EMHMM-Toolbox: Eye-Movement analysis with Hidden Markov Models (HMMs)

Antoni B. Chan, City University of Hong Kong
Janet H. Hsiao, University of Hong Kong
Tim Chuk, University of Hong Kong

2017-Jan-21, v0.5

DESCRIPTION

This is a MATLAB toolbox for analyzing eye movement data with hidden Markov Models (HMMs). The major functions of the toolbox are:

1) Estimating HMMs for an individual's eye-gaze data.
2) Clustering individuals' HMMs to find common strategies.
3) Visualizing HMMs and fixation data.
4) Statistical tests to see if two HMMs are different.

You can find more information about the methodology in the below reference papers.

REFERENCES

If you use this toolbox, please cite the following papers:

- For learning HMMs for eye gaze data: Tim Chuk, Antoni B. Chan, and Janet H. Hsiao, "Understanding eye movements in face recognition using hidden Markov models", *Journal of Vision*, 14(11):8, Sep 2014.
- For clustering HMMs with the VHEM algorithm: Emanuele Coviello, Antoni B. Chan, and Gert R.G. Lanckriet. "Clustering hidden Markov models with variational HEM". *Journal of Machine Learning Research*, 15(2):697-747, Feb 2014.

CONTACT INFO

Send comments, bug reports, feature requests to Antoni Chan (abchan@cityu.edu.hk).

DEMO

In this section, we will go through one of the demo programs to show the functions of the toolbox. In MATLAB, change to the main directory of the toolbox and run the following to setup the paths.

```
>> setup
```

Now change to the “demo” folder, run "demo_faces" to run a demo script.

```
>> cd demo
>> demo_faces
```

This script will first load the fixation data from an Excel spreadsheet, demodata.xls. The spreadsheet has 4 columns: SubjectID, TrialID, FixX, and FixY, which correspond to the subject ID, trial number (ID), and X and Y fixation locations. The fixation data is

automatically separated according to the subject ID and trial number. In each trial, it assumes that the fixations occur in order. The output will look like this:

```
Reading demodata.xls
- found SubjectID in column 1
- found TrialID in column 2
- found FixX in column 3
- found FixY in column 4
- found 10 subjects:
1 2 3 4 5 6 7 8 9 10
    * subject 1 had 56 trials
    * subject 2 had 56 trials
    * subject 3 had 56 trials
    * subject 4 had 56 trials
    * subject 5 had 56 trials
    * subject 6 had 56 trials
    * subject 7 had 56 trials
    * subject 8 had 56 trials
    * subject 9 had 56 trials
    * subject 10 had 56 trials
```

The program found 10 subjects with 56 trials each.

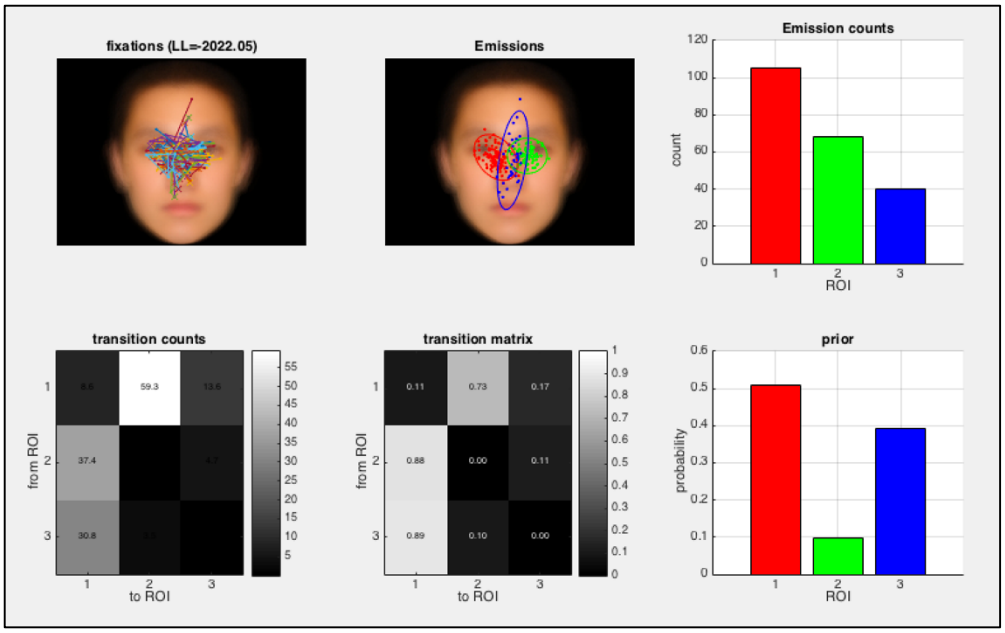
Next, an individual HMM will be estimated from each subject's fixation data. We use the variational Bayesian EM algorithm to estimate the HMMs. It can automatically select the number of ROIs. In this case, we try K=2 and K=3 ROIs. For each K, the VB algorithm is run 50 times with random initializations, and the run with the highest log-likelihood is kept. The best model over all the candidate Ks will be selected. The output for one subject looks like this:

```
=== running Subject 1 ===
-- vbhmm K=2: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50
best run=20; LL=-2031.89
-- vbhmm K=3: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50
best run=40; LL=-2022.05
best model: K=3; L=-2020.25
```

