

Эмбеддинги, Word2Vec, визуализация в UMAP



Подготовила
Ольга Филимонова, гр. 331

Как обрабатывать текст

Чтобы подать текстовые данные на вход модели машинного обучения (в частности, нейросети), нужно каким-то образом представить текст в виде числовой последовательности. Как это можно сделать?

Заметим, что текст — это последовательность токенов. Токенами могут выступать:

- - буквы;
- - части слов;
- - слова;
- - словосочетания
- - предложения;
- - ...

Получается, задачу кодирования текста можно свести к задаче кодирования токенов. Тогда текст представляется последовательностью токенов, а кодировка текста собирается из последовательности закодированных токенов.



Векторные представления токенов и текстов



Кодирование токенов:
One-hot encoding



Кодирование текста:
Bag of words (BoW)

Словарь

1.	cat
2.	dog
3.	...
i.	mother
...	

Размер каждого
вектора = n (50.000)

cat = [1, 0, 0, ..., 0]
dog = [0, 1, 0, ..., 0]
mother = [0, 0, 0, ..., 1, ..., 0]

i-я координата

Словарь

1.	a
2.	and
...	
14.	are
...	
145.	cat
...	
257.	dog
...	
678.	is
...	
1537.	sleeping


1. a cat and a dog are sleeping
2. a dog is walking

BoW для этих предложений:

1. [2, 1, 0, ..., 1, 0, ..., 0, 1, 0, ..., 1, 0, ..., 0, ..., 1, ... 0]
2. [1, 0, 0, ..., 0, 0, ..., 0, 0, 0, ..., 1, 0, ..., 1, ..., 0, ... 0]
↑ ↑ ↑ ↑ ↑ ↑
1 2 14 145 257 678 1537

*Также популярны Tf-Idf и LSA (латентный семантический анализ)

Контекстные эмбединги слов. Word2Vec

Вова _____ сессию
Михаил Юрьевич _____ Вову 

Ride _____(bicycle)
Bicycle _____(frame)

Вектор слова — строка матрицы

	a	horse	ride	bicycle	frame	rose
a		256	45	230	90	134
horse	256		137	4	2	5
ride	45	137		120	34	3
bicycle	230	4	120		76	2
frame	90	2	34	76		0
rose	134	5	3	2	0	

a		256	45	230	90	134
horse	256		137	4	2	5
ride	45	137		120	34	3
bicycle	230	4	120		76	2
frame	90	2	34	76		0
rose	134	5	3	2	0	

50.000

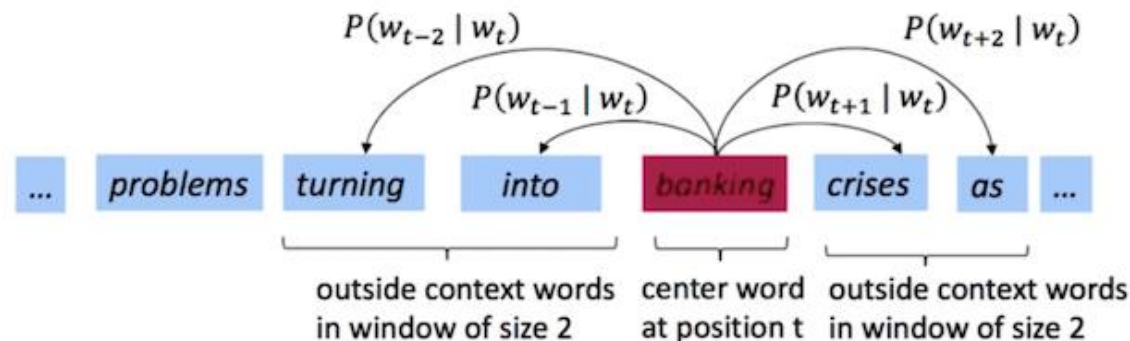
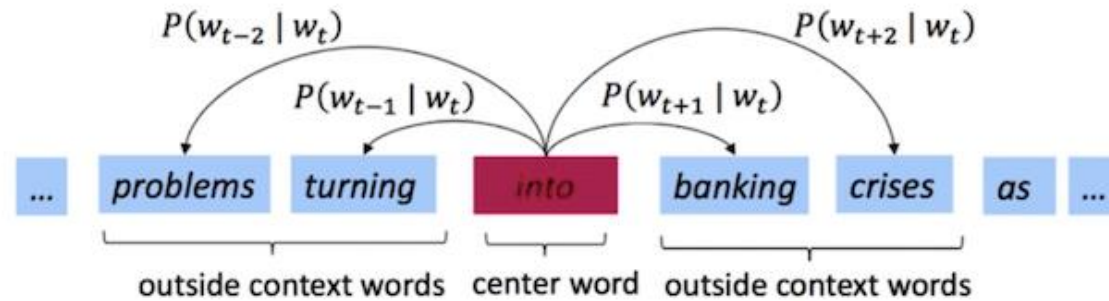
a	45	56	41	134	-67	14
horse	25	32	10	67	1	0
ride	43	-17	-4	-10	49	32
bicycle	23	21	10	56	6	-2
frame	9	2	34	6	45	0
rose	54	59	-31	12	-45	9

