A2_plotting

March 3, 2020

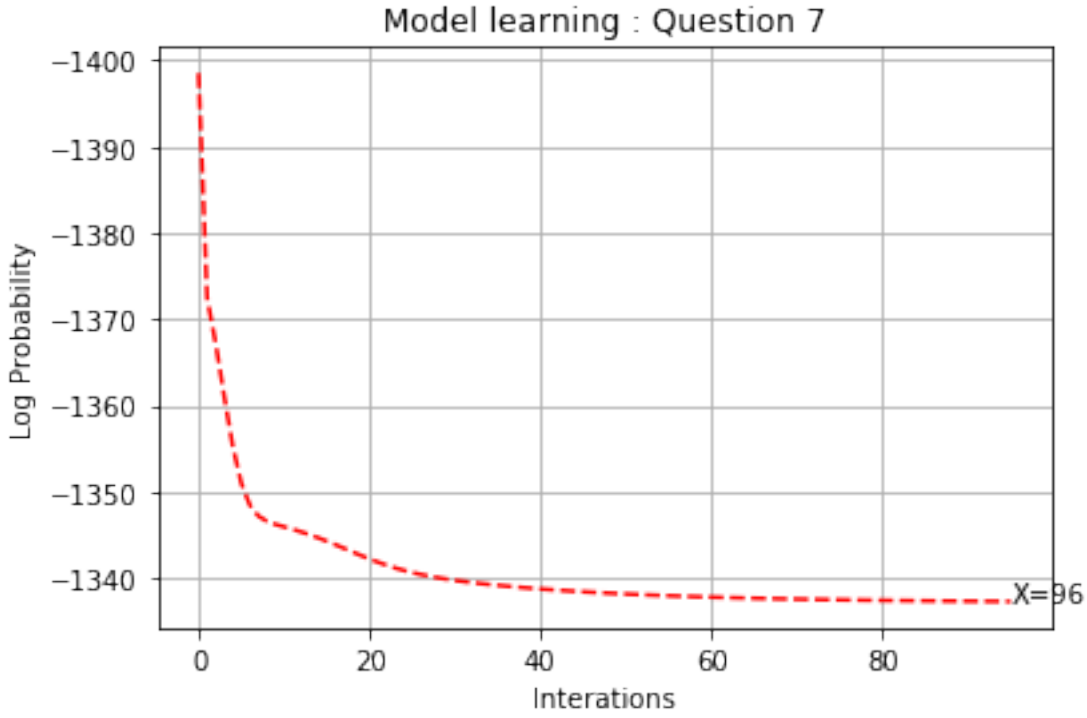
```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pickle

        def readAsArr(filename):
            file = open(filename,'r');
            line=file.readline();
            listArr=line.split(" ")
            x=np.asarray(listArr[:-1])
            x=x.astype(np.float)
            return x


        def plotting(x,title):
            fig, ax = plt.subplots()
            print('End Iterations='+str(x.size))
            print('Converged Log probability='+str(x[x.size-1]))
            ax.plot(x,'r--')
            text='X='+str(x.size)
            ax.annotate(text,xy=(x.size-1,x[x.size-1]))
            ax.set_xlabel('Interations')
            ax.set_ylabel('Log Probability')
            ax.set_title('Model learning : '+title)
            ax.grid(True)
            plt.gca().invert_yaxis()
```

0.0.1 With T=1000

Question 7 The algorithm converges as seen from the learning curve below.The total iterations were 96. Convergence is obtained when the algorithm settles to run at a particular value and no further improvement on that value can be obtained.From the graph below the algorithm hits a saturation at log probability of around -1337 and keeps that value in next iterations.

```
In [2]: x=readAsArr('q7')
        plotting(x,"Question 7")
```

```
End Iterations=96
Converged Log probability=-1337.2786220415824
```



Question 8 The initialisation of all matrices was done randomly.Although the algorithm converged,it took more iterations than before and also the values do not match very closely to the original values.THis could be due to the algorithm getting stuck at a local maxima.Below are the values of each matrix estimated after convergence and also the learning graph.