To visualize an HMM and the data, we use the following 2 commands:

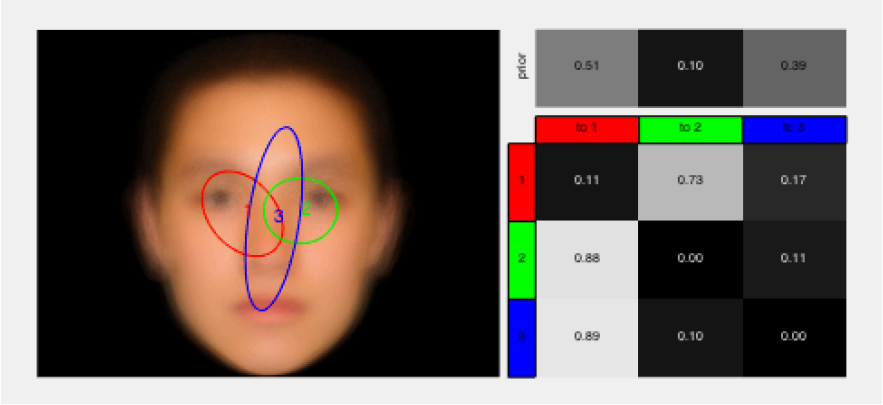
```
>> vbhmm_plot(hmms{1}, data{1}, faceimg)
>> figure, vbhmm_plot_compact(hmms{1}, faceimg)
```

The first command looks at the HMM of the first subject. "faceimg" contains the name of the image file of the face to put in the background. The figure from `vbhmm_plot` looks like this (Figure 1 from the demo):



The top-left plot shows the fixation sequences, and the "x" is the first fixation. The top-middle plot shows the emission densities (ROIs) and the corresponding fixations. The top-right plot shows the total number of fixations in each ROI. On the bottom, the left and middle plots show the transition counts and transition matrix. In the transition matrix (bottom-middle), each row is a probability distribution. I.e., each row is normalized to sum to 1. Finally, the bottom-right shows the prior probability, i.e., the probability of the ROI of the first fixation. The ROIs are automatically sorted according to the most likely fixation path: ROI 1 is the most probable initial fixation; ROI 2 is the most likely next fixation, given a fixation in ROI 1; etc. In this example, the subject has 50% probability to look at ROI 1 (red region). Then has a 73% of transitioning to the green ROI 2. In the green ROI 2, the subject will look back at ROI 1 (88% probability). So the subject tends to look back and forth between the eyes. In 40% of the cases, the subject will first look in the middle of the face at ROI 3, and then transition to the red ROI 1.

