

Policies

- Due 9 PM, March 8th, via Gradescope.
- You are free to collaborate on all of the problems, subject to the collaboration policy stated in the syllabus.
- You should submit all code used in the homework. We ask that you use Python 3.6+ for your code, and that you comment your code such that the TAs can follow along and run it without any issues.

Submission Instructions

- **In the report, include any images generated by your code along with your answers to the questions.** For instructions specifically pertaining to the Gradescope submission process, see https://www.gradescope.com/get_started#student-submission.
- Since we are only using Gradescope this year, we ask that you upload both your code and solution pdf together as a single pdf and mark each page with the problem number. There are various free online tools to combine pdfs. To download your code as a pdf from Jupyter notebook, you can navigate to File -> Download as -> "PDF via Latex" or alternatively download the Latex file and compile it yourself.

1 Probability Refresher [0 points]

This problem is a probability refresher and is worth 0 points. It should take you no more than 2-4 minutes to solve for each part. If you find that you are taking more time, you should probably watch the Probability recitation.

Problem A [0 points]: If $X, Y \sim \mathcal{N}(0, 1)$, compute $\Pr[X < \frac{Y}{\sqrt{2}}]$.

Solution A: $\Pr[X < Y/\sqrt{2}] = \Pr[\frac{Y}{\sqrt{2}} - X > 0]$. We define a new variable $D = \frac{Y}{\sqrt{2}} - X$. Note that $E[D] = E[\frac{Y}{\sqrt{2}}] - E[X] = 0$ and $\text{Var}(D) = (\frac{1}{\sqrt{2}})^2 \text{Var}(Y) + \text{Var}(X) = \frac{1}{2} + 1 = \frac{3}{2}$. This means that $D \sim \mathcal{N}(0, \sqrt{\frac{3}{2}})$ and $\Pr[D > 0] = 0.5$. This makes sense because X and Y should be equivalent such that $\Pr[X < \frac{Y}{\sqrt{2}}] = \Pr[Y < \frac{X}{\sqrt{2}}] = 0.5$.

Problem B [0 points]: If $X, Y \sim \mathcal{N}(0, 1)$, compute $\Pr[X + Y > 0 | X > 0]$.

Solution B: $\Pr[X + Y > 0 | X > 0] = \frac{\Pr[X+Y>0 \text{ and } X>0]}{\Pr[X>0]} = 2 \cdot \frac{\Pr[X+Y>0 \text{ and } X>0]}{\Pr[X>0]}$.
 $2 \cdot \frac{\Pr[X+Y>0 \text{ and } X>0]}{\Pr[X>0]} = 2 \int_0^\infty \int_y^\infty \frac{1}{2\pi} e^{-\frac{1}{2}(x^2+y^2)} dx dy \approx 0.125$

Problem C [0 points]: Prove that σ -algebras are closed under finitely many intersections. Hint: Recall that they are closed under finitely many unions and complements.

Solution C: Want to show that if $S_1, S_2, S_3, \dots, S_n \in A$ for subsets S_i and σ -algebra A , then $\cap_{i=1}^n S_n \in A$. By definition of a σ -algebra, $S \in A \implies \bar{S} \in A$. Also by definition of a σ -algebra, $\cup_{i=1}^n \bar{S}_i \in A$. The complement of this union must also be in A . This complement would be the set of all elements that are present in all of the S_i , i.e. $\cap_{i=1}^n S_n \in A$.

Problem D [0 points]: If X is a continuous random variable that takes on values in the range $(0, \infty)$, and Y is another random variable such that the marginal density of X and Y is given by

$$f_{X,Y}(x,y) = \frac{1}{y} e^{-x/y} e^{-y}$$

What is the probability distribution function of Y ?

Solution D: *The probability distribution function of Y is $p(y)$. $p(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dx = \int_0^{\infty} \frac{1}{y} e^{-x/y} e^{-y} dx$
 $= -\frac{1}{y^2} e^{-y}$*

2 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:

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

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 $p(x | y)$, and let $\theta_{xjc} = P(x_j | x_1, \dots, x_{j-1}, y = c)$. Assuming we store each θ_{xjc} , how many parameters are needed to represent this factorization? Use big-O notation.

