CS 422/622 Project 5

Due 12/08 11:59pm

In this project, you will be using Scikit-Learn to test out all the models/algorithms we have learned in this class on a real-world problem. You will use the ACL IMDB dataset provided with the project. This dataset contains movie reviews and sentiment scores (positive and negative). We're going to create a Bag of Words (BOW) approach to generate feature vectors and then apply different models from sklearn to the data. A BOW approach simply treats a document (review) as a bag of words (removing all ordering information). We want to create a vocabulary (V) and then treat each document as a vector that is of size len(V) that consists of word counts. So if you have the document "My name is Emily, what is your name?" this would be represented by a vector of length 6, with 1s in all the positions except for "name" and "is" which would have counts of 2. Then we use this count vector as our feature vector for each review and train a model using these feature vectors.

DO NOT INCLUDE A COPY OF THE DATASET IN YOUR SUBMISSION. Only submit your 2 py files.

Logistics: You must implement everything stated in this project description that is marked with an **implement** tag. Whenever you see the **write-up** tag, that is something that must be addressed in the project README.txt. **You cannot import any packages unless specifically noted by the instructor.** In this project, you may import the following packages:

```
import numpy
import sklearn
import sys
import os
import random
```

Deliverables: Each student should submit a single ZIP file, containing their project code (*.py files) and your writeup (README.txt). You should create a folder called Project5 (exactly like this with the capital P) and put all your code and your writeup in this folder. Then zip this folder up. That way when I extract your folder I can run my tests from outside the directory without having to change anything. Your zip file should be named lastname_firstname_project5.zip. Your code should run without errors in a linux environment. If your code does not run for a particular problem, you will lose 50% on that problem. You should submit only one py file for each problem. If your file does not match the filename provided exactly, you will lose 50% on that problem.

Grading: I have provided you with a test script (test_script.py) you can use to test out your functions. I will be using a similar test script, so if your code works with this script it should work with mine! The output of the test script should look something like this if you have implemented everything correctly. **Each student must work independently. You are to submit your own original work.**

1 Utilities (50 points)

File name: **utilities.py**

Implement: You will implement three functions listed here and detailed below.

```
def generate_vocab(dir, min_count, max_files)
def create_word_vector(fname, vocab)
def load_data(dir, vocab, max_files)
```

Write-Up: Describe your implementation concisely.

`def generate_vocab(dir, min_count, max_files)`

This function will take a starting directory `dir` (e.g. "aclImdb/train"), `min_count` which is the minimum number of times you want to see a word before adding it to a vocabulary. If `min_count=2` then we only want to consider words that have been seen 2 or more times in the dataset as a part of our vocabulary. This

function also takes a parameter `max_files` which is just for implementation purposes. This allows you to do small tests without using the full dataset. The full dataset takes a long time to generate feature vectors, so you may want to use only 200 files to start with. If `max_files= -1` then all files are used. This returns a list or numpy array of the vocabulary. Remember that when using `max_files`, you should be sure to grab an even number of positive and negative samples.

`def create_word_vector(fname, vocab)`
This function takes the vocabulary and a review file and generates a feature vector. `fname` is the filename for a review. Assume that the aclImdb directory is in the same directory as the test_script.py. This returns one feature vector.

`def load_data(dir, vocab, max_files)`
This function loads the data, returning a set of feature vectors and associated labels. It should return two lists/arrays `X, Y`. `max_files` is again for implementation reasons, to allow for smaller tests. If `max_files= -1` then all files are used.

2 ML (50 points)

File name: **ml.py**

You will use sklearn to test many different models on the ACL IMDB data.
Implement: You will implement the functions listed here and detailed below.

```
def dt_train(X,Y)
def kmeans_train(X)
def knn_train(X,Y,K)
def perceptron_train(X,Y)
def nn_train(X,Y, hls)
def pca_train(X,K)
def pca_transform(X,pca)
def svm_train(X,Y,k)
def model_test(X,model)
def compute_F1(Y, Y_hat)
```

All of the `_train`— functions are very similar. They take data, in the form of feature vectors and labels (except K-Means and PCA) and produce models. The models are then passed to `model_test` to make predictions on a set of test data. `knn_train` has a `K` which is the number of neighbors to be considered. `nn_train` has `hls` which stands for hidden layer size. This can be a tuple of any size, depending on how many layers you want (e.g. (5,2) or (3)). `pca_train` has a `K` for the number of principal components to keep in learning the transformation. `pca_transform` applies the learned transformation `pca` to the data. `svm_train` has a `k` which stands for kernel. This is a string. See the sklearn documentation for this. `compute_F1` should take the labels and the predictions and compute the F1 score.
Write-Up: Describe your implementations concisely.

The result of the test script gives the following results:

```
Decision Tree: 0.6250000000000001
Decision Tree + PCA: 0.5242718446601942
KMeans: 0.2769230769230769
KMeans + PCA: 0.2769230769230769
KNN: 0.576
KNN + PCA: 0.5736434108527131
Perceptron: 0.6095238095238096
Perceptron + PCA: 0.5454545454545454
Neural Network: 0.45161290322580644
Neural Network + PCA: 0.5000000000000001
SVM: 0.6222222222222222
SVM + PCA: 0.583941605839416
```