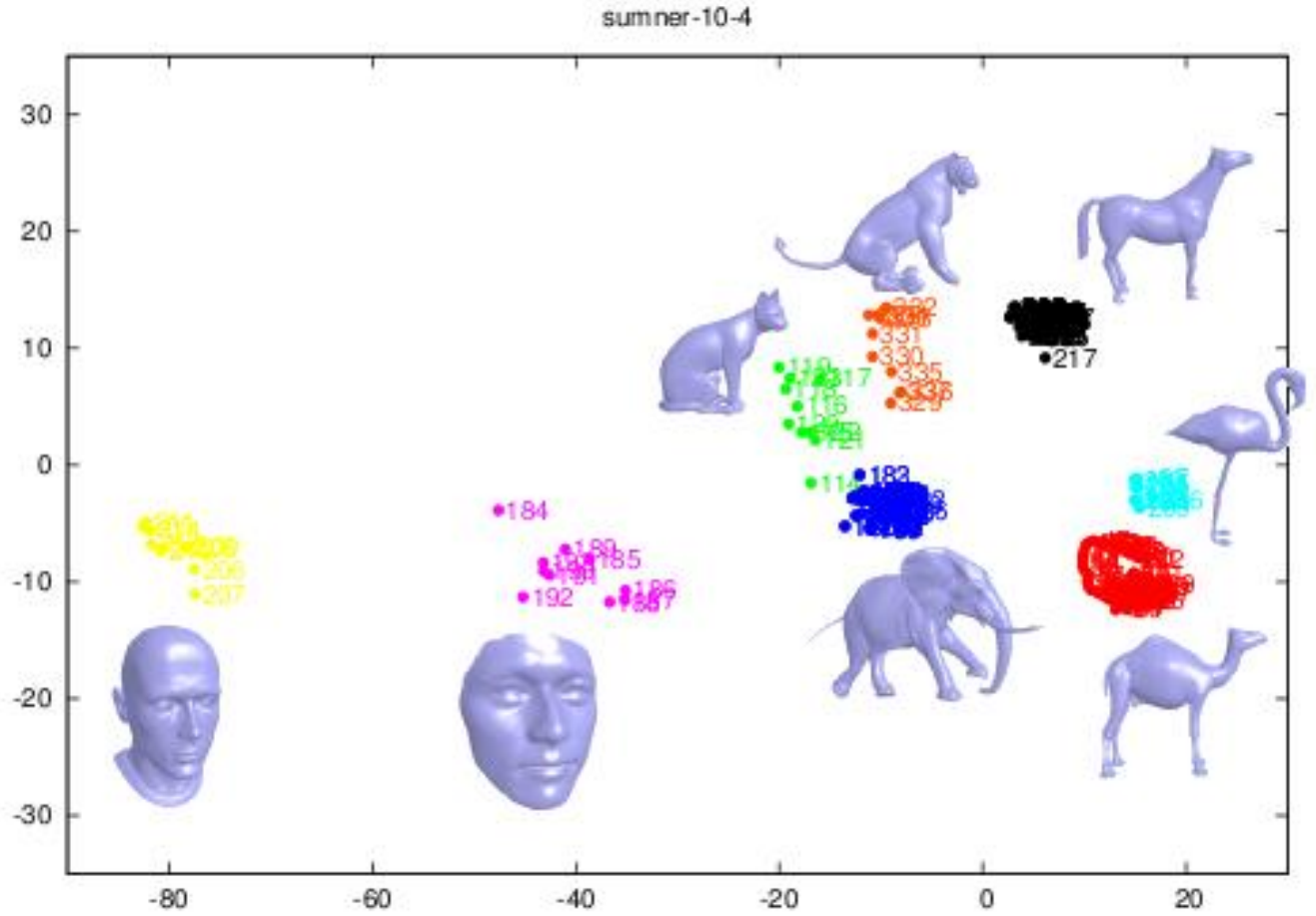
Manifold embeddings (Visualization)

ВИЗУАЛИЗАЦИЯ

Задача визуализации состоит в отображении объектов в 2х- или 3хмерное пространство с сохранением отношений между ними.



1. MULTIDIMENSIONAL SCALING (MDS)

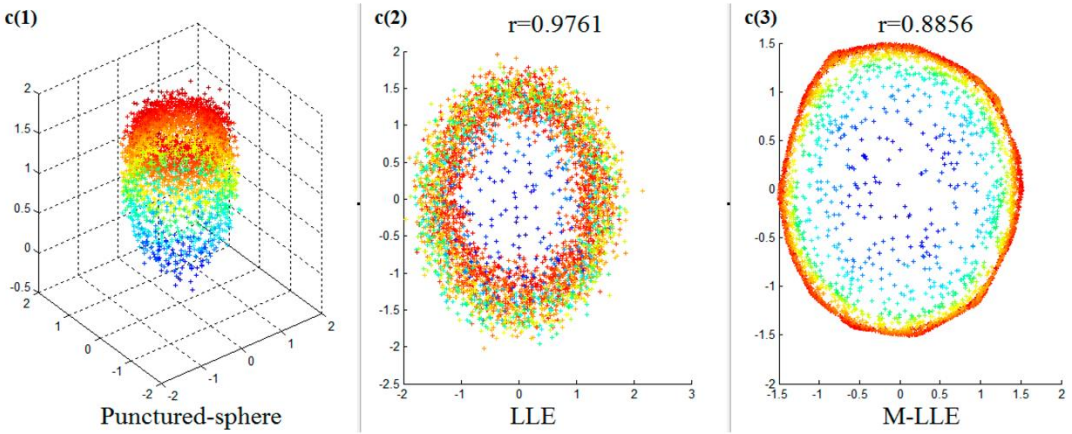
Идея метода – ***минимизация квадратов отклонений между исходными и новыми попарными расстояниями***:

$$\sum_{i \neq j}^{l} \left(\rho\left(x_i, x_j\right) - \rho\left(z_i, z_j\right)\right)^2 \rightarrow \min_{z_1, \ldots, z_l}$$

2. LOCALLY LINEAR EMBEDDING (LLE)

Locally Linear Embedding (LLE) — это алгоритм нелинейного понижения размерности, который состоит из локального применения PCA, но глобально получается нелинейным.

