Кластеризация. Topic Modelling. PCA, SVD, pLSA.

Маша Шеянова, masha.shejanova@gmail.com

План

- Обучение без учителя
- Задача кластеризации, алгоритмы
- Тематическое моделирование
- Снижение размерности

Обучение без учителя

Для чего?

Итак, у нас есть данные, но нет про них “правильных ответов”. Можем ли мы всё ещё сделать с ними что-то толковое?

- научиться по контексту слова предсказывать само слово (ага, Word2Vec)
- рекламное агентство может раскидать пользователей по “кучкам” по интересам и таргетить каждую отдельно
- “человеко-читаемо” нарисовать на плоскости сложное явление
- снизить требуемый объём памяти и время вычислений

Обучение без учителей — часто инмтрумент для **анализа**, а не продукта. Ещё чаще — **промежуточный шаг**, чтобы потом лучше решить задачу с учителем.

Виды

- кластеризация

“Раскидай мои фотографии по папочкам. Сам реши, по каким”.

У нас есть выборка объектов, но нет заданных классов. Мы хотим разбить их на группы так, чтобы объекты в разных группах сильно отличались.

- снижение размерности

Часто приходится иметь дело с данными больших размерностей. Их сложно хранить и обрабатывать. Мы хотим снизить размерность, оставив наиболее значимые компоненты. Пример: визуализация в 2D.

Supervised vs. Unsupervised ML

С учителем:

- нужны размеченные данные (дорого, не всегда есть, зависим от качества)
- много хороших алгоритмов, метрик, понятно, чего мы хотим достичь
- если данные хорошие и из них можно извлечь говорящие признаки, хороших результатов добиться легко

Без учителя:

- разметка не нужна, ура! данные валяются на каждом шагу.
- непонятно, как измерять качество (или сложнее понять)
- хороших результатов добиться сложно (сначала понять бы, что нужно)

Кластеризация

Что и зачем

Разбивает объекты на кучки по неизвестному признаку. Сделать так, чтобы похожее было с похожим.

Например:

- разбить покупателей на кучки, а потом понять, что кому нужно
- управлять новостными потоками, понимая что о чём
- разложить фотографии по папочкам
- активное обучение (какие данные надо разметить для supervised ml)
- найти необычное поведение (чего угодно где угодно, например тех же покупателей)

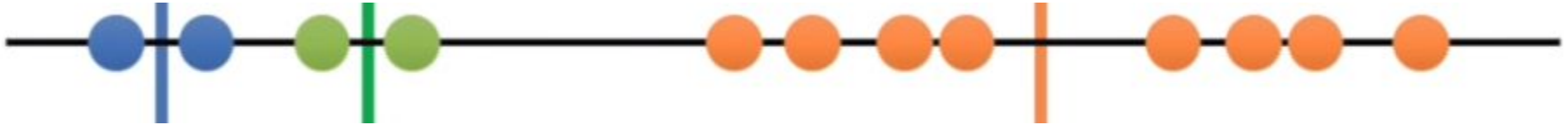
k-means: идея (картинки — из видео)

Допустим, у нас есть точки на прямой, и мы хотим разбить их на 3 кластера.

Давайте закинем в эти данные (куда придётся) 3 кружка — центроида — и для каждой точки найдём, к какому кружку она ближе.

k-means: идея

Когда все точки покрашены в цвета самых близких центроидов, мы двигаем центроиды в среднее значение подключённых к ним точек...



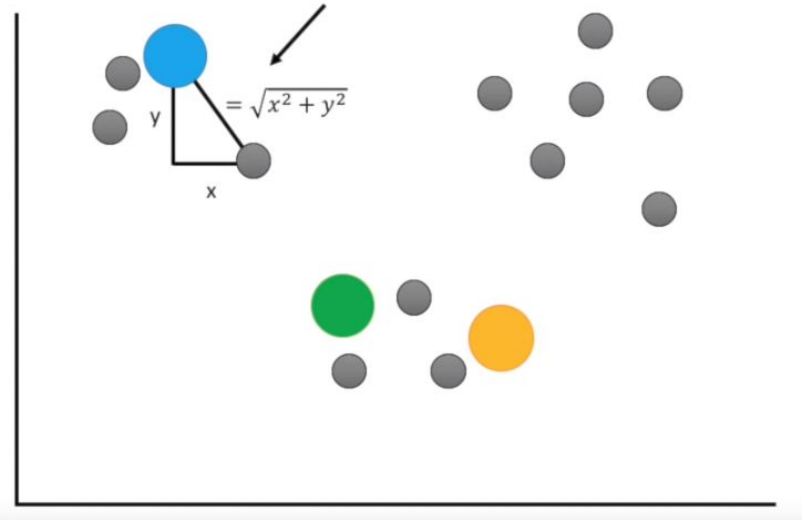
... и повторяем предыдущий шаг. Пока не сойдётся.

