Анализ алгоритма приближенного умножения матриц

РУКОВОДИТЕЛЬ: ГРИГОРЬЕВ СЕМЁН
ВЫПОЛНИЛА: МУРЫЧЕВА НАТАЛЬЯ

СПбАУ
2018 г.

Введение

- Умножение матриц используется во многих задачах, поэтому актуальным остается вопрос возможности улучшения асимптотики
- В задаче поиска транзитивного замыкания графа встречаются разреженные - булевые матрицы большой размерности
- Было решено проверить идею о компромиссе между временем работы и точностью, а именно алгоритм Расмуса Пага

Аналоги

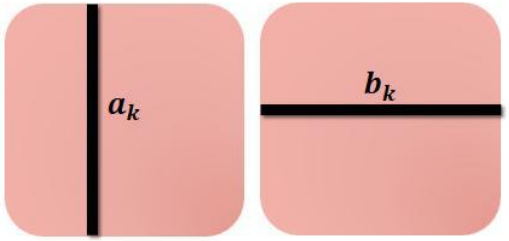
- **Разреженные матрицы**
  - Существует большое количество оптимизаций наивного алгоритма
  - Множество подходов к распараллеливанию вычислений
  - Алгоритм Кохена, асимптотика $\boldsymbol{O(N \log N)}$, где N – верхняя оценка $nnz(A), nnz(B)$
  - Алгоритм Юстера-Звика, асимптотика $\boldsymbol{O\left(N^{0.7} n^{1.2} + n^{2+o(1)}\right)}$
- **Булевые матрицы**
  - Алгоритм четырех русских с модификацией, асимптотика $\boldsymbol{O(n^2 / \log n)}$
- **Приближенное умножение**
  - Алгоритм Бока-Чалакома для разреженных, асимптотика $\boldsymbol{O(n \log n)}$, высокая точность

Задачи

- Провести теоретический анализ применимости алгоритма приближенного умножения матриц для задачи поиска транзитивного замыкания графа
- Реализовать данный алгоритм для разреженных - булевых матриц
- Провести численные эксперименты на случайно сгенерированных матрицах
- Провести численные эксперименты на реальных данных

Алгоритм и асимптотика

- **Алгоритм**
  a. Используем $\boldsymbol{Count\ Sketch}$, позволяющий сжимать данные
  b. Столбцы и строки матриц A и B подаются $\boldsymbol{CS}$
  c. $\boldsymbol{Count\ Sketch}$ – это полином, степени $\boldsymbol{b}$. Используем $\boldsymbol{FFT}$ для улучшения асимптотики



- **Асимптотика**
  a. $\boldsymbol{O(N + n\ b \log b)}$, где N – верхняя оценка на $\mathrm{nnz}(A)$ и $\mathrm{nnz}(B)$.
  b. Для повышения точности, будем считать $\boldsymbol{d}$ $Count\ Sketch$-ей, выбирая для каждого элемента $(\boldsymbol{C})_{\boldsymbol{i^* j^*}}$ результирующей матрицы медиану.
  Итоговая асимптотика $\boldsymbol{O\big(d(N + n\ b \log b)\big)}$

Оценки и пред подсчет

- **Оценки**
  a. В случае $\boldsymbol{d = 1}$ каждый элемент результирующей матрицы является несмещенной оценкой с дисперсией равной $\|\boldsymbol{C}\|_{\boldsymbol{F}}^{\boldsymbol{2}}/\boldsymbol{b}$
  b. Пусть $\boldsymbol{nnz(C) \leq b/8}$, $\boldsymbol{d \geq 6 \ln n}$. Тогда алгоритм находит произведение матриц точно с высокой вероятностью

- **Пред подсчет** (оценка на $\boldsymbol{nnz(C)}$)
  a. Грубая верхняя оценка, асимптотика $\boldsymbol{O(1)}$
  b. Более точная верхняя оценка, асимптотика $\boldsymbol{O(n \log n)}$
  c. Точная верхняя оценка, очень плохая асимптотика

