CS568 : Data Mining

Review of related work and Code Architecture

Kushal Sangwan (180101096)
Milind Prabhu (180101091)
Varhade Amey Anant (180101087)

Overview of the selected algorithm, variants of the algorithm, related work review, code architecture and class design, implementation details

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Contents

1 PROBLEM DEFINITION AND OVERVIEW

1.1 Bottleneck issue this clustering problem resolves

This algorithm solves the co-clustering problem for words and documents together. The existing algorithms either cluster the documents based on word distributions or cluster words based on common occurrence in documents. This algorithm simultaneously clusters both words and documents through partitioning the corresponding bipartite graph. This is possible because of the duality between the word clustering and the document clustering i.e, related documents have common words and related words occur in the same document.

1.2 Existing algorithms

Also, existing graph theoretic algorithms for the document-clustering problem usually choose the vertices of the graph to represent documents and the weights on edges between vertices to be some measure of similarity between the documents. These algorithms take quadratic time in terms of the number of documents to construct the similarity graph which is computationally prohibitive. The algorithm proposed in the paper chooses the bipartite graph as the similarity graph between words and documents which can be constructed in linear time with respect to the number of documents.

2 IDEA, MATHEMATICAL INTUITION

2.1 Mathematical Representation

The main idea of the algorithm is to exploit the duality between the document clustering and word clustering problem, i.e, one induces the other and thus better word clustering leads to better document clusters. The "best" for both cases will correspond to finding a partitioning of their bipartite graph with total weight of "cross" edges between partitions being minimum while maintaining the balance between partitions (i.e Normalized cut) for which the following algorithm is proposed.

Considering this, the intuition is to come up with a combined 'cost function' which captures both the aspects:

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{weight(V_1)} + \frac{cut(V_1, V_2)}{weight(V_2)}$$

Using, the properties of the Laplacian $\mathbf{L}$, the Weight matrix $\mathbf{W}$ and a generalized partition vector $\boldsymbol{q}$ we discover a equivalence between the Q function and a standard expression from linear algebra

$$\frac{\mathbf{q^T L q}}{\mathbf{q^T W q}} = Q(V_1, V_2)$$

The discreteness condition on q makes this optimization problem NP-hard. Finding the minimum for a real relaxation to expression on the left is what the following algorithm tries to do , but instead of using eigenvalues and eigenvectors, it tries to use the second left and right singular vectors of an appropriately scaled word-document matrix to yield good bipartitionings.

2.2 Algorithm implementation details

The **Bipartitioning algorithm** involves the following three steps. Although the last two steps might seem very obscure they can be justified by some neat mathematics.

(1) **Graph Representation:** Represent the data as a bipartite graph $G(U, V, E)$, the two disjoint sets U,V being the set of documents and the set of words appearing in these documents respectively. An edge between a document and word is assigned a weight which is a measure

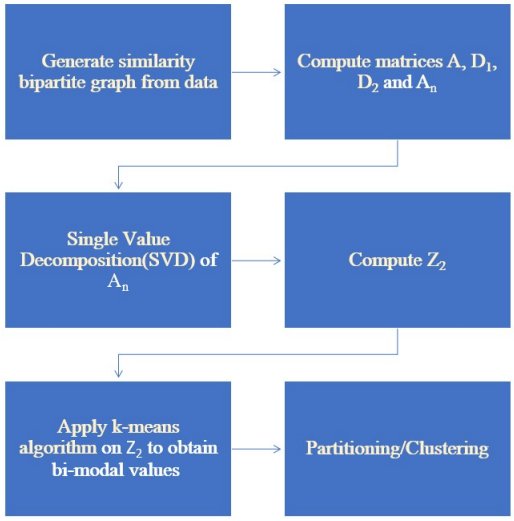

Fig. 1. Overview of the bipartitioning algorithm

of the word's relation to the document (say, frequency of the word in the document). For convenience we let $|U| = w$ (the number of documents) and $|V| = d$ the number of words.

(2) **Singular Value Decomposition (SVD):** Let A be the the word by document matrix. Also, D_1 be a $w \times w$ diagonal matrix such that $D_1(i,i) = \sum_j A_{(i,j)}$ and D_2 be a $d \times d$ diagonal matrix D_2 such that $D_2(j,j) = \sum_i A_{(i,j)}$. The second step of the algorithm involves the computation of the second singular vectors u_2 and v_2 of the normalized matrix A_n where $A_n = D_1^{-1/2} A D_2^{-1/2}$. Here while u clusters the words , v clusters the documents.

(3) **Using k-means**: As the above vectors correspond to a real relaxation to the optimal generalized partition vector, one needs to find two bi-modal values to cluster using u and v. Run k-means on the one dimensional data $[D_1^{-1/2}u_2, D_2^{-1/2}v_2]$ to obtain these values and then finally the clustering.

