

问题描述

这次应该不会自动编译了吧。。。。

我们要实现下面两个算法之一。

LinearTimeSVD 算法

线性时间 SVD 算法的策略是：选取矩阵 A 的 c 列，每列重新标注，形成一个新矩阵 $C \in \mathbb{R}_{m \times c}$ 。然后通过对 $C^T C$ 进行通常的 SVD 分解，计算 C 的奇异值和左奇异向量。这可以近似 A 的奇异值和左奇异向量。

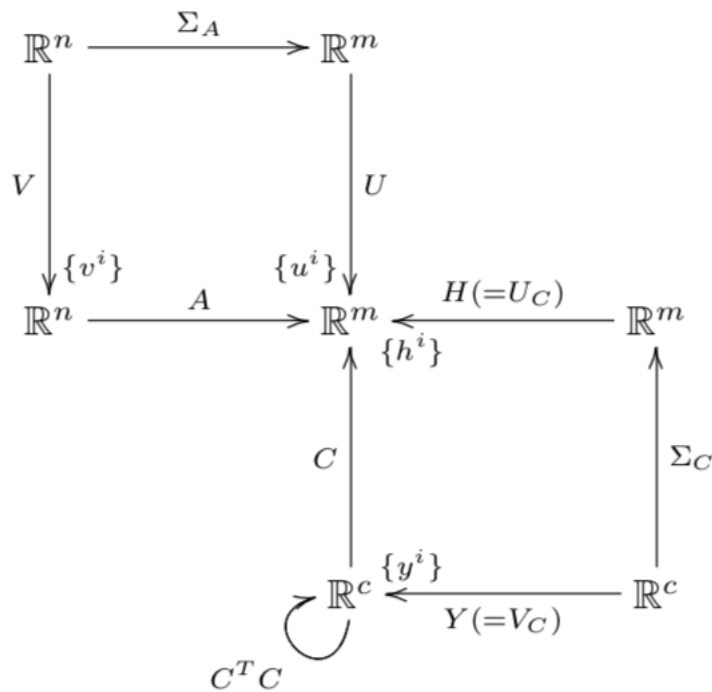


图 1: LinearTimeSVD 算法

图片1是该算法的示意图。

Proto-Algorithm: Solving the Fixed-Rank Problem

第二种算法的思路如下图2所示：

PROTO-ALGORITHM: SOLVING THE FIXED-RANK PROBLEM

Given an $m \times n$ matrix \mathbf{A} , a target rank k , and an oversampling parameter p , this procedure computes an $m \times (k + p)$ matrix \mathbf{Q} whose columns are orthonormal and whose range approximates the range of \mathbf{A} .

- 1 Draw a random $n \times (k + p)$ test matrix $\mathbf{\Omega}$.
- 2 Form the matrix product $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.
- 3 Construct a matrix \mathbf{Q} whose columns form an orthonormal basis for the range of \mathbf{Y} .

图 2: Proto-Algorithm: Solving the Fixed-Rank Problem

我在这次作业中，选择了第一种算法，用 python 语言对其进行实现。

实验

随机生成数据

通过下面的设置取得一个矩阵，并求出其前 $r \in \{5, 10, 15, 20\}$ 大的奇异值，以及其奇异向量。

```
m = 2048
n = 512
p = 20
A = randn(m, p) * randn(p, n)
```

得到的结果如下：

r	exp1: t	exp2: t	exp3: t
5	0.08126497268676758	0.06409597396850586	0.07390093803405762
10	0.03443193435668945	0.025016069412231445	0.03003096580505371
15	0.03320908546447754	0.035800933837890625	0.03665590286254883
20	0.034197092056274414	0.03660106658935547	0.040490150451660156

我对 $r = \{5, 10, 15, 20\}$ 进行了 3 组实验，得到的运行时间如上表所示。可见运行时间较为稳定，可以在线性时间内运行完毕。

rSVD-single-pass 数据集

这个部分里，我复现了<https://github.com/WenjianYu/rSVD-single-pass>中生成随机数据集的算法，写在 *generatedata.py* 文件中。并测试了对于 $m = n = 2 \times 10^3$ 的矩阵的计算时间的三次试验结果¹。

r	exp1-timing	exp2-timing	exp3-timing
50	0.4229423999786377	0.4193081855773926	0.4090569019317627
100	0.4109618663787842	0.5056757926940918	0.4087069034576416
150	0.451876163482666	0.4589710235595703	0.43385791778564453
200	0.5109848976135254	0.5014669895172119	0.4837648868560791
500	0.994359016418457	0.9467556476593018	0.9239499568939209
1000	2.2344188690185547	2.214768886566162	2.300198793411255
1500	4.6173412799835205	4.693850994110107	4.715956926345825

可见，运行时间比较稳定，算法是有效的。

¹我尝试生成了 2×10^5 阶矩阵的数据集，但是内存超了，所以没有呈现在报告上。特此说明。