РТУ МИРЭА

«Разложение матриц по сингулярным значениям»

Документация к проекту

Руководители проекта:

Парфенов Денис Васильевич

Дроздов Игорь Юрьевич

Участники:

Абрамов Семён, КМБО-01-20

Виноградова Арина, КМБО-01-20

Грузберг Александр, КМБО-01-20

Едренников Данила, КМБО-01-20

Искенеева Камила, КМБО-01-20

Ишутин Андрей, КМБО-01-20

Каргаполов Руслан, КМБО-01-20

Корешкова Виктория, КМБО-01-20

Кулибаба Данил, КМБО-01-20

Мустафин Рамиль, КМБО-01-20

Нам Александр, КМБО-04-20

Савельев Марк, КМБО-01-20

Романцов Андрей, КМБО-01-20

1.1.2024

Оглавление

Глава 1. Введение

Сингулярные значения матрицы представляют собой важное понятие в области линейной алгебры и анализа данных. В широком смысле они позволяют выделить ключевые направления или "факторы" в данных тем самым “сжимать” информацию, выделяя наиболее значимые признаки. Таким образом, сингулярные значения помогают находить основные закономерности и существенную информацию в больших объемах данных.

Сингулярные значения широко используются в различных областях благодаря своей способности выделять ключевые особенности в данных. Среди задач, в которых используется SVD можно выделить нахождение решений систем линейных уравнений, сжатие изображений, выделение признаков, проектирование рекомендательных систем. В анализе текстов SVD используются для выделения тем и структур в больших текстовых корпусах. Эти области применения подчеркивают важность сингулярных значений в обработке данных и раскрывают их универсальность в различных научных и прикладных задачах.

Цели

1. Изучить и проанализировать современные методы и алгоритмы разложения матрицы по сингулярным значениям с математической точки зрения.
2. Выявить наиболее эффективные методы и алгоритмизировать их.
3. Провести необходимые тесты для проверки действия алгоритмов.

Технические подробности

Для реализации методов был использован C++.

Среда разработки – Visual Studio.

Реализация алгоритмов представлена на /SVD-project

Временные рамки проекта

VII-VIII семестр обучения студентов – бакалавров направления «Прикладная математика и информатика».

Глава 2. Общие теоретические сведения

Будем рассматривать произвольный линейный оператор $A\colon \mathbb{R}^n \to \mathbb{R}^m$, который в дальнейшем будем отождествлять с его матрицей.

Теорема 1. Пусть $M \in L(V, W), \dim V = n, \dim W = m, p = \min\{n, m\}$. Тогда существуют такие числа $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ и ортонормированные базисы $\{\boldsymbol{v_1}, \dots, \boldsymbol{v_n}\}, \{\boldsymbol{w_1}, \dots, \boldsymbol{w_m}\}$ пространств V и W соответственно такие, что $M\boldsymbol{v_i} = \sigma_i \boldsymbol{w_i}$ и $M^*\boldsymbol{w_i} = \sigma_i \boldsymbol{v_i}$ для каждого $i = 1, \dots, p$ и что $M\boldsymbol{v_i} = \boldsymbol{0}_W$ и $M^*\boldsymbol{w_i} = \boldsymbol{0}_V$ для всех $i > p$.

Из приведённой выше теоремы следует, что матрицу A можно разложить следующим образом:

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

где $\Sigma = diag(\sigma_1, \dots, \sigma_p, 0, \dots, 0) \in \mathbb{R}^{n \times m}$, $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$; $U, V -$ матрицы, чьи столбцы представляющие собой векторы, образующие ортонормированные базисы в пространствах $\mathbb{R}^n$ и $\mathbb{R}^m$ соответственно.

Такое разложение называется сингулярным разложением матрицы A (singular value decomposition, SVD).

$\sigma_1, \dots, \sigma_p$ называют сингулярными числами.

Матрицы U и V дают полное представление о образе и ядре матрицы A: строки V^T, соответствующие нулевым сингулярным значениям являются базисом ядра матрицы A, а столбцы U, соответствующие ненулевым сингулярным значениям образуют базис образа матрицы A.

Продемонстрируем геометрический смысл сингулярного разложения. Легче всего это сделать, когда $n = m$.

Линейное отображение A представляет собой композицию операторов поворота и растяжения. Если мы будем рассматривать сферу единичного радиуса в $\mathbb{R}^n$, то A будет отображать её в эллипсоид, длины полуосей которого будут соответствовать ненулевым сингулярным значениям матрицы A.

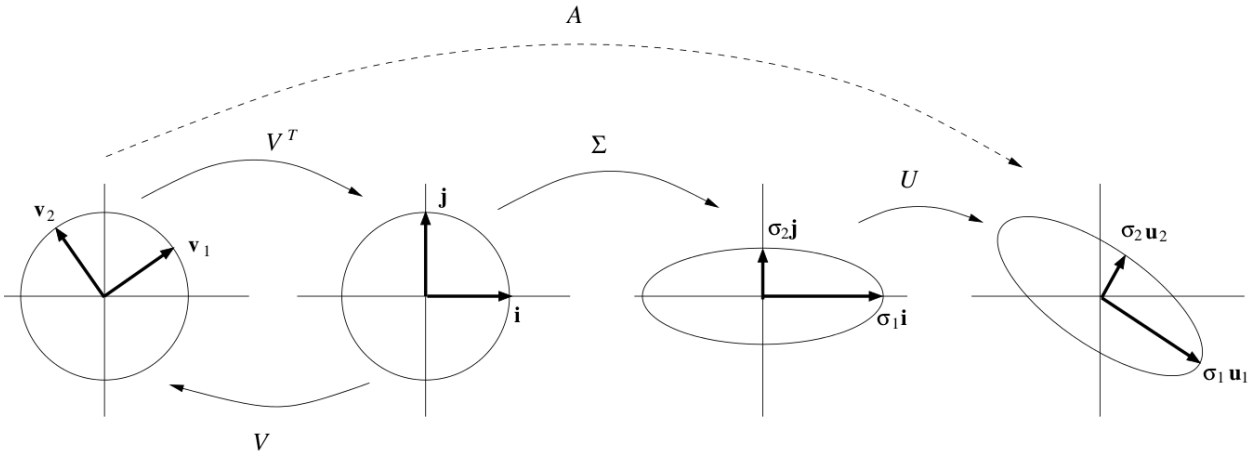

Рисунок 1. Геометрический смысл сингулярного разложения на примере оператора A: $\mathbb{R}^2 \to \mathbb{R}^2$

Сингулярное разложение матрицы — популярный приём, который используется при решении многих прикладных задач. Он широко применяется в цифровой обработке сигналов, обработке изображений, машинном обучении, прикладной статистике, при проектировании рекомендательных систем и даже в орбитальной механике.

Бидиагональная матрица – матрица с ненулевыми элементами вдоль главной диагонали и диагональю выше (ниже).

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & \dots \\ 0 & a_{22} & a_{23} & 0 & \dots \\ 0 & 0 & a_{33} & a_{34} & \dots \\ 0 & 0 & 0 & a_{44} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$ — верхняя бидиагональная матрица

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 & \dots \\ a_{21} & a_{22} & 0 & 0 & \dots \\ 0 & a_{32} & a_{33} & 0 & \dots \\ 0 & 0 & a_{43} & a_{44} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$ — нижняя бидиагональная матрица

Супердиагональ квадратной матрицы – это диагональ, состоящая из элементов, которые лежат непосредственно над элементами, составляющую главную диагональ. Индексы супердиагональных элементов: i, $j = i + 1$.

