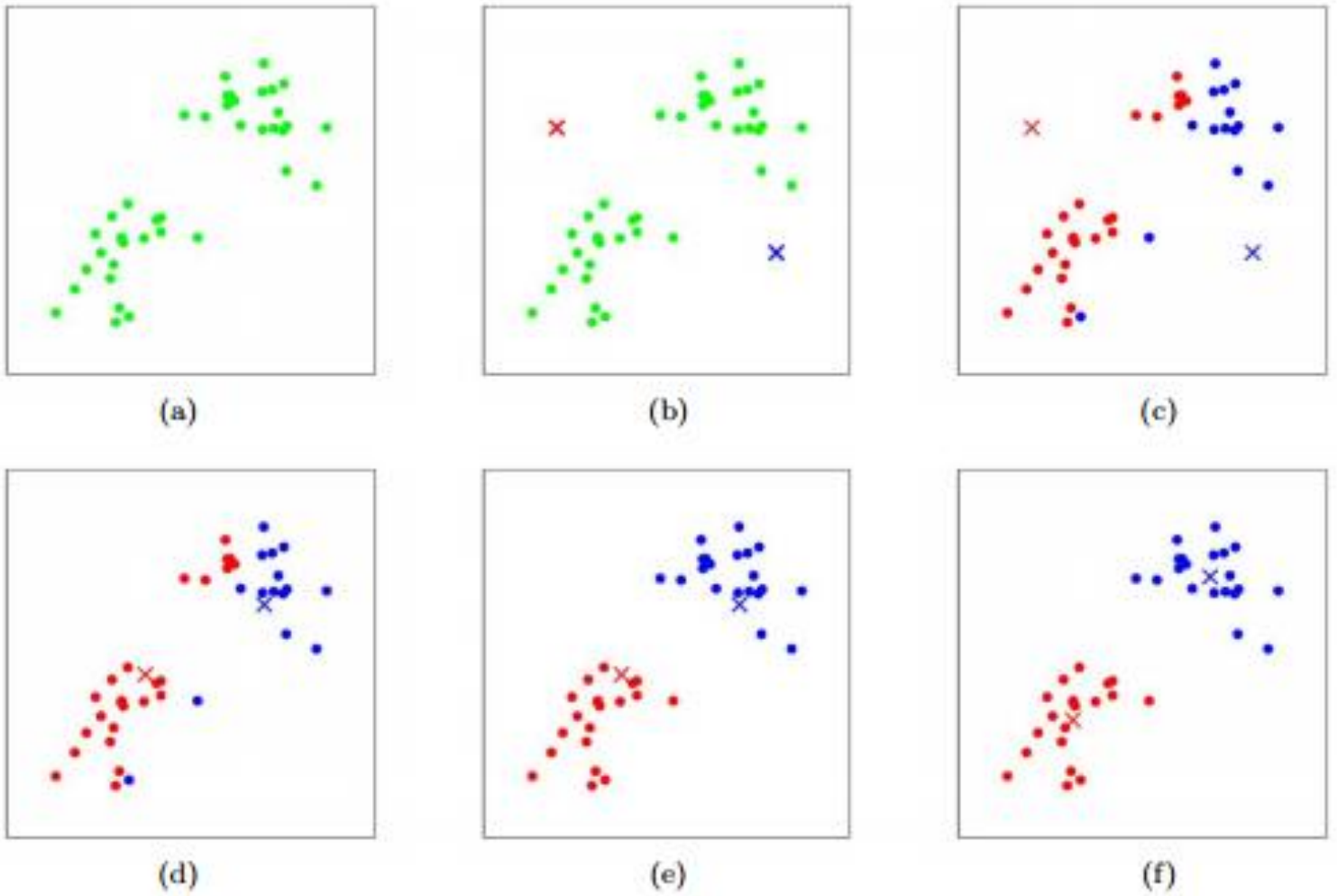
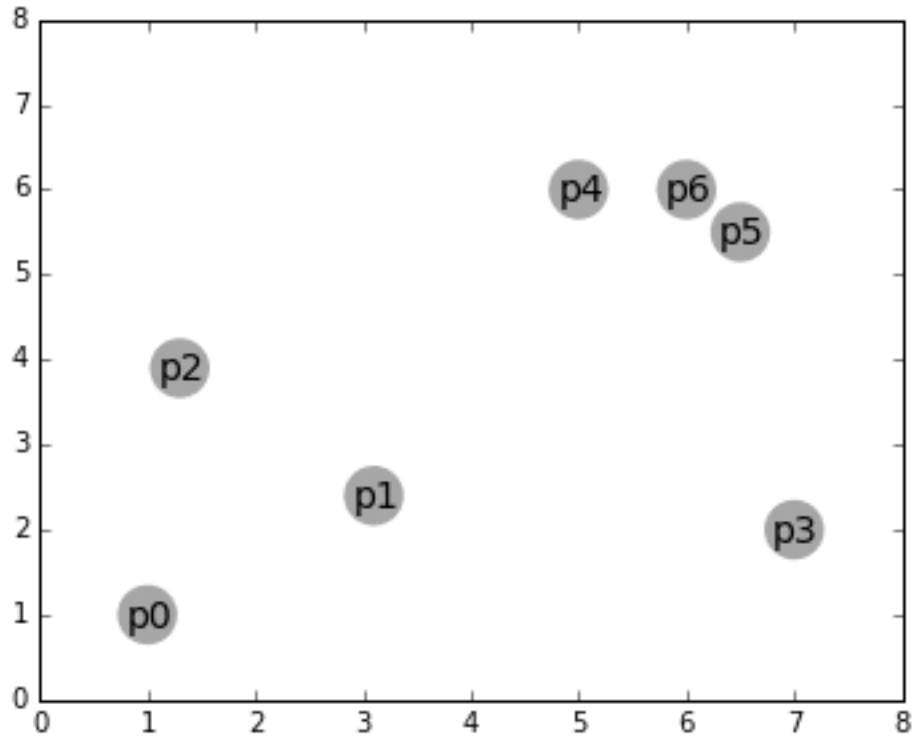
Обучение без учителя