Solution A: Note that

$$\begin{aligned} p(x | y = c) &= \frac{P(x_1, x_2, \dots, x_n, y = c)}{P(y = c)} \\ &= \frac{1}{C} \prod_{j=1}^D P(x_j | x_1, x_2, \dots, x_{j-1}, y = c) \\ &= \frac{1}{C} \prod_{j=1}^D \theta_{xjc} \end{aligned}$$

So if we know every θ_{xjc} , then we can know $p(x | y = c)$. Since they’re all dependent on each other, we’d need to know the $j-1$ x ’s before for the x_j . Thus, for the final $j = D$, we’d need $D-1$ prior x ’s. $x_j \in \{0, 1\}$, this means we have 2^{D-1} different possible x ’s to store to get our final result. Using this method, we would need to store $2^{D-1}C = O(2^D C)$ parameters.

Problem B [5 points]: Assume we did no such factorization, and just used the conditional probability $p(x | y = c)$. 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: With no factorization, we would just need to memorize the probability distribution for each possible pair of (x, y) . Since there are C classes for y and x is a D dimensional vector with each x_j having one of the binary values $\{0, 1\}$, we need to store $O(2^D C)$ parameters. Same as previous part.

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: *With small N , we run the risk of over-fitting by using the full model. It is likely that the Naive Bayes model will be better since for small N it is unlikely that the full model will be able to assess accurately the dependence relationships between the features without incorrectly fitting noise.*

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: *For large N , the opposite is true and the full model would likely give a lower test error. This is because the full model can capture more complexity and with a lot of sample data, there would be enough information to accurately fit 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 $p(y | x)$, 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: $P(y|x) = \frac{P(x \cap y)}{P(x)}$

In naive bayes, $P(x \cap y)$ takes $O(1)$ to loop through the entire table to check, but it takes $O(D)$ to calculate each one and there are C classes so $O(DC)$ to create that table. $P(x)$ is $O(D)$ which is just frequency counts across the look up table. This collapses to just $O(DC)$ altogether.

For the full model, once all the parameter estimates have been computed, we'd have θ_{xjc} by construction of part 2A. Thus we can get $P(x|y)$ in constant time but will need to examine all C classes, making it $O(C)$. $P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x \cap y)P(y)}{P(x)P(y)} = \frac{P(x \cap y)}{P(x)}$. Converting to array index as a $O(D)$ operation. Thus the final time complexity is $O(C+D)$.

3 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 $0, 1, \dots, X - 1$). 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 of under the “Part A” heading of `HMM.ipynb` suffices.)

Solution A:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

```

File #0:
Emission Sequence      Max Probability State Sequence
#####
25421                  31033
01232367534           22222100310
5452674261527433      1031003103222222
7226213164512267255    1310331000033100310
0247120602352051010255241 22222222222222222222103

File #1:
Emission Sequence      Max Probability State Sequence
#####
77550                  22222
7224523677            2222221000
505767442426747        222100003310031
72134131645536112267    10310310000310333100
4733667771450051060253041 22210000322222310322223

File #2:
Emission Sequence      Max Probability State Sequence
#####
60622                  11111
4687981156             2100202111
815833657775062         021011111111111
21310222515963505015    0202011111111111021
6503199452571274006320025 1110202111111102021110211

File #3:
Emission Sequence      Max Probability State Sequence
#####
13661                  00021
2102213421             3131310213
166066262165133         133333133133100
53164662112162634156    20000021313131002133
1523541005123230226306256 1310021333133133313133

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	2222222222222222222103	9.365e-24

Solution B:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>


```

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

```

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

```

Backward Algorithm:

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`.