500

метод понижения размерности
(PCA, TSNE)

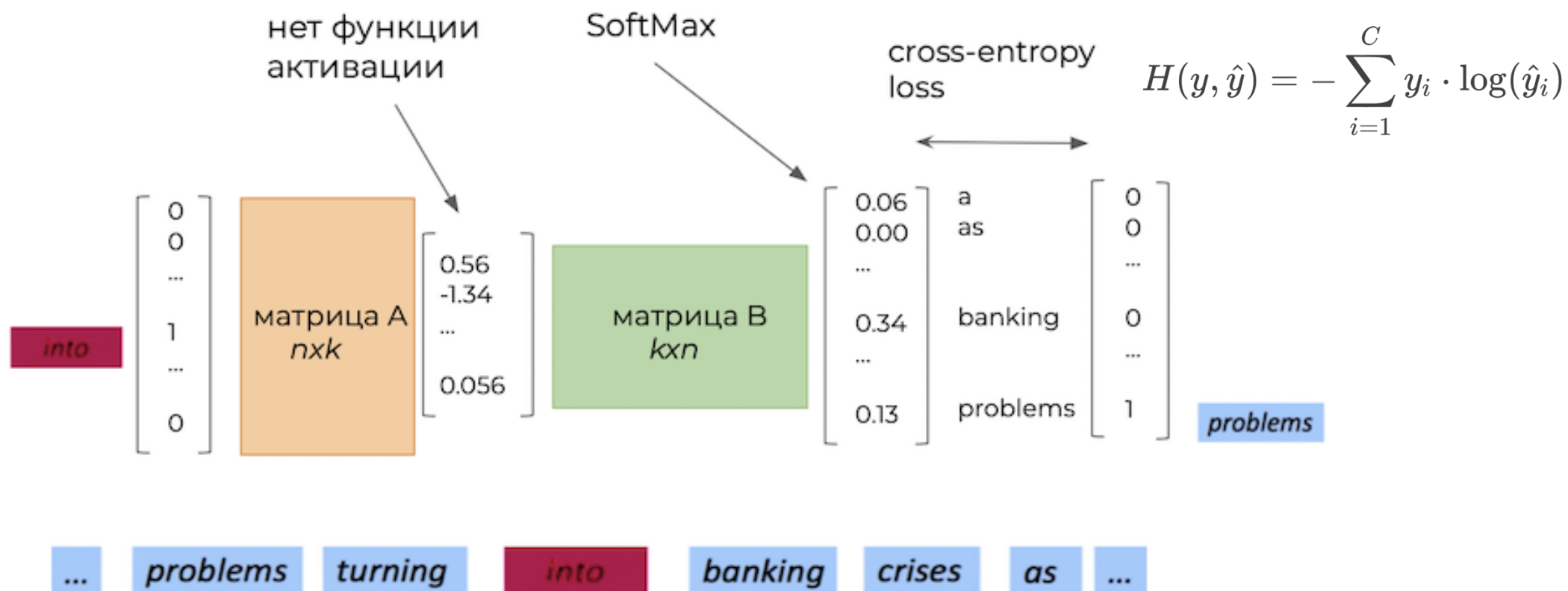
Word2Vec

- Хотим выучить векторы слов небольшой размерности, которые отражали бы смысл слов: их можно было бы сравнивать между собой с помощью некой метрики.
- Такие выученные векторы мы будем называть **эмбедингами слов**.
- Как выучивать такие векторы: мы будем учить нейросеть по центральному слову скользящего окна предсказывать слова, которые могут находиться в контексте (стоять вокруг этого слова).