- LLE предполагает, что каждая точка в исходном пространстве может быть аппроксимирована как линейная комбинация своих ближайших соседей. Задача LLE заключается в том, чтобы найти такое представление точек в низкоразмерном пространстве, которое сохраняет эти линейные зависимости.



LOCALLY LINEAR EMBEDDING (LLE)

1. Поиск ближайших соседей:

Для каждой точки $\mathbf{x}_i$ в исходном пространстве:

- Найдите K ближайших соседей ($\mathbf{x}_j$) с использованием меры расстояния, например, Евклидового расстояния.
- Это задает локальное окрестностное пространство для каждой точки.

2. Вычисление весов реконструкции:

Для каждой точки $\mathbf{x}_i$ подберите веса w_{ij}, которые минимизируют ошибку реконструкции:

$$\mathcal{E} = \sum_i \|\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j\|^2$$

при ограничениях:

- $w_{ij} = 0$, если $\mathbf{x}_j$ не является ближайшим соседом $\mathbf{x}_i$;
- $\sum_j w_{ij} = 1$.

Эти веса отражают линейную зависимость точки $\mathbf{x}_i$ от её соседей.

LOCALLY LINEAR EMBEDDING (LLE)

3. Встраивание в низкоразмерное пространство:

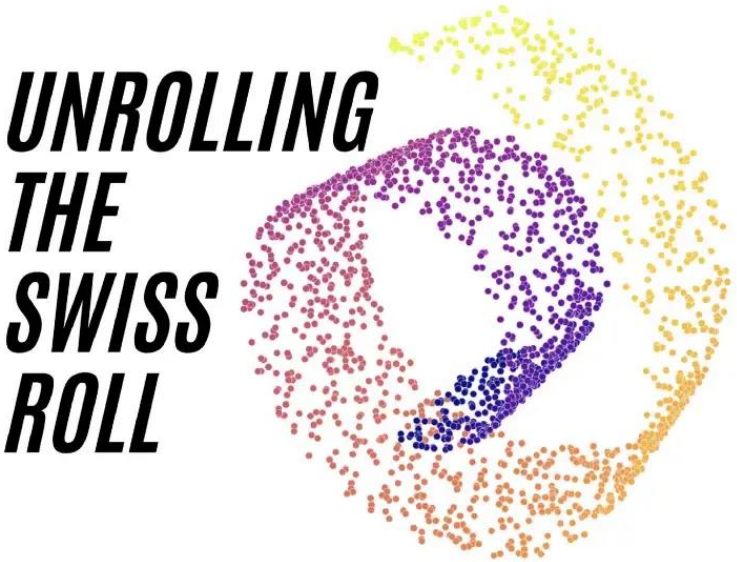
Теперь нужно найти координаты $\mathbf{y}_i$ в новом пространстве размерности d, которые минимизируют аналогичную ошибку реконструкции, но уже с использованием рассчитанных весов w_{ij}:

$$\Phi = \sum_i \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|^2$$

Это сводится к задаче поиска собственных векторов (решение задачи собственных чисел). Выбираются $d + 1$ наименьших собственных значений, исключая тривиальное решение, соответствующее нулевому собственному значению.

3. ISOMAP

- Isomap – комбинация нескольких алгоритмов для нелинейного снижения размерности, сохраняющего локальную структуру данных.



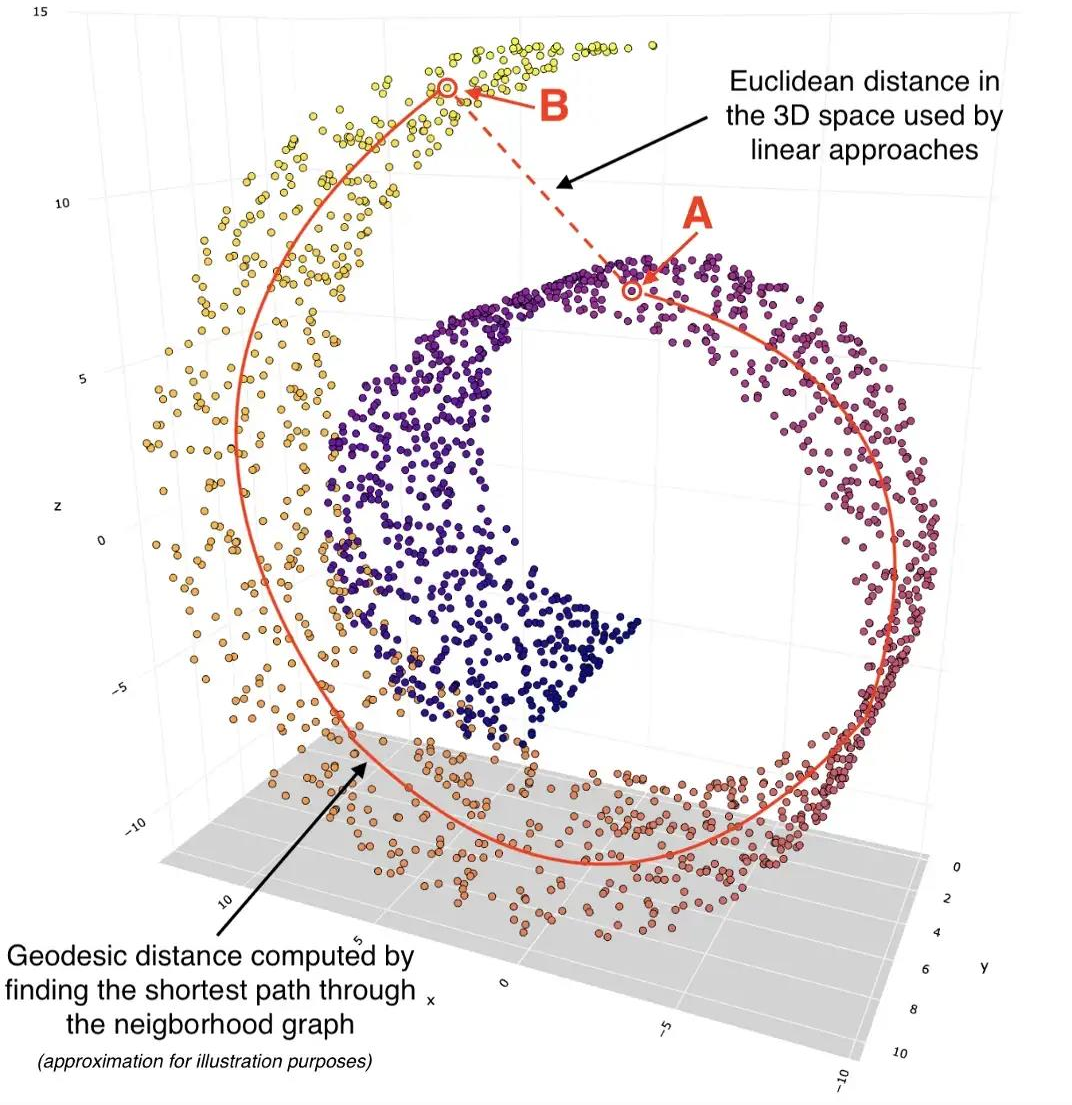
https://towardsdatascience.com/isomap-embedding-an-awesome-approach-to-non-linear-dimensionality-reduction-fc7efbca47a0