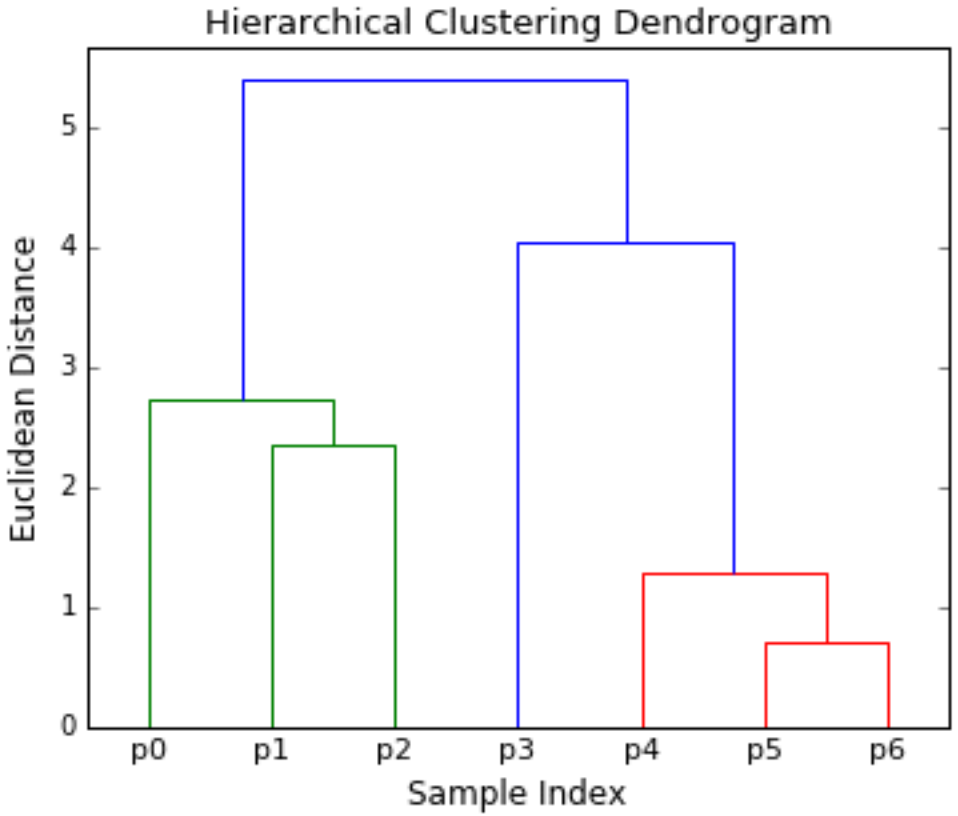
Лекция 8

Повторение. K-means

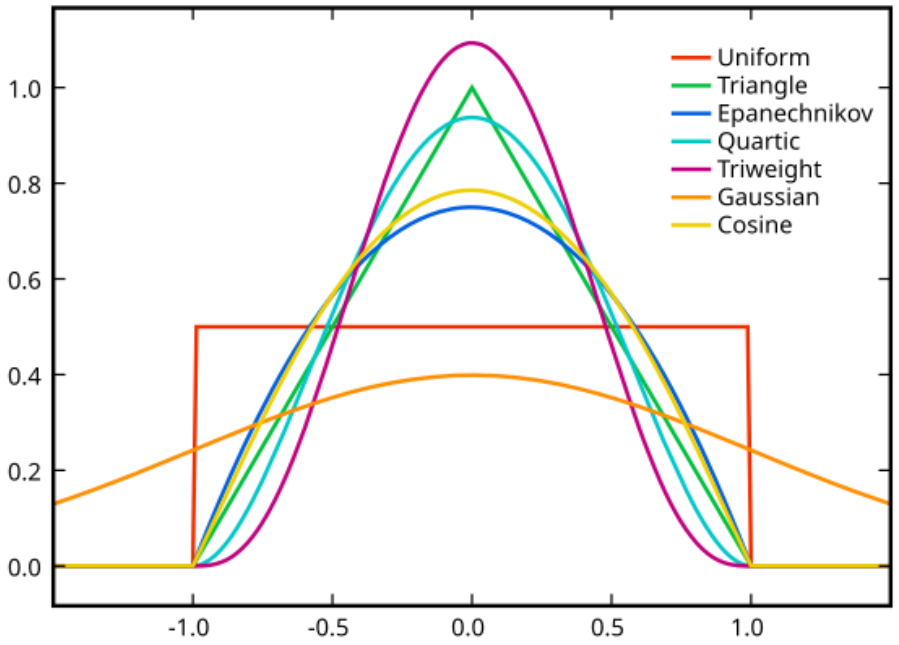


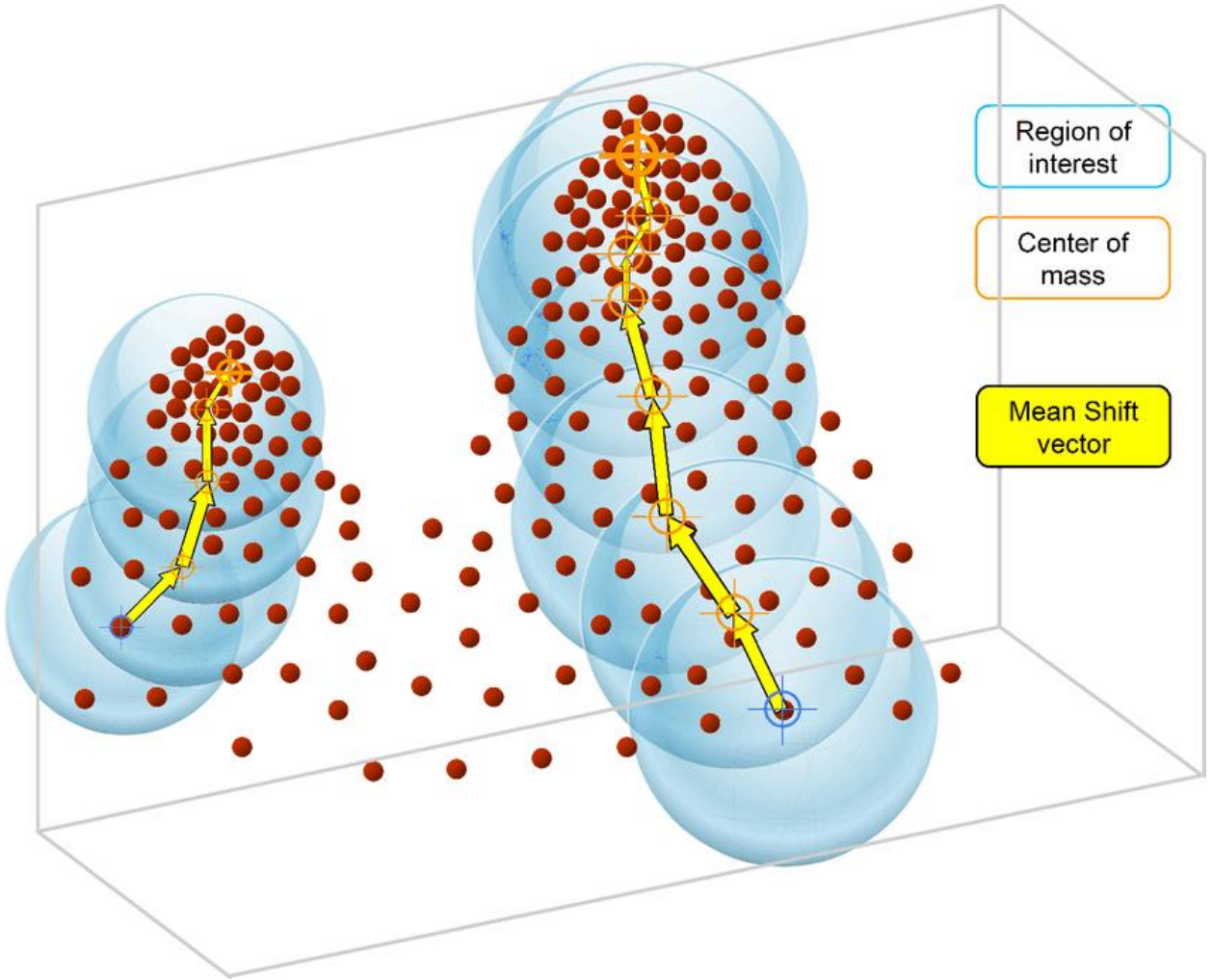
Повторение. Иерархическая кластеризация



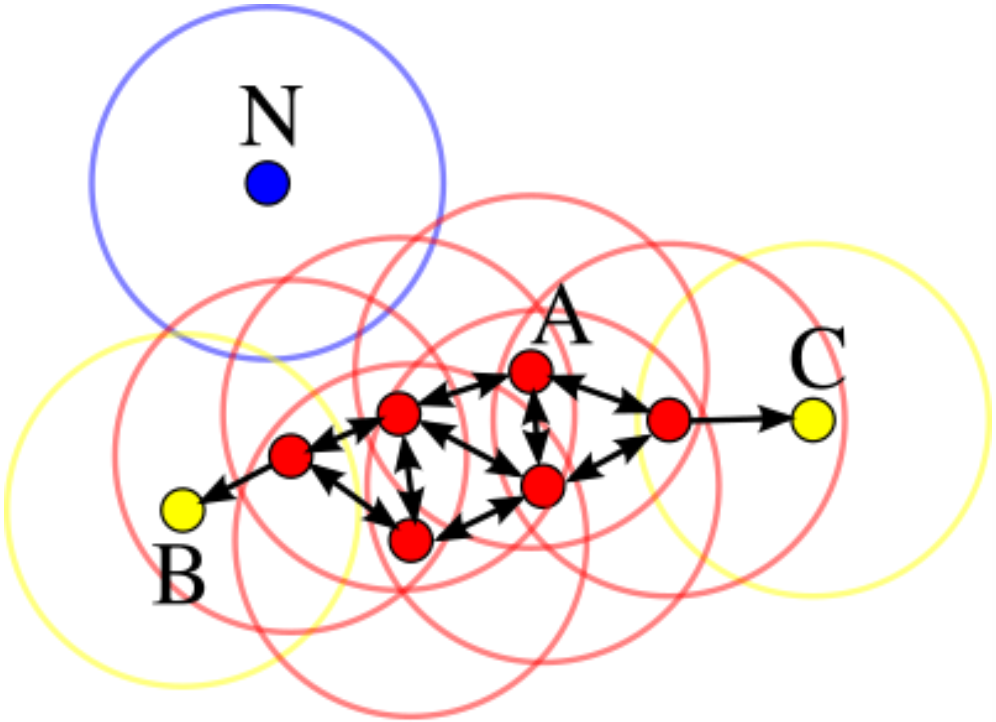


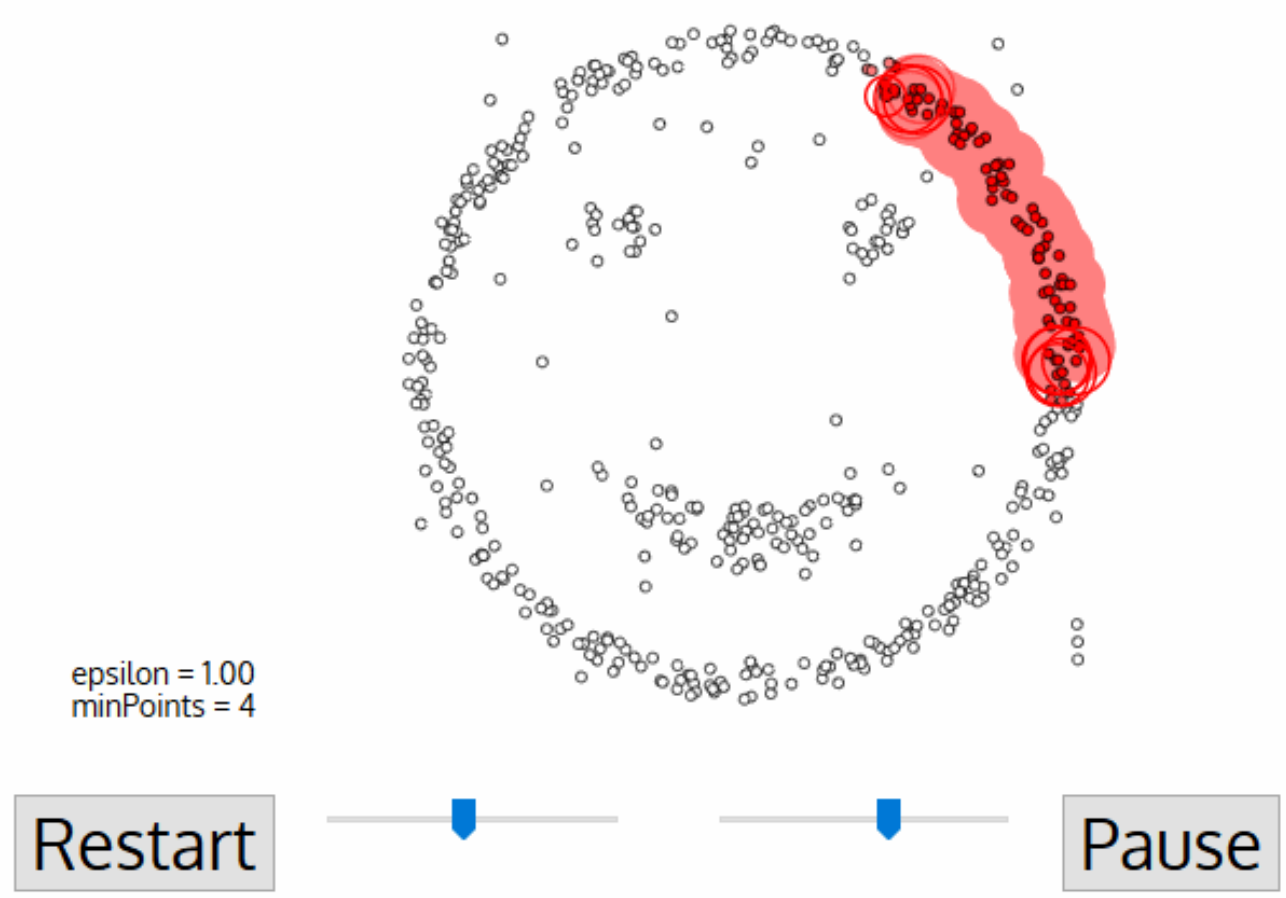
Повторение. Mean Shift





Повторение. DBSCAN





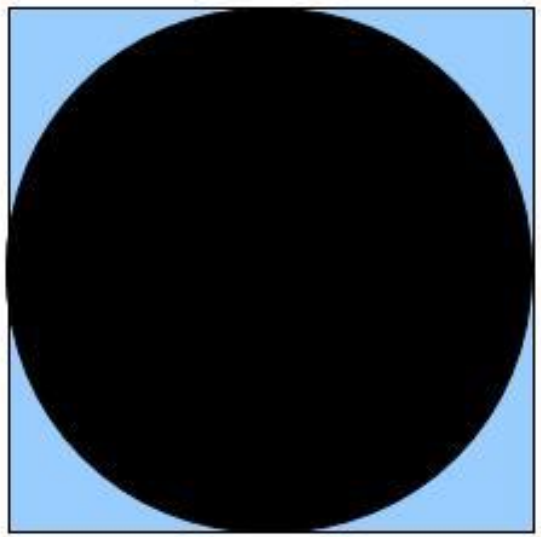
Уменьшение размерности

Задача: есть матрица X_0 размера $N \times M$

Хотим: получить матрицу X размера $N \times m$
$m < M$

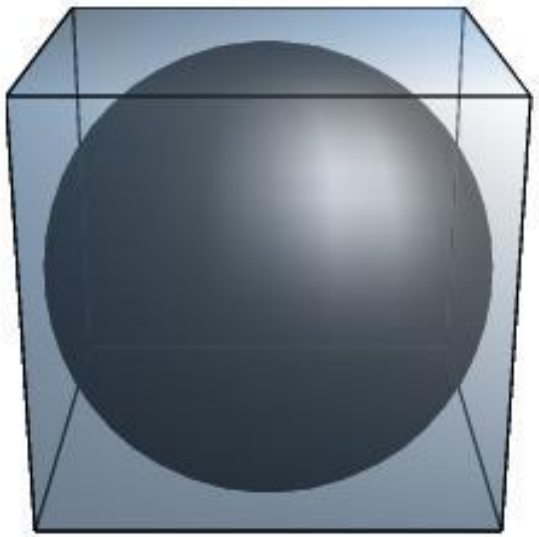
Уменьшение размерности. Зачем?

2 измерения



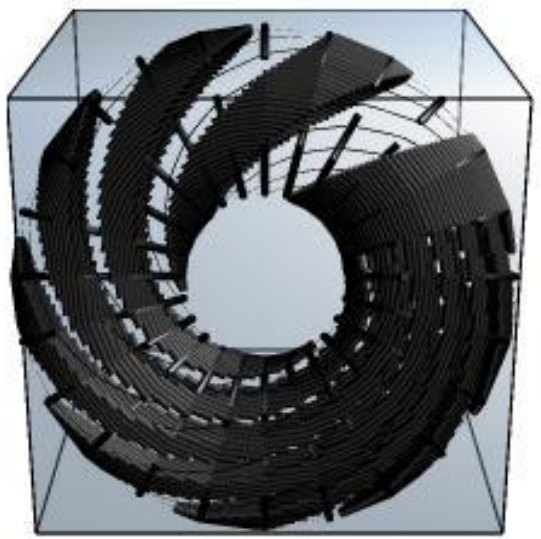
Охвачено 78.5% пространства

3 измерения



Охвачено 46.4% пространства

4...10 измерений

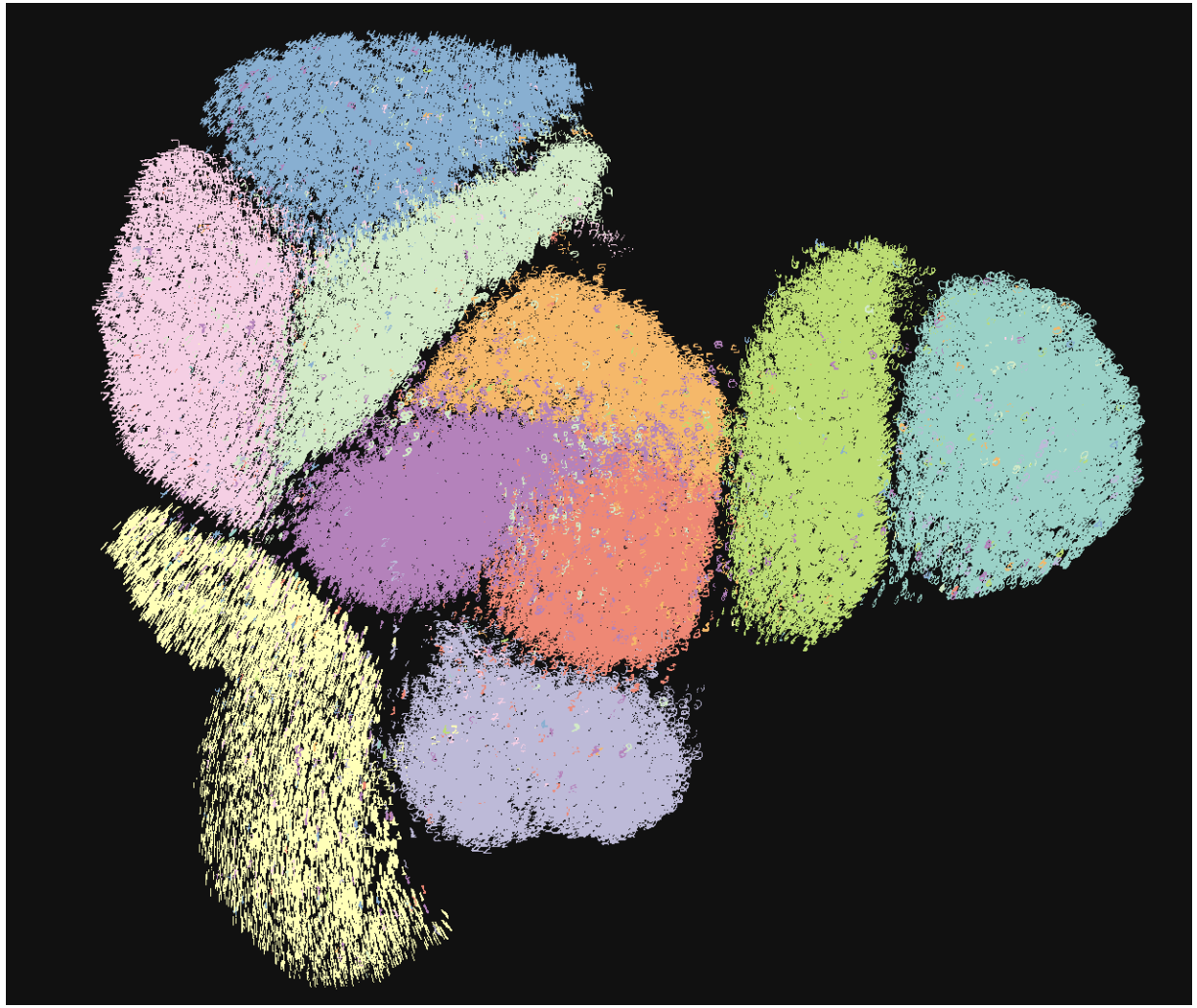


Охвачено 30.8%...0.25% пространства

Уменьшение размерности. Зачем?