The second command makes a compact plot of the HMM and doesn't show the data. The figure looks like this (Figure 2 from the demo):



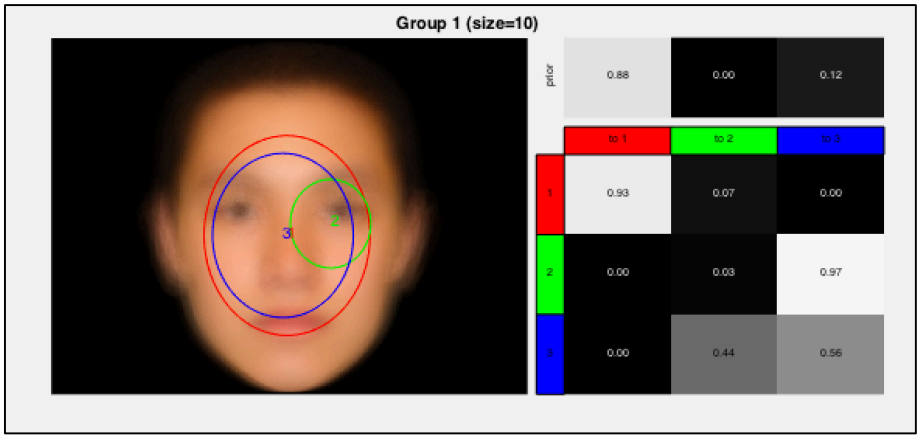
On the left side are the ROIs (labeled with colors and numbers). The right-top shows the prior probabilities, and the right-bottom shows the transition matrix. The rows/columns are labeled with colors and numbers to indicate the corresponding ROIs.

The demo script plots all the subjects in one figure (Figure 3 from the demo):

Given the 10 individuals' HMMs, the demo script then computes the overall eye gaze pattern by clustering the individuals into one group. Here the variational HEM algorithm is used to cluster the HMMs. The algorithm is run 50 times with random initializations, and the run with the highest log-likelihood is kept. The output of the demo script looks like this:

```
=== Clustering (1 group) ===
VHEM Trial: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50
Best run is 39: LL=-1011.22
```

The corresponding plot of the group HMM is here (Figure 4 from the demo):



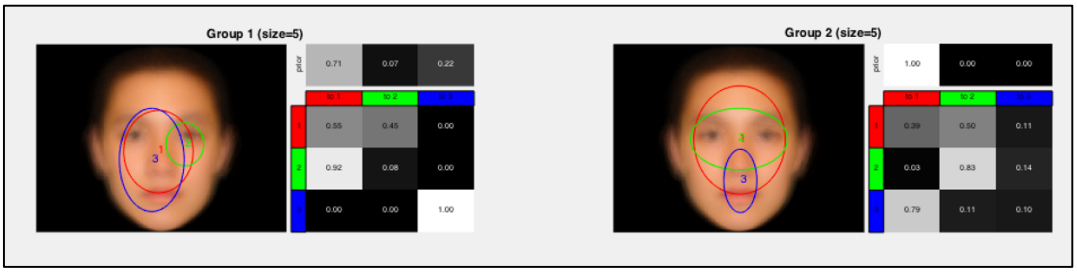
For the overall HMM for all subjects, two of the ROIs are large and cover most of the face features. This is mainly because there are different strategies used by each subject, and merging them all together yields a "washed out" HMM, which lacks specific details. One interesting thing is that the right eye has its own ROI (green ROI 2) and the transition matrix shows very high probability of moving from ROI 2 to ROI 3. This indicates that a common strategy of subjects looking at the right eye consistently, and then look around the center of the face.

To discover the different strategies among the subjects, next the demo script clusters the individual HMMs into 2 groups using VHEM. Here is the output:

```
=== Clustering (2 groups) ===
VHEM Trial: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50
Best run is 37: LL=-1005.72

...
Group membership:
  group 1 = [1 3 6 8 9]
  group 2 = [2 4 5 7 10]
```

Here is the plot of the two group HMMs (Figure 5 from the demo):