ISOMAP

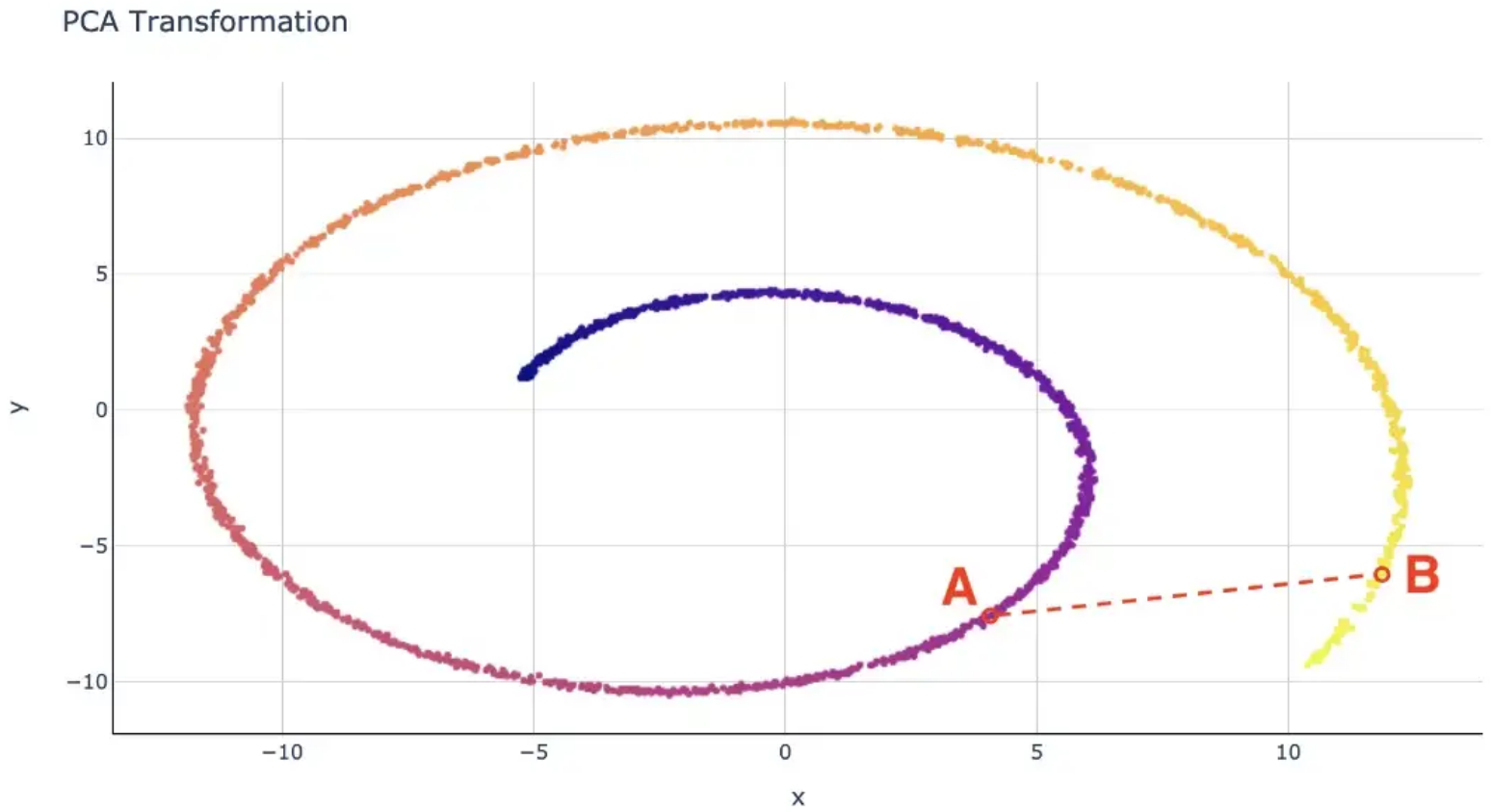
Алгоритм:

1) Используем KNN для вычисления расстояний между ближайшими соседями
2) Создаем граф, где только соседи соединены ребрами друг с другом
3) Вычисляем кратчайший путь между двумя точками по ребрам графа
4) Используем MDS для построения проекции в низкоразмерное пространство.