На трёхмерной плоскости

... то же самое, считаем расстояние от центоридов до точек.

Правда, вместо простого xc - xn теперь более хитрый подсчёт расстояния.

Евклидово расстояние:

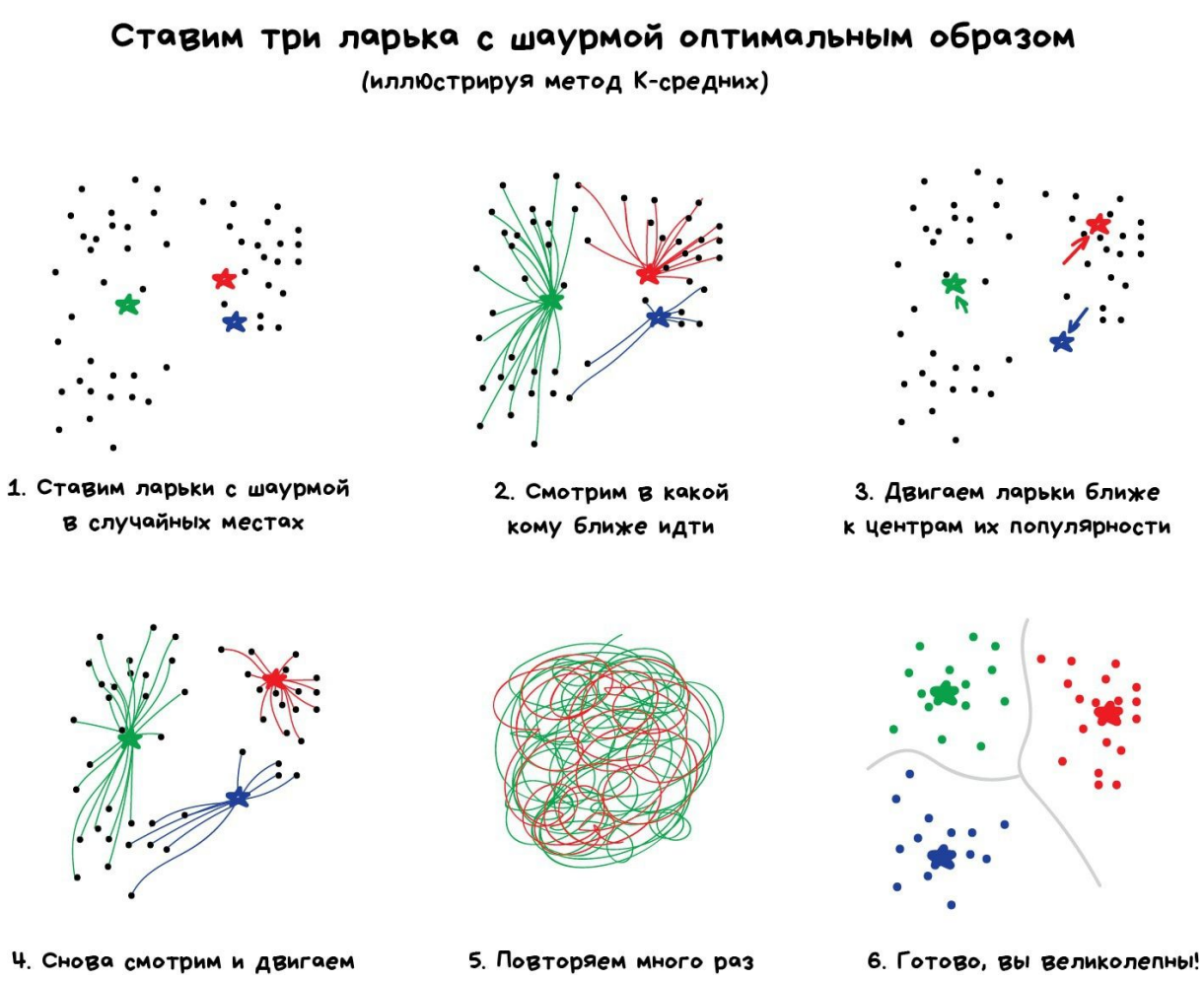


$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

K-means...

