Lab 7: Dimensionality Reduction and Unsupervised Machine Learning

In the first half of the class, we've been focusing on data cleaning and exploratory data analysis (EDA), first with visualization and then with statistical hypothesis testing. You've primarily been using data sets with only a handful of variables. You could analyze each variable with visualizations and statistics to find relationships. High dimensional data sets, however, have too many variables for you to analyze each variable individually. We need to turn to more sophisticated techniques such dimensionality reduction and clustering.

In this lab, you are going to analyze 63,542 emails. You will convert the raw text into a feature matrix using a "bag of words" model. Each column of the feature matrix corresponds to one word, each row corresponds to one email, and the entry stores the number of times that word was found in that email. You will perform dimensionality reduction using the Truncated SVD method, cluster the emails, and compare the "inherent" structure to the given class labels.

1 Load the data

1. Extract the provided email_json.zip file. It should create a directory called "email_json". Each email is stored as a separate JSON document with a name in the format "message_XXXXX.json".
2. Remember when I introduced the Titanic data set in the first lab and I asked the class to import the file by hand? That practice will pay off here. Write some code to find and load all of the JSON documents. You should have a list of dicts when done (Hint: Review the built-in Python json library).
3. Convert the list of dicts into a Pandas DataFrame. The DataFrame should have 5 columns and 63,452 rows (Hint: use the DataFrame.from_records() or DataFrame.from_dict() functions).
4. Answer the following question in your writeup:
   (a) What are the column names and their types?

2 Extract Features

1. By themselves, strings of the message bodies are not amenable to analysis. We need to convert them to a feature matrix.

2. Use Scikit Learn's CountVectorizer class with the binary=True flag to create a feature matrix from the message bodies. When doing this set min_df=10 to exclude any words that do not appear in at least 10 emails. The "bodies" column of the DataFrame can be used as a list of strings and passed directly into the fit_transform() method of the CountVectorizer.

3. Answer the following question in your writeup:

   (a) How many rows and columns does the feature matrix have? How many nonzero entries are in the matrix?

   (b) Calculate and report the sparsity ratio (100 * number of nonzero entries divided by maximum possible entries).

   (c) What do the rows and columns of this matrix means? What is recorded in each entry of the matrix?

   (d) The vectorizer returns a sparse matrix in compressed row format (CSR). Assume that the sparse matrix uses one 32-bit (4-byte) floating point number and one 32-bit (4-byte) integer for each nonzero entry and one 32-bit (4-byte) integer for each row. Calculate the memory usage for the sparse matrix.

3 Dimensionality Reduction/Visualization

1. This matrix has too much information to be directly useful. Use the fit_transform() function of Scikit Learn's TruncatedSVD class to transform the original feature matrix into a new feature matrix with 10 columns (variables or components). Scikit Learn's TruncatedSVD method is similar to its PCA method, but it works with sparse matrices.

2. Plot the explained variance ratios of the components (Hint: use the explained_variance_ratio_ property of the TruncatedSVD class).

3. Create a scatter plot using the two components with the highest explained variance ratios (Hint: plt.scatter(proj_matrix[:, i], proj_matrix[:, j])).

4. Create a second scatter plot using the same two components. This time, color the points based on the category column of the DataFrame. All spam messages should be one color; all ham messages should be a second color.

5. Answer the following question in your writeup:

   (a) Which two components have the highest explained variance ratios?

4 Clustering

1. To perform further analysis, we want to cluster the emails. By clustering the emails, each message will be assigned a cluster id (e.g., 0, 1, 2, etc.). There are numerous clustering algorithms, each with strengths and weaknesses. We cannot cover them all

in lecture and so begin this section by reviewing the clustering algorithms available in Scikit Learn. Choose a clustering algorithm that you think would be appropriate for this data set.

2. Cluster the samples using the two SVD components that you determined to have the highest explained ratio AFTER performing any necessary operations to transform them. The clustering algorithm should return a 1D numpy array of cluster labels (e.g., 0, 1, 2, etc.) for each point.

3. As in 3.4., create a scatter plot of the two SVD components that you determined to have the highest explained ratio. The clustering algorithm should label the points so that all points in the same cluster have the same cluster id (With two clusters, there should only be two cluster ids). Color the points according to their cluster labels. If the clustering does not look correct, try adjusting the parameters of the clustering algorithm you chose or choose a different algorithm.

