1 Class-Conditional Densities for Binary Data [25 Points]

This problem will test your understanding of probabilistic models, especially Naive Bayes. Consider a generative classifier for C classes, with class conditional density $p(x|y)$ and a uniform class prior $p(y)$. Suppose all the D features are binary, $x_j \in \{0,1\}$. If we assume all of the features are conditionally independent, as in Naive Bayes, we can write:

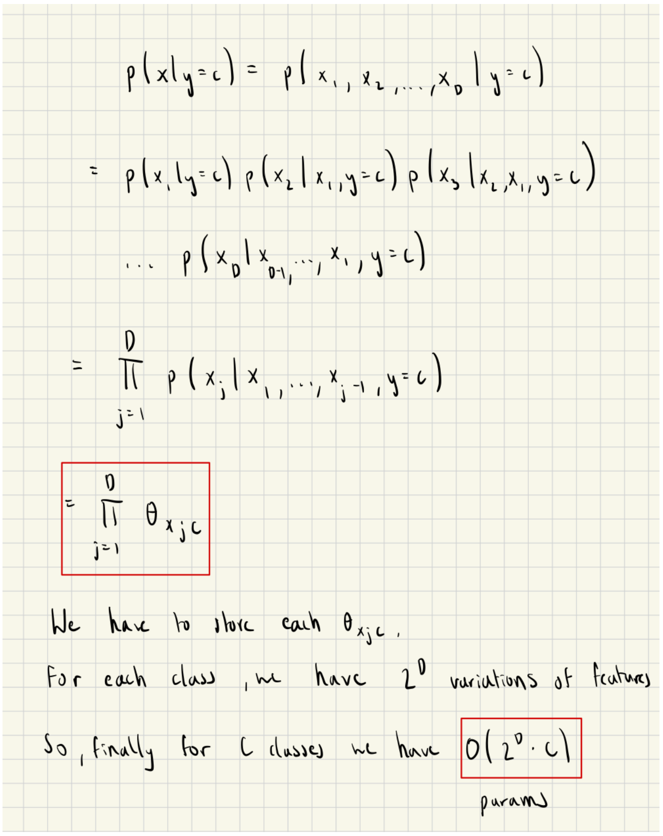
$$p(x \mid y = c) = \prod_{j=1}^{D} P(x_j \mid y = c)$$

This requires storing DC parameters.

Now consider a different model, which we will call the 'full' model, in which all the features are fully *dependent*.

Problem A [5 points]: Use the chain rule of probability to factorize $p(x \mid y)$, and let $\theta_{xjc} = P(x_j|x_{1,\ldots,j-1}, y = c)$. Assuming we store each θ_{xjc}, how many parameters are needed to represent this factorization? Use big-O notation.

Solution A.:



$$p(x|y=c) = p(x_1, x_2, \ldots, x_D \mid y=c)$$

$$= p(x_1|y=c)\, p(x_2|x_1, y=c)\, p(x_3|x_2, x_1, y=c)$$

$$\ldots\, p(x_D|x_{D-1}, \ldots, x_1, y=c)$$

$$= \prod_{j=1}^{D} p(x_j|x_1, \ldots, x_{j-1}, y=c)$$

$$\boxed{= \prod_{j=1}^{D} \theta_{xjc}}$$

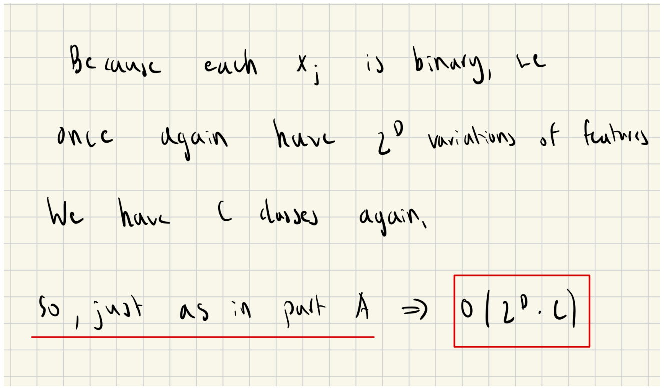
We have to store each θ_{xjc}.