... на примере ларьков с шаурмой (отсюда, конечно).

А вот здесь есть ещё хорошая визуализация в движении.



Но что делать, если кластеры построились плохо?

И вообще, как понять, что они построены плохо, ведь ответов у нас нет!
Возьмём каждый кластер и посчитаем **дисперсию**!

Дисперсия: $M\left[\left(X - M[X]\right)^2\right]$ Чем больше отклонение от центра, тем хуже.

А теперь будем случайным образом кидать центроиды несколько раз и использовать дисперсию как критерий, насколько хорошо получилось.

K-means: алгоритм

- случайно определить k центроидов
- для каждого объекта найти ближайший центроид и приписать его к соответствующему кластеру
- передвинуть центроиды к центрам своих кластеров
- посчитать дисперсию
- повторять предыдущие шаги, пока не сойдётся
- когда сошлось, посчитать дисперсию
- повторять все предыдущие шаги сколько хотим, а потом выбрать лучший результат

k-means в sklearn

from sklearn.cluster import KMeans

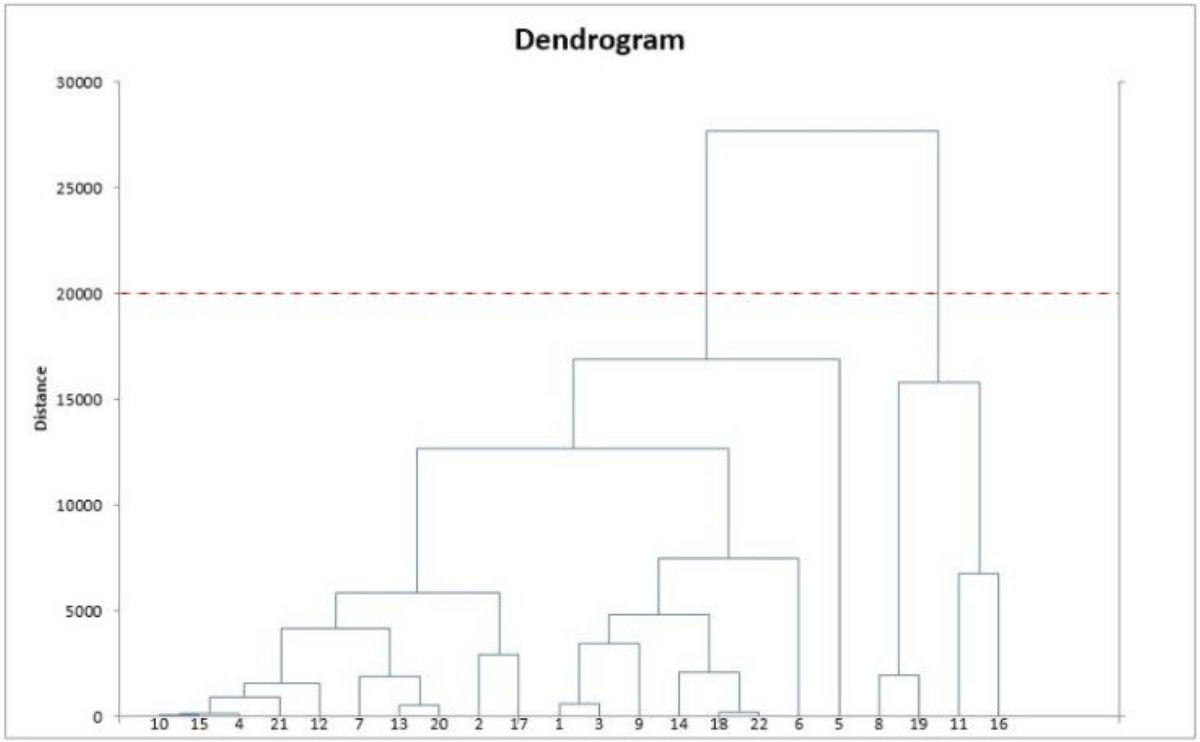
- n_clusters — количество центроидов (дефолт 8)
- init — как инициализировать центроиды
  - k-means++ — умный способ кинуть центроиды хорошо
  - random — случайным образом
  - ndarray — задать самим
- n_init — сколько раз кидать разные центроиды
- max_iter — сколько раз двигать центроиды

Hierarchical clustering

Почти то же самое , но позволяет решить, сколько кластеров надо, позже.

