Метод Гаусса нахождения определителя матрицы с выбором наибольшего элемента по всей матрице.

Арутюнян Ани

24 ноября 2024 г.

1 Постановка задачи

Найти определитель, с помощью блочного метода Гаусса, с поиском наибольшего элемента в матрице

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

2 Описание алгоритма

Пусть есть матрица

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \tag{1}$$

Теперь запишем матрицу A' - блочный вид матрицы A

$$A' = \begin{pmatrix} A_{1,1}^{m\times m} & A_{1,2}^{m\times m} & \cdots & A_{1,k}^{m\times m} & A_{1,k+1}^{m\times l} \\ A_{2,1}^{m\times m} & A_{2,2}^{m\times m} & \cdots & A_{2,k}^{m\times m} & A_{2,k+1}^{m\times l} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{k,1}^{m\times m} & A_{k,2}^{m\times m} & \cdots & A_{k,k}^{m\times m} & A_{k,k+1}^{m\times l} \\ A_{k+1,1}^{l\times m} & A_{k+1,2}^{l\times m} & \cdots & A_{k+1,k}^{l\times m} & A_{k+1,k+1}^{l\times l} \end{pmatrix} \tag{2}$$

Далее используя элементарные преобразования блоков мы приведем ее к верхнетреугольному виду. Это будет корректным преобразованием, так как по сути, это все те же элементарные преобразования блоков и столбцов.

Аналогично выбору наибольшего элемента a_{11} в обычном методе Гаусса(для уменьшения погрешности), мы будем выбирать блок имеющий наименьшую норму обратной матрицы, таким образом мы будем максимально избегать матриц близких к вырожденным. Далее этот блок мы переставим на место верхнего левого блока матрицы $\overline{A}$(передвижением строк и столбцов из блоков)

3 Формулы преобразования p строки и зануления столбца

Опишем формулы по которым происходит умножение строки блоков на обратный:

$$A_{p,s}^{m\times m} = D_p^{m\times m} A_{p,s}^{m\times m},\ s = p+1,\dots,k \tag{3}$$

$$A_{p,k+1}^{m\times l} = D_p^{m\times m} A_{p,k+1}^{m\times l} \tag{4}$$

Теперь опишем формулы задающие изменение столбцов, соответственно блоки под нашим главным элементом в р столбце они обнулят:

$$A_{i,j}^{m\times m} = A_{i,j}^{m\times m} - A_{i,p}^{m\times m} A_{p,j}^{m\times m},\ i = p+1,\dots,k;\ j = p+1,\dots,k \tag{5}$$

$$A_{i,k+1}^{m\times l} = A_{i,k+1}^{m\times l} - A_{i,p}^{m\times m} A_{p,k+1}^{m\times l},\ i = p+1,\dots,k \tag{6}$$

$$A_{k+1,j}^{l\times m} = A_{k+1,j}^{l\times m} - A_{k+1,p}^{l\times m} A_{p,j}^{m\times m},\ j = p+1,\dots,k \tag{7}$$

$$A_{k+1,k+1}^{l\times l} = A_{k+1,k+1}^{l\times l} - A_{k+1,p}^{l\times m} A_{p,k+1}^{m\times l} \tag{8}$$

В них также были учтены особые случае(если они возникают) в $A_{k+1,k+1}$

Итоговый вид нашей матрицы после того как мы приведем ее к верхнетреугольному виду:

$$A' = \begin{pmatrix} E_{1,1}^{m\times m} & A_{1,2}^{m\times m} & \cdots & A_{1,k}^{m\times m} & A_{1,k+1}^{m\times l} \\ 0 & E_{2,2}^{m\times m} & \cdots & A_{2,k}^{m\times m} & A_{2,k+1}^{m\times l} \\ 0 & 0 & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & E_{k,k}^{m\times m} & A_{k,k+1}^{m\times l} \\ 0 & 0 & \cdots & 0 & E_{k+1,k+1}^{l\times l} \end{pmatrix} \tag{9}$$

После умножение каждого блока на первоначальный, произведение диагональных элементов этой верхнетреугольной матрицы и даст на значение определителя первоначальной матрицы.

