**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра технической кибернетики

Дисциплина: Теория матриц

Практическая работа № 5

Тема: «Сингулярное разложение матрицы»

Выполнил:

Студент группы МТК-233

Орлов-Куреши М. Н.

Проверил:

Кариков Е. Б.

Белгород 2023

**Цель работы:** изучить сингулярное разложение. Реализовать сингулярное разложение на языке программирования Python.

**Сингулярное разложение**

Сингулярное разложение – это разложение прямоугольной вещественной или комплексной матрицы, имеющее широкое применение, в силу своей наглядной геометрической интерпретации, при решении многих прикладных задач.

Пусть матрица *M* размера *m×n* (*m* строк на *n* столбцов) состоит из элементов поля *K*, где *K* – либо поле вещественных чисел, либо поле комплексных чисел.

Неотрицательное вещественное число *σ* называется сингулярным числом матрицы *M*, если и только если существуют два вектора единичной длины *u*∈*Km* и *v*∈*Kn* такие, что:

*Mv* = *σu* и *M\*u* = *σv*.

Такие векторы *u* и *v* называются, соответственно, левым сингулярным вектором и правым сингулярным вектором, соответствующим сингулярному числу *σ*.

Сингулярные числа матрицы и ее сингулярные собственные векторы не следует путать с обыкновенными собственными числами и собственными векторами той же матрицы *M*.

Сингулярные числа матрицы *M* вычисляются:

– как собственные числа матрицы *MM\**, если размеры матрицы *M* связаны соотношением *m*≤*n* (если число строк меньше или равно числу столбцов) или

– как собственные числа матрицы *M\*M*, если размеры матрицы *M* связаны соотношением *m*>*n* (если число строк больше числа столбцов).

Левые сингулярные собственные векторы матрицы *M* – это собственные векторы матрицы *MM\**.

Правые сингулярные собственные векторы матрицы *М* – это собственные векторы матрицы *M\*M*.

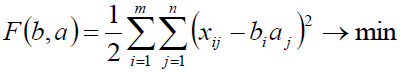
Сингулярным разложением матрицы *M* порядка *mn* является разложение следующего вида:

*M* = *UΣV\**

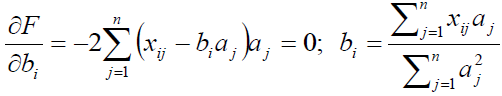
где *Σ* – размера *m*×*n*, у которой элементы, лежащие на главной диагонали – это сингулярные собственные числа (а все элементы, не лежащие на главной диагонали, являются нулевыми), а матрицы *U* (порядка *m*) и *V* (порядка *n*) – это две унитарные матрицы, состоящие из левых и правых сингулярных векторов соответственно (а *V\** – это сопряжѐнная матрица к *V*).

**Простой итерационный алгоритм сингулярного разложения**

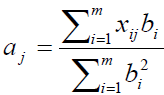
Основная процедура-поиск наилучшего приближения произвольной *m*×*n* матрицы *X* = (*xij*) матрицей вида *baT* = (*biaj*), где *b* – *m*-мерный вектор-столбец, а *a* – *n*-мерный вектор-столбец, методом наименьших квадратов:

.

Решение этой задачи дается последовательными итерациями по явным формулам. При фиксированном векторе *a* = (*aj*) значения *b* = (*bi*), доставляющие минимум форме *F*(*b*,*a*), однозначно и явно определяются из равенств *∂F*/*∂bi* = 0:

.

Аналогично, при фиксированном векторе *b* = (*bi*) определяются значения *a* = (*aj*):

.

В качестве начального приближения вектора *a* берется случайный вектор единичной длины, вычисляем вектор *b*, далее для этого вектора *b* вычисляем вектор *a* и т.д. Каждый шаг уменьшает значение *F*(*b*,*a*). В качестве критерия остановки используется малость относительного уменьшения значения минимизируемого функционала *F*(*b*,*a*) за шаг итерации (Δ*F*/*F*) или малость самого значения *F*.