The **Multipartitioning algorithm** is similar to the Bipartitioning algorithm. In step (2), $l = \lceil \log_2(k) \rceil$ singular vectors of A_n, $u_2, u_3 \cdots u_{l+1}$ and $v_2, v_3 \cdots v_{l+1}$ are computed. The l value corresponds to the fact in the optimal partitioning vector,we can have $2^l = k$ different values corresponding to the k different clusters. To obtain these values for the real relaxation solution obtained from step (2),in step (3) the k means algorithm is executed on the l-dimensional data $\begin{bmatrix} D^{-1/2}U \\ D^{-1/2}V \end{bmatrix}$ where $U = [u_2, u_3 \cdots u_{l+1}]$ and $V = [v_2, v_3 \cdots v_{l+1}]$.

3 EVALUATION OF THE ALGORITHM

The algorithm is evaluated both for its bipartitioning and multipartitioning variants on large as well as small datasets. The paper shows the confusion matrices for the results of these tests and mentions that the goodness was gauged by the computation of the confusion matrix as well as by **purity** and **entropy** calculated from the confusion matrix.

Purity is an external evaluation criterion of cluster quality. It is the percentage of the total number of objects(data points) that were classified correctly. Entropy is another measure of performance of the algorithm. A higher entropy indicates that the cluster outputs by the algorithm in reality contains data points which belong to different labels or classes thereby saying that the algorithm is not doing well.

For the evaluation of the bipartitioning algorithm, the datasets Medline(medical abstracts), Cranfield(Aeronautical abstracts) and Cisi(information retrieval abstracts) were used. The tests were performed by mixing two of the above three datasets. In some cases care was taken to discard stop words and words that occurred either in a large fraction of the documents or in an insignificant number of documents. The authors however point out that the algorithm did well even when stop words were included.

For the multipartitioning algorithm, the evaluation was performed on a combination of the previous three datasets and also on a Yahoo datasets which consisted of news articles 6 categories: Business, Entertainment, Health, Politics, Sports and Technology. The documents in the Yahoo dataset were html pages and were pre-processed to discard HTML tags.

4 EXISTING IMPLEMENTATIONS, DATASETS

Primarily, two types of datasets are used, one for the bipartitioning and one for the multipartitioning varaint of the algorithm each. We were able to find some of the datasets used by the author:

- Bipartitoning Algorithm
  Cranfield, *Medline* and *Cisi*
- Multipartitoning Algorithm
  The above three datasets and the Yahoo dataset [Not Available].

The specific code used by the author was not available however we did find a *python implementation* of the algorithm in the sci-kit learn library

5 REAL WORLD APPLICATIONS

The applications of the document clustering problem are:

- Biomedical applications to classify patient symptoms and medical diagnoses
- A customer relationship management (CRM) application where you want to co-cluster customers and items purchased.
- Movie recommendation engines engines co-cluster accumulated movie ratings from viewers. When a new viewer submits a score for a film she liked, the engine recommends other movies based on classifying the rating she provided to a cluster of audience movie ratings.
- Document Clustering

5.1 Standard Library Implementation

The sci-kit learn python package has an implementation of the spectral co-clustering algorithm and it also cites this paper itself as the primary reference.

sci-kit learn 2.4.1 Special Co-Clustering

6 EXISTING INCREMENTAL ALGORITHMS

To the best of our knowledge, there is no incremental version of the selected algorithm. In case of algorithms for partitioning dynamic bipartite graph, they function on evolutionary data, which means they take into account the historical relationship between the data points into consideration and want smoother transitions in clusters[2], which is not what we want for incremental clustering. The two main steps involved in our algorithm are finding the second eigenvector of the Laplacian and running k-means on it. We therefore thought it might be worthwhile to check if each of these steps can be made incremental to make the overall algorithm incremental.

The step which finds the second eigenvector involves performing the SVD. Therefore we checked to see what work had been done on making SVD incremental. Existing algorithms used for incremental SVD [7] just handle the scenario of addition of data points assuming arrival in chunks while completely ignoring the deletion as well as updation possibility.

Although we did not find incremental algorithms for SVD, we did find some work [4] which discusses how changing the similarity matrix changes the eigenvalue system of the Laplacian. At the end we anyway calculate the second eigenvector of L using right and left singular vectors of A_n (normalized incidence matrix), and the modifications in this vector is only the thing that we require which can be given by the following algorithm. Incremental Spectral Co-Clustering [4] computes changes $\delta\lambda$ and δq in eigenvalues and eigenvectors. It models addition and deletion of data points as similarity changes, and differentiates $Lq = \lambda Dq$ for getting the formulas for changes in eigen spectrum, which are used to change λ and q until both converge.

