Обработка и визуализация данных. Матричные разложения

Цель занятия

В результате обучения на этой неделе вы:

- узнаете основные методы обработки данных, упрощающих решение задач машинного обучения
- познакомитесь ближе с библиотекой matplotlib, позволяющей визуализировать данные
- познакомитесь с различными видами матричного разложения, используемыми в задачах машинного обучения

План занятия

Используемые термины

Данный пункт опционален. Если вы считаете, что необходимо вставить определение термина предыдущего занятия или модуля, то продублируйте термин. Вместе тем, не стоит увеличивать данный пункт и добавлять в него избыточные термины.

Предпочтительно давать краткие и ёмкие определения. Если определение термина заимствовано из другого источника, укажите ссылку на ресурс.

Конспект занятия

1. Задача снижения размерности

В задаче снижения размерности мы говорим про снижение размерности признакового пространства, в котором находятся все объекты, с которыми мы будем работать.

Рассмотрим некоторую задачу, в которой многомерная выборка. В ней сотни или тысячи признаков данных. Это вызывает несколько проблем, так как далеко не все модели позволят нам качественно работать с такими данными.

Какие могут быть проблемы?

- Эти данные **сложно визуализировать**. Мы не сможем адекватно визуализировать 1000-мерное пространство, не имея специальной математической подготовки.
- При работе с высокоразмерными данными модели могут **медленно обучаться**. Это может быть не только долго, но и дорого, как на этапе обучения, так и на этапе инференса.
- Некоторые модели хуже работают в случае обработки многомерных данных.

NB. **Проклятие размерности**. Почему не выгодно продавать бесконечномерные арбузы? Представим себе, что у нас есть некоторое пространство, которое равномерно заполнено точками. Начнем с двумерного пространства. У нас есть сетка, и на пересечении линий находятся точки. Что такое арбуз? Есть центр, вокруг него мякоть (арбуз считаем шаром), на границе с ненулевой толщиной есть корка. Площадь корки и площади мякоти (в двумерном случае) друг с другом как-то соотносятся. В трехмерном случае это уже объем мякоти и объем корки. Соотношение поменяется. При увеличении размерности отношение будет меняться. С какого-то момента обнаружится, что почти весь объем фокусируется в корке (можно взять предел от интеграла – объема корки). Поэтому бесконечномерные арбузы продавать невыгодно, никто не будет покупать только корку, когда мякоти бесконечно мало.

Вернемся к миру машинного обучения. Если у нас в пространстве точки распределены равномерно, то в случае большой размерности, это точки будут находиться примерно на одинаковом удалении от центра. А значит с помощью метрических алгоритмов мы не сможем их отличить.

Большая размерность – это зачастую сложно вычислительно и сложно с точки зрения математики, перестают работать некоторые методы.

2. Метод главных компонент

Понижение размерности

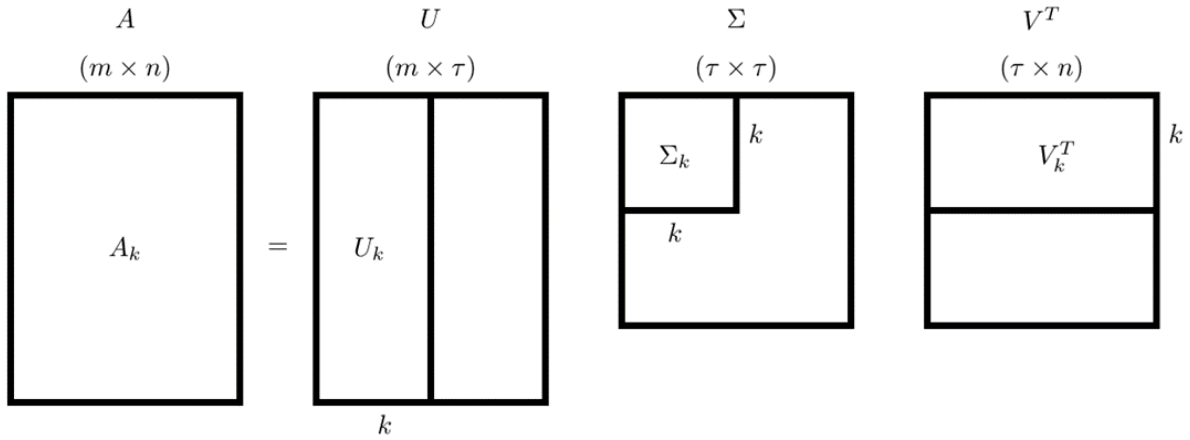
Данных может быть много. Признаковое пространство может быть очень высокой размерности. Мы можем обратиться к линейным методам для снижения размерности. Что можно сделать с матрицей A (матрица, описываемая нашими признаками) для снижения размерности признакового пространства? Её можно