Блочная матрица – представление матрицы, при котором она рассекается вертикальными и горизонтальными линиями на прямоугольные части – блоки:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1t} \\ A_{21} & A_{22} & \cdots & A_{2t} \\ \cdots & \cdots & \cdots & \cdots \\ A_{s1} & A_{s1} & \cdots & A_{st} \end{bmatrix},$$

где блок $A_{\alpha\beta}$ имеет размер $m_\alpha \times n_\beta$ для $\alpha = 1, 2, \ldots, s$ и $\beta = 1, 2, \ldots, t$.

Далее, мы рассмотрим самые популярные алгоритмы для вычисления SVD, модификации которых используются в современных библиотеках для численных вычислений.

Вращения Гивенса

Вращения Гивенса — это метод в численной линейной алгебре, используемый для внесения нулей в матрицы или решения систем линейных уравнений.

Каждое вращение Гивенса определяется углом θ и действует на пару строк или столбцов матрицы, вращая их в плоскости, определенной этими двумя измерениями, таким образом, чтобы один из элементов пары стал равен нулю. Преобразование описывается матрицей вида:

$$\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos(\theta) & \cdots & -\sin(\theta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin(\theta) & \cdots & \cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

где элементы *cos(θ)* и *sin(θ)* расположены на пересечении *i*-й и *j*-й строк и столбцов, что позволяет вращать *i-й* и *j-й* столбцы (или строки) на угол θ.

Применение вращения Гивенса позволяет последовательно обнулить выбранные элементы матрицы без значительного увеличения вычислительной

сложности или потери численной стабильности. Это делает вращения Гивенса ценным инструментом в различных алгоритмах численного анализа, включая алгоритмы для вычисления сингулярных значений и собственных значений матриц.

Глава 3. Алгоритмы итеративного уточнения сингулярного разложения матрицы

Описание

В данном разделе описаны быстрые численные алгоритмы для повышения точности вычисления сингулярных векторов для вещественной матрицы, которые предложены в [1].

$$A = U\Sigma V^T \quad (3.1)$$

— сингулярное разложение, где $A \in R^{mxn}$, $U \in R^{mxm}, \Sigma \in R^{mxn}, V \in R^{nxn}$.

Мы предполагаем, что m ≥ n

Приближение: $\hat{u}_i \approx u_i, \hat{\sigma}_i \approx \sigma_i, \hat{v}_i \approx v_i$.

Для $k \leq n$ положим:

$$U_{1:k} := (u_1, \dots, u_k), \quad \widehat{U'}_{1:k} := (\hat{u}'_1, \dots, \hat{u}'_k) \quad \in R^{mxk},$$

$$V_{1:k} := (v_1, \dots, v_k), \quad \widehat{V'}_{1:k} := (\hat{v}'_1, \dots, \hat{v}'_k) \quad \in R^{nxk}$$

$$\hat{u}'_i := \frac{\hat{u}_i}{||\hat{u}_i||}, \hat{v}'_i := \frac{\hat{v}_i}{||\hat{v}_i||}$$

$$\hat{\Sigma}_k := diag(\hat{\sigma}'_1, \dots, \hat{\sigma}'_k) \in R^{kxk}, \text{где } \hat{\sigma}'_k := (\hat{u}'_i)^T A \hat{v}'_i$$

$$R_{1:k} := A\widehat{V'}_{1:k} - \widehat{U'}_{1:k}\hat{\Sigma}'_k, \quad S_{1:k} := A^T\widehat{U'}_{1:k} - \widehat{V'}_{1:k}\hat{\Sigma}'_k,$$

$$\delta_k := \min\left|\sigma'_i - \sigma_j\right|, \text{где } 1 \leq i \leq k, k < j \leq \max(n, k+1) \quad (3.2)$$

Если $\delta_k > 0$, то из (2) следует, что

$$\sqrt{\left\|\sin\Theta\left(U_{1:k}, \widehat{U'}_{1:k}\right)\right\|_F^2 + \left\|\sin\Theta\left(V_{1:k}, \widehat{V'}_{1:k}\right)\right\|_F^2} \leq \frac{\sqrt{\|R_{1:k}\|_F^2 + \|S_{1:k}\|_F^2}}{\delta_k}, \quad (3.3)$$

где $\Theta\left(U_{1:k}, \widehat{U'}_{1:k}\right)$ и $\Theta\left(V_{1:k}, \widehat{V'}_{1:k}\right)$- матрицы канонических углов между $U_{1:k}$ и $\widehat{U'}_{1:k}$ и между $V_{1:k}$ и $\widehat{V'}_{1:k}$ соответственно, а $\|\cdot\|_F$ обозначает норму Фробениуса. Из (3.3) следует, что по мере уменьшения значения δ_k в (3.2) точность вычисленных сингулярных векторов также снижается. Следовательно, для получения достаточно точных результатов при разложении по сингулярным значениям можно использовать итерационные методы уточнения.

Обозначения

$(\cdot)_h$ и $(\cdot)_l$ - результаты числовой арифметики, где все операции внутри круглых скобок выполняются с большей и меньшей точностью соответственно. Мы используем следующие комбинации точности [более высокая точность и более низкая точность], [четырехкратная точность и двойная точность] и [двойная точность и одинарная точность]. Для простоты мы опустим члены, меньшие $O(m^k n^l), k + l = 3$ из подсчета операций.

Итеративный метод уточнения со смешанной точностью

Мы рассматриваем линейную систему $Ax = b$ для $x, b \in R^n$ и не сингулярную матрицу $A \in R^{n \times n}$. Пусть $\hat{x} \approx x$ – вычисленное решение $Ax = b$. Существует итеративный алгоритм уточнения с арифметикой смешанной точности для повышения точности $\hat{x}$.

Алгоритм 1. Итерационное уточнение приближенного решения $\hat{x}$ линейной системы $Ax = b$.

Начальное значение $\hat{x}$ получено с использованием арифметики меньшей точности.

