Введение

Алгоритм `svdlapack` представляет собой обёртку над подпрограммами библиотеки LAPACK для вычисления сингулярного разложения матрицы (SVD).

Библиотека LAPACK предоставляет высокоэффективные реализации алгоритмов линейной алгебры, оптимизированные для современных вычислительных архитектур. В проекте SLEPc (Scalable Library for Eigenvalue Problem Computations) реализован интерфейс к этим алгоритмам, позволяющий эффективно вычислять сингулярные разложения как плотных, так и разреженных матриц.

Алгоритм вычисления SVD

Алгоритм `svdlapack` использует подпрограммы LAPACK для вычисления SVD плотных матриц. Основные этапы алгоритма следующие:

1. Приведение матрицы к двухдиагональной форме

Первым шагом является приведение исходной матрицы A к двухдиагональной форме B с помощью ортогональных преобразований (отражений Хаусхолдера). Это записывается как:

$$A = Q_L B Q_R^\top,$$

где:

- Q_L — ортогональная матрица размера $m \times m$.
- Q_R — ортогональная матрица размера $n \times n$.
- B — двухдиагональная матрица размера $m \times n$, которая имеет ненулевые элементы только на главной диагонали и первой поддиагонали (для $m \geq n$) или наддиагонали (для $m < n$):

$$B = \begin{pmatrix} d_1 & e_1 & & & \\ & d_2 & e_2 & & \\ & & \ddots & \ddots & \\ & & & d_{n-1} & e_{n-1} \\ & & & & d_n \end{pmatrix}.$$

Преобразование к двухдиагональная форме существенно сокращает вычислительную сложность последующих шагов, поскольку операции теперь выполняются над матрицей с простой структурой.

2. Вычисление SVD двухдиагональной матрицы

Для двухдиагональной матрицы B сингулярные значения и векторы вычисляются с использованием итеративных алгоритмов, специально адаптированных для такой структуры. Основные методы включают:

- **QR-алгоритм для двухдиагональных матриц**: Этот алгоритм применяет последовательность ортогональных преобразований для диагонализации двухдиагональной матрицы. Поскольку B имеет простую структуру, каждое итеративное преобразование может быть выполнено эффективно.

- **Алгоритм Кобылыша–Рейша (Divide and Conquer)**: Этот метод разбивает матрицу на подматрицы меньшего размера, решает задачу для них и затем объединяет решения.

- **Метод обобщённых квадратов (Golub–Kahan)**: Используется для вычисления сингулярных значений и векторов путём решения последовательности линейных систем.

В результате этого шага получаются диагональная матрица Σ с сингулярными значениями и ортогональные матрицы U_B и V_B, содержащие сингулярные векторы bidiagonal матрицы B:

$$B = U_B \Sigma V_B^\top.$$

Нужно обратить внимание, что код не содержит явной реализации конкретных алгоритмов, таких как QR-алгоритм для двухдиагональных матриц, Алгоритм Кобылыша–Рейша и метод Голуба–Кахана. Вместо этого, при вызове функции DSSolve через объект DS, LAPACK самостоятельно выбирает наиболее эффективный алгоритм для данной задачи.

3. Обратное преобразование сингулярных векторов

После вычисления SVD двухдиагональной матрицы B необходимо преобразовать полученные сингулярные векторы обратно к исходной матрице A:

$$\begin{aligned} U &= Q_L U_B, \\ V &= Q_R V_B. \end{aligned}$$

Это достигается умножением сингулярных векторов U_B и V_B на ортогональные матрицы Q_L и Q_R, которые были получены на первом шаге. Поскольку Q_L и Q_R являются ортогональными, это преобразование не влияет на ортогональность сингулярных векторов.

Обобщённое сингулярное разложение (GSVD)

Обобщённое сингулярное разложение используется для пар матриц A и B размера $m \times n$ и $p \times n$ соответственно. Цель GSVD состоит в нахождении матриц U, V и X, удовлетворяющих:

$$\begin{aligned} A &= U \Sigma_A X^\top, \\ B &= V \Sigma_B X^\top, \end{aligned}$$

где:

- U — ортогональная матрица размера $m \times m$.

- V — ортогональная матрица размера $p \times p$.

- X — обратимая матрица размера $n \times n$.

- Σ_A и Σ_B — диагональные матрицы размера $m \times n$ и $p \times n$ соответственно, удовлетворяющие условию:

$$\Sigma_A^\top \Sigma_A + \Sigma_B^\top \Sigma_B = I_n.$$

В GSVD сингулярные значения характеризуют относительные вклады матриц A и B в общее решение. Этот метод особенно полезен в задачах, где необходимо учитывать влияние двух различных систем или наборов данных.

Гармоническое сингулярное разложение (HSVD)

Гармоническое сингулярное разложение применяется в задачах, где матрица A связана с метрикой, заданной положительно определённой матрицей Ω. Цель HSVD — найти разложение:

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

при условии, что:

- Векторы U ортонормированы относительно метрики Ω:

  $$U^\top \Omega U = I_p,$$

  где I_p — единичная матрица размера $p \times p$.

