How to Invest in Movies

Deng Zhantao, Huang Yu-Ting, Xia Shengzhao, Zhang Yinan
Department of Electrical and Electronic Engineering, EPFL, Lausanne, Switzerland

***Abstract*—In this project, we created a graph model from TMDb dataset to predict the success of a new movie. After filtering by heat kernel, the diffused signals could demonstrate the distance between nodes in the graph. We split the graph nodes into training and validation set, and used this characteristic of the heat kernel to perform regression. In order to do that, we defined a loss function and used gradient descent to implement the optimization. In the end, we analyzed the importance of different features to the success of some specific genres of movies and hoped to predict the return on investment(ROI) of a new movie.**

I. Motivations

Movie industry is an up-and-coming hot investment area in the new century. For an investor, whether a movie will succeed is a key point to decide the investment. The success of a movie could be impacted by lots of factors, such as theme, cast, production company. However, it seems that the successful factors of different genres of movies would be different. For a specific genre, which factor would influence it most? This project would mainly focus on analyzing TMDb data, answering this problem and, hopefully, help investors to do a wise investment.

II. Problem formulation

In the film industry, there were several ways to define "success", such as return on investment(ROI), film awards, popularity and ratings. In this project, we regarded ROI as the success, which were generally decided by the public. We dropped the film awards because it involved technical decision from professionals, and the relevant information was not provided in the dataset. We attempted to analyze the influence of movie features to the success, and we tried to predict ROI of a new movie.

III. Dataset exploration and acquisition

The Dataset were collected from TMDB, which contains around 4800 movies. Data points include cast, crew, keywords, budget, revenue, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

Based on our problem formulation, we reserved actor, director, budget, keywords, production companies as our features, which were prior knowledge of a movie. In addition, vote averages, popularity and revenue were defined as the output, which were posterior information of a movie, and also what we wanted to predict.

In the following, we discussed the data exploration and acquisition process.

A. Actor

Actors are essential aspect of movies and most interested us. In this part, we decided to take a deep view into the actor data. From Kaggle TMDb dataset, we obtained 54201 distinct actors. Here, we constructed actor social network, visualized and explored it via networkx package.

1) Filtering: First, among these distinct 54201 actors, there were actually a great number actors who only appeared in a few movies, we regarded them as not important actors and dumped them for simplicity. After filtering actors appearing less than 5 movies, 3794 actors were left and we plotted the distribution of the number of actors' movies as figure 1. We can see that most actors stared in 5 to 10 movies (actually 70.45%) and very few actors appeared in more than 40 movies.

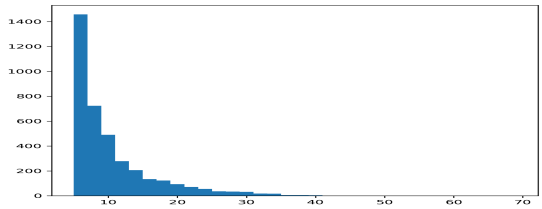


Figure 1: Histogram of actor filtered

2) Sub-sampling: After that, we created adjacency matrix according to co-appearance, i.e. we planed to weigh edges in actor social network according to the number of movies they appeared together. However, 3794 nodes/actors were still too massive for visualization and analysis, we sub-sampled 400 nodes from them according the distribution above, which, hopefully, could still preserve the generality of data and easier to utilize. After sub-sampling, we constructed a 400×400 adjacency matrix with 0.01 sparsity. We showed the sparsity graphs as figure 2, where the left was before sub-sampling and the right was sub-sampled. It can be seen that the sparsity of adjacency apparently descended.

Then, we can easily create a network according to the sub-sampled adjacency matrix. However, we found that the network was not fully connected and contained 10 components. We picked the largest component for following procedures, which accounted for 97.25% of all nodes.

3) Spectral clustering: We, then, applied spectral clustering method to our graph to see whether the actor social network would form clusters. First, we visualized the spectrum of the Laplacian by plotting the eigenvalues and we could

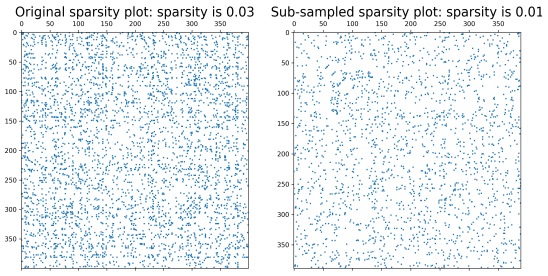


Figure 2: Actor Sparsity Graphs

see that the largest gap appeared in the Laplacian spectrum after the second eigenvalue, which means the data might have 2 clusters. Secondly, We used Laplacian eigenmaps method to embed our graph in a 6-dimensional Euclidean space (pick first 6 eigenvectors of Normalized Laplacian) and run k-means in this $\mathbb{R}^6$ embedding space to do a binary clustering. We visualized the k-mean result in 2-dimensions embedding space and the outcome can be seen as figure 3, where the left was spectrum of the Laplacian and the right was k-mean results visualized on 2-dimensions.