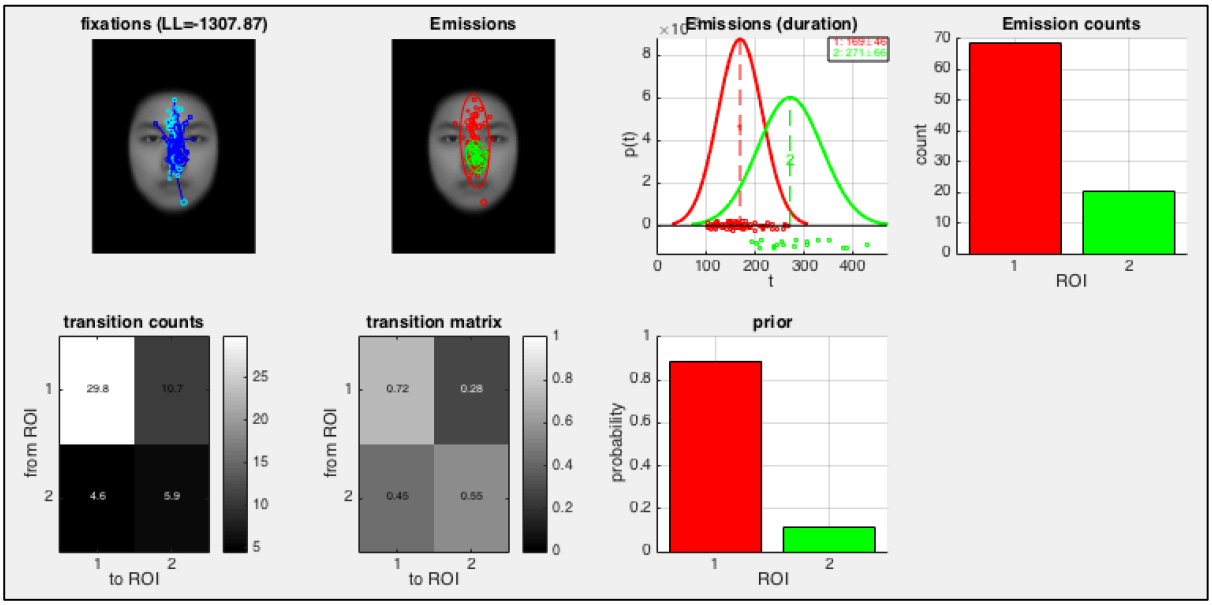
After clustering into two groups, we see two separate strategies. For Group 1 (left), they tend to first look around the center of the face (nose), and then the right eye. For Group 2 (right), they tend to look at the two eyes (ROI 2), and the mouth (ROI3).

Finally, we test whether the two group HMMs are different. We take the data for subjects in Group 1, and calculate the log-likelihood of this data under the Group 1 HMM and under the Group 2 HMM. The average difference between the log-likelihoods is an estimate of the Kullback-Leibler (KL) divergence, which is a dissimilarity measure for probability distributions. A KL divergence of 0 means that the two distributions (HMMs) are the same. We run a paired t-test on the two lists of log-likelihoods to see if the average log-likelihood difference is significantly different from 0. Since KL divergence is not symmetric, we need to do this twice: once using the data from Group 1, and once using the data from Group 2. The test results are below:

```
test group hmm1 different from group hmm2: t(4)=6.28769; p=0.00163401
test group hmm2 different from group hmm1: t(4)=4.00247; p=0.0080485
```

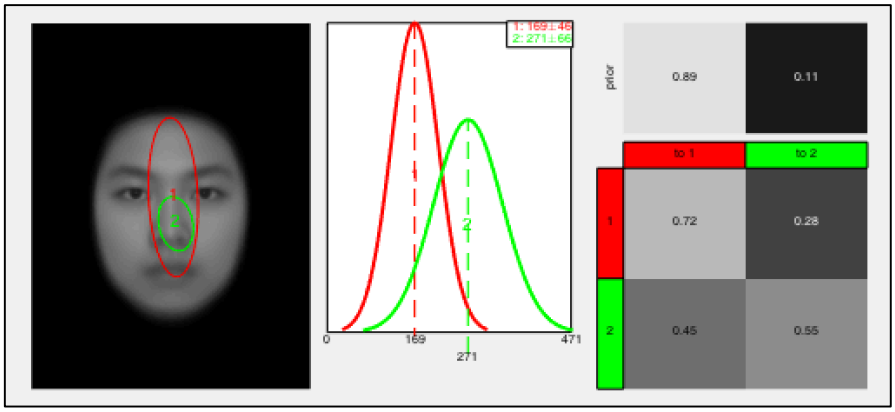
In both cases, the two HMMs are significantly different (p=0.0016 and p=0.008).

