CS290H Graph Laplacians and Spectra

Final Project Report

Categorization of biomedical articles with spectral clustering

By Arvind C. Rajasekaran

Abstract

Clustering is the process of grouping together similar entities. This can be applied to documents, and has applications in information retrieval, where similar documents are clustered and retrieved together. In this project I explore spectral clustering and its variants Normalized cut, as well as non-negative matrix factorization(NMF) and its relevance in clustering biomedical documents. In spectral clustering we try to transform the given data in such a manner that conventional algorithms like kmeans can easily detect the correct patterns. Though spectral and NMF clustering are known to give good performances on newsgroup dataset, their performance on biomedical text which involves alphanumeric, abbreviations and ambiguous words have not been properly explored. This project compares the performance of spectral clustering, NMF clustering and kmeans clustering on pubmed abstracts.

Introduction to clustering methods

Generally clustering methods can be classified as agglomerative and partitional. Agglomerative clustering groups the documents into hierarchical trees or dendograms by merging similar clusters to form parent cluster. But these have complexity of the order O(n^2 log n). Because of this quadratic order of complexity, bottom up clustering could become expensive for large datasets. On the other hand partition based clustering composes a corpus into a set of disjoint clusters. Typically this includes Kmeans, naïve bayes and Gaussian mixture models. Kmeans produces a cluster set that minimizes the mean squared error while naïve bayes and Gaussian models assign documents to clusters that provide maximum likelihood probability. Kmeans assumes each cluster has a compact shape, naïve bayes assumes all the dimensions are independent of each other, and Gaussian mixture model assumes that each cluster can be approximated by a Gaussian distribution. Obviously these assumptions are often not true and document clustering results go terribly wrong with broken assumptions.

There has been numerous studies that perform document clustering using latent semantic indexing. This method basically projects each document into the singular vector space using SVD and conducts document clustering using traditional data clustering algorithms like Kmeans in the transformed space. Although it was claimed that each dimension of the singular vector space captures a base latent semantic of the document corpus, and that each document is indexed by the base latent semantics in this space, negative values in some of the dimensions generated by the SVD make the above explanations less meaningful.

In recent years spectral clustering based on graph partitioning theories has emerged as one of the most prominent document clustering methods. These methods model each document using an undirected graph in which each node represents a document and each edge (i , j) is assigned a weight to represent the similarity between documents i and j. The graph clustering is performed by finding the best cuts of the graph that optimize predefined criterion function. The optimization of the criterion function usually leads to the computation of singular vectors or eigen vectors of graph affinity matrices and the clustering results can be derived from the eigenvector space. Many criterion functions such as normalized cut, average cut, average association, min-max cut have been proposed along with efficient algorithms for optimizing their solutions.

Document term matrix and Similarity matrix

A document-term matrix or term-document matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms. There are various schemes for determining the value that each entry in the matrix should take. One such scheme is tf-idf. In this project the document term matrix with tf-idf is used.

The similarity matrix can be constructed from term-document matrix using Euclidean distance or jaccard similarity or cosine similarity between the document vectors. For out experiment cosine similarity is used.

Spectral clustering and normalized cuts

The method is based on cutting the graph of object's similarities using methods of spectral graph theory. In recent years this theory has been strongly developed, especially in direction of graph clustering algorithms where the most well-known are: Shi–Malik (2000),Kannan–Vempala–Vetta (2000), Jordan–Ng–Weiss (2002) and Meila–Shi (2000).

Representing the objects that are to be clustered with graph nodes and the w weights of the graph edges with objects similarities, the partitioning problem is reduced to finding optimal cutset. Sometimes cutset gives the results that are different than intuitive nodes partition, so other measures are introduced – example - Normalized cut which increases its value for clusters that have nodes with small sum of edge weights.

1. Preprocessing and data normalization

At this stage the data are preprocessed into their computational representation. If we are clustering the text documents typically Vector Space Model (VSM) is used. In this representation weighting with Term Frequency and Inverse Document Frequency (TFIDF) allows to calculate similarities between documents. In our experiments we use cosine distance which is known to be the suitable similarity measure for sparse vectors.

2. Spectral mapping

This stage distinguished spectral approach. Using the data from step 1 the typically Laplacian matrix is built and then appropriate number of its eigenvectors is calculated.

3. Clustering

The objects represented with spectral mapping are divided into two or more sets. Sometimes it is enough to find appropriate cut of the n-element, sorted collection which divide this collection into two clusters. In other methods this step is more complicated and performs partitioning in new representation space (provided by spectral mapping) using standard clustering algorithm e.g. k-Means

Shi–Malik is realized in following steps:

(a) Calculate eigenvectors of similarity Laplacian graph.

(b) Sort elements of the dataset according to second smallest eigenvector value, which is denoted as x1,x2, ...,xn.

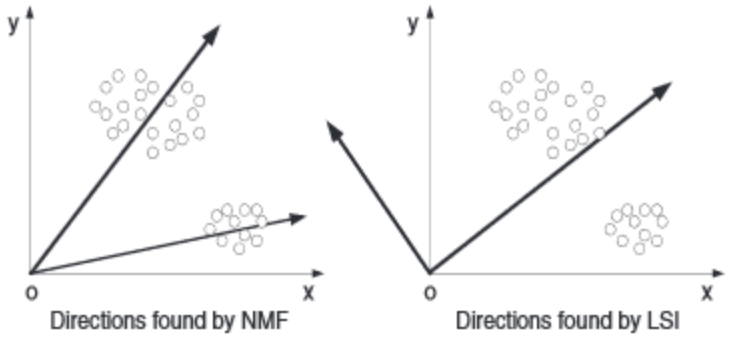
(c) Calculate the partition{{x1, x2..., xi},{xi+1, xi+2, ..., xn}}($1 \leq i \leq n1$) having the smallest NCut.

(d) If given partition has NCut value smaller than given a priori value (that means it is better) then this method in each of the divided sets is run again, otherwise the algorithm stops

Non-negative Matrix Factorization

Document clustering by LSI and spectral clustering strive to find semantic structure of the corpus by computing eigen vectors of certain matrices. The derived latent semantic space is orthogonal and each document can take negative values in some directions. In contrast non-negative matrix factorization doesnot have the orthogonal requirement and it guarantees that documents take only non negative values. Thus NMF is superior due to the following reasons

1. First, when overlap exists among clusters, NMF can still find a latent semantic direction for each cluster, while the orthogonal requirement by the SVD or the eigenvector computation makes the derived latent semantic directions less likely to correspond to each of the clusters.
2. Second, with NMF, a document is an additive combination of the base latent semantics, which makes more sense in the text domain.
3. Third, as the direct benefit of the above two NMF characteristics, the cluster membership of each document can be easily identified from NMF, while the latent semantic space derived by the LSI or the spectral clustering does not provide a direct indication of the data partitions, and consequently, traditional data clustering methods such as K-means have to be applied in this eigenvector space to find the final set of document clusters.



Dataset

For the dataset I have used a collection of 216 abstracts belonging to 6 journals on Pubmed. These documents are available for download from pubmed in xml and are converted to text files using the DTD's from pubmed and a parser written in java.

Experimental Result

Due to the large size of the term document matrices, clustering was performed on a small set of the documents (6 journals, implying 6 clusters). A total of 216 abstracts belonging to the 6 journals were converted to term document matrices and similarity matrices using Term to Matrix Generator tool for matlab. Performance of k-means, non-negative matrix factorization and N-cut spectral clustering are compared.

Algorithm	**No of documents(clusters)**	**Rand Index**	**Accuracy**
Kmeans	216(6)	81.8182	58.3333
NMF	216(6)	95.4545	91.6667
Ncut-Spectral	216(6)	72.7273	41.6667

Observations and Future work

It can be seen that non-negative matrix factorization gives much better results compared to traditional kmeans and n-cut based methods. This shows that NMF can make a huge difference in biomedical information retrieval. Due to the orthogonal vector requirement in NCut, the results are not as good as Kmeans or NMF. Expanding this analyses to a larger number of clusters using a cloud computing cluster and exploring related algorithms in the kmeans, spectral and NMF family of clustering algorithms remain future work.

References

1. A vector space model for document retrieval – G Salton
2. Learning Spectral Clustering – Francis Bach

3. Document Clustering Based On Non-negative Matrix Factorization – Wei Xu
4. Text to matrix generator matlab toolbox

Appendix

Java parser for xml to text conversion

```
import java.nio.charset.Charset;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.io.*;

import java.nio.file.DirectoryStream.Filter;

import java.util.*;

import java.io.FileWriter;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import com.google.common.io.*;

import org.apache.commons.io.FileUtils;

import org.w3c.dom.Document;

public class FileWalk {

   File startDir;

   Filter filter;

   public FileWalk( File sDir ){

         this(sDir, null);

   }


   public FileWalk(File sDir, Filter fil){

         startDir = sDir;

         filter = fil;

   }
```

```
    public void FSWalk () throws FileNotFoundException {
        validateDirectory(startDir);
        recurse(startDir);
    }
    public void recurse (File startDir) {
        File[] contents = startDir.listFiles();

        for (int i = 0; i < contents.length; i++){
            //add item to tree.
          //fun(contents[i].getAbsolutePath());
                                        if (contents[i].isFile())
                                        {
                                                String ext =
com.google.common.io.Files.getFileExtension(contents[i].getAbsolutePath());
                                                //System.out.print(ext);
            if(ext.equals("nxml"))
                                                func(contents[i].getAbsolutePath());
                                        }
          else if (! contents[i].isFile()){
                    recurse(contents[i]);
            }
        }
    }
    public static void func(String absolutePath) {
                    // TODO Auto-generated method stub
                    try {
                            File fXmlFile = new File(absolutePath);
```

```
                        File parent= fXmlFile.getParentFile();
                        File source = new File("C:/Users/Arvind/Desktop/mod");
                        FileUtils.copyDirectory(source, parent);
                        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
                        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                        Document doc = dBuilder.parse(fXmlFile);
                        String newPath=absolutePath.replace("medline", "med");
                         Path newFile = Paths.get(newPath);
                         Files.createDirectories(newFile.getParent());
                         if(!newFile.toFile().exists())
                         Files.createFile(newFile);
                         PrintWriter out = new PrintWriter(newFile.toString());
                        //optional, but recommended
                        //read this - http://stackoverflow.com/questions/13786607/normalization-in-dom-parsing-with-java-how-does-it-work
                        doc.getDocumentElement().normalize();
                        if(!(doc.getElementsByTagName("journal-title").getLength()==0))
                                out.println(doc.getElementsByTagName("journal-title").item(0).getTextContent());
                        if(!(doc.getElementsByTagName("article-title").getLength()==0))
                                out.println(doc.getElementsByTagName("article-title").item(0).getTextContent());
                        if(!(doc.getElementsByTagName("abstract").getLength()==0))
                    out.println(doc.getElementsByTagName("abstract").item(0).getTextContent());
                        if(!(doc.getElementsByTagName("body").getLength()==0))
                        out.println(doc.getElementsByTagName("body").item(0).getTextContent());
                        out.close();
                      } catch (Exception e) {
                        e.printStackTrace();
```

```
                }
        }
/**
* Directory is valid if it exists, does not represent a file, and can be read.
*/
public void validateDirectory (File aDirectory) throws FileNotFoundException {
if (aDirectory == null) {
 throw new IllegalArgumentException("Directory should not be null.");
}
if (!aDirectory.exists()) {
 throw new FileNotFoundException("Directory does not exist: " + aDirectory);
}
if (!aDirectory.isDirectory()) {
 throw new IllegalArgumentException("Is not a directory: " + aDirectory);
}
if (!aDirectory.canRead()) {
 throw new IllegalArgumentException("Directory cannot be read: " + aDirectory);
}
}
}
```

Spectral clustering using N-Cut code

```
function [V1,D1]=ncut(W);
% [V1,D1]=ncut(W,nv);
%
% solve generalized eigenproblem Wy={\mu}My
%
% optional third argument specifies minimum allowed weight (e.g. 0.01)


m=sum(W,1);
N=length(m);
M=sparse(1:N,1:N,m);
B=inv(sqrt(M));
```

```
% solve generalized eigensystem
OPTIONS.tol=1e-4;
OPTIONS.maxit=20;
%OPTIONS.disp=0;
C=B*W*B;
format long
[V1,D1]=eig(C);

D1=1-D1;
V1=B*V1;

% sort the eigenvalues
[dvalues,dindex]=sort(diag(D1));

V1=V1(:,dindex);
D1=diag(dvalues);
format
```

Rand index and accuracy calculation code:

```
function [Acc,rand_index,match]=AccMeasure(T,idx)
%Measure percentage of Accuracy and the Rand index of clustering results
% The number of class must equal to the number cluster

%Output
% Acc = Accuracy of clustering results
% rand_index = Rand's Index,  measure an agreement of the clustering results
% match = 2xk mxtrix which are the best match of the Target and clustering
results

%Input
% T = 1xn target index
% idx =1xn matrix of the clustering results

% EX:
% X=[randn(200,2);randn(200,2)+6,;[randn(200,1)+12,randn(200,1)]];
T=[ones(200,1);ones(200,1).*2;ones(200,1).*3];
% idx=kmeans(X,3,'emptyaction','singleton','Replicates',5);
%  [Acc,rand_index,match]=Acc_measure(T,idx)

k=max(T);
n=length(T);
for i=1:k
    temp=find(T==i);
    a{i}=temp; %#ok<AGROW>
end

b1=[];
t1=zeros(1,k);
for i=1:k
    tt1=find(idx==i);
    for j=1:k
       t1(j)=sum(ismember(tt1,a{j}));
```

```
    end
    b1=[b1;t1]; %#ok<AGROW>
end
    Members=zeros(1,k);

P = perms((1:k));
    Acc1=0;
for pi=1:size(P,1)
    for ki=1:k
        Members(ki)=b1(P(pi,ki),ki);
    end
    if sum(Members)>Acc1
        match=P(pi,:);
        Acc1=sum(Members);
    end
end

rand_ss1=0;
rand_dd1=0;
for xi=1:n-1
    for xj=xi+1:n
        rand_ss1=rand_ss1+((idx(xi)==idx(xj))&&(T(xi)==T(xj)));
        rand_dd1=rand_dd1+((idx(xi)~=idx(xj))&&(T(xi)~=T(xj)));
    end
end
rand_index=200*(rand_ss1+rand_dd1)/(n*(n-1));
Acc=Acc1/n*100;
match=[1:k;match];
```

Matlab Diary – Non Negative Matrix Factorization and Kmeans example for 6 documents

>> tmg_gui

Creating directory for any text results.

[TEXT_RESULTS] directory has not been used. Directory is going to be deleted

==========

Applying TMG for file/directory C:\Users\Arvind\Desktop\New folder...

==========

Using delimiter: emptyline

Line Delimiter: Yes

No stoplist used...

No stemming is used...

Update step: 10000

Minimum term length: 3

Maximum term length: 30

Minimum local frequency: 1

Maximum local frequency: Inf

Minimum global frequency: 1

Maximum global frequency: Inf

Using (tfx) term-weighting scheme

==========

Parsing documents...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Cal_J_Emerg_Med\Cal_J_Emerg_Med_2007_Feb_8(1)_15-21.txt...

==========

Parsing document 1...

Number of terms: 3908...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Cancer_Chemother_Pharmacol\Cancer_Chemother_Pharmacol_2008_Jul_6_62(2)_321-329.txt...

==========

Parsing document 2...

Number of terms: 4748...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Cancer_Med\Cancer_Med_2012_Aug_7_1(1)_47-58.txt...

==========

Parsing document 3...

Number of terms: 6492...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Case_Rep_Gastroenterol\Case_Rep_Gastroenterol_2007_Aug_7_1(1)_38-47.txt...

==========

Parsing document 4...

Number of terms: 2509...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Chest\Chest_2013_Jul_25_144(1)_284-305.txt...

==========

Parsing document 5...

Number of terms: 12777...

==========

Parsing file C:/Users\Arvind\Desktop\New folder\Childs_Nerv_Syst\Childs_Nerv_Syst_2010_Aug_16_26(8)_1009-1019.txt...

==========

Parsing document 6...

Number of terms: 4931...

Using (tfx) term-weighting scheme

==========

Results:

==========

Number of documents = 6

Number of terms = 7084

Average number of terms per document (before the normalization) = 5894.17

Average number of indexing terms per document = 4682.33

Sparsity = 20.9721%

Estimated time for parsing and converting the files: 0.049965 seconds

Estimated time for constructing tdm and the other workspace parts: 0.412901 seconds

Removed 281 terms using the term-length thresholds...

Removed 0 terms using the global thresholds...

Removed 0 elements using the local thresholds...

Removed 0 empty terms...

Removed 0 empty documents...

==========

==========

WARNING!

Save the update_struct output argument in order to update your

collection...

==========

```
>> [I,H]=nnmf(A,6);
>> full(H)

ans =
```

0 0 0.0000 0 1.0000 0.0000

0 0 1.0000 0.0000 0 0.0000

0 0 0.0000 0 0.0000 1.0000

1.0000 0 0.0000 0.0000 0 0.0000

0.0000 1.0000 0.0000 0.0000 0 0.0000

0.0000 0 0.0000 1.0000 0 0.0000

Matlab Diary – Kmeans

```
kmeans(A',6)


ans =


     5
     3
     6
     2
     4
     1
```