1. Вычислите $r \leftarrow (b - A\hat{x})_h$
2. Преобразуйте значение r в значение с меньшей точностью.
3. Вычислите $y \leftarrow (A^{-1}r)_l$
4. Обновите $\hat{x}$ как $\hat{x} \leftarrow (\hat{x} + y)_{h\ or\ l}$

until точность $\hat{x}$ не станет достаточной

Итеративное уточнение для симметричной декомпозиции по собственным значениям

I_n – единичная матрица $n \times n$. Предполагаем, что $A = A^T \in R^{n \times n}$. Пусть $X \in R^{n \times n}$ ортогональна, $D \in R^{n \times n}$ диагональна и $A = XDX^T$. i-е столбцы X являются собственными векторами $x_{(i)} \in R^n$, а i-е диагональные элементы D являются собственными значениями $\lambda_i \in R$ для $i = 1, \ldots, n$. Здесь мы предполагаем это $\lambda_i \neq \lambda_j$ для $i \neq j$. Для $\hat{X} \approx X$ мы определяем матрицу

ошибок $E \in R^{n\times n}$ таким образом, что $X = \hat{X}(I_n + E)$. Алгоритм для вычисления $\tilde{E} \approx E$ и преобразования $\hat{X}$ в $\tilde{X}$ как $\tilde{X} \leftarrow \hat{X}(I_n + E)$. Алгоритм сходится квадратично при условии, что матрица ошибок E удовлетворяет следующему условию:

$$\|E\|_2 < \min\left(\frac{\min_{1\le i<j\le n}|\lambda_i - \lambda_j|}{10\sqrt{n}\|A\|_2}, \frac{1}{100}\right). \quad (3.4)$$

Алгоритм 2. Уточнения приближенных собственных векторов $\hat{X} \in R^{n\times n}$ для вещественной симметричной матрицы $A \in R^{n\times n}$. Предположим $\widetilde{\lambda_\imath} \neq \widetilde{\lambda_\jmath}$ для $i \neq j$. Общая стоимость составляет от $6n^3$ до $8n^3$ операций.

Требуется: $A \in R^{n\times n}, \hat{X} \in R^{n\times n}$

Убедитесь: $\tilde{X} \in R^{n\times n}, \widetilde{D} \in R^{n\times n}$

function $[\tilde{X}, \widetilde{D}] \leftarrow RefSyEv(A, \hat{X})$

$$R \leftarrow \left(I_n - \hat{X}^T\hat{X}\right)_h$$

$$S \leftarrow \left(\hat{X}^T A\hat{X}\right)_h$$

$$\widetilde{\lambda_\imath} \leftarrow \left(\frac{s_{ii}}{1 - r_{ii}}\right)_h \text{ для } (1 \le i \le n)$$

$$\widetilde{D} \leftarrow diag\left(\widetilde{\lambda_1}, \dots, \widetilde{\lambda_n}\right)$$

$$F \leftarrow \left(S + R\widetilde{D}\right)_h$$

$$\widetilde{e_{\imath\jmath}} \leftarrow \begin{cases} \left(\frac{f_{ii}}{\widetilde{\lambda_\jmath} - \widetilde{\lambda_\imath}}\right)_h & \text{для } (1 \le i, j \le n) \\ \left(\frac{r_{ii}}{2}\right)_h & \text{иначе} \end{cases}$$

$$\tilde{X} \leftarrow \left(\hat{X} + \hat{X}\tilde{E}\right)_h$$

end function

Здесь $S = (s_{ij})$, $R = (r_{ij})$, $F = (f_{ij})$, $\tilde{E} = (\tilde{e}_{ij})$.

Алгоритм для уменьшения требуемого параметра-математической точности алгоритма 2. Он основан на итеративном уточнении для решения

линейных систем с использованием арифметики смешанной точности. Из $R = I_n - \hat{X}^T\hat{X}$ и $S = \hat{X}^T A\hat{X}$ в алгоритме 2, F в строке 6 удовлетворяет

$$F = S + R\widetilde{D} = \hat{X}^T A\hat{X} + (I_n - \hat{X}^T\hat{X})\widetilde{D} = \hat{X}^T(A\hat{X} - \hat{X}\widetilde{D}) + \widetilde{D}$$

Поскольку на диагональную часть F ссылки нет, ее можно вычислить следующим образом

$$F := \hat{X}^T W \quad (3.5)$$

где $W \coloneqq A\hat{X} - \hat{X}\widetilde{D}$. Условие сходимости такое же, как для алгоритма 2.

Алгоритм для уменьшения требуемого параметра-математической точности алгоритма 2. Он основан на итеративном уточнении для решения линейных систем с использованием арифметики смешанной точности. Из $R = I_n - \hat{X}^T\hat{X}$ и $S = \hat{X}^T A\hat{X}$ в алгоритме 2, F в строке 6 удовлетворяет

$$F = S + R\widetilde{D} = \hat{X}^T A\hat{X} + (I_n - \hat{X}^T\hat{X})\widetilde{D} = \hat{X}^T(A\hat{X} - \hat{X}\widetilde{D}) + \widetilde{D}$$

Поскольку на диагональную часть F ссылки нет, ее можно вычислить следующим образом

$$F \coloneqq \hat{X}^T W \quad (3.6)$$

где $W \coloneqq A\hat{X} - \hat{X}\widetilde{D}$. Условие сходимости такое же, как для алгоритма 2.

Алгоритм 3. Уточнение приближенных собственных векторов $\hat{X} \in R^{n\times n}$ для вещественной симметричной матрицы $A \in R^{n\times n}$. Предположим $\widetilde{\lambda_\iota} \neq \widetilde{\lambda_j}$ для $i \neq j$. Общая стоимость составляет $6n^3$ операций.

Требуется: $A \in R^{n\times n}, \hat{X} = \left(x_{(1)}, \dots, x_{(n)}\right) \in R^{n\times n}$

Убедитесь: $\tilde{X} \in R^{n\times n}, \widetilde{D} \in R^{n\times n}$

function $[\tilde{X}, \widetilde{D}] \leftarrow RefSyEv2(A, \hat{X})$

$P \leftarrow \left(A\hat{X}\right)_h$

$r_{ii} \leftarrow (1 - \widehat{x_{(\iota)}}^T \widehat{x_{(\iota)}})$ для $(1 \le i \le n)$

$s_{ii} \leftarrow \left(\widehat{x_{(\iota)}}^T p_{(i)}\right)_h$ для $(1 \le i \le n)$

$\widetilde{\lambda_\iota} \leftarrow \left(\frac{s_{ii}}{1 - r_{ii}}\right)_h$ для $(1 \le i \le n)$

$$\widetilde{D} \leftarrow diag\left(\widetilde{\lambda_1}, \dots, \widetilde{\lambda_n}\right)$$

$$W \leftarrow \left(P - \hat{X}\widetilde{D}\right)_h$$

$$F \leftarrow \left(\hat{X}^T W\right)_l$$

$$\widetilde{e_{ij}} \leftarrow \begin{cases} \left(\dfrac{f_{ii}}{\widetilde{\lambda_j} - \widetilde{\lambda_i}}\right)_h & i \neq j \\ \left(\dfrac{r_{ii}}{2}\right)_h & \text{иначе} \end{cases} \quad \text{для } (1 \leq i, j \leq n)$$

$$\tilde{X} \leftarrow \left(\hat{X} + \left(\hat{X}\tilde{E}\right)_l\right)_h$$

end function

Итеративное уточнение для разложения по сингулярным значениям

Здесь мы вводим итеративное уточнение для полного разложения по сингулярным значениям. Далее мы предполагаем, что

$$\sigma_1 > \sigma_2 > \cdots > \sigma_n$$

и приближение $\widetilde{\sigma_i}$ для σ_i удовлетворяет $\widetilde{\sigma_i} \neq \widetilde{\sigma_j}$ для $i \neq j$. Пусть $\widehat{U} \approx U$ и $\widehat{V} \approx V$ для U и V в (1). Определите матрицы ошибок $F \in R^{m\times m}$ и $G \in R^{n\times n}$ такими

$$U = \widehat{U}(I_m + F) \text{ и } V = \widehat{V}(I_n + G)$$

соответственно. Алгоритм, основанный на той же идее, что и алгоритм 2. Он вычисляет $\tilde{F} \approx F$ и $\tilde{G} \approx G$ и обновляет $\widehat{U}$ и $\widehat{V}$ до U и V как $\widetilde{U} \leftarrow \widehat{U}(I_n + F)$ и $\tilde{V} \leftarrow \hat{V}(I_n + G))$ соответственно. Этот процесс показан в алгоритме 4. Обратите внимание, что для $T, \widehat{U}, \widetilde{U}, R \in R^{m\times m}$ у нас есть

$$T = (T_1 T_2), \widehat{U} = \left(\widehat{U}_1 \widehat{U}_2\right), \widetilde{U} = \left(\widetilde{U}_1\ \widetilde{U}_2\right), R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$$

