1 Class-Conditional Densities for Binary Data

Problem A.

(i) Using the chain rule of probability to factorize $p(x|y)$ and letting $\theta_{xjc} = P(x_j|x_{1,...j-1}, y = c)$, we observe that for a fixed j, we can write

$$p(x_j|x_{j-1}, ...x_1)$$

as

$$p(x_j = a_j|x_1 = a_1, x_2 = a_2, ..., x_{j-1} = a_{j-1})$$

for which there would be 2^j such probabilities. In other words, the number of θ_{xj_0c} is 2^{j_0} assuming a fixed j_0. The number of parameters needed to represent the factorization asked, using big-O notation can be expressed by:

$$\sum_{j_0=1}^{D} 2^{j_0} \times C \approx 2^{D+1} \times C \approx O(2^D C)$$

Assuming we store each θ_{xjc}, the number of parameters needed to represent this factorization is: $O(2^D C)$

(ii) Assuming we did no such factorization and just used the joint probability $p(x|y = c)$, the number of parameters needed to estimate in order to be able to compute $p(x|y = c)$ for arbitrary x and c is: $O(2^D C)$. This answer is the same as my answer in the previous part.

Problem B.

If the sample size N is very small, the Naive Bayes model would give a lower test set error. The full model would be more prone to overfitting on a small dataset since the full model uses more parameters.

Problem C.

If the sample size N is very large, the full model would give a lower test set error. The Naive Bayes model would be more prone to underfitting on a large dataset while the full model uses more parameters and would most likely produce more accurate results, giving a lower test set error. Since the dataset is large, the full model would also be less likely to overfit than before in problem B.

Problem D.

The computational complexity of making a prediction using Naive Bayes for a single test case is: O(CD). For each of the C classes we must compute the independent likelihoods of which there are D of and then multiply them all together. Therefore, the computational complexity is O(CD).
The computational complexity of making a prediction with the full model for a single test case is: O(CD). We are told that for the full-model case, converting a D-bit vector to an array index is an O(D) operation and we must do this for each of the C classes. We recall that we have assumed a uniform class prior. Therefore, the computational complexity is O(CD).

2 Sequence Prediction

Problem A.

See Jupyter Notebook for code. The max-probability state sequence for each of the five input sequences at the end of the corresponding file:

File #0:

Emission Sequence	Max Probability State Sequence
25421	31033
01232367534	22222100310
5452674261527433	1031003103222222
7226213164512267255	1310331000033100310
0247120602352051010255241	2222222222222222222222103

File #1:

Emission Sequence	Max Probability State Sequence
77550	22222
7224523677	2222221000
505767442426747	222100003310031
72134131645536112267	10310310000310333100
4733667771450051060253041	2221000003222223103222223

File #2:

Emission Sequence	Max Probability State Sequence
60622	11111
4687981156	2100202111
815833657775062	021011111111111
21310222515963505015	02020111111111111021
6503199452571274006320025	1110202111111102021110211

File #3:

Emission Sequence	Max Probability State Sequence
13661	00021
2102213421	3131310213
166066262165133	133333133133100
53164662112162634156	20000021313131002133
1523541005123230226306256	1310021333133133133133133

File #4:

Emission Sequence	Max Probability State Sequence
23664	01124
3630535602	0111201112
350201162150142	011244012441112
00214005402015146362	11201112412444011112
2111266524665143562534450	2012012424124011112411124

File #5:

Emission Sequence	Max Probability State Sequence
68535	10111
4546566636	1111111111
638436858181213	110111010000011
13240338308444514688	00010000000111111100
0111664434441382533632626	2111111111111100111110101

Problem B.

(i) See Jupyter notebook for code. Probabilities of emitting the five-input sequences at the end of the corresponding file using the Forward algorithm:

File #0:

Emission Sequence	Probability of Emitting Sequence
25421	4.537e-05
01232367534	1.620e-11
5452674261527433	4.348e-15
7226213164512267255	4.739e-18
0247120602352051010255241	9.365e-24

File #1:

Emission Sequence	Probability of Emitting Sequence
77550	1.181e-04
7224523677	2.033e-09
505767442426747	2.477e-13
72134131645536112267	8.871e-20
4733667771450051060253041	3.740e-24

File #2:

Emission Sequence	Probability of Emitting Sequence
60622	2.088e-05
4687981156	5.181e-11
815833657775062	3.315e-15
21310222515963505015	5.126e-20
6503199452571274006320025	1.297e-25

