

# Лабораторная работа 1: Алгоритмы разложения матриц. РСА.

**Цель работы:** Использование методов матричного разложения в алгоритмах обработки данных (метода главных компонент).

**Описание:** В этой лабораторной работе сделан акцент на использовании методов матричного разложения для решения прикладной задачи - использования метода главных компонент (РСА) для анализа произвольного массива данных. Помимо этого предлагается реализовать метод Kernel PCA для решения задачи классификации линейной-неразделимых данных.

**Предлагаемые методы:** При выполнении данной лабораторной работы предлагается использовать метод главных компонент для анализа массива входных данных. Основной идеей данного алгоритма является снижение размерности анализируемого объекта путём линейного преобразования входных данных в новую координатную систему таким образом, что при помощи меньшего числа измерений можно описать большую дисперсию входных данных. Данное линейное преобразование можно представить следующим образом:

Пусть  $X = \mathbf{x}_1, \dots, \mathbf{x}_n^T$  - матрица входных данных, где  $\mathbf{x}_i$  - вектор длины  $m$ , описывающий  $i$ -ую запись входных данных. Матрица входных данных должна быть центрирована (по каждому признаку):  $x'_{ij} = x_{ij} - 1/n \sum_k x_{kj}$ , где  $i$  - индекс вектора данных, а  $j$  - индекс признака. Линейное преобразование переводит вектор записи входных данных в новую форму  $\mathbf{t} = (t_1, \dots, t_s)$ , где  $s$  - параметр, определяющий, на сколько главных компонент проецируются данные, а  $W$  - матрица линейного преобразования, размерности  $m \times s$ .

$$\mathbf{t}_i = \mathbf{x}_i W \quad (1)$$

Задача определения главных компонент допускает использование собственных значений и собственных векторов матрицы ковариации. Требование ортогональности и задание максимальной дисперсии при помощи компонент приводит к тому [для доказательства см. источники, например [2]], что  $w_j$  соответствуют собственным векторам матрицы  $X^T X$ . Вклад  $j$ -ой компоненты в описание дисперсии данных пропорционален отношению сингулярного числа  $\sigma_j$  к сумме сингулярных чисел  $\sum_k \sigma_k$  матрицы  $X^T X$ .

## Использование матричных разложений:

Нахождение главных компонент, описывающих данные, при помощи собственных векторов/значений нормальной матрицы  $X^T X$ , можно произвести через матричные разложения.

Подобный подход предполагает сингулярное разложение матрицы данных  $X$ , имеющей размерности  $n \times m$ . Сингулярное разложение является обобщением спектрального разложения на прямоугольные матрицы. Матрица измерений представляется через матричное произведение (2).

$$X = U \Sigma V^T \quad (2)$$

Здесь  $\Sigma = \text{Diag}\{\sigma_1, \dots, \sigma_p\}$ ,  $p = \min(m, n)$  - прямоугольная диагональная матрица размерности  $n \times m$ , где на диагонали находятся неотрицательные числа  $\sigma_i$ ,

называемые сингулярными числами. Сингулярные числа определяются через собственные значения нормальной матрицы  $XX^T$ , которые мы обозначим как  $\lambda_i(XX^T)$ :

$$\sigma_i(X) = \sqrt{\lambda_i(XX^T)} \quad (3)$$

$U$  - квадратная матрица  $n \times n$ , которая содержит “левые сингулярные векторы” матрицы  $X$ , которые соответствуют собственным векторам матрицы  $XX^T$ .

Аналогично определяется матрица  $V$ , содержащая “правые сингулярные векторы” (соответствуют собственным векторам матрицы  $X^TX$ ).

Вычисление сингулярных векторов и сингулярных чисел для матрицы  $X$  можно производить следующим образом. Для определения сингулярных значений матрицы нужно вычислить собственные значения матрицы  $XX^T$ . Левые и правые сингулярные векторы определяются через собственные векторы матриц  $XX^T$  и  $X^TX$ .

### QR-алгоритм

Одним из эффективных методов вычисления собственных векторов и чисел является QR-алгоритм, основанный на последовательном применении QR-разложения к матрице.