After you implement 2C, please compare your output transition and observation matrices with the state transition and output emission matrices below. You should make sure your outputs match before continuing.

$$\text{Transition matrix} = \begin{pmatrix} 2.833\text{e-}01 & 4.714\text{e-}01 & 1.310\text{e-}01 & 1.143\text{e-}01 \\ 2.321\text{e-}01 & 3.810\text{e-}01 & 2.940\text{e-}01 & 9.284\text{e-}02 \\ 1.040\text{e-}01 & 9.760\text{e-}02 & 3.696\text{e-}01 & 4.288\text{e-}01 \\ 1.883\text{e-}01 & 9.903\text{e-}02 & 3.052\text{e-}01 & 4.075\text{e-}01 \end{pmatrix}$$

Observation Matrix =

$$\begin{pmatrix} 1.486\text{e-}01 & 2.288\text{e-}01 & 1.533\text{e-}01 & 1.179\text{e-}01 & 4.717\text{e-}02 & 5.189\text{e-}02 & 2.830\text{e-}02 & 1.297\text{e-}01 & 9.198\text{e-}02 & 2.358\text{e-}03 \\ 1.062\text{e-}01 & 9.653\text{e-}03 & 1.931\text{e-}02 & 3.089\text{e-}02 & 1.699\text{e-}01 & 4.633\text{e-}02 & 1.409\text{e-}01 & 2.394\text{e-}01 & 1.371\text{e-}01 & 1.004\text{e-}01 \\ 1.194\text{e-}01 & 4.299\text{e-}02 & 6.529\text{e-}02 & 9.076\text{e-}02 & 1.768\text{e-}01 & 2.022\text{e-}01 & 4.618\text{e-}02 & 5.096\text{e-}02 & 7.803\text{e-}02 & 1.274\text{e-}01 \\ 1.694\text{e-}01 & 3.871\text{e-}02 & 1.468\text{e-}01 & 1.823\text{e-}01 & 4.839\text{e-}02 & 6.290\text{e-}02 & 9.032\text{e-}02 & 2.581\text{e-}02 & 2.161\text{e-}01 & 1.935\text{e-}02 \end{pmatrix}$$