File #3:

Emission Sequence	Probability of Emitting Sequence
13661	1.732e-04
2102213421	8.285e-09
166066262165133	1.642e-12
53164662112162634156	1.063e-16
1523541005123230226306256	4.535e-22

File #4:

Emission Sequence	Probability of Emitting Sequence
23664	1.141e-04
3630535602	4.326e-09
350201162150142	9.793e-14
00214005402015146362	4.740e-18
2111266524665143562534450	5.618e-22

File #5:

Emission Sequence	Probability of Emitting Sequence
68535	1.322e-05
4546566636	2.867e-09
638436858181213	4.323e-14
13240338308444514688	4.629e-18
0111664434441382533632626	1.440e-22

(ii) See Jupyter notebook for code. Probabilities of emitting the five-input sequences at the end of the corresponding file using the Backward algorithm (same as using the Forward Algorithm):

File #0:

Emission Sequence	Probability of Emitting Sequence
25421	4.537e-05
01232367534	1.620e-11
5452674261527433	4.348e-15
7226213164512267255	4.739e-18
0247120602352051010255241	9.365e-24

File #1:

Emission Sequence	Probability of Emitting Sequence
77550	1.181e-04
7224523677	2.033e-09
505767442426747	2.477e-13
72134131645536112267	8.871e-20
4733667771450051060253041	3.740e-24

File #2:

Emission Sequence	Probability of Emitting Sequence
60622	2.088e-05
4687981156	5.181e-11
815833657775062	3.315e-15
21310222515963505015	5.126e-20
6503199452571274006320025	1.297e-25

File #3:

Emission Sequence	Probability of Emitting Sequence
13661	1.732e-04
2102213421	8.285e-09
166066262165133	1.642e-12
53164662112162634156	1.063e-16
1523541005123230226306256	4.535e-22

File #4:

Emission Sequence	Probability of Emitting Sequence
23664	1.141e-04
3630535602	4.326e-09
350201162150142	9.793e-14
00214005402015146362	4.740e-18
2111266524665143562534450	5.618e-22

File #5:

Emission Sequence	Probability of Emitting Sequence
68535	1.322e-05
4546566636	2.867e-09
638436858181213	4.323e-14
13240338308444514688	4.629e-18
0111664434441382533632626	1.440e-22

Problem C.

See Jupyter Notebook for code. The learned state transition and the output emission matrices are:

```
##########################################################################
                      Running Code For Question 2C
##########################################################################


Transition Matrix:
##########################################################################
2.833e-01   4.714e-01   1.310e-01   1.143e-01
2.321e-01   3.810e-01   2.940e-01   9.284e-02
1.040e-01   9.760e-02   3.696e-01   4.288e-01
1.883e-01   9.903e-02   3.052e-01   4.075e-01


Observation Matrix:
##########################################################################
1.486e-02   2.288e-02   1.533e-02   1.179e-02   4.717e-03   5.189e-03   2.830e-03   1.297e-02   9.198e-03   2.358e-04

1.062e-02   9.653e-04   1.931e-03   3.089e-03   1.699e-02   4.633e-03   1.409e-02   2.394e-02   1.371e-02   1.004e-02

1.194e-02   4.299e-03   6.529e-03   9.076e-03   1.768e-02   2.022e-02   4.618e-03   5.096e-03   7.803e-03   1.274e-02

1.694e-02   3.871e-03   1.468e-02   1.823e-02   4.839e-03   6.290e-03   9.032e-03   2.581e-03   2.161e-02   1.935e-03
```

Problem D.

See Jupyter Notebook for code. The learned state transition and the output emission matrices are:

```
######################################################################
                     Running Code For Question 2D
######################################################################


Transition Matrix:
######################################################################
3.062e-16   5.885e-05   2.355e-02   9.764e-01
1.634e-10   4.762e-01   4.478e-01   7.603e-02
9.196e-01   7.604e-02   8.671e-12   4.384e-03
6.021e-06   6.045e-01   3.955e-01   1.913e-09


Observation Matrix:
######################################################################
1.229e-01   2.017e-02   5.762e-02   1.853e-01   1.639e-01   7.472e-02   1.338e-01   3.066e-02   1.639e-01   4.708e-02

1.149e-01   1.528e-01   8.055e-02   3.553e-05   1.656e-01   2.297e-17   7.575e-08   2.449e-01   1.201e-01   1.211e-01

1.145e-01   7.329e-12   1.073e-01   1.864e-01   1.039e-13   2.141e-01   9.499e-02   6.365e-02   1.402e-01   7.877e-02

1.973e-01   7.784e-02   1.338e-01   9.754e-02   1.139e-01   1.254e-01   1.093e-01   2.740e-02   1.175e-01   1.153e-33
```

Problem E.

