Визуализация многомерных данных с помощью t-SNE

Целью уменьшения размерности является сохранение структуры многомерных данных в низкоразмерном пространстве. t-SNE (t-distributed stochastic neighbor embedding, стохастическое вложение соседей с t-распределением) относится к нелинейным методам уменьшения размерности для визуализации многомерных данных. Данный алгоритм относится к машинному обучению без учителя (unsupervised learning). Обучение без учителя означает отсутствие меток у данных, в отличии от обучения с учителем, где каждый объект имеет метку.

Исходные данные – это объекты, которые находятся в многомерном пространстве. Задача – представить их в 2-х или 3-х мерном пространстве. Объекты, которые похожи друг на друга и находятся рядом друг с другом в многомерном пространстве, должны иметь аналогичную структуру и в пространстве меньшей размерности.

t-SNE позволяет понять, на какое количество кластеров могут быть поделены данные, то есть этот метод не только визуализирует данные, но также и позволяет их исследовать.

Основой t-SNE является SNE, представленный Хинтоном и Ровейсом в 2002 году. t-SNE имеет два важных отличия от SNE 2002 года:

1) Используется симметричная версия SNE;
2) Вместо нормального распределения используется распределение Стьюдента для вычисления расстояния в пространстве низкой размерности.

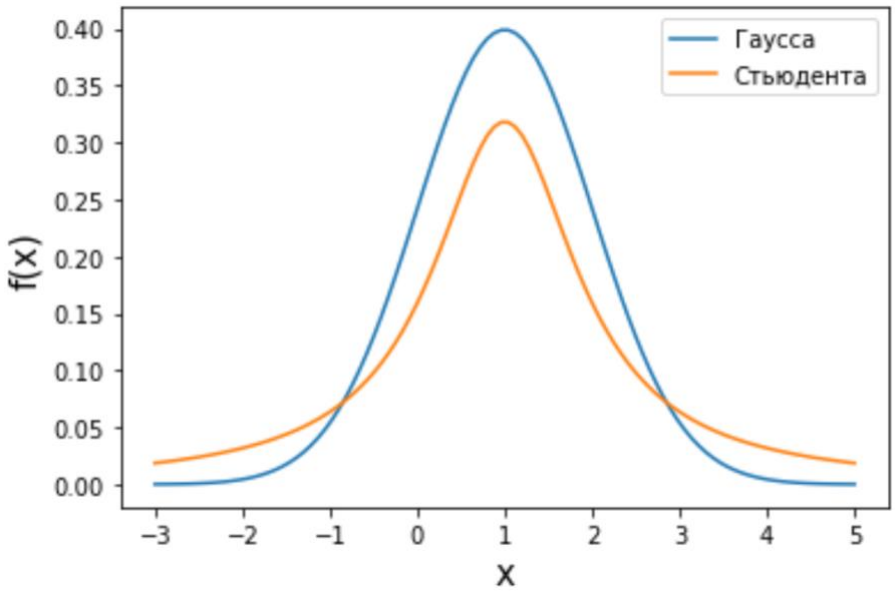


Рис. 1

В 2002 году появилось стохастическое вложение соседей (англ. Stochastic Neighbor Embedding, SNE) – это первая интерпретация изучаемого метода. Первый шаг алгоритма заключается в преобразовании многомерного евклидового расстояния между точками в условные вероятности, которые отражают сходство точек. Делается это по следующей формуле:

$$p(i|j) = \frac{e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}}{\sum_{k \neq i} e^{-\left(\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}}, \quad (1)$$

где $p(i|j)$ – это сходство двух объектов в пространстве высокой размерности;

$\|x_i - x_j\|^2$ – евклидово расстояние между двумя векторами;

σ_i^2 – дисперсия.

Норма 2, или Евклидова норма, или $\|x\|_2$ рассчитывается как $\sqrt{\sum_{i=1}^{n}(x_i)^2}$, где x_i – это координаты вектора).

Сумма всех p для одной точки равна 1. Значение $p(i|i)$ устанавливается равным нулю, так как необходима только попарная схожесть объектов. Если точки находятся близко друг к другу, то они считаются похожими и значение

будет относительно высоким. Для каждой точки в пространстве рассчитывается ее похожесть со всеми остальными точками.

Не существует единственного значения дисперсии σ_i^2, которая центрируется над каждой многомерной точкой x_i, для всех точек из набора данных. В плотных областях значение дисперсии меньше, чем для более разреженных областей. Дисперсии σ_i^2 для каждой точки выбирается адаптивно с учетом фиксированного значения перплексии, которое задается пользователем.

