Explore Co-clustering on Job Applications

Qingyun Wan
SUNet ID:qywan

1 Introduction

In the job marketplace, the supply side represents the job postings posted by job posters and the demand side presents job seekers who would like to apply for suitable jobs. In the platforms of job marketplace, like Linkedin, Indeed and etc., it is crucial for these job posting platforms to recommend desirable jobs to job seekers based on their career interests to deliver high value for both job posters and seekers to become qualified matches, and ultimately serve a reliabled ecosystem of job markeplace themselves.

In the recommender system, there are two general approaches to provide recommendations. Consider the job marketplace. One approach is based on content, where job seekers' preferences are predicted depending on how their explicit skills, titles, industries and etc. match the jobs. It involves extensive data from job seekers and jobs so that features can be complicated. Another approach is consider jobs preferred by similar job seekers via collaborative filtering which leverages less data. In collaborative filtering, one way to obtain similar job seekers is through clustering [1] by pre-training K-means clustering of seekers based on their applications to jobs. However, compared to one-way clustering, co-clustering on both job seekers and jobs simultaneously can may provide better performance and the constructed clusters can be directly used to improve the quality of job recommendations.

In the follow sections, first, I will briefly discuss related work on co-clustering and the data set to use in this project. Second, I will introduce two co-clustering methods I experimented to deal with job applications. Third, I will discuss the validation process, compare the performances of co-clustering methods with the baseline algorithm - K-means and demonstrate visual comparision. At the end is the conclusion.

2 Related Work

There are many co-clustering approaches that simultaneously clustering rows and columns of a given matrix experimented on clustering documents and words. Some of them are representative while the others more or less extend their ideas, as they use different methodologies but acheive the the goal. [2] attempted to minimize the loss of mutual information between original and clustered random variables. [3] proposed a spectral co-clustering algorithm based on bipartite graph, turning co-clustering into graph partitioning. [4] and [5] focused on low-rank matrix factorization and related it with simultaneously clustering rows and columns. The latter two involves matrix decompostion which are slow and computationally expensive so the work [6] that utilizes co-clustering in collaboritive filtering doesn't pick them for static training.

To explore co-clustering jobs and job seekers via job application in this project, intuitively, [3] and [4] can possibly provide insightful clusters. The former one tries to build clusters that have more intra-cluster applications in and less inter-cluster applications, which can produce exclusive job clusters. The latter one instead can discover latent clusters for job seekers and jobs during matrix trifactorization to approximate the original seeker-job matrix.

3 Data set and Preprocessing

The data set is from CareerBuilder's competition (https://www.kaggle.com/c/job-recommendation/data). It contains job applications lasting for 13 weeks. For each application record, each row in the original data set contains UserID, ApplicationDate and JobID. ApplicationDate is not considered in the project when splitting the data set into training and testing sets though it seems more fair to use job applications happened later to test against clusters trained on previous job applications, due to the fact that job postings are usually posted in a limited period of time and receives the majority of all applications as soon as they are posted and as a result, jobs applied in the training set will hardly appear in the testing set if splitted by ApplicationDate, which leads to insufficient and less representative testing datapoints. Therefore, I extracted UserID and JobID for each application from the raw data, removed duplicated applications, shuffled them randomly and transformed to 0-1 valued user-job matrices, whose rows represent unique users, columns represent unique jobs and entries represent whether the user applies for the job, since both of the two co-clustering methods take an input of user-job matrix.

One concern is that the original user-job matrix is very sparse which contains non-trivial noise for co-clustering. So I also filtered out jobs whose job has less than certain number of job applications to reduce the sparsity. Table 1 shows the statistics when using different threshold for each job to filter:

So the general preprocessing steps are:

1. Create two data sets of job applications from the original data set by filtering on different number (75 and 100) of job applications per job.

Table 1

Minimum # of Job Applications per Job	Density	# of Job Applications	# of Jobs	# of Users
75	0.32%	79666	823	29957
100	0.73%	32873	271	16449

2. For each data set generated above, split into 5 partitions and create 5 pairs of training set and testing set for 5-fold cross validation.
3. For each training set, convert it into 0-1 user-job matrix for clustering. Two data sets yield user-job matrices with different densities. The sizes of training and testing set for each density is in Table 2.

