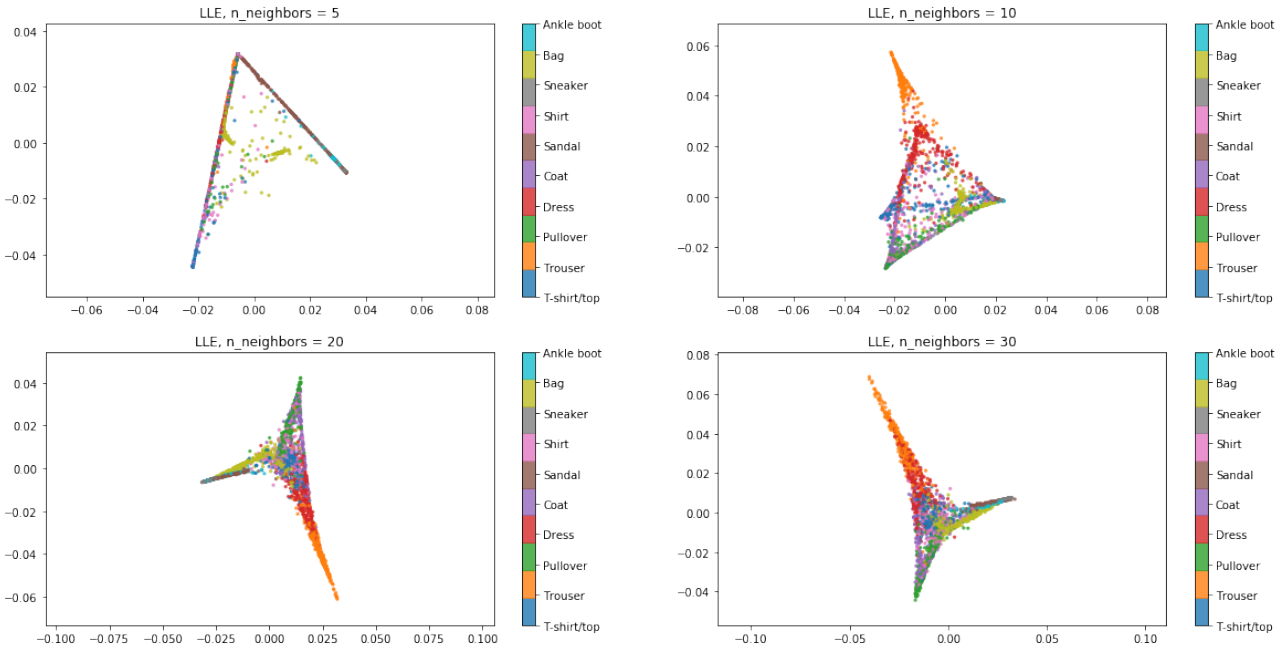

Рис. 13: LLE-визуализация FMNIST

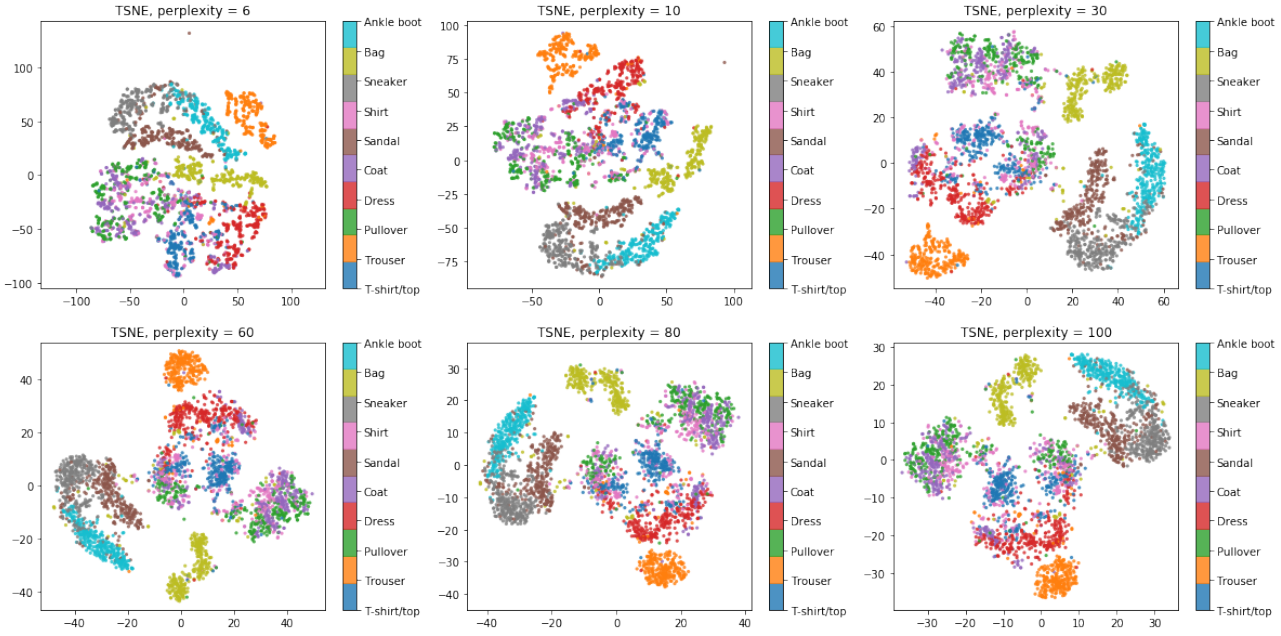

Рис. 14: t-SNE-визуализация FMNIST

Данные хорошо группируются по кластерам, однако все кластеры находятся близко друг к другу.

Как правило, для t-SNE классическая рекомендация — использовать бóльшие значения параметра `perplexity` для больших наборов данных. На примере подвыборки из FMNIST (рисунок 14) видно, что при маленьких значениях параметра «облака» классов оформлены плохо. С ростом значения `perplexity` классы отдельных видов одежды располагаются друг от друга дальше, то есть алгоритм лучше учитывает глобальную структуру данных. Значение параметра около 60 кажется оптимальным.

В случае UMAP (рисунок 15) визуально кажется, что значимых улучшений при различных значениях па-