For each class, we have 2^D variations of features

So, finally for C classes we have $\boxed{O(2^D \cdot C)}$ params

Problem B [5 points]: Assume we did no such factorization, and just used the joint probability $p(x \mid y = c)$. How many parameters would we need to estimate in order be able to compute $p(x|y = c)$ for arbitrary x and c? How does this compare to your answer from the previous part? Again, use big-O notation.

Solution B.:



Because each x_j is binary, we once again have 2^D variations of features

We have C classes again.

<u>So, just as in part A</u> $\Rightarrow \boxed{O(2^D \cdot C)}$

Problem C [2 points]: Assume the number of features D is fixed. Let there be N training cases. If the sample size N is very small, which model (Naive Bayes or full) is likely to give lower test set error, and why?

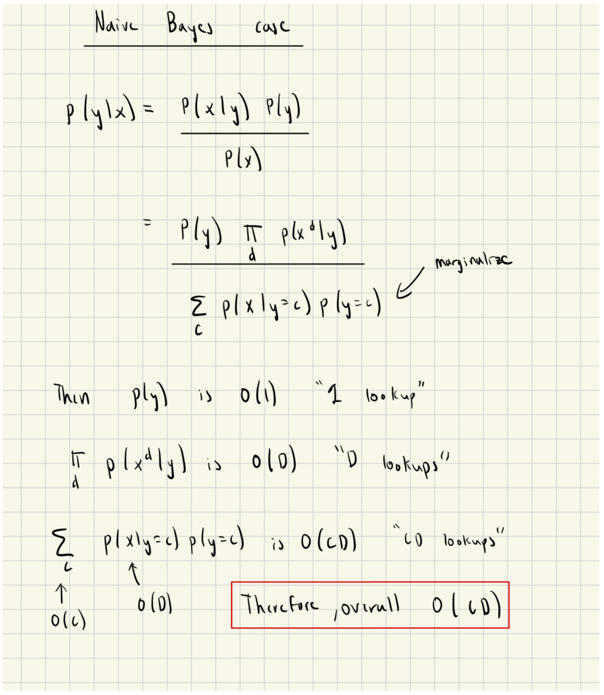
Solution C.: *If the sample size N is very small, the Naive Bayes model is likely to give lower test set error. This is because the full model, trying to capture the dependency between features, will learn too many parameters for such a small training set, and thus will overfit to the data.*

Problem D [2 points]: If the sample size N is very large, which model (Naive Bayes or full) is likely to give lower test set error, and why?

Solution D.: *Our answer is the exact opposite from Part C. If the sample size N is very large, the full model is likely to give lower test set error. This is because the full model, with a large sample size, can accurately model the dependency between features, and therefore it will avoid overfitting the data. The Naive Bayes model on the flip side will likely underfit the data.*

Problem E [11 points]: Assume all the parameter estimates have been computed. What is the computational complexity of making a prediction, i.e. computing $p(y \mid x)$, using Naive Bayes for a single test case? What is the computation complexity of making a prediction with the full model? For the full-model case, assume that converting a D-bit vector to an array index is an $O(D)$ operation. Also, recall that we have assumed a uniform class prior.

Solution E.:



Naive Bayes case

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

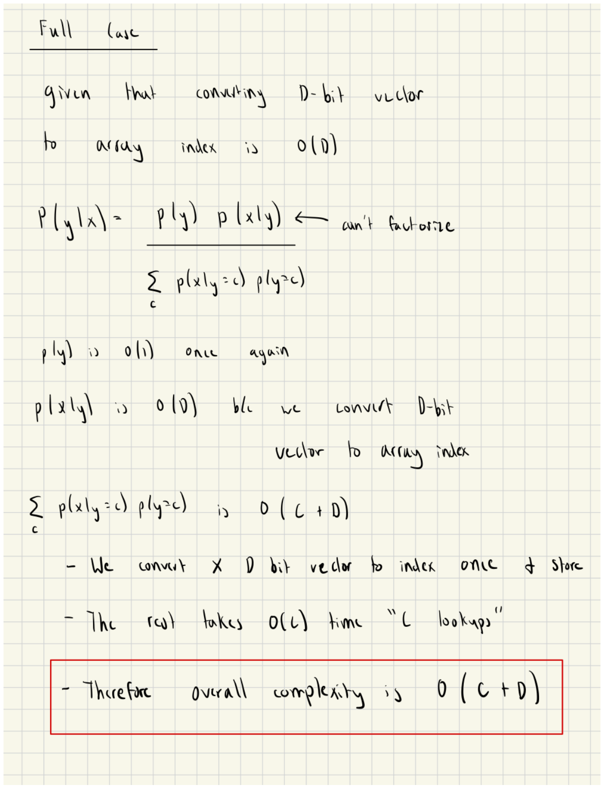
$$= \frac{p(y) \prod_d p(x^d|y)}{\sum_c p(x|y=c)\,p(y=c)} \swarrow \text{marginalize}$$