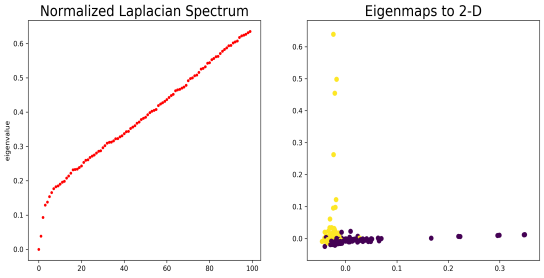


Figure 3: Spectral clustering

4) Visualizing network: We visualized the largest component of social network via networkx and labeled the nodes according to the results of K-means as figure 4.

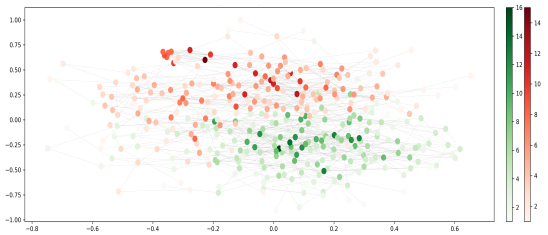


Figure 4: Actor Social Networks

From actor social networks and Eigenmaps, we found that actors can be well partitioned into 2 clusters. We can find 5 most 'sociable' actors of each cluster as figure 5, Julian Glover and Vera Farmiga are most 'sociable' actors of each cluster. Above all, we can see that actors do form communities. There might be some latent factors resulting these communities (clusters), however, it was hard to explain it concretely.

	Name	Degree
128	Julian Glover	15
111	Jason Ritter	12
44	Spencer Wilding	12
87	David Kelly	11
15	Elwin 'Chopper' David	11

(a) cluster 0

	Name	Degree
108	Vera Farmiga	16
78	Kevin Corrigan	14
41	Adam LeFevre	13
144	Marcia Gay Harden	13
190	Carlos Alazraqui	13

(b) cluster 1

Figure 5: 5 most 'sociable' actors

B. Budget

The budget value ranged from 0 to 3E+8. There were some missing value in the budget, denoted as 0 in the dataset. Besides, there were wrong values. For instance, the budget of movie "Escape from Tomorrow" (id=27) was recorded as 650, while the groundtruth was 6.5E+5 as we checked on the internet. We assumed that budget less than 1000 were wrong value, and we treated nodes with missing and wrong value as isolated nodes. Namely, they were not connected to any other node in the budget subgraph.

C. Keyword

1) Word2Vec: The word2vec takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words, which can depict semantic meaning of word compared to Bag-of-Words method. In order to train a good word2vec model, we scraped much more movie data (217723 movies) via TMDb's API and used keywords to do it. A simple way to investigate the learned representations is to find the closest words for a specified word. To test the performance of our model, we show an example of a given word space:

```
('space travel', 0.999713659286499),
('alien invasion', 0.9996951222419739),
('astronaut', 0.999618411064148),
('spacecraft', 0.9995511174201965),
```

Figure 6: Similar words of 'space'

2) Bag-of-Words: More than 9000 keywords were used to describe the movies. If applying the bag-of-words model directly and building the adjacency matrix, it could not represent associations between movies that had keywords of similar meanings. The cleaning process was as follows based on the method mention in [1]. First, group keywords that had the same root. Then replaced keywords by synonyms of higher occurrence frequency if necessary. Lastly, delete those keywords that appeared in no more than four movies. After cleaning, roughly 2000 keywords were kept. Finally, we utilize this method since it has a better result.

D. Company

Production companies might determine the quality of a movie. Figure7 shows several highly productive companies.

