УДК 681.3

Олександр О. Марченко, к.ф.- м.н., доц.
Еміл М. Насіров, аспірант
Степан М. Паламарчук, студент

Паралелізація невід'ємної факторизації розріджених матриць надвеликої розмірності

У роботі описано побудову моделі паралелізації обчислення невід'ємної факторизації розріджених матриць надвеликої розмірності. Реалізації запропонованих моделей були порівняні в обробці надвеликої матриці.

Ключові слова: штучний інтелект, комп'ютерна лінгвістика, паралельні обчислення

Київський національний університет імені Тараса Шевченка, 03680, м. Київ, пр-т. Глушкова 4д, e-mail: enasirov@gmail.com

Oleksandr O. Marchenko, Ph.D. (Physics & Mathematics), Associate Professor
Emil M. Nasirov, Postgraduate Student
Stepan M. Palamarchuk, Student

Parallel non-negative huge sparse matrix factorization

The paper proposes parallel methods of non-negative huge sparse matrix factorization. Implementations of proposed methods were tested and compared on huge matrix processing.

Keywords: artificial intelligence, computational linguistics, parallel computations

Taras Shevchenko National University of Kyiv, 03680, Kyiv, Glushkova ave., 4d, e-mail: enasirov@gmail.com

Статтю представив чл.-кор. НАН України, д.ф.- м.н., проф. Анісімов А. В.

Сьогодні невід'ємна факторизація матриць та тензорів є дуже популярною технологією в штучному інтелекті взагалі, та в комп'ютерній лінгвістиці зокрема. Використовуючи невід'ємну факторизацію в рамках парадигми латентно-семантичного аналізу, комп'ютерні лінгвісти застосовують даний підхід для розв'язання таких прикладних задач, як класифікація, кластеризація текстів та термінів[1, 2], побудова мір семантичної близькості[3], автоматичне виділення лінгвістичних структур та відношень (*Selectional Preferences* [4] та *Verb Sub-Categorization Frames* [5]) і багато інших.

Дана робота описує побудову моделі паралелізації обчислення невід'ємної факторизації розріджених матриць надвеликої розмірності, що представляє особливий інтерес та актуальність з точки зору її застосування у великих NLP системах загального тематичного призначення, не обмежених використанням лише для вузьких предметних областей.

Задача невід'ємної факторизації розріджених матриць надвеликої розмірності постала в процесі розробки системи визначення міри семантичної близькості-зв'язності за технологією Латентного Семантичного Аналізу[6]. Для побудови системи був оброблений великий текстовий корпус статей англомовної Вікіпедії. Обробка корпусу полягала у лексичному аналізі та лематизації лексем речень статей та в обчисленні частот вживання множини слів та словосполучень англійської мови в складі різних статей англомовної Вікіпедії. В результаті була побудована велика матриця Слова×Статті яка містила частотну оцінку вживання слів в текстах статей Вікіпедії. Розмірність матриці дорівнює 2,437,234 слів-словосполучень на 4,475,180 статей англомовної Вікіпедії. Після цього для усунення з матриці випадкових даних був встановлений пороговий рівень T=3, частотні оцінки в матриці, які менші за пороговий рівень T, були обнулені). В результаті була побудована розріджена матриця надвеликого розміру - 156,236,043 ненульових елементів при розмірі 2,437,234×4,475,180. Для невідємної факторизації розрідженої матриці такого розміру знадобилася розробка спеціальної моделі паралелізації матричних обчислень, яка була реалізована із

© О. О. Марченко, Е. М. Насіров, С. М. Паламарчук, 2014

застосуванням паралельних розподілених обчислень та обчислень на GPU.

Алгоритм невід'ємної матричної факторизації

Алгоритм невід'ємної матричної факторизації виконує декомпозицію невід'ємної матриці V на невід'ємні матриці W та H таким чином, щоб $V \approx WH$

В якості функції оцінки може бути використана функція виміру відстані між двома невід'ємними матрицями. Однією з таких мір є квадрат Евклідової метрики:

$$\mu = \|A - B\|^2 = \sum_{ij} (A_{ij} - B_{ij})^2$$

Така цільова функція обмежена знизу. Нижня границя 0 досягається тоді і тільки тоді коли $A = B$.