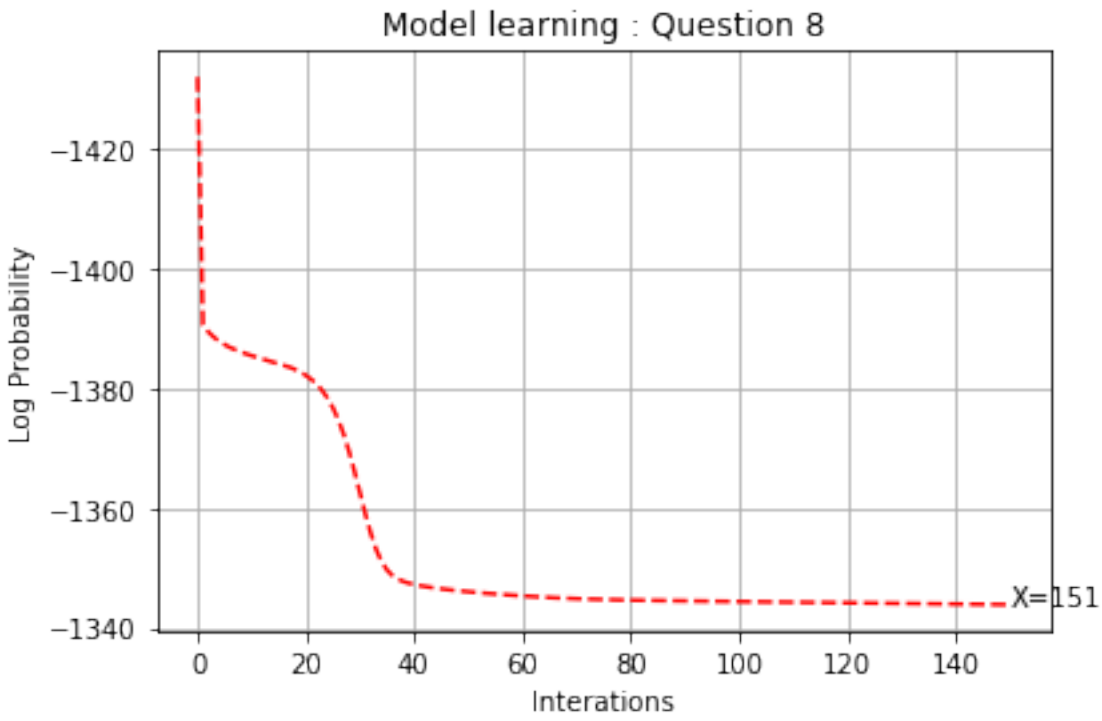
A =[0.217 0.045 0.737 0.110 0.826 0.062 0.25 0.569 0.180]

B = [0.576 0.278 0.097 0.047 0.004 0.296 0.318 0.381 0.802 0.057 0.062 0.077]

P = [0 0 0.999]

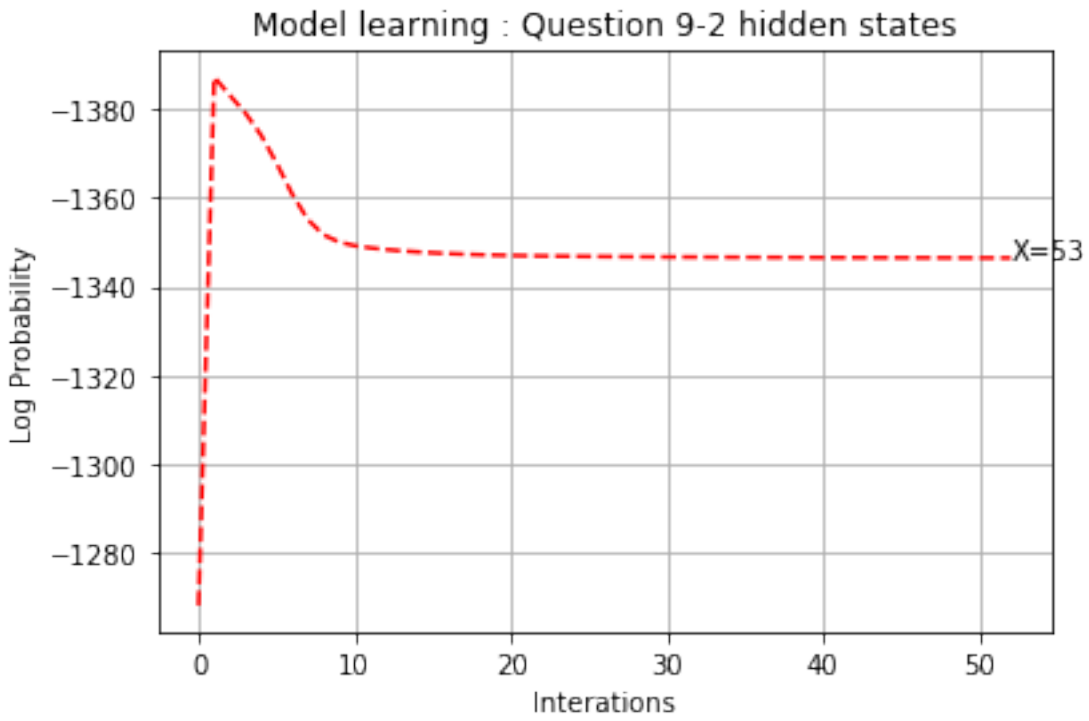
```
In [3]: x=readAsArr('q8')
        plotting(x,"Question 8")
```

```
End Iterations=151
Converged Log probability=-1343.8724472460804
```



