ECE 562 Machine Learning, Fall 2019
Instructor: Zafer Aydın
Computer Homework 8

Introduction

Download the template codes for Udacity's Intro to Machine Learning course by running the following command in a terminal

git clone https://github.com/udacity/ud120-projects.git

Submit your scripts as q1.py and/or a text document that includes your answers to the questions below through Canvas.

Assignment

1. A classic way to overfit an algorithm is by using lots of features and not a lot of training data. You can find the starter code in feature_selection/find_signature.py. Get a decision tree up and training on the training data, and print out the accuracy on training data. How many training points are there, according to the starter code?

2. What's the accuracy of the decision tree you just made on test set? (Remember, we're setting up our decision tree to overfit -- ideally, we want to see the test accuracy as relatively low.)

3. Take your (overfit) decision tree and use the feature_importances_ attribute to get a list of the relative importance of all the features being used. We suggest iterating through this list (it's long, since this is text data) and only printing out the feature importance if it's above some threshold (say, 0.2--remember, if all words were equally important, each one would give an importance of far less than 0.01). What's the importance of the most important feature? What is the index number of this feature (assuming that indices start from zero)?

4. In order to figure out what words are causing the problem, you need to go back to the TfIdf and use the feature number that you obtained in the previous part of the mini-project to get the associated words. You can return a list of all the words in the TfIdf by calling get_feature_names() on it; pull out the word that's causing most of the discrimination of the decision tree. What is it?

5. This word seems like an outlier in a certain sense, so let's remove it and refit. Go back to text_learning/vectorize_text.py (in computer homework 7), and remove this word from the emails using the same method you used to remove "sara", "chris", etc. Rerun vectorize_text.py, and once that finishes, rerun find_signature.py. Any other outliers pop up? What word is it? (Define an outlier as a feature with importance >0.2, as before).

6. Update vectorize_test.py one more time, and rerun. Then run find_signature.py again. Any other important features (importance>0.2) arise? What word is it?

7. What's the accuracy of the decision tree now? We've removed two "signature words", so it will be more difficult for the algorithm to fit to our limited training set without overfitting. Remember, the whole point was to see if we could get the algorithm to overfit--a sensible result is one where the accuracy isn't that great!