Параметр перплексии позволяет указать плотность точек для кластера (или же количество соседей). Перплексия описывает как далеко алгоритм собирается искать соседей в исходном многомерном пространстве, чтобы считать их принадлежащими одному и тому же множеству. Если установить это значение очень низким, то алгоритм будет пытаться разделить объекты на более мелкие кластеры в результирующем 2D-пространстве. Если значение слишком велико, то разные классы будут сгруппированы в одном кластере. Если параметр перплексии равен 30, то для 30 ближайших точек будут свои значения p, как только выходим за предел 30 соседей, все становится равным 0.

Если x_i – это выброс, то расстояние от нее до других точек будет очень большим, следовательно, $p(i|j)$ для такой точки будут чрезвычайно малы для всех j. В итоге получается, что функция стоимости будет почти нулевой для любого $q(i|j)$. Поэтому положение низкоразмерного аналога y_i определялось бы очень неточно и не было бы особой разницы в том, где он расположен. Для решения этой проблемы был создал симметричный SNE.

В симметричном SNE (Symmetric Stochastic Neighbor Embedding, Symmetric SNE) определяются совместные вероятности p_{ij} в многомерном пространстве как симметричные условные вероятности по формуле

$$p_{ij} = \frac{p(i|j) + p(j|i)}{2n}.$$

Это дает гарантию того, что

$$\sum_j p_{ij} > \frac{1}{2n}$$

для всех точек x_i. В результате этого преобразования каждая точка данных x_i вносит значительный вклад в функцию стоимости. Так же теперь $p_{ij} = p_{ji}$.

У точек x_i и x_j многомерного пространства есть аналоги y_i и y_j в пространстве низкой размерности. Для них также вероятность q_{ij}.

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

$q_{ij} = q_{ji}$, а q_{ii} устанавливается равным нулю. q_{ij} – это сходство двух объектов в пространстве низкой размерности.

Второе отличие t-SNE от SNE – это использование t-распределения, которое имеет более широкие «хвосты» (Рис. 1). Сделано это для того, чтобы решить проблему скученности в пространстве низкой размерности.

Если точки отображения y_i и y_j правильно моделируют сходство между точками данных высокой размерности x_i и x_j, условные вероятности p_{ij} и q_{ij} будут равны. t-SNE стремится найти низкоразмерное представление данных, которое минимизирует несоответствие между p_{ij} и q_{ij}.

Функция стоимости данного метода – это расстояние Кульбака-Лейблера, которое рассчитывается по формуле

$$\mathcal{L} = \sum_{i,j} p_{ij}\, log_2 \frac{p_{ij}}{q_{ij}}$$

Если все p равны всем q, то расстояние Кульбака-Лейблера (расстояние между плотностями распределения для каждой точки в пространстве) будет равно 0. Это то, к чему стремится алгоритм – минимизировать функцию стоимости с помощью метода градиентного спуска по отношению к y_i. Если p_{ij} велико, то нужно большое значение для q_{ij}. Если p_{ij} мало, то q_{ij} должно иметь тоже низкое значение для представления точек, которые находятся далеко друг от друга.

Градиент функции стоимости выглядит следующим образом:

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

Точки пространстве низкой размерности генерируются случайным образом. Затем, с помощью градиентного спуска, идет поиск оптимальных точек y_i, чтобы функция стоимости была минимальна. А будет она минимальна тогда, когда p_{ij} и q_{ij} будут равны, то есть структура данных в пространстве с высокой размерностью будет аналогична структуре данных в пространстве с низкой размерностью.

Рассмотрим использование t-SNE на примере набора данных, который содержит 16 признаков для описания животных из зоопарка, а также классы, к которым эти животные относятся: млекопитающие, птицы, рептилии, рыбы, амфибии, насекомые и беспозвоночные.

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
```

```
data = pd.read_csv('zoo.csv')
```

```
data
```

animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	class_type
turtle	0	0	1	0	0	1	0	0	1	1	0	0	4	1	1	1	3
chameleon	0	0	1	0	0	0	0	1	1	1	0	0	4	1	1	0	3
iguana	0	0	1	0	0	0	1	1	1	1	0	0	4	1	1	1	3
lizard	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	0	3
gecko	0	0	1	0	0	0	0	1	1	1	0	0	4	1	1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
vespa	0	0	1	0	1	0	1	0	0	1	1	0	6	0	0	0	6
bicho-pau	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	7
caracol-da-mata-atlantica	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
caranguejeira	1	0	1	0	0	0	1	0	0	1	1	0	8	0	0	0	7
sauva-limao	1	0	1	0	0	0	1	0	0	1	1	0	6	0	0	0	7

Перед тем, как данные визуализировать, удалим из датафрейма столбцы animal_name и class_type и проведем нормализацию данных.