Отже, при використанні Евклідової метрики, факторизація матриці полягає в мінімізації $\|V - WH\|^2$ при умові невід'ємності *W* та *H*.

Така цільова функція не зростаюча при наступних правилах:

$$H_{ij} \leftarrow H_{ij} \frac{(W^T V)_{ij}}{(W^T WH)_{ij}} \quad (1)$$

$$W_{ij} \leftarrow W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}} \quad (2)$$

Виконання ітерацій алгоритму продовжується дот тих пір, поки не буде досягнута стаціонарна точка, або не буде виконана максимальна кількість ітерацій[7].

Аналіз моделі

В Таблиці 1 представлено необхідні об'єми пам'яті для зберігання матриць W та H при різних k при використанні типу даних float (32bit).

K	*100*	*200*	*300*
W	0.98Gb	1.95Gb	2.92Gb
H	1.79Gb	3.58Gb	5.37Gb
Всього	2.76Gb	5.53Gb	8.29Gb

Таблиця 1. Необхідні об'єми пам'яті для зберігання матриць W та H при різних k

Описаний алгоритм на кожній ітерації потребує в 2 рази більше пам'яті для збережень матриць (не враховуючи потреби на збереження початкової матриці V). Враховуючи такі потреби в пам'яті, для виконання алгоритму на одному комп'ютері необхідно проводити збереження даних на жорсткий диск. Далі буде розглянуто та порівняно 2 підходи до паралелізації алгоритму: локальний та розподілений.

Паралелізація алгоритму з використанням GPU

Для спрощення, зробимо заміну в (1) та (2): $H' = H^T$. Отримаємо

$$(H'_t)_{ij} = (H'_{t-1})_{ij} \frac{(V^T W_{t-1})_{ij}}{(H'_{t-1} W_{t-1}^T W_{t-1})_{ij}} \quad (4)$$

$$(W_t)_{ij} = (W_{t-1})_{ij} \frac{(VH'_t)_{ij}}{(W_{t-1} H_t'^T H'_t)_{ij}} \quad (5)$$

Таким чином, обидві формули (4) та (5) ми можемо представити в одному вигляді, завдяки заміні H', *W* та V^T, або *W*, H' та V на A, B та S відповідно:

$$A_{ij} = A_{ij} \frac{(SB)_{ij}}{(AB^T B)_{ij}} \quad (6)$$

Формула (6) може бути представлена як послідовність чотирьох кроків(7):

$$C = SB \quad (7a)$$

$$K = B^T B \quad (7b)$$

$$D = AK \quad (7c)$$

$$A_{ij} = Aij \frac{C_{ij}}{D_{ij}} \quad (7d)$$

Такий порядок обчислень є оптимальним для виразу (6). Ці кроки маюсь обчислювальну складність $O(k * (nnz(S) + n))$, $O(k^2 m)$, $O(k^2 n)$ та $O(kn)$ відповідно, де $nnz(S)$ – кількість ненульових елементів матриці S. Перші три кроки(7a-7c) підтримуються бібліотеками CUDA cuSPARSE та cuBLAS. Четвертий крок (7d) для виконання потребує реалізації власного ядра GPU, але в той же час, це відносно швидка операція і тому може бути виконана на CPU.

Так як матриці занадто великі для збереження в пам'яті GPU, ці операції(7a-7d) повинні виконуватись над частинами, з врахуванням

необхідності зменшення об'ємів обміну даними з графічним адаптером.

