ECE 562 Machine Learning, Fall 2019
Instructor: Zafer Aydın
Computer Homework 1

Introduction

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

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

Submit your scripts as q1.py…q10.py and a text document that includes your answers to the questions below through Canvas.

Assignment

1. Create and train a Naive Bayes classifier in naive_bayes/nb_author_id.py. Use it to make predictions for the test set. What is the accuracy?

2. An important topic that we didn't explicitly talk about is the time to train and test our algorithms. Put in two lines of code, above and below the line fitting your classifier, like this:

```
t0 = time()
< your clf.fit() line of code >
print "training time:", round(time()-t0, 3), "s"
```

Put similar lines of code around the clf.predict() line of code, so you can compare the time to train the classifier and to make predictions with it. What is faster, training or prediction?

3. Go to the svm directory to find the starter code (svm/svm_author_id.py).

Import, create, train and make predictions with the sklearn SVC classifier. When creating the classifier, use a linear kernel (if you forget this step, you will be unpleasantly surprised by how long the classifier takes to train). What is the accuracy of the classifier?

4. Place timing code around the fit and predict functions, like you did in the Naive Bayes mini-project. How do the training and prediction times compare to Naive Bayes?

5. One way to speed up an algorithm is to train it on a smaller training dataset. The tradeoff is that the accuracy almost always goes down when you do this. Let's explore this more concretely: add in the following two lines immediately before training your classifier.

```
features_train = features_train[:len(features_train)/100]
labels_train = labels_train[:len(labels_train)/100]
```

These lines effectively slice the training dataset down to 1% of its original size, tossing out 99% of the training data. You can leave all other code unchanged. What's the accuracy now?

6. Keep the training set slice code from the last quiz, so that you are still training on only 1% of the full training set. Change the kernel of your SVM to "rbf". What's the accuracy now, with this more complex kernel?

7. Keep the training set size and rbf kernel from the last quiz, but try several values of C (say, 10.0, 100., 1000., and 10000.). Which one gives the best accuracy? Once you've optimized the C value for your RBF kernel, what accuracy does it give? Does this C value correspond to a simpler or more complex decision boundary?

ECE 562 Machine Learning, Fall 2019
Instructor: Zafer Aydın
Computer Homework 1

8. Now that you've optimized C for the RBF kernel, go back to using the full training set. In general, having a larger training set will improve the performance of your algorithm, so (by tuning C and training on a large dataset) we should get a fairly optimized result. What is the accuracy of the optimized SVM?

9. What class does your SVM (0 or 1, corresponding to Sara and Chris respectively) predict for element 10 of the test set? The 26th? The 50th? (Use the RBF kernel, C=10000, and 1% of the training set. Normally you'd get the best results using the full training set, but we found that using 1% sped up the computation considerably and did not change our results--so feel free to use that shortcut here.)

And just to be clear, the data point numbers that we give here (10, 26, 50) assume a zero-indexed list. So the correct answer for element #100 would be found using something like answer=predictions[100].

10. There are over 1700 test events--how many are predicted to be in the "Chris" (1) class? (Use the RBF kernel, C=10000., and the full training set.)