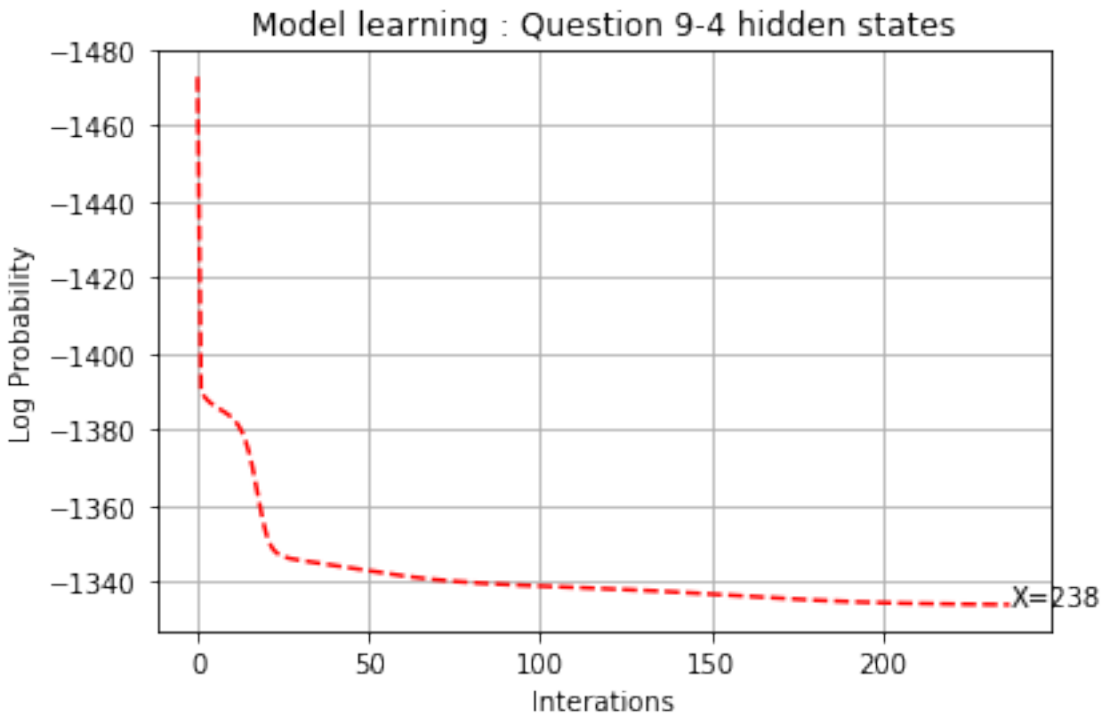
Question 9 9a. 2 hidden states: With 2 hidden states the algorithm seems to diverge initially,but convergences later ,but a with a little lower log probability than with 3 hidden states as seen before.

```
In [4]: x=readAsArr('q9-2')
        plotting(x,"Question 9-2 hidden states")

End Iterations=53
Converged Log probability=-1346.4169525028515
```



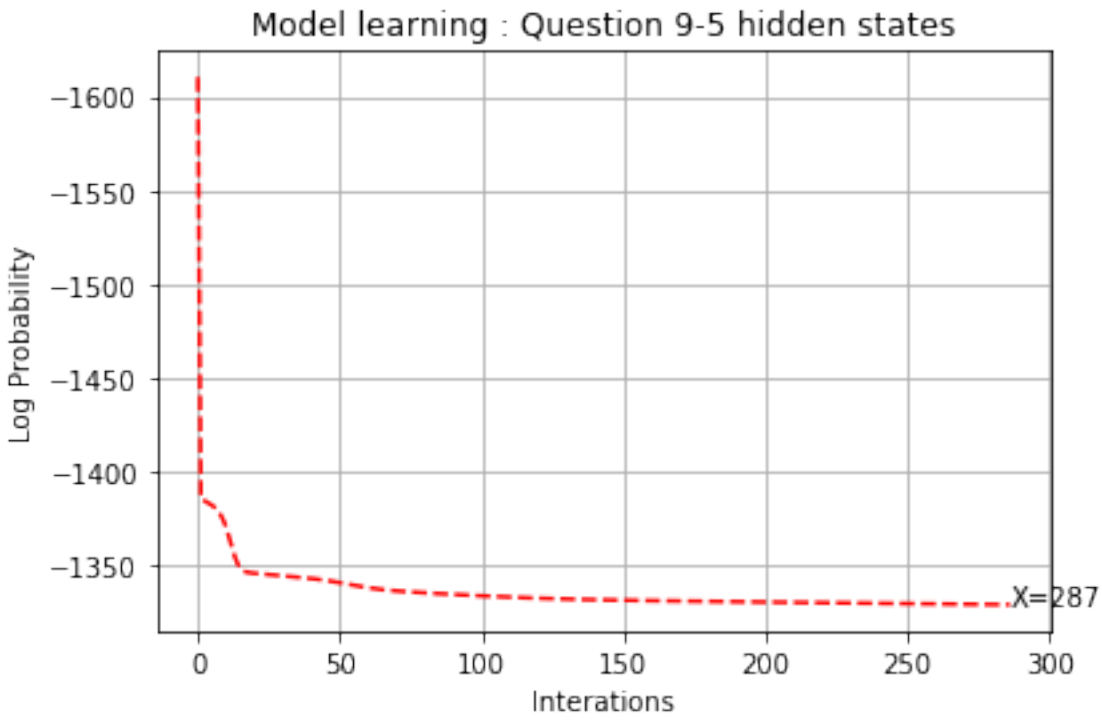
9b. 4 hidden states: With 4 hidden states,the algorithm converges smoothly with higher log probability than 2 hidden states.

```
In [5]: x=readAsArr('q9-4')
        plotting(x,"Question 9-4 hidden states")

End Iterations=238
Converged Log probability=-1333.88677277289
```



