Каноническая аппроксимация тензоров и ее реализация на Python

Кузнецов М.А.

1 Введение

Аппроксимация многомерных массивов играет важную роль в приложениях. Однако вместо заданного многомерного массива часто нужно пользоваться его приближением, свойства которого известны, возможно. в отличие от заданного. Такие аппроксимации удобно строить используя следующее представление многомерного массива (*тензора*)

Определение

Тензором A размерности d назовем многомерный массив, элементы которого $\mathrm{A}(\mathrm{i}_1,\mathrm{i}_2,\ldots,\mathrm{i}_d)$ имеют d индексов.

Определение

Каноническим разложением многомерного массива (*тензора*) называется представление вида

$$A(i_1, i_2, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \ldots U_d(i_d, \alpha), \tag{1}$$

где U_k называются *факторами* канонического разложения, а r — каноническим рангом.

Уравнение (1) является основным.

2 Численные эксперименты

В данном параграфе будут изложены в графическом виде результаты работы программы, реализующей метод ALS. В качестве входных данных подавались:

- Размерность тензора $d = 3$
- Ранг r переменный
- Размерности мод $dimension_i$ переменные

2.1 Численные эксперименты для случайных тензоров

В качестве входного тензора подается тензор, случайным образом полученный программно (с помощью процедуры gettensor) наперед заданного ранга и размерностей мод.

Первый цикл экспериментов призван был установить характер поведения нормы невязки

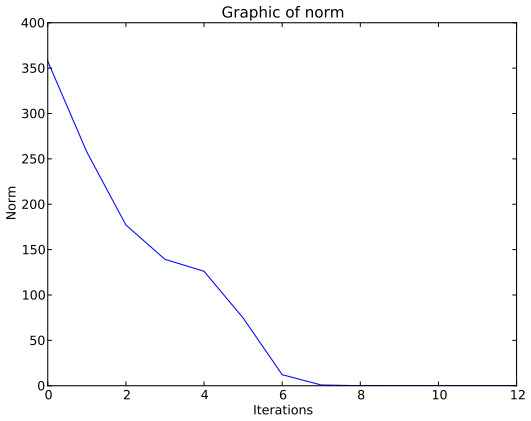
$$max|A(i_1, i_2, i_3) - Approximation(i_1, i_2, i_3)| \tag{2}$$

где Approximation(i_1,i_2,i_3) — аппроксимация заданного тензора, построенная с помощью алгоритма ALS, реализованного на Python.

Ниже приводятся графики поведения нормы невязки (2) в зависимости от числа итераций.

- Для случайного тензора ранга $r = 5$

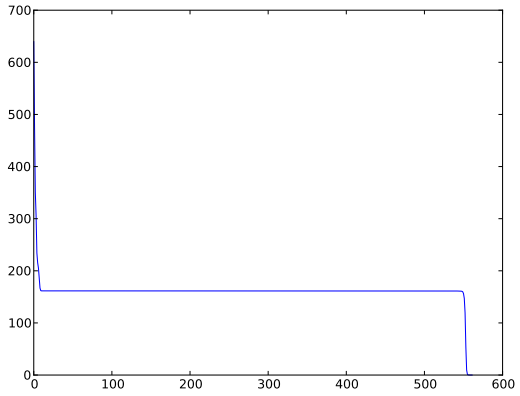
[fontsize=,linenos]python from test import * from numpy import * from pylab import * d=3 dimension=[32,32,32] r=5 a,u0=randomtensor(r,dimension,size(dimension)) eps=1e-6 a1,u,no=ALSproc(a,d,r,dimension,eps) plot(no) xlabel('Iterations') ylabel('Norm') title('Graphic of norm') savefig('rnd5.pdf')



- Для случайного тензора ранга $r = 10$

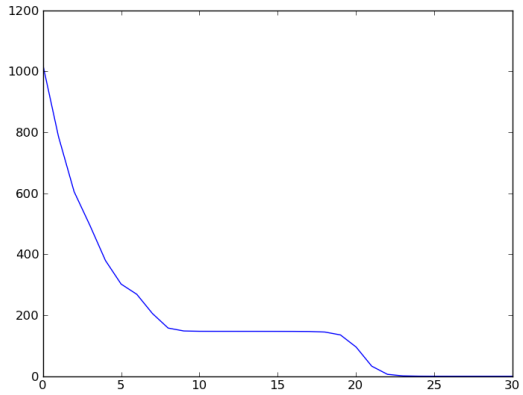
На этом примере метод попал в локальный минимум функционала (2), вследствии чего невязка убывает медленно почти на всем протяжении времени работы алгоритма. Однако миновав локальный минимум, метод сошелся очень быстро.