The transition and emission matrices from 2C and 2D are quite different – the matrices for 2D hold values that spread over a greater range and magnitude, much sparser. Most of the values in the observation matrix and transition matrix for 2C are in the magnitude range of e-03 to e-01 (see previous problems) while values in the observation matrix and transition matrix for 2D are in the magnitude range of e-16 to e-01 (see previous problems). Therefore, we see that both the observation and transition matrices produced from unsupervised learning range over a greater spread. The supervised learning matrices from 2C would be a more accurate representation of Ron's moods and how they affect his music choices.

Problem F.

See Jupyter notebook for code. Using the six models to probabilistically generate five sequences of emissions from each model, each of length 20:

File #0:
Generated Emission
##
01647037427220364422
53352574156766127422
77052435530775664776
27114455746004725554
40752167204202400235

File #1:
Generated Emission
##
05312654242645652255
40743157256435544404

40063577434065154777
42237355571157531505
51435242766757526177

File #2:
Generated Emission
###
96579235952538977969
58555152522857777521
57705876073218563905
33121031697160597952
55710277359022615912

File #3:
Generated Emission
###
15652525001511234036
33561124252410152136
10320214640000255652
21000241166545026626
11100203512151435601

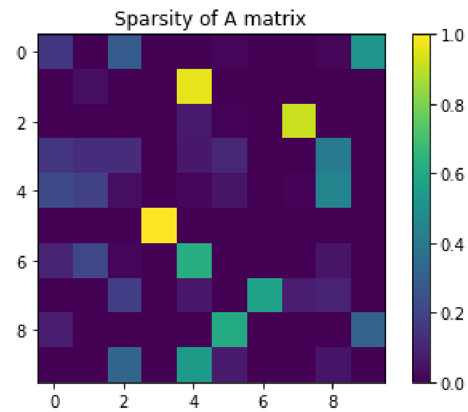
File #4:
Generated Emission
###
56440230205431064556
36463533214232006040
52612643501664224655
12563054306263333340
40426656044353365252

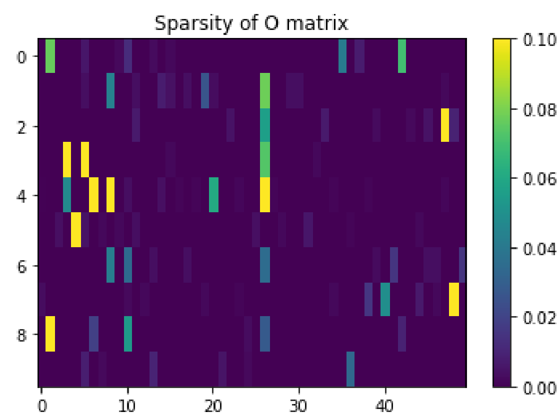
File #5:
Generated Emission
###
13008354171836834520
56424304568382068413
11818106685842313244
64545826838536015168

41012382038113213588

Problem G.

The trained A and O matrices are sparse. The sparsity affects the transition and observation behavior at each state. A very small value in the matrices indicates that the likelihood of moving to or from particular states, or observing a particular emission from state is very small.





Problem H.

The sample emission sentences from the HMM grow more and more grammatically correct as the number of hidden states increase, to an extent. When there is only one hidden state, words are independently sampled from a stationary distribution repeatedly so the HMM is essentially repeating itself. When the number of hidden states is unknown while training an HMM for a fixed observation set, we can increase the training data likelihood by allowing more hidden states. This is because, although not optimal, we could allow a hidden state for each output in the training sequence which would then achieve near perfect training data likelihood.

Problem I.

A state that I find semantically meaningful is state 9. The wordcloud indicates what you would expect of the Constitution as a whole with the top words being ”state,” ”congress,” president,” etc. This state differs from the other states because it bears the most striking resemblance to the wordcloud that was generated when visualizing the entirety of the Constitution (this was the first visualization created in the Jupyter notebook). Most of the words are important nouns related to the U.S. government, and these words are clumped together which makes sense.