EE219 Project 1

Classification Analysis on Textual Data

Winter 2018

Introduction:

Statistical classification refers to the task of identifying a category, from a predefined set, to which a data point belongs, given a training data set with known category memberships. Classification differs from the task of clustering, which concerns grouping data points with no predefined category memberships, where the objective is to seek inherent structures in data with respect to suitable measures. Classification turns out as an essential element of data analysis, especially when dealing with a large amount of data. In this project, we look into different methods for classifying textual data.

Dataset and Problem Statement:

In this project, we work with "20 Newsgroups" dataset. It is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, each corresponding to a different topic.

We highly recommend using python as your programming language. The easiest way to load the data is to use the built-in dataset loader for 20 newsgroups from scikit-learn package. As an example, if you want to load only "comp.graphics" category, then you can use the following command:

```
01. from sklearn.datasets import fetch_20newsgroups
02.
03. categories = ['comp.graphics']
04. graphics_train = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, random_state=42)
```

For the purposes of this project we will be working with only 8 of the classes as shown in Table 1. Load the training and testing data for the following 8 subclasses of two major classes 'Computer Technology' and 'Recreational activity'.

Table 1. Subclasses of 'Computer technology' and 'Recreational activity'

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

a) 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. To get started, perform the following to the data of the 8 classes above: **plot a histogram** of the number of training documents per

class to check if they are evenly distributed. Note that the data set is already balanced and so in this case we do not need to balance. But in general, as a data scientist you need to be aware of this issue.

Modeling Text Data and Feature Extraction:

The primary step in classifying a corpus of text is choosing a proper document representation. This representation should not include too much irrelevant information, which may lead to computational intractability and over fitting. One common representation of documents is called “Bag of Words”, where a document is represented as a histogram of term frequencies, or other statistics of the terms, within a fixed vocabulary. As such, a corpus of text can be summarized into a term-document matrix whose entries are some statistic of the terms.

First a common sense filtering is done to drop certain words or terms: To avoid unnecessarily large feature vectors (vocabulary size), terms that are too frequent in almost every document, or are very rare, are dropped out of the vocabulary. The same goes with special characters, common stop words (e. g. and, the etc.), In addition, appearances of words that share the same stem in the vocabulary (e. g. goes vs going) are merged into a single term.

Next, one can consider using the normalized count of the vocabulary words in each document to build representation vectors. A popular numerical statistic to capture the importance of a word to a document in a corpus is the Term Frequency-Inverse Document Frequency (TFxIDF) metric. This measure takes into account count of the words in the document, as normalized by a certain function of the frequency of the individual words in the whole corpus. For example, if a corpus is about computer accessories then words such as “computer” “software” “purchase” will be present in almost every document and their frequency is not a distinguishing feature for any document in the corpus. The discriminating words will most likely be those that are specialized terms describing different types of accessories and hence will occur in fewer documents. Thus, a human reading a particular document will usually ignore the contextually dominant words such as “computer”, “software” etc. and give more importance to specific words. This is like when going into a very bright room or looking at a bright object, the human perception system usually applies a saturating function (such as a logarithm or square-root) to the actual input values before passing it on to the neurons. This makes sure that a contextually dominant signal does not overwhelm the decision-making processes in the brain. The TFxIDF functions draw their inspiration from such neuronal systems.

b) Perform the following on the documents of balanced data of 8 classes, to convert them into numerical feature vectors. First **tokenize** each document into words. Then, **excluding the stop words, punctuations**, and using **stemmed** version of words, **create a TFxIDF vector representations**. TFxIDF vector representation of a document is defined as follows:

$$TFxIDF(t,d) = tf(t,d) * idf(t)$$

where $tf(t,d)$ represents the frequency of term t in document d, and inverse document frequency is defined as:

$$idf(t) = \log[\, n \,/\, df(t)] + 1$$

where n is the total number of documents, and $df(t)$ is the document frequency; the document frequency is the number of documents that contain the term t.

Through part i, use two settings for minimum document frequency of vocabulary terms, namely **min_df=2** and **min_df=5**. **Report** the final number of terms you extracted.