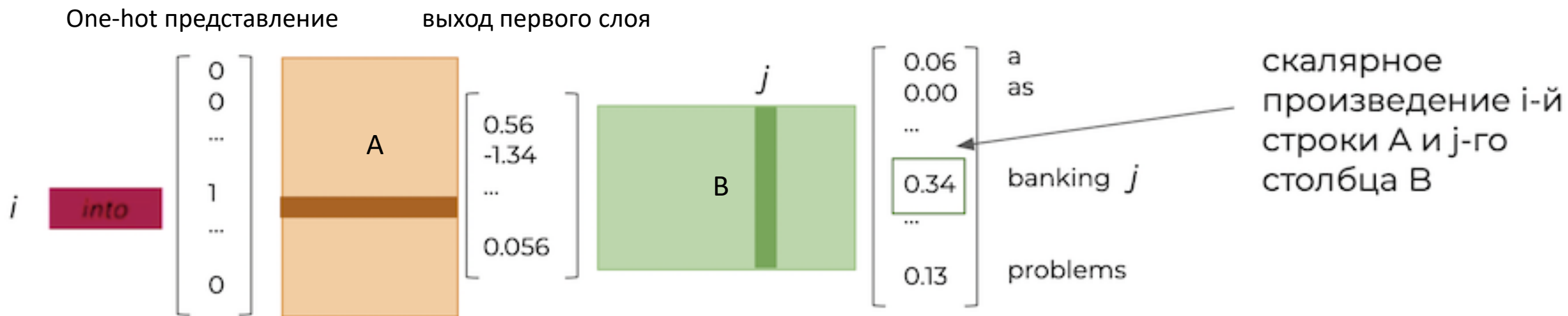


Word2Vec

- Ставится задача **классификации**. Количество классов — размер словаря n .
- На вход нейросеть принимает слово, выдает n значений — распределение на слова в словаре.
- Лосс-функция — кросс-энтропия между распределением, выданным сетью, и верным распределением (one-hot вектором)



Word2Vec

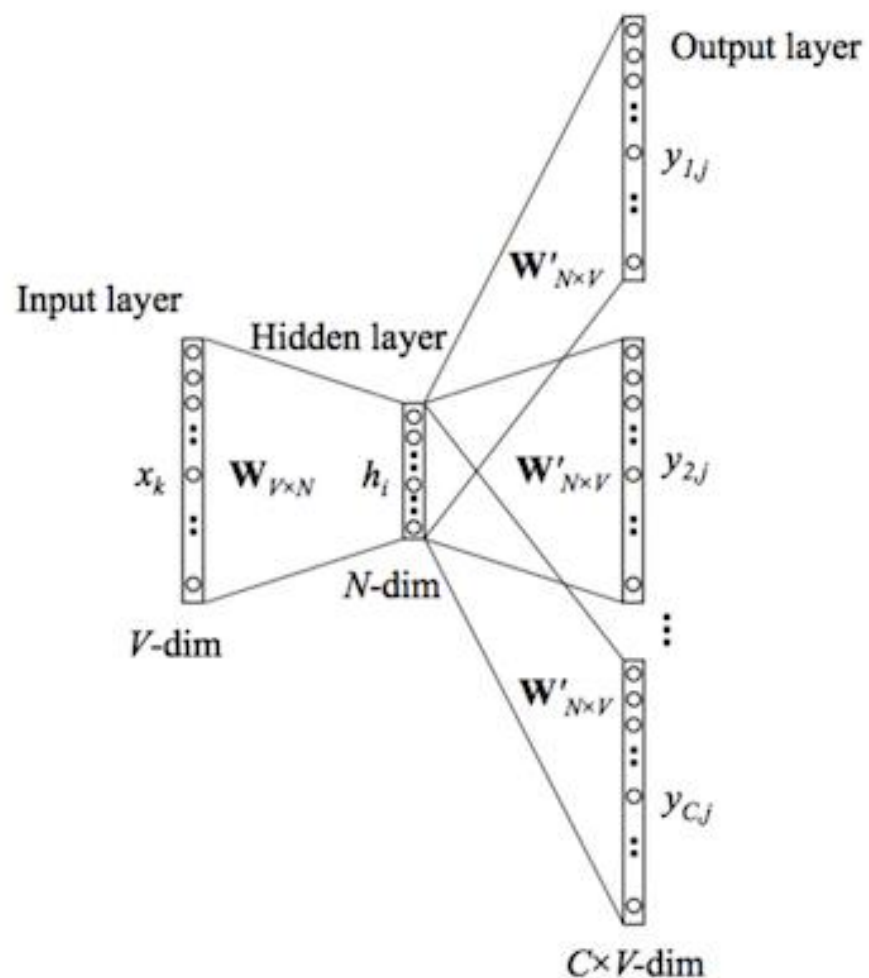


- Нейросеть в процессе обучения выучивает такие векторы слов, чтобы скалярное произведение между векторами двух слов, которые часто используются в контексте, было как можно больше.
- Таким образом, в качестве эмбедингов слов можно взять строки матрицы A или столбцы матрицы B.