The toolbox can also learn HMMs for eye fixation data consisting of both fixation location (x,y) and duration (time in milliseconds). An example can be found in the "demo_faces_duration.m" script. The script loads the data from "jov_duration.xls". The Excel spreadsheet now has a 5th column with the header "FixD", which is for the fixation duration. The commands used to learn a subject's HMM and group HMMs are the same. Here is an example of one subject's HMM using fixation location and duration:



The 3rd plot shows the distribution of fixation duration for each ROI. For this subject, they tend to have a quick fixation on the face, and then sometimes fixate longer on the nose. The fixation duration data is shown at the bottom of the ROI duration plots. Note that the x-axis values are the durations, and the y-values are randomly selected to better visualize the data.

Here is a compact plot of the same HMM:



More details can be found in the demo script “demo_faces_duration.m”.

That’s all! There are more demos for clustering and comparing HMMs in the demo scripts "demo_faces_jov_clustering.m" and "demo_faces_jov_compare.m".

FUNCTION USAGE

In this section, we provide more details about the main function calls and their input parameters. More detailed help can be found in MATLAB for each function by using the help command on the function (e.g., `help vbhmm_learn`).

Options are passed to the main functions, `vbhmm_learn` and `vhem_cluster`, using a structure where each field represents one parameter setting. Values are not specified in the structure will be automatically filled-in using default values.

vbhmm_learn - estimate an HMM from eye fixation data

Usage:

```
[hmm,L] = vbhmm_learn(data,K,vbopt)
```

INPUTS	Description
`data`	Nx1 cell array, where each element is a fixation sequence and N is the number of sequences. data{i} is a TxD matrix, where T is the sequence length, and D is the dimension of a single fixation point (typically 2 for x and y coordinates).
`K`	The number of hidden states (ROIs) to use. If given a vector, the K will be automatically selected among the entries in the vector.
`vbopt`	structure containing options (see below)
`vbopt.alpha`	The Dirichlet distribution concentration parameter for the prior distribution of the initial fixation. Large values encourage a uniform prior, while small values encourage a concentrated prior (default=0.1). It should be a positive number. Another way to think of it is in terms of "virtual" samples. A typical way to estimate the probability of something is to count the number of samples that it occurs and then divide by the total number of samples, i.e. P = (# times it occurred) / (# samples). The alpha parameter of the Dirichlet adds a “virtual” sample to this estimate, so that the probability estimated is P = (# times it occurred + alpha) / (# samples). Hence, for small alpha, the probability estimate will just follow the data, while for large alpha, it forces all the probabilities to be very similar (i.e., uniform).
`vbopt.epsilon`	The concentration parameter for the Dirichlet distribution on the rows of the transition matrix (default=0.1). The meaning is similar to alpha above, but for the probabilities in the transition matrix.
`vbopt.mu`	prior mean of the ROIs (default = [256;192]). Typically, it should be at the center of the image. Alternatively, you can use the average fixation location. When fixation duration is included in the data, then the default is [256;192;250].
`vbopt.W`	Inverse variance of the inverse Wishart distribution (default=0.005). This parameter determines the variance (width) of the ROI prior. For example, W=0.005 means that the variance of the ROI prior is 1/0.005=200. Hence, the ROI prior has a standard deviation of 14 pixels, i.e., the ROI ellipses have width of 14*4 = 56 pixels. A vector can be specified, in which case the (x,y,d) widths of the ROIs can be different.

vbopt.v	The degree-of-freedom of the inverse Wishart, v > D-1. (default=10). Larger values give preference to diagonal covariance matrices.
vbopt.beta	The Wishart concentration parameter (default=1). Large values encourage estimated ROIs to be similar to the prior ROI (mean & W), while small value ignores the prior.
vbopt.numtrials	The number of trails to run when using 'random' initialization (default=50).
vbopt.showplot	1 means show a plot of the HMM (default). 0 means don't show a plot.
vbopt.verbose	0 means don't show any messages. 1 means show a few messages showing progress. (default) 2 means show lots of messages.