Table 2

Minimum # of Job Applications per Job	Training	Testing
75	63732	15934
100	26298	6575

4 Methods

4.1 One-way clustering: K-means

The baseline clustering method is one-way clustering on users using K-means. Each user is represented by a vector u where

$$u_i = \begin{cases} 1 & \text{if the user applies to the } i\text{th job} \\ 0, & \text{otherwise} \end{cases}$$

K-means will cluster these user vectors based on the Euclidean distance.

4.2 Co-clustering: Nonnegative Matrix Tri-factorization

This method is proposed in [4]. Basically, Given the nonnegative user-job matrix X, Nonnegative Matrix Tri-factorization is derived from NMF which factorizes the nonnegative X into 2 nonnegative matrices,

$$X \approx FG^T$$

If imposing orthogonality on both F and G, the objective is:

$$\min_{F\geq 0, G\geq 0} ||X - FG^T||^2, s.t. F^T F = I, G^T G = I$$

As shown in [7], it is equivalent to simultaneously runninig K-means clusterins of rows and columns of X. Since the orthogonality requirement is very strict, instead, we can consider the 3-factor decomposition,

$$X \approx FSG^T$$

and the objective function then becomes

$$\min_{F\geq 0, G\geq 0} ||X - FSG^T||^2, s.t. F^T F = I, G^T G = I$$

where F is the cluster indicator matrix of row clusters and G is the cluster indicator matrix of column clusters because as described in [4], G is the K-means clustering result using kernel matrix $X^T F F^T X$ to calculate the distance and similarily, F is the K-means clustering result using kernel matrix $XGG^T X^T$.

Since our user-job matrices contain 0-valued entries when the corresponding user (row) doesn't apply to the corresponding job (column), I replaced 0 with a small fractional number 0.01 which can be distinctive from the 1-valued entries before running this method then implemented the following algorithm to minimize the objective function and obtain F and G:

Algorithm NMTF

Initialization:

1. Run K-means of columns to obtain column cluster centroids as G.
2. Run K-means of rows to obtain row cluster centroids as F.
3. Let $S = F^T XG$

Update rules:

$$G := G \frac{X^T FS}{GG^T X^T FS}$$

$$F := F \frac{XGS^T}{FF^T XGS^T}$$

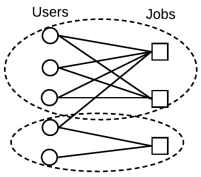
$$X := S \frac{F^T XG}{F^T FSG^T G}$$

Then the cluster membership of rows and columns are extracted from F and G.

4.3 Co-clustering: Spectral Co-clustering

This method is from [3] which leverages partitioning bipartite graph to achieve co-clustering. Giving the 0-1 user-job matrix X, a bipartite graph is constructed with two vertex sets representing users and jobs respectively with edge weight be 1

Figure 1



or 0 as illustrated in Figure 1. Then the adjancy matrix is

$$M = \begin{bmatrix} 0 & X \\ X^T & 0 \end{bmatrix}$$

Let D be

$$\begin{bmatrix} D_{row} & 0 \\ 0 & D_{col} \end{bmatrix}$$

where $D_{row}(i,i) = \sum_j X_{i,j}$, $D_{col}(j,j) = \sum_j X_{i,j}$ We have the Laplacian matrix $L = D - M$. It is proved in [3] that the second eigenvector of the problem $Lz = \lambda Dz$ leads to a relaxed solution of finding the minimum normalized cut (a cut is defined as the sum of weights of edges connecting two partitions) of this bipartite graph, in the meantime balancing partitioning sizes. [3] also shows to obtain the second eigenvector of L, we need to compute singular vectors of $D_{row}^{-\frac{1}{2}} X D_{col}^{-\frac{1}{2}}$ first. To get the clustering result, we can directly run K-means on the second eigenvector of L which is a reduced-dimensional data set.