As for a list of stop words, you can refer to the list provided in scikit-learn package. You can run the following commands to take a look at this list.

```
01. from sklearn.feature_extraction import text
02. stop_words = text.ENGLISH_STOP_WORDS
```

c) In order to quantify how significant a word is to a class, we can define a measure called TFxICF. It is the same as TFxIDF, except that a class sits in place of a document; that is for a term t and a class c, the measure is computed as

$$TFxICF(t,c) = tf(t,c) * icf(t)$$

where $tf(t,c)$ represents the term frequency in a class c , and inverse class frequency is defined as:

$$icf(t) = \log[\, n_{classes} \,/\, cf(t)] + 1$$

and $n_{classes}$ is the total number of classes, and $cf(t)$ is the class frequency which is the number of classes within which there is at least a document containing the term t.

Find the 10 most significant terms in each of the following classes with respect to TFxICF measure. Note that in this part of your assignment, $n_{classes}$ is 20. (You can use the unbalanced dataset of all 20 classes for this part)

`comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian.`

Feature Selection:

After these operations, the dimensionality of the representation vectors (TFxIDF vectors) ranges in the order of thousands. However, the document-term tfidf matrix is sparse and low-rank. Besides, learning algorithms may perform poorly in high-dimensional data, which is sometimes referred to as “The Curse of Dimensionality”. As a remedy, one can select a subset of the original features, which are more

relevant with respect to certain performance measure, or transform the features into a lower dimensional space.

In this project, we use Latent Semantic Indexing (LSI), a dimensionality reduction transform method, which minimizes mean squared residual between the original data and reconstruction from its low-dimensional approximation. Recall that our data is the term-document tfidf matrix, whose rows correspond to TFxIDF representation of the documents.

The LSI representation is obtained by computing left and right singular vectors corresponding to the largest values of the term-document tfidf matrix. LSI is similar to Principal Component Analysis (PCA), and see the lecture notes for their relationships.

Eigenvectors corresponding to the dominant eigenvalues obtained by LSI correspond to most prevailing combinations of terms in the corpus, which can be thought of as "topics" or "semantic concepts". Thus, the new low dimensional representation is the magnitudes of the projections of the document vectors onto these latent "topics".

Let D denote the $t \times d$ term-document matrix with rank r where each of the d columns represents a document vector of dimension t. The singular value decomposition results in

$$D = U\Sigma V^T,$$

Where Σ is an $r \times r$ diagonal matrix of singular values of D, U is a $t \times r$ matrix of left singular column vectors, and V is a $d \times r$ matrix of right singular vectors. Dropping all but k largest singular values and corresponding singular vectors gives the following truncated approximation

$$D_k = U_k \Sigma_k V_k^T.$$

The approximation is the best rank k approximation of D in the sense of minimizing the sum of squared differences between the entries of D and D_k.

The $t \times k$ matrix U_k can now be used as a projection matrix to map each $t \times 1$ document column vector $\vec{d}$ into a k-dimensional representation

$$\overrightarrow{d_k} = U_k^T \vec{d}.$$

d) Apply LSI to the TFxIDF matrix corresponding to the 8 classes. and pick k=50; so each document is mapped to a 50-dimensional vector. Alternatively, reduce dimensionality through Non-Negative Matrix Factorization (NMF) and compare the results of the parts e-i using both methods.

Learning Algorithms

For the following parts (e-h) of the project, your task would be to classify the documents into two categories ''Computer Technology' vs 'Recreational Activity'. Refer to Table 1 to find 4 subclasses comprising each of the two classes. In other words, you need to combine documents of sub-classes of each class to form the set of documents for each class.

Classification accuracy can be evaluated using different measures called precision, recall, F-score, etc. Precision is the proportion of items placed in the category that are really in the category, and recall is the proportion of items in the category that are actually placed in the category.

Depending on application, the true positive rate (TPR) and the false positive rate (FPR) have different levels of significance. In order to characterize the trade-off between the two quantities we plot the receiver operating characteristic (ROC) curve. For binary classification, the curve is created by plotting the true positive rate against the false positive rate at various threshold settings on the probabilities assigned to each class (let us assume probability p for class 0 and $1-p$ for class 1). In particular, a threshold t is applied to value of p to select between the two classes. The value of threshold t is swept from 0 to 1 to produce a pair of precision and recall for each value of t.

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.

Linear Support Vector Machines aim to learn a vector of feature weights, $\widehat{w}$, and an intercept, b, given the training data set. Once the weights are learned, the label of a data point is determined by thresholding $W^T.\vec{x}+b$ with 0, i.e. $sign(W^T.\vec{x}+b)$. Alternatively, one produce probabilities that the data point belongs to either class, by applying a logistic function instead of hard thresholding, i.e. calculating $\sigma(W^T.\vec{x}+b)$.