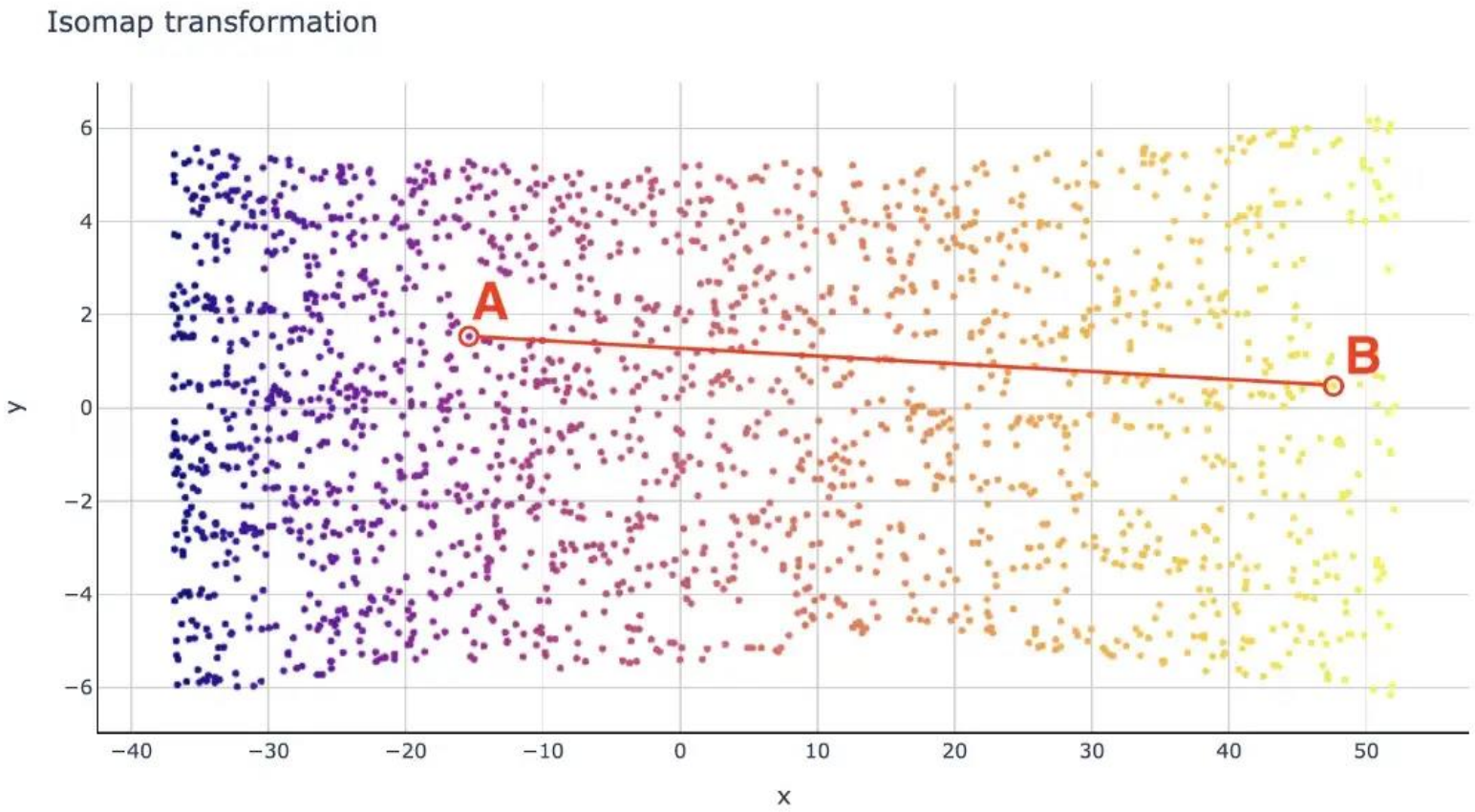
ISOMAP: ПРИМЕР



ISOMAP: ПРИМЕР



ISOMAP: ПРИМЕР



4. TSNE

t-SNE – t-distributed stochastic neighbor embedding

- *При проекции нам важно не сохранение расстояний между объектами, а сохранение пропорций:*

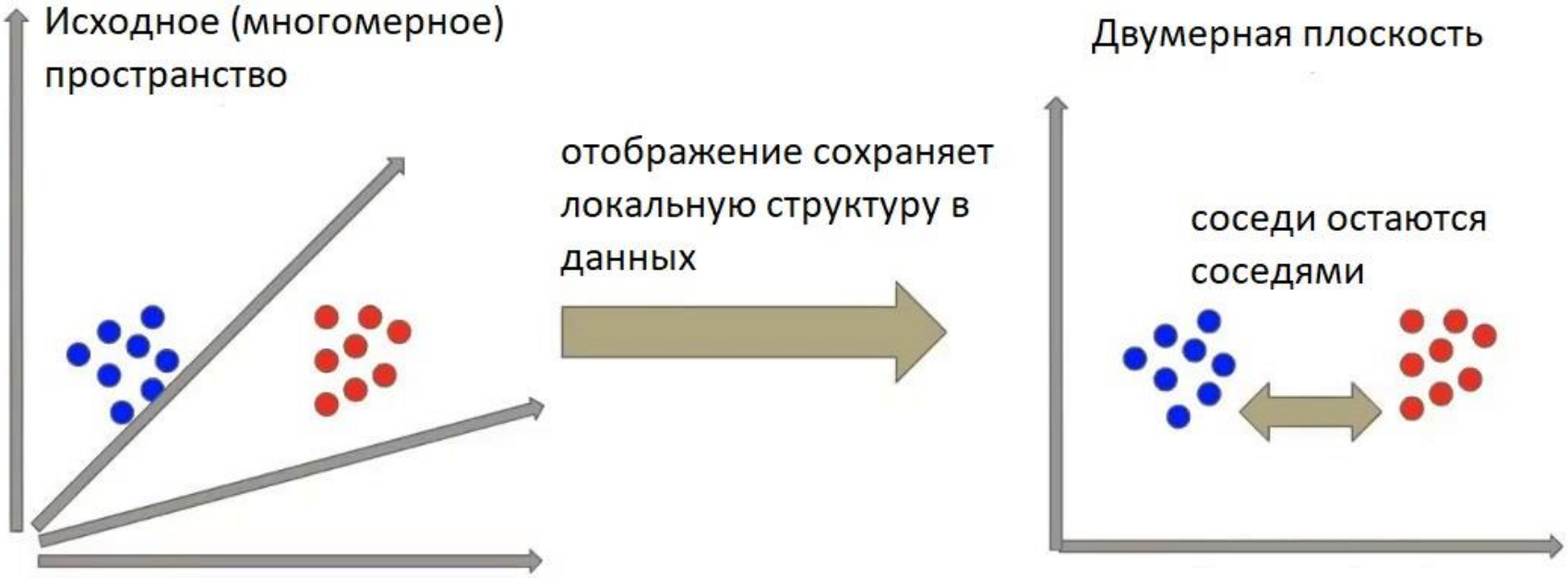
$$\rho(x_1, x_2) = \alpha\rho(x_1, x_3) \Rightarrow \rho(z_1, z_2) = \alpha\rho(z_1, z_3)$$

