CSC246 Project3
Junfei Liu(jliu137)

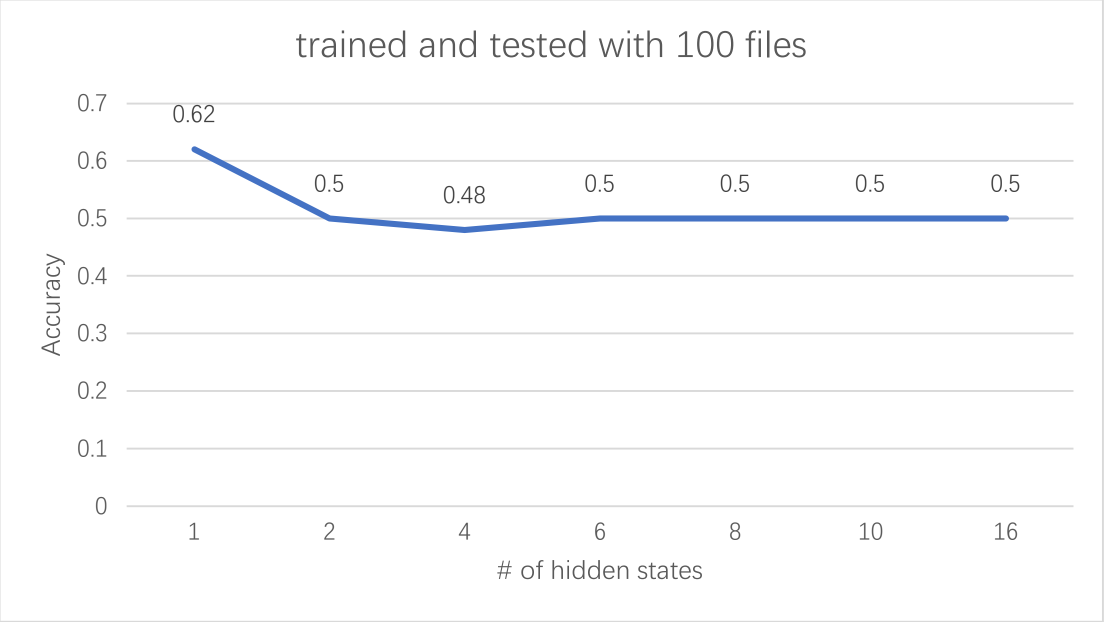
As an introduction, my program implements a hidden Markov model on independent datasets that is able to perform necessary calculations such as forward-backward and EM to generate a model which can improve likelihood each iteration. The algorithm can keep running and increase the log likelihood until it has converged or the maximum iteration is reached.

I have finished both building a classifier and exploring convergence experiments of which the results will be illustrated below. It would be the best if extra credits can be awarded. Because of time constraints and hardware limitations, data points could be too few to back up findings. Run instructions are in readme.txt.

Classifier

I ran the tests on the classifier with two sets of parameters. In the first set, the models are trained with 100 files, 50 maximum iterations, 0.01 tolerance (the minimum improvement on log likelihood per iteration. The training will stop if LL increases less than tolerance), and number of hidden states in [1,2,4,6,8,10,16], and tested on 100 test files. The accuracy is determine by # of correct / # of total test files.

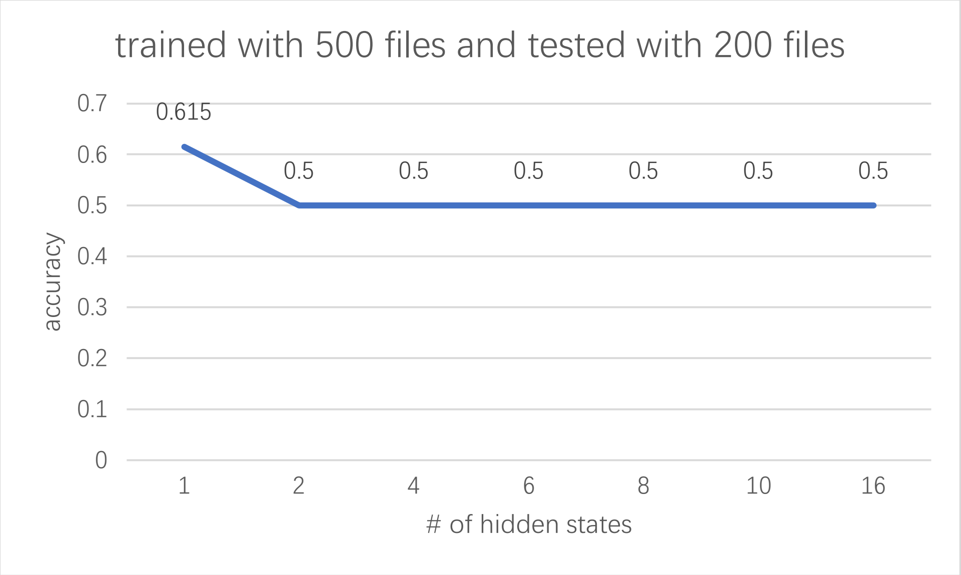
# of hidden states	1	2	4	6	8	10	16
accuracy	0.62	0.5	0.48	0.5	0.5	0.5	0.5