разложить на несколько матриц. Матричные операции все линейные. И можно попытаться использовать матрицу меньшего ранга для описания данных.

Мы хотим матрицу A разложить на три матрицы:

$$A = U\Sigma V^T$$

$$A_k = U_k\Sigma_k V_k^T = \left(U_k\Sigma_k\right)V_k^T = U_k\left(\Sigma_k V_k^T\right)$$



Дополнительные ограничения:

$$U \in R^{m\times r};\ UU^T = I$$

$$V \in R^{n\times r};\ VV^T = I$$

$$\Sigma = diag(\sigma_1, \dots, \sigma_r);\ r = rank(M)$$

Матрицы U и V ортогональные, то есть они являются обратными для самих себя в случае транспонирования. Из этого следует, что столбцы у них линейно независимы, норма у матриц U и V равна единице.

Если матрица обладает линейно независимыми столбцами, с единичной нормой, то это матрица поворота. То есть поворачивает наше признаковое пространство некоторым образом.

Итак, U и V – матрицы поворота.

Σ – диагональная матрица, на её диагонали стоят некоторые числа. При умножении на такую матрицу мы будем растягивать или сжимать различные направления в нашем пространстве. Если U и V – ортогональные матрицы, то каждому вектору соответствует лишь один диагональный элемент из матрицы Σ.

То есть по сути мы разбиваем любое преобразование на поворот (матрицы U и V) и сжатие/растяжение (матрица Σ). В этом случае мы можем попытаться выбрать для себя оптимальные направления, вдоль которых у нас максимальная информация.

Например, вдоль этих направлений максимальная дисперсия. После чего можно использовать эти направления с наибольшей дисперсией, а все остальные игнорировать.

Теперь мы можем понять, вдоль какого направления данные максимально разнородные, больше всего не похожи друг на друга, то есть лучше всего сможем различать точки между собой. При проекции на данное направление точки будут реже всего попадать в одну и ту же координату.

Поскольку вектора матриц U и V линейно независимы, то дисперсию мы можем измерять с помощью соответствующей σ. Более того, мы можем упорядочить σ по невозрастанию. То есть $\sigma_1 \geq \sigma_2$, $\sigma_2 \geq \sigma_3$, и т.д.

Если мы хотим получить k-мерное представление наших данных, нам достаточно взять первые k векторов матрицы U и соответствующие для них первые k сингулярных значений из матрицы Σ и домножить на первые k векторов матрицы V. Тогда мы получим некоторое приближение исходной матрицы A, теперь ранга k, и при этом оно будет оптимальным с точки зрения минимизации дисперсии.

Теорема (Экарта-Янга). Сингулярное разложение (SVD) дает нам оптимальное представление матрицы высокого ранга матрицей более низкого ранга k:

$A_k = U_k \Sigma_k V_k^T$

$\forall B_k: rank(B_k) = k$

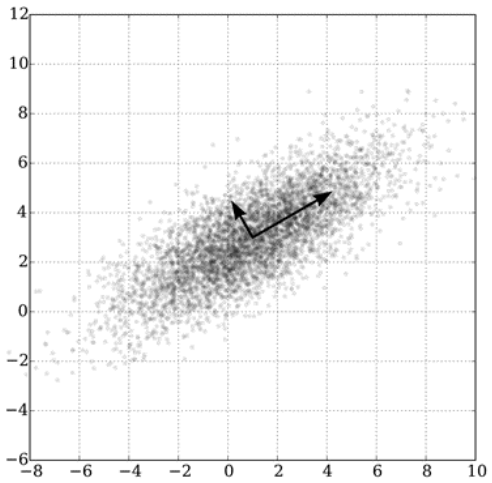
$||A - B_k||_F \geq ||A - A_k||_F$ – ошибка по норме Фрабениуса.