OUTPUTS	Description
hmm	Structure containing all the HMM parameters and other information. Use "help vbhmm_learn" for more details.
L	The log-likelihood of the data for the hmm.

vbhmm_auto_hyperparam – automatically select some hyperparameters for vbhmm_learn

This is a helpful function to automatically set the hyperparameters mu and W for vbhmm_learn.

Usage:

```
[vbopt] = vbhmm_auto_hyperparam(vbopt, data, img, opt)
```

INPUTS	Description
vbopt	Existing vbopt structure
data	the same format as vbhmm_learn. Or a cell array of subjects' data: data{s}{i} = the i-th trial for the s-th subject.
img	the template image (or filename)
opt	Option settings; 'c' = set mu as the center of the image, and W so that the prior ROI is 1/8 the width of the image. For duration, set the mean as 250ms and standard deviation as 25ms. 'd' = use a data-driven approach. Set the mu as the mean fixation location and duration. Set W according to the inverse variance of the data. Assumes that the ROI is circular in x-y dimensions.

OUTPUTS	Description
vbopt	New options structure with mu and W set.

vhem_cluster – cluster HMMs into groups using VHEM

Usage:

```
[group_hmms] = vhem_cluster(hmms, K, S, hemopt)
```

INPUTS	Description
hmms	Nx1 cell array of HMMs, each learned with vbhmm_learn.
K	The number of groups to cluster the data
S	The number of states (ROIs) in each group HMM. Use [] to automatically select the number of states as the median number of states in the input HMMs.
hemopt	structure containing options (see below)
hemopt.trials	number of trials with random initialization to run (default=50)
hemopt.verbose	0 means don't show any messages. 1 means show a few messages showing progress (default) 2 means show lots of messages.

OUTPUTS	Description
group_hmms	A structure containing the group HMMs and other information. Use "help vhem_cluster" for more details.
group_hmms.hmms	A 1xK cell array, where each entry is a group representative HMM.
group_hmm.LogL	The log-likelihood score of the group HMMs.
group_hmm.label	A 1xN vector, where each entry is the cluster assignment for the i-th input HMM.
group_hmm.groups	A 1xK cell array, where each entry contains the group members for that group.

TOOLBOX CONTENTS

The contents of the emhmm toolbox are listed below. Here only the important functions for the user are highlighted.

Directory/Filename	Description
emhmm_version.m	Toolbox version number and history
setup.m	Setup MATLAB path for the toolbox
demo/	
demo_faces.m	Simple example for learning individual HMMs and clustering HMMs.
demo_faces_duration.m	Simple example using fixation location and duration.
demo_faces_jov_clustering.m	Another example of learning and clustering HMMs.
demo_faces_jov_compare.m	Example of comparing HMMs learned from correct and incorrect face recognition trials.
src/hem	
vhem_cluster.m	Cluster HMMs with VHEM algorithm
vhem_plot.m	Visualize the group HMMs
vhem_plot_clusters.m	Visualize the group HMMs and cluster members
src/hmm	
vbhmm_learn.m	Learn HMM from data
vbhmm_plot_compact.m	Visualize an HMM using a compact format
vbhmm_plot.m	Visualize an HMM
vbhmm_kld.m	Calculate the KL divergence between 2 HMMs

vbhmm_ll.m	Calculate the log-likelihood of data for an HMM
vbhmm_auto_hyperparam.m	Automatically select some hyperparameters
src/stats	
stats_ttest.m	Run a t-test to compare log-likelihoods of two HMMs.
src/util	
read_xls_fixations.m	Read fixations from an Excel spreadsheet in standard format.