- найти N кластеров
- слить два кластера, которые ближе всего (for some definition of “ближе”)
- пересчитать расстояния между кластерами
- продолжать, пока не останется один кластер
- выбрать, какое расстояние будем считать достаточным, чтобы разбивать на кластеры

Hierarchical clustering

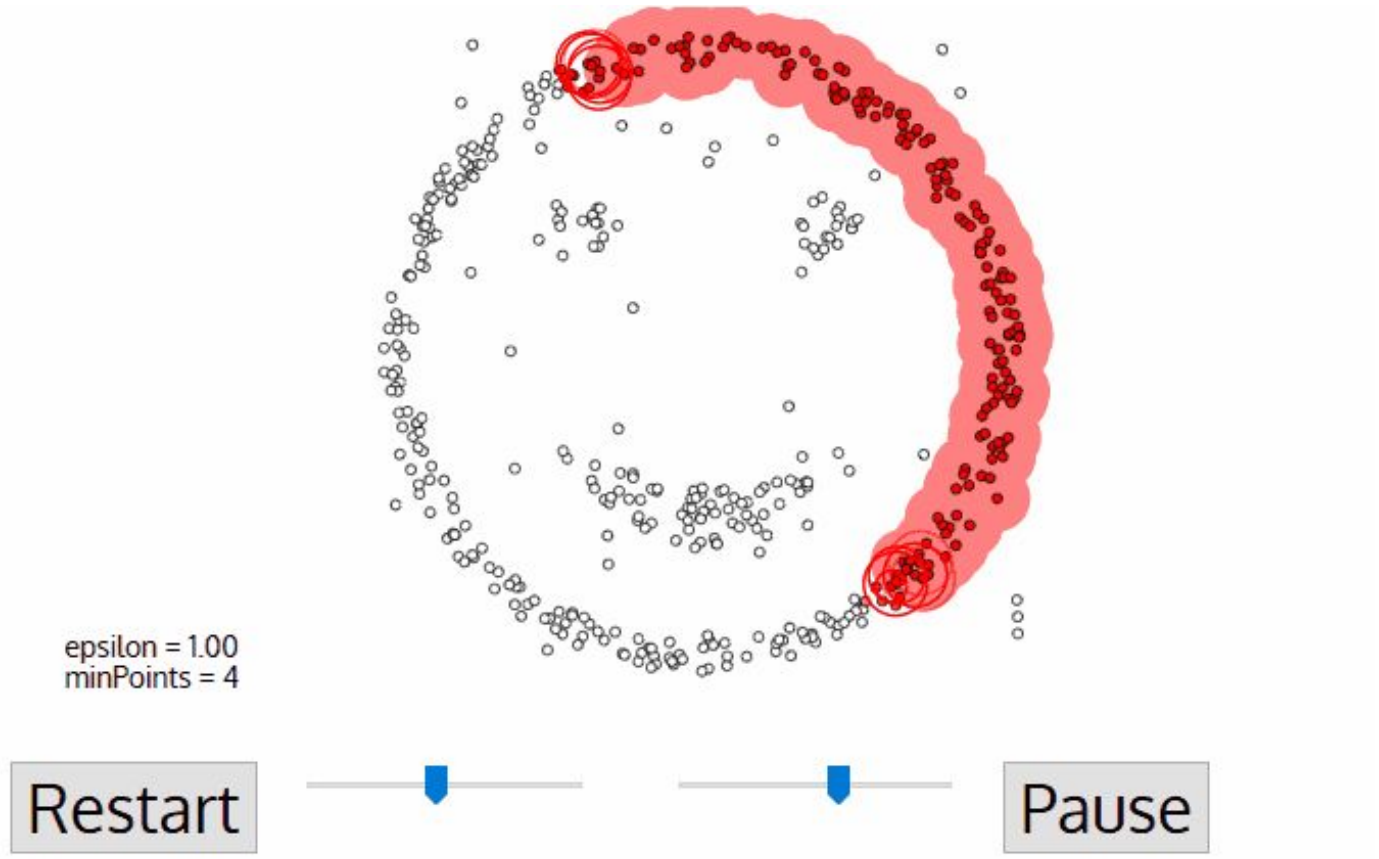


Дендрограмма кластеров.