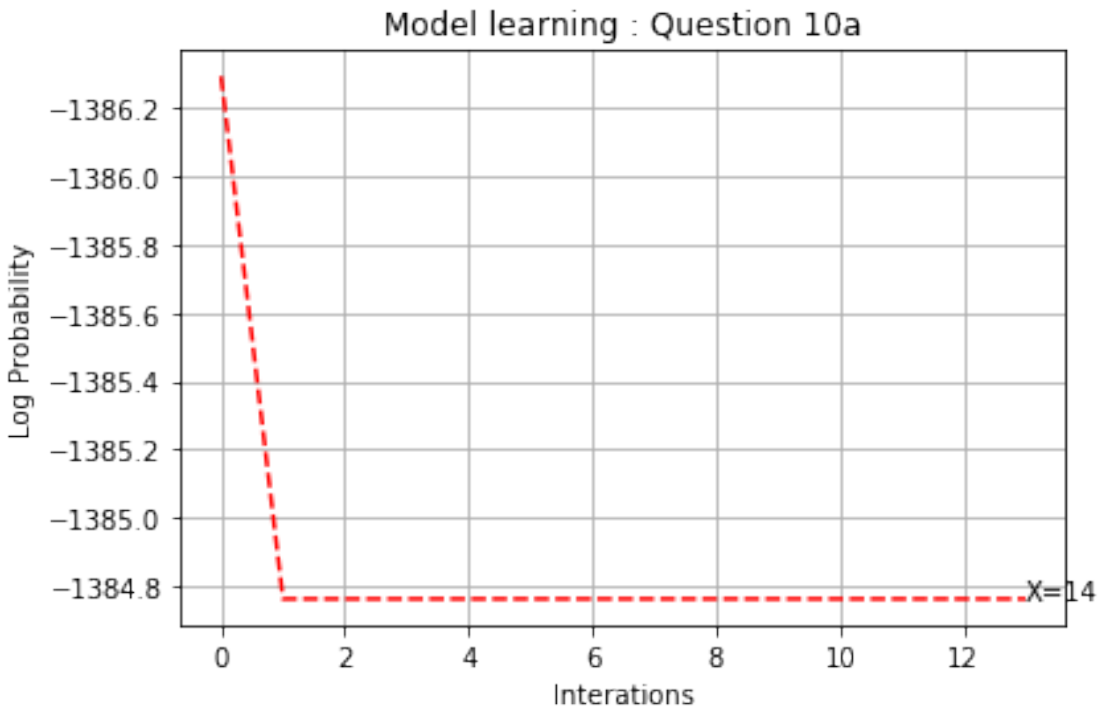
9c. 5 hidden states:When tried with 5 hidden states sharply moves towards convergence higher log probability.This shows that the number of hidden states and emissions play an important role in the convergence of the algorithm.

```
In [6]: x=readAsArr('q9-5states')
        plotting(x,"Question 9-5 hidden states")

End Iterations=287
Converged Log probability=-1328.9074347887688
```



Question 10 a 10a. Uniform distribution initialisation :With uniformly distributed matrices,the algorithm learning almost follows a linear curve and converges rather fast,but to lower log probability value.

```
In [7]: x=readAsArr('q10-a')
        plotting(x,"Question 10a")

End Iterations=14
Converged Log probability=-1384.7622487829913
```