7 RELATED VARIANTS OF THE ALGORITHM

7.1 Variant 1: Co-clustering documents and words using Bipartite Isoperimetric Graph Partitioning [5]

Details: It also solves the co-clustering problem by partitioning the bipartite graph but formulates normalized cut problem as that of minimizing the isoperimetric ratio for the clusters. The spectral SVD algorithm in general fails to get solutions for some families of problems for example, roach graphs and also, eigenvalues in the bulk of the spectrum are typically harder to approximate with the popularly used lanczos method for SVD. Instead of doing the SVD it solves a system of linear equations to get the two partitions for a bipartitioning problem. It gets the linear equation system by doing three things: using a Cost function and Lagrange multiplier for the optimization problem, differentiating for getting the optimum and thus getting equation $Lz = d$, where d is nothing but vector of outgoing degree of each vertex, then it eliminates one vertex to make L-non singular and solves the formed linear equations. k-means is used on the obtained solution to form the clusters.

For multipartitioning it recursively calls the bipartitioning algorithm until the isoperimetric ratio falls below a certain value. This ICA algorithm is proved to be computationally faster,better in quality and stable then SVD based algorithm based on experimental results.

7.2 Variant 2: Learning A Structured Optimal Bipartite Graph for Co-Clustering [3]

Details: This paper also proposes an algorithm which takes as input a bipartite similarity graph as in the selected paper and outputs k clusters. The main motivation for this approach was that algorithms such as those proposed in [1] first process the original graph and later on apply the k-means algorithm (or some other similar algorithm) to obtain the final clustering. The k-means algorithm is sensitive to the initialization and might possibly make the clustering performance unstable and sub-optimal. The algorithm attempts to find the bipartite graph with k connected components which is closest to the input graph. The new bipartite graph learned maintains an explicit cluster structure, from which clustering results can be obtained without post-processing.

7.3 Variant 3: Efficient semi-supervised Spectral Co-clustering with Constraints [6]

Details: The SCM algorithm proposed here solves the optimization problem of minimizing the normalized cut along along with maximizing the inclusion of the knowledge that some rows or columns are known to be in the same cluster a priori. For example in document co-clustering, it may already be known that papers of conferences KDD and ICDM belong to same group as both conferences are based on data science, while the words' clustering and co-clustering can be taken to be similar. Most co-clustering algorithms do not use this information and hence may produce inaccurate clusterings. Such info is taken into account by modifying the normalized matrix A_n taken in our algorithm to also include the incidence and degree matrix of the constraints. The purity and normalized mutual information obtained for word-document and graph-pattern co-clustering datasets were much higher than our selected algorithm in presence of 5% constraints.

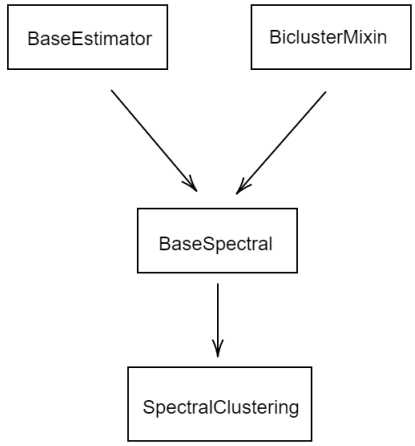

Fig. 2. The class hierarchy of the SpectralClustering class in the scikit-learn library.

8 CODE ARCHITECTURE USED IN THE SCIKIT-LEARN LIBRARY

8.1 Data Structures

8.1.1 Input Data Representation.

(1) X: an $m \times n$ matrix representing the bipartite graph where $X[i, j]$ denotes the association of row i with column j.
(2) D_1: $m \times m$ diagonal matrix with row sums
(3) D_2: $n \times n$ diagonal matrix with column sums
(4) A_n: $m \times n$ The normalized matrix

8.1.2 Data generated.

(1) u: an $l \times m$ matrix of left singular vectors where $l = \lceil \log_2 k \rceil$
(2) v: an $l \times n$ matrix of right singular vectors
(3) z: $[D_1^{-1/2}u \quad D_2^{-1/2}v]$ matrix of dimension $l \times (m + n)$

8.2 Class Design

The code architecture is explained through the below classes and their properties and methods (see Figure 2)

(1) SpectralCoClustering :
Properties
- `n_clusters` The number of clusters. Has value 2 for the bipartitioning algorithm and higher values for multipartitioning.
- `svd_method` To decide which algorithm out of 'randomized' and 'arpack' to use to compute SVD.
- `n_svd_vecs` Number of vectors used in calculating the SVD .
- `mini_batch` Boolean specifying whether to use mini-batch k-means or not.
- `init` Initialization method of k-means algorithm.
- `n_init` Number of random initialization for k-means

Attributes

- `row_labels_, column_labels_` The results of clusterings are stored in these arrays

Methods

- `__init__` Used for parameter initialization.
- `_fit(self, X)` Computes the SVD of normalized matrix and finds the `row_labels_, column_labels_` array values.

