Question 6

(a) refer to hmm.py

(b)

```
PROBLEMS 14  OUTPUT  DEBUG CONSOLE  TERMINAL                                  1: Python
python.exe "c:/Users/darre/Documents/NUS/BT3102/Project/projectfiles forwardbackwards edition/projectfiles 131120 0103H/projectfiles/hmm.py
"
Naive prediction accuracy:     908/1378 = 0.6589259796806967
Naive prediction2 accuracy:    970/1378 = 0.7039187227866474
Viterbi prediction accuracy:   1012/1378 = 0.7343976777939042
Viterbi2 prediction accuracy:  1055/1378 = 0.7656023222060958
iter 0 prediction accuracy:    315/1378 = 0.22859216255442671
iter 10 prediction accuracy:   340/1378 = 0.2467343976777939
average squared error for 981 examples: 3.418960244648318
PS C:\Users\darre\Documents\NUS\BT3102\Project\projectfiles forwardbackwards edition\projectfiles 131120 0103H\projectfiles>
```

We get an accuracy of 0.229 after the random initialization

(c) We get an accuracy of 0.247 after 10 iterations of our forward-backward algorithm. This is a slight improvement from random initialization.

(d)

```
log_likelihood at iteration 1: -138512.02859052288
change in log likelihood from previous iteration: 138512.02859052288
log_likelihood at iteration 2: -116673.24573884734
change in log likelihood from previous iteration: 21838.78285167554
log_likelihood at iteration 3: -116036.89633145223
change in log likelihood from previous iteration: 636.3494073951151
log_likelihood at iteration 4: -115003.76888068602
change in log likelihood from previous iteration: 1033.127450766202
log_likelihood at iteration 5: -113705.65522369511
change in log likelihood from previous iteration: 1298.113656990914
log_likelihood at iteration 6: -112244.42507060622
change in log likelihood from previous iteration: 1461.2301530888944
log_likelihood at iteration 7: -110691.27637351886
change in log likelihood from previous iteration: 1553.148697087352
log_likelihood at iteration 8: -109138.82162432352
change in log likelihood from previous iteration: 1552.4547491953417
log_likelihood at iteration 9: -107523.4422993721
change in log likelihood from previous iteration: 1615.3793249514274
log_likelihood at iteration 10: -105813.58141941341
change in log likelihood from previous iteration: 1709.86087995868
```

We observe that the log likelihood converges rapidly in the first 4 iterations, before diverging slightly in the next 3 iterations, converging very slightly at iteration 8, then diverging again in the last 2 iterations.

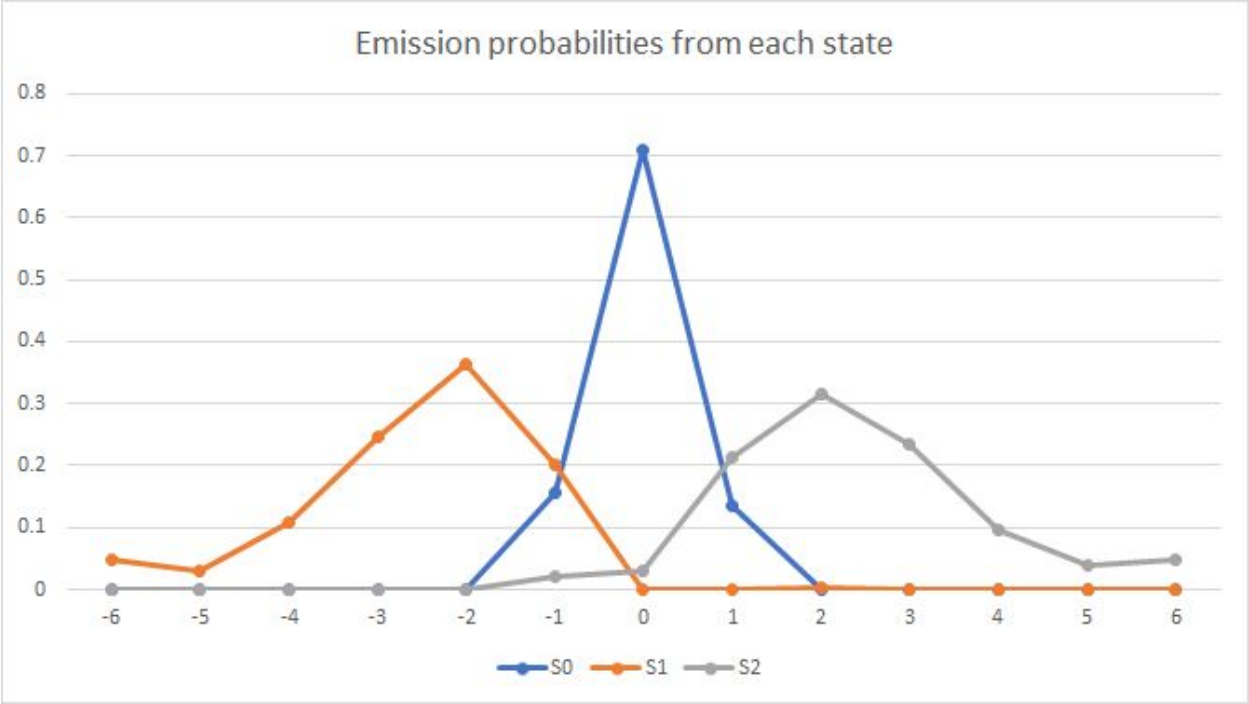
Question 7

(i) Reuse the forward-backward function you have implemented for POS tagging tweets to learn the output probabilities and transition probabilities for the training data in cat price changes train.txt. This function also accepts cat states.txt. Set the maximum iteration of the forward-backward algorithm to a large value (e.g., max iter = 100000) and the convergence threshold to a small value (e.g., thresh = 1e-4). This allows the forward-backward algorithm to run to convergence to get reasonable values of the output probabilities and transition probabilities. Write the output probabilities and transition probabilities to cat output probs.txt and cat trans probs.txt respectively. (The data and set of possible states should be small enough for you to run the algorithm to convergence if you have implemented it efficiently.)

(ii) Examine the output probabilities. For each state s (s0, s1, s2), provide the probabilities that it generates positive integers, zero, and negative integers, i.e., $P(0<x\leq 6|y=s), P(x=0|y=s), P(-6\leq x<0|y=s)$. What semantics has your HMM learned for each state (s0, s1, s2), i.e., what does each state represent?

State	Prob. positive	Prob. negative	Prob. zero
s0	0.135183	0.156704	0.708112
s1	0.00225	0.997422	0.000326
s2	0.94800	0.021954	0.02998

Our HMM learns that s1 is when the stock prices decrease, s0 is the state when stock prices remain the same,and s2 is the state when stock prices increase.



Comparing the probability graph for each emission to the initial distribution of price changes in cat_price_dev shows a roughly similar distribution peaking around 0 which indicates that our emission probabilities are likely to follow from the data we are given and are not abnormal.

(iii) Examine the transition probabilities. Using the semantics in your answer to the previous question, what has your HMM learned about the transitions from state to state?

Initial state	Next state	Prob (to 4d.p)
s0	s0	0.7318
	s1	0.0785
	s2	0.1367
s1	s0	0.2183
	s1	0.1084
	s2	0.6212
s2	s0	0.2439
	s1	0.6126
	s2	0.0913

As seen from the table above, the highest transitional probability for s0 is to s0, for s1 it is to s2, and for s2 it is to s1 at 0.7318, 0.6212 and 0.6126 respectively. This suggests that stock prices tend to stay stationary when prices do not change, increase in the future when the current stock prices are decreasing, and decrease in the future when the current prices are increasing.

(iv) After you have learned the output probabilities and transition probabilities, you are given a sequence of consecutive intraday stock price changes $x_1,x_2,...,x_n$. How can you use these probabilities to predict the next stock price change x_{n+1}?