с $T_1, \widehat{U}_1, \widetilde{U}_1 \in R^{m\times n}$, $T_2, \widehat{U}_2, \widetilde{U}_2 \in R^{m\times(m-n)}$, $R_{11} \in R^{n\times n}$, $R_{12}, {R_{21}}^T \in R^{n\times(m-n)}$, $R_{22} \in R^{(m-n)\times(m-n)}$.

Алгоритм 4. Уточнения приближенных сингулярных векторов $\widehat{U} \in R^{m\times m}$ и $\widehat{V} \in R^{n\times n}$ для вещественной матрицы $A \in R^{m\times n}$. Предположим $\widetilde{\sigma_i} \neq \widetilde{\sigma_j}$ для $i \neq j$. Общая стоимость составляет от $3m^3 + 2m^2n + 2mn^2 + 3n^3$ до $4m^3 + 2m^2n + 2mn^2 + 4n^3$ операций.

Требуется: $A \in R^{m\times n}, \widehat{U} \in R^{m\times m}, \widehat{V} \in R^{n\times n}$

Убедитесь: $\widetilde{\mathrm{U}} \in R^{m\times m}, \tilde{\Sigma} \in R^{m\times n}, \tilde{V} \in R^{n\times n}$

function$[\widetilde{\mathrm{U}}, \tilde{\Sigma}, \tilde{V}] \leftarrow RefSVD(A, \widehat{U}, \widehat{V})$

$$R \leftarrow \left(I_m - \widehat{U}^T \widehat{U}\right)_h$$

$$S \leftarrow \left(I_n - \widehat{V}^T \widehat{V}\right)_h$$

$$T \leftarrow \left(\widehat{U}^T A \widehat{V}\right)_h$$

$$\widetilde{\sigma_i} \leftarrow \left(\frac{t_{ii}}{1 - \frac{r_{ii} + s_{ii}}{2}}\right)_h \text{ для } (1 \leq i \leq n)$$

$$\tilde{\Sigma}_n \leftarrow diag(\widetilde{\sigma_1}, \ldots, \widetilde{\sigma_n});\ \tilde{\Sigma} \leftarrow \left(\tilde{\Sigma}_n, O_{n,m-n}\right)^T$$

$$C_\alpha \leftarrow \left(T_1 + R_{11}\,\tilde{\Sigma}_n\right)_h;\ C_\beta \leftarrow \left({T_1}^T + S\,\tilde{\Sigma}_n\right)_h$$

$$D \leftarrow \left(\tilde{\Sigma}_n C_\alpha + C_\beta \tilde{\Sigma}_n\right)_h$$

$$E \leftarrow \left(C_\alpha \tilde{\Sigma}_n + \tilde{\Sigma}_n C_\beta\right)_h$$

$$\widetilde{g_{ij}} \leftarrow \begin{cases} \left(\dfrac{d_{ij}}{\tilde{\sigma_j}^2 - \tilde{\sigma_i}^2}\right)_h & i \neq j \\ \left(\dfrac{s_{ii}}{2}\right)_h & \text{иначе} \end{cases} \quad \text{для } (1 \leq i, j \leq n)$$

$$\widetilde{f_{ij}} \leftarrow \begin{cases} \left(\dfrac{e_{ij}}{\tilde{\sigma_j}^2 - \tilde{\sigma_i}^2}\right)_h & (i \neq j, i, j \leq n) \\ \left(-\dfrac{t_{ji}}{\tilde{\sigma_i}}\right)_h & (i \leq n < j) \\ \left(r_{ij} - \widetilde{f_{ji}}\right)_h & (j \leq n < i) \\ \left(\dfrac{r_{ij}}{2}\right)_h & \text{иначе} \end{cases} \quad \text{для } (1 \leq i, j \leq n)$$

$$\tilde{U} \leftarrow \left(\hat{U} + \hat{U}\tilde{F}\right)_h; \ \tilde{V} \leftarrow \left(\hat{V} + \hat{V}\tilde{G}\right)_h$$

end function

Алгоритм быстрого итеративного уточнения разложения матрицы по сингулярным значениям

Напомним, что разложением матрицы $A \in \mathbb{R}^{m\times n}$ по сингулярным значениям (SVD) называется такое соотношение

$$A = U\Sigma V^T, \tag{3.7}$$

где $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ – ортогональные матрицы, составленные из левых сингулярных векторов $u_{(i)} \in \mathbb{R}^m, i = 1, \dots, m$ и правых сингулярных векторов $v_{(i)} \in \mathbb{R}^n, i = 1, \dots, n$ соответственно, а $\Sigma \in \mathbb{R}^{m\times n}$ – прямоугольная диагональная матрица, составленная из сингулярных значений $\sigma_i \in \mathbb{R}$, $i = 1, \dots, n$ (полагается, что $m \geq n$).

Авторы Yuki Uchino, Takeshi Terao и Katsuhisa Ozaki предлагают быстрый (по сравнению с иными описанными в [1]) алгоритм для полного разложения матрицы по сингулярным значениям. Здесь и далее полагается, что сингулярные значения удовлетворяют следующему соотношению:

$$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_n,$$

а их приближённые значения $\tilde{\sigma}_i$ таковы, что $\tilde{\sigma}_i \neq \tilde{\sigma}_j$ для $i \neq j$.

Далее для приближённых значений будет использоваться диакритический знак ^, а для уточнённых (то есть вычисленных с более высокой точностью в результате работы алгоритма) – знак $\sim$.

