Online Applications of HodgeRank Problems

Taoli Shen Advisor: Xiaozhe Hu

Abstract

Background
Ranking problems are very practical and ubiquitous in our daily life. The idea of rankings or ratings exists in every football championship season or every customized Netflix movie recommendation.
In mathematics, solving ranking problems is a direct application of least squares regressions on graphs. This idea is best summarized and innovated by Jiang et al, who created a systematic approach to solve statistical ranking problems called HodgeRank.
Historical Methods
The main idea is the application of Hodge decomposition, which gives residue that contains information on the ranking of the subjects. Researchers have been active in this field to explore algorithms to solve the HodgeRank problems numerically and more efficiently. In a recent paper of Colley et al, they show the effectiveness of unsmoothed aggregation algebraic multigrid (UA-AMG) when it comes to solving graph-based least squares. Other iterative methods such as Successive Subspace Correction (SSC) also demonstrate good efficiency when solving least squares problems.
New Algorithm and Comparison
In this paper, we develop an algorithm that aims to quickly update the subjects' ranking online. This algorithm is based on UA-AMG with a faster convergence rate. In reality, one may find its application in online agencies such as movie ratings. At the end of the paper, we provide experiments testing against other iterative methods to demonstrate the fast convergence rate of our algorithm when solving large-graph-based least squares problems.

Key Concept

- HodgeRank Theory: The most important mathematical foundation of my research is the HodgeRank theory
- $$\min_r ||B^T r - f||_W^2 \Rightarrow BWB^T r = BWf$$

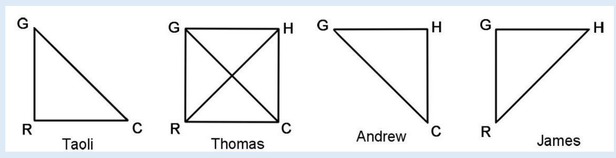
  Where B^T is the incidence matrix, f is the edge flow, W is the weight matrix, and r is our desired ranking.

Movie Ranking Simulation

Suppose we have 4 movies screening in Somerville Theatre, Get Out, Roma, COCO, and Hamilton. My goal is to figure out which one is the best according to popularity and ratings.
Now my housemates and I went to those and rated each on a 1~10 scale, but not everyone watched all the movies.

Movie/voters	Taoli	Thomas	Andrew	James
Get out (G)	8	5	5	7
Roma (R)	10	4		7
Coco (C)	6	9	10	
Hamilton (H)		6	5	10

In order to figure out the ranking, we need to transfer the information onto a graph

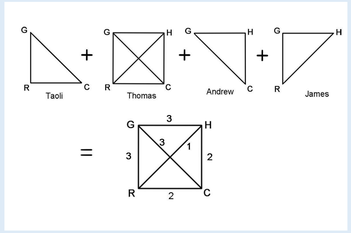


$$G = (V, E, w)$$

- ***V***: a set of vertices that symbolizes the objects being rated
- ***E***: a set of edges connecting the vertices
- ***w***: a set of weights that tells the "importance" of each edge

Movie Ranking Simulation

By assembling all the subgraphs together, we acquire the weights of each edge



$$W = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} CG \\ CH \\ CR \\ GH \\ GR \\ HR \end{matrix}$$

Edge flow f can be understood as the pairwise comparison between the average rating of each movie.
It is defined as

$$f(i,j) = \frac{\sum_{v \in voters} R(v,i) - R(v,j)}{|voters|}$$

where $R(v,i)$ is voter v's rating of movie i.

Now assemble everything according to the HodgeRank theory, $\min_r ||B^T r - f||_W^2 \Rightarrow BWB^T r = BWf$

We have

$$\begin{bmatrix} -1 & -1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \boldsymbol{r} = \begin{bmatrix} -1 & -1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{7}{3} \\ -4 \\ \frac{1}{2} \\ -\frac{4}{3} \\ -\frac{1}{3} \\ \frac{5}{2} \end{bmatrix}$$

And using an iterative method called AMG, we get that

$$r = [1.2379 \quad -1.2517 \quad 0.5517 \quad -0.5379]^T \quad \begin{matrix} \mathbf{C} & \mathbf{G} & \mathbf{H} & \mathbf{R} \end{matrix}$$

Which means that Coco ranks 1st, Hamilton ranks 2nd, Roma 3rd, and Get Out comes last

Online Ranking Update

Question: Now we know how to calculate the ranking given a fixed set of data, but in real world the data is constantly changing, so how can we efficiently update the rankings in a timely manner?

Method 1: <u>do the above step again from scratch</u>
However, we know that the more data needs processing, the slower it would become. It would be easy to recalculate the ranking of only 4 movies, but it would be incredibly slow if there were 40000 movies with millions of ratings.