Варианты Word2Vec

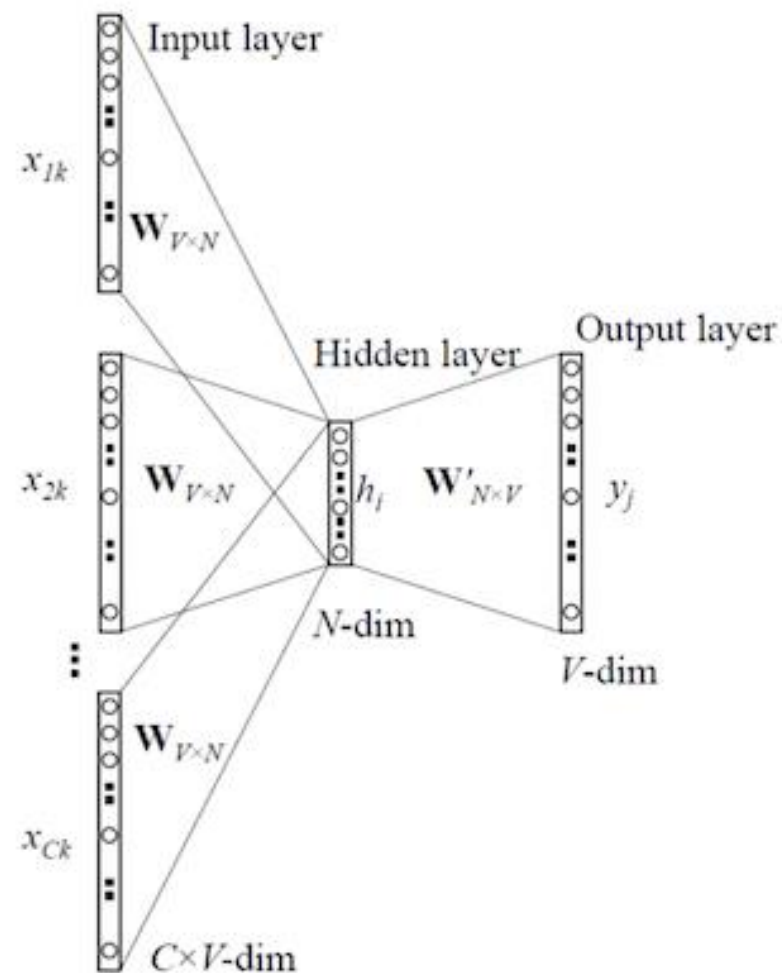
Предсказание контекста по центральному слову