Выводы (теория)

- В случае $\boldsymbol{8nnz(C) > n}$, получаем $\boldsymbol{b = n}$
  - Следовательно асимптотика алгоритма становится $\boldsymbol{O\left(d\left(N + n^2 \log n\right)\right)}$.
  - Мы не можем воспользоваться второй оценкой на точность
  - Первая оценка дает недостаточно информации для постуточнения ответа

- Размер $\boldsymbol{CS}$ зависит от неизвестного вначале значения $\boldsymbol{nnz(C)}$
  - Предложенный автором алгоритма способ нахождения верхней границы $\boldsymbol{nnz(C)}$ имеет плохую асимптотику
  - Алгоритм грубой оценки работает за $\boldsymbol{O(1)}$, но не является достаточно точным, что скажется на точности решения

Численные эксперименты

n	$nnz(A)$, $nnz(B)$, %	$nnz(C)$, %	$8nnz(C) \leq b$	b	Потеряли, %	Приобрели, %	Нашли, %
512	30	100	Нет	512	35	0	65
512	2	19	Нет	512	32	73	67
512	0,5	1,3	Нет	512	8	91	92
512	0.05	0,02	Да	512	0	0	100
512	0,01	0,001	Да	16	0	92	100
512	0,05	0,01	Да	256	0	4	100
512	0,01	0,001	Да	32	0	58	100
512	0,001	0	Да	1	0	0	100
312	12	0,01	Да	128	0	0	100
312	30	0	Да	2	0	0	100
337	48	0,02	Да	256	0	0	100
337	10	0,01	Да	128	0	0	100
341	35	0,1	Нет	341	0	0	100

Численные эксперименты

n	$nnz(A)$, $nnz(B)$, %	$nnz(C)$, %	$8nnz(C) \leq b$	b	Потеряли, %	Приобрели, %	Нашли, %
512	0,5	1,3	Нет	512	8	92	92
512	0,05	0,01	Нет	165	52	0	48
512	0,03	0,004	Нет	55	27	0	73
512	0,01	0,001	Нет	5	0	0	100
512	0,001	0	Да	5	0	0	100
512	0,5	1,3	Нет	512	8	91	92
512	0,05	0,01	Да	330	33	0	30
512	0,03	0,004	Нет	55	44	0	56
512	0,01	0,001	Да	10	33	0	67
312	12	0,01	Нет	20	44	0	56
312	30	0	Да	140	0	0	100
337	10	0,01	Нет	30	25	0	75
341	38	0	Да	341	0	0	100

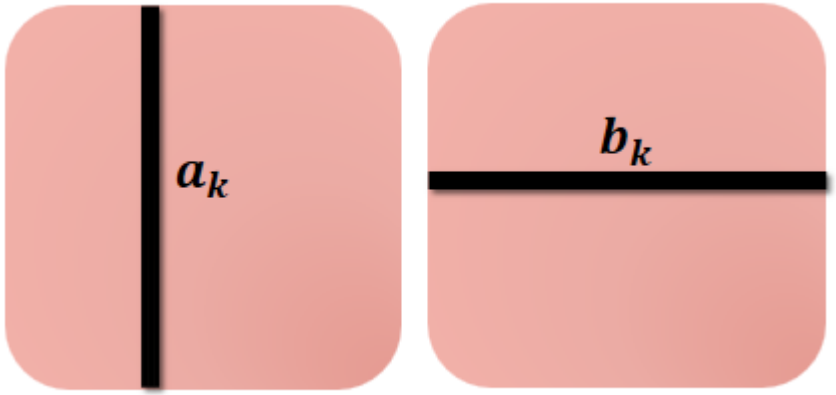
Выводы (практика)