- Векторы V ортонормированы в стандартном смысле:

  $$V^\top V = I_p.$$

При этом сингулярные значения σ_i и векторы u_i, v_i удовлетворяют следующим соотношениям:

$$\begin{aligned} Av_i &= \sigma_i \Omega u_i, \\ A^\top \Omega u_i &= \sigma_i v_i. \end{aligned}$$

HSVD находит применение в задачах, где необходимо учитывать весовые коэффициенты или корреляции между элементами данных, например, во взвешенном методе наименьших квадратов или в обработке сигналов.

Имплементация

В этой секции рассмотрим, как алгоритм вычисления сингулярного разложения, описанный ранее, реализован в коде файла `svdlapack.c` из библиотеки SLEPc. Этот файл представляет собой обёртку над подпрограммами LAPACK для вычисления SVD, GSVD и HSVD.

Основные функции, реализующие алгоритм:

- `SVDSetUp_LAPACK`
- `SVDSolve_LAPACK`
- `SVDSolve_LAPACK_GSVD`
- `SVDSolve_LAPACK_HSVD`
- `SVDSetDSType_LAPACK`
- `SVDCreate_LAPACK`

Функция `SVDSetUp_LAPACK`

Эта функция отвечает за инициализацию и настройку объекта SVD для использования методов LAPACK. Основные действия функции:

1. **Получение размеров матриц**:
   - Получение размеров исходной матрицы A с помощью `MatGetSize`.
   - Если вычисляется обобщённое SVD (GSVD), то также получаются размеры матрицы B.
2. **Установка параметров**:
   - Установка параметра `ncv`, который определяет количество сингулярных значений для вычисления.
   - Для стандартного SVD `ncv` устанавливается равным N (числу столбцов матрицы A).
   - Для GSVD `ncv` устанавливается как минимальное из M, N и P, где P — число строк матрицы B.
3. **Выделение памяти**:
   - Выделение памяти для хранения сингулярных векторов и значений с помощью `SVDAllocateSolution`.
4. **Инициализация DS**:
   - Инициализация объекта `DS` (Direct Solver), который будет использоваться для решения задачи SVD с помощью LAPACK.
5. **Проверка параметров**:
   - Проверка на неподдерживаемые функции и установка необходимых флагов.

Функция `SVDSolve_LAPACK`

Эта функция реализует основной алгоритм вычисления SVD для плотных матриц:

1. **Сбор матрицы на одном процессе**:
   - Используется функция `MatCreateRedundantMatrix` для создания копии распределённой матрицы A на одном процессе.
2. **Конвертация матрицы**:
   - Полученная матрица преобразуется в формат `MATSEQDENSE` с помощью `MatConvert`.
3. **Настройка объекта DS**:
   - Матрица A копируется в объект `ds` с помощью `DSGetMat` и `MatCopy`.
   - Устанавливается состояние объекта `ds` как `DS_STATE_RAW`.
4. **Решение задачи SVD**:
   - Вызывается функция `DSSolve`, которая вычисляет сингулярные значения и векторы матрицы A.

5. **Сортировка результатов**:

   - Функция `DSSort` сортирует полученные сингулярные значения и векторы.

6. **Копирование результатов**:

   - Сингулярные значения и векторы извлекаются из объекта `ds` и копируются в соответствующие структуры объекта `svd`.

Функция `SVDSolve_LAPACK_GSVD`

Эта функция отвечает за вычисление обобщённого сингулярного разложения (GSVD):

1. **Сбор матриц A и B**:

   - Матрицы A и B собираются на одном процессе и преобразуются в формат `MATSEQDENSE`.

2. **Настройка объекта DS**:

   - Матрицы A и B устанавливаются в объект `ds` с помощью `DSGetMat`.
   - Устанавливается состояние объекта `ds` как `DS_STATE_RAW`.

3. **Решение задачи GSVD**:

   - Вызывается функция `DSSolve`, которая вычисляет обобщённые сингулярные значения и векторы.

4. **Копирование результатов**:

   - Полученные сингулярные значения и векторы копируются в объект `svd`.

Функция `SVDSolve_LAPACK_HSVD`

Эта функция реализует гармоническое сингулярное разложение (HSVD):

1. **Сбор матрицы A**:

   - Матрица A собирается и конвертируется в последовательный формат.

2. **Установка весов Ω**:

   - Вектор весов ω устанавливается в объекте `ds` с помощью `DSGetMatAndColumn`.

3. **Решение задачи HSVD**:

   - Вызывается функция `DSSolve`, учитывающая метрику Ω.

4. **Копирование результатов**:

   - Результаты копируются в объект `svd`, сохраняется информация о знаках сингулярных значений.

Функция `SVDSetDSType_LAPACK`

Эта функция устанавливает тип объекта `DS` в зависимости от решаемой задачи (SVD, GSVD, HSVD).

Функция SVDCreate_LAPACK

Эта функция создаёт контекст для использования метода LAPACK в SVD.