4 Функции получения и заполнения блока, в них также будет описан общий формат блоков, с которыми мы работаем

Реализация функций get_block , put_block:

```
void get_block (double *matrix, double *ptr_block, int n, int m, int i, int j)
{
  int k = n / m;
  int l = n - k * m;
  int col = (j < k ? m : l);
  int row = (i < k ? m : l);
  double *matrix_block = matrix + i * n * m + j * m;
  for (int a = 0; a < row; a++)
    {
      for (int b = 0; b < col; b++)
        {
          ptr_block[a * m + b] = matrix_block[a * n + b];
        }
    }
}



void put_block (double *matrix, double *ptr_block, int n, int m, int i, int j)
{
  int k = n / m;
  int l = n - k * m;
  int col = (j < k ? m : l);
  int row = (i < k ? m : l);
  double *matrix_block = matrix + i * n * m + j * m;
  for (int a = 0; a < row; a++)
    {
      for (int b = 0; b < col; b++)
        {
          matrix_block[a * n + b] = ptr_block[a * m + b];
        }
    }
}
```

Описание параметров фуннкций:

1. $matrix$, matrix_block - Указатель на нашу матрицу A и на нужный блок в ней
2. ptr_block - Указатель в памяти, где мы храним блок, с которым рабтаем(в данном случае либо записываем в него, либо из него)
3. n, m - Размерность матрицы A и блоков, с которыми раотаем(matrix_block, ptr_block)
4. i, j - Индексы matrix_block для матрицы A', используя их и арифметические операции над указателями, мы получаем указатель на начало matrix_block в матрице A

Описание алгоритма функций:

Как видно, в обоих алгоритмах мы вначале вычисляем необходимые нам параметры, такие как:

1. k - количество "целых"блоков
2. l - по сути размерности оставшихся частей после выделения цельных блоков, может быть как по строке, так и по столбцу(но одинаков, так как матрица была квадратной)

3. row, col - это количетсво строк/столбцов в нашем блоке(либо он целый, либо какая-то его часть урезана, либо он квадратный урезанный - $l \times l$)

Далее на примере get _block видно, что мы определяем, какой блок нам предстоит поместить в ptr_block далее итерируемся по matrix _block и считываем его. Для метода put _block всё в целом аналогично, кроме того, что мы уже записываем из нашего рабочего блока ptr _block в соответствующий matrix _block.

5 Оценка сложности алгоритма

Для начала выпишем некоторые базовые операции из алгоритма, для которых сложность уже установлена в большинстве учебных пособий и одно уточнение:

1. Сложность нахождения обратной матрицы размера $m \times m$ обычным методом Гаусса равна $\frac{8}{3} \times m^3 + O(m^2)$.

2. Сложность вычитания или сложения матриц одинаковой размерности m на m равна m^2 операций.

3. Сложность умножения двух матриц одинаковой размерности m на m равна $2m^3 - m^2$.

4. Будем рассмативать ситуацию, когда $\frac{n}{m}$ целое число, так как остаточная часть глобально на сложность повлиять не может

Рассчет сложности алгоритма: Для начала операции с матрицей A:

1. Нахождение всех обратных матриц, для выбора нужного блока: $(k-p)^2$ раз

$$\begin{aligned}(\frac{8}{3}m^3 + O(m^2)) \times \sum_{i=1}^{k} i^2 = (\frac{8}{3}m^3 + O(m^2)) \times (\frac{1}{3}k^3 + \frac{1}{2}k^2 + \frac{1}{6}k) = \\ = \frac{4}{3}n^2 m + \frac{4}{3}nm^2 + O(n^2 + nm)\end{aligned} \tag{10}$$

2. Умножение строки на обратную матрицу:

$$\begin{aligned}(2m^3 - m^2) \times \sum_{i=1}^{k-1} i = (2m^3 - m^2) \times (\frac{k^2}{2} - \frac{k}{2}) = \\ = n^2 m - nm^2 + O(n^2 + nm)\end{aligned} \tag{11}$$

3. Мультипликативные операции при занулении столбцов:

$$\begin{aligned}(2m^3 - m^2) \times \sum_{i=1}^{k-1} i^2 = (2m^3 - m^2) \times (\frac{k^3}{3} - \frac{k^2}{6} - \frac{k}{6}) = \\ = \frac{2}{3}n^3 - \frac{1}{3}n^2 m - \frac{1}{3}nm^2 + O(n^2 + nm)\end{aligned} \tag{12}$$

4. Сложность обратного хода Гаусса: Мультипликативных операций:

$$m^2 \times \sum_{i=1}^{k-1} i = \tag{13}$$

5. Аддитивных аналогично вышенаписанному, тогда итоговая сложность всего алгоритма будет:

$$S(n,m) = \frac{2}{3}n^3 + 2n^2 m + O(n^2 + m^2) \tag{14}$$

6 Сравнение сложности

$$S(n,1) = \frac{2}{3}n^3 + O(n^2) \tag{15}$$

$$S(n,n) = \frac{8}{3}n^3 + O(n^2) \tag{16}$$