- Алгоритм показал ожидаемые результаты на случайно сгенерированных матрицах
- На реальных матрицах хорошие результаты в случае правильной верхней оценки на $\boldsymbol{nnz}(\boldsymbol{C})$
- Вопрос о выборе алгоритма для нахождения верхней оценки на $\boldsymbol{nnz}(\boldsymbol{C})$ остается открытым
- Предложено рассмотреть другие быстрые алгоритмы умножения разреженных матриц (алгоритмы Кохена, Юстера-Звика, Бока и четырех русских для булевых)

Результаты

- Реализован алгоритм приближенного умножения для разреженных – булевых матриц
- Проведены численные эксперименты на сгенерированных данных
- Проведены численные эксперименты на реальных матрицах
- Предложены альтернативные подходы к решению задачи

Алгоритм



*Посчитаем **Count Sketch** для каждого элемента, тогда* $\sum_{k=1}^{n} p_{a_k b_k}$ *будет скетчем всей матриц*

***Count Sketch** – это полином. Как его можно построить?*

Пусть есть две функции

$s: [n] \to \{-1, +1\}$ *- рандомная хэш-функция такая, что*

$$s(i,j) = s_1(i)s_2(j)$$

$h: [n] \to \{0, \dots, b\ -1\}$ *- рандомная функция нарезки такая, что*

$$h(i,j) = \ h_1(i) + \ h_2(j)\ mod\ b$$

*Тогда **Count Sketch** для вектора u будет равен* $p_u^{h_t, s_t}(x) = \sum_{i=1}^{n} s_t(i) u_i x^{h_t(i)}$

И искомый скетч для пары векторов u, v можно найти, как $p_{uv}^{*}(x) = \sum_{k} \sum_{\substack{i,j \\ h(i,j)=k}} s(i,j)(uv)_{ij} x^k$

*Улучшим асимптотику, используя **FFT** для перемножении полиномов.*

Имеем $p_{uv}^{*}(w^t) = \left(\sum_i s_1(i) u_i w^{t\, h_1(i)}\right)\left(\sum_j s_2(j) v_j w^{t\, h_2(j)}\right)$

Псевдокод

tion $s : [n] \to \{-1, +1\}$. That is, the sketch is the vector $(c_0, \ldots, c_{b-1})$ where $c_k = \sum_{i,\, h(z_i)=k} s(z_i)\, w_i$. An unbiased estimator for the total weight of an item z is $c_{h(z)} s(z)$. To

function CompressedProduct(A, B, b, d)
  for $t := 1$ **to** d **do**
    $s_1[t], s_2[t] \in_R \text{Maps}(\{1, \ldots, n\} \to \{-1, +1\})$
    $h_1[t], h_2[t] \in_R \text{Maps}(\{1, \ldots, n\} \to \{0, \ldots, b-1\})$
    $p[t] := \underline{0}$
    for $k := 1$ **to** n **do**
      $(pa, pb) := (\underline{0}, \underline{0})$
      for $i := 1$ **to** n **do**
        $pa[h_1(i)] := pa[h_1(i)] + s_1[t](i)\, A_{ik}$
      end for
      for $j := 1$ **to** n **do**
        $pa[h_2(j)] := pb[h_2(j)] + s_2[t](j)\, B_{kj}$
      end for
      $(pa, pb) := (\text{FFT}(pa), \text{FFT}(pb))$
      for $z := 1$ **to** b **do**
        $p[t][z] := p[t][z] + pa[z]\, pb[z]$
      end for
      end for
    end for
  end for
  for $t := 1$ **to** d **do** $p[t] := \text{FFT}^{-1}(p[t])$ **end for**
  return (p, s_1, s_2, h_1, h_2)
end

function Decompress(i, j)
  for $t := 1$ **to** d **do**
    $X_t := s_1[t](i)\, s_2[t](j)\, p[t][(h_1(i) + h_2(j)) \bmod b]$
  return Median$(X_1, \ldots, X_d)$
end