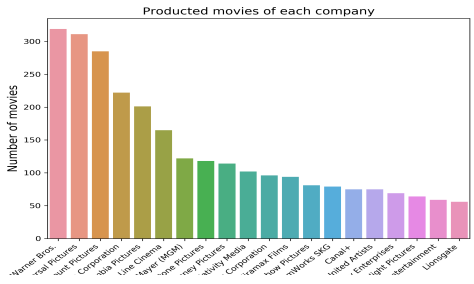


Figure 7: The frequency distribution map of movies producted by different companies

IV. Graph Model

We would construct N adjacency matrices after data exploration and acquisition, denoted as A^i. Our final graph A was a weighted sum of N adjacency matrices; namely, $A = \sum_{i=1}^{N} v_i A^i$, and v_i was the weight vector. Then we constructed the normalized Laplacian via $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D was the degree matrix.

In the following, we discussed how we constructed the adjacency matrices.

A. Actor, Director and Keyword

We only kept four actors for each movie if the total number of cast was larger than four. To avoid possible mistakes later, special characters in the cast names were replaced by '_'. For instance, if a name contained ', The library function CountVectorizer() will treat it as two names separated by '.

Sometimes, the same actor played different roles in a movie. If the feature vector for a movie contained 2, it meant that the same actor was counted twice. In this case, we should make sure that each feature vector generated by CountVectorizer() only contained 0 or 1 when using the bag-of-words model.

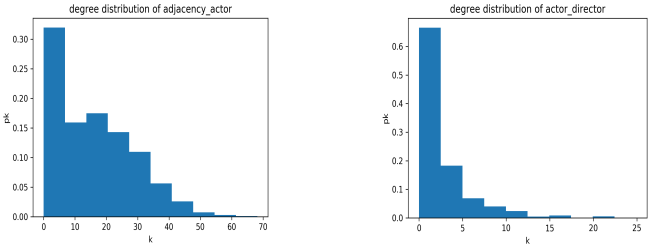
The former two directors of each movie were kept if the number was greater than two. And then everything was just like how we built the actor network.

Figure8 shows that the degree distribution of the actor network and the director network.

After cleaning keywords, we exploited the same bag-of-words model to extract feature vectors and built the keyword network in terms of similarities.

B. Budget

In budget subgraph, we used Euclidean distance to define similarity between nodes, and gaussian function to turn distances into weight.



(a) actor network (b) director network

Figure 8: degree distribution

C. Production company

We extracted production companies for each movie and denoted them as subsets. The union of all these subsets served as the bag of companies. In this way, we could generate feature vectors which only contain 0 and 1 for each movie.

V. Proposed Solution

We assumed that the signal on the graph was smooth so as to do the prediction. That is to say, if two movies (nodes) resembled each other, they should produce similar output. For instance, if two movies shared the same big movie star actor, they were expected to have similar popularity or revenue.

In order to know which feature was the most important factor to the success, we attempted to solve the weight vector v_i. Namely, the bigger the weight, the more it would contribute to the final graph, implying the importance of that subgraph.

We separated the signal into known and validation set, diffused the known signal on the graph and checked how signal changed at validation set via diffusion.

For instance, signal at node A was known and signals at node B, C were for validation; A was close to B while far from C. After placing a dirac function on node A and diffused via heat kernel, B would receive higher value and C would receive lower value. We compared the received signal at B and C with its groundtruth. The loss function was defined as the L2-norm of difference between groundtruth and signal after diffusion, which would be explained in details in the next section.

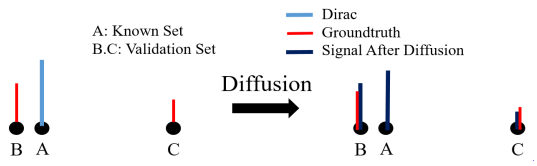


Figure 9: Diffusion Graphs

Heat kernel was expressed as e^{-tL}, which could also be

approximated as Maclaurin series[2].

$$H_t = e^{-tL} \approx I - tL + \frac{1}{2}t^2L^2$$

$$L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

$$= I - (\sum_{i=1}^{N} v_i D_i)^{-\frac{1}{2}} (\sum_{i=1}^{N} v_i A_i)(\sum_{i=1}^{N} v_i D_i)^{-\frac{1}{2}}$$

In the equation, we could see how weight vector v_i and diffusion parameter t were related to heat kernel. We decided the optimal vector v_i and t via gradient descent.

VI. Loss function, Gradient and Its implementation

A. Loss function and gradient

In the beginning, we clarified some notations. S: signal, Y: groudtruth, H_t: heat kernel, N: the number of valid entries in groundtruth, f, mask to choose the known and validation indices, A: the weighted adjacency matrix, A^i: the adjacency matrix of the ith subgraph, D: the degree matrix of A, L: the normalized Laplacian of A, λ: the regularization parameter, v_k: the kth entry of weight vector v.