4. Calculate a confusion matrix for the ham / spam labels versus the cluster labels.

5. Answer the following question in your writeup:

   (a) Assess the cluster assignments. Visually, the data should form two distinct groups or clusters. What sort of structure do you observe? Do you think that clustering will allow you to recover the label?

   (b) Why did you choose the clustering algorithm that you did?

   (c) What sort of transformations do you need to apply to your data prior to clustering? Why are they necessary?

   (d) Compare the ham/spam labels to the cluster labels using the confusion matrix you generated. Are spam messages in both clusters or a single cluster? Are all of the messages in the clusters with spam labeled as spam? Was your clustering successful?

5 Calculating Document Frequencies of Words

1. From part V, we now have clusters, but we don't know why specific emails were clustered together. Let's try to see if we can ascribe meaning to the clusters. Create a separate matrix for each cluster containing the rows for the points in that cluster.

2. Convert the matrices to CSC format. We will be accessing the data column-wise. Column indexing is significantly faster for the CSC format than the CSR format (Hint: Review the scipy.sparse module).

3. Calculate the document frequency of each word for each cluster. The document frequency is the number of documents that contain each word. Since the feature matrix is binary, we can simply sum across the columns (Note: The resulting matrices should have the same shapes with one entry for each word in the vocabulary).

6 Find Enriched Words with Statistical Testing

1. We are going to use a Binomial test to determine if the number of occurrences of a given word in a given cluster is higher than what would be expected from the other cluster (you may find scipy.stats.binom_test useful here). Here is the code for calculating the expected probability of a specific word based on cluster 1 and using it to test if the word is enriched in cluster 0:

```
cluster_1_expected_prob = doc_freq_cluster_1 / num_emails_cluster_1
pvalue = stats.binom_test(doc_freq_cluster_0, num_emails_cluster_0,
                          cluster_1_expected_prob, alternative="greater")
```

   The null hypothesis that the relative document frequencies of the observed cluster are less or equal to those of the tested. The alternative hypothesis is that the document frequency is higher in cluster 0 than cluster 1.

2. Try testing if the words "works" and "love" are enriched in cluster 0. For "works", I got an expected probability of ≈ 0.109 for cluster 1 and observed probability of ≈ 0.041 for cluster 0. The Binomial test applied to this information returns a p-value of 0.999, indicating that we fail to reject the hypothesis. That is, the observed frequency for cluster 0 is NOT greater than the frequency for cluster 1. For "love", I got an expected probability of ≈ 0.004 for cluster 1 and observed probability of ≈ 0.035 for cluster 0. The Binomial test applied to this information should return a p-value of 0.0, indicating that we should reject the null hypothesis. That is, the observed frequency for cluster 0 is greater than the frequency for cluster 1. NOTE: your results will depend on the specific clustering algorithm you apply.

3. Wrap your code for 6.1. in a loop to find enriched words for cluster 0. Calculate the p-value for every word. If the p-value < 0.05, add a tuple of (pvalue, word, cluster 0 document frequency to a list (Hint: Iterate over the vocabulary_ dict to get each word and its index). When you do this filter out any words that contain non-alphabetic characters (Hint: Use the isalpha() method of strings).

4. Sort the words in ascending order by their p-values and print out the first 200 words.

5. Repeat with the clusters reversed so you can find words enriched in cluster 1.

6. Answer the following question in your writeup:

   (a) Skim through the top 200 words for each cluster. Can you identify any patterns for either of the clusters?

7 Submission Instructions

- Write up the answers to the questions in a short word document; aim for around 2 pages of text and include all graphics generated. Add footnotes identifying which sentence addresses which questions. Write in complete sentences organized into paragraphs – your goal is to explain what you've done and what you've learned to your audience (me!). Accordingly, you may seek to emulate some of the sections of the whale paper describing their data. Include the appropriate plots you've generated as mentioned above. Convert this to pdf and submit it. Submit your .ipynb file as well.
- The grading rubric for this assignment will be available in Canvas.
- NO OTHER SUBMISSION TYPES WILL BE ACCEPTED.
- **Late policy**: 5% of total points deducted per day for three days – after that no submissions allowed.