Solution C:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

```
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?

Below are the transition and observation matrices for the unsupervised Hidden Markov Model at the zero-th iteration.

$$\text{Transition Matrix} = \begin{pmatrix} 2.003\text{e-}01 & 3.720\text{e-}01 & 5.642\text{e-}02 & 3.713\text{e-}01 \\ 1.581\text{e-}01 & 2.147\text{e-}01 & 4.197\text{e-}01 & 2.075\text{e-}01 \\ 2.941\text{e-}01 & 1.475\text{e-}02 & 4.032\text{e-}01 & 2.880\text{e-}01 \\ 1.759\text{e-}01 & 4.205\text{e-}01 & 1.617\text{e-}01 & 2.419\text{e-}01 \end{pmatrix}$$

Observation Matrix =

$$\begin{pmatrix} 2.585\text{e-}02 & 7.773\text{e-}02 & 3.923\text{e-}02 & 5.058\text{e-}02 & 1.447\text{e-}01 & 5.407\text{e-}02 & 9.356\text{e-}02 & 1.891\text{e-}01 & 1.854\text{e-}01 & 1.398\text{e-}01 \\ 9.821\text{e-}02 & 5.024\text{e-}02 & 2.915\text{e-}02 & 1.760\text{e-}01 & 9.364\text{e-}02 & 2.102\text{e-}02 & 1.131\text{e-}01 & 1.409\text{e-}01 & 1.112\text{e-}01 & 1.664\text{e-}01 \\ 8.272\text{e-}03 & 1.104\text{e-}01 & 9.597\text{e-}02 & 1.303\text{e-}02 & 1.340\text{e-}01 & 1.781\text{e-}01 & 1.239\text{e-}01 & 5.434\text{e-}02 & 1.755\text{e-}01 & 1.065\text{e-}01 \\ 1.028\text{e-}01 & 1.516\text{e-}01 & 2.977\text{e-}02 & 1.650\text{e-}01 & 1.375\text{e-}01 & 1.584\text{e-}01 & 3.856\text{e-}02 & 1.615\text{e-}01 & 3.851\text{e-}02 & 1.641\text{e-}02 \end{pmatrix}$$

Below are the transition and observation matrices for the unsupervised Hidden Markov Model at the first iteration.

$$\text{Transition Matrix} = \begin{pmatrix} 1.887\text{e-}01 & 3.750\text{e-}01 & 5.354\text{e-}02 & 3.828\text{e-}01 \\ 1.516\text{e-}01 & 2.204\text{e-}01 & 4.083\text{e-}01 & 2.196\text{e-}01 \\ 2.832\text{e-}01 & 1.556\text{e-}02 & 3.924\text{e-}01 & 3.089\text{e-}01 \\ 1.647\text{e-}01 & 4.254\text{e-}01 & 1.559\text{e-}01 & 2.540\text{e-}01 \end{pmatrix}$$

Observation Matrix =

$$\begin{pmatrix} 6.272\text{e-}02 & 5.705\text{e-}02 & 8.345\text{e-}02 & 5.738\text{e-}02 & 1.380\text{e-}01 & 5.191\text{e-}02 & 8.730\text{e-}02 & 1.572\text{e-}01 & 2.127\text{e-}01 & 9.231\text{e-}02 \\ 2.152\text{e-}01 & 3.537\text{e-}02 & 5.388\text{e-}02 & 1.786\text{e-}01 & 8.065\text{e-}02 & 1.888\text{e-}02 & 8.992\text{e-}02 & 1.053\text{e-}01 & 1.168\text{e-}01 & 1.054\text{e-}01 \\ 1.988\text{e-}02 & 8.040\text{e-}02 & 1.846\text{e-}01 & 1.378\text{e-}02 & 1.243\text{e-}01 & 1.626\text{e-}01 & 1.081\text{e-}01 & 4.324\text{e-}02 & 1.937\text{e-}01 & 6.944\text{e-}02 \\ 2.222\text{e-}01 & 1.001\text{e-}01 & 5.614\text{e-}02 & 1.652\text{e-}01 & 1.169\text{e-}01 & 1.376\text{e-}01 & 3.193\text{e-}02 & 1.205\text{e-}01 & 3.943\text{e-}02 & 9.930\text{e-}03 \end{pmatrix}$$

Below are the transition and observation matrices for the unsupervised Hidden Markov Model at the final iteration (iteration 1000).

$$\text{Transition Matrix} = \begin{pmatrix} 4.345\text{e-}01 & 1.559\text{e-}01 & 9.612\text{e-}02 & 3.134\text{e-}01 \\ 5.904\text{e-}15 & 2.996\text{e-}01 & 7.004\text{e-}01 & 1.328\text{e-}11 \\ 5.219\text{e-}01 & 3.828\text{e-}18 & 2.259\text{e-}01 & 2.522\text{e-}01 \\ 5.016\text{e-}03 & 3.679\text{e-}01 & 7.971\text{e-}03 & 6.191\text{e-}01 \end{pmatrix}$$

Observation Matrix =

$$\begin{pmatrix} 2.235\text{e-}01 & 8.353\text{e-}09 & 7.603\text{e-}02 & 9.263\text{e-}02 & 2.315\text{e-}02 & 1.023\text{e-}02 & 1.273\text{e-}01 & 3.294\text{e-}01 & 1.288\text{e-}10 & 1.177\text{e-}01 \\ 2.260\text{e-}01 & 3.339\text{e-}05 & 7.427\text{e-}14 & 2.039\text{e-}01 & 3.226\text{e-}02 & 1.151\text{e-}01 & 1.575\text{e-}01 & 3.954\text{e-}30 & 8.256\text{e-}02 & 1.828\text{e-}01 \\ 3.403\text{e-}02 & 1.351\text{e-}01 & 1.900\text{e-}01 & 5.006\text{e-}02 & 2.247\text{e-}01 & 2.150\text{e-}05 & 6.462\text{e-}02 & 2.926\text{e-}02 & 2.723\text{e-}01 & 3.988\text{e-}65 \\ 8.931\text{e-}02 & 1.173\text{e-}01 & 1.035\text{e-}01 & 9.344\text{e-}02 & 1.505\text{e-}01 & 2.098\text{e-}01 & 9.588\text{e-}07 & 7.679\text{e-}02 & 1.594\text{e-}01 & 1.312\text{e-}07 \end{pmatrix}$$

Please use these outputs to check your code.

Solution D:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

```

Transition Matrix:
#####
2.003e-01  3.720e-01  5.642e-02  3.713e-01
1.581e-01  2.147e-01  4.197e-01  2.075e-01
2.941e-01  1.475e-02  4.032e-01  2.880e-01
1.759e-01  4.205e-01  1.617e-01  2.419e-01