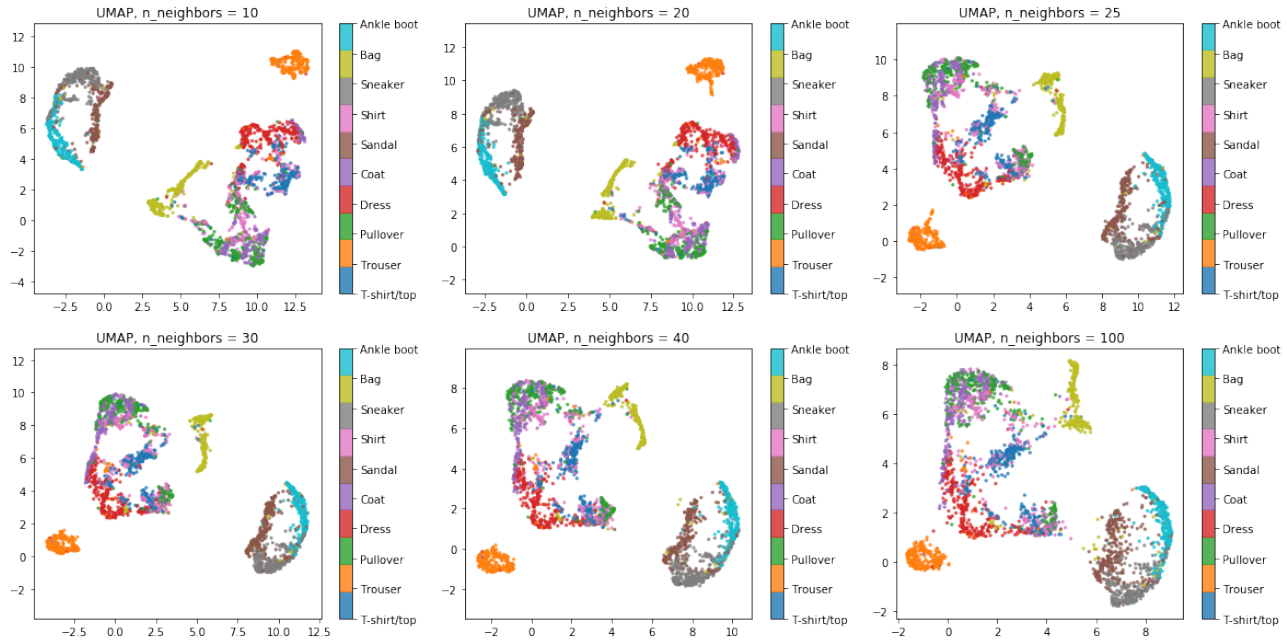

Рис. 15: UMAP-визуализация FMNIST

раметра нет. Тем не менее, слишком большое число соседей приводит к тому, что границы классов становятся размытыми.

Таким образом, результаты экспериментов показывают, что лучшими при решении различных задач оказались t-SNE и UMAP. Первый выигрывает в качестве визуализации, а второй — в её скорости, при этом не сильно уступая t-SNE по качеству. Линейный MDS оказался хуже всех прочих методов по обоим показателям. TMAP, сопоставимый по времени работы с UMAP, проигрывает на некоторых стандартных наборах данных по качеству визуализации, так как был создан в первую очередь для решения специфических научных задач.

Список литературы

[1] Mehala Balamurali и Arman Melkumyan. «t-SNE based visualisation and clustering of geological domain». В: *International Conference on Neural Information Processing*. Springer. 2016, с. 565—572.

[2] Andreas Buja и др. «Data visualization with multidimensional scaling». В: *Journal of computational and graphical statistics* 17.2 (2008), с. 444—472.

[3] Geoffrey E Hinton и Sam T Roweis. «Stochastic neighbor embedding». В: *Advances in neural information processing systems*. 2003, с. 857—864.

[4] Joseph B Kruskal. «Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis». В: *Psychometrika* 29.1 (1964), с. 1—27.

[5] John A Lee и Michel Verleysen. «Quality assessment of dimensionality reduction: Rank-based criteria». В: *Neurocomputing* 72.7-9 (2009), с. 1431—1443.

[6] Laurens van der Maaten и Geoffrey Hinton. «Visualizing data using t-SNE». В: *Journal of machine learning research* 9.Nov (2008), с. 2579—2605.

[7] Leland McInnes, John Healy и James Melville. «Umap: Uniform manifold approximation and projection for dimension reduction». В: *arXiv preprint arXiv:1802.03426* (2018).

[8] Daniel Probst и Jean-Louis Reymond. «A probabilistic molecular fingerprint for big data settings». В: *Journal of cheminformatics* 10.1 (2018), с. 66.

[9] Daniel Probst и Jean-Louis Reymond. «Visualization of very large high-dimensional data sets as minimum spanning trees». В: *Journal of Cheminformatics* 12.1 (2020), с. 1—13.

[10] Edgar Roman-Rangel и Stephane Marchand-Maillet. «Assessing Deep Learning Architectures for Visualizing Maya Hieroglyphs». В: *Mexican Conference on Pattern Recognition*. Springer. 2017, с. 137—146.

[11] Sam T Roweis и Lawrence K Saul. «Nonlinear dimensionality reduction by locally linear embedding». В: *science* 290.5500 (2000), с. 2323—2326.

[12] Joshua B Tenenbaum, Vin De Silva и John C Langford. «A global geometric framework for nonlinear dimensionality reduction». В: *science* 290.5500 (2000), с. 2319—2323.

[13] Jarkko Venna и Samuel Kaski. «Local multidimensional scaling». В: *Neural Networks* 19.6-7 (2006), с. 889—899.