Выбираем, сколько кластеров нам надо и проводим прямую, где хотим.

DBSCAN

Density-based spatial clustering.



Метрики оценки кластеризации

Как я уже говорила, измерить это гораздо сложнее. Но метрики всё же есть.

- Rand Index
- Adjusted Rand Index
- Гомогенность
- Полнота
- V-мера

Снижение размерности

Что и зачем

В общем случае — у нас есть признаковое пространство на много-много измерений (например, мешок слов по корпусу, и каждое слово — признак). Мы хотим “сжать” их как-то так, чтобы потерять минимум информации.

Каждое новое “измерение” — элемент вектора — будут заключать в себе обобщённое представление нескольких элементов из большого вектора.

- сжать картинку
- убрать несущественные признаки
- тематическое моделирование

Тематическое моделирование

Что и зачем

Тема — “о чём документ” $\simeq$ набор часто совместно встречающихся слов

Мы считаем, что тема употребление того или иного слова зависит от темы. А тема — от документа.

Зачем:

- поиск в электронных библиотеках
- трекинг новостных сюжетов
- “продвинутый” эмбеддинг документа

topic modeling vs. clustering

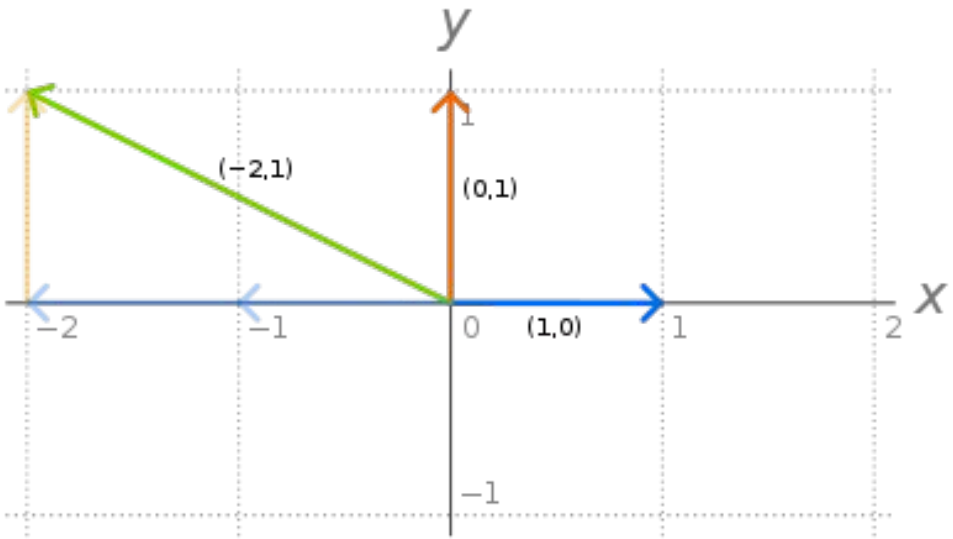
Что похожего: есть документы, раскидываем их по кучкам, заранее не знаем по каким.

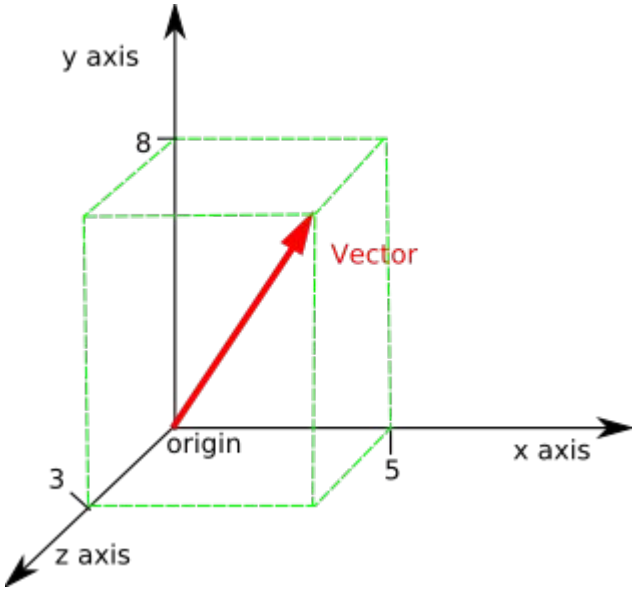
Что разного: у одного документа может быть высокая степень принадлежности больше, чем к одной теме.