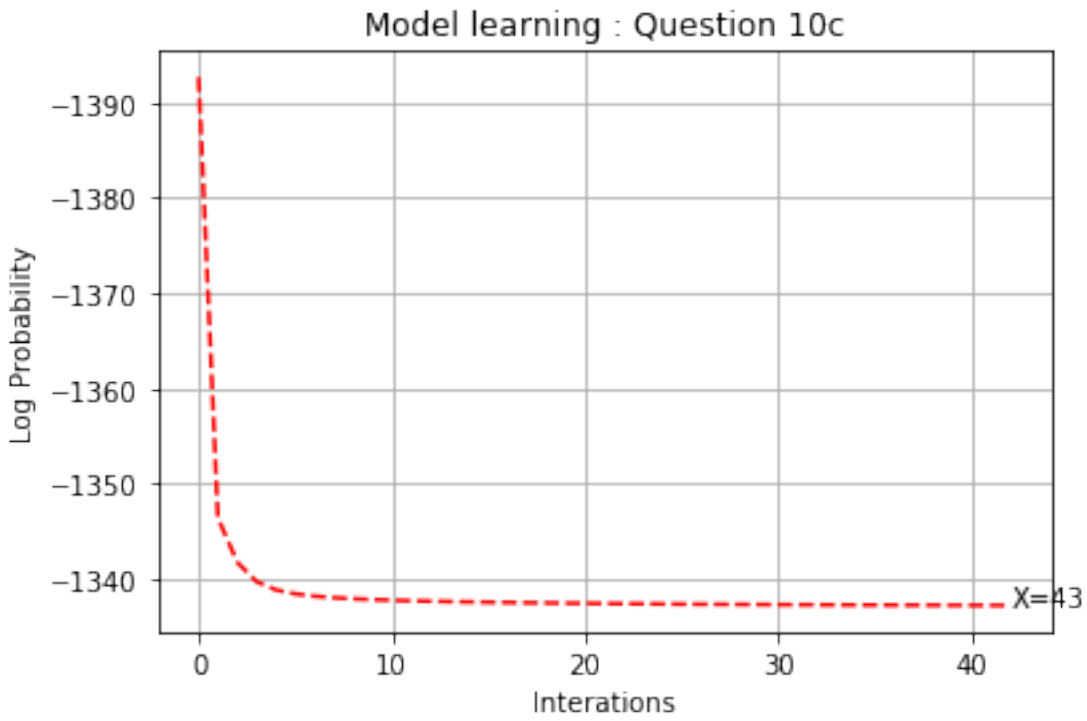
Question 10 b

b. Diagonal A matrix :With a diagonal A matrix,the learning did converge to a real values.This can be apparent since with a digonal matrix,since the probability could be 1 at one state and others would be zero,the transition of the states are never possible and the state remains in itself always.

Question 10 c

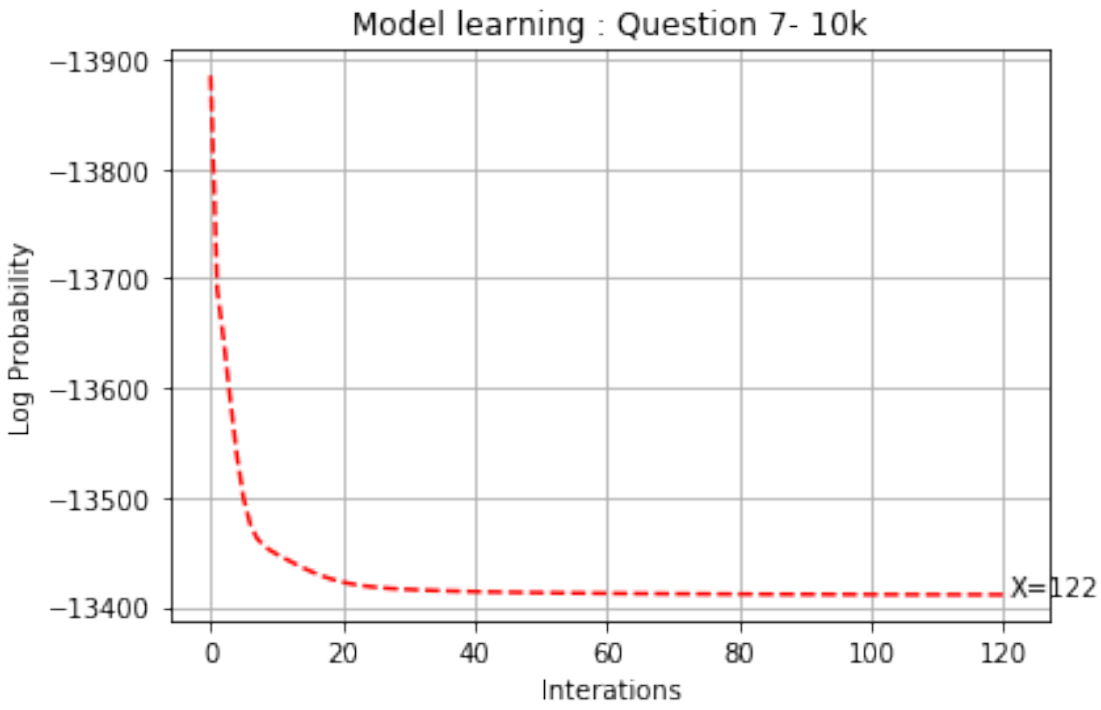
c. Initialization close to solution : The learnig curve is smooth and the algorithm converges to a higher log probability value in few iterations.This shows that if the initialisation is done to the actual values as close as possible,the algorithm estimation is fast and more accurate.

```
In [8]: x=readAsArr('q10-c')
        plotting(x,"Question 10c")

End Iterations=43
Converged Log probability=-1337.187761994451
```