```
D = data.drop(['class_type','animal_name'], axis = 1)
```

```
scaler = preprocessing.MinMaxScaler()
D = pd.DataFrame(scaler.fit_transform(D), columns = D.columns)
D
```

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize
0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	1.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	1.0
3	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
108	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.75	0.0	0.0	0.0
109	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.75	0.0	0.0	0.0
110	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.00	0.0	0.0	0.0
111	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.00	0.0	0.0	0.0
112	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.75	0.0	0.0	0.0

Далее представлен пример визуализации с использованием библиотеки sklearn.

```
T = TSNE(n_components=2,perplexity=25, random_state=123)
```

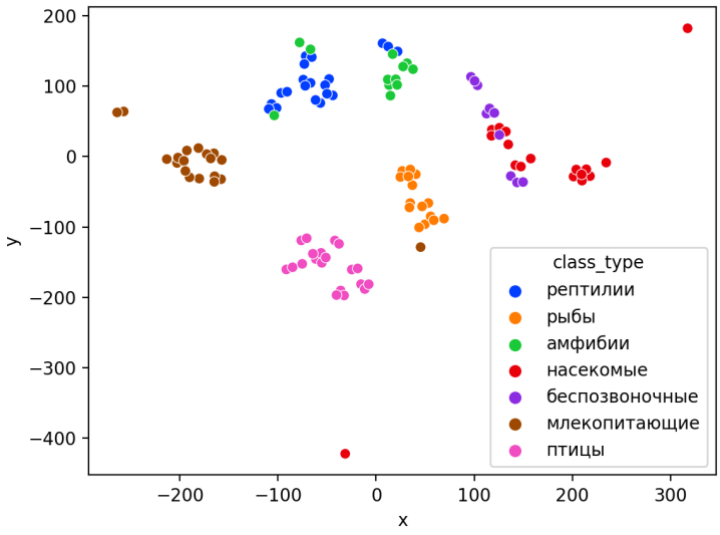
```
TSNE_features = T.fit_transform(D)
```

```
TSNE_features[1:4,:]
```

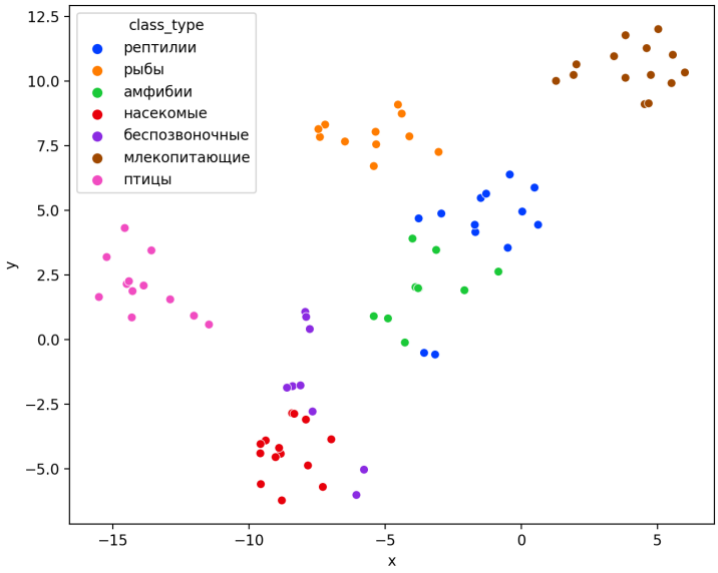
```
array([[-3.7687461,  4.692154 ],
       [ 0.619534 ,  4.4472136],
       [-1.4843574,  5.4836273]], dtype=float32)
```

