Automation of Biological Research: 02-450/02-750

Carnegie Mellon University

Homework 4

Version: 1.0; updated 10/11/2015

Due: October 27 by 11:59pm

Hand-in: Email your responses to: **ABR-instructors@googlegroups.com**

Scenario: You download a piece of software that predicts whether two proteins will bind to one another based their primary sequences. Specifically, the program estimates the probability the molecules bind, so the raw output is a number from 0 to 1. You aren't entirely sure how to interpret the program's score, so you decide to test some of the predictions with an *in vitro* assay. The goal is to estimate a *threshold* value on the program's output such that values at or above this threshold are likely to be true positives.

For this assignment, you will be implementing the CAL and DHM algorithms that were discussed in class, and a simple random learner. Before you begin, you should re-read the "Two Faces of Active Learning" paper, the CAL paper, and the DHM paper.

For both problems, the hypothesis class is threshold functions on the unit interval. That is, a function which returns 1 for any point that is greater than or equal to a threshold value, *t*, and zero otherwise. The learning algorithms will be used to estimate the value of *t* from data.

Partial implementations of the CAL and DHM algorithms are provided in the folders **q1** and **q2**, respectively. You will need to complete the implementations. Specifically, you need to complete the functions in **runExperimentsQ1.m** and **runExperimentsQ2.m**.

A number of pre-written functions are provided. You don't need to change these files, but you will find them useful. The most important of these functions is called **learnQ1** and **learnQ2**. The **learn** functions implement simple algorithms for returning consistent models from the given data. Note that if you call **ABRHW4_Q1(noise)** and **ABRHW4_Q2(noise)** from the matlab prompt they will run 20 experiments with your implementation and plot the results. The argument **noise** determines the probability that the oracle mislabels instances.

Your job is to implement the main loops of the **runExperiments** functions. This will require:

1) maintaining the sets of labeled and unlabeled data through each iteration,
2) calling the **learn** subroutines with the appropriate arguments,
3) execute the logic of the CAL and DHM algorithms,
4) implement a learner that selects random points for labeling (for comparison with CAL/DHM)
5) tracking the generalization errors of CAL/DHM and the random learner as a function of the number of calls to the oracle,

You should hand in a single file containing the various plots and discussion described below. You should also hand in your code.

Problem 1: CAL (35 points):

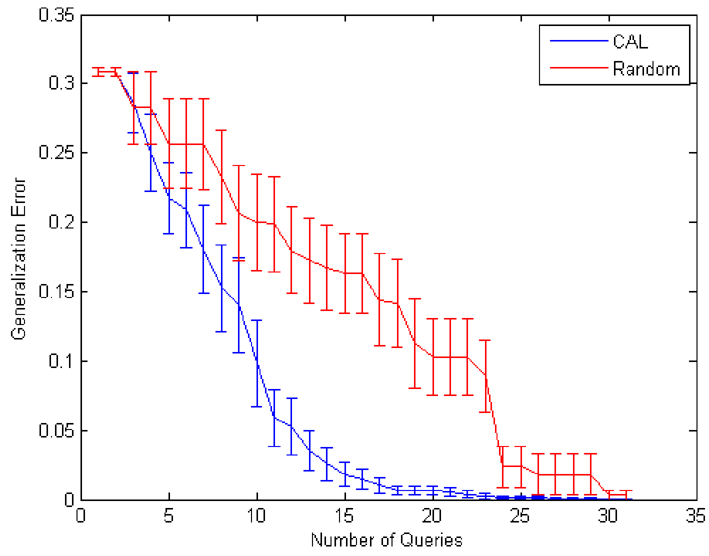
A. (20 points) Complete the implementation of CAL and the random learner in **runExperimentsQ1.m**. We recommend using the pseudo-code description in Figure 6 – right of the "Two Faces of Active Learning" paper for guidance.

B. (5 points) Call the function **ABRHW4_Q1(0)** from the matlab prompt. This will run your code 20 times on separable data and produce a plot like the one below, showing the generalization error of the CAL and random learner as a function of the number of queries, with standard error bars. Recall that standard error is:

$$SE = \sigma/\sqrt{n}$$

where σ is the standard deviation (of the error for a given number of queries) and n is the number of samples (in this case, n = 20). The matlab function **errorbar(X,E)** plots the curve **X** with the errors in **E**.

Be sure to hand in your plot.



C. (5 points) Call the function **ABRHW4_Q1(0.1)** from the matlab prompt. This will run your code 20 times on non-separable (i.e., noisy) data and produce a plot like the one above. Discuss the differences in the generalization curves for the version with and without noise. Does CAL outperform the random learner in the presence of noise? Why or why not? **Be sure to hand in your plot.**

D. (5 points) Call the function **ABRHW4_Q1(x)** from the matlab prompt for several values of 0.1 < x < 0.5. Each call will run your code 20 times on non-separable data and produce a plot like the one above. For which value of x does the random learner start to perform as good as, or better than CAL? **Hand in your plot for that value of x.**

What to hand in: Your code; your plots (for parts B-D); your response to parts C and D.

Problem 2: DHM (65 points)

A. (50 points) Complete the implementation of DHM in **runExperimentsQ2.m**. We recommend using the pseudo-code description in Figure 16 of the “Two Faces of Active Learning” paper for guidance.
B. (5 points) Call the function **ABRHW4_Q2_PartB** from the matlab prompt. This will run your code 20 times on separable data and produce a plot like the one above. Compared to CAL, how well does DHM perform on non-separable data? **Be sure to hand in your plot.**
C. (5 points) Call the function **ABRHW4_Q2_PartC** from the matlab prompt. This will reproduce a plot like figure 9 in the “Two Faces of Active Learning” paper. **Be sure to hand in your plot.**
D. (5 points) Call the function **ABRHW4_Q2_PartD** from the matlab prompt. This will reproduce a plot like figure 11 (bottom panel) in the “Two Faces of Active Learning” paper. **Be sure to hand in your plot.**

What to hand in: Your code; your plots (for parts B-D).