CS273a Homework #1

Introduction to Machine Learning: Fall 2016

Due: Monday October 3rd, 2016

Write neatly (or type) and show all your work!

This homework (and many subsequent ones) will involve data analysis and reporting on methods and results using Python code. I recommend that you use iPython Notebook, or if you prefer, Word, OpenOffice, or LaTeX to create a document with your answers and any code snippets that will tell us what you did and how you did it, then export the finished report as a PDF file and upload it to EEE. It is important that you include enough detail that we know how you solved the problem, since otherwise we will be unable to grade it.

I prefer to receive a *single* electronic document when possible; for some homeworks you may find it more convenient to do parts by hand, in which case a hard copy of that portion, or of the entire thing, is fine. Please *do not* upload a ZIP file containing multiple figures, as this is difficult to interpret without associated comments. If (for some reason) you need to submit large amounts of code, you can upload it in a *separate* ZIP file, but please ensure that there is enough information in your PDF report to understand what you did without examining the code itself.
Summary: turn in AT MOST (1) pdf stand-alone report file; (2) zip of supporting code; (3) hard copy of non-electronic work.

It is often useful to create a script that solves the homework (for example, in iPython notebook or as a python file); if so, you may find it useful to set the random number seed at the beginning, e.g., `numpy.random.seed(0)`, to ensure consistent behavior each time.

Problem 1: Python & Data Exploration

In this problem, we will explore some basic statistics and visualizations of an example data set. First, download the zip file for Homework 1, which contains some course code (the `mltools` directory) and the "Fisher iris" data set, and load the latter into Python:

```
import numpy as np
import matplotlib.pyplot as plt

iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the text file
Y = iris[:,-1]              # target value is the last column
X = iris[:,0:-1]            # features are the other columns
```

The Iris data consist of four real-valued features used to predict which of three types of iris flower was measured (a three-class classification problem).

(a) Use `X.shape[1]` to get the number of features, and `X.shape[0]` to get the number of data points.

(b) For each feature, plot a histogram ("`plt.hist`") of the data values

(c) Compute the mean of the data points for each feature (`np.mean`)

(d) Compute the variance and standard deviation of the data points for each feature

(e) "Normalize" the data by subtracting the mean value from each feature, and dividing by its standard deviation. (This will make the data zero-mean and unit variance.) Show your code. Note: you can do this with a for-loop (straightforward but slow), or in a simpler "vectorized" form using arithmetic operators directly on the numpy arrays.

(f) For each pair of features (1,2), (1,3), and (1,4), plot a scatterplot (see '`plt.plot`" or "`plt.scatter`") of the feature values, colored according to their target value (class). (For example, plot all data points with $y = 0$ as blue, $y = 1$ as green, etc.)

Problem 2: kNN predictions

In this problem, you will continue to use the Iris data and explore a KNN classifier using provided `knnClassify` python class. While doing the problem, please explore the implementation to become familiar with how it works.

First, we will shuffle and split the data into training and test subsets:

```
iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the data
Y = iris[:,-1]
X = iris[:,0:-1]

# Note: indexing with ":" indicates all values (in this case, all rows);
#  indexing with a value ("0", "1", "-1", etc.) extracts only that one value (here, columns);
#  indexing rows/columns with a range ("1:-1") extracts any row/column in that range.

import mltools as ml
# We'll use some data manipulation routines in the provided class code
# Make sure the "mltools" directory is in a directory on your Python path, e.g.,
#   export PYTHONPATH=${PYTHONPATH}:/path/to/parent/dir
# or add it to your path inside Python:
#   import sys
#   sys.path.append('/path/to/parent/dir/');

X,Y = ml.shuffleData(X,Y);                  # shuffle data randomly
# (This is a good idea in case your data are ordered in some pathological way,
#   as the Iris data are)

Xtr,Xte,Ytr,Yte = ml.splitData(X,Y, 0.75);  # split data into 75/25 train/test
```

Learner Objects Our learners (the parameterized functions that do the prediction) will be defined as python objects, derived from either an abstract classifier or abstract regressor class. The abstract base classes have a few useful functions, such as computing error rates or other measures of quality. More importantly, the learners will all follow a generic behavioral pattern, allowing us to train the function on a data set (i.e., set the parameters of the model to perform well on those data), and make predictions on a data set.

