CS273a Homework #2

Introduction to Machine Learning: Fall 2013

Due: Tuesday October 15th, 2013

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

Please remember to turn in at most two documents, one with any handwritten solutions, and one PDF file with any electronic solutions.

Data

For the first two problems, please use the following data:

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.

Problem 1: Bayes Classifiers

First, 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., $p(y)$ and all the $p(x_i|y)$.

(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 (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?)

Problem 2: Decision Trees

(a) Calculate the entropy of the class variable y

(b) Calculate the information gain for each feature x_i. Which feature should I split on first?

(c) Draw the complete decision tree that will be learned from these data.

Problem 2: K-Nearest Neighbors and Validation

In this problem, you will use regression data and explore a KNN regression function using provided code. First, shuffle and split the data into training and test subsets:

```
iris=load('data/mcycle80.txt'); y=mcycle(:,end); X=mcycle(:,1:end-1);
[X y] = shuffleData(X,y);                % shuffle data randomly to avoid pathological orders
[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 regressor, `knnRegress`. The methods associated with this class are the Matlab `.m` files located within it. The constructor is `knnRegress`; all the other functions are called by providing a `knnRegress` object as the first argument. (That tells Matlab / Octave where to look for the function.) So, you can build and "train" a kNN regressor on `Xtr,Ytr` and make predictions on some data `Xte` with it using e.g.,

```
knn = knnRegress( Xtr, Ytr, K );  % replace or set K to some integer
YteHat = predict( knn, Xte );      % make predictions on Xtest
```

If your data are 1D, you can visualize a data set and a classifier's decision regions using e.g.,

```
plotRegress1D( knn, Xtr, Ytr );  % make 1D regression plot with data (Xtr,Ytr)
```

This essentially does what you did in the previous homework – use the automatic axes of the data scatterplot and compute predictions at closely spaced points, then plot them.

(a) Plot the resulting regression function for varying values of $K = [1, 5, 10, 50]$ using `plotRegress1D`.

(b) Compute the mean squared error on both the training and test data as a function of $K = [1, 2, 4, 8, 16, 32, 64]$. You can do this most easily with a for-loop:

```
K=[1, 2, 4, 8, 16, 32, 64];
for k=1:length(K)
  learner = knnRegress(...    % train model
  Yhat = predict(...           % predict results on training data or test data
  knnMse(k) = ...               % compute the mean squared error
end;
figure; semilogx(...           % 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?

(c) Often, it may be that the number of training examples available are not enough to split out a validation set and be confident of its accuracy (here, for example, you have only 20 points). In such cases, when determining a model parameter such as K, it is often helpful to use *cross-validation*. In cross-validation, we perform several splits of the data and train/test on each, then use the overall average to determine a value of K. You can do a 5-fold validation test, for example, by

```
nFolds = 5;
for iFold = 1:nFolds,
  [Xti,Xvi,Yti,Yvi] = crossValidate(X,Y,nFolds,iFold);  % take ith block of (X,Y) as testing
  learner = knnRegress(...   % train on Xti, Yti
  % now compute the MSE on Xvi, Yvi and save it
end;
% the overall estimated validation performance is the average of the performance on each fold
```

Use 5-fold cross-validation to re-estimate the validation accuracy as a function of K. What is the best value of K predicted by this method?

Problem 3: Bayes Classifiers

Now, using the Iris data, we will explore a classifier based on Bayes rule. First, we'll use only the first two features of Iris. Shuffle and split the data into training and test subsets as before:

```
iris=load('data/iris.txt'); y=iris(:,end); X=iris(:,1:2);  % take only 2 features
[X y] = shuffleData(X,y);                % shuffle data randomly to avoid pathological orders
[Xtr Xte Ytr Yte] = splitData(X,y, .75);  % split data into 75/25 train/test
```

(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 validation 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.

Problem 4: Decision Trees

We'll use the same Iris data again to build a decision tree classifier. Again, start with only the first two features.

(a) Use the `treeClassify` class to build a decision tree classifier of your training Iris data. Because of the default settings my code uses, to get a "standard" decision tree you will need to specify: `minParent=0`, `maxDepth=inf`, `minScore=-1`, `nFeatures=inf`

(b) Plot the classification boundary that results using `plotClassify2D`

(c) Complexity of a decision tree classifier is often controlled by setting a depth cutoff, so that for depth d there are at most 2^d leaf nodes. Use your validation data to evaluate the test performance of a decision tree classifier at various maximum depth values d (specified with `maxDepth`, and use this to suggest a value of d for test performance.

(d) Repeat your search over d using a classifier built using all four features. What value would you choose in this case?