[fontsize=,linenos]python from test import * from numpy import * from pylab import * d=3 dimension=[32,32,32] r=10 a,u0=randomtensor(r,dimension,size(dimension)) eps=1e-6 a1,u,no=ALSproc(a,d,r,dimension,eps) plot(no) xlabel('Iterations') ylabel('Norm') title('Graphic of norm') show()



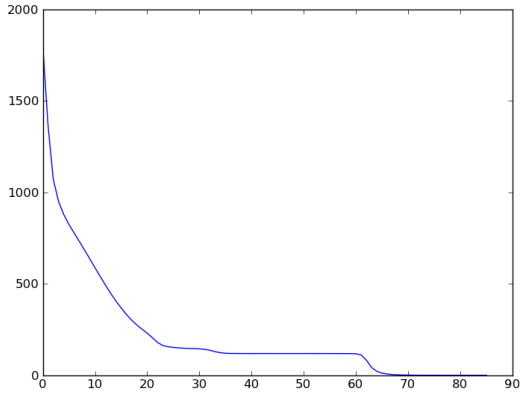
- Для случайного тензора ранга $r = 25$

[fontsize=,linenos]python from test import * from numpy import * from pylab import * d=3 dimension=[32,32,32] r=25 a,u0=randomtensor(r,dimension,size(dimension)) eps=1e-6 a1,u,no=ALSproc(a,d,r,dimension,eps) plot(no) xlabel('Iterations') ylabel('Norm') title('Graphic of norm') show()



- Для случайного тензора ранга $r = 100$

[fontsize=,linenos]python from test import * from numpy import * from pylab import * d=3 dimension=[32,32,32] r=100 a,u0=randomtensor(r,dimension,size(dimension)) eps=1e-6 a1,u,no=ALSproc(a,d,r,dimension,eps) plot(no) xlabel('Iterations') ylabel('Norm') title('Graphic of norm') show()

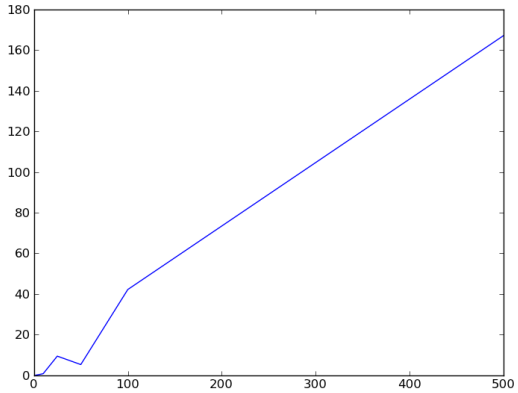


Несмотря на то, что скорость убывания невязки может варьироваться в зависимости от ранга и начального приближения, невязка убывает монотонно.

Следующая серия экспериментов показывает нрафическую зависимость времени выполнения программы от:

- ранга r при фиксированных размерностях тензора

[fontsize=,linenos]python from test import * from numpy import * from pylab import * from time import * d=3 dimension=[32,32,32] r=[2,3,5,10,20,50,100] mar=zeros((2,7)) for i in xrange(0,7): t=time() a,u0=randomtensor(r[i],dimension,size(dimension)) eps=1e-6 a1,u,no=ALSproc(a,d,r[i],dimension,eps) mar[0,i]=time()-t mar[1,i]=r[i] plot(mar[1],mar[2]) xlabel('rank') ylabel('time') title('Graphic of time') show()



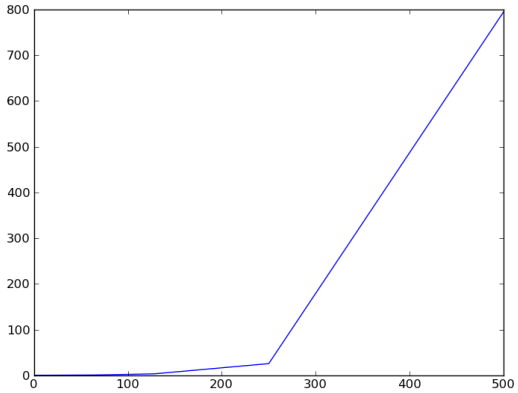
в ходе этого эксперимента размерности мод $dimension_i$ брались равными между собой и равными 32 а ранг менялся r = 2,3,5,10,25,50,100,500. Исходя из графика, можно сделать вывод, что время зависит от ранга как O(r)

- размерностей тензора $dimension_i$ (i = 1,. . . ,3) при фиксированном ранге Эта

[1]DEFINITION NOT FOUND: 1
[2]DEFINITION NOT FOUND: 0

серия экспериментов проводилась с целью изучения зависимости времени выполнения программы от размерностей мод

$dimension_i$ = 32,64,128,250,500 и ранге $r = 5$.



Судя по графику время выполнения программы пропорционально $O(n^{1,2})$