5 Experiments

5.1 Preprocessing the testing set

Since there are no labels for either jobs or users, the way to validate the trained clusters and generate usefuly metrics is to verify job applications in the testing set against the trained clusters. Since we shouldn't verify the same job applications in the testing set as the training set and are unable to verify jobs and users in the testing set that are unseen in the training set, I preprocessed the testing test to extract unseen job applications whose corresponding jobs and users were involved in the training set.

5.2 Metrics

Due to lack of label, the metrics to verify the effectiveness of clustering algorithms are **recall**, **accuracy** and **F1-score**. To compute them, we need to compute the metrices:

- **True Positive**: a job applied by a user in the testing set is in the same cluster of this user in the trained clusters.
- **False Positive**: a job not applied by a user in the testing set is in the same cluster of this user in the trained clusters.
- **True Negative**: a job not applied by a user in the testing set is not in the same cluster of this user in the trained clusters.
- **False Negative**: a job applied by a user in the testing set is not in the same cluster of this user in the trained clusters.

In practical, the ways to determine the cluster which a job belongs to are different for different clustering methods:

- **K-means**: Any cluster if any user in this cluster applies to this job in the training set
- **NMTF**: Though it is co-clustering jobs and users, the cluster labels of jobs and users in the same co-cluster are different. So follow the K-means way.
- **Spectral Co-clustering**: Since if jobs and users are in the same co-cluster, their cluster labels are same. So use the job cluster label.

The metric to determine the optimal number of cluster is silhouette score. The silhouette score for each user i is

$$s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))}$$

$a(i)$ is the dissimilarity to the user's cluster and $b(i)$ is the minimum dissimilarity to other clusters where the disimilarity to a cluster for a user is defined by

$$1 - \frac{\text{\# of jobs applied in this cluster by this user}}{\text{total \# of jobs applied by this user}}$$

The average of the cluster silhouette score (i.e., the average of the user silhouette score in this cluster) will be used to determine the optimal number of clusters.

5.3 Experiments and Results

5.3.1 Performance

The recall, accuracy and F1-score are computed on different number of clusters for two data sets with different densities. The results are illustrated in Figure 2, 3 and 4 (Density 1 is larger and less dense than Density 2).

We can see although the recall of K-means is not bad since the jobs are overlapping in different clusters, it can provide many true-positives.

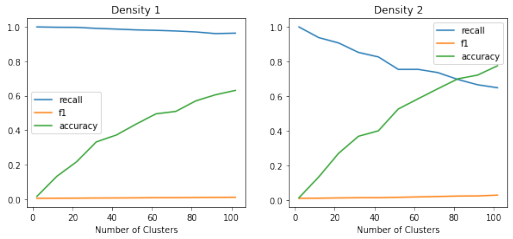


Figure 2: K-means

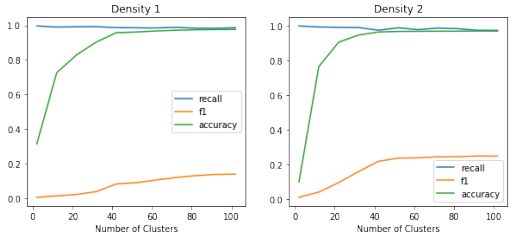


Figure 3: NMTF (Nonnegative Matrix Tri-factorization)

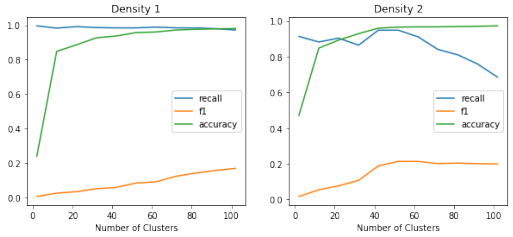
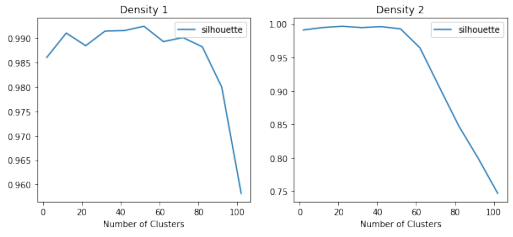


Figure 4: Spectral Co-clustering

When coming to accuracy, co-clustering methods NMTF and spectral co-clustering have much higher accuracies since their job clusterings are more exclusive, there are much less false-positives.