Приведение матрицы к двухдиагональной форме в коде

Важным шагом алгоритма вычисления SVD является приведение исходной матрицы A к двухдиагональной форме B. В коде файла `svdlapack.c` этот процесс реализован через вызовы подпрограмм LAPACK внутри объекта `DS` (Direct Solver).

В функции `SVDSolve_LAPACK` происходит следующее:

1. **Подготовка матрицы**:
   - Матрица A собирается на одном процессе с помощью `MatCreateRedundantMatrix`.
   - Конвертация матрицы в формат `MATSEQDENSE` с помощью `MatConvert`.
   - Полученная матрица устанавливается в объекте `DS` с помощью `DSGetMat`.
2. **Инициализация объекта DS**:
   - Тип объекта устанавливается как `DSSVD` через функцию `DSSetType(svd->ds, DSSVD)`.
3. **Вызов функции `DSSolve`**:
   - Функция `DSSolve` вызывает соответствующие подпрограммы LAPACK для вычисления SVD.
4. **Выбор алгоритма LAPACK**:
   - В зависимости от размеров матрицы и настроек, `DSSolve` может вызвать следующие подпрограммы:
     - `?GESVD` (например, `DGESVD`): Стандартная подпрограмма для вычисления SVD, использующая последовательную редукцию к двухдиагональной форме.
     - `?GESDD` (например, `DGESDD`): Подпрограмма, реализующая метод разделения и завоевания (Divide and Conquer).
   - Здесь символ `?` заменяется на соответствующий префикс типа данных: `s`, `d`, `c` или `z`.
5. **Редукция к двухдиагональной форме**:
   - Подпрограммы `?GESVD` и `?GESDD` внутри себя приводят матрицу A к двухдиагональной форме B с помощью ортогональных преобразований.
   - Редукция выполняется с использованием подпрограммы `?GEBRD` (например, `DGEBRD`), которая применяет отражения Хаусхолдера.
   - Результат редукции представляется в виде:

     $$A = Q_L B Q_R^\top,$$

     где Q_L и Q_R — ортогональные матрицы, а B — двухдиагональная матрица.
6. **Вычисление SVD двухдиагональной матрицы**:

- Для двухдиагональной матрицы B используются специализированные подпрограммы LAPACK, соответствующие трём основным методам:
  - **QR-алгоритм для двухдиагональных матриц**:
    - Реализован в подпрограмме `?BDSQR`.
    - Использует последовательность вращений Гивенса для диагонализации B.
  - **Алгоритм разделения и завоевания (Divide and Conquer)**:
    - Реализован в подпрограмме `?BDSDC`.
    - Разбивает B на подматрицы меньшего размера, вычисляет SVD для них и объединяет результаты.
  - **Метод обобщённых квадратов (Golub–Kahan)**:
    - Хотя LAPACK не предоставляет прямой реализации метода Голуба–Кахана для SVD двухдиагональной матрицы, аналогичные идеи используются в алгоритмах, оптимизированных для вычисления сингулярных значений и векторов.
- Выбор конкретной подпрограммы может зависеть от настроек и оптимизации LAPACK на данной системе.

7. **Обратное преобразование сингулярных векторов**:

   - После получения сингулярных векторов U_B и V_B для матрицы B, необходимо преобразовать их к сингулярным вектором исходной матрицы A:

   $$U = Q_L U_B, \quad V = Q_R V_B.$$

   - Это преобразование выполняется внутри подпрограмм LAPACK, используя накопленные ортогональные преобразования.

8. **Копирование результатов**:

   - После завершения работы `DSSolve` сингулярные значения σ_i и векторы U, V извлекаются из объекта `DS` и копируются в соответствующие структуры объекта `svd`.

В коде функции `SVDSolve_LAPACK` можно увидеть вызов `DSSolve`, который внутри себя выполняет описанные шаги по редукции матрицы и вычислению SVD.

Структуры данных и их значение

- `SVD`: Главная структура, представляющая объект сингулярного разложения. Хранит информацию о матрицах A, B, параметрах алгоритма и результатах вычислений.

- `DS`: Структура Direct Solver, используется для решения плотных матричных задач с помощью LAPACK.

- `Mat`: Структура для представления матриц в PETSc/SLEPc.

- `Vec`: Структура для представления векторов. Используется для хранения сингулярных векторов.

- `BV`: Блок векторов, представляет собой набор векторов для эффективных операций.

Referenses

- `MatCreateRedundantMatrix`: Создаёт копию распределённой матрицы на одном процессе. [Документация]
- `MatConvert`: Конвертирует матрицу в другой формат. [Документация]
- `DSSolve`: Решает задачу собственных значений или SVD. [Документация]
- `DSSort`: Сортирует вычисленные значения. [Документация]
- `DSSynchronize`: Синхронизирует данные между процессами. [Документация]
- `BVGetColumn` и `BVRestoreColumn`: Доступ к столбцу блок-вектора. [Документация]
- `VecGetArray` и `VecRestoreArray`: Доступ к внутреннему массиву вектора. [Документация]