PCA

Базис линейного пространства

Стандартный базис:





Замена базиса

На самом деле, базисные вектора можно выбирать как угодно — главное чтобы можно было выразить через них все вектора пространства.

(И чтобы сами базисные вектора нельзя было выразить друг через друга).

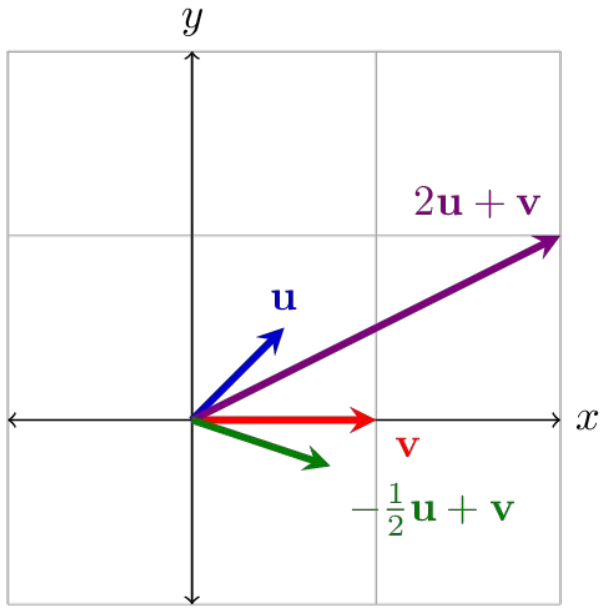
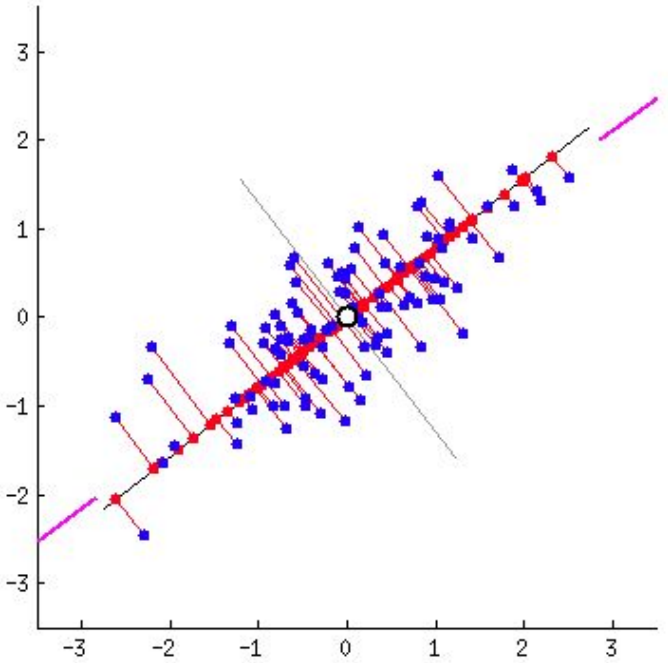


Figure 1: Vector combinations.

PCA

Найдём такой базис, чтобы как можно лучше выразить как можно больше значений за счёт фиксированного количества базисных векторов.

Сделаем проекцию всех данных на эти вектора.



SVD

Давайте вспомним умножение матриц.

(Источник картинки)

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

При матричном умножении, каждая **строка первой матрицы** (слева) “умножается” на каждый **столбец второй матрицы** (справа).

“Умножается” — значит, берётся скалярное произведение векторов.