На вход алгоритма поступает матрица $A \in \mathbb{R}^{m\times n}$ и матрицы приближённых сингулярных векторов $\hat{U} \in \mathbb{R}^{m\times m}, \hat{V} \in \mathbb{R}^{n\times n}$. На выходе ожидается получение матриц уточнённых сингулярных векторов $\tilde{U} \in \mathbb{R}^{m\times m}, \tilde{V} \in \mathbb{R}^{n\times n}$ и матрицы уточнённых сингулярных значений $\tilde{\Sigma} \in \mathbb{R}^{m\times n}$. Опишем алгоритм пошагово:

1. $P \leftarrow \left(A\hat{V}\right)_h$ //символ $(\cdot)_h$ означает вычисление с двойной точностью;
2. $Q \leftarrow \left(A^T\hat{U}_1\right)_h$ //$\hat{U}_1 \in \mathbb{R}^{m\times n}$ – левый блок блочной матрицы $\left(\hat{U}_1\hat{U}_2\right) = \hat{U}$;
3. $r_{ii} \leftarrow \left(1 - \left(\hat{u}_{(i)}\right)^T\hat{u}_{(i)}\right)_h$, где $1 \leq i \leq n$;
4. $s_{ii} \leftarrow \left(1 - \left(\hat{v}_{(i)}\right)^T\hat{v}_{(i)}\right)_h$, где $1 \leq i \leq n$;

5. $t_{ii} \leftarrow \left(\left(\hat{u}_{(i)}\right)^T \hat{p}_{(i)}\right)_h$, где $1 \le i \le n$ //$p_{(i)} \in P = \left(p_{(1)}, p_{(2)}, \dots, p_{(n)}\right)$;

6. $\tilde{\sigma}_i \leftarrow \left(\frac{t_{ii}}{1-(r_{ii}+s_{ii})/2}\right)_h$, где $1 \le i \le n$;

7. $\tilde{\Sigma}_n \leftarrow diag(\tilde{\sigma}_1, \tilde{\sigma}_2, \dots, \tilde{\sigma}_n)$;

8. $\tilde{\Sigma} \leftarrow \left(\tilde{\Sigma}_n, O_{n,m-n}\right)^T$ //$O_{n,m-n} \in \mathbb{R}^{n\times(m-n)}$ – матрица нулей;

9. $C_\gamma \leftarrow \left(P - \hat{U}_1\tilde{\Sigma}_n\right)_h$;

10. $C_\delta \leftarrow \left(Q - \hat{V}\tilde{\Sigma}_n\right)_h$;

11. $C_\alpha \leftarrow \left(\hat{U}_1{}^T C_\gamma\right)_l$ //символ $(\cdot)_l$ означает вычисление с одинарной точностью;

12. $C_\beta \leftarrow \left(\hat{V}^T C_\delta\right)_l$;

13. $D \leftarrow \left(\tilde{\Sigma}_n C_\alpha + C_\beta \tilde{\Sigma}_n\right)_h$ //матрица $D = \left(d_{ij}\right)$;

14. $E \leftarrow \left(C_\alpha \tilde{\Sigma}_n + \tilde{\Sigma}_n C_\beta\right)_h$ //матрица $E = \left(e_{ij}\right)$;

15. $\tilde{g}_{ij} \leftarrow \begin{cases} \left(\frac{d_{ij}}{\tilde{\sigma}_j{}^2 - \tilde{\sigma}_i{}^2}\right)_h, (i \ne j) \\ \left(\frac{s_{ii}}{2}\right)_h, (\text{иначе}) \end{cases}$, где $1 \le i, j \le n$;

16. $\tilde{f}_{ij} \leftarrow \begin{cases} \left(\frac{e_{ij}}{\tilde{\sigma}_j{}^2 - \tilde{\sigma}_i{}^2}\right)_h, (i \ne j) \\ \left(\frac{r_{ii}}{2}\right)_h, (\text{иначе}) \end{cases}$, где $1 \le i, j \le n$ //матрица $\tilde{F}_{11} = \left(\tilde{f}_{ij}\right)$;

17. $\tilde{F}_{12} \leftarrow \left(\tilde{\Sigma}_n{}^{-1} P^T \hat{U}_2\right)_h$;

18. $\tilde{F}_{21} \leftarrow \left(\left(\hat{U}_2{}^T C_\gamma\right)_l \tilde{\Sigma}_n{}^{-1}\right)_h$;

19. $\tilde{F}_{22} \leftarrow \left(\frac{1}{2}\left(I_{m-n} - \hat{U}_2{}^T \hat{U}_2\right)\right)_h$ //$I_{m-n} \in \mathbb{R}^{(m-n)\times(m-n)}$ – единичная матрица;

20. $\tilde{U} \leftarrow \left(\hat{U} + \left(\hat{U}\tilde{F}\right)_l\right)_h$ //$\tilde{F} = \begin{pmatrix} \tilde{F}_{11} & \tilde{F}_{12} \\ \tilde{F}_{21} & \tilde{F}_{22} \end{pmatrix}$ – блочная матрица, где $\tilde{F}_{11} \in \mathbb{R}^{n\times n}, \tilde{F}_{12} \in \mathbb{R}^{n\times(m-n)}, \tilde{F}_{21} \in \mathbb{R}^{(m-n)\times n}, \tilde{F}_{22} \in \mathbb{R}^{(m-n)\times(m-n)}$;

21. $\tilde{V} \leftarrow \left(\hat{V} + \left(\hat{V}\tilde{G}\right)_l\right)_h$ //матрица $\tilde{G} = (\tilde{g}_{ij})$.

Общая «стоимость» алгоритма составляет $3m^3 + 2m^2n + 3mn^2 + 4n^3$ операций в лучшем случае и $4m^3 + 4mn^2 + 4n^3$ – в худшем, где

- $2n^3$: $\hat{V}^T C_\delta$;
- $2mn^2$: $A\hat{V}, A^T\hat{U}_1, \hat{U}_1{}^T C_\gamma$;
- $2mn(m-n)$: $P^T\hat{U}_2, \hat{U}_2{}^T C_\gamma$;
- $m(m-n)^2$ или $2m(m-n)^2$: $\hat{U}_2{}^T\hat{U}_2$;
- $2m^3$: $\hat{U}\tilde{F}$;
- $2n^3$: $\hat{V}\tilde{G}$.

Глава 4. Метод Якоби

Дадим описание методу, предложенному в [2] и [5]. Изначально нам дана матрица $A \in Mat(n,m), n \geq m$. В обратном случае, мы можем транспонировать матрицу и найти SVD для нее, потому что таким образом мы найдём SVD и для изначальной матрицы. Вот доказательство того, что это можно сделать

$$\boldsymbol{A^T = U'\Sigma' V'^T}$$

$$A = (A^T)^T = (U'\Sigma' V'^T)^T = V'\Sigma'^T U'^T = U\Sigma V^T$$

Существует два варианта метода Якоби – односторонний и двусторонний. Двусторонний подходит только для симметричных квадратных матриц. Цель нашего проекта – алгоритмы получения сингулярного разложения **для матриц общего вида**, поэтому мы его рассматривать не будем, хотя он очень схож с односторонним.

Суть одностороннего метода Якоби состоит в том, чтобы с помощью последовательности поворотов сделать так, чтобы столбцы матрицы стали ортогональными. Некоторые столбцы матрицы могут стать нулевыми, но в этом нет ничего страшного

Поворотом мы называем матрицу поворота Якоби $R \in Mat(m,m)$. Индексом поворота будем называть пару (i,j). Поворот с индексом (i,j) приводит матрицу A к матрице $A' = AR^{(i,j)}$, у которой $a'_{ij} = a'_{ji} = 0$. Основная мысль – матрица поворота позволяет занулять элементы исходной матрицы, и мы пользуемся этим свойством.

Очевидно, что зануляя случайные элементы матрицы, мы не приведём её к нужному виду. Нужна стратегия выбора индекса следующего поворота. Изначально было доказано, что алгоритм остается корректным, если использовать циклическую стратегию выбора поворота: поочередно применяются повороты с индексами $(1,2),(1,3),\dots,(1,m),(2,3),(2,4),\dots,(2,m),\dots,(m-1,m)$. Однако такой

алгоритм требует очень много времени, поэтому мы будем использовать стратегию «с выбором цели» (Jacobi Target Selection). Такая стратегия среди всех известных обеспечивает наибыстрейшую сходимость алгоритма.