Изначально имеем 784 признака

Визуализируем двумерно



Уменьшение размерности

Варианты для уменьшения размерности:

1. Убирать признаки (feature selection)
2. Переходить в иное пространство признаков (feature extraction)
    1. PCA
    2. T-SNE
    3. UMAP

Feature Selection

3 вариации:

1. Методы фильтрации
2. Методы-обертки
3. Методы представлений

Feature Selection. Методы фильтрации

Методы фильтрации оценивают каждый признак независимо друг от друга.

Выбираются признаки с высокой корреляцией с целевой переменной.

Используется в качестве предобработки датасета от избыточных или неинформативных признаков

Set of all features → Selecting the best subset → Learning algorithm → Performance

Feature selection. Методы фильтрации

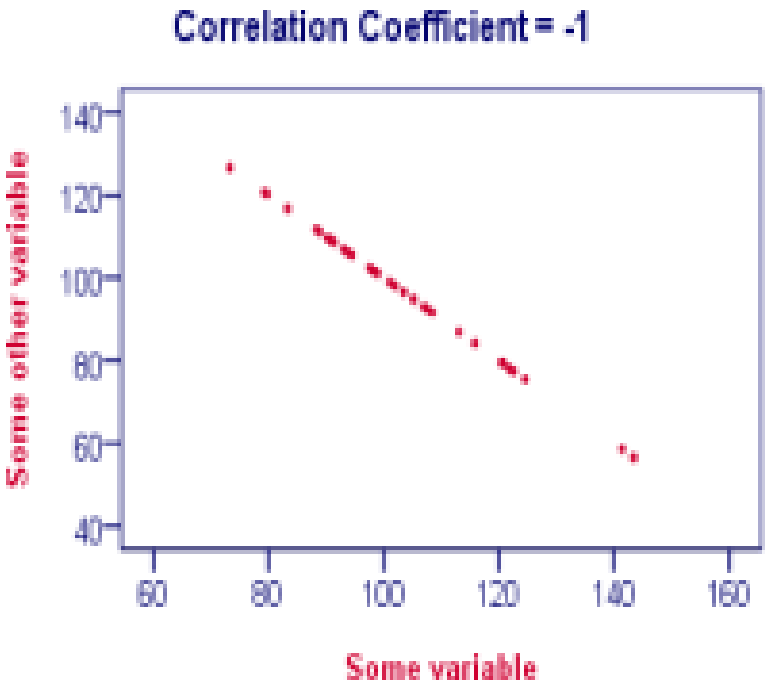
Удаляем константные признаки (определяем по нулевой дисперсии)

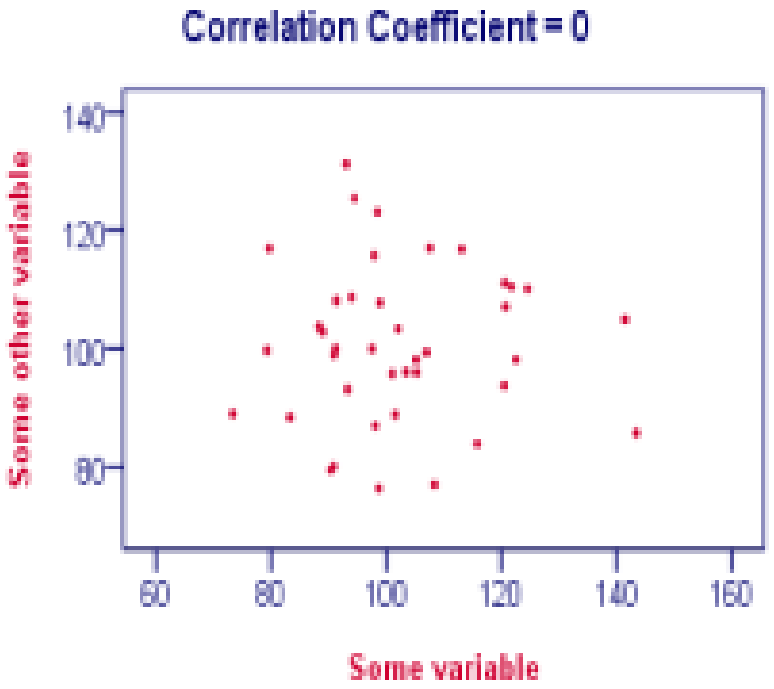
Или признаки, в которых много пропусков (условно >95% is None)

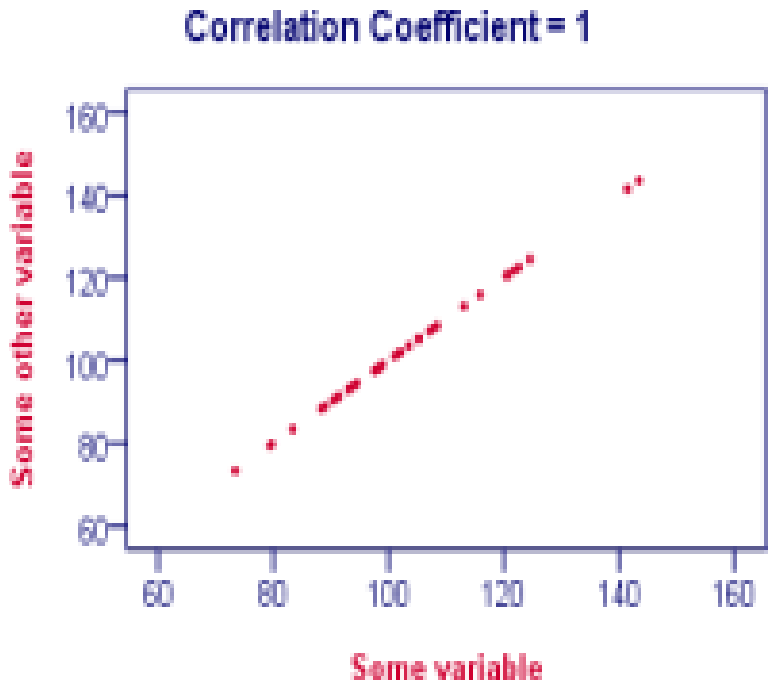
#	Car Make	Car Model	Car Year
1	Toyota	Camry	2018
2	Toyota	Corolla	2019
3	Toyota	Camry	2018
4	Toyota	Corolla	2019
...	...	...	...
...	...	...	...
...	...	...	...
9999	Toyota	Camry	2018
10000	Toyota	Camry	2018

Car Make	
Toyota	100%

Feature selection. Корреляция





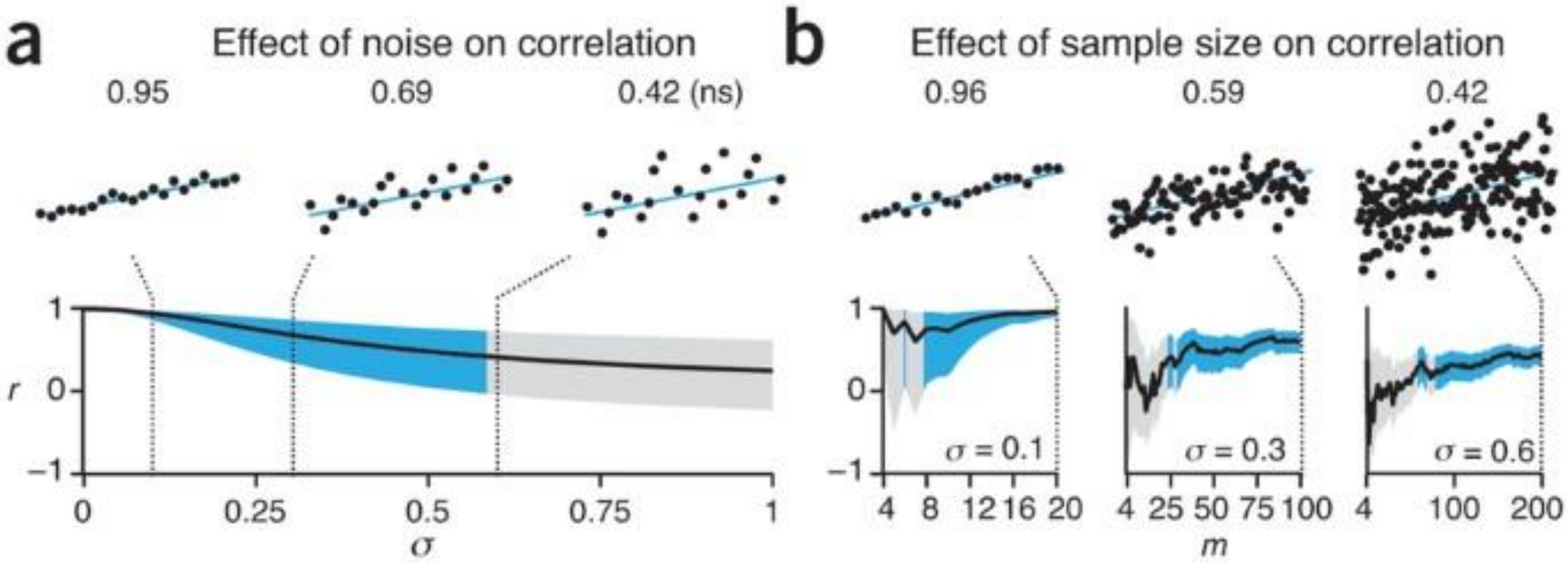


Feature selection. Методы фильтрации

Можно выбирать признаки по их взаимосвязи с целевой переменной с помощью:

1. Всевозможные корреляции (Пирсона, Крамера)
2. Статистические критерии (T-score, Chi-square)
3. Information Gain (как в решающих деревьях)