The learning process involves solving the following optimization problem:

$$\min \frac{1}{2} \| w \|_2^2 + \gamma \sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } y_i\left(w^T.\vec{x}_i + b\right) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0, \text{ for all } i \in \{1, \dots, n\}.$$

Minimizing the sum of the slack variables corresponds to minimizing the loss function on the training data. On the other hand, minimizing the first term, which is basically a regularization, corresponds to maximizing the margin between the two classes. Note that in the objective function, each slack variable represents the amount of error that the classifier can tolerate for a given data sample. The tradeoff parameter γ controls relative importance of the two components of the objective function. For instance,

when $\gamma \gg 1$, misclassification of individual points is highly penalized, which is called "Hard Margin SVM". In contrast, a "Soft Margin SVM", which is the case when $\gamma \ll 1$, is very lenient towards misclassification of a few individual points as long as most data points are well separated.

e) Use hard margin SVM classifier (SVC) by setting γ to a high value, e.g. 1000, to separate the documents into 'Computer Technology' vs 'Recreational Activity' groups. In your submission, **plot the ROC curve**, **report the confusion matrix** and **calculate** the **accuracy, recall** and **precision** of your classifier. **Repeat the same procedure for soft margin SVC** (set γ to 0.001)

f) Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10^k \big| -3 \leq k \leq 3, k \in Z\}$. Report the confusion matrix and calculate the accuracy, recall and precision of the classifier.

g) Next, we use naïve Bayes algorithm for the same classification task. The algorithm estimates the maximum likelihood probability of a class given a document with feature set $\vec{x}$, using Bayes rule, based upon the assumption that given the class, the features are statistically independent.

Train a multinomial naïve Bayes classifier and plot the ROC curve for different values of the threshold on class probabilities. You should report your ROC curve along with that of the other algorithms. Again, Report the confusion matrix and calculate the accuracy, recall and precision of your classifier.

h) Repeat the same task with the logistic regression classifier, and plot the ROC curve for different values of the threshold on class probabilities. You should report your ROC curve along with that of the other algorithms. Provide the same evaluation measures on this classifier.

i) Now, repeat part (h) by adding a regularization term to the optimization objective. Try both l_1 and l_2 norm regularizations and sweep through different regularization coefficients, ranging from very small ones to large ones. How does the regularization parameter affect the test error? How are the coefficients of the fitted hyperplane affected? Why might one be interested in each type of regularization?

Multiclass Classification:

So far, we have been dealing with classifying the data points into two classes. In this part, we explore multiclass classification techniques through different algorithms.

Some classifiers perform the multiclass classification inherently. As such, Naïve Bayes algorithm finds the class with maximum likelihood given the data, regardless of the number of classes. In fact, the probability of each class label is computed in the usual way, then the class with the highest probability is picked; that is

$$\hat{c} = \arg\min_{c \in C} p(c|\overrightarrow{x}).$$

where c denotes a class to be chosen, and $\hat{c}$ denotes the optimal class.

For SVM, however, one needs to extend the binary classification techniques when there are multiple classes. A natural way to do so is to perform a one versus one classification on all $\binom{|C|}{2}$ pairs of classes, and given a document the class is assigned with the majority vote. In case there is more than one class with the highest vote, the class with the highest total classification confidence levels in the binary classifiers is picked.

An alternative strategy would be to fit one classifier per class, which reduces the number of classifiers to be learnt to $|C|$. For each classifier, the class is fitted against all the other classes. Note that in this case, the unbalanced number of documents in each class should be handled. By learning a single classifier for each class, one can get insights on the interpretation of the classes based on the features.

i) In this part, we aim to learn classifiers on the documents belonging to the classes mentioned in part b; namely

`comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian.`

Perform Naïve Bayes classification and multiclass SVM classification (with both One VS One and One VS the rest methods described above) and report the confusion matrix and calculate the accuracy, recall and precision of your classifiers.

Submission: Please submit a zip file containing your report, and your codes with a readme file on how to run your code to ee219.winter2018@gmail.com. The zip file should be named as "Project1_UID1_UID2_..._UIDn.zip" where UIDx are student ID numbers of the team members. If you had any questions you can send an email to the same address.