A Faster Singular Value Decomposition Algorithm for Low Rank Matrices

Команда:

Squad of
Vladimir
Dobrigin

Участники:

1. Владимир Добрыгин
2. Газиз Абдрахман
3. Владислав Гаухов

Постановка задачи

- SVD имеет множество применений во многих алгоритмах, (например, в рекомендательных системах).
- В большинстве этих приложений матрица, подлежащая декомпозиции, представляет собой матрицу низкого ранга и большого размера, скорость вычисления SVD сильно растет при увеличении размерностей.
- В нашем проекте мы реализуем более быстрый стандартный алгоритм для таких матриц и исследуем скорость вычисления SVD по сравнению с библиотечным вариантом и итоговую ошибку аппроксимации.

Описание метода

- Метод заключается в следующих шагах:

1. Разбиении исходной матрицы на блоки меньших размеров.
2. Расчет SVD для полученных блоков.
3. Уменьшение размерностей матриц при помощи усечения малых сингулярных значений.
4. Объединение блоков в итоговое разложение

Разбиение на блоки

Разбиение основывается на факте:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 V_1^T \\ \Sigma_2 V_2^T \end{bmatrix} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} E$$

$$\begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} U_1\Sigma_1 & U_2\Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T & 0 \\ 0 & V_1^T \end{bmatrix} = E \begin{bmatrix} V_1^T & 0 \\ 0 & V_1^T \end{bmatrix}$$

где $X_1 = U_1\Sigma_1 V_1^T, X_2 = U_2\Sigma_2 V_2^T$ - усеченное SVD разложение (рангов k, l), E – матрица размерности $n \times k + l$.

Упрощение вычислений

Для упрощения вычислений, можно использовать замену и QR разложение:

$$Q = U_2 - U_1 U_1^T U_2 = U_o R$$

Тогда выражение принимает вид:

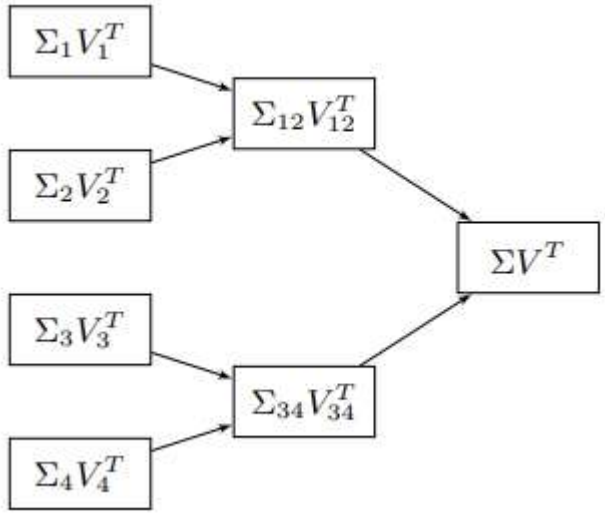
$$\begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} U_1 & U_o \end{bmatrix} \begin{bmatrix} \Sigma_1 & (U_1^T U_2)\Sigma_2 \\ 0 & R\Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} = \begin{bmatrix} U_1 & U_o \end{bmatrix} E \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix}$$

где E уже имеет размерность $k + l \times k + l$.

- После расчета SVD отдельных блоков, необходимо объединить результаты. Для этого вычисляется и усекается SVD матрицы E из предыдущих слайдов.

$U_{11}\Sigma_{11}$	$U_{12}\Sigma_{12}$	·	·	$U_{1c}\Sigma_{1c}$
·	·	·	·	·
·	·	·	·	·
$U_{d1}\Sigma_{d1}$	$U_{d2}\Sigma_{d2}$	·	·	$U_{dc}\Sigma_{dc}$

$\Sigma_1 V_1^T$
·
·
$\Sigma_d V_d^T$



- На каждом шаге блоки объединяются (образуя дерево) сначала по строкам, а затем по столбцам

Итоговый алгоритм

- Выбираются гиперпараметры c, d, γ.
- Матрица разбивается на блоки размера $c \times d$.
- Для каждого блока считается SVD, производится усечение сингулярных значений меньших чем γ.
- Блоки объединяются и итоговое SVD считается по ранее описанным формулам.