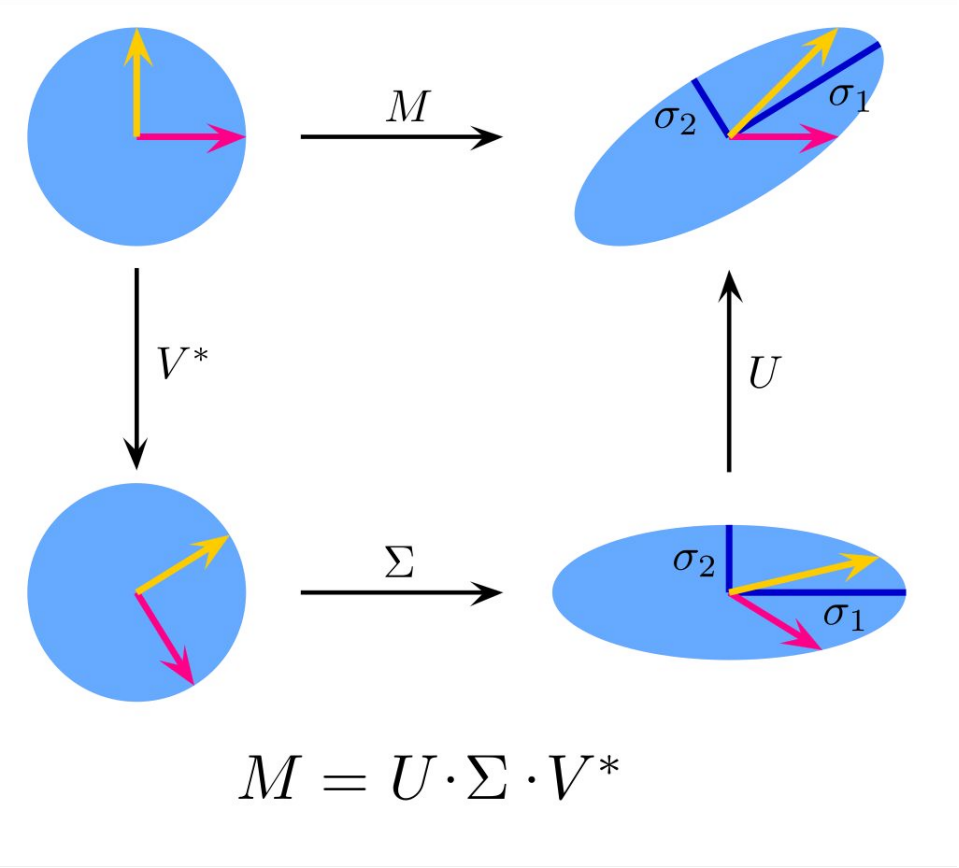
SVD

Любую матрицу M можно разложить на произведение трёх матриц: $M = U * \Sigma * V^*$

U, V^* — матрицы поворота

Σ — матрица растяжения

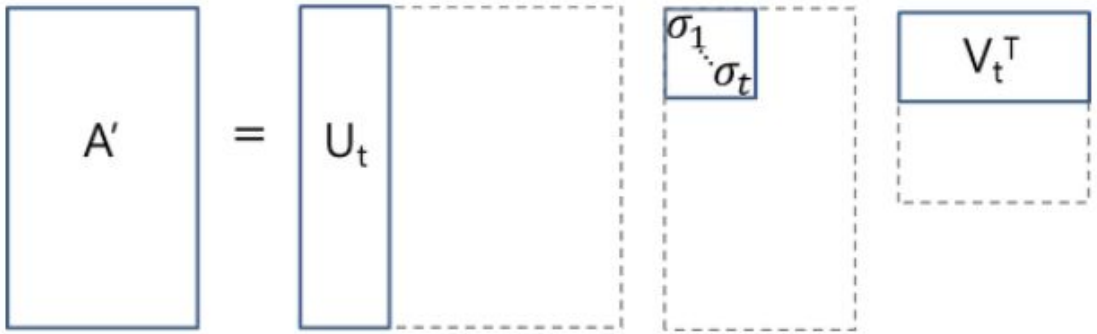
У Σ числа стоят только на главной диагонали, причём они убывают



Truncated SVD

$$A \approx U_t S_t V_t^T$$

Intuitively, think of this as only keeping the t most significant dimensions in our transformed space.

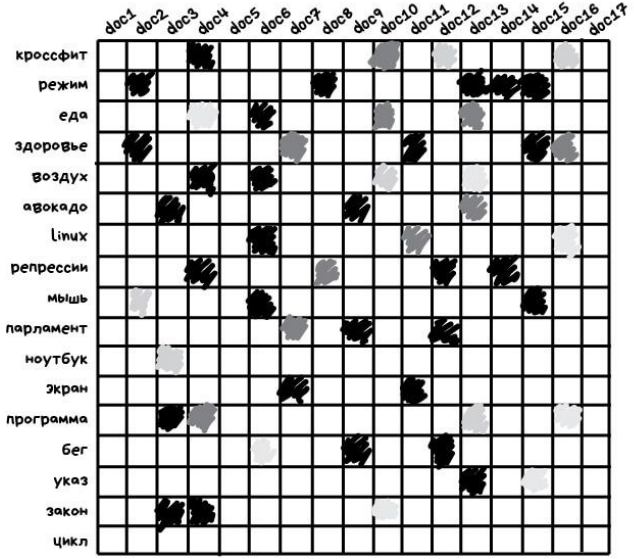


(скрин из вот этой статьи)