QR-разложение представляет матрицу (в общем случае прямоугольную матрицу  $A \in \mathbb{C}^{m \times n}$ ,  $m \geq n$ ) как произведение унитарной матрицы (то есть матрицы с ортогональными столбцами)  $Q$  и верхней треугольной матрицы  $R$ . В матричной форме оно принимает вид:

$$A = QR \quad (4)$$

$$\begin{bmatrix} | & | & \dots & | \\ a_1 & a_2 & \dots & a_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & \dots & | \\ q_1 & q_2 & \dots & q_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}, \quad (5)$$

где  $a_i \in \mathbb{C}^m$ ,  $i = 1, \dots, n$ ,  $q_i \in \mathbb{C}^m$ ,  $i = 1, \dots, n$  - ортогональные векторы:  $q_i \cdot q_j = 0$ ,  $i \neq j$ ,  $r_{ii} \neq 0$ . Алгоритм матричного разложения основан на методе ортогонализации Грама-Шмидта:

```

for  $i = 1$  to  $n$  do
    |  $v_i = a_i$  ;
end
for  $i = 1$  to  $n$  do
    |  $r_{ii} = |v_i|$  ;
    |  $q_i = v_i / r_{ii}$  ;
    for  $j = i + 1$  to  $n$  do
        |  $r_{ij} = q_i^* v_j$  ;
        |  $v_j = v_j - r_{ij} q_i$  ;
    end
end

```

**Algorithm 1:** Модифицированный алгоритм на основе ортогонализации Грама-Шмидта.

В общем случае, QR-алгоритм применим к матрицам Хессенберга: необходимо ввести преобразование исходной матрицы, для которой ищем собственные вектора/числа. Описания QR-алгоритма и QR-разложения в полной форме представлены в книге [3].

## Ход работы и задачи:

### 1. Метод главных компонент:

- Реализуйте сингулярное матричные разложения (не используя готовые решения вроде `numpy.linalg.svd`, или `numpy.linalg.eig` для поиска собственных векторов и чисел), рекомендуемый метод определения собственных чисел и векторов - QR-алгоритм;
- Выберите произвольную задачу машинного обучения на размеченной выборке, содержащую минимум 5 признаков, выберите подходящую модель для решения задачи МО (искусственную нейронную сеть);
- Используйте написанный метод сингулярного разложения для получения матрицы преобразований данных к главным компонентам и значения объясняемых дисперсий;
- Определите достаточное число компонент для описания процесса, визуализируйте данные (в т.ч. отобразите распределение целевой переменной в пространстве признаков через диаграммы рассеяния), проведите анализ полученных компонент, если получается выделить "закономерности"/физический смысл;
- Оцените изменения в качестве обученных моделей МО после преобразования данных. Определите изменения во времени обучения;
- (Дополнительное задание) Реализуйте методы сингулярного разложения матрицы, имеющие меньшую вычислительную сложность, чем классический подход (основанный на собственных векторах и собственных числах матрицы  $X^T X$ ).

### 2. Kernel PCA:

- Выберите произвольный набор линейно-неразделимых данных для задачи классификации;
- Реализуйте методы вычисления матрицы для различных ядер (например, на основе полиномиального, радиальной базисной функции, сигмоидального ядер) и её последующего спектрального разложения;
- Проведите сравнительный анализ применения PCA и Kernel PCA к данным, сравните проекции на первые главные компоненты при использовании разных ядер, проверьте линейную разделимость. Проверьте, преобразованы ли данные путём к линейно-разделимой форме.
- Оцените изменение значений метрик качества классификации на исходных данных и на преобразованных.

## Список литературы

- [1] Banerjee, S., Roy, A., Linear Algebra and Matrix Analysis for Statistics (1st ed.), Chapman and Hall/CRC, 2014, <https://doi.org/10.1201/b17040>.
- [2] I. T. Jolliffe, Principal Component Analysis, Springer New York, NY, 2002, <https://doi.org/10.1007/b98835>.
- [3] Trefethen, L.N. and Bau, D., Numerical Linear Algebra. 1997, Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics

- [4] Menon, Aditya Krishna and Elkan, Charles, “Fast Algorithms for Approximating the Singular Value Decomposition”, Association for Computing Machinery, New York, NY, USA, 2011. year = 2011,
- [5] Schölkopf, B., Smola, A., Müller, KR. Kernel principal component analysis. Springer, Berlin, Heidelberg, 1997, <https://doi.org/10.1007/BFb0020217>.
- [6] Quan Wang, Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models, 2014, <https://doi.org/10.48550/arXiv.1207.3538>.