TSNE

t-SNE – t-distributed stochastic neighbor embedding

- *При проекции нам важно не сохранение расстояний между объектами, а сохранение пропорций:*

$$\rho(x_1, x_2) = \alpha\rho(x_1, x_3) \Rightarrow \rho(z_1, z_2) = \alpha\rho(z_1, z_3)$$

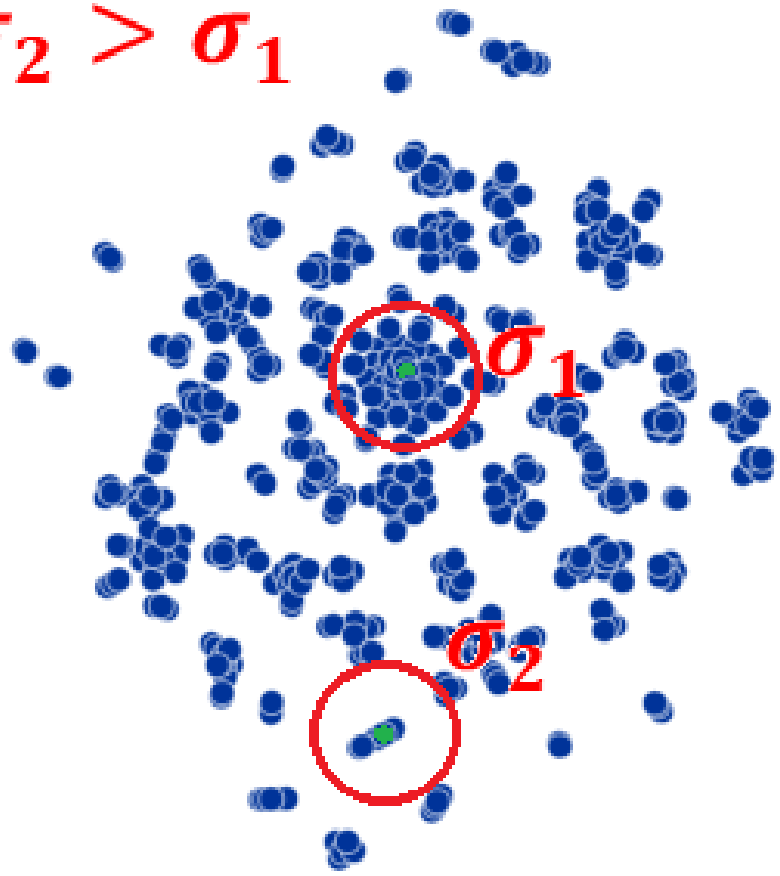


БЛИЗОСТЬ ОБЪЕКТОВ В ИСХОДНОМ ПРОСТРАНСТВЕ

$$p(i|j) = \frac{\exp(-\left||x_i - x_j|\right|^2 / 2\sigma_j^2)}{\sum_{k \neq j} \exp(-\left||x_k - x_j|\right|^2 / 2\sigma_j^2)}$$

(затем симметризуем $p(i|j)$)

- *объекты из окрестности x_j приближаются нормальным распределением*
- *чем кучнее объекты из этой окрестности, тем меньше берётся значение σ_j^2*