Observation Matrix:
#####
2.585e-02  7.773e-02  3.923e-02  5.058e-02  1.447e-01  5.407e-02  9.356e-02  1.891e-01  1.854e-01  1.398e-01
9.821e-02  5.024e-02  2.915e-02  1.760e-01  9.364e-02  2.102e-02  1.131e-01  1.409e-01  1.112e-01  1.664e-01
8.272e-03  1.104e-01  9.597e-02  1.303e-02  1.340e-01  1.781e-01  1.239e-01  5.434e-02  1.755e-01  1.065e-01
1.028e-01  1.516e-01  2.977e-02  1.650e-01  1.375e-01  1.584e-01  3.856e-02  1.615e-01  3.851e-02  1.641e-02

Iteration 0:

Transition Matrix:
#####
1.887e-01  3.750e-01  5.354e-02  3.828e-01
1.516e-01  2.204e-01  4.083e-01  2.196e-01
2.832e-01  1.556e-02  3.924e-01  3.089e-01
1.647e-01  4.254e-01  1.559e-01  2.540e-01

Observation Matrix:
#####
6.272e-02  5.705e-02  8.345e-02  5.738e-02  1.380e-01  5.191e-02  8.730e-02  1.572e-01  2.127e-01  9.231e-02
2.152e-01  3.537e-02  5.388e-02  1.786e-01  8.065e-02  1.888e-02  8.992e-02  1.053e-01  1.168e-01  1.054e-01
1.988e-02  8.040e-02  1.846e-01  1.378e-02  1.243e-01  1.626e-01  1.081e-01  4.324e-02  1.937e-01  6.944e-02
2.222e-01  1.001e-01  5.614e-02  1.652e-01  1.169e-01  1.376e-01  3.193e-02  1.205e-01  3.943e-02  9.930e-03

Iteration 1:

Transition Matrix:
#####
4.345e-01  1.559e-01  9.612e-02  3.134e-01
5.904e-15  2.996e-01  7.004e-01  1.328e-11
5.219e-01  3.828e-18  2.259e-01  2.522e-01
5.016e-03  3.679e-01  7.971e-03  6.191e-01

Observation Matrix:
#####
2.235e-01  8.353e-09  7.603e-02  9.263e-02  2.315e-02  1.023e-02  1.273e-01  3.294e-01  1.288e-10  1.177e-01
2.260e-01  3.339e-05  7.427e-14  2.039e-01  3.226e-02  1.151e-01  1.575e-01  3.954e-30  8.256e-02  1.828e-01
3.403e-02  1.351e-01  1.900e-01  5.006e-02  2.247e-01  2.150e-05  6.462e-02  2.926e-02  2.723e-01  3.988e-65
8.931e-02  1.173e-01  1.035e-01  9.344e-02  1.505e-01  2.098e-01  9.588e-07  7.679e-02  1.594e-01  1.312e-07

Iteration 1k

```

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:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

The resulted transition and observation matrices from 2D has much smaller numbers (on average) than that of

2C (more sparse). For example the smallest number in the observation matrix reaches $3.988 * 10^{-65}$, while the lowest that the observation matrix reaches in 2C is $2.358 * 10^{-3}$. In that sense, there's also much more variability in 2D than in 2C since the range is so big.

It is more likely that supervised learning provides a more accurate representation because they actually have access to the moods that correlate to the music choices. 2C is directly maximizing the probabilities of seeing x and y 's, but for 2D, we're maximizing the probabilities of seeing x by repeatedly approximating the marginal probabilities through alpha and beta and then updating model parameters.

To improve the unsupervised method, we can experiment with some unbalanced initial start state probabilities rather than a uniform distribution; for example, a random distribution. If we expect our distribution to be more heavily weighed towards some state than others, than this may help. For example in English, it's much more likely to start off a given sentence with a noun or pronoun than a verb, so using an unbalanced distribution would likely produce a more accurate representation.

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:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

```
File #0:
Generated Emission
#####
11636355320667005130
52173010253065050441
72474350211637407436
51727130663641650714
16453153323107206317

File #1:
Generated Emission
#####
43073243674754532623
67225601541004540336
04471702364644562031
10670433655707274466
74064353237054474465