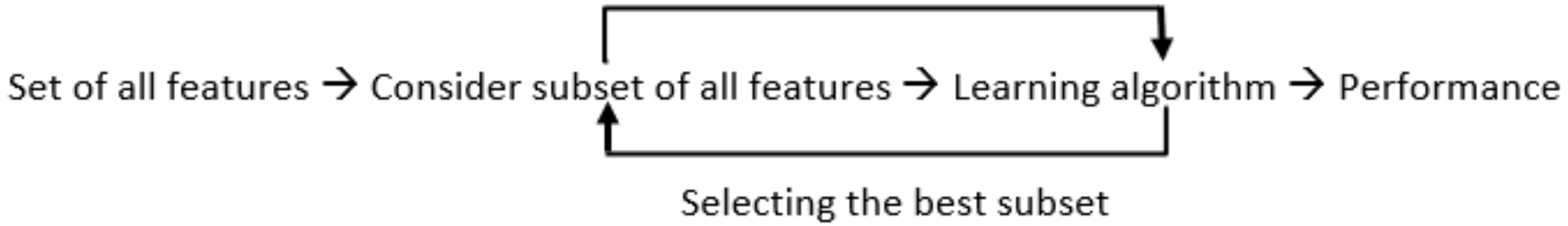
Корреляции не всегда достаточно



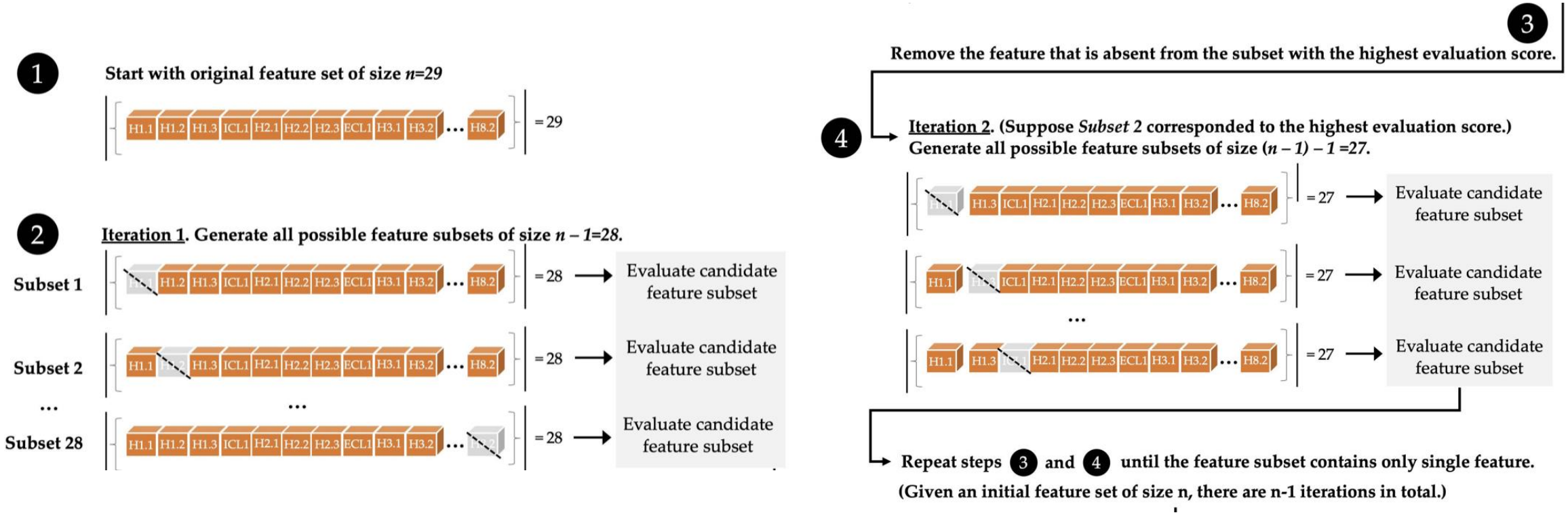
Feature selection. Методы обертки

Используются жадные алгоритмы отбора признаков

На каждой итерации берем подмножество признаков, смотрим и выбираем лучшее



Feature selection. Sequential Feature Selection

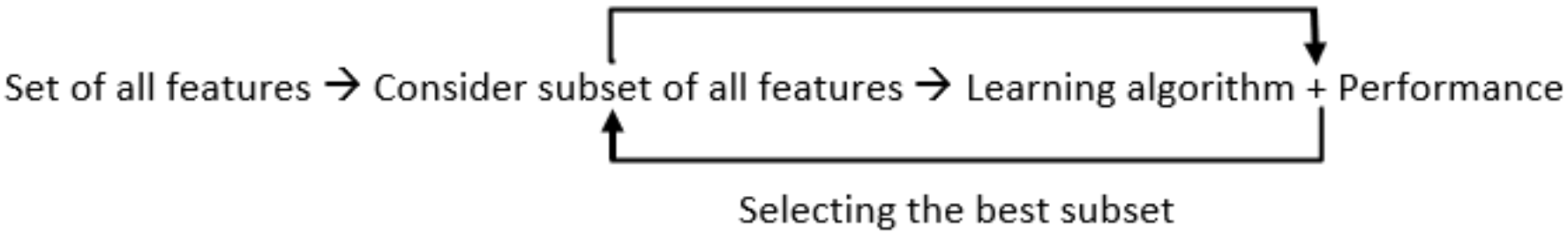


Бывает прямой (forward) и обратный (backward)

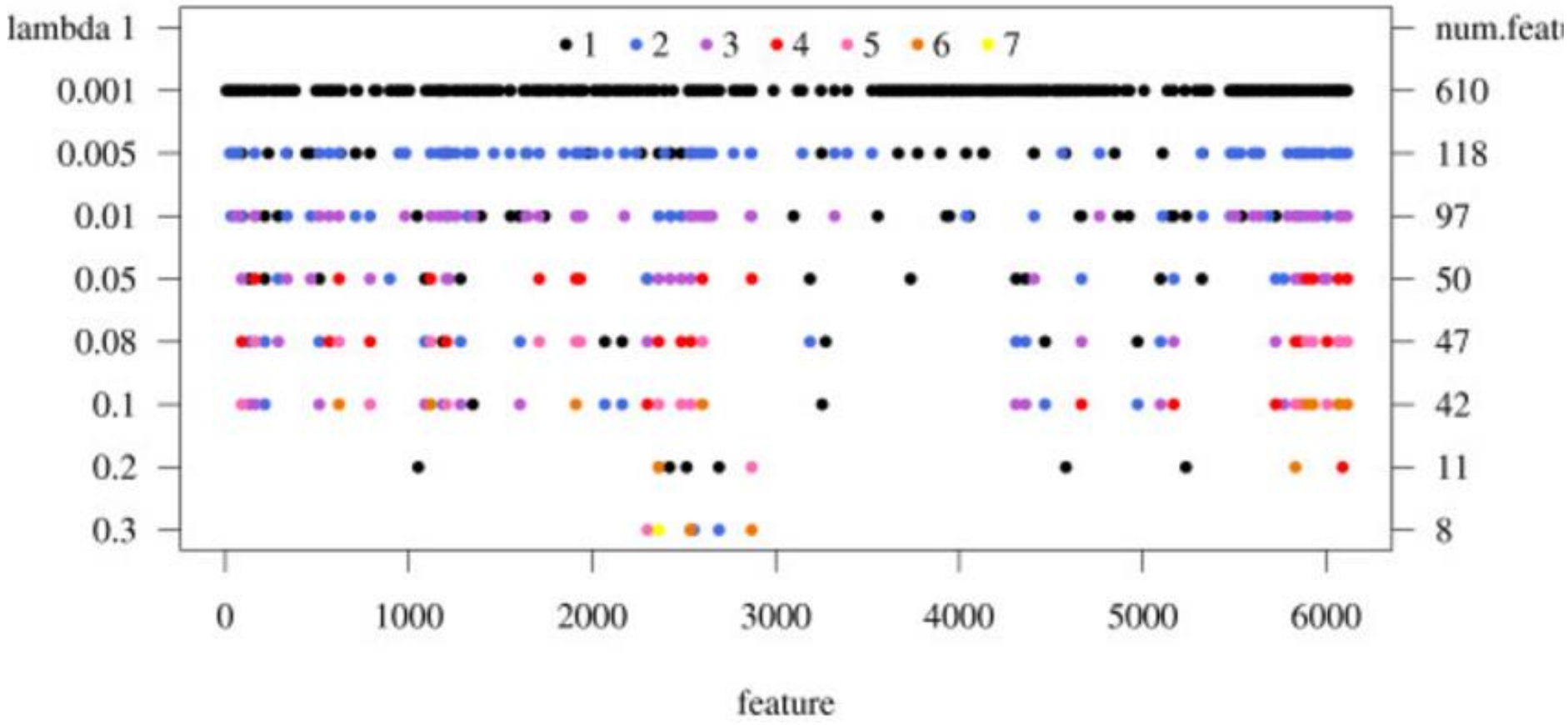
Feature Selection. Методы представлений

Пул признаков оценивается сразу во время обучения модели

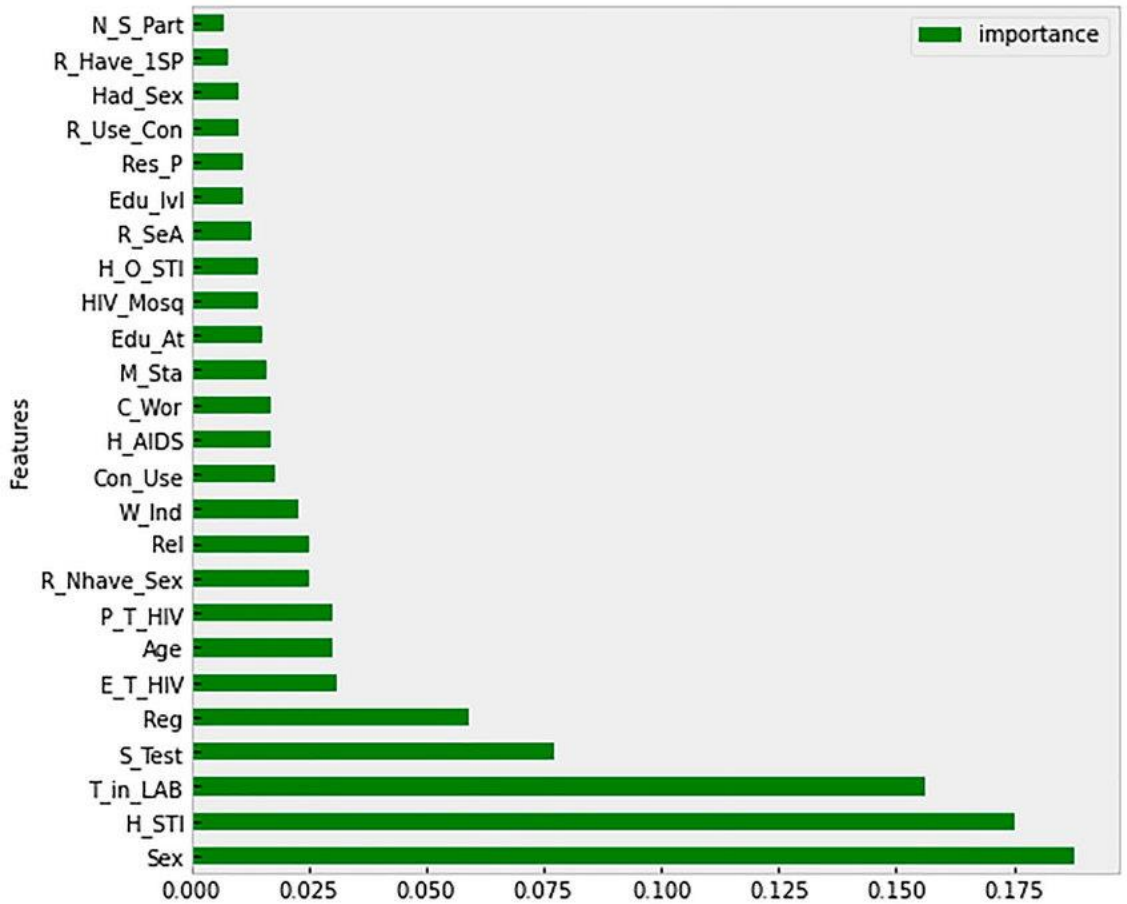
Учитывается влияние сразу нескольких признаков и их взаимосвязь между собой



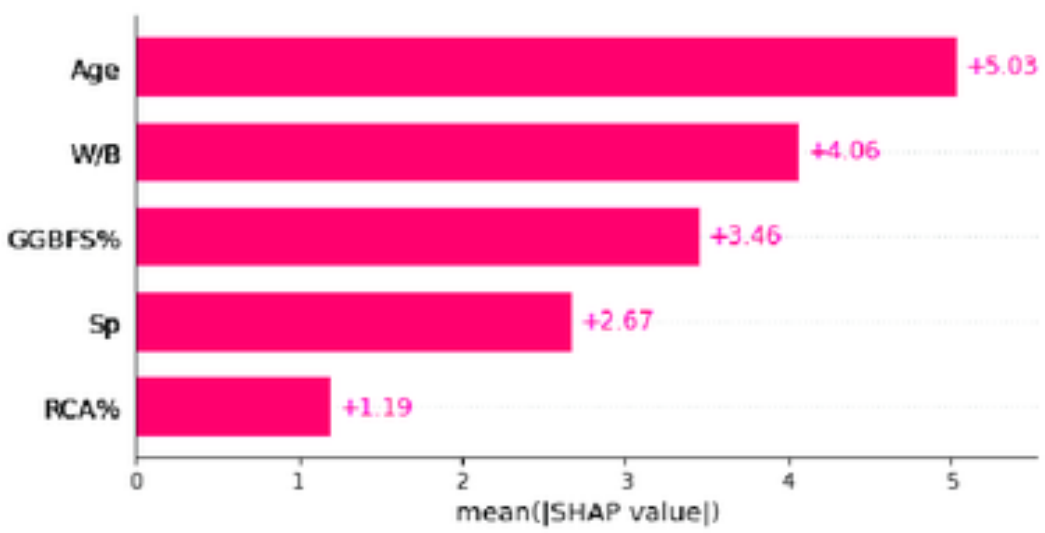
Feature Selection. LASSO (L1)