Truncated SVD = LSA (latent semantic analysis) in topic modeling

Разделение документов по темам

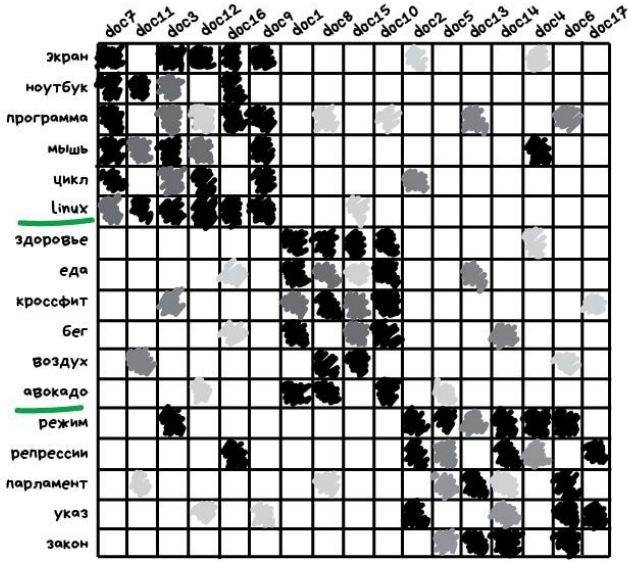
(источник)



1. Строим матрицу как часто каждое слово встречается в каждом документе (чернее - чаще)

SVD

2. Раскладываем



3. Получаем наглядные кластера по тематикам (даже если слова не встречались вместе)

Латентно-семантический Анализ (LSA)

На собачках

При большом количестве компонент разница незаметна.

(источник)

Truncated SVD в sklearn

from sklearn.decomposition import TruncatedSVD

гиперпараметры:

- n_components — какого размера должны быть конечные векторы
- algorithm — randomized, arpack
- n_iter

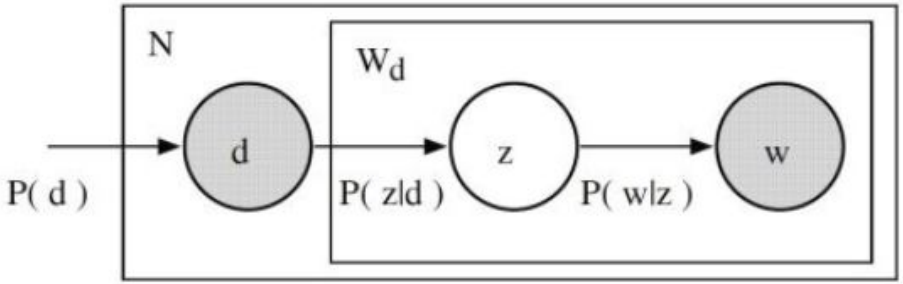
Может применяться в связке с классификацией.

pLSA

Probabilistic Latent Semantic Analysis

- given a document d, topic z is present in that document with probability $P(z|d)$
- given a topic z, word w is drawn from z with probability $P(w|z)$

Источник картинки



Вероятность встретить слово W и документ D: $P(D,W) = P(D)\sum_Z P(Z|D)P(W|Z)$

Как это работает?

P(D), P(Z|D) и P(W|Z) — параметры нашей модели

P(D) — находится из корпуса

P(Z|D) и P(W|Z) оптимизируются с помощью EM алгоритма (expectation-maximization)

“EM is a method of finding the **likeliest parameter estimates** for a model which depends on unobserved, latent variables (in our case, the topics)”

Что ещё бывает?

- TSNE — t-distributed Stochastic Neighbor Embedding
- NMF — тоже про разложения матриц

Используется в тематическом моделировании:

- LDA (a Bayesian version of pLSA)
- ARTM — LDA, но с регуляризацией
- bigARTM — ARTM с наворотами)))

Ресурсы

Почитать

- Machine Learning for Humans: Unsupervised Learning
- The 5 Clustering Algorithms Data Scientists Need to Know
- Topic Modeling with LSA, PLSA, LDA & lda2Vec
- про кластеризацию на Хабрахабр (рус)

Посмотреть

- StatQuest: k-means
- StatQuest: hierarchical clustering
- StatQuest: PCA