Допустим, мы привели матрицу A к матрице $B = AV_1 \dots V_t$, где V_i – матрица i-ого поворота, у которой все столбцы ортогональны. Найдём матрицы U и S Найдем норму $\|b_i\|_2$ каждого ненулевого столбца b_i. Поменяем столбцы матрицы B так, чтобы их нормы шли в порядке невозрастания, т.е. так, чтобы $\|b_i\|_2 \leq \|b_j\|_2$ при $i > j$. Если мы меняем b_i и b_j местами, то также нужно поменять местами строки v_i, v_j матрицы V. Матрица S формируется следующим образом:

$$\Sigma_{ij} = 0 \text{ при } i \neq j, ;$$

$$\Sigma_{ii} = \{\|b_i\|_2, b_i - \text{ненулевой столбец } 0, \text{иначе}$$

Нормируем столбцы матрицы B, допишем к ним $n - m$ нулей, чтобы матрица B стала $n \times n$. Теперь заменим нулевые столбцы матрицы B столбцами, ортогональными ненулевым столбцам – первые m компонент этих столбцов должны равняться 0, а одна из оставшихся компонент – 1, остальные также должны быть равными 0. Так мы получаем матрицу U.

Получаем сингулярное разложение U, Σ, V.

Матрицу будем считать ортогональной, когда

$b_i^T b_j < \varepsilon \|A\|_F = \varepsilon \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^m}$, где ε – параметр сходимости, произвольное маленькое число, которое вводит пользователь.

Глава 5. Алгоритмы для сингулярного разложения бидиагональных матриц

Алгоритм «Сингулярные значения бидиагональной матрицы методом бисекции»

Помимо метода Якоби, в [2] предлагаются алгоритмы для сингулярного разложения матриц бидиагонального вида. На вход алгоритма поступают переменные n, **B**, α, β и *tol*. Приведем описание данных переменных:

B – бидиагональная матрица (ненулевые элементы расположены только на главной и одной из соседних диагоналей)

n – размерность матрицы **B**

α и β – левая и правая граница полуинтервала [α, β), в котором будем искать сингулярные значения B.

tol – числовая точность алгоритма

n_{low}, n_{up}, n_{mid} – количество сингулярных значений находящихся левее нижней (верхней) границы или середины интервала.

Опишем основной алгоритм по шагам:

1. Вычислим n_α использовав вспомогательную функцию *Negcount*(n, **B**, α)
2. Аналогичным образом вычислим n_β = *Negcount*(n, **B**, β)
3. Если $n_\alpha = n_\beta$, на полуинтервале [α, β) нет сингулярных значений
4. В противном случае список [α, n_α, β, n_β] становится частью Worklist
5. Пока Worklist не пуст будем выполнять следующие шаги:
    a. Предыдущие значения Worklist удаляются. На первой итерации [low, n_{low}, up, n_{up}] = [α, n_α, β, n_β], на следующих итерациях берем вычисленные значения из предыдущих итераций
    b. Переменная mid = (low + up) / 2
    c. Проверяем неравенство up – low >= *tol*

        Если неравенство выполнено проделываем следующие шаги:

- n_{mid} = *Negcount*(n, B, mid)
- Если $n_{mid} > n_{low}$ в работу Worklist идет список [low, n_{low}, mid, n_{mid}]
- Если $n_{up} > n_{mid}$, в работу Worklist идет список [mid, n_{mid}, up, n_{up}]

  Если же неравенство не выполнено, выполняем цикл
- От i в диапозоне [$n_{low} + 1$, n_{up}], элемент массива *w*, стоящий на позиции i – n_{α} равен mid (одно из искомых сингулярных значений).

Результат выполнения алгоритма – массив *w*, состоящий из сингулрных значений **B** в заданном диапозоне.

Алгоритм *Negcount* в псевдокоде выше вычисляет число сингулянрных значений меньше чем μ

На вход алгоритма поступают переменные n, **B** и μ

Опишем алгоритм *Negcount* по шагам:

1. t = -μ
2. При k от 1 до n-1

   $d = \beta_{k,k}^2 + t$

   Если d < 0, число *Negcount* = *Negcount* + 1

   $t = t * (\beta_{k,k+1}^2 / d) - \mu$
3. $d = \beta_{n,n}^2 + t$
4. Если d < 0, число *Negcount* = *Negcount* + 1

Эффективность алгоритма зависит от размера матрицы и заданной числовой точности tol. Итеративный процесс разбиения интервала может быть неэффективным для больших матриц или малых значений tol. Однако, алгоритм позволяет находить сингулярные значения с высокой точностью в заданном интервале.

Итак, описанный алгоритм представляет собой метод численного поиска сингулярных значений бидиагональной матрицы в заданном интервале с заданной точностью. Он может быть эффективным для небольших и средних размеров матриц при умеренной числовой точности.

Алгоритм «Разложение по сингулярным значениям бидиагональной матрицы с высокой относительной точностью»

На вход алгоритма поступает верхняя бидиагональная матрица B размером $n \times n$.

На выходе ожидаются диагональная матрица Σ размером $n \times n$, ортогональные матрицы U и V размером $n \times n$ такие, что $B = U\Sigma V^T$.

Опишем пошагово алгоритм:

1. Вычислим $\underline{\sigma}$, где $\underline{\sigma} = \underline{\sigma}(B) := \min_{||x||=1} ||Bx||$.
2. Вычислим $\bar{\sigma} = max_i(b_{i,i}, b_{i.i+1})$.
3. Повторяем следующие действия:
   a. Для всех $i = 1, \ldots, n-1$ положим $b_{i.i+1} = 0$.
   b. Определим наименьшее значение p и наибольшее значение q так, чтобы матрица B стала блочной.

$$B = \begin{pmatrix} B_{1,1} & 0 & 0 \\ 0 & B_{2,2} & 0 \\ 0 & 0 & B_{3,3} \end{pmatrix},$$

   где $B_{1,1}$ – матрица размера $p \times p$; $B_{2,2}$ – матрица размера $(n - p - q) \times (n - p - q)$, у которой элементы, лежащие на супердиагонали, ненулевые; $B_{3,3}$ – диагональная матрица размера $q \times q$.

   c. Если $q = n$, то диагональ матрицы Σ – это диагональ матрицы B. Останавливаемся.
   d. Если $i = p + 1, \ldots, n - q - 1$, $b_{i,i} = 0$, то применим метод Гивенса так, чтобы $b_{i,i+1} = 0$ и матрица $B_{2,2}$ оставалась все еще верхней бидиагональной. В противном случае необходимо применить следующий алгоритм.

Алгоритм «Шаг Деммеля–Кахана»

Опишем алгоритм, предложенный в [3]. На вход поступают числа n, p и q, матрицы B, Q, P, где B – верхняя бидиагональная матрица размера $n \times n$; Q и P состоят из ортогональных векторов таких, что $A = QBP^T$; значения $\overline{\sigma}$ и $\underline{\sigma}$.