Then $p(y)$ is $O(1)$ "1 lookup"

$\prod_d p(x^d|y)$ is $O(D)$ "D lookups"

$\underset{\substack{\uparrow \\ O(C)}}{\sum_c} \underset{\substack{\uparrow \\ O(D)}}{p(x|y=c)}\, p(y=c)$ is $O(CD)$ "CD lookups"

$\boxed{\text{Therefore, overall } O(CD)}$



Full case

given that converting D-bit vector to array index is $O(D)$

$$p(y|x) = \frac{p(y)\,p(x|y)}{\sum_c p(x|y=c)\,p(y=c)} \leftarrow \text{can't factorize}$$

$p(y)$ is $O(1)$ once again

$p(x|y)$ is $O(D)$ b/c we convert D-bit vector to array index

$\sum_c p(x|y=c)\,p(y=c)$ is $O(C+D)$

- We convert x D bit vector to index once & store
- The rest takes $O(C)$ time "C lookups"

$\boxed{\text{- Therefore overall complexity is } O(C+D)}$

2 Sequence Prediction [75 Points]

In this problem, we will explore some of the various algorithms associated with Hidden Markov Models (HMMs), as discussed in lecture. For these problems, make sure you are **using Python 3** to implement the algorithms. Please see the HMM notes posted on the course website—they will be helpful for this problem!

Sequence Prediction

These next few problems will require extensive coding, so be sure to start early! We provide you with eight different files:

- You will write all your code in `HMM.ipynb`, within the appropriate functions where indicated by "TODO" in the comments. There is no need to modify anything else aside from what is indicated. There should be no need to write additional functions or use NumPy in your implementation, but feel free to do so if you would like.

The supplementary data folder contains 6 files titled `sequence_data0.txt`, `sequence_data1.txt`, ... , `sequence_data5.txt`. Each file specifies a **trained** HMM. The first row contains two tab-delimited numbers: the number of states Y and the number of types of observations X (i.e. the observations are $0, 1, ..., X-1$). The next Y rows of Y tab-delimited floating-point numbers describe the state transition matrix. Each row represents the current state, each column represents a state to transition to, and each entry represents the probability of that transition occurring. The next Y rows of X tab-delimited floating-point numbers describe the output emission matrix, encoded analogously to the state transition matrix. The file ends with 5 possible emissions from that HMM.