Результаты

- При получении результатов строятся матрицы размера $m \times n$, заполненные случайными значениями.
- Исследуется зависимость скорости выполнения программы и величины ошибки от
  - Размера матрицы
  - Размера блоков
  - γ
  - Ранга матрицы
  - Распределения сингулярных значений матрицы.

Зависимость от γ

- Гиперпараметры: $m = n = 10000, rg(A) = 50$, сингулярные значения $\sigma_i \in U(0,1], \mathrm{c} = \mathrm{d} = 1000$.
- Время работы np.linalg.svd(A): **231.5с**.

γ	Время работы алгоритма, сек.	Ошибка
0.0001	75.3	2.7e-15
0.1	71	2.9e-2
0.5	65.3	0.4

- При увеличении γ сильно растет ошибка, время уменьшается не так значительно

Зависимость от распределения сингулярных значений

- Изменим распределение сингулярных чисел на распределение Парето с параметром a

a	Время работы алгоритма, сек.	Ошибка
0.5	29	1e-3
1	31	1e-3
3	31	1e-3
10	33	1e-3
20	32	1e-3

- Особо изменений нет

Зависимость от размеров блока

- Зафиксируем теперь $\gamma = 0.01$, сингулярные значения $\sigma_i \in U(0,1]$.
- Изменим размер блока:

$c \times d$	Время работы алгоритма, сек.	Ошибка
10000×100	118.2	1e-3
100×10000	162.9	1e-3
5000×100	88.3	1e-3
2500×100	93.87	1e-3
1250×100	133.8	1e-3
$\mathbf{10000 \times 1000}$	**29**	1e-3
10000×2000	49.9	1e-3
5000×1000	42.3	1e-3
2500×1000	170	1e-3
$\mathbf{1000 \times 10000}$	**76**	1e-3

Зависимость от размеров блока (продолжение)

Выводы из таблицы на предыдущем слайде:

- На блоках размера $c \times d$ алгоритм работает быстрее при $c \gg d$.
- Наилучшие результаты показывает разбиение по столбцам.
- Ошибка не зависит от размера блока.

Зависимость от ранга матрицы

- Зафиксируем теперь еще и размер блока: 10000×1000
- Рассмотрим, что происходит при изменении ранга:

rg(A)	Время работы алгоритма, сек.	Ошибка
5	29	1e-3
25	31	
50	32.8	1e-3
100	35.3	1e-3
200	39.9	1e-3
500	59.2	2e-3
1000	146	2e-3
2000	263	2e-3

большого ранга алгоритм работает медленнее библиотечного SVD.

Зависимость от размера матрицы

$m \times n$	$c \times d$	Время работы алгоритма, сек.	Ошибка	Время работы np.linalg.svd, сек
10000×10000	10000×1000	**32**	2e-3	**248**
100000×1000	10000×1000	26.7	3e-15	10.1
1000000×100	10000×100	16.8	2e-15	3.3
10000000×10	100000×10	7.5	6e-15	2.5
1000×100000	1000×1000	63.4	2e-15	12
1000×100000	1000×10000	63.7	2e-15	12
100×1000000	100×2500	71.4	2e-3	14.2
10×10000000	10×1000	44.5	1e-15	4

Зависимость от размера матрицы (продолжение)

Выводы из таблицы на предыдущем слайде:

- Быстрее всего алгоритм работает с «высокими и худыми» и квадратными матрицами.
- Более того, с «высокими и худыми» блоками алгоритм работает заметно лучше.
- С «низкими и толстыми» матрицами скорость заметно ниже.
- np.linalg.svd работает быстрее, если n или m достаточно мало, но ощутимо медленнее в случае больших n и m.

Выводы

- Основные результаты:
  - Реализовали алгоритм из статьи.
  - Проверили на практике, что на больших размерностях алгоритм действительно даёт ускорение с несущественной ошибкой.
  - Исследовали скорость работы для блоков разных размеров, разного параметра уссечения на матрицах разного размера, ранга, распределений сингулярных значений.
- Как хотелось бы улучшить:
  - Добавить поддержку параллельных вычислений для ускорения.
  - Избавиться от возможных утечки памяти в нашей реализации.

References

- Код - https://github.com/gazizabdrakhman/NLA_Project1/
- Статья - A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices(V. Vasudevan, M.Ramakrishna) https://arxiv.org/pdf/1710.02812.pdf