На выходе ожидаются матрицы B, Q, P такие, что $A = QBP^T$; Q и P состоят из ортогональных векторов; матрица B имеет меньшие недиагональные элементы, чем на входе. В памяти матрицы B, Q, P перезаписываются.

Опишем пошагово алгоритм:

1. Пусть $B_{2,2}$ – блок матрицы B, состоящий только из элементов главной диагонали матрицы B, с индексами строк и столбцов вида $p+1, \dots, n-q$.
2. Если $tol^* \underline{\sigma} \leq \varepsilon_0 \overline{\sigma}$, тогда:
   a. Определяем значение c', как $c' = c = 1$;
   b. Для $k = p+1,\ n-q-1$
      - Введем значения $\alpha = cb_{k,k},\ \beta = b_{k,k+1}$;
      - Определим c и s:
      $$[\alpha \quad \beta]\begin{bmatrix} c & s \\ -s & c \end{bmatrix} = [r \quad 0], \text{где } r = \sqrt{\alpha^2+\beta^2}.$$
      - Если $k \neq p+1$, то $b_{k-1,k} = s'r$.
      - Переопределим матрицу P
      $P \leftarrow PR_{k,k+1}(c,s)$, где $R_{k,k+1}(c,s)$ – матрица вращений.
      - Переопределим значения α и β так, что $\alpha = c'r, \beta = sb_{k+1,k+1}$.
      - Определим c' и s':
      $$\begin{bmatrix} c' & -s' \\ s' & c' \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha^2+\beta^2} \\ 0 \end{bmatrix}.$$
      - Переопределим матрицу Q
      $Q \leftarrow QR_{k,k+1}(c,-s)$, где $R_{k,k+1}(c,-s)$ – матрица вращений.
      - $b_{k,k} = \sqrt{\alpha^2+\beta^2}$.

c. $b_{n-q-1,n-q} = \left(b_{n-q,n-q}c\right)s'$;

$b_{n-q,n-q} = (b_{n-q,n-q}c)c'$.

В противном случае применяем алгоритм «Шаг Голуб–Кахана» к n, p, q, B, Q, P.

Алгоритм «Разложение по сингулярным значениям»

Информация на входе:

- n – количество столбцов у рассматриваемой матрицы A
- B – нижняя бидиагональная матрица размера (n + 1) на n.

Информация на выходе:

- $\sum$ - диагональная матрица размера n на n
- U – ортогональная матрица размера (n+1) на (n+1)
- V – ортогональная матрица размера n на n, такая, что $B = U\sum V^T$

Шаги алгоритма:

Если $n < n_0$, тогда вызываем алгоритм Голуб – Рейнш SVD с входными данными $n, n+1, B$, чтобы получить на выходе $\sum, U, V$. Иначе пусть $B = \begin{pmatrix} B_1 & \alpha_k \boldsymbol{e}_k & 0 \\ 0 & \beta_k \boldsymbol{e}_1 & B_2 \end{pmatrix}$, где $k = n/2$.

a. Вызываем DC_SVD($k-1, B_1, \sum_1, U_1, W_1$).
b. Вызываем DC_SVD($n-k, B_2, \sum_2, U_2, W_2$).
c. Разбиваем $U_i = (Q_i, \boldsymbol{q}_i)$, для i = 1,2, где $\boldsymbol{q}_i$ – вектор-столбец.
d. Извлекаем $l_1 = Q_1^T \boldsymbol{e}_k, \lambda_1 = \boldsymbol{q}_1^T \boldsymbol{e}_k, l_2 = Q_2^T \boldsymbol{e}_1, \lambda_2 = \boldsymbol{q}_2^T \boldsymbol{e}_1$.
e. Разбиваем B как

$$B = \begin{pmatrix} c_0\boldsymbol{q}_1 & Q_1 & 0 & -s_0\boldsymbol{q}_1 \\ s_0\boldsymbol{q}_2 & 0 & Q_2 & c_0\boldsymbol{q}_2 \end{pmatrix} \begin{pmatrix} r_0 & 0 & 0 \\ \alpha_k l_1 & \sum_1 & 0 \\ \beta_k l_2 & 0 & \sum_2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & W_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & W_2 \end{pmatrix}^T =$$

$$= (Q \quad q) \begin{pmatrix} M \\ 0 \end{pmatrix} W^T,$$

где $r_0 = \sqrt{(\alpha_k \lambda_1)^2 + (\beta_k \lambda_2)^2}, c_0 = \frac{\alpha_k \lambda_1}{r_0}, s_0 = \frac{\beta_k \lambda_2}{r_0}$.

f. Вычисляем сингулярные значения M, решая характеристическое уравнение $f(w) = 1 + \sum_{k=1}^{n} \frac{z_k^2}{d_k^2 - w^2} = 0$, обозначаем вычисленные сингулярные значения $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_n$.

g. Для $i = 1, \dots, n$ вычислить $\check{z}_i =$

$$\sqrt{(\hat{w}_n^2 - d_i^2)\prod_{k=1}^{i-1}\frac{(\hat{w}_k^2 - d_i^2)}{(d_k^2 - d_i^2)}\prod_{k=1}^{n-1}\frac{(\hat{w}_k^2 - d_i^2)}{(d_{k+1}^2 - d_i^2)}}.$$

h. Для $i = 1, \dots, n$ вычислить сингулярные векторы

$$\boldsymbol{u}_i = \left(\frac{\check{z}_1}{d_1^2 - \hat{w}_i^2}, \dots, \frac{\check{z}_n}{d_n^2 - \hat{w}_i^2}\right) \Big/ \sqrt{\sum_{k=1}^{n}\frac{\check{z}_k^2}{(d_k^2 - \hat{w}_i^2)^2}}$$

$$\boldsymbol{v}_i = \left(-1, \frac{d_2\check{z}_2}{d_2^2 - \hat{w}_i^2}, \dots, \frac{d_n\check{z}_n}{d_n^2 - \hat{w}_i^2}\right) \Big/ \sqrt{1 + \sum_{k=2}^{n}\frac{(d_k\check{z}_k)^2}{(d_k^2 - \hat{w}_i^2)^2}}$$

Получим $U = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_n], V = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n]$.

Возвращаем $\sum = \begin{pmatrix} diag(\hat{w}_1, \hat{w}_2, \dots, \hat{w}_n) \\ 0 \end{pmatrix}, U \leftarrow (QU \quad q), V \leftarrow WV.$

Алгоритм «Высокая относительная точность двунаправленных сингулярных значений»

Данный алгоритм основан на методе «Без квадратного корня» для вычисления сингулярных значений бидиагональной матрицы с высокой относительной точностью – это метод выбора, когда требуются только сингулярные значения.

На вход алгоритма поступает число n и верхняя двудиагональная матрица B размера $n \times n$.

На выходе ожидается сумма $\sum$ диагональных матриц размера $n \times n$, содержащих сингулярные значения матрицы B.