Для виразу (7a) повинні розкласти матриці $S = (S_1'|S_2'|\dots|S_t')^T$ та $B = (B_1|B_2|\dots|B_r)$ та підрахувати C:

$$C=\begin{pmatrix} S_1'B_1 & \dots & S_1'B_r \\ \dots & \dots & \dots \\ S_t'B_1 & \dots & S_t'B_r \end{pmatrix} \quad (8)$$

Для виразу (7b) варто розглядати $B = (B'_1|B'_2|\dots|B'_t)$, а отже $K = B'^T_1 B'_1 \dots B`^T_t B`_t$. Підрахунок цього виразу не потребує додаткового завантаження будь-яких матриць в пам'ять графічного адаптера після виконання (7a).

Для підрахунку (7c) ми можемо залишити в пам'яті GPU матрицю K і помножити матрицю A як стовпчик блоків.

Також можна зменшити використання пам'яті завдяки комбінуванню (7d) та (7c), так як (7d) – по елементне правило і єдине інше правило, в якому використовується матриця A – (7c). Не буде потреби зберігати в пам'яті матрицю D, якщо виконувати (7d) на блоці з матриці A, який необхідний для обчислення певного блоку матриці D.

Складність підрахунку Евклідової метрики μ між початковою та факторизовано моделлю складає $O(nm)$. Такий підрахунок просто реалізовується з використанням графічних адаптерів.

Розподілений алгоритм

Наступним кроком для покращення швидкодії є використання мережі ПК для проведення паралельних розподілених обчислень. Розглянемо можливі розподіленні моделі. Припустимо, що мережа складається з двох вузлів. Можливі наступні моделі розподілення обчислень:

1. *Підраховувати* W *та* H' *окремо на різних вузлах.* Таким чином обидва вузли будуть перебувати в одному з двох альтернативних режимів. Або будуть вузлом підтримки іншого вузла (передача даних на інший вузол) або будуть проводити підрахунки (обчислювати матрицю за яку відповідають, за допомогою даних, отриманих з вузла підтримки). В такій моделі розподілення на кожній ітерації ми повинні будемо передати по мережі дві матриці такого ж розміру, як і W та H. Також вузол розрахунків буде в основному простоювати, тому що (7a) є найбільш витратним кроком з усіх чотирьох.
2. *Розділити матриці* W *та* H' *на блоки та розподілити їх між вузлами.* $H' = (H'_1|H'_2)$ та $W = (W_1|W_2)$. Робимо перший вузол відповідальним за H'_1, W_1, другий за H'_2, W_2. При такому підході кожен вузол виступає як у ролі вузла підтримки, так і у ролі вузла розрахунків одночасно. В такій моделі вузли повинні передавати по мережі об'єм даних рівний $1.5 \cdot (sizeof(W) + sizeof(H))$, де $sizeof(X)$ – об'єм пам'яті, необхідний для зберігання матриці X.
3. *Розділити матриці* W *та* H' *на блоки та розділити їх між вузлами.* $H' = (H_1|H_2)^T$ та $W = (W_1'|W_2')^T$. Робимо перший вузол відповідальним за H_1, W_1', другий за H_2, W_2'. В цій моделі, аналогічно до попередньої, кожен вузол мережі виступає одночасно як у ролі вузла підтримки, так і у ролі вузла розрахунків. Об'єм передачі даних вузлом рівний $sizeof(W) + sizeof(H)$

В кожній з представлених моделей також є необхідність передачі одної або більш ніж одної матриці [k;k], але їх розміром можна знехтувати в порівнянні з матрицями *W* та *H*.

Очевидно, що третя модель найкраще підходить з точки зору мережевого обміну даними та використання GPU. Тому саме ця модель була використана в нашому дослідженні.

Варто також зазначити, що ріст мережі призведе до експоненціального росту сумарного об'єму переданих даних, хоча об'єм передачі даних одного вузла буде не більш ніж $2 \cdot (sizeof(W) + sizeof(H))$.

Для кращого розподілення розрахунків між вузлами мережі, враховуючи розрідженість початкової матриці V, необхідно виконати перестановки рядків та стовпчиків для нормалізації кількості ненульових елементів в кожному блоці.