В результате для матрицы *X* = (*xij*) получается наилучшее приближение матрицей *P1* вида  (здесь верхним индексом обозначен номер приближения). Далее, из матрицы *X* вычитается полученная матрица *P1*, и для полученной матрицы уклонений *X1* = *X* – *P1* вновь ищется наилучшее приближение *P2* этого же вида и т.д., пока, например, норма (определитель) *Xk* не станет достаточно малой. В результате получается итерационная процедура разложения матрицы *X* в виде суммы *q* матриц ранга *1*, то есть

.

Полагаем



и нормируем векторы *al*, *bl*:

.

В результате получена аппроксимация сингулярных чисел *σl* и правых *al* и левых *bl* сингулярных векторов.

К достоинствам этого алгоритма относится его простота и возможность почти без изменений перенести его на данные с пробелами, а также взвешенные данные.

Существуют различные модификации базового алгоритма, улучшающие точность и устойчивость. Например, векторы главных компонент *al* при разных *l* должны быть ортогональны по «построению», однако при большом числе итерации (большая размерность, много компонент) малые отклонения от ортогональности накапливаются и может потребоваться специальная коррекция *al* на каждом шаге, обеспечивающая его ортогональность ранее найденным главным компонентам.

Для квадратных симметричных положительно определенных матриц описанный алгоритм превращается в метод прямых итераций для поиска собственных векторов.

**Реализация алгоритма на языке программирования Python**

class SVDMixin:

    @staticmethod

    def F(X, a, b):

        err = 0

        for i in range(X.row):

            for j in range(X.column):

                err += (X[i, j] - b[i, 0] \* a[j, 0]) \*\* 2

        return err / 2

    def svd(self):

        n = self.row

        m = self.column

        X = self.copy()

        u = Matrix(m, 1, [random.randint(0,1) for i in range(m)])

        v = Matrix(n, 1, [0]\*n)

        U = Matrix(m, 0)

        V = Matrix(n, 0)

        s = Matrix(n, m, [0] \* (n\*m))

        u\_list = []

        v\_list = []

        s\_list = []

        last = 1

        curr = 0

        for \_ in range(min(n,m)):

            u = Matrix(m, 1, [random.randint(1,1) for i in range(m)])

            last = 1

            curr = 0

            while (last-curr) > 0.0000001:

                last = SVDMixin.F(X, u, v)

                for i in range(n):

                    sum\_x\_u = 0

                    sum\_u2 = 0

                    for j in range(m):

                        sum\_x\_u += X[i, j] \* u[j, 0]

                        sum\_u2 += u[j, 0] \*\* 2

                    v[i, 0] = sum\_x\_u / sum\_u2

                for i in range(m):

                    sum\_x\_v = 0

                    sum\_v2 = 0

                    for j in range(n):

                        sum\_x\_v += X[j, i] \* v[j, 0]

                        sum\_v2 +=  v[j, 0] \*\* 2

                    u[i, 0] = sum\_x\_v / sum\_v2

                curr = SVDMixin.F(X, u, v)

            X = X - (v \* u.T)

            s\_list.append(u.norm() \* v.norm())

            u.divide\_by\_num(u.norm())

            v.divide\_by\_num(v.norm())

            u\_list.append(u.copy())

            v\_list.append(v.copy())

        for i in range(m):

            U.add\_column(u\_list[i])

        for i in range(min(m,n)):

            V.add\_column(v\_list[i])

        s\_list = s\_list[:min(n,m)]

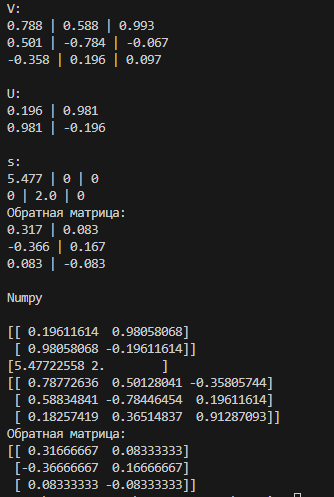
        s\_list.sort()

        for i in range(min(n,m)):

            s[i, i] = s\_list[i]

        return U, s, V

**Скриншоты работы программы**



**Вывод:** в ходе работы было изучено и реализовано сингулярное разложение. Также было реализовано нахождение псевдообратной матрицы при помощи сингулярного разложения.

**Список литературы**

1. Юдин Д.А. Прикладные аспекты теории матриц: учебное пособие / Д.А. Юдин. - Белгород: Изд-во БГТУ, 2016.