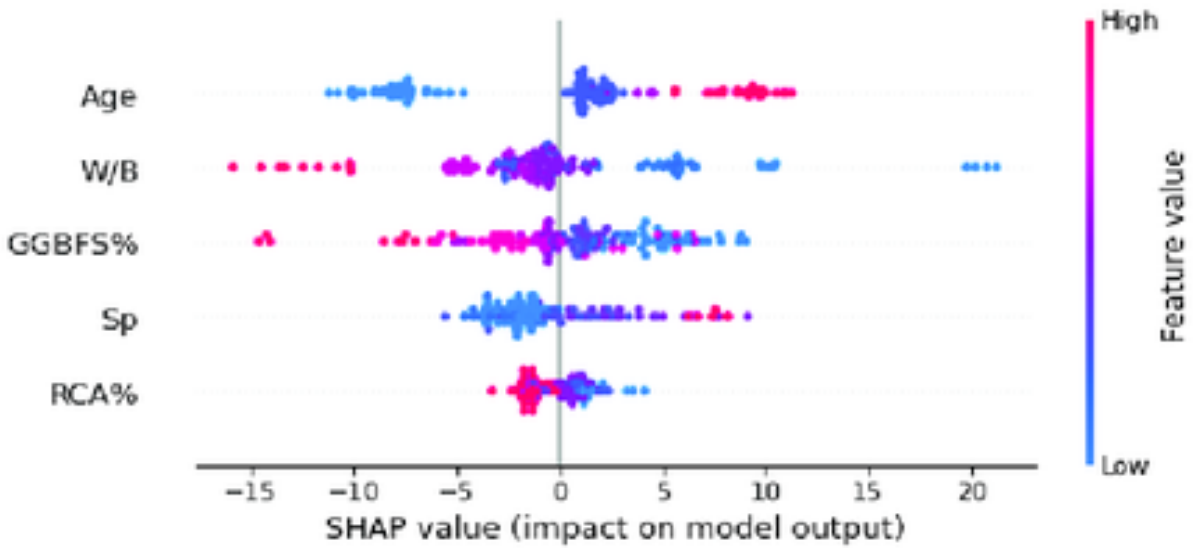
Feature Selection. Random Forest



Feature Selection. Gradient Boosting

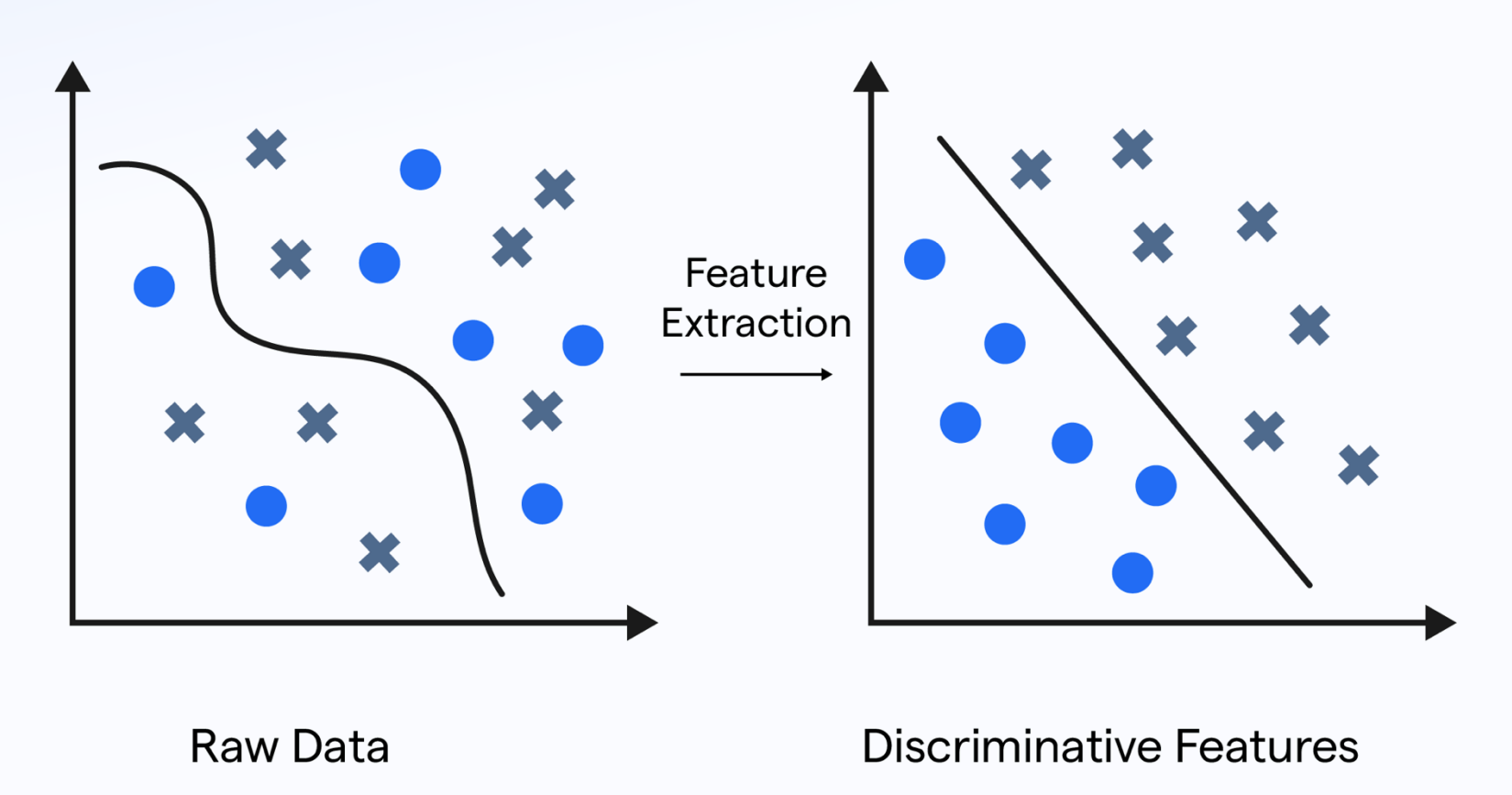


(a)



(b)

Feature Extraction. Переходим в иное пространство признаков



Метод главных компонент (PCA)

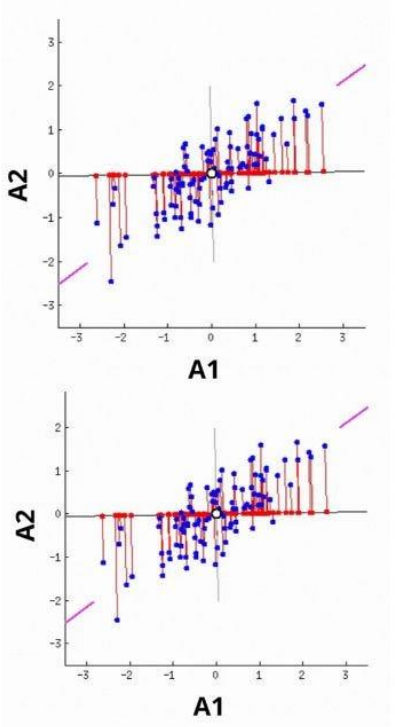
Проецирует данные в пространство меньшей размерности

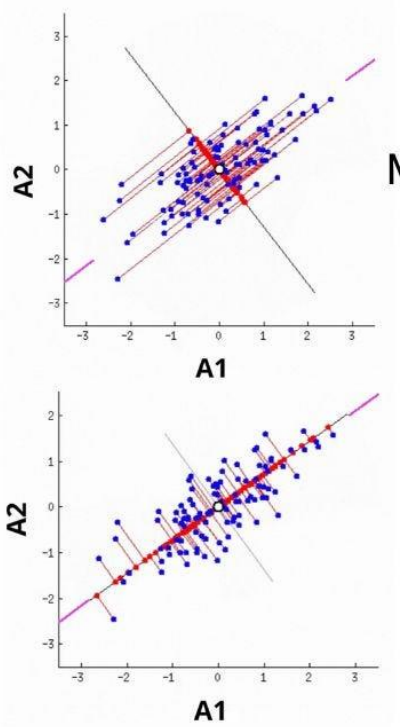
Но проецирует так, чтобы в новом пространстве мы **потеряли** как можно **меньше информации**

enjoy algorithms

Кроме того, отображение в новом пространстве будет линейно зависимо от изначального:

$$Z = XW^T$$





Most unsuitable

Most suitable

Алгоритм PCA

Шаг 1 – считаем ковариационную матрицу

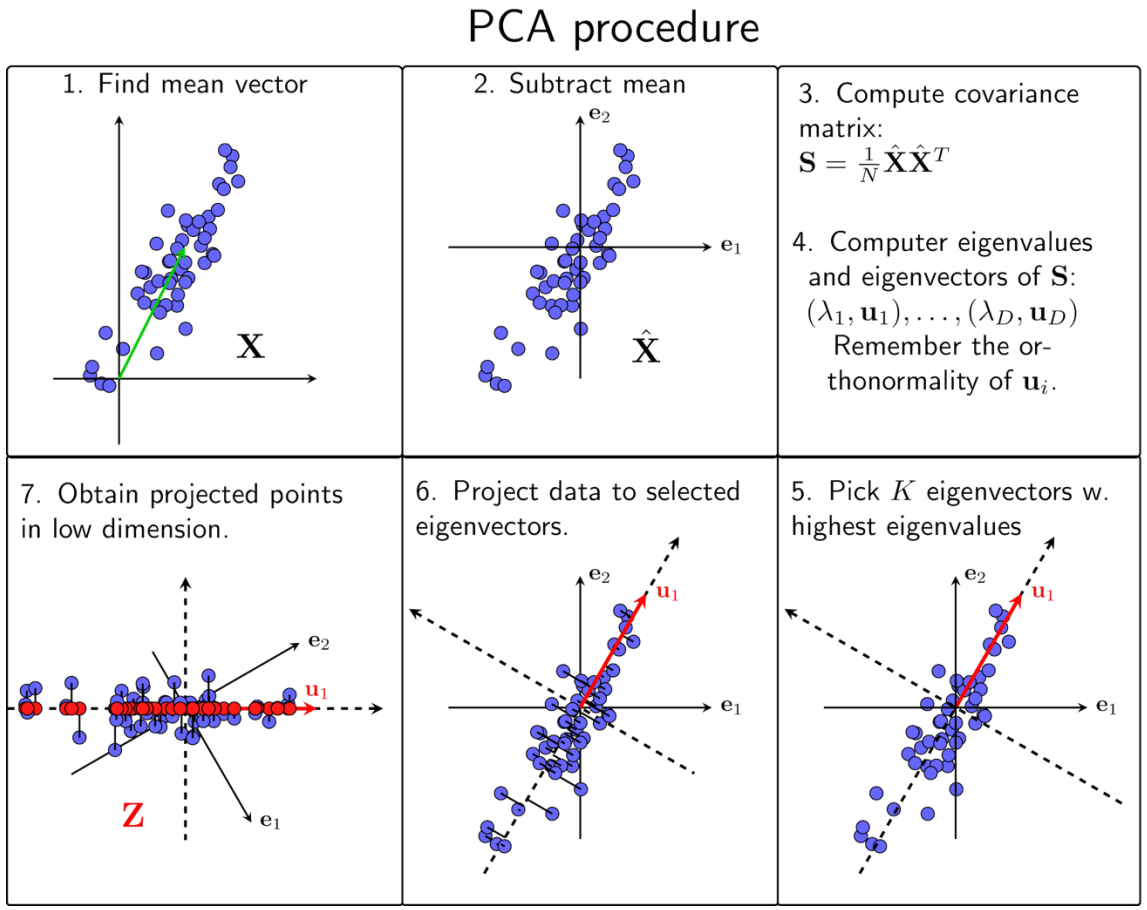
$$S = XX^T$$

Шаг 2 – считаем собственные вектора и собственные значения

$$Sx = \lambda x$$

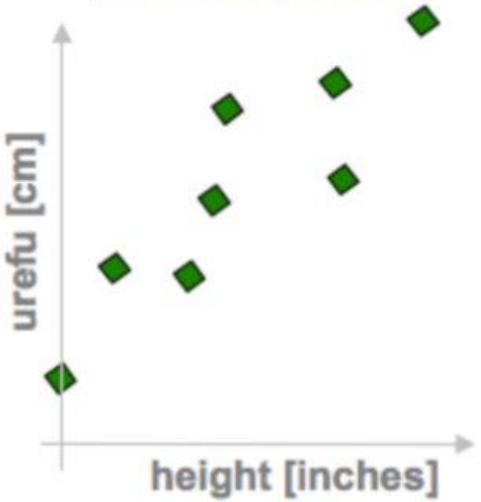
x – показывает направление, вдоль которых дисперсия будет максимальна

λ – показывает дисперсию по каждому направлению

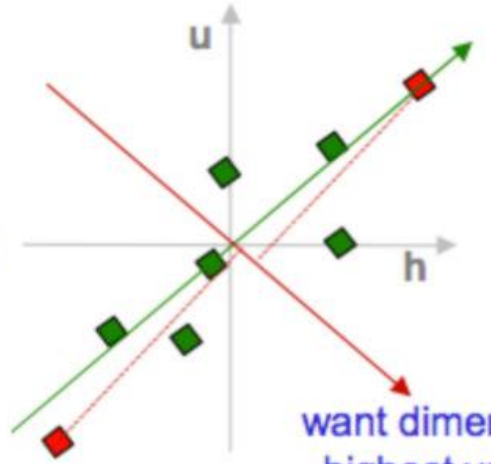


PCA in a nutshell

1. correlated hi-d data
("urefu" means "height" in Swahili)



2. center the points



want dimension of highest variance

3. compute covariance matrix

	h	u
h	2.0	0.8
u	0.8	0.6

$$\text{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

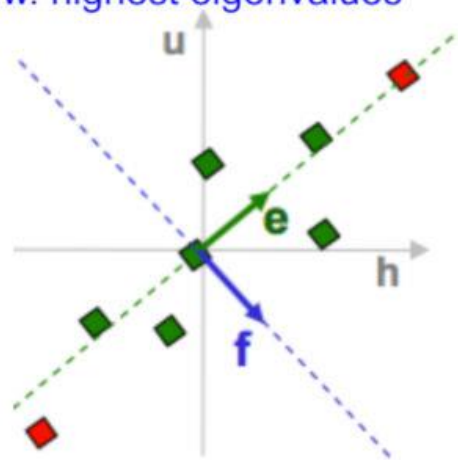
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