Результати аналізу

Був реалізований розподілений алгоритм з використанням GPU, який був описаний вище, та проведена факторзація розрідженої матриці оцінок части вживання слів англійської мови в різних статтях Вікіпедії. Для порівняння швидкодії була також реалізована локальна модель з використанням GPU та жорсткого диску для запису даних.

Обидві моделі працювали з однаковими вхідними даними. Локальна версія була запущена на одному з вузлів з тими ж обмеженнями пам'яті. Для проведення тестів були використані наступні апаратні засоби: Intel Core i7 CPU, NVIDIA GeForce GTX560 1Gb, 8Gb RAM, 1Gbit LAN, SATA III HD.

В таблиці 2 порівняно час та ресурси необхідні для виконання однієї ітерації в локальній та розподіленій версії. В таблиці 3 порівняно час та ресурси, необхідні для підрахунку функції оцінки μ. Данні таблиць отриманні при k=300.

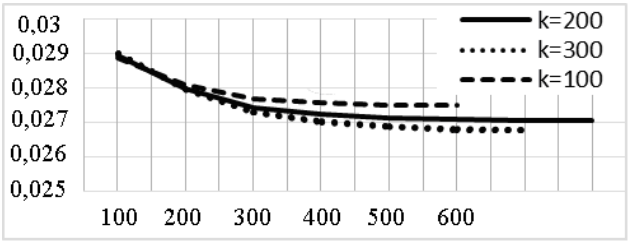
	Local	Distributed
Прочитано	34.44Gb	6.22Gb
Записано	16.58Gb	6.22Gb
Час ітерації (розрахунки)	58s	62s
Час ітерації (I/O)	729s	287s

Таблиця 2. Результати порівняння виконання ітерації локальної та розподіленої версії

	Local	Distributed
Прочитано	13.66Gb	6.22Gb
Записано	0	6.22Gb
Час (розрахунки)	45865s	11371s
Час (I/O)	192s	280s

Таблиця 3. Результати порівняння обчислення функції оцінки μ

На графіку 1 показано отриману залежність якості результатів факторизації від k та кількості ітерацій.



Графік 1

Висновки

В роботі описано локальний та розподілений підходи до невід'ємної факторизації надвеликих матиць. В результаті роботи було отримано високі показники швидкодії роботи розподіленого алгоритму в порівнянні з локальним. Подальша робота полягає в аналізі та зменшенні об'ємів передачі даних між вузлами, а отже, і необхідного для цього часу.

Список використаних джерел

1. *Xu W., Liu X., Gong Y.* Document-clustering based on 4n-negative matrix factorization// In Proceedings of SIGIR'2003, July 28–August 1, Toronto, CA, pp. 267–273.
2. *Farial Shahnaz, Michael W. Berry, V. Paul Pauca, Robert J. Plemmons* Document clustering using nonnegative matrix factorization// Information Processing and Management: Volume 42 Issue 2, March 2006, p.p. 373 – 386
3. *Anatoly Anisimov, Oleksandr Marchenko, Andrey Nikonenko, Elena Porkhun, Volodymyr Taranukha* Ukrainian WordNet: Creation and Filling// FQAS-2013, p.p. 649-660
4. *Tim Van de Cruys* A Non-negative Tensor Factorization Model for Selectional Preference Induction. Journal of Natural Language Engineering, 2010, 16(4), p.p. 417–437.
5. *Tim Van de Cruys, Laura Rimell, Thierry Poibeau, and Anna Korhonen* Multi-way Tensor Factorization for Unsupervised Lexical Acquisition// Proceedings of COLING-2012, p.p. 2703–2720.
6. *Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman* Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science: – 1990: – 41 (6): – pp. 391–407.
7. *Daniel D. Lee and H. Sebastian Seung* Algorithms for non-negative matrix factorization// In In NIPS, MIT Press, 2000, p.p. 556-562.

Надійшла до редколегії 16.12.13