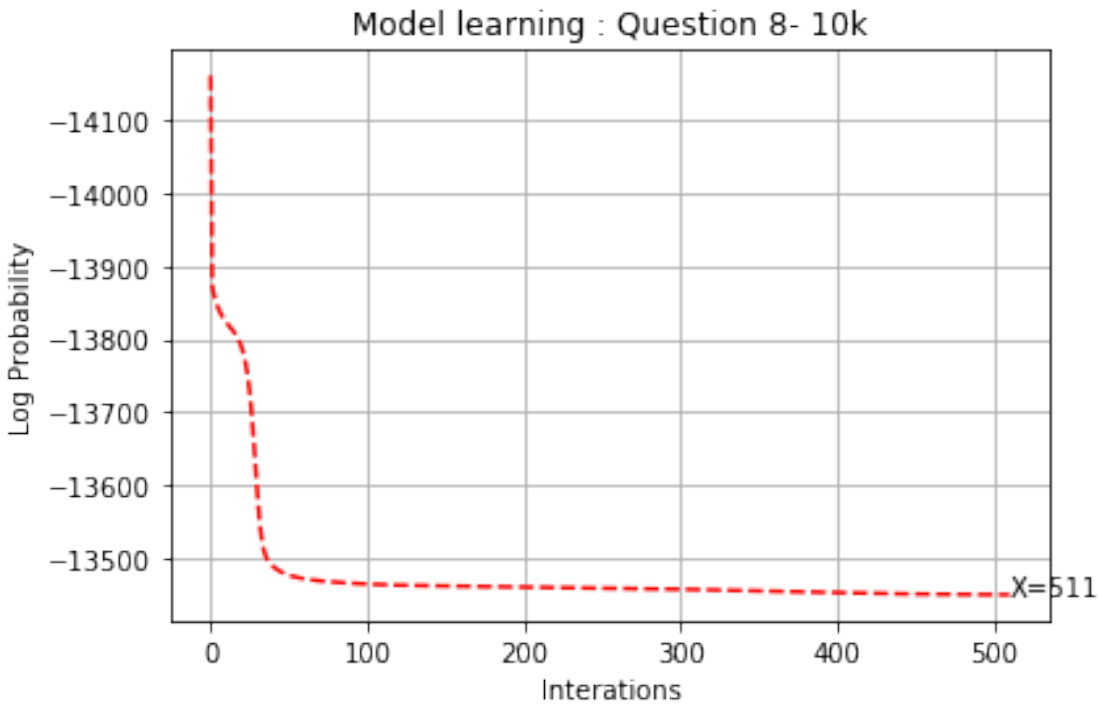


0.0.2 With T=10000

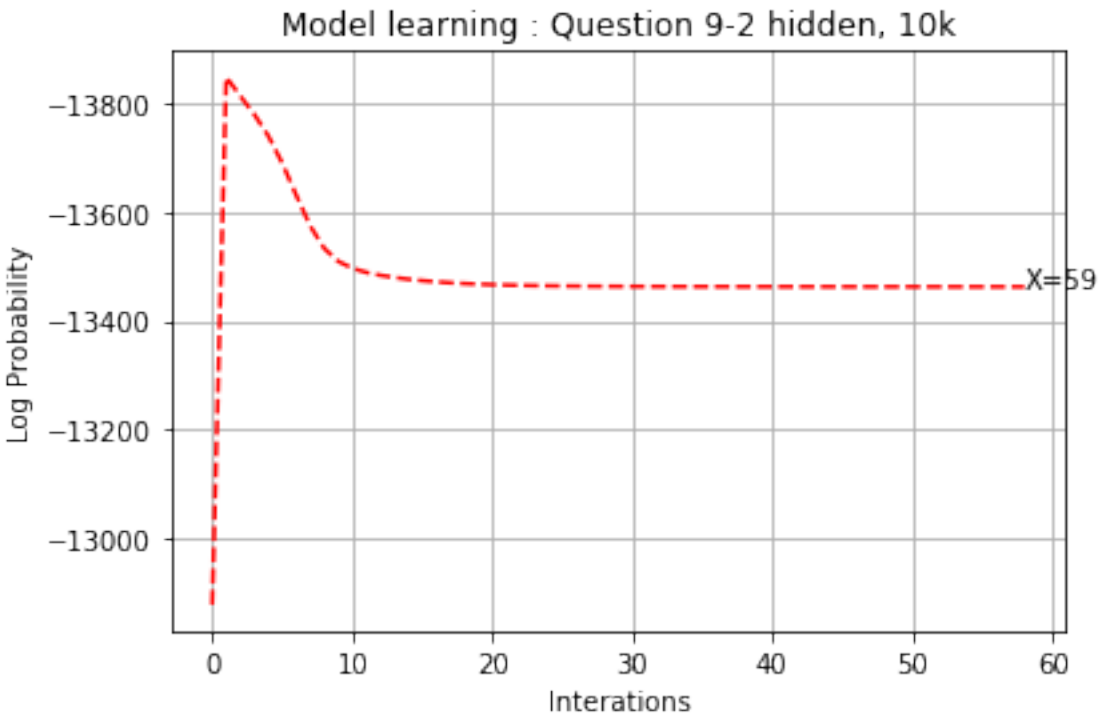
```
In [9]: x=readAsArr('q7-10k')
        plotting(x,"Question 7- 10k")

End Iterations=122
Converged Log probability=-13411.152157143357
```