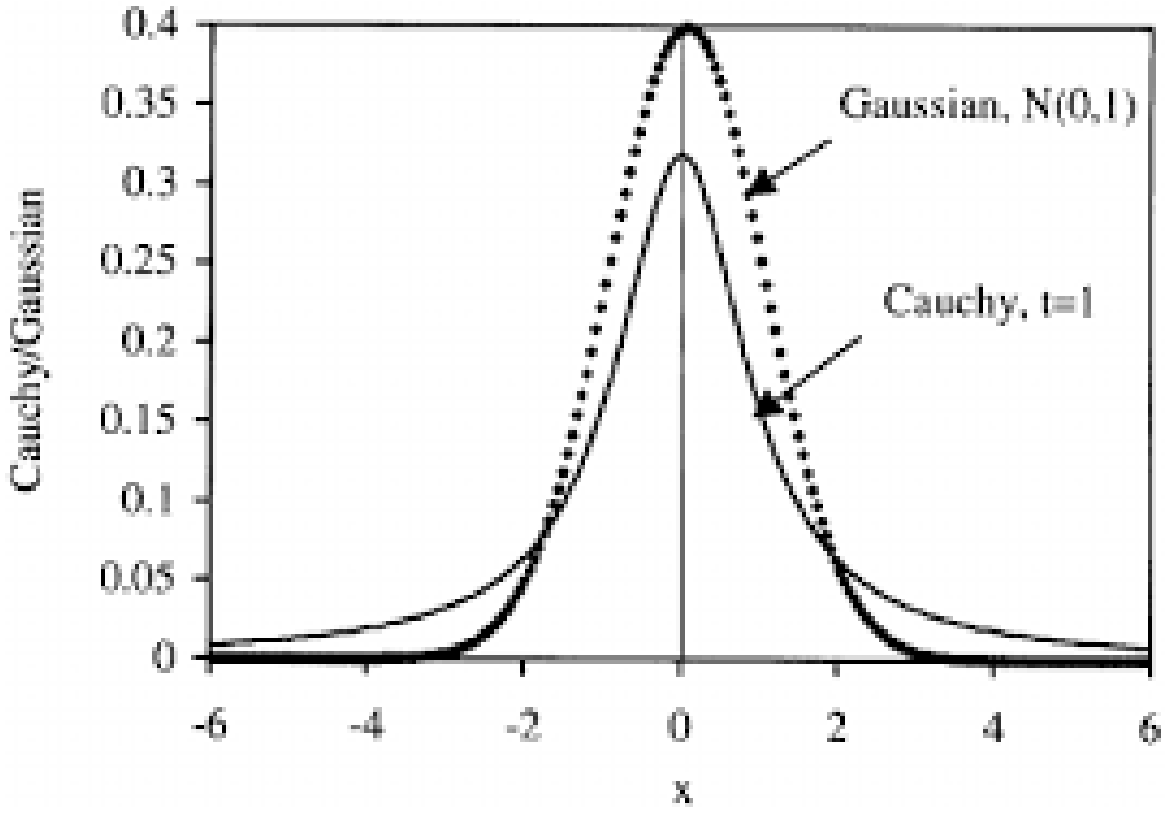
БЛИЗОСТЬ ОБЪЕКТОВ В НОВОМ ПРОСТРАНСТВЕ

- *В пространстве большой размерности можно разместить несколько объектов так, чтобы расстояния между ними были малы, но сохранить это свойство в низкоразмерном пространстве довольно сложно.*
- Будем измерять сходство объектов в новом пространстве с помощью распределения Коши, так как оно не так сильно штрафует за увеличение расстояний между объектами:

$$q_{ij} = \frac{\left(1 + \left|\left|z_i - z_j\right|\right|^2\right)^{-1}}{\sum_{k \neq j}\left(1 + \left|\left|z_k - z_j\right|\right|^2\right)^{-1}}$$

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ И РАСПРЕДЕЛЕНИЕ КОШИ

- Будем измерять сходство объектов в новом пространстве с помощью распределения Коши, так как оно не так сильно штрафует за увеличение расстояний между объектами:



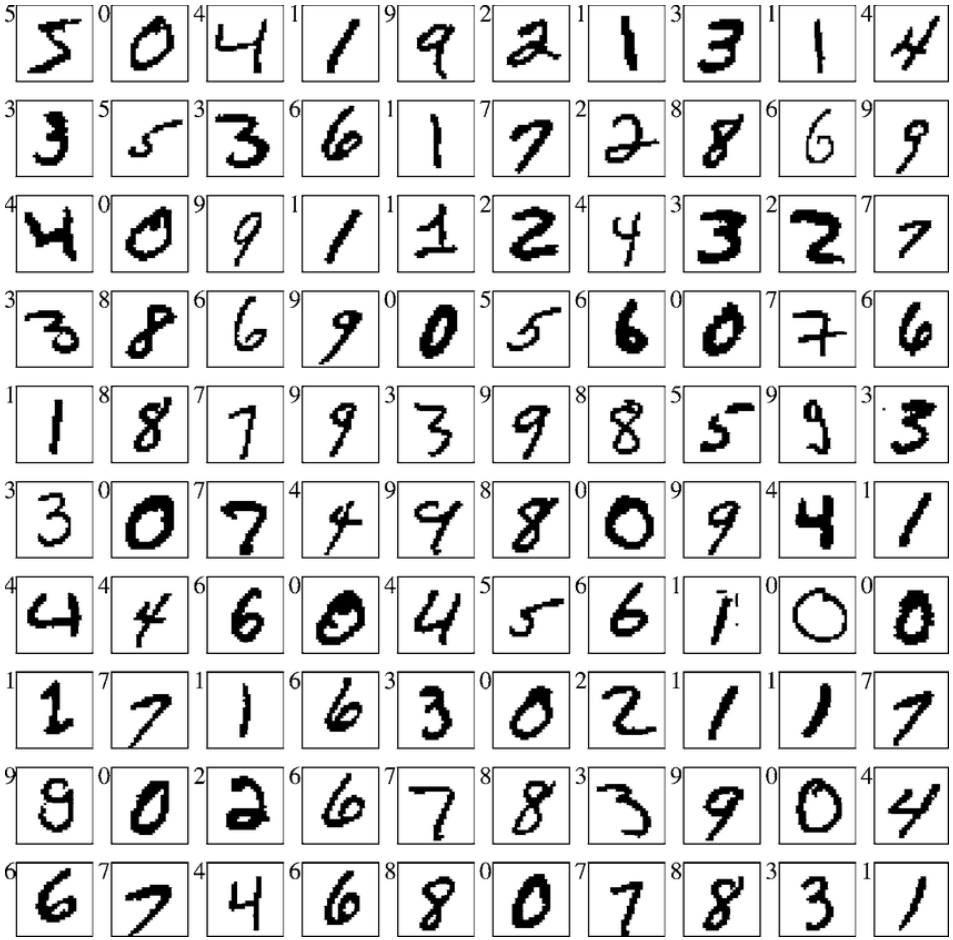
ОБУЧЕНИЕ TSNE

- Для построения проекций z_i объектов x_i будем минимизировать расстояние между исходным и полученным распределениями (минимизируем дивергенцию Кульбака-Лейблера).

