**First Example**

>> load example1 %Charge data

>> plot(data(:,1),data(:,2),'x') %See data in plot (2 dimensions)

>> figure

>> bar(stateseq) %See state sequence (3 states)



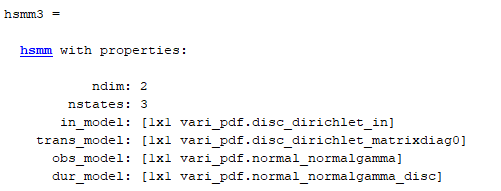


See the data are 3 states with mean (0,0), (10,10) and (20,20)

The first is to create the model HSMM

>> hsmm3=hsmm(2,3); %Create HSMM object with data of 2 dimension and 3 states

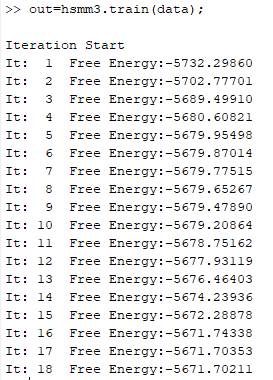
>> hsmm3 %Show the object structure data



This model is created with component default and non-informative prior

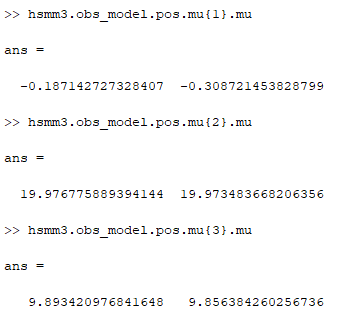
For train the model use:

>> out=hsmm3.train(data);



After the train hsmm3 is the trained model, additionally there is an output which include important information.

If I want to see the mean of trained model, use:



The out structure data is:

freeenergy: -5.671702109481846e+03 % Free Energy

loglik: -5.514143254939676e+03 % Log Likelihood

totaldiv: -1.575588545421694e+02 % Total of KL Divergence

freearray: [1x18 struct] % Array with detail about iteration

decodevar: [1x1 struct] % Structure regarding with decode process

totaliter: 18 % Iteration total

stateseq: [1x1009 double] % State Sequence estimated

nstates: 3 % States Number, in this case is the trivial information

Normally it is not know the number of cluster, in this case is possible to create the HSMM object only with dimension number:

>> hsmm3=hsmm(2); %Create HSMM object with data of 2 dimension

The train command is the same:

>> out=hsmm3.train(data);

In this case the algorithm check with different number of cluster choosing the one with the most free energy. The output is:

freeenergy: -5.673197823351075e+03

loglik: -5.515638968808905e+03

totaldiv: -1.575588545421693e+02

freearray: [1x18 struct]

decodevar: [1x1 struct]

totaliter: 18

stateseq: [1x1009 double]

nstates: 3