```
DATA=D.copy()
DATA['x']=TSNE_features[:,0]
DATA['y']=TSNE_features[:,1]
```

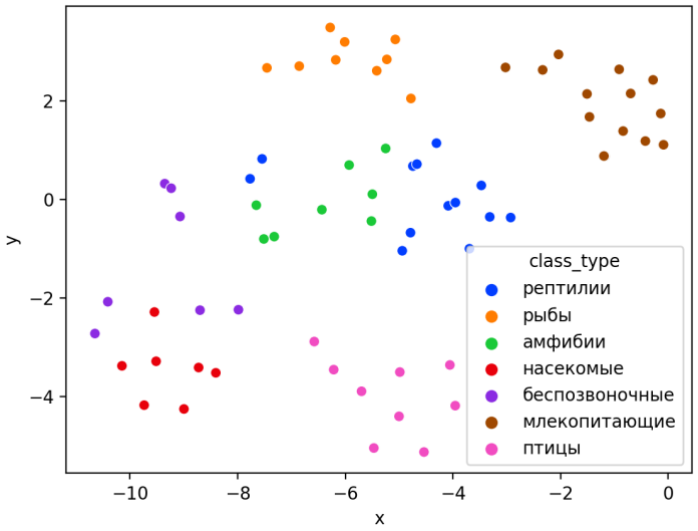
```
%matplotlib
fig = plt.figure()
sns.scatterplot(x='x',y='y',hue=data['class_type'],data=DATA, palette='bright')
plt.show()
```



perplexity=5



perplexity=25



perplexity=50

В результате визуализации видим, что явно выделены кластеры для млекопитающих, рыб и птиц. Насекомые находятся рядом с беспозвоночными, а амфибии с рептилиями.

Если добавить в исходный набор новые данные, то результат будет совершено иным, так как плотности распределений будут изменены и расстояние Кульбака-Лейблера, соответственно, тоже.

t-SNE – это случайный процесс, поэтому в зависимости от инициализации генератора случайных чисел получаются разные результаты, поэтому необходимо зафиксировать генератор. Сегменты, которые визуализируются с помощью данного алгоритма, могут немного перемешиваться друг с другом, это связано с тем, что объекты очень похожи

друг на друга и, соответственно, в многомерном пространстве расположены близко друг к другу.

Визуализация многомерных данных с помощью UMAP

UMAP (Uniform Manifold Approximation and Projection, равномерная аппроксимация многообразия) – алгоритм машинного обучения без учителя, который позволяет визуализировать многомерные данные, путем нелинейного снижения размерности. Лиланд Макиннес, Джон Хили и Джеймс Мелвилл, авторы алгоритма, считают, что UMAP намного быстрее и более вычислительно эффективный, чем t-SNE, а также лучше справляется с задачей переноса глобальной структуры данных в новое, уменьшенное пространство.

Алгоритм имеет достаточно сложное математическое обоснование, поэтому кратко рассмотрим принцип работы UMAP. UMAP состоит из двух этапов: построение многомерного графа и сопоставление ему графа из пространства низкой размерности.

Первый шаг – это расчет расстояния между всеми исследуемыми объектами с помощью некоторой метрики. Это может быть евклидово расстояние, манхэттенское расстояние или, например, косинусное расстояние, с помощью которого можно исследовать текстовые данные. Далее для каждого объекта определяется список его k ближайших соседей и строится взвешенный граф для соседей. Далее создается новый граф в пространстве низкой размерности и его ребра приближаются к исходному. Для этого минимизируется расстояние Кульбака-Лейблера для каждого ребра графа из исходного и низкоразмерного пространства.

Основные параметры, которые необходимо задать для работы алгоритма:

n_components – размерность итогового пространства (по умолчанию 2);

n_neighbors – количество соседей, которые рассматриваются для каждого объекта (по умолчанию 15). Если это значение мало, то будут рассматриваться самые ближайшие соседи и это приведет к большому

количеству кластеров, если же это значение велико, то будет хорошо сохранена глобальная структура, но более мелкие детали будут потеряны

min_dist – минимальное расстояние между точками в малых измерениях (по умолчанию 0.1), очень низкие значения приведут к более плотным кластерам, а высокие значения препятствуют объединению точек;

metric – это название формулы вычисления расстояния между точками. (по умолчанию euclidean).

Рассмотрим использование UMAP на том же примере.