To find the optimal v with a scale constraint $\|v\|_1 \leq 1$, firstly we needed to define a proper loss function to calculate the error between predictions and groundtruth signals. Since ridge regression had very good properties in computation and LASSO (least absolute shrinkage and selection operator) produced sparse solutions; we implemented both of them. Here we used ridge regression as an example to compute the gradient.

The loss function was defined as:

$$Loss = \frac{1}{2N}\|f \odot (H_t \cdot S - Y)\|_2^2 + \frac{\lambda}{2}\|v\|_2^2$$
$$= \frac{1}{2N}\sum_i f_i \left[\sum_j H_{ij}S_j - Y_i\right]^2 + \frac{\lambda}{2}\sum_k v_k^2$$

where $\odot$ represented the Hadamard production. As shown in the last section, H_t was a function of the normalized Laplacian L, which was again a function of weighted adjacency matrix A, which was a function of v_k. Thus, according to the chain rule, we could write the gradient as

$$\frac{\partial Loss}{\partial v_k} = \sum_{i,j}\frac{\partial Loss}{\partial H_{ij}}\frac{\partial H_{ij}}{\partial v_k}$$
$$= \sum_{i,j}\frac{\partial Loss}{\partial H_{ij}}\sum_{k,l}\frac{\partial H_{ij}}{\partial L_{kl}}\frac{\partial L_{kl}}{\partial v_k}$$
$$= \sum_{i,j}\frac{\partial Loss}{\partial H_{ij}}\sum_{k,l}\frac{\partial H_{ij}}{\partial L_{kl}}\sum_{m,n}\frac{\partial L_{kl}}{\partial W_{mn}}\frac{\partial W_{mn}}{\partial v_k}$$

Note that we ignored regularization term in the derivation, since it was trivial and one could add it in the end. We expressed each term separately in the following:

$$\frac{\partial W_{mn}}{\partial v_k} = W^k_{m,n}$$

$$\frac{\partial L_{kl}}{\partial W_{mn}} = \begin{cases} (m,n) = (k,l), \\ \frac{1}{2}D_{kk}^{-\frac{3}{2}}W_{kl}D_{ll}^{-\frac{1}{2}} - D_{kk}^{-\frac{1}{2}}D_{ll}^{-\frac{1}{2}} + \frac{1}{2}D_{kk}^{-\frac{1}{2}}W_{kl}D_{ll}^{-\frac{3}{2}} \\ m = k, n \neq l, \\ \frac{1}{2}D_{kk}^{-\frac{3}{2}}W_{kl}D_{ll}^{-\frac{1}{2}} \\ m \neq k, n = l, \\ \frac{1}{2}D_{kk}^{-\frac{1}{2}}W_{kl}D_{ll}^{-\frac{3}{2}} \end{cases}$$

$$\frac{\partial H_{ij}}{\partial L_{kl}} = \begin{cases} (k,l) = (i,j), & -t + \frac{t^2}{2}(L_{ii} + L_{jj}) \\ k = i, l \neq j, & \frac{1}{2}t^2L_{lj} \\ k \neq i, l = j, & \frac{1}{2}t^2L_{ik} \end{cases}$$

$$\frac{\partial Loss}{\partial H_{ij}} = \frac{1}{N}[(H_iS - Y_i)f_i]S_j$$

Similarly, the derivative with respect to t could be written as:

$$\frac{\partial Loss}{\partial t} = \frac{1}{N}\sum_{i,j}\left[(H_tS - Y) \odot f \cdot S^T \odot (-L + tL^2)\right]_{ij}$$

where summation was the sum of all entries in the matrix.

B. Implementation

We used gradient descent as the optimizer. The loss function was a nonconvex, so we adopted adaptive step size to alleviate this problem.

The parameter was the following:

	MaxIter	Step_t	Step_{V_k}	Lambda	Gain
Lasso	1000	$\frac{(5e-8)}{(iters+1)^{0.75}}$	$\frac{0.2}{0.75(iters+1)}$	0.01	5
Ridge	1000	$\frac{(5e-8)}{(iters+1)^{0.75}}$	$\frac{0.05}{0.75(iters+1)}$	0.01	5

Table I: Ridge and Lasso regression parameter

VII. Result

Movies had different kinds of genres. The underlying success factors might differ in different types of movies. Thus, movies of a certain type were to be extracted before training parameters. For example, if we were interested in "Romantic" movies, we selected all the movies that contained 'Romance' in the column 'genres' from the TMDb dataset.