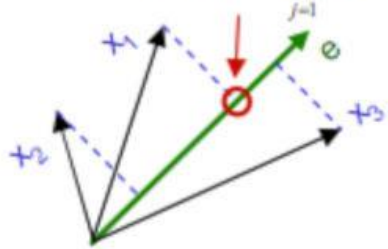
`eig(cov(data))`

5. pick m<d eigenvectors
w. highest eigenvalues

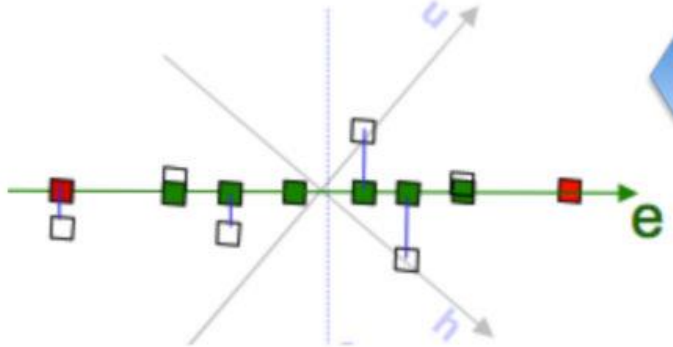


6. project data points to those eigenvectors

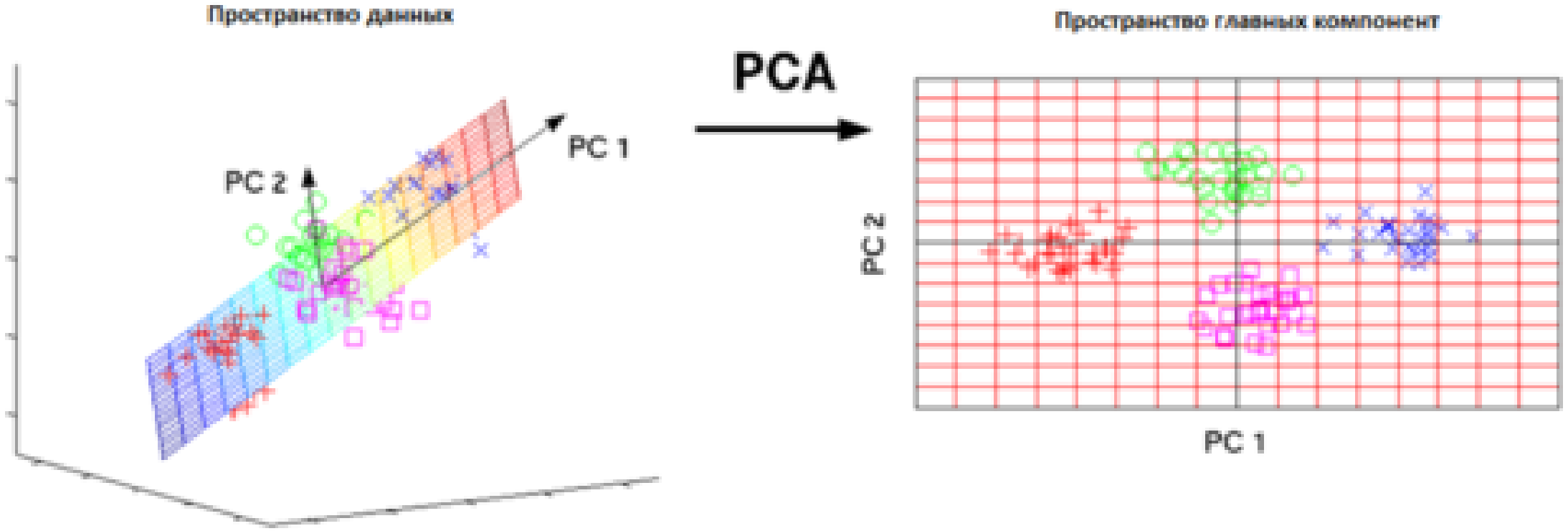
$$x_e' = x^T e = \sum_{j=1}^{d} x_{ij} e_j$$



7. uncorrelated low-d data



PCA наглядно



T-SNE

Расшифровывается – t-Stochastic Neighbor Embedding

Старается строить точки в новом пространстве так, чтобы изначально близкие точки были близки и в новом пространстве. Аналогично с далекими объектами

SNE

Изначально есть точки $\{x_i | x_i \in X\}$. Преобразовываем их в точки $\{y_i | y_i \in Y\}$ в более низкоразмерном пространстве.

Введем понятие вероятности выбора точкой x_i в качестве соседа точку x_j

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

SNE

Аналогично введем вероятности для нового пространства

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

По изначальной постановке задачи, чем $q_{j|i}$ ближе к $p_{j|i}$, тем лучше.

Но как измерить близость между ними?

SNE

Близость через дивергенцию Кульбака-Лейблера:

$$KL(P|Q) = \sum_j p_j \log \frac{p_j}{q_j}$$

Определим целевую функцию как:

$$C = \sum_i KL(p_i|q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

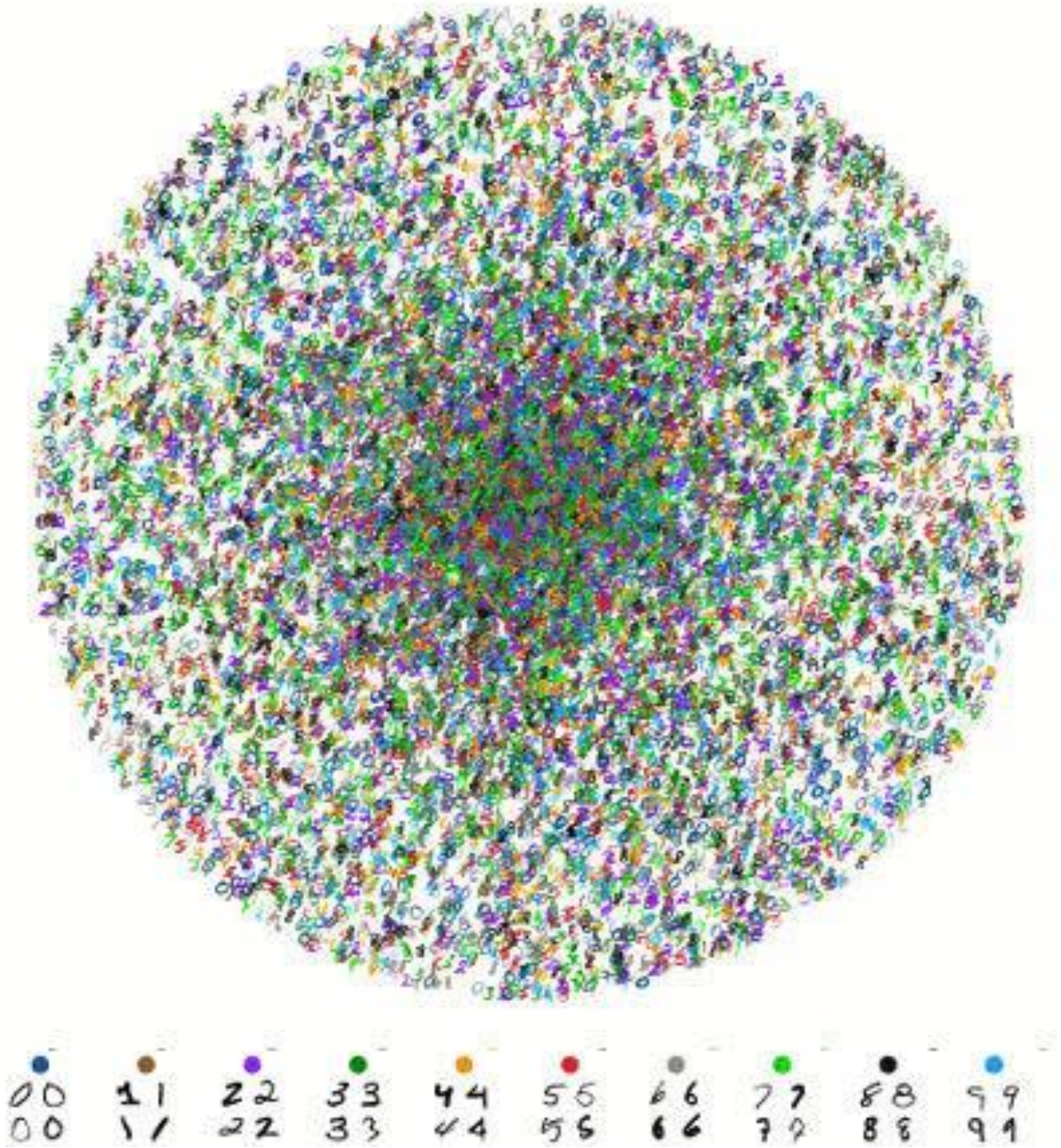
SNE

Далее случайно создаем множество точек y_i из нормального распределения $\mathcal{N}\left(0, \frac{1}{\sqrt{2}}\right)$.

Оптимизируем положение каждой точки при помощи градиентного спуска:

$$\frac{dC}{dy_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

T-SNE



T-SNE

До сих пор мы рассматривали реализацию просто алгоритма SNE.

T-SNE предлагает нововведения:

1. Так как KL-дивергенция несимметрична, можно использовать модификацию:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2|X|}$$

2. Использовать распределение Стьюдента с одной степенью свободы, а не Гаусса, так как распределение Стьюдента обладает более тяжелыми хвостами

T-SNE

Продолжаем с нововведениями:

3. Вводится «раннее сжатие», которое заставляет точки сжиматься в нуле (с помощью L2-регуляризации), чтобы кластера могли перемешиваться и в конечно итоге корректно расположиться друг относительно друга.

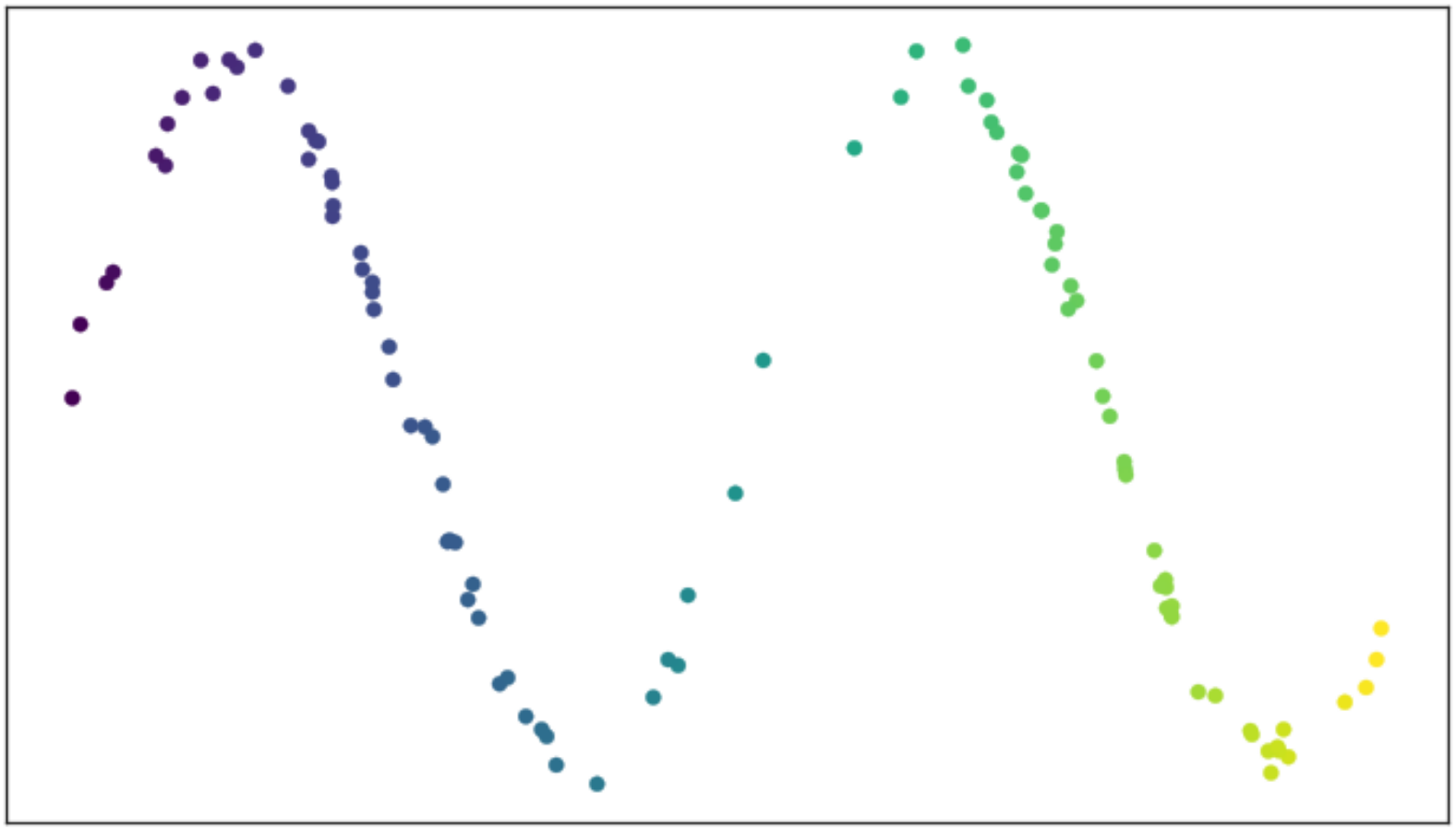
4.Вводится «раннее преувеличение», когда мы сильнее учитываем p_{ij} на первых итерациях, которое раскидывает широко кластера друг от друга и позволяет им наилучшим образом перестраиваться