Норма Фрабениуса – сумма квадратов разности элементов матриц.

PCA

Что это дает для машинного обучения?

Допустим у нас есть некоторые данные:



Допустим, данные обладают некоторой структурой в пространстве. В данном случае это эллипс. Направление наибольшей дисперсии – направление главной (большой) полуоси. Вторая главная компонента направлена вдоль малой полуоси.

Что делать в случае, когда главные компоненты определены неоднозначно? Например, в случае центральной симметрии. Тут можно выбрать любое направление, как первую главную компоненту, а вторая будет ей ортогональна.

Метод главных компонент (PCA) по факту просто ищет направление наибольшей дисперсии последовательно.

Глядя на разложение

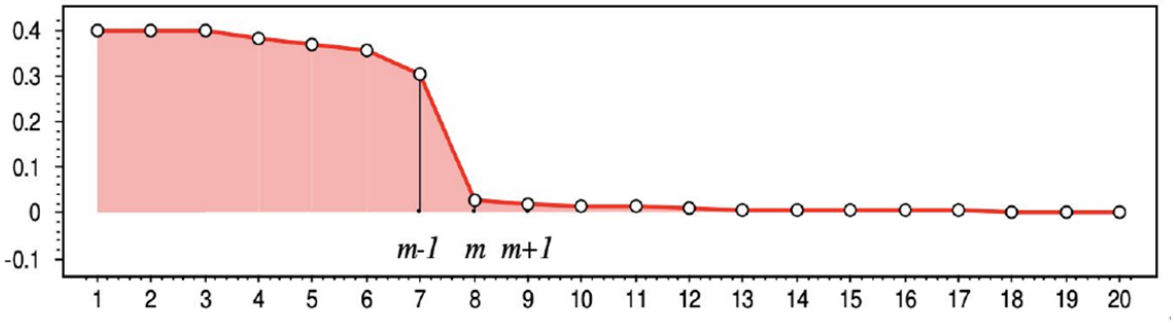
$X = U\Sigma V^T$,

можно понять, что

- матрица V указывает нам на оптимальные направления
- $U\Sigma$ – новые главные компоненты, которые описывают наши данные.

Если мы хотим с помощью k-мерного представления выделить наиболее информативное k-мерное подпространство из нашего признакового пространства, нужно выбрать k компонент с наибольшей дисперсией, а все остальные занулить.

Как выбрать это оптимальное k? Если данных много, то можно построить PCA по всем компонентам. Тогда мы уловим всю дисперсию. А потом можно посмотреть на матрицу Σ.



Мы увидим, что элементы матрицы Σ могут убывать некоторым образом. Здесь получается, что данные находятся в m-мерном подпространстве, которое содержит почти всю информацию о наших данных. Небольшие потери – цена за то, чтобы снизить размерность.

Из каких соображений выбирают число m?

$$E_m = \frac{\left\|GU^T - F\right\|^2}{\|F\|^2} = \frac{\lambda_{m+1} + \ldots + \lambda_n}{\lambda_1 + \ldots + \lambda_n} \leq \varepsilon$$

Смотрят, сколько элементов нужно отбросить, чтоб ошибка все еще не превышала заданное значение. Часто это называют **методом складного ножа**.

Таким образом, строят PCA на все компоненты и смотрят, сколько компонент можно отбросить, чтобы их суммарный вклад в дисперсию был меньше, чем допустимая ошибка. Допустимую ошибку выбираем сами.

Применение на практике PCA

Рассмотрим применение PCA на практике.

- PCA зависит от того, в каких шкалах наши данные.

Важно! Отнормируйте и отцентрируйте ваши данные!

- PCA позволяет снизить размерность, и мы уже не получаем исходное признаковое представление точек. Мы получаем представление уже в новом признаковом пространстве.