```
D = data.drop(['class_type','animal_name'], axis = 1)
```

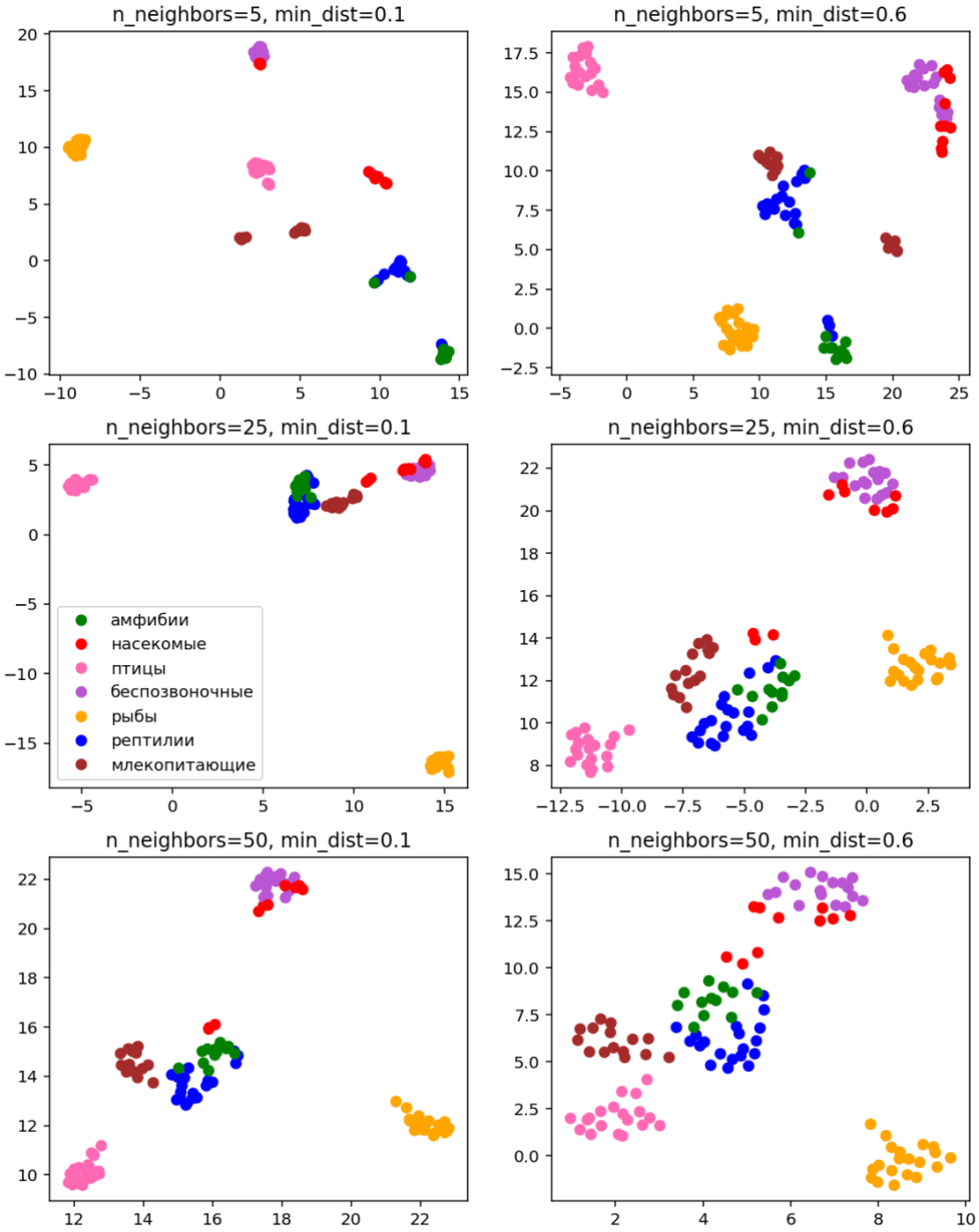
```
scaler = preprocessing.MinMaxScaler()
D = pd.DataFrame(scaler.fit_transform(D), columns = D.columns)
D
```

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize
0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	1.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	1.0
3	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.50	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
108	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.75	0.0	0.0	0.0
109	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.75	0.0	0.0	0.0
110	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.00	0.0	0.0	0.0
111	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.00	0.0	0.0	0.0
112	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.75	0.0	0.0	0.0

Далее представлен пример визуализации с использованием библиотеки umap-learn (для установки используйте в командной строке pip install umap-learn) для различных параметров.

```
n_n = (5,25,50) #n_neighbors
m_d = (0.1,0.6) #min_dist
```

```
um = dict()
for i in range(len(n_n)):
    for j in range(len(m_d)):
        um[(n_n[i],m_d[j])]=(umap.UMAP(n_neighbors=n_n[i], min_dist=m_d[j], random_state=123).fit_transform(DATA))
```



С помощью UMAP получили результат, очень похожий на то, что дает t-SNE.

Практическое задание

Выполнить визуализацию многомерных данных, используя t-SNE. Необходимо использовать набор данных MNIST или fashion MNIST (можно использовать и другие готовые наборы данных, где можно наблюдать разделение объектов по кластерам). Рассмотреть результаты визуализации для разных значений перплексии.

Выполнить визуализацию многомерных данных, используя UMAP с различными параметрами n_neighbors и min_dist. Рассчитать время работы алгоритма с помощью библиотеки time и сравнить его с временем работы t-SNE.

Оформить отчет о проделанной работе. Отчет должен содержать результаты визуализации для разных значений параметров и выводы.