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 $m \times n$ (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 $u \in K^m$ и $v \in K^n$ такие, что:

$Mv = \sigma u$ и $M^* u = \sigma v$.

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 $m \leq n$ (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 M – это собственные векторы матрицы M^*M.

Сингулярным разложением матрицы M порядка mn является разложение следующего вида:

$M = U\Sigma V^*$

где Σ – размера $m\times n$, 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 $m\times n$ матрицы $X = (x_{ij})$ матрицей вида $ba^T = (b_i a_j)$, где b – m-мерный вектор-столбец, а a – n-мерный вектор-столбец, методом наименьших квадратов:

$$F(b,a) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{ij} - b_i a_j\right)^2 \to \min.$$

Решение этой задачи дается последовательными итерациями по явным формулам. При фиксированном векторе $a = (a_j)$ значения $b = (b_i)$, доставляющие минимум форме $F(b,a)$, однозначно и явно определяются из равенств $\partial F/\partial b_i = 0$:

$$\frac{\partial F}{\partial b_i} = -2\sum_{j=1}^{n}\left(x_{ij} - b_i a_j\right)a_j = 0; \quad b_i = \frac{\sum_{j=1}^{n} x_{ij} a_j}{\sum_{j=1}^{n} a_j^2}.$$

Аналогично, при фиксированном векторе $b = (b_i)$ определяются значения $a = (a_j)$:

$$a_j = \frac{\sum_{i=1}^{m} x_{ij} b_i}{\sum_{i=1}^{m} b_i^2}.$$

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 ($\Delta F/F$) или малость самого значения F.

В результате для матрицы $X = (x_{ij})$ получается наилучшее приближение матрицей P_1 вида $b^1 \cdot (a^1)^T = (b_i^1 a_j^1)$ (здесь верхним индексом обозначен номер приближения). Далее, из матрицы X вычитается полученная матрица P_1, и для полученной матрицы уклонений $X_1 = X - P_1$ вновь ищется наилучшее приближение P_2 этого же вида и т.д., пока, например, норма (определитель) X_k не станет достаточно малой. В результате получается итерационная процедура разложения матрицы X в виде суммы q матриц ранга 1, то есть

$$X = P_1 + P_2 + \ldots + P_l + \ldots + P_q, \quad P_l = b^l \cdot (a^l)^T, \, l = 1, 2, \ldots, q.$$

Полагаем

$$\sigma_l = |a^l| \cdot |b^l|$$

и нормируем векторы a^l, b^l:

$$a^l = a^l / |a^l|; \; b^l := b^l / |b^l|.$$

В результате получена аппроксимация сингулярных чисел σ_l и правых a^l и левых b^l 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 a^l 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 a^l 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

```
V:
0.788 | 0.588 | 0.993
0.501 | -0.784 | -0.067
-0.358 | 0.196 | 0.097

U:
0.196 | 0.981
0.981 | -0.196

s:
5.477 | 0 | 0
0 | 2.0 | 0
Обратная матрица:
0.317 | 0.083
-0.366 | 0.167
0.083 | -0.083

Numpy

[[ 0.19611614  0.98058068]
 [ 0.98058068 -0.19611614]]
[5.47722558 2.        ]
[[ 0.78772636  0.50128041 -0.35805744]
 [ 0.58834841 -0.78446454  0.19611614]
 [ 0.18257419  0.36514837  0.91287093]]
Обратная матрица:
[[ 0.31666667  0.08333333]
 [-0.36666667  0.16666667]
 [ 0.08333333 -0.08333333]]
```

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.