Method 2: <u>find a formula that explains the relationship between the old ranking $r^{(k)}$ and the new ranking $r^{(k+1)}$</u>
suppose we have 40000 movies in the database, but there would probably only be 20 ratings updated in the next hour. Therefore, we only need to focus on the change of a few edges in the graph G. The computational complexity would therefore be largely reduced.

Suppose $r^{(k)}$, W, B^T, and f are known, and we received a new rating of a movie. Then we have the change below, and the goal is to calculate the new ranking $r^{(k+1)}$

Before update	$r^{(k)}$	B^T	W	f
After update	$r^{(k+1)}$	B^T	$\hat{W}$	$\hat{f}$

Notice $\hat{W} = \begin{bmatrix} W_1 \\ \hat{W}_2 \end{bmatrix}$, where W_1corresponds to the weights that remain the same, and $\hat{W}_2$corresponds to the changes. The same goes for $\hat{f} = \begin{bmatrix} f_1 \\ \hat{f}_2 \end{bmatrix}$, and $B^T = \begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix}$.

With some algebra, we found that
Before update

$$\min_{r^{(k)}} ||B^T r^{(k)} - f||_W^2 \iff \underbrace{[B_1 W_1 B_1^T + B_2 W_2 B_2^T] r^{(k)}}_{L^{(k)}} = \underbrace{B_1 W_1 f_1 + B_2 W_2 f_2}_{b^{(k)}} \quad *$$

After update

$$\min_{r^{(k+1)}} ||B^T r^{(k+1)} - \hat{f}||_{\hat{W}}^2 \iff \underbrace{[B_1 W_1 B_1^T + B_2 \hat{W}_2 B_2^T] r^{(k+1)}}_{L^{(k+1)}} = \underbrace{B_1 W_1 f_1 + B_2 \hat{W}_2 \hat{f}_2}_{b^{(k+1)}} \quad **$$

Where **L** is the famous graph laplace matrix in mathematics, and **b** is a vector.
Using $*$, $**$, and the fact that $\hat{W}_2 = W_2 + \delta W_2$, and $\hat{f}_2 = f_2 + \delta f_2$, we found the following formula with some algebra

$$r^{(k+1)} = (L^{(k)} + \delta W_2 B_2 B_2^T)^\dagger (b^{(k)} + B_2(w_2 \delta f_2 + \delta W_2 f_2 + \delta W_2 \delta f_2))$$

Where $\dagger$ is the Moore–Penrose inverse (pseudoinverse)

Online Ranking Update

To expand the pseudoinverse, we consulted *Generalized Inversion of Modified Matrices* by Carl Meyer. And here's the final formula that connects $r^{(k)}$and $r^{(k+1)}$

$$r^{(k+1)} = r^{(k)} + (w_2 \delta f_2 + \delta W_2 f_2 + \delta W_2 \delta f_2) L^{(k)\dagger} B_2 - \frac{\delta W_2 (L^{(k)\dagger} B_2 B_2^T L^{(k)\dagger} b^{(k)})}{1 + \delta W_2 B_2^T L^{(k)\dagger} B_2} - \frac{W_2 \delta W_2 \delta f_2 + \delta W_2^2 \delta f_2 (L^{(k)\dagger} B_2 B_2^T L^{(k)\dagger} B_2)}{1 + \delta w_2 B_2^T L^{(k)\dagger} B_2}$$

However, notice we have quite a few $L^{(k)\dagger} B_2$ in this formula. This is may be problematic in that this term still makes the computational complexity high due to the fact that inverting a matrix is very expensive. Therefore, it is imperative to find an approximation to $L^{(k)\dagger} B_2$

Luckily, the vector B_2 has a very special structure. It has only two nonzero element 1 & -1, i.e.

$$B_2 = [0 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 0 \quad -1 \quad 0 \quad \dots \quad 0]$$

This property makes the product $L^{(k)\dagger} B_2$ very sparse (have many 0's), hence faster to compute.

Currently, we are working on using the idea of effective resistance to find a local approximation (at nonzeros points) of $L^{(k)\dagger} B_2$. Once this step is done, we are very close to completion.

Future Steps

For the next few weeks, we will be working on approximating the term $L^{(k)\dagger} B_2$and test our formula in MATLAB.

For the rest of the summer and this coming fall, we will be working on writing the algorithm based on our theoretical foundation and test its performance with real world dataset.

Reference

Jiang, Xiaoye, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. "Statistical ranking and combinatorial Hodge theory." Mathematical Programming 127, no. 1 (2011): 203-244.

Colley, Charles, Junyuan Lin, Xiaozhe Hu, and Shuchin Aeron. "Algebraic multigrid for least squares problems on graphs with applications to hodgerank." In 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 627-636. IEEE, 2017.

Meyer, Carl D. "Generalized Inversion of Modified Matrices." SIAM Journal on Applied Mathematics 24, no. 3 (1973): 315-23.