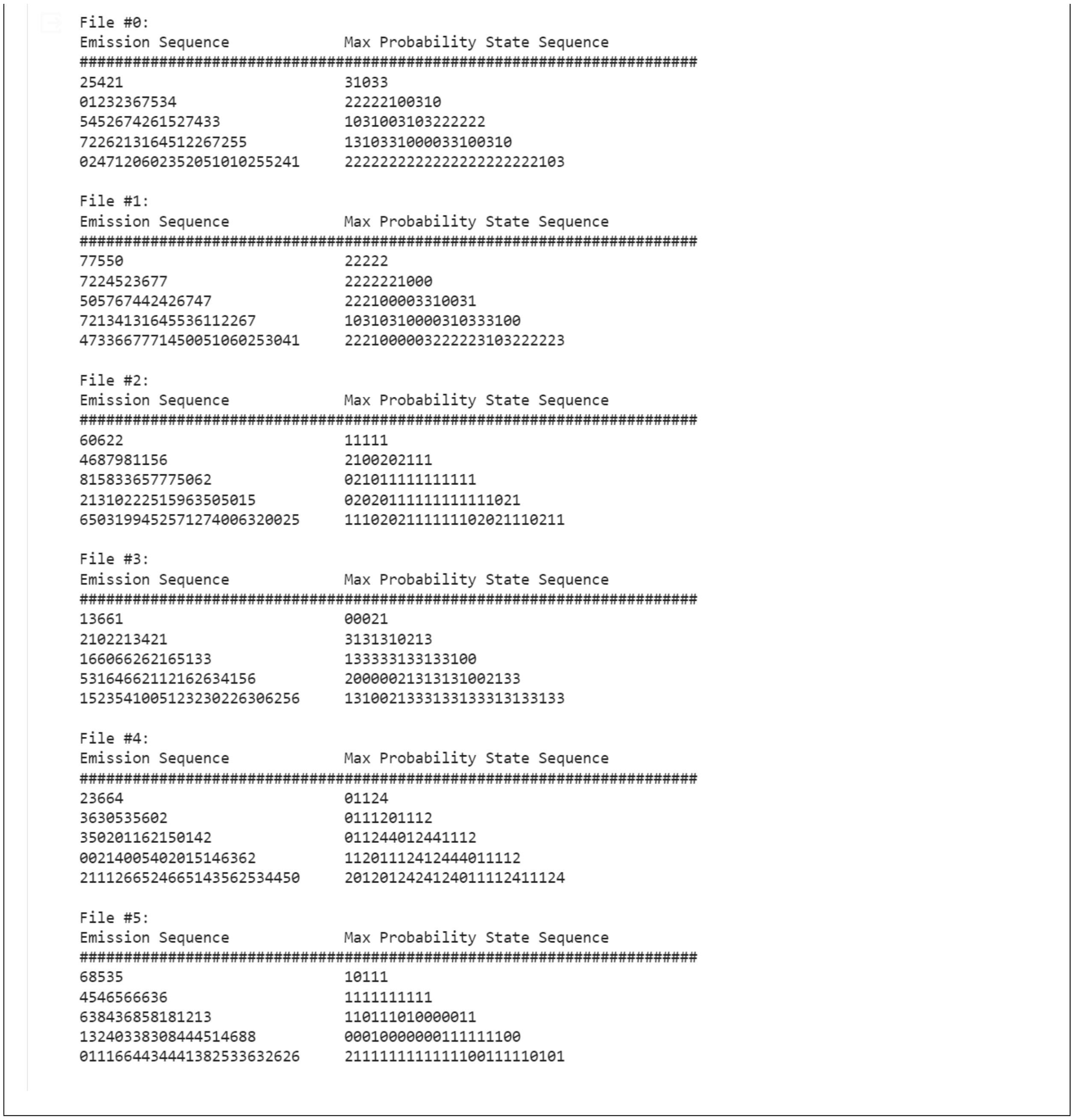
The supplementary data folder also contains one additional file titled `ron.txt`. This is used in problems 2C and 2D and is explained in greater detail there.

Problem A [10 points]: For each of the six trained HMMs, find the max-probability state sequence for each of the five input sequences at the end of the corresponding file. To complete this problem, you will have to implement the Viterbi algorithm. Write your implementation well, as we will be reusing it in a later problem. See the end of problem 2B for a big hint!

Show your results on the 6 files. (Copy-pasting the results under the "Part A" heading of `HMM.ipynb` suffices.)

Solution A.:

https://colab.research.google.com/drive/1xpwbKdWaCz9b50-DbbfbApuaXVXmT0Ox?usp=sharing



```
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
7213413164553611226 7         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     1310021333133133313133133

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 [17 points]: For each of the six trained HMMs, find the probabilities of emitting the five input sequences at the end of the corresponding file. To complete this problem, you will have to implement the Forward algorithm and the Backward algorithm. You may assume that the initial state is randomly selected along a uniform distribution. Again, write your implementation well, as we will be reusing it in a later problem.

Note that the probability of emitting an input sequence can be found by using either the α vectors from the Forward algorithm or the β vectors from the Backward algorithm. You don't need to worry about this, as it is done for you in `probability_alphas()` and `probability_betas()`.

Implement the Forward algorithm. Show your results on the 6 files.
Implement the Backward algorithm. Show your results on the 6 files.

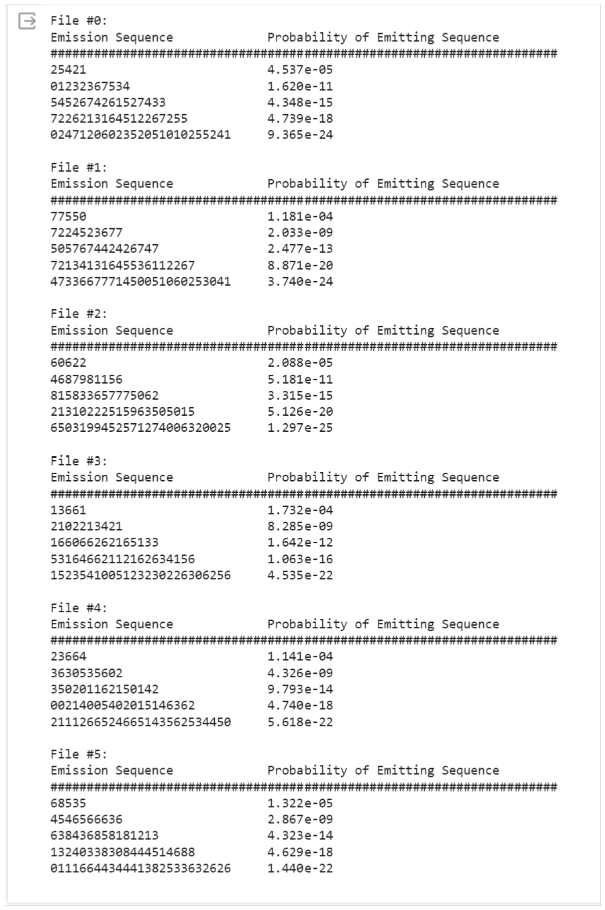
After you complete problems 2A and 2B, you can compare your results for the file titled `sequence_data0.txt` with the values given in the table below:

Dataset	Emission Sequence	Max-probability State Sequence	Probability of Sequence
0	25421	31033	4.537e-05
0	01232367534	22222100310	1.620e-11
0	5452674261527433	1031003103222222	4.348e-15
0	7226213164512267255	1310331000033100310	4.739e-18
0	0247120602352051010255241	2222222222222222222222103	9.365e-24

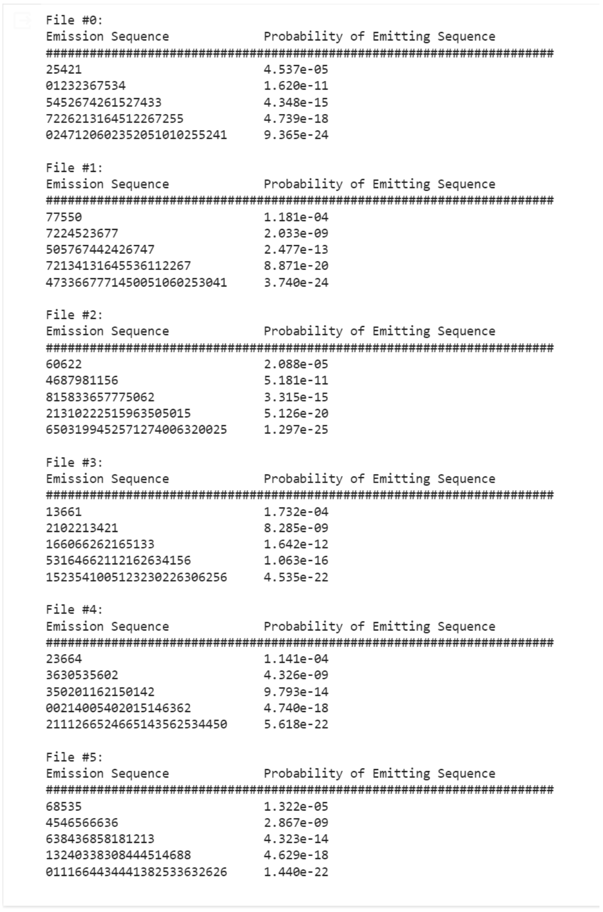
Solution B:

https://colab.research.google.com/drive/1xpwbKdWaCz9b50-DbbfbApuaXVXmT0Ox?usp=sharing

Forwards pass (L) and Backwards Pass (R):



