ECE219 Large-Scale Data Mining: Models and Algorithms **(Due: Fri, 01/24/20)**

Project 1: Classification Analysis on Textual Data

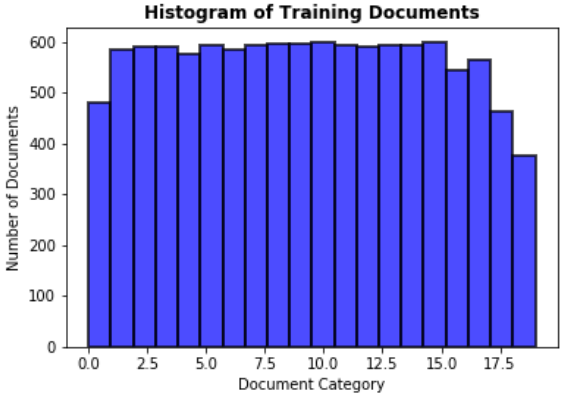
Instructor: Vwani Roychowdhury Zhaoxi Yu(104880239) & Yuhao Yin(104880239)

1 Getting Familiar with the Dataset

The dataset that we are working with in this project, is called "20 Newsgroups", a collection of approximately 20,000 documents partitioned into 20 different topics, including Computer Technology, Recreational Activity, Science, etc. Based on the textual content, we are expected to identify the document topics, which is frequently referred to as a statistical classification task.

In a classification problem, one should make sure to properly handle any imbalance in the relative sizes of the data sets corresponding to different classes. To do so, one can either modify the penalty function (i.e., assign more weight to errors from minority classes), or alternatively, down-sample the majority classes, to have the same number of instances as minority classes.

QUESTION 1. To get started, plot a histogram of the number of training documents for each of the 20 categories to check if they are evenly distributed.



From the histogram, we can safely conclude that the "20 Newsgroups" dataset is already balanced (especially for the categories we'll mainly work on).

2 Binary Classification

We start with binary classification, i.e., the main task here would be to classify the documents into two classes "Computer Technology" vs "Recreational Activity". Since we originally have 8 different topic categories, we first need to convert them to binary target, 0 representing Computer Technology and 1 representing Recreational Activity, following the projection rule summarized in the following table.

Computer Technology	Recreational Activity
comp.graphics	rec.autos
comp.os.ms-windows.misc	rec.motorcycles
comp.sys.ibm.pc.hardware	rec.sport.baseball
comp.sys.mac.hardware	rec.sport.hockey

For example, the first 20 training data before and after this transformation look like,

$$\textit{Original Target}: [6, 7, 4, 2, 1, 3, 0, 7, 5, 3, 0, 5, 5, 5, 3, 1, 3, 0, 0, 2]$$
$$\textit{Binary Target}: [1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0]$$

2.1 Feature Extraction

The primary step in classifying a corpus of text is choosing a proper document representation. A good representation should retain enough information that enable us to perform the classification, yet in the meantime, be concise to avoid computational intractability and over-fitting.

In this project, we choose the "Bag of Words" representation, where a document is represented as a histogram of term frequencies, and normalize the count of vocabulary words using the "Term Frequency-Inverse Document Frequency(TF-IDF)" metric.

QUESTION 2. Use the following specs to extract features from the textual data:

- Use the `"english"` stopwords of the `CountVectorizer`
- Exclude terms that are numbers (e.g. "123", "-45", "6.7" etc.)
- Perform lemmatization with `nltk.wordnet.WordNetLemmatizer` and `pos_tag`
- Use `min_df=3`

Report the shape of the TF-IDF matrices of the train and test subsets respectively.

Following the instructions provided, we extract term frequency features from the raw textual data, drop terms that are very rare, by setting `min_df=3` in the `CountVectorizer`, perform lemmatization with `nltk.wordnet.WordNetLemmatizer` and `pos_tag`.

It's worth noting that we exclude not only **numbers**, but also **any term that is not alphabetic**, with the assumptions that non-letter terms do not provide actual information concerning the topic class.

The shape of the TF-IDF matrices of the train and test subsets are reported as follows,

Train	(4732, 12609)
Test	(3150, 12609)

Table 1: Shape of Document-term TF-IDF Matrices

2.2 Dimensionality Reduction

The dimensionality of TF-IDF vectors above ranges in the order of thousands. Since the document-term TF-IDF matrix is sparse and low-rank, we can transform the features into a lower dimensional space in order to circumvent the curse of dimensionality.

In this project, we use two dimensionality reduction methods: **Latent Semantic Indexing (LSI)** and **Non-negative Matrix Factorization (NMF)**, both of which minimize mean squared residual between the original data and a reconstruction from its low-dimensional approximation.

QUESTION 3. Reduce the dimensionality of the data using the method above

- Apply LSI to the TF-IDF matrix corresponding to the 8 categories with $k = 50$; so each document is mapped to a 50-dimensional vector.
- Also reduce dimensionality through NMF ($k = 50$) and compare with LSI: Which one is larger, the $\|X - WH\|_F^2$ in NMF or the $\|X - U_k \Sigma_k V_k^\top\|_F^2$ in LSI? Why is the case?

We employ functions `TruncatedSVD` and `NMF` both from `sklearn.decomposition`, to perform LSI and NMF respectively on the document-term IF-IDF matrices. After that, we further compute the squared frobenius norm of difference matrices between the original data and a reconstruction from their respective low-dimensional approximation. The results are reported as below,

Squared Frobenius Norm for LSI	4120.0215
Squared Frobenius Norm for NMF	4155.7582

Table 2: Squared Frobenius Norm for LSI and NMF

We conclude that squared frobenius norm for NMF is larger, since reconstruction from largest k eigenvalues as well as their corresponding left and right eigenvectors mathematically guarantees to have the smallest mean squared residuals compared to the original data.

2.3 Classification Algorithms

In this part, we use the dimension-reduced training data from LSI to train various types of classifiers, and evaluate the trained classifiers on test data with different classification measures, including **accuracy, precision, recall, F-1 score** and **ROC curve**.

2.3.1 SVM

Linear Support Vector Machines have been proved efficient when dealing with sparse high dimensional datasets, including textual data. They have been shown to have good generalization accuracy, while having low computational complexity.

The learning process of the parameter w and b involves solving the following optimization problem:

$$\begin{aligned} \min_{w,b} \ & \frac{1}{2}\|w\|_2^2 + \gamma \sum_{i=1}^{n} \xi_i \\ s.t. \ & y_i(w^\top x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad \forall i \in \{1, \cdots, n\} \end{aligned}$$

where x_i is the ith data point, and $y_i \in \{0, 1\}$ is the class label of it. Note that the tradeoff parameter γ controls relative importance of minimizing the loss function on the training data versus maximizing the margin between the two classes.

QUESTION 4. Hard margin and soft margin linear SVMs:

- Train two linear SVMs and compare:
  - Train one SVM with $\gamma = 1000$ (hard margin), another with $\gamma = 0.0001$ (soft margin).
  - Plot the **ROC curve**, report the **confusion matrix** and calculate the **accuracy**, **recall**, **precision** and **F-1 score** of both SVM classifier. Which one performs better?
  - What happens for the soft margin SVM? Why is the case?
    - Does the ROC curve of the soft margin SVM look good? Does this conflict with other metrics?
- Use cross-validation to choose γ (use average validation accuracy to compare):
  Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10^k | -3 \leq k \leq 3, k \in \mathbb{Z}\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this best SVM.