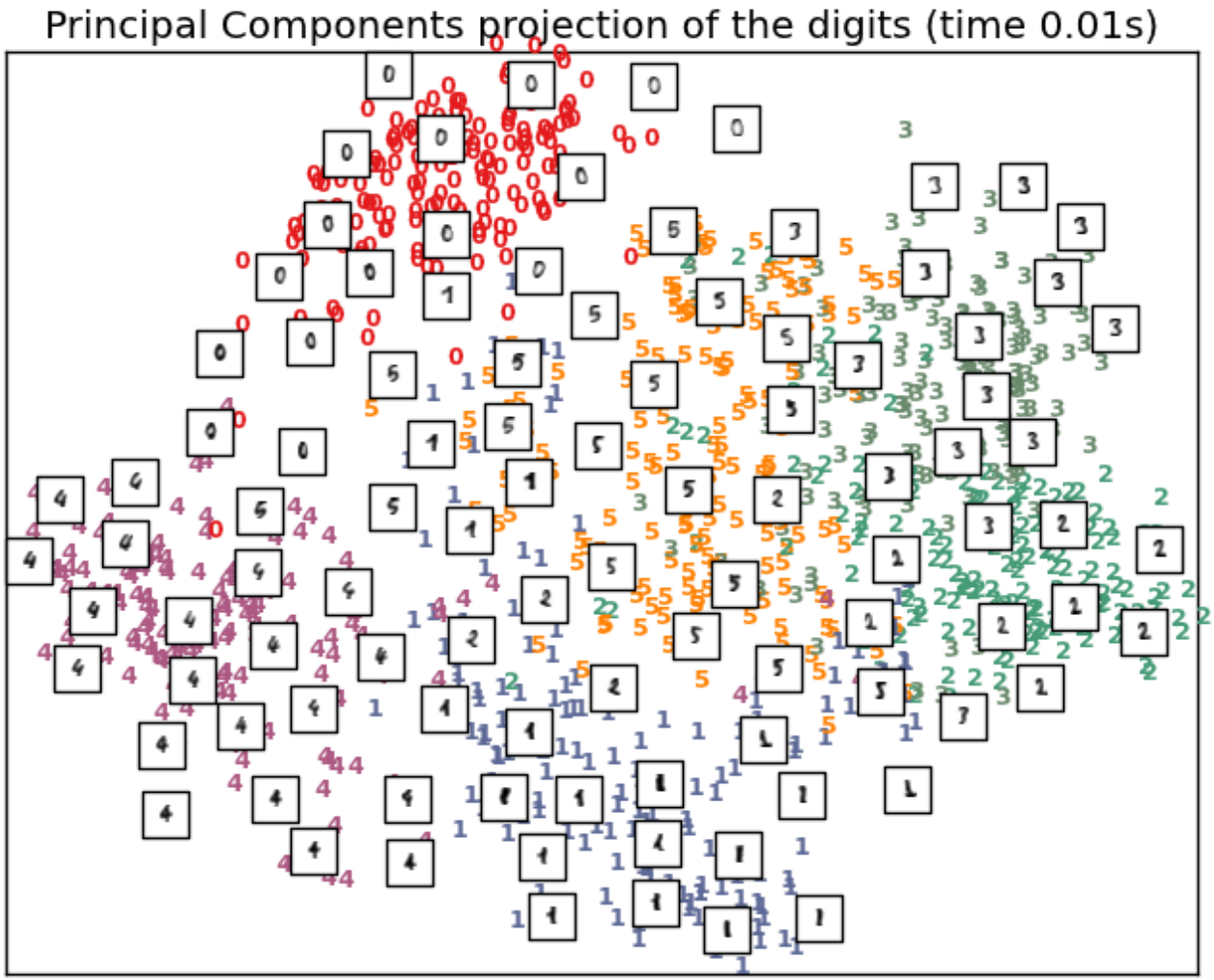
$$KL(p||q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \to \min_{z_1,\ldots,z_l}$$

TSNE (ПРИМЕР)

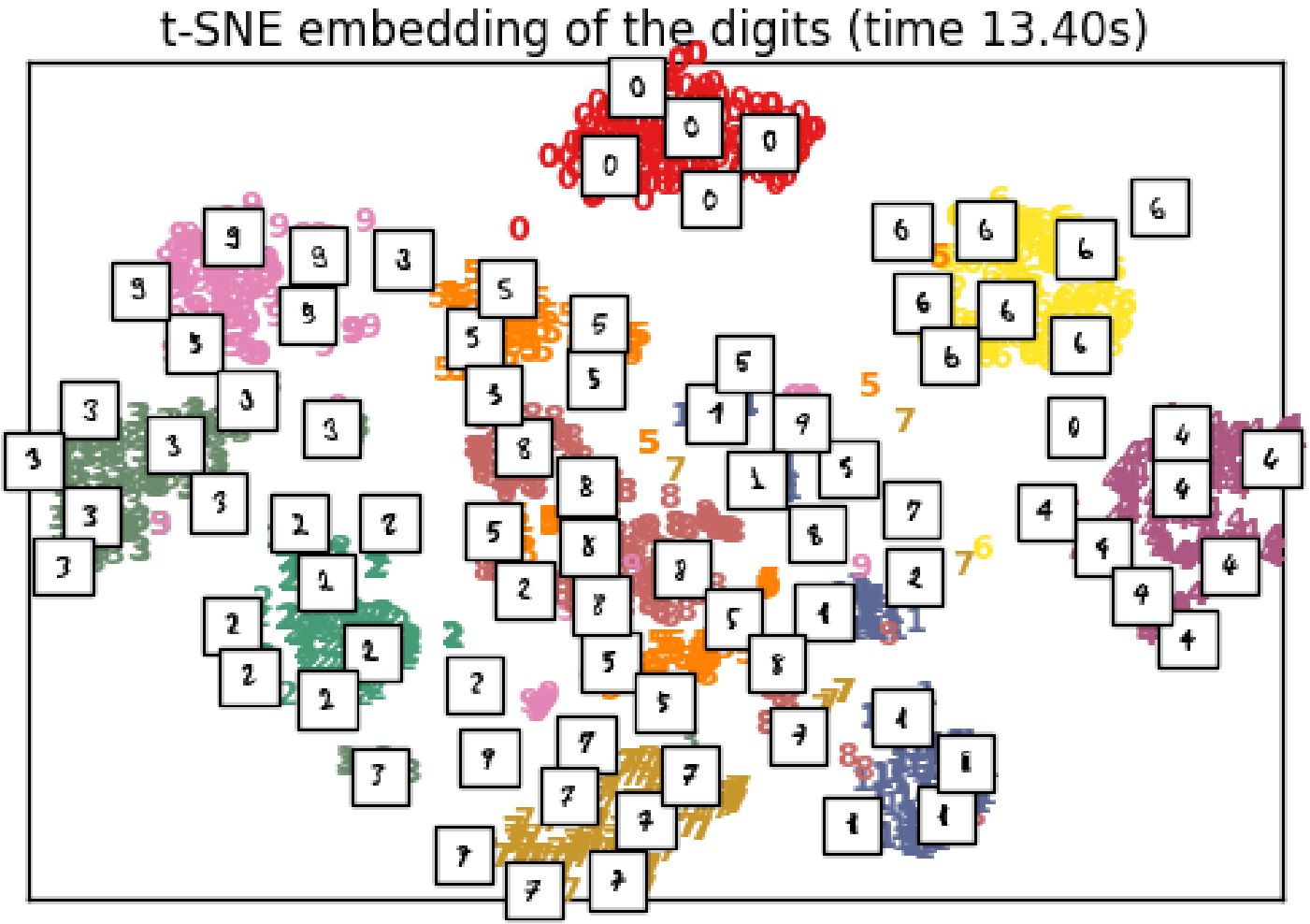
- MNIST – датасет из различных написаний десятичных цифр, где каждая картинка размера 28x28.