Пошаговое описание алгоритма:

1. Возведем в квадрат диагональные и недиагональные элементы B, чтобы сформировать массивы s и e соответственно, то есть для $i = 1, \dots, n-1$, $s_i = b_{i,i}^2$, $e_i = b_{i,i+1}^2$, последний элемент массива s равен $s_n = b_{n,n}^2$.
2. Повторим:
   a. Для всех $i = 1, \dots, n-1$, установим $e_i = 0$, если соблюден критерий относительной сходимости.
   b. Определим наименьшее значение p и наибольшее значение q, чтобы B можно было заблокировать как матрицу вида

$$B = \begin{pmatrix} B_{1,1} & 0 & 0 \\ 0 & B_{2,2} & 0 \\ 0 & 0 & B_{3,3} \end{pmatrix} \qquad \begin{matrix} p \\ n-p-q \\ q \end{matrix}$$

   где $B_{3,3}$ – диагональное значение, $B_{2,2}$ – не имеет нулевой супердиагональной записи.
   c. Если $q = n$, то $\sum = \sqrt{diag(s)}$. **Останавливаемся.**

Если $i = p + 1, \dots, n - q - 1, s_i = 0,$ Тогда применяем заданные вращения таким образом, чтобы $e_i = 0$ и $B_{2,2}$ было верхней двудиагональной матрицей. Иначе, применяем алгоритм, описанный ниже.

Алгоритм «Приведение матрицы Хаусхолдера к двудиагональной форме»

Рассмотрим алгоритм, описанный в [4]. В советской математической литературе метод приведения матрицы Хаусхолдера к двудиагональной форме чаще называется методом отражений. Сама же двудиагональная форма называется также «бидиагональной».

На вход алгоритма поступает матрица A, числа m и n, такие, что матрица A размера m×n.

На выходе ожидаются матрицы B, V и U такие, что матрица B – верхняя бидиагональная, U и V являются результатом матрицы Хаусхолдера, где $A=UBV^T$.

Опишем пошагово алгоритм:

1. B ← A (Пропустим этот шаг, если A должно быть перезаписано на B)
2. $U = I_{m\times n}$. (Создадим матрицу U размера m×n)
3. $V = I_{n\times n}$. (Создадим матрицу V размера n×n)
4. Определим матрицу Хаусхолдера Q_k (для k = 1 ,…, n) со следующим свойством: умножение слева столбца на матрицу Q_k оставляет компоненты 1 ,…, k – 1 неизменными, причём

$$Q_k \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_{k-1,k} \\ b_{k,k} \\ b_{k+1,k} \\ \vdots \\ b_{m,k} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_{k-1,k} \\ s \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \text{ где } s = \pm\sqrt{\sum_{i=k}^{m} b_{i,k}^2}.$$

6. Переопределим (для k = 1 ,…, n) матрицу B

B ← Q_k B.

7. Переопределим (для k = 1 ,…, n) матрицу U

U ← U Q_k.

Если $k \leq n - 2$, то определим матрицу Хаусхолдера (для $k = 1, \ldots, n$) P_{k+1} со следующим свойством: умножение справа строки на матрицу P_{k+1} оставляет компоненты $1, \ldots, k$ неизменными, причём

$[0 \quad \cdots \quad 0 \quad b_{k,k} \quad b_{k,k+1} \quad b_{k,k+2} \quad \cdots \quad b_{k,n}] \quad P_{k+1} =$

$[0 \quad \cdots \quad 0 \quad b_{k,k} \quad s \quad 0 \quad \cdots \quad 0]$, где $s = \pm\sqrt{\sum_{j=k+1}^{n} b_{k,j}^2}$.

8. Переопределим (для $k = 1, \ldots, n$) матрицу B

$B \leftarrow B P_{k+1}$.

9. Переопределим (для $k = 1, \ldots, n$) матрицу V

$V \leftarrow P_{k+1} V$.

Алгоритм «шаг алгоритма Голуба-Кахана»

На вход алгоритма поступают, числа n, p, q, матрицы B, Q, P такие, что матрица B размера n×n является верхней бидиагональной, Q и P имеют ортогональные столбцы, а матрица $A = QBP^T$.

На выходе ожидаются матрицы B, Q и P такие, что матрица B – верхняя бидиагональная, Q и P имеют ортогональные столбцы, а недиагональные элементы выходной матрицы B меньше, чем недиагональные элементы входной матрицы. (Матрицы B, Q и P перезаписываются в хранилище)

Опишем пошагово алгоритм:

1. Введём матрицу $B_{2,2}$, которая является диагональным блоком матрицы B с номерами строки и столбца p + 1, … , n – q.
2. Найдём матрицу $B_{2,2}^T$.
3. Введём C такой, что C – нижняя правая подматрица размером 2×2 матрицы $B_{2,2}^T$ $B_{2,2}$.
4. Найдём собственные значения λ_1, λ_2 подматрицы C.
5. Из чисел λ_1, λ_2 найдём то, что ближе к элементу $c_{2,2}$ матрицы C.
6. Введём μ = найденному числу п.5.
7. Введём k = p + 1.
8. Введём $\alpha = b_{k,k}^2 - \mu$.
9. Введём $\beta = b_{k,k}\, b_{k,k+1}$.

Все последующие шаги выполнять при k = p + 1, … , n – q – 1.

10. Введём c = cos(θ) и s = sin(θ) такие, что $[\alpha \quad \beta] \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha^2 + \beta^2} & 0 \end{bmatrix}$.
11. Введём матрицу $R_{k,k+1}(c,s)$ – матрица вращение Гивенса, действующая на столбцы k и k + 1 во время умножения справа.
12. Переопределим $B \leftarrow B\, R_{k,k+1}(c,s)$.
13. Переопределим $P \leftarrow P\, R_{k,k+1}(c,s)$.
14. Приравняем $\alpha = b_{k,k}$, $\beta = b_{k+1,k}$.

15. Приравняем c = cos(θ) и s = sin(θ) так, что $\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha^2 + \beta^2} \\ 0 \end{bmatrix}$.
16. Введём матрицу $R_{k,k+1}(c,–s)$ – матрица вращение Гивенса, действующая на столбцы k и k + 1 во время умножения слева.
17. Переопределим B ← $R_{k,k+1}(c,–s)$ B.
18. Переопределим Q ← Q $R_{k,k+1}(c,s)$.
19. Если $k \leq n – q – 1$, то приравняем

$\alpha = b_{k,k+1}$, $\beta = b_{k,k+2}$.

Заключение

В ходе группового проекта был проведен анализ методов и алгоритмов разложения матрицы по сингулярным значения. Цель проекта была достигнута. Выбранные алгоритмы оказались наиболее эффективными, а потому являются наилучшими из исследуемых.

Список литературы

1. Uchino Yuki, Terao Takeshi, Ozaki Katsuhisa. 2022.08.05 – Acceleration of Iterative Refinement for Singular Value Decomposition. 10.21203/rs.3.rs-1931986/v1.
2. A. R. Heesterman – Handbook of Linear Algebra. Chapman & Hall/CRC Taylor & Francis Group 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742
3. Cline, Alan & Dhillon, Inderjit. (2013). Computation of the Singular Value Decomposition. 10.1201/b16113-68.
4. Businger, P., Golub, G.: Linear least squares solutions by Householder transformations. Numer. Math.7, 269–276 (1965).
5. M. Becˇka, G. Okša, M. Vajteršic, Dynamic ordering for a parallel block-Jacobi SVD algorithm, Parallel Comput. 28 (2002) 243–262.