(2) BaseSpectral :

Methods

- `_check_parameters` Verifies the parameters used for initialization
- `_svd` Returns the first 'n_components' of left and right singular vectors u and v, discarding the first `n_discard` components
- `_k_means` The k-means algorithm is run and returns the centroid and the labels

(3) BaseEstimator :

- `_validate_data` Used to validate the training data.

(4) BiClusterMixin :

- Mixin class for all bicluster estimators in scikit-learn.

(5) ABCMeta :

- This is a meta class defining the properties of BaseSpectral class. For example, parents may be defined by ABCmeta (just a hyphothetical example) and as BaseSpectral is an instance of ABCmeta, it has the same parents. Basically classes are instances of meta classes.

8.3 Summary

The SpectralCoclustering Class implements the algorithm. First, the `scale_normalize` method obtains the `normalized_data(`A_n`)` , `row_diag(`D_1`)` , `col_diag(`D_2`)` from the incidence matrix `X` . Then the `svd` method takes the arguments `normalized_data`, number of singular vectors ($l = log_2 k$), and vectors to discard which is 1(we start from second singular vector) and returns matrix `u` and `v` containing left and right singular vectors. Then `row_diag*u` and `col_diag*v` are stacked together to obtain the matrix `z`. `k_means` method is run on the matrix `z` to get list the of labels. These labels are used to construct indicator vectors for the clusters and stack them together for both rows and columns. The `k_means` and `svd` method belong to the parent class BaseClustering. In `k_means`, either the mini-batch k-means or batch k-means is executed depending on the given option and similarly in svd, one out of `randomized` and `arpack` is executed.

9 IMPLEMENTATION IN C++

We did not find any ready to use implementation of the algorithm in C++ language. We have implemented the complete algorithm in C++ using the *Armadillo* and MLpackpackage for linear algebra and other standard matrix methods. We have implemented some parts while some are left

9.1 Input Format and Datasets

The inputs to the algorithm are a file containing the word by document matrix and another which contains the true labels. The second file is used only to print the confusion matrix and is not necessary to obtain the clustering.

We have only tested the code on very simple test cases to verify its correctness(see figure **??**). The test files have been provided in the example_code folder. We plan to test this on standard datasets such as the medline,cranfield and cisi datasets available at *Common IR Test Collections*. They were available in Harwell-Boeing compressed format and we have converted them to matrix market format using our converter program.

These datasets are to be further pairwise combined to get matrices for MedCran, MedCisi datasets. These matrices are the ones which are fed to the algorithm. Further for multipartitioning combination of all three is needed.

9.2 Implementation

We have implemented the Spectral Co clustering class,Base Clustering class ,normalize_Data classes of our implementation plan . We are also getting cluster assignments and the corresponding confusion matrix on small inputs.

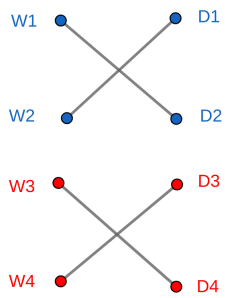

Fig. 3. The sample test case. The algorithm partitions the graph into the red and blue clusters.

```
milind@Workstation73:~/data_mining/new_files$ ./a.out
The cluster assignments are:
       0       0       1       1       0       0       1       1

The confusion matrix for the obtained clustering is:
       4       0
       0       4
```

Fig. 4. The output of the algorithm

9.3 File Structure

The primary file having implementation of classes mentioned is `kpartition.cpp`. The file used for the conversion of matrix formats is `converter.cpp`. We have created a sample input matrix, which is read from the file `input.txt`.

9.4 Future Work

The MedCisi, MedCran and Classic3 combination datasets have not been created by us yet from the converted matrix market format. We have hence, not completed the testing and evaluation of the algorithm on the datasets used by the author in the publication.

REFERENCES

[1] I. S. Dhillon. 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*, pages 269–274, 2001.

[2] N. Green, M. Rege, X. Liu, and R. Bailey. Evolutionary spectral co-clustering. In *The 2011 international joint conference on neural networks*, pages 1074–1081. IEEE, 2011.

[3] F. Nie, X. Wang, C. Deng, and H. Huang. Learning a structured optimal bipartite graph for co-clustering. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4132–4141, 2017.

[4] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1):113–127, 2010.

[5] M. Rege, M. Dong, and F. Fotouhi. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 532–541. IEEE, 2006.

[6] X. Shi, W. Fan, and P. S. Yu. Efficient semi-supervised spectral co-clustering with constraints. In *2010 IEEE International Conference on Data Mining*, pages 1043–1048, 2010.

[7] X. Zhou, J. He, G. Huang, and Y. Zhang. Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, 81(4):717–733, 2015.