PCA (ПРИМЕР)



TSNE (ПРИМЕР)



ВИЗУАЛИЗАЦИЯ PCA И TSNE

http://projector.tensorflow.org/

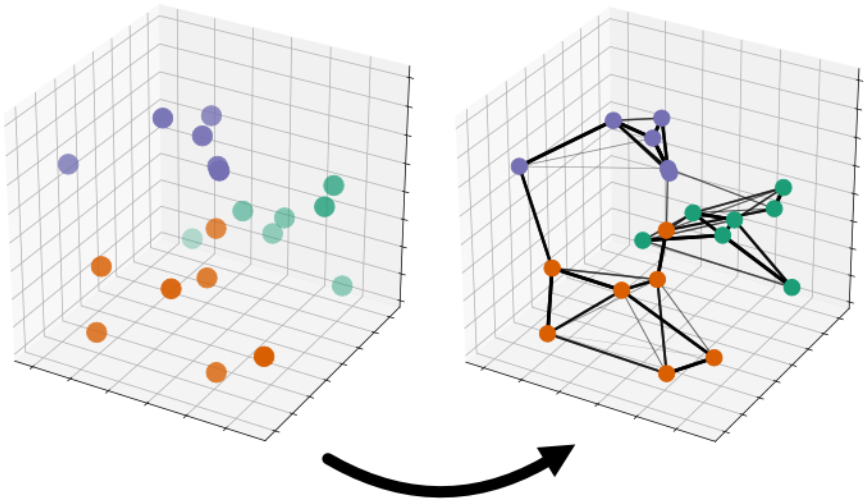
UMAP – БЫСТРАЯ АЛЬТЕРНАТИВА TSNE

- **Моделирование данных как графа:**
  1. Данные представляются в виде графа, где объекты являются вершинами, а связи ребра определяются на основе расстояния или меры близости.
  2. Для каждой точки рассчитываются k-ближайших соседей и ребром с точкой соединятся только ближайшие соседи
- **Построение вероятностного распределения:**

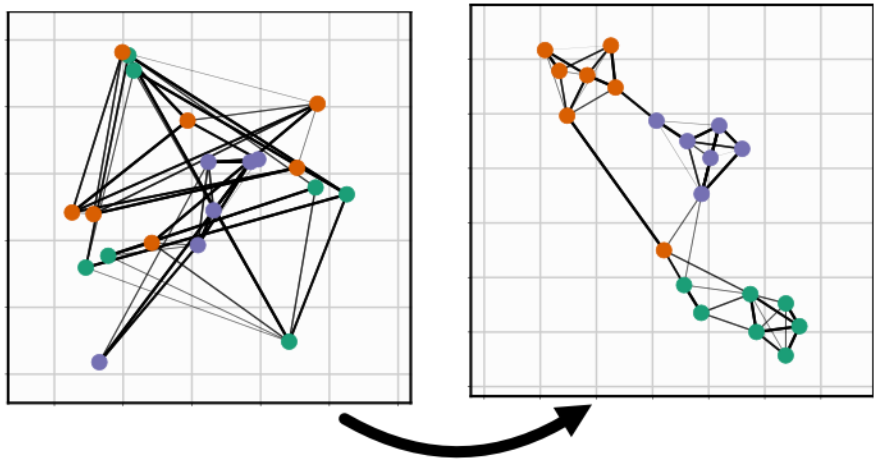
  На основе расстояний создается граф, где веса рёбер отражают вероятность соседства двух точек. Это описывает локальную структуру данных.
- **Оптимизация в низкомерном пространстве:**

  UMAP пытается минимизировать расхождение между графом в исходном высокоразмерном пространстве и графом, созданным в низкоразмерном пространстве (обычно 2D или 3D).

UMAP



Compute a graphical representation
of the dataset



Learn an embedding that preserves
the structure of the graph

UMAP – БЫСТРАЯ АЛЬТЕРНАТИВА TSNE

Пусть P_i – веса в исходном графе, Q_i – веса в графе в низкоразмерном пространстве.

UMAP минимизирует расхождение между G_{high} и G_{low}, измеряя их "похожесть" с помощью **двоичной кросс-энтропии**:

$$\mathcal{L}_{\text{UMAP}} = \sum_{i,j} \left(-P_{ij} \log Q_{ij} - (1 - P_{ij}) \log(1 - Q_{ij}) \right)$$

Каждая часть здесь связана с определённой структурой графа:

- **Первая часть** $-P_{ij} \log Q_{ij}$:
  - Наказывает модель, если в G_{low} точки, близкие в G_{high}, оказываются далеко.
  - Это стимулирует сохранение **локальной структуры** данных.
- **Вторая часть** $-(1 - P_{ij}) \log(1 - Q_{ij})$:
  - Наказывает модель, если в G_{low} точки, которые должны быть далёкими (малые P_{ij} в G_{high}), оказываются близко.
  - Это помогает поддерживать **глобальную структуру** данных.