$X_k = U_k \Sigma_k$

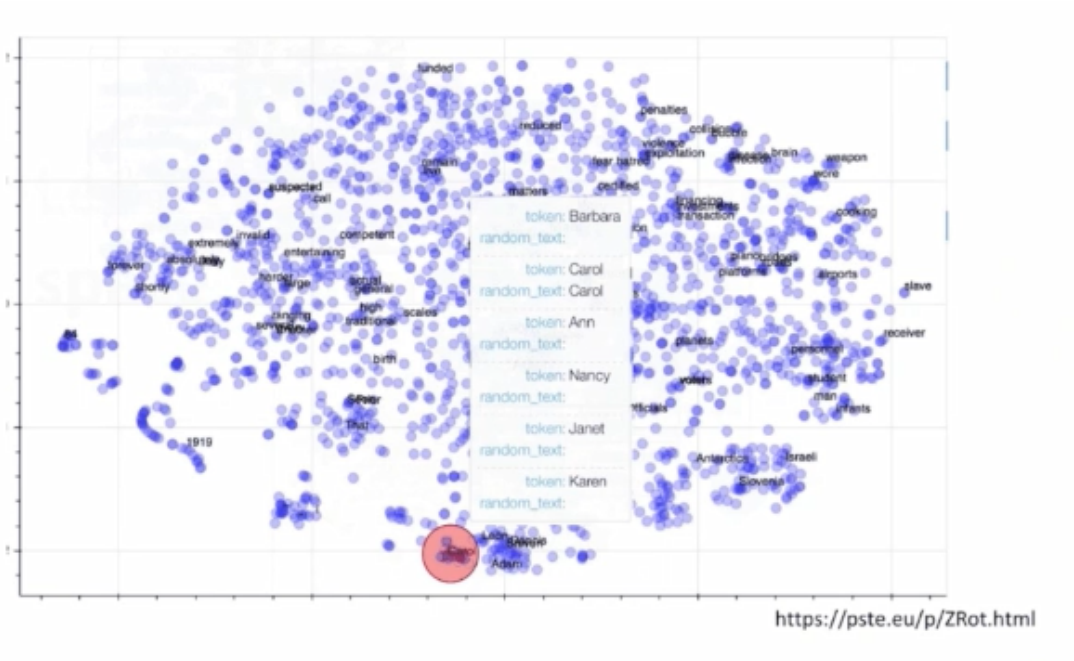
Каждые k признаков являются линейной комбинацией исходных (поворот, сжатие/растяжение)

- . Для возвращения в исходное пространство понадобится матрица V:

$$\overline{X} = U_k \Sigma_k V_k^T$$

Но исходные данные будут восстановлены с некоторой ошибкой.

Пример 1. Есть множество точек. Каждая точка – векторное представление слова, которое из 300-мерной размерности было приведено к 2-мерной картинке[1]:



Все эти точки были построены на основании контекстной близости слов.

Пример 2. PCA позволит снизить размерность данных, даже если это картинки[2]. Каждая черно-белая картинка – это матрица.

[1] https://github.com/yandexdataschool/nlp_course/tree/2019/week01_embeddings

[2] https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

С помощью SVD снизим размерность матрицы. Берем первые 16 компонент:

Если взять 50 компонент[3]:

[3] https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

Если увеличить количество компонент до 250[4]:

Таким образом, PCA – замечательный метод, который позволяет снизить размерность данных, и при этом он линейный и он чувствителен к шкалам наших данных (также как и метод ближайших соседей, как и L1 и L2 регуляризации).

4 https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184

3. Матричные разложения

Для вставки формул используйте инструмент Вставка → Формула. Поясните буквенные обозначения в формуле. Если формула будет далее упоминаться в конспекте, присвойте формуле номер и по тексту делайте ссылку на номер формулы.

$$ax^2 + bx + c = 0, \quad (1)$$

где *a*, *b*, *c* — коэффициенты квадратного уравнения, *x* — неизвестная переменная.

Дополнительные материалы для самостоятельного изучения

1. https://github.com/yandexdataschool/nlp_course/tree/2019/week01_embeddings
2. https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184
3. https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184
4. https://www.deeplearningbook.org/