CS273a Homework #1

Introduction to Machine Learning: Winter 2015

Due: Tuesday January 13th, 2015

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

This may be your first homework using MATLAB ; please see the course webpage for links to tutorials to help you start using it. In matlab, "help <functionname>" is often a good idea; e.g., "help plot", "help find"...

For MATLAB reports such as this one, I recommend that you use Word, OpenOffice, or LaTeX to create a document with your answers and any code snippets that will tell us what you did, 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.

In Windows or Mac, you can import MATLAB figures into your document directly using copy-/paste operations, or you can export a figure from MATLAB directly to pdf using e.g. "print -dpdf <filename>". You can also use this to create and import JPEG or PNG files, but if you do so, please ensure that they are of sufficient resolution to be clear in the resulting document ("-r<value>" controls the resolution).

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.

Some students like to create a script to solve their homework; if so, you may find it useful to set the random number seed at the beginning, e.g., `rand('state',0); randn('state',0);` to ensure consistent behavior each time. Alternatively, if you merely want to remember what commands you executed, the `diary` function may be helpful.

Problem 0: Getting Connected

Please join our class forum on Piazza,
`http://piazza.com/uci/winter2015/cs178`
Please post your questions and discussion points on Piazza, rather than by email to me or the TA, since chances are that other students have the same or similar questions, and will be helped by seeing the discussion.

Problem 1: Matlab & Data Exploration

In this problem, we will explore some basic statistics and visualizations of an example data set. First, download and load the "Fisher iris" data set into Matlab (or Octave):

```
iris=load('data/iris.txt');     % load the text file
y = iris(:,end);           % target value is last column
X = iris(:,1:end-1);       % features are other columns
whos                       % show current variables in memory and sizes
```

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 `size(X,2)` to get the number of features, and `size(X,1)` to get the number of data points.

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

(c) Compute the mean of the data points for each feature (`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 (easy, but slow in Matlab), or in a "vectorized" form using `repmat` or `bsxfun` (faster, but harder to read).

(f) For each pair of features (1,2), (1,3), and (1,4), plot a scatterplot (see '`plot`" or "`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.) Note: if you wish to overlay several plot commands, use "`hold on`" before subsequent plots; to stop this behavior, use "`hold off`".

Problem 2: kNN predictions

In this problem, you will continue to use the Iris data and explore a KNN classifier using provided `knnClassify` Matlab class. For more information on classes in general, and this class in particular, see the Matlab OOP notes on the course page.

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

```
iris=load('data/iris.txt'); y=iris(:,end); X=iris(:,1:end-1);
% Note: indexing with ":" indicates all values (in this case, all rows);
%  indexing with a value ("1", "end", etc.) extracts only that one value (here, columns);
%  indexing rows/columns with a range ("1:end-1") extracts any row/column in that range.

[X y] = 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] = splitData(X,y, .75);  % split data into 75/25 train/test
```

Class Objects We will use Matlab classes to implement our learner methods. Matlab classes are a bit annoying to use, particularly the "old style" that are compatible with Octave. However, the usefulness outweighs the flaws.

An old-style class is created using a directory preceded by @. For example, included in the code is a kNN classifier, `@knnClassify`. The methods associated with this class are the Matlab `.m` files located within it. The constructor is `knnClassify`; all the other functions are called by providing a `knnClassify` object as the first argument. (That tells Matlab / Octave where to look for the function.) 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 = knnClassify( Xtr, Ytr, K );  % replace or set K to some integer
YteHat = predict( knn, Xte );      % make predictions on Xtest
```

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

```
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.)

(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=1:length(K)
  learner = knnClassify(...    % TODO: complete code to train model
  Yhat = 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
end;
figure; 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 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 Bayes classifier (using the joint probability of the features x, as opposed to a naïve Bayes classifier) for these data?

Problem 4: Gaussian Bayes Classifiers

Now, using the Iris data, we will explore a classifier based on Bayes rule. Again, we'll use only the first two features of Iris, shuffled and split in to training and test sets as before.

(a) Splitting your training data by class, compute the empirical mean vector and covariance matrix of the data in each class. (You can use `mean` and `cov` for this.)

(b) Plot a scatterplot of the data, coloring each data point by its class.

(c) Use `plotGauss2D` to plot contours on your scatterplot for each class, i.e., plot a Gaussian contour for each class using its empirical parameters, in the same color you used for those data points.

(d) Visualize the classifier and its boundaries that result from applying Bayes rule, using

```
bc = gaussBayesClassify( Xtr, Ytr );
plotClassify2D(bc, Xtr, Ytr);
```

(e) Compute the empirical error rate (number of misclassified points) on the training and test data.

(f) Now re-load the data, using all four features. Build another classifier with all features (you won't be able to plot this one) and compute its training and validation accuracy.