ссылка

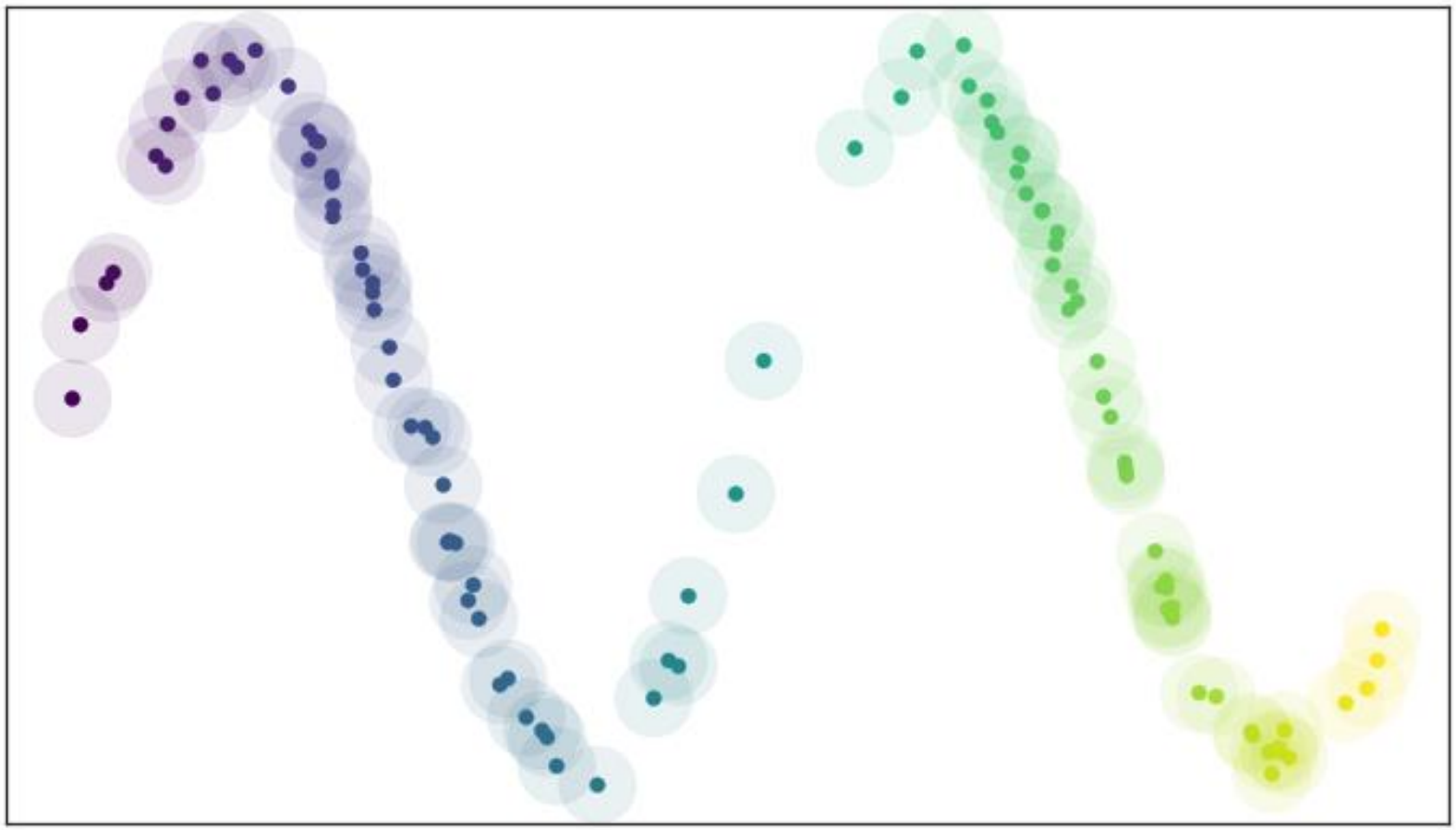
T-SNE

Посмотреть https://distill.pub/2016/misread-tsne/

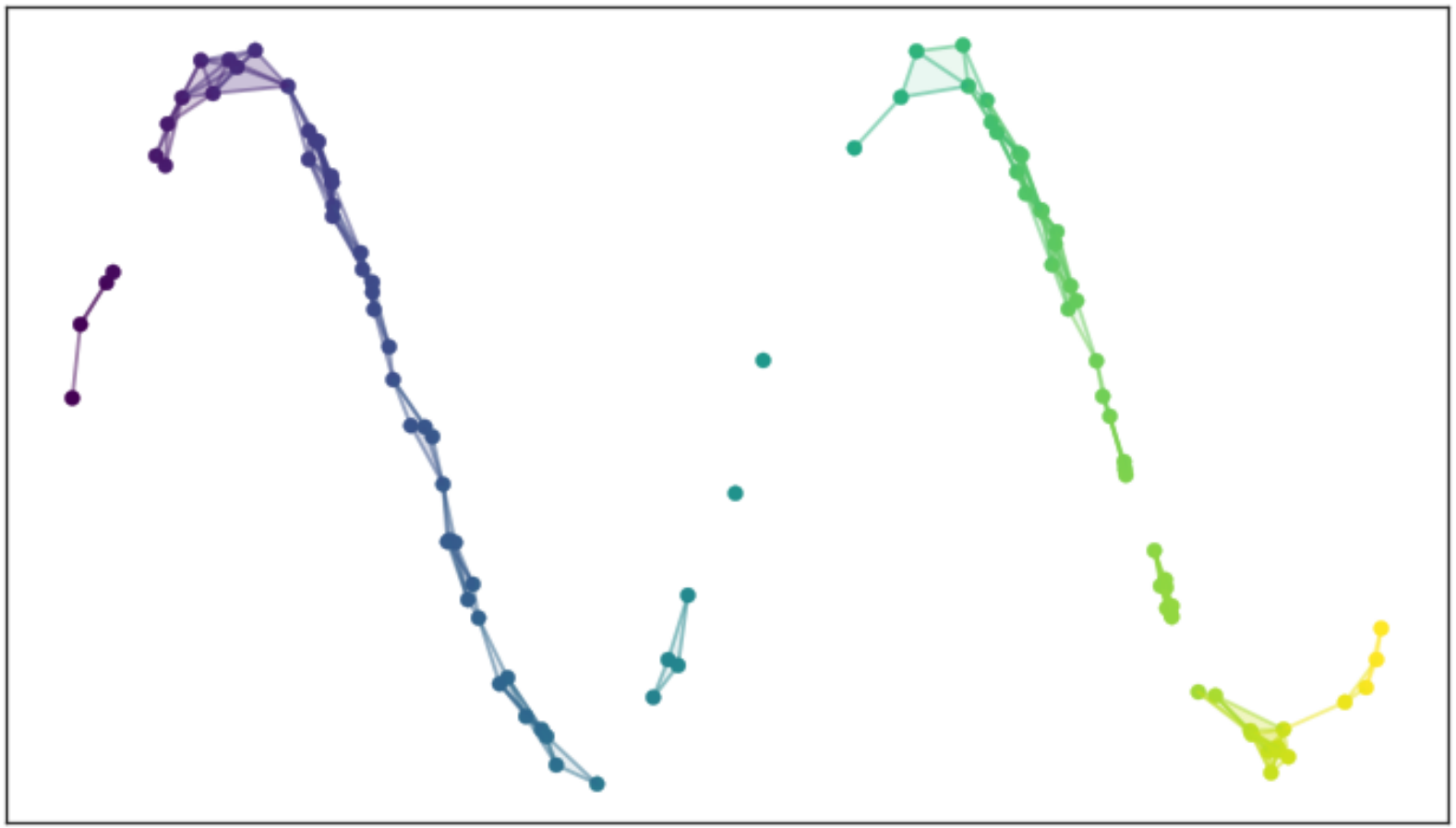
UMAP



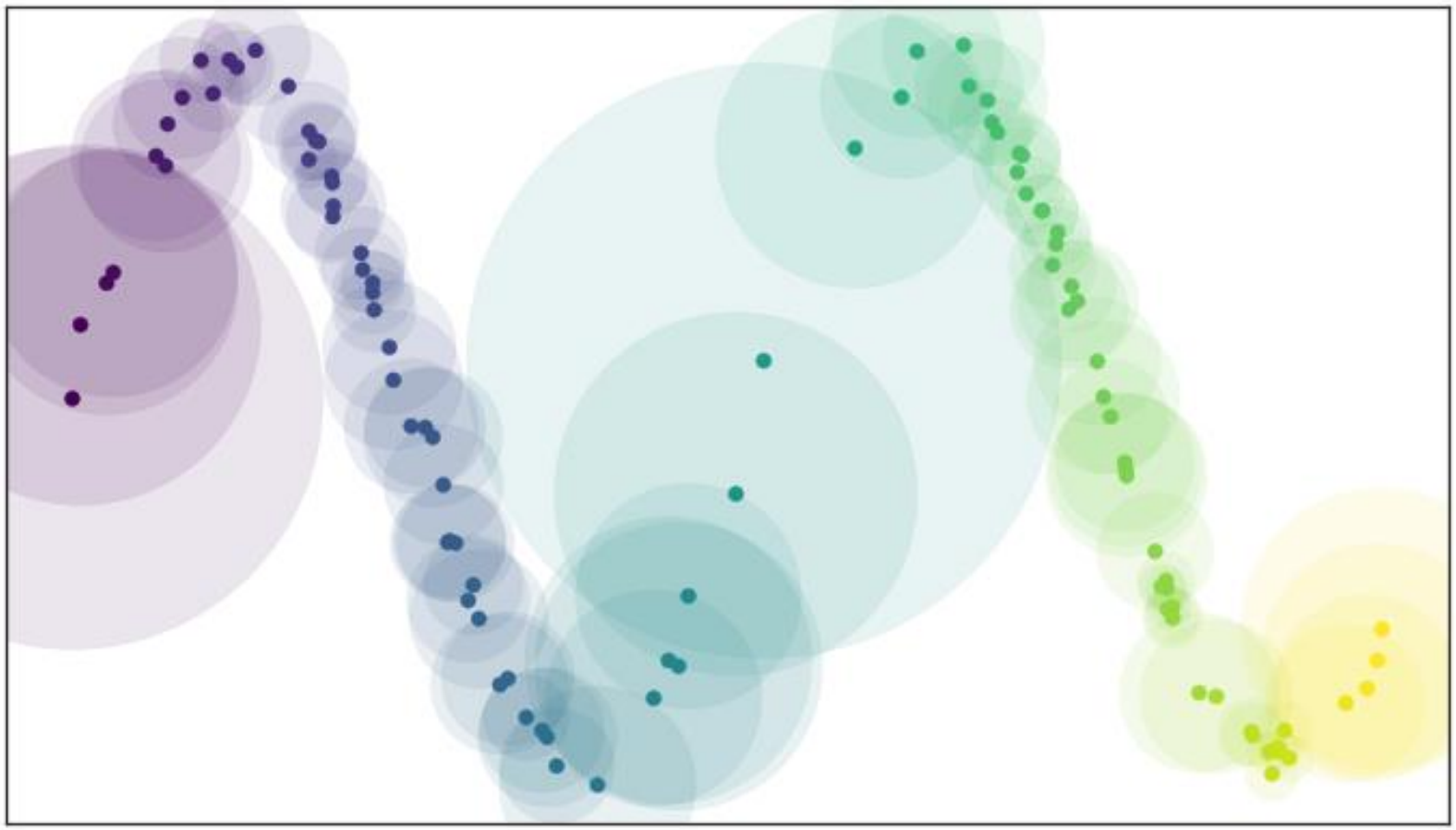
UMAP



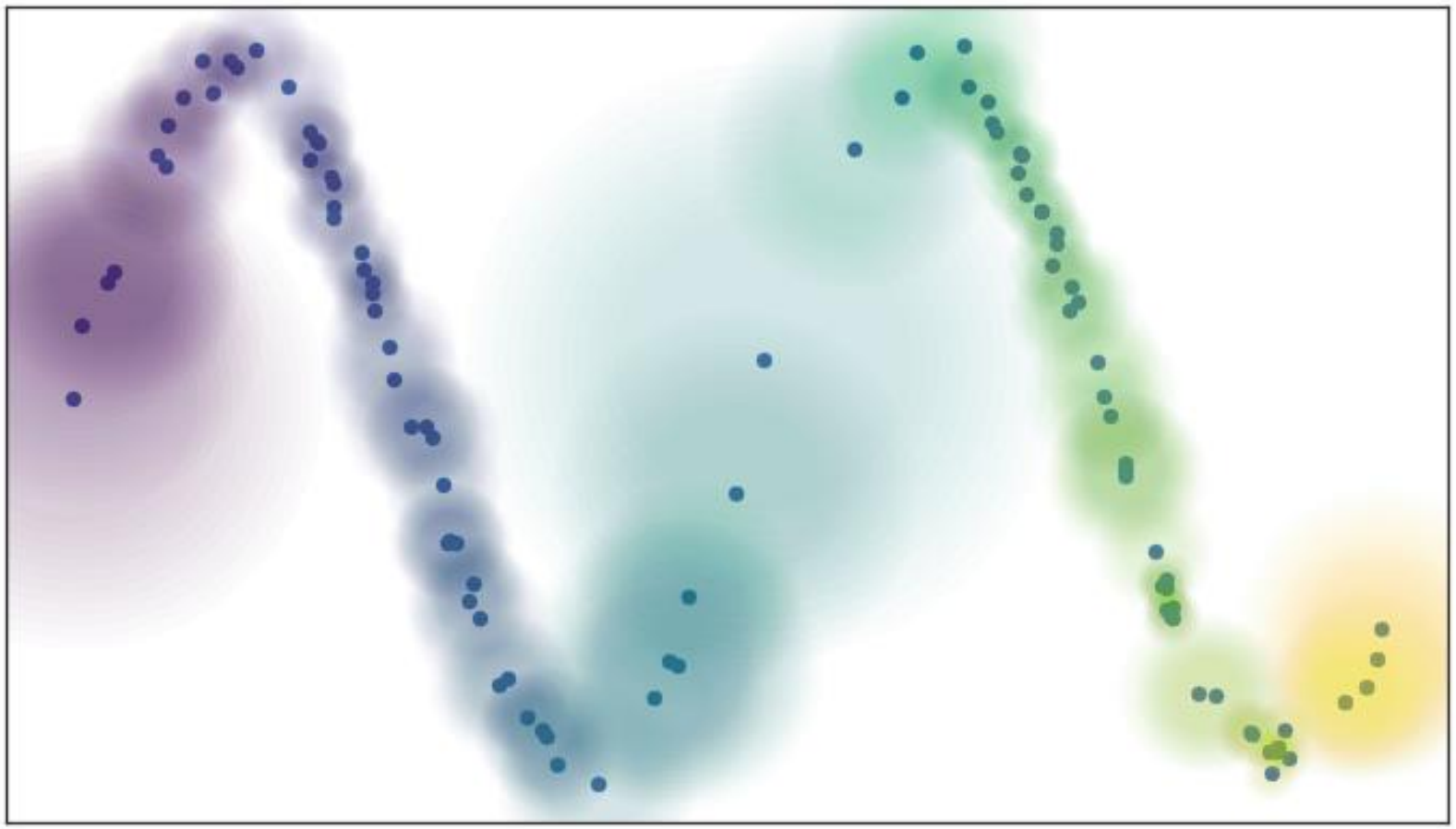
UMAP



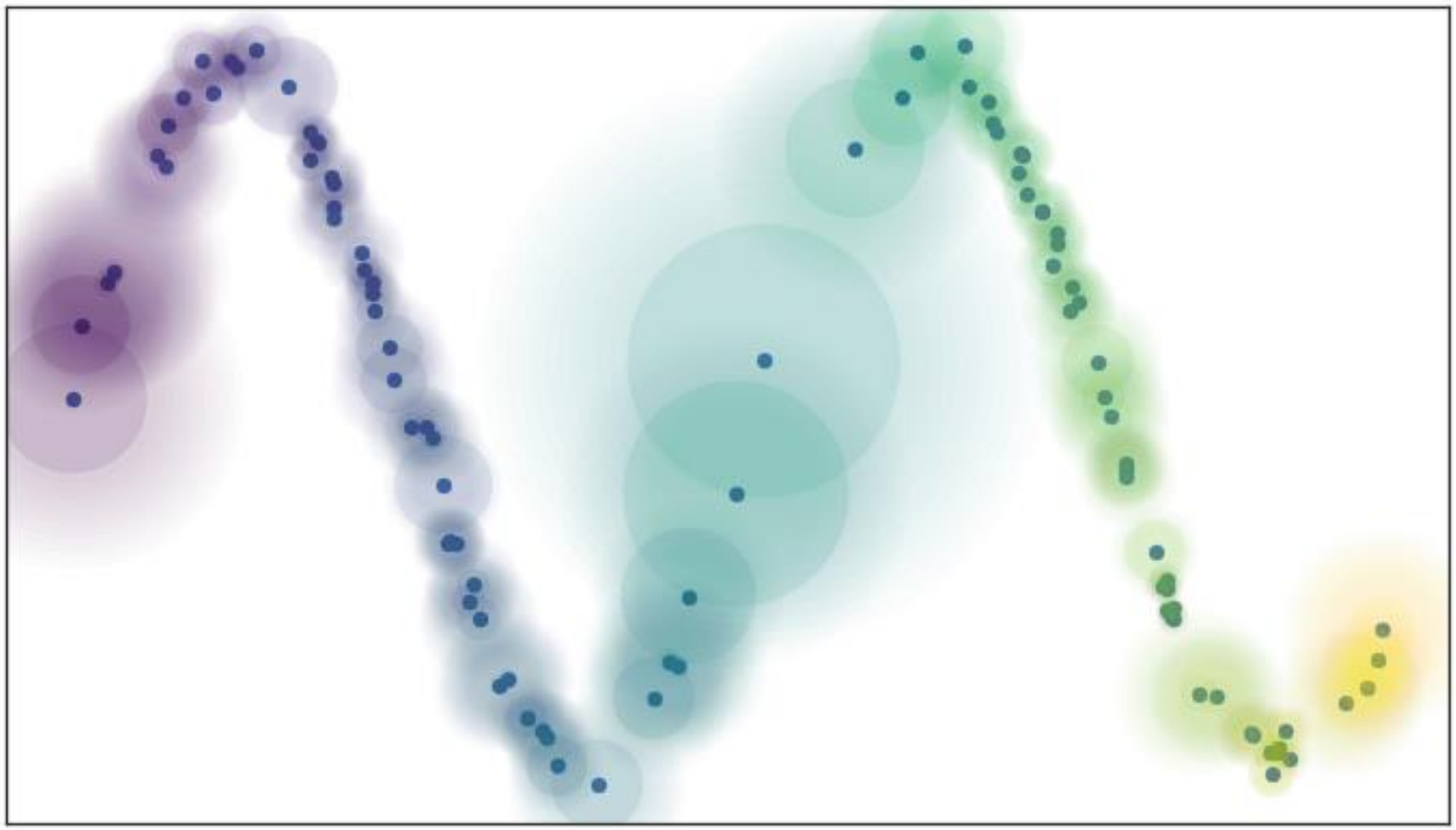
UMAP



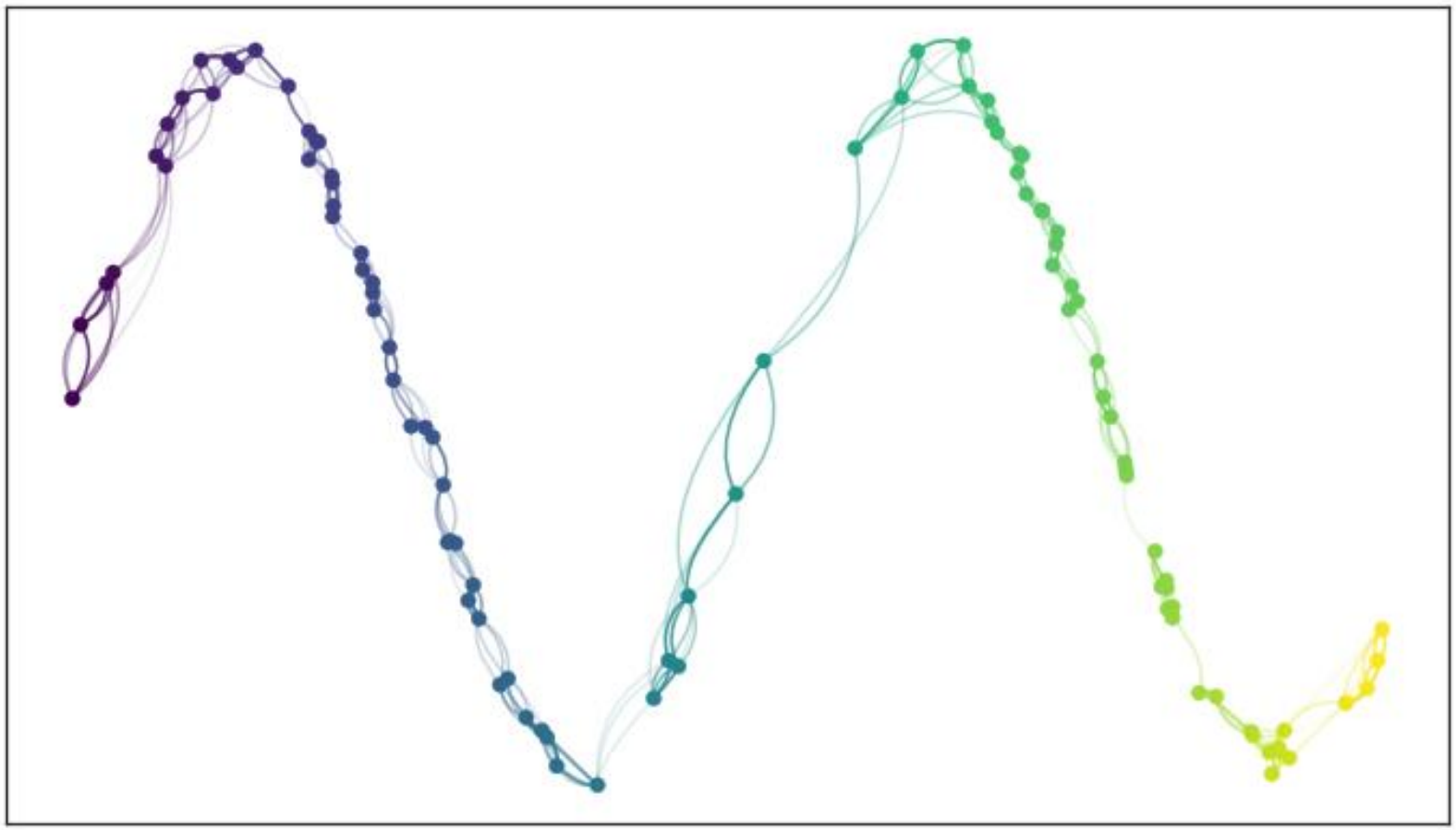
UMAP



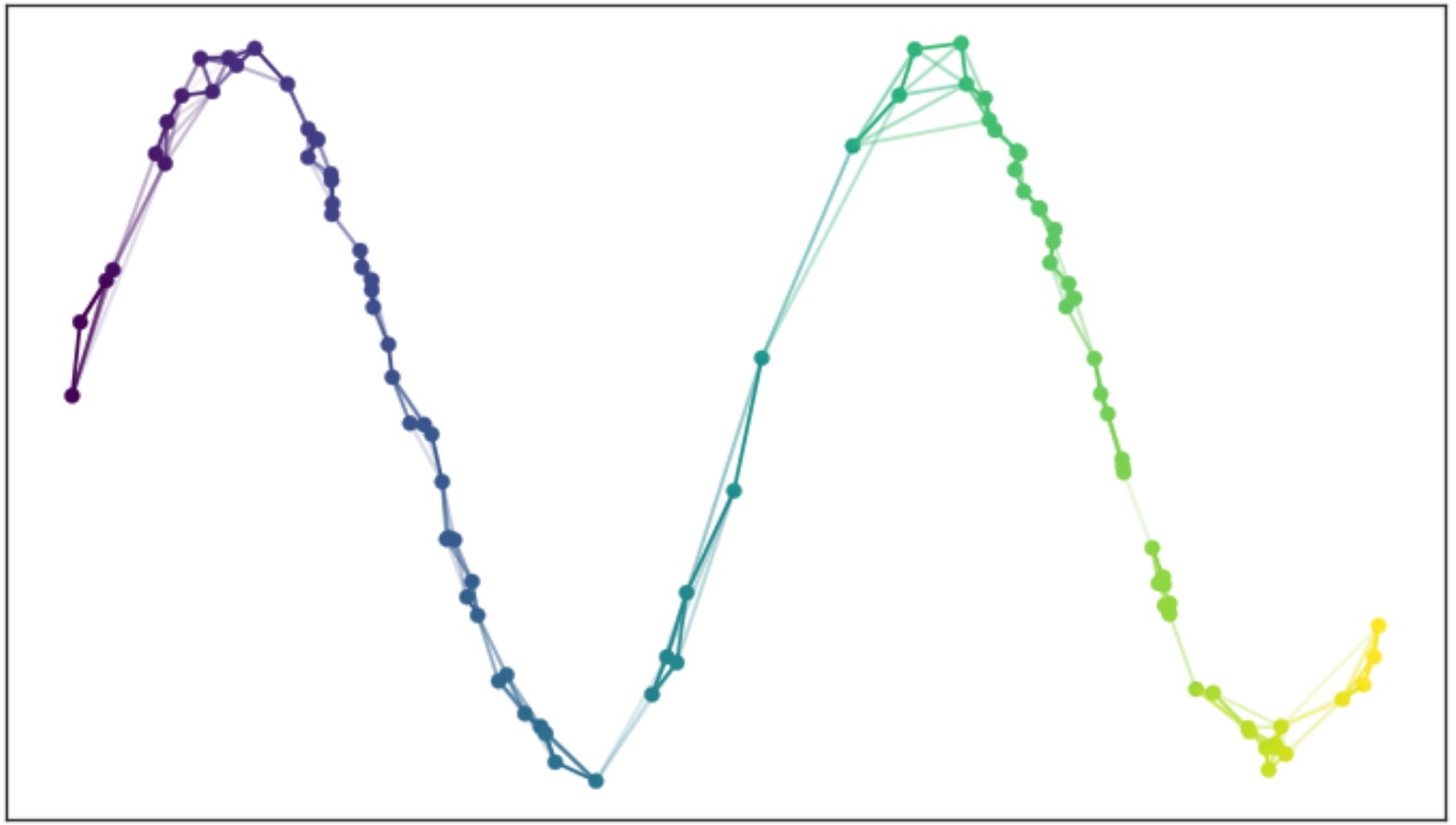
UMAP



UMAP



UMAP



ссылка

UMAP

Введем функцию потерь:

$$C = \sum_{e \in E} w_h(e) \log \frac{w_h(e)}{w_l(e)} + \big(1 - w_h(e)\big) \log \left(\frac{\big(1 - w_h(e)\big)}{\big(1 - w_l(e)\big)} \right) \to \min_{w_l}$$

где $w_h(e)$ -- функция принадлежности нечеткого множества из ребер в высокоразмерном пространстве

$w_l(e)$ -- функция принадлежности нечеткого множества из ребер в низкоразмерном пространстве

Минимизируем C градиентным спуском, изменяя w_l (по сути меняя конфигурацию точек в новом графе)

Красивые визуализации

https://pair-code.github.io/understanding-umap/

https://grantcuster.github.io/umap-explorer/