So, you can build and "train" a kNN classifier on `Xtr,Ytr` and make predictions on some data `Xte` with it using e.g.,

```
knn = ml.knn.knnClassify()     # create the object and train it
knn.train(Xtr, Ytr, K)     # where K is an integer, e.g. 1 for nearest neighbor prediction
YteHat = knn.predict(Xte)  # get estimates of y for each data point in Xte

# Alternatively, the constructor provides a shortcut to "train":
```

```
knn = ml.knn.knnClassify( Xtr, Ytr, K );
YteHat = predict( knn, Xte );
```

If your data are 2D, you can visualize a data set and a classifier's decision regions using e.g.,

```
ml.plotClassify2D( knn, Xtr, Ytr );  # make 2D classification plot with data (Xtr,Ytr)
```

This function plots the training data, then calls `knn`'s `predict` function on a densely spaced grid of points in the 2D space, and uses this to produce the background color. Calling the function with `knn=None` will plot only the data.

(a) Modify the code listed above to use only the first two features of X (e.g., let X be only the first two columns of `iris`, instead of the first four), and visualize (plot) the classification boundary for varying values of $K = [1, 5, 10, 50]$ using `plotClassify2D`.

(b) Again using only the first two features, compute the error rate (number of misclassifications) on both the training and test data as a function of $K = [1, 2, 5, 10, 50, 100, 200]$. You can do this most easily with a for-loop:

```
K=[1,2,5,10,50,100,200];
for i,k in enumerate(K):
  learner = ml.knn.knnClassify(...  # TODO: complete code to train model
  Yhat = learner.predict(...    # TODO: complete code to predict results on training data
  errTrain[i] = ...             # TODO: " " to count what fraction of predictions are wrong
  #TODO: repeat prediction / error evaluation for test data

plt.semilogx(...                #TODO: " " to average and plot results on semi-log scale
```

Plot the resulting error rate functions using a semi-log plot ("`semilogx`"), with training error in red and test error in green. Based on these plots, what value of K would you recommend?

Problem 3: Bayes Classifiers

In order to reduce my email load, I decide to implement a machine learning algorithm to decide whether or not I should read an email, or simply file it away instead. To train my model, I obtain the following data set of binary-valued features about each email, including whether I know the author or not, whether the email is long or short, and whether it has any of several key words, along with my final decision about whether to read it ($y = +1$ for "read", $y = -1$ for "discard").

x_1 know author?	x_2 is long?	x_3 has 'research'	x_4 has 'grade'	x_5 has 'lottery'	y $\Rightarrow$ read?
0	0	1	1	0	-1
1	1	0	1	0	-1
0	1	1	1	1	-1
1	1	1	1	0	-1
0	1	0	0	0	-1
1	0	1	1	1	1
0	0	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	1	1	1	-1

In the case of any ties, we will prefer to predict class +1.

I decide to try a naïve Bayes classifier to make my decisions and compute my uncertainty.

(a) Compute all the probabilities necessary for a naïve Bayes classifier, i.e., the class probability $p(y)$ and all the individual feature probabilities $p(x_i|y)$, for each class y and feature x_i

(b) Which class would be predicted for $\underline{x} = (0\ 0\ 0\ 0\ 0)$? What about for $\underline{x} = (1\ 1\ 0\ 1\ 0)$?

(c) Compute the posterior probability that $y = +1$ given the observation $\underline{x} = (1\ 1\ 0\ 1\ 0)$.

(d) Why should we probably not use a "joint" Bayes classifier (using the joint probability of the features x, as opposed to a naïve Bayes classifier) for these data?

(e) Suppose that, before we make our predictions, we lose access to my address book, so that we cannot tell whether the email author is known. Should we re-train the model, and if so, how? (e.g.: how does the model, and its parameters, change in this new situation?) Hint: what will the naïve Bayes model over only features $x_2 \ldots x_5$ look like, and what will its parameters be?