

Laboratory Worksheet, Friday, Feb. 20.

There are several projects to work our way through today. They require two fixes for software that isn't installed permanently on this network. They involve two different computing environments, and both are based on line commands. Parts I. and II. use Matlab. Part III. goes back over the ECORRD problem, using a fix to be able to use the SeqAln package to SW align pairs of sequence with p -values determined by simulation. Finally, part IV. simply is to hear reports about data collection for PHX project. We should gather it all in a space accessible to all as we locate it.

A. The Matlab Fix. Go to the *Web Resources* link from the course home page. Go to Kevin Murphy's distribution site. Download the *HMM Toolbox*, which will require you to download as well the three toolboxes: *KPMtools*, *KPMstats*, and Ian Nabney's *netlab*. In the Mac's these should mount themselves on the desktop. Now call Matlab from the dock (it is the brown cone shaped icon in the bar at the bottom of the desktop). At the command line prompt (which looks a bit like `>>`) type `addpath ~/Desktop/HMM` [return], and so on, according to the names of the four downloads. This sets the paths to these functions for the permanently mounted Matlab in the lab. These mounted toolboxes will *disappear* when you log off. You can save them in your personal space, if you have a way to use them within Matlab there.

I. The Viterbi Path Through an HMM. The point here is simply to take the data from the dishonest casino example we discussed, and find the Viterbi algorithm's prediction for the most probable path through the HMM. The data are already stored for you in what is called a Matlab Workspace. Such files are denoted by a `.mat` extension in Matlab, commands are denoted with a `.m` extension. In the *548 Resources* page you will find a workspace file `dicedata.mat`. A workspace file, saved, can be reloaded, and is a convenient way to reuse variables and data you have already gotten into Matlab form. This file contains several variables:

Variable	Description
----------	-------------

O	This is a vector of length 300 containing the observed rolls, in Matlab format.
OB	This is a 2×6 matrix; the two rows give the emission probabilities. (The first row is for the fair die.)
PR	Is a 2-vector, the prior probabilities for the entry into the first state.
B	This is the 2×2 matrix of transition probabilities for the hidden Markov chain.

If we have time, I will explain why I made the choice which I did for the prior distribution.

(It is set at (2/3, 1/3).)

As noted in lecture Wednesday, we will use *viterbi_path.m* from the toolbox *HMM*, and the input for this program is the set of probabilities, if $y(t)$ is the data observed, $t = 1, \dots, 300$, given as

$$obslik(i, t) = P(y(t)|Q(t) = i),$$

where $Q(t)$ is the (hidden) state at position t ($i = 1$ [fair] or 2 [loaded]). The command *dataOL.m* in the *548 Resources* – you should download this into your *HMM* directory – converts the observations into the obslik input data, given the emission probabilities matrix *OB*. The command would be

```
>> C = dataOL(OB, 0);
```

which will store *C*. Now you can run

```
>> path = viterbi_path(PR, B, C)
```

This will print out a 300-vector of 1's and 2's for the Viterbi path through the model. Compare this to the path given in the example data. If we had time, you could work to present the data a little more cleanly: change 1 to F and 2 to L, and print a 3×300 matrix where the first row is the observed data, the middle row is the actual data for F or L, and the bottom row is the Viterbi prediction (F or L in each slot).

II. Training an HMM. This will use the command *learn_dhmm.m* in the *HMM* toolbox. I will only sketch the procedure here. You should first pull up the file for the command from the *HMM* directory and read it. What are the inputs? What is the structure of the *data* input? We will have data for training a model where we will fix upon 2 hidden states beforehand, each with 6 emissions. Given this simple architecture of the model we are looking for, how many parameters are we looking to estimate in order to train the HMM model?

We will get the data by simply taking our original data sequence *O* and permuting it. Matlab has a random permutation generator called *RANDPERM* (you can look it up in the help window to see what its arguments are). The proposal is to permute *O* randomly about 500 or 600 times, and build an array of data sequences which can be used for Baum-Welch (EM) training. We probably won't be able to finish this this time around, but let's start, if possible.

Think about the following questions: do we really expect the same parameters to emerge from training that we put into it? Why or why not? What happens if we try to fit the data to an HMM with three hidden states?

III. Putative Highly Expressed Genes (PHX). We have to discuss the results of your search for data for this project, and agree on how to store it for common use. Next we have to discuss how the component programming parts are coming.

B. The SeqAln Fix. The local systems people have stored the Waterman program suite *SeqAln* locally at <http://www.math.lsa.umich.edu/courses/seqaln-2.0.dmg> as a disk image. Download this file to your desktop, where it becomes an icon. From the systems person here: "This is a MacOSX Jaguar binary, and the executable files can be found in that tree. So...students have to download and mount the dmg file and then type: `/Volumes/seqaln-2.0/seqaln/bin/SOMEPROGRAM` to run a program. Hope this helps."

IV. ECORRD Again. Now go back to the problem from earlier (on the second PS). We can now use *SeqAln* software. In the *SeqAln* online man text you should look up the structure of the *pvlocal* command. The *SeqAln* directory has an online man pages section where you will find the options, etc., for most *SeqAln* commands. There are also stored a library of scoring matrices. You could check the various matrices, and use the ones originally proposed in the course last week.