```
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
```



```
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
```

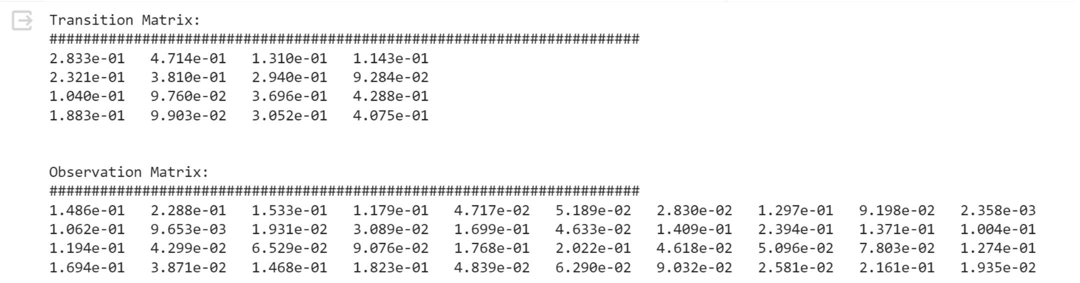
HMM Training

Ron is an avid music listener, and his genre preferences at any given time depend on his mood. Ron's possible moods are happy, mellow, sad, and angry. Ron experiences one mood per day (as humans are known to do) and chooses one of ten genres of music to listen to that day depending on his mood.

Ron's roommate, who is known to take to odd hobbies, is interested in how Ron's mood affects his music selection, and thus collects data on Ron's mood and music selection for six years (2190 data points). This data is contained in the supplementary file `ron.txt`. Each row contains two tab-delimited strings: Ron's mood and Ron's genre preference that day. The data is split into 12 sequences, each corresponding to half a year's worth of observations. The sequences are separated by a row containing only the character `-`.

Problem C [10 points]: Use a single M-step to train a supervised Hidden Markov Model on the data in `ron.txt`. What are the learned state transition and output emission matrices?

Solution C.:



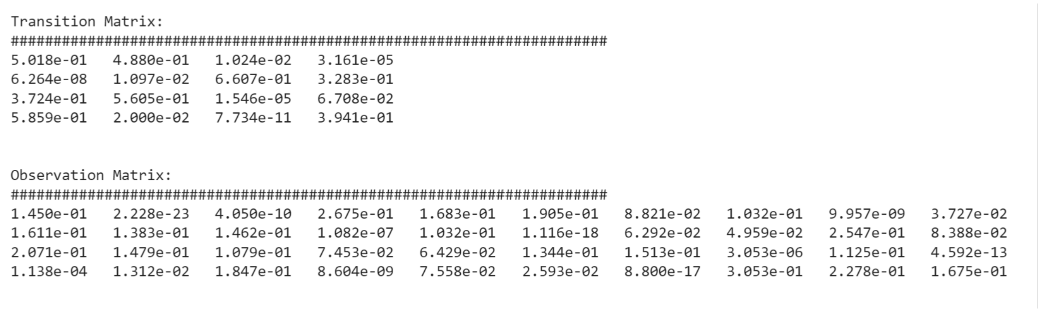
```
Transition Matrix:
######################################################################
2.833e-01   4.714e-01   1.310e-01   1.143e-01
2.321e-01   3.810e-01   2.940e-01   9.284e-02
1.040e-01   9.760e-02   3.696e-01   4.288e-01
1.883e-01   9.903e-02   3.052e-01   4.075e-01


Observation Matrix:
######################################################################
1.486e-01   2.288e-01   1.533e-01   1.179e-01   4.717e-02   5.189e-02   2.830e-02   1.297e-01   9.198e-02   2.358e-03
1.062e-01   9.653e-03   1.931e-02   3.089e-02   1.699e-01   4.633e-02   1.409e-01   2.394e-01   1.371e-01   1.004e-01
1.194e-01   4.299e-02   6.529e-02   9.076e-02   1.768e-01   2.022e-01   4.618e-02   5.096e-02   7.803e-02   1.274e-01
1.694e-01   3.871e-02   1.468e-01   1.823e-01   4.839e-02   6.290e-02   9.032e-02   2.581e-02   2.161e-01   1.935e-02
```

Problem D [15 points]: Now suppose that Ron has a third roommate who is also interested in how Ron's mood affects his music selection. This roommate is lazier than the other one, so he simply steals the first roommate's data. Unfortunately, he only manages to grab half the data, namely, Ron's choice of music for each of the 2190 days.

In this problem, we will train an unsupervised Hidden Markov Model on this data. Recall that unsupervised HMM training is done using the Baum-Welch algorithm and will require repeated EM steps. For this problem, we will use 4 hidden states and run the algorithm for 1000 iterations. The transition and observation matrices are initialized for you in the helper functions `supervised_learning()` and `unsupervised_learning()` such that they are random and normalized.

What are the learned state transition and output emission matrices? **(We will make a Piazza post about the seeds you should use. Please report the results with the specified seeds.)**

Solution D.:



```
Transition Matrix:
######################################################################
5.018e-01   4.880e-01   1.024e-02   3.161e-05
6.264e-08   1.097e-02   6.607e-01   3.283e-01
3.724e-01   5.605e-01   1.546e-05   6.708e-02
5.859e-01   2.000e-02   7.734e-11   3.941e-01


Observation Matrix:
######################################################################
1.450e-01   2.228e-23   4.050e-10   2.675e-01   1.683e-01   1.905e-01   8.821e-02   1.032e-01   9.957e-09   3.727e-02
1.611e-01   1.383e-01   1.462e-01   1.082e-07   1.032e-01   1.116e-18   6.292e-02   4.959e-02   2.547e-01   8.388e-02
2.071e-01   1.479e-01   1.079e-01   7.453e-02   6.429e-02   1.344e-01   1.513e-01   3.053e-06   1.125e-01   4.592e-13
1.138e-04   1.312e-02   1.847e-01   8.604e-09   7.558e-02   2.593e-02   8.800e-17   3.053e-01   2.278e-01   1.675e-01
```

Problem E [5 points]: How do the transition and emission matrices from 2C and 2D compare? Which do you think provides a more accurate representation of Ron's moods and how they affect his music choices? Justify your answer. Suggest one way that we may be able to improve the method (supervised or unsupervised) that you believe produces the less accurate representation.

Solution E.: *It is immediately apparent that the transmission and emission matrices from 2C have less variance and seem to lie roughly along the same order of magnitude (10^{-1} to 10^{-2} for the transition matrix and 10^{-1} to 10^{-4} for the observation matrix). Contrasting with this, the matrices from 2D have a large amount of variance with values on the order of magnitude scaling from 10^{-1} to 10^{-11} for the transition matrix and 10^{-1} to 10^{-23} for the observation matrix.*

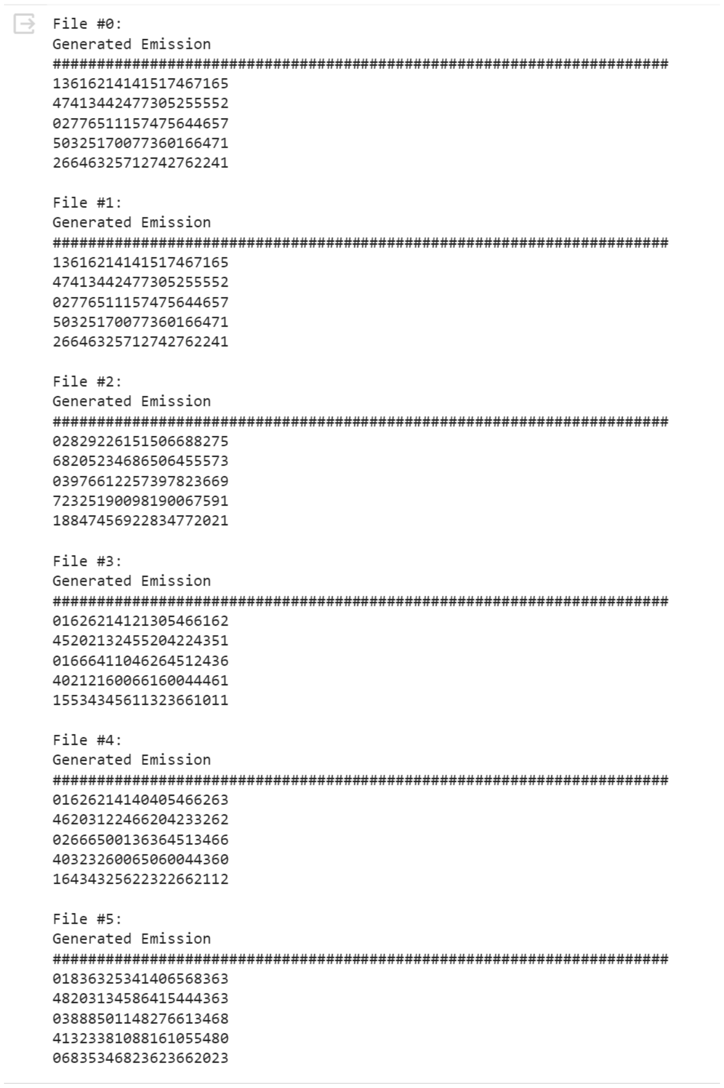
The model from 2C (supervised learning) provides a more accurate representation of Ron's moods and how they affect his music choices. We deduce that because of the lower variance between values of the transition and emission matrices. It also makes sense because we have closed-form solutions for the matrices in supervised learning but the convergence for unsupervised learning has a large dependence on our initialization values. To improve the unsupervised learning, we could feed the model more data so it could learn the hidden states better or we could initialize our matrices to better values.

Sequence Generation

Hidden Markov Models fall under the umbrella of generative models and therefore can be used to not only predict sequential data, but also to generate it.

Problem F [5 points]: Load the trained HMMs from the files titled `sequence_data0.txt,...,sequence_-data5.txt`. Use the six models to probabilistically generate five sequences of emissions from each model, each of length 20. Show your results.

Solution F.:



```
File #0:
Generated Emission
######################################################################
13616214141517467165
47413442477305255552
02776511157475644657
50325170077360166471
26646325712742762241

File #1:
Generated Emission
######################################################################
13616214141517467165
47413442477305255552
02776511157475644657
50325170077360166471
26646325712742762241

File #2:
Generated Emission
######################################################################
02829226151506688275
68205234686506455573
03976612257397823669
72325190098190067591
18847456922834772021

File #3:
Generated Emission
######################################################################
01626214121305466162
45202132455204224351
01666411046264512436
40212160066160044461
15534345611323661011

File #4:
Generated Emission
######################################################################
01626214140405466263
46203122466204233262
02666500136364513466
40323260065060044360
16434325622322662112

File #5:
Generated Emission
######################################################################
01836325341406568363
48203134586415444363
03888501148276613468
41323381088161055480
06835346823623662023
```

Visualization & Analysis

Once you have implemented the above, load and run `2_notebook.ipynb`. In this notebook, you will apply the HMM you have implemented to the Constitution. There is no coding required for this part, only analysis. To run the notebook, however, you will likely need to install the `wordcloud` package. Please refer to the provided installation instructions if you get an error when running `pip install wordcloud`.

Answer the following problems in the context of the visualizations in the notebook.

Problem G [3 points]: What can you say about the sparsity of the trained A and O matrices? How does this sparsity affect the transition and observation behaviour at each state?

Solution G.: *We can see that the trained A and O matrices are both very sparse. The matrix values, when converted to intensities, are mostly very dark (intensities and values close to 0), leaving behind very few realistic transitions. This means that the transition and observation behavior at each state will largely be the same from iteration to iteration because the model cannot deviate much despite the process being random. This might make it easier to test the accuracy of the model emissions, but it also can lead to a model with poor generalization.*

Problem H [5 points]: How do the sample emission sentences from the HMM change as the number of hidden states is increased? What happens in the special case where there is only one hidden state? In general, when the number of hidden states is unknown while training an HMM for a fixed observation set, can we increase the training data likelihood by allowing more hidden states?

Solution H.: *As the number of hidden states increases, the sample emission sentences from the HMM are more coherent and are increasingly likely to have been text from the Constitution. In the special case of one hidden state, our sample emission is the following: "Bill the crimes be laws states stated between authorityto and of on and determine of the receive person or and of in census be carrying..." We can see that there is no apparent order to the words selected. Because the model stays in the same state the entire time, the word emissions are generated exclusively off the training data word frequencies and they are generated randomly as well. When we allow more hidden states, we generally increase the training data likelihood, similar to how more layers in a network can lead to better training accuracy.*

Problem I [5 points]: Pick a state that you find semantically meaningful, and analyze this state and its wordcloud. What does this state represent? How does this state differ from the other states? Back up your claim with a few key words from the wordcloud.

Solution I.: *I found State 0 the most semantically meaningful because it contained the words that I found most likely to be contained in the Constitution. State 0's predominant words were senate, constitution, united, congress, president, and legislature. Each of these words is essential to governmental function and we can also see that the words are correlated and serve similar purposes. This state differed from other states because it contained related words and it made sense in the context of the Constitution. For example, state 3 contained the word 'state' in three different variations which isn't representative of the entire Constitution. State 8's major words were 'may' and 'except' which are not correlated in the slightest and are not indicative of the Constitution.*