Generally, the model is performing poorly on the classifier task as the accuracy is close to 50% with most models trained with different number of hidden states. The performance of model trained with one hidden states perform significantly better. To increase the credibility of test results, I tested with models trained and tested with more files.

In the second set, the models are trained with 500 files, 50 maximum iterations, and 0.001 tolerance, and tested on 200 test files.

# of hidden states	1	2	4	6	8	10	16
accuracy	0.615	0.5	0.5	0.5	0.5	0.5	0.5



However, the second set of tests showed almost identical result. The model with one hidden state reaches accuracy over 60% and other models perform the same as a random number generator does. Surprisingly, the accuracy remains 50% for most data points accidentally. The significant figures are shown with 3 digits so the test results are genuinely half accurate and half inaccurate.

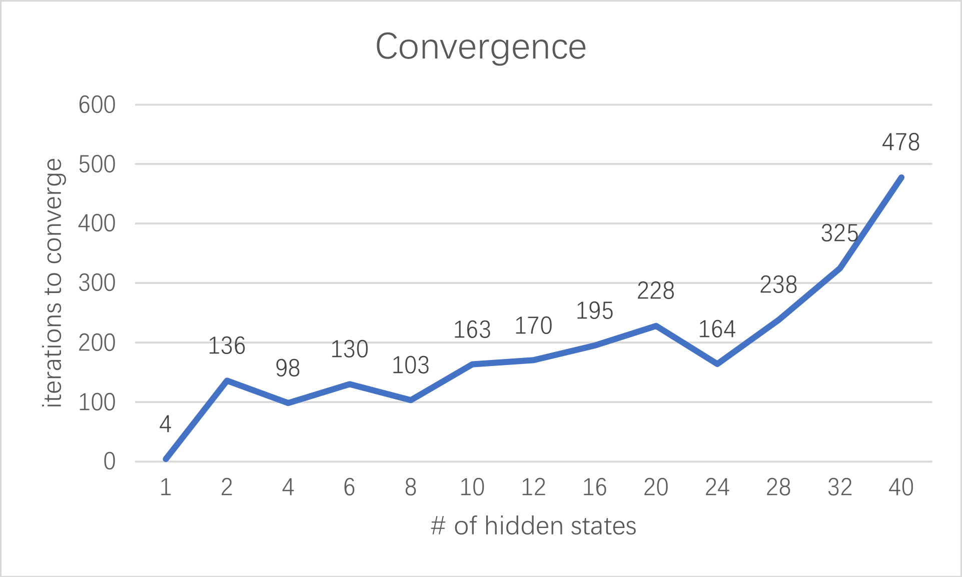
In summary, the hidden Markov models trained with hundreds of independent sequences generally perform poorly on classification except one with only one hidden states. An obvious possibility of overfitting is behind my test data, but any firm conclusion cannot be made due to insufficient supporting data.

Convergence

I define convergence by the concept of tolerance introduced earlier. As long as the improvement is less than a given number, the model is regarded as converged.

The following tests are run with 100 training files and 0.01 tolerance. By setting maximum number of iteration to 1000, the model will keep iterating by EM until it converges.

# of hidden states	1	2	4	6	8	10	12	16	20	24	28	32	40
iterations to converge	4	136	98	130	103	163	170	195	228	164	238	325	478



The data points do not show an apparent trend and the relationship between the number of hidden states and the number of iterations to converge remains uncertain. If the linear regression technique is employed to fit a line for this set of test data, it is likely to overfit. What I can conclude on this experiment is the number of iterations needed to converge generally increase with increasing number of hidden states, especially after a threshold has been reached, in this set of data 24-28 hidden states, yet no firm conclusion can be made.

How my program works

This is a supplementary information for you to grade my project in case you need to actually look at my code. The hmm model is build using two classes: HMM_one and HMM_all. HMM_one constructs a typical hidden Markov model with forward-backward algorithm implemented and carry the calculation of alpha, beta, gamma, digamma, scaling, etc. HMM_all will sum over the results of HMM_one trained on each independent sequence, re-estimate the model parameters by EM, and decide whether it should run the next iteration.

I wrote additional command line arguments to better test the codes. Please type python [file name] -h for this information and refer to readme.txt for more run instructions.