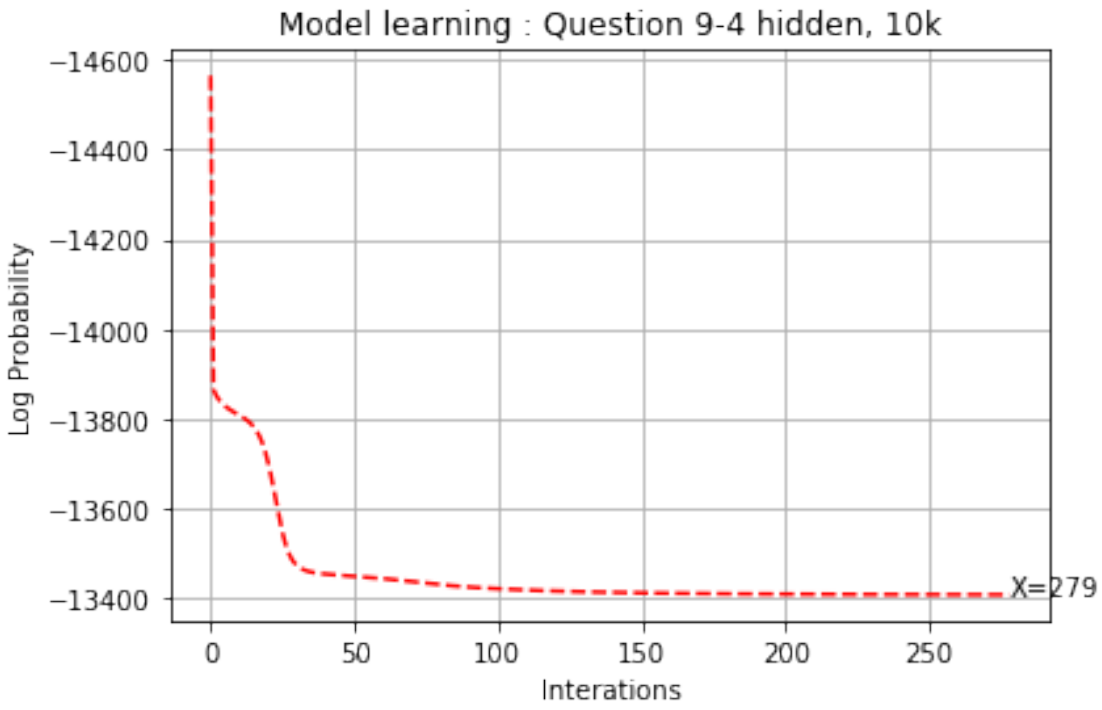

```
In [10]: x=readAsArr('q8-10k')
         plotting(x,"Question 8- 10k")

End Iterations=511
Converged Log probability=-13450.487387884183
```




```
In [11]: x=readAsArr('q9-2_10k')
         plotting(x,"Question 9-2 hidden, 10k")

End Iterations=59
Converged Log probability=-13462.770322729091
```