Skip-gram



Предсказание центрального слова по контексту

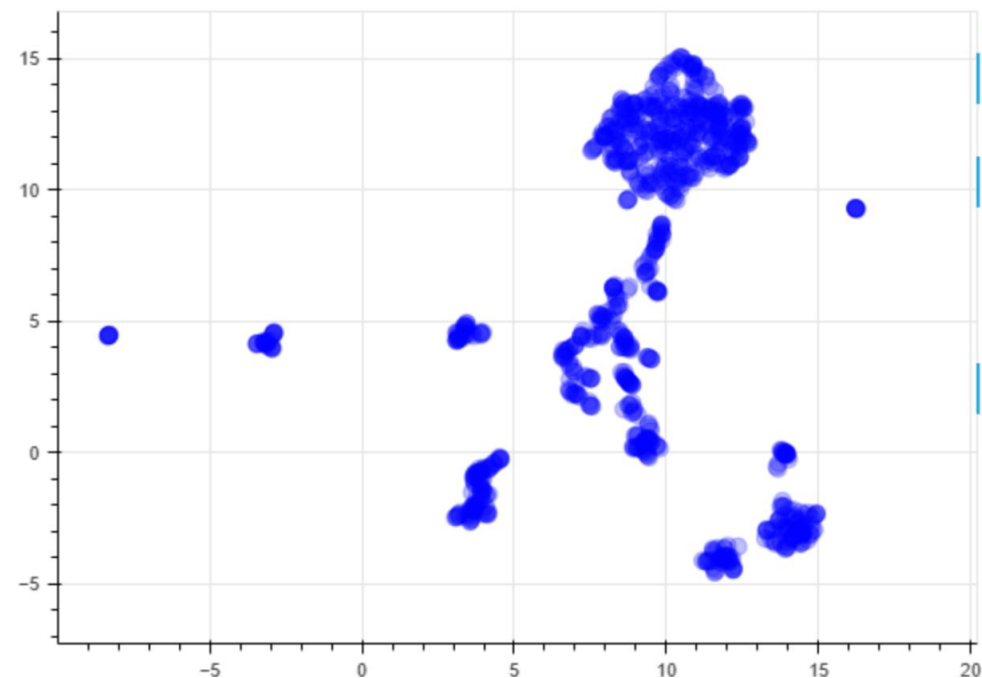
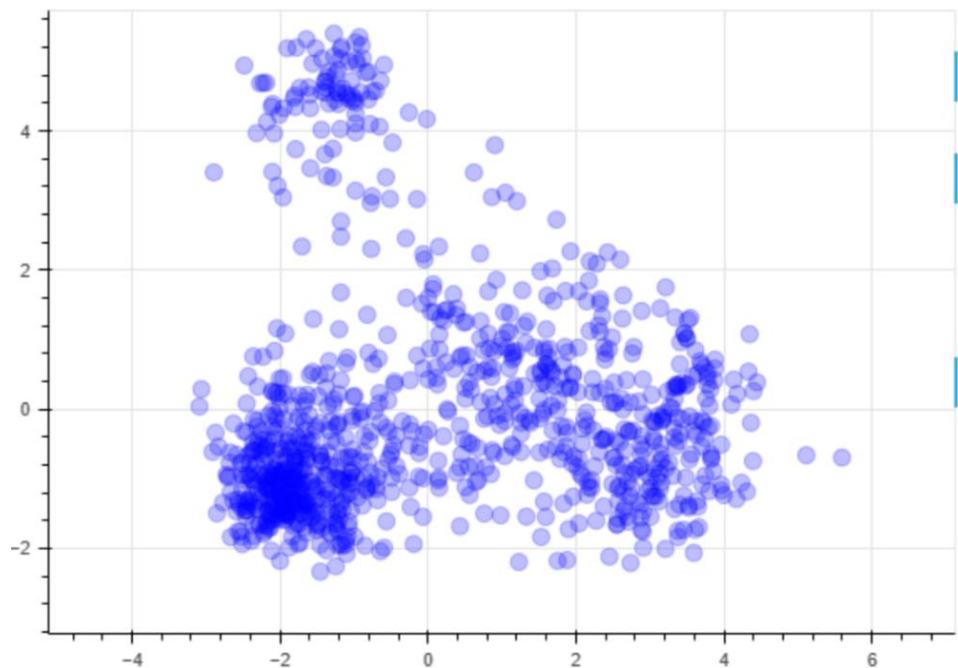
CBOW



Принцип работы UMAP (Uniform Manifold Approximation and Projection)

UMAP — это алгоритм нелинейного снижения размерности и визуализации данных, который сохраняет как локальную, так и глобальную структуру данных. Он работает в два этапа:

1. Построение графа в высокоразмерном пространстве (анализ локальных связей).
2. Оптимизация низкоразмерного представления (сохранение структуры графа).



Построение графа в исходном многомерном пространстве

Для каждой пары точек (i, j) определяется **вероятность сходства** в высокоразмерном пространстве:

$$p_{i|j} = \exp \left(-\frac{d(x_i, x_j) - \rho_i}{\sigma_i} \right),$$

где:

$p_{i|j}$ — это вероятность того, что точка x_i является ближайшим соседом точки x_j **при условии**, что x_j уже выбрана в качестве опорной точки.

- $d(x_i, x_j)$ — расстояние между точками.
- ρ_i — расстояние до ближайшего соседа (гарантирует локальную связность).
- σ_i — параметр, подбираемый так, чтобы $\sum_j p_{i|j} = \log_2(n_neighbors)$.

Получаем итоговую матрицу сходств:

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} \cdot p_{j|i}.$$

Имеем граф:

- Точки — вершины графа.
- Рёбра — сходства p_{ij} (чем выше p_{ij} , тем сильнее связь).



Переход в низкоразмерное пр-во и оптимизация

UMAP минимизирует **кросс-энтропию** между двумя распределениями:

- p_{ij} (сходство в высокоразмерном пространстве).
- q_{ij} (сходство в низкоразмерном пространстве):

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

где a, b — параметры, подобранные эвристически.

Функция потерь:

$$\mathcal{L} = \sum_{i,j} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + (1 - p_{ij}) \log \left(\frac{1 - p_{ij}}{1 - q_{ij}} \right).$$



Перейдем в ноутбук!