We can implement a modified version of the Viterbi algorithm. The algorithm should generate the most likely sequence of tags, $y_1,y_2,...,y_n$ that emitted the stock price changes $x_1,x_2,...,x_n$. Then, we take the last predicted state, y_n, and retrieve the probabilities of it transitioning to each state in y_{n+1}.

To determine which state the final state yn transitions to, we perform prior sampling.
Let S be the set of all states, S = {s0, s1, s2}, and let s_p represent the previous state and s_n represent the next state after the transition. Then, let R(i, s_p) be the sum of the conditional probabilities of all s_n up to and including the i^{th} state in set S, given s_p. So, if s_p = s1:

$R(0, s1) = P(y_{n+1} = s0 \mid y_n = s1)$
$R(1, s1) = P(y_{n+1} = s0 \mid y_n = s1) + P(y_{n+1} = s1 \mid y_n = s1)$
$R(2, s1) = P(y_{n+1} = s0 \mid y_n = s1) + P(y_{n+1} = s1 \mid y_n = s1) + P(y_{n+1} = s2 \mid y_n = s1)$

Since these probabilities naturally add up to $\Sigma_{s \in S} P(yn+1 = s \mid yn = s_p)$ for any s_p, we generate a random number from 0 to $\Sigma_{s \in S} P(yn+1 = s \mid yn = s_p)$ and demarcate mutually exclusive ranges of numbers for each state. If the randomly generated number falls within the range ascribed to the particular resulting state $y_{n+1} = s_n$, we take it that the $y_n = s_p$ made a transition to $y_{n+1} = s_n$.

In other words,

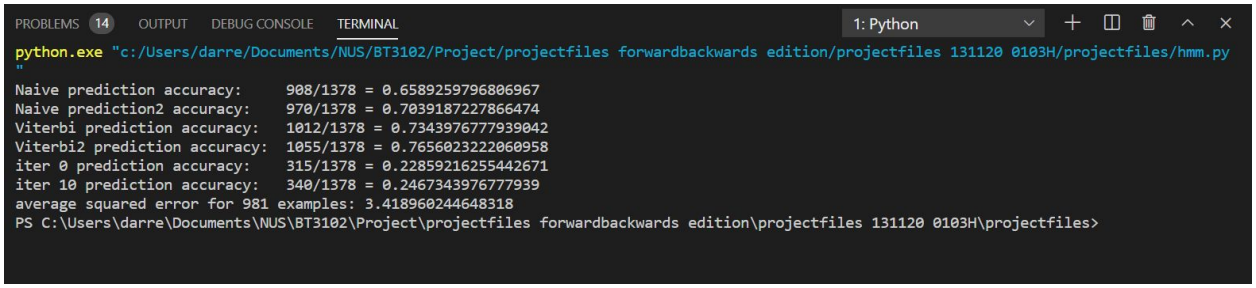
Randomly generated number r	We pick s_n to be
$0 <= r < R(0, s_p)$	s0
$R(0, s_p) <= r < R(1, s_p)$	s1
$R(1, s_p) <= r < R(2, s_p)$	s2

We do a similar thing for the output probabilities, conditioned on the resulting state s_n, and also use random numbers to sample from demarcated, mutually exclusive ranges of numbers. Based on the range selected, we will get the emission x_{n+1} to be generated.

(v) Write a function cat predict that implements your answer to the previous question. This function takes the probabilities in cat output probs.txt and cat trans probs.txt, and predicts the next price change for each sequence in cat price changes dev.txt. This function also accepts cat states.txt. Write your predictions to cat predictions.txt with one prediction per line.

(vi) What is the average squared error of your predictions? Naively predicting the next price change to be equal to the previous price change gives an average squared error of 3.94. Can you do better?

Yes, we can! :-) 3.42



(Remember to submit cat output probs.txt, cat trans probs.txt, and cat predictions.txt.)