5.3.2 Silhouette Analysis for Number of Clusters

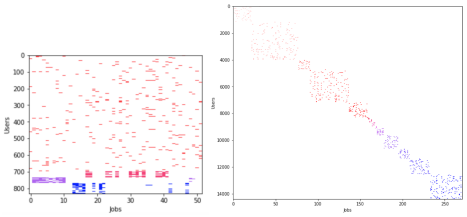
By running 5-fold cross-validation and computing silhouette scores on different number of clusters for two data sets with different densities, the result is: By observing the coordinates where the



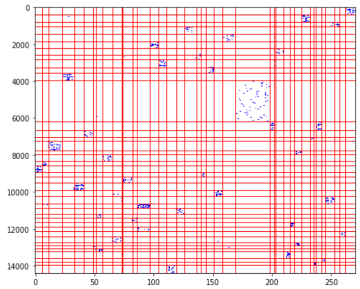
silhouette scores start to drop, for the first data set which is larger and more sparse, the optimal number of cluster is ≤ 80. For the second one which is smaller and more dense, the optimal number is ≤ 50.

5.3.3 Visual Comparision

The figures (a), (b) and (c) visualize example clustering results using K-means, Nonnegative Matrix Tri-factorization and Spectral Co-clustering on user-job matrix.



(a) K-means (b) Spectral Co-clustering



(c) NMTF

We can observe some straightforward facts that

1. Spectral Co-clustering produces more balanced co-clusters of users and jobs as its algorithm is designed for while Nonnegative Matrix Tri-factorization may result in one big cluster while the others are relatively small.
2. Nonnegative Matrix Tri-factorization allows specifying different numbers of clusters on rows and columns though normally they are specified as same. Then it requires extra steps to corresponding the user cluster to the job cluster if they actually represent the same co-cluster, as their cluster labels are different, which is not as convenient as Spectral Co-clustering.
3. Like NMTF, one-way clustering using K-means produces inbalanced clusters. More importantly, unlike the co-clustering methods, from the visualization we can tell in the big cluster, the jobs applied by the users in the same cluster overlap heavily with the ones

in other clusters, which will introduce many noises for job recommendation, while the users in the rest small clusters are matched so accurately that it might miss potential suitable jobs to recommend.

6 Conclusion

It is shown that simultaineously clustering users and jobs based on merely job applications using different co-clustering methods can produce exclusive clustering of jobs, which improves the accuracy as jobs with potentially higher apply rate can be recommended compared to one-way clustering. In addition, Spectral co-clustering is designed to construct more balanced clusters, which is more ideal for recommendation by supplying more accurate pools of jobs to recommend.

In the future, since both co-clustering methods are slow because they leverage matrix decomposition, it's beneficial to explore more scalable co-clustering methods so that we can co-cluster efficiently on more sparse data set.

7 Contributions

All is done by myself.

8 Github

The source code is in
https://github.com/qingyunwan/cs229-project.

References

[1] Ungar, L. H., & Foster, D. P. (1998, July). Clustering methods for collaborative filtering. In AAAI workshop on recommendation systems (Vol. 1, pp. 114-129).

[2] Dhillon, I. S., Mallela, S., & Modha, D. S. (2003, August). Information-theoretic co-clustering. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 89-98). ACM.

[3] Dhillon, I. S. (2001, August). Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 269-274). ACM.

[4] Ding, C., Li, T., Peng, W., & Park, H. (2006, August). Orthogonal nonnegative matrix t-factorizations for clustering. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 126-135). ACM.

[5] Long, B., Zhang, Z. M., & Yu, P. S. (2005, August). Co-clustering by block value decomposition. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 635-640). ACM.

[6] George, T., & Merugu, S. (2005, November). A scalable collaborative filtering framework based on co-clustering. In Data Mining, Fifth IEEE international conference on (pp. 4-pp). IEEE.

[7] Ding, C., He, X., & Simon, H. D. (2005, April). On the equivalence of nonnegative matrix factorization and spectral clustering. In Proceedings of the 2005 SIAM International Conference on Data Mining (pp. 606-610). Society for Industrial and Applied Mathematics.