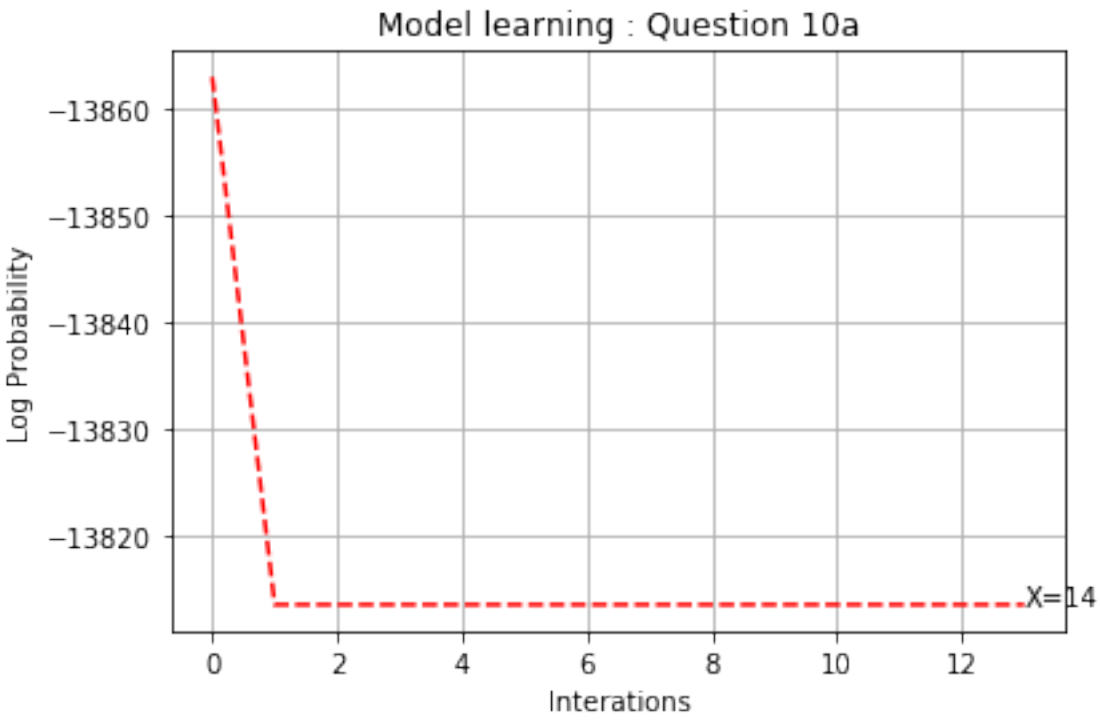

```
In [12]: x=readAsArr('q9-4_10k')
         plotting(x,"Question 9-4 hidden, 10k")

End Iterations=279
Converged Log probability=-13409.23080182654
```



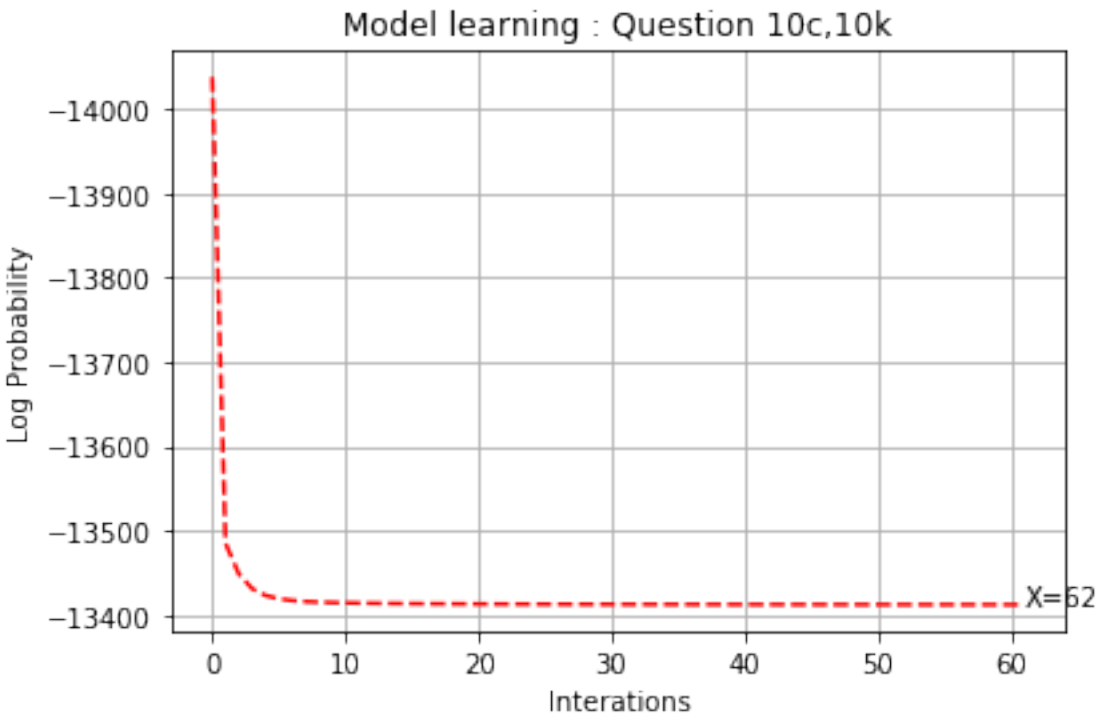
In [ ]:

In [13]:
```
x=readAsArr('q10-a-10k')
plotting(x,"Question 10a")
```

```
End Iterations=14
Converged Log probability=-13813.655738944082
```




```
In [14]: x=readAsArr('q10-c-10k')
         plotting(x,"Question 10c,10k")

End Iterations=62
Converged Log probability=-13411.867956853732
```


Model learning : Question 10c,10k
Log Probability
−14000
−13900
−13800
−13700
−13600
−13500
−13400
0
10
20
30
40
50
60
Interations
X=62

In [ ]:

In [ ]: