**Data Analytics II:**

**Assignment #5**

1. Import the Bag of Words datasets from the UCI repository
   1. Combine all of the datasets into a common corpus
   2. Create a word-document matrix from this corpus
   3. Using SKLearn, find the truncated singular value decomposition of this matrix, retaining the first 100 dimensions
      1. Are these dimensions interpretable?

Yes

* + 1. What does dimension 1 represent

bush

* + 1. What do the top 10 dimensions represent?

Bush, Kerry, november, poll, democratic, house, war, general, Iraq, campaign

* 1. Now, retain dimensions 2-101
     1. Are these dimensions interpretable?

Yes

* + 1. What does dimension 1 represent

Kerry

* + 1. What do the top 10 dimensions represent?

Kerry, november, poll, democratic, house, war, general, Iraq, campaign, republicans

* 1. Determine the centroid of each corpus from the Bag of Words dataset in LSA space.

﻿component1 0.014204

component2 0.010195

component3 0.007853

component4 0.005756

component5 0.005572

component6 0.003674

component7 0.006446

component8 0.004936

component9 0.003939

component10 -0.002763

component11 0.005273

component12 0.003916

component13 0.004122

component14 0.002364

component15 0.004346

component16 0.003554

component17 0.002974

component18 0.003859

component19 0.004820

component20 0.003278

component21 0.002035

component22 0.002494

component23 0.003301

component24 0.003016

component25 0.002450

component26 0.002345

component27 0.002691

component28 0.002767

component29 0.000596

component30 0.003505

………………………….

component72 -0.001099

component73 0.001417

component74 0.001092

component75 0.001581

component76 0.000194

component77 0.000145

component78 0.001574

component79 -0.001079

component80 0.001971

component81 -0.002915

component82 0.001733

component83 0.001883

component84 0.002650

component85 0.001651

component86 0.002636

component87 0.000900

component88 -0.002087

component89 -0.000415

component90 -0.000573

component91 0.004686

component92 -0.001234

component93 -0.000717

component94 0.004123

component95 0.001416

component96 0.002402

component97 0.000331

component98 0.000061

component99 0.002200

component100 0.001189

component101 -0.000371

* 1. Determine the average cosine similarity between documents within in each corpus. Next, determine the average cosine similarity between documents across corpora.

Average cosine is 0.011496

* + 1. Does LSA work well as a tool for clustering corpora?

No

1. Apply TF-IDF weighting to the common corpus
   1. Do steps 1b-1f using this TF-IDF weighted corpus. Do the results change? If so, how and why?

Step b doesn’t change.

For step c. After applying TF-IDF, the 1-10 dimensions still represent Bush, Kerry, november, poll, democratic, house, war, general, Iraq, campaign,

For step d, the 2-11 dimensions now represent Kerry, november, poll, democratic, house, war, general, Iraq, campaign, senate. Because tf-idf use a different weighting technique and punish the words that appear more often and scale up the rare ones. After the weighting, senate is more important than republicans.

For step e, the new result is:

﻿component1 0.014599

component2 0.010062

component3 0.007446

component4 0.006071

component5 0.005296

component6 0.004869

component7 0.006173

component8 0.005353

component9 0.005363

component10 0.001209

component11 0.006684

component12 0.002500

component13 0.003999

component14 0.003227

component15 0.003701

component16 0.003328

component17 0.002907

component18 0.005700

component19 0.003388

component20 0.002073

component21 0.004250

component22 0.003360

component23 0.003477

component24 0.004421

component25 -0.002821

component26 0.002802

component27 -0.000637

component28 0.005010

component29 0.000523

component30 0.000906

component72 0.000160

component73 0.000669

component74 0.000693

component75 0.001867

component76 0.000172

component77 0.002110

component78 0.002356

component79 0.001188

component80 -0.000644

component81 0.000498

component82 0.002626

component83 -0.000442

component84 0.000075

component85 -0.000444

component86 -0.000922

component87 -0.002213

component88 0.000450

component89 0.001191

component90 0.003591

component91 0.001304

component92 -0.002735

component93 -0.000320

component94 -0.000790

component95 0.000778

component96 -0.000776

component97 0.001182

component98 0.000596

component99 -0.001768

component100 0.000194

component101 0.002614

We can see that the centroid shifted. Because tfidf uses a different weighting technique

For step f, the average cosine is now 0.008653, and it decreased. It’s also because tfidf is using different weighting techniques

1. Do the sample PCA assignment at <http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html#introduction>

In jupyter Notebook

1. Do the sample PCA demo at <https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html>

In Jupyter Notebook

1. In each of these cases, the total number of principal components was given to you as part of the assignment. Now, your task is to determine how many principal components to retain. Import the data from <https://github.com/rupakc/UCI-Data-Analysis/tree/master/Boston%20Housing%20Dataset/Boston%20Housing>
   1. Generate the covariance matrix of this dataset (don’t forget to normalize and center your data!)
      1. Which variables strongly covary?

We can see the CRIM strongly covary with RAD and TAX

ZN strongly covary with INDUS, NOX, AGE and DIS

INDUS strongly covary with ZN, NOX, AGE, DIS, RAD, TAX and LSTAT

CHAS strongly covary with none of the other variables

NOX strongly covary with ZN, INDUS, AGE, DIS, RAD, TAX, LSTAT

RM strongly covary with LSTAT

AGE strongly covary with ZN, INDUS, NOX, DIS, TAX and LSTAT

DIS strongly covary with ZN, INDUS, NOX, AGE, TAX

RAD strongly covary with CRIM, INDUS, NOX, TAX

TAX strongly covary with CRIM, INDUS, NOX, AGE, DIS, RAD, LSTAT

PTRATIO strongly covary with none of the other variables

B strongly covary with none of the other variables

LSTAT strongly covary with INDUS, NOX, RM, AGE, TAX

* 1. Calculate the eigenvalues of the covariance matrix

Eigenvalues

[6.1389812 1.43611329 1.2450773 0.85927328 0.83646904 0.65870897

0.5364162 0.39688167 0.06363502 0.27749173 0.16963823 0.18638271

0.22067394]

* + 1. Generate a scree plot of the eigenvalues

In Jupyter Notebook

* 1. Using the screeplot, perform a Principal Component Analysis of the data
     1. Technique #1: Choose the number of components based on how many eigenvalues are greater than 1.0. How much variance does this approach explain?

Based on the eigenvalues list, we can see that the first eigenvalue is 6.1389812, the second one is 1.43611329, the third one is 1.2450773, the fourth one is 0.8592738. All others are smaller than the fourth one. Therefore, if we only choose the number of components based on how many eigenvalues are greater than 1 we would choose the first 3 eigenvalues. The variance explained by this approach is 67.7133894

* + 1. Technique #2: After examining the screeplot, identify the “knee in the curve” – choose the number of components at this location. How much variance does this approach explain?

I think the “knee in the curve” in the screeplot is the 9th eigenvalue. If we choose the first 9 eigenvalues, the variance explained is 95.08411979

* + 1. What is the advantage of Technique #1 over Technique #2 and vice versa?

The advantage of Technique #1 over Technique #2 is that it’s clearer how many/what eigenvalues you should keep. Because the criteria are clearer. The “knee in the curve” in screeplot can be different depending on a person’s view.

The advantage of Technique #2 over Technique #1 is that combined with the knowledge of data, it can lead to a less overfit (or in this specific case, underfit) model, since Kaiser’s criterion is subject to noise

* 1. Examine the factors resulting from your preferred PCA. How would you explain these factors?

I choose Technique 1, and the factors resulting from it are :

Matrix W:

[[-0.2509514 0.31525237 -0.24656649]

[ 0.25631454 0.3233129 -0.29585782]

[-0.34667207 -0.11249291 0.01594592]

[-0.00504243 -0.45482914 -0.28978082]

[-0.34285231 -0.21911553 -0.12096411]

[ 0.18924257 -0.14933154 -0.59396117]

[-0.3136706 -0.31197778 0.01767481]

[ 0.32154387 0.34907 0.04973627]

[-0.31979277 0.27152094 -0.28725483]

[-0.33846915 0.23945365 -0.22074447]

[-0.20494226 0.30589695 0.32344627]

[ 0.20297261 -0.23855944 0.3001459 ]

[-0.30975984 0.07432203 0.26700025]]

Interpretation:

Component1=-0.2509514\*CRIM+0.25631454\*ZN-0.34667207\*INDUS-0.00504243\*CHAS-0.34285231\*NOX+0.18924257\*RM-0.3136706\*AGE+0.32154387\*DIS-0.31979277\*RAD-0.33846915\*TAX-0.20494226\*PTRATIO+0.20297261\*B-0.30975984\*LSTAT

Component2=0.31525237\*CRIM+0.3233129\*ZN-0.11249291\*INDUS-0.45482914\*CHAS-0.21911553\*NOX-0.14933154\*RM-0.31197778\*AGE+0.34907\*DIS+0.27152094\*RAD+0.23945365\*TAX+0.30589695\*PTRATIO-0.23855944\*B+0.07432203\*LSTAT

Component3=-0.24656649\*CRIM-0.29585782\*ZN+0.01594592\*INDUS-0.28978082\*CHAS-0.12096411\*NOX-0.59396117\*RM+0.01767481\*AGE+0.04973627\*DIS-0.28725483\*RAD-0.22074447\*TAX+0.32344627\*PTRATIO+0.3001459\*B+0.26700025\*LSTAT

1. Extra credit: Collect data from a source of your choice and run a PCA. Provide a one-page writeup explaining the meaning of the resulting decomposition.

Do the sample k-means clustering demo at: <https://scikit-learn.org/0.18/auto_examples/text/document_clustering.html>

This demo first uses LSA to carry out dimensionality reduction before clustering. What is the difference between the results obtained using just LSA and the results obtained using LSA + k-means?

When you only use LSA the result obtained is:

﻿Explained variance of the SVD step: 24%

When you use LSA+k-means, the result obtained is:

﻿Homogeneity: 0.496

Completeness: 0.494

V-measure: 0.495

Adjusted Rand-Index: 0.444

Silhouette Coefficient: 0.040

Top terms per cluster:

Cluster 0: space nasa access digex gov shuttle pat com moon launch

Cluster 1: com keith article henry sgi caltech posting livesey toronto nntp

Cluster 2: god com people jesus sandvik bible religion don christian think

Cluster 3: graphics image thanks files university 3d com file help gif

Do the clustering again, only without performing LSA first. Are the results the same? If not, why not? Which result gives more meaningful clusters and why?

No. Without performing LSA first, the result obtained is:

﻿Homogeneity: 0.247

Completeness: 0.372

V-measure: 0.297

Adjusted Rand-Index: 0.168

Silhouette Coefficient: 0.005

Top terms per cluster:

Cluster 0: access digex pat net prb hst com express online communications

Cluster 1: keith sgi livesey morality objective caltech com solntze wpd jon

Cluster 2: space com university graphics posting host nntp article like nasa

Cluster 3: god sandvik jesus com kent people bible apple christian believe

We can see that Homogeneity, Completeness, V-measure, Adjusted Rand-Index and Silhouette Coefficient all decreased.

Because K-means clustering will suffer from the curse of dimensionality when there are too many features. Therefore, the results given back after using LSA for dimension reduction are more meaningful.

Load the iris dataset using SKLearn and perform a k-means clustering on this dataset.

Generate a screeplot (horizontal axis is number of clusters and vertical axis is sum of squared distance between each point and the cluster centroid). Use the screeplot to choose the appropriate number of clusters and interpret the results.

I chose 4 as the number of clusters because the fluctuation of the line is very small after the 4th point, which means that 4 is probably the right number for clusters

The above technique may be prone to overfitting. Use 10-fold cross-validation to select the appropriate number of clusters and interpret the results.

Because the goal is to minimize the within cluster sum of squared distance between the testing set and its closest centroid, therefore I chose 3 as the number of clusters for the SSD calculated (after punishing the number of clusters) for 3 clusters is the smallest

Using the US Census dataset from the UCI repository: <https://archive.ics.uci.edu/ml/datasets/US+Census+Data+%281990%29>

Generate a screeplot (horizontal axis is number of clusters and vertical axis is sum of squared distance between each point and the cluster centroid). Use the screeplot to choose the appropriate number of clusters and interpret the results.

I chose 4 as the number of clusters, because after 6 the line starts to get pretty flat.

The above technique may be prone to overfitting. Use 10-fold cross-validation to select the appropriate number of clusters and interpret the results.

According to the results given back by the 10-fold cross validation, it’s better to choose 1 or 2 as the number of clustering.

Extra credit: Collect Twitter data using a measure of your choice and perform a k-means clustering. Use TF-IDF as appropriate.

Using the techniques identified above, determine the appropriate number of clusters.

I cannot determine the appropriate number of clusters because it’s still going down steeply when the number of clusters is 50

One heuristic for determining the number of clusters for text data is m\*n/t, where m is the number of terms, n is the number of documents, and t is the number of nonzero entries in the term-document matrix. How does this heuristic perform compared to the screeplot technique?

Better. At least I can know what the number of clusters is supposed to be

Do the sample LDA demo at: <https://scikit-learn.org/stable/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html> (you can, but need not, skip the NMF part)

Carry out the same analysis with 5 topics, and with 50 topics.

Please check Q11\_5topics.py and Q\_11\_50topics.py

Carry out the same analysis with hyperparameter values  = 50/T and  = W/2000

Please check Q11\_Partb.py

Load the NYSK dataset from <https://archive.ics.uci.edu/ml/datasets/NYSK>

Fit a topic model to the data using LDA. Choose the number of topics using Perplexity as a criterion.

Choose a subset of the most informative topics and plot them over time. What do these topics indicate?

Extra credit: Download data from Twitter and perform a topics over time analysis using LDA