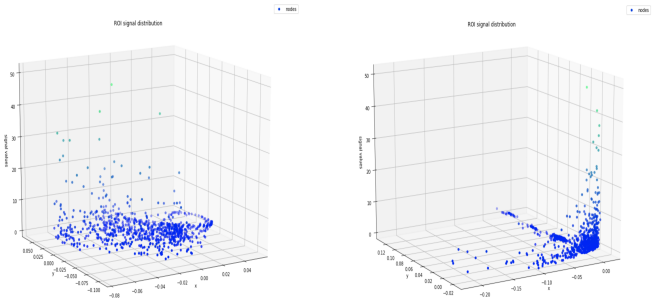
A. *Analysis on the importance of features*

First of all, we set the initial value of each component of weight vector to be equal, i.e. $v_i = v_j$, and plotted the signal on this initial adjacency; namely, the initial adjacency was just a simple average of the subgraphs. Then, we found the optimal v via gradient descent and plotted the signal on the final adjacency matrix. In Figure 10, we could observe that the diffused signals scattered at first; however, they gathered after we tuned parameters. This demonstrated that similar signals (ROI) were gathered geometrically after we found the optimal weight vector, which meant our graph become smoother[3]. This was the key for us to do prediction.

# Genre	Comedy	Drama	Romance
Quantity	1722	2297	894
Success Factor1	Budget	Budget	Budget
Success Factor2	Keyword	Company	Company

Table II: Comparison of success factors of different genres

In Figure11, we attached the optimal vector of comedy, drama and romance. We could observe that budget was the most successful factor in three genres. During the process of film making, the company would first decide the story, the cast, then evaluated the budget. We interpreted that budget was actually a latent variable; in other words, budget would be decided after actors and the story, and it was common knowledge that if one wished to earn more, one should spend more. For comedy, the second most successful factor was keywords, which was reasonable since when people would be more likely to be attracted by the theme (keywords) of a comedy than other factors such as actors or crews. For drama and romance movies, people obviously would focus much on its plot and an experienced production company was actually the firm guarantee to it. And it explained why production company was the second successful factor of drama and romance movies.



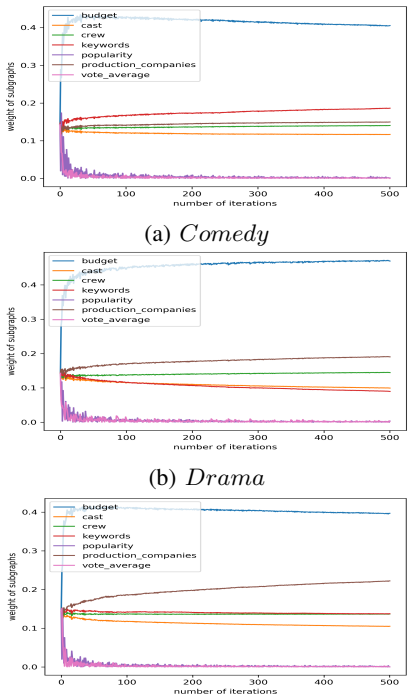
(a) Before parameter optimization

(b) After parameter optimization

Figure 10: The ROI signal plotted on the graph before and after parameter optimization

B. *ROI Prediction*

Here we clarified how to make predictions about a new movie based on similarity measurement. When a new node came in, we could calculate its distance to each movie node in every subgraph. Next, a weighted combination of distance vectors in each subgraph was taken to obtain a new distance vector. The mean ROI of k nearest neighbors of this new movie could serve as the prediction. However, this method could not predict black horse in box office, i.e. we predicted the ROI of La La Land to be 0.5, but it achieved 18.36 ROI in 2016, which was much higher than most of romance moives in the IMDb dataset. Thus, it is unlikely to pridict its ROI accurately with the limited dataset. Therefore, we would like to explore a better method in the future works.



(a) *Comedy*

(b) *Drama*

(c) *Romance*

Figure 11: The variation of weight vector along the number of iterations

VIII. Summary

In conclusion, we could analyze successful factors to movies of different genres in our graph. Furthermore, our graph could potentially be used to predict ROI of a new movie, but further work needed to be done to improve accuracy.

References

[1] F. Daniel, "Film rrecommendation engine," https://www.kaggle.com/fabiendaniel/film-recommendation-engine.

[2] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, no. 50, pp. 19 735–19 740, 2007.

[3] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.