File #2:
Generated Emission
#####
05723018995667857666
68773238667967182096
30590953824014801332
28219510325987937677
89740430465656063790

File #3:
Generated Emission
#####
64661414661261263113
21656555125024620404
30340154442231643540
22064213436126336412
25111013121131313330

File #4:
Generated Emission
#####
35454412425233105463
13414433402514456210
11121060111621606602
21500552145301404246
54656313306540150523

File #5:
Generated Emission
#####
65042717601248052231
21764251044151543713
10008343866606586613
62106603182775262166
03512352504367456184
```

Visualization & Analysis

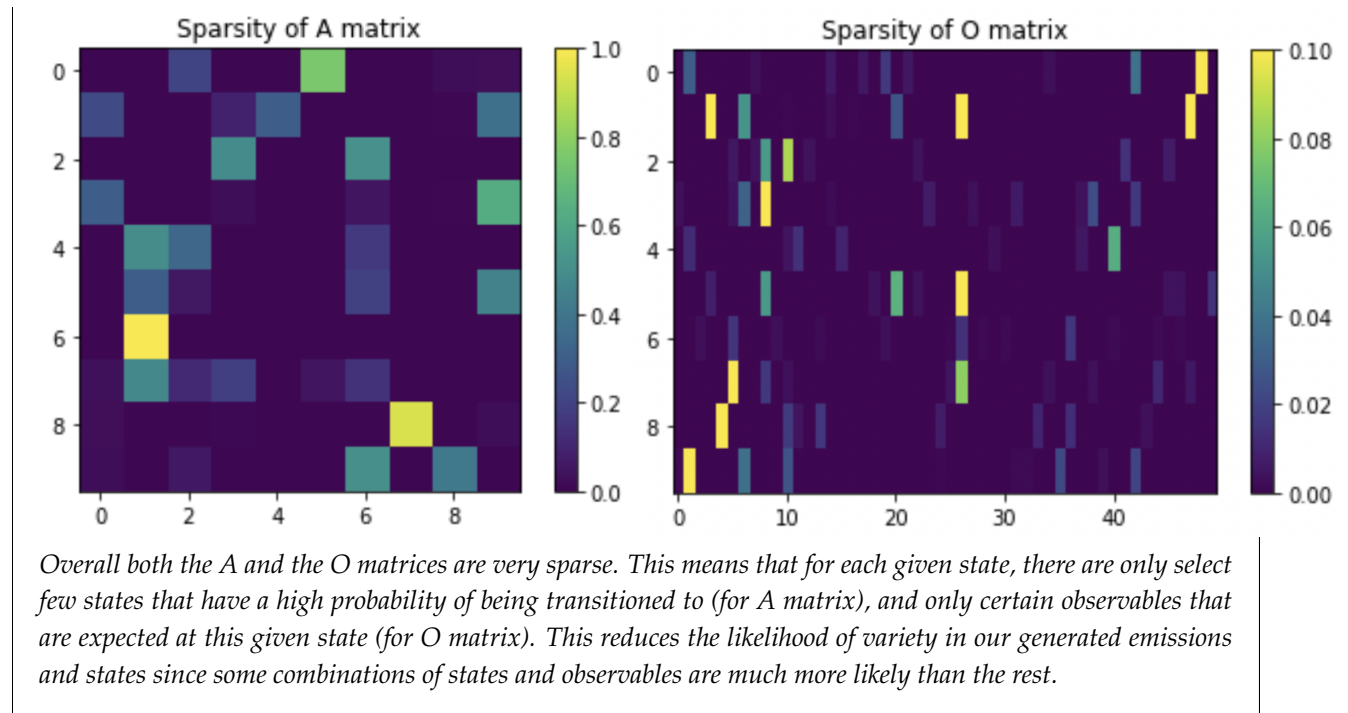
Once you have implemented the above, load and run `2_notebook.ipynb`. In this notebook, you will apply the HMM you have implemented to the Constitution. There is no coding required for this part, only analysis. To run the notebook, however, you will likely need to install the `wordcloud` package. Please 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:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>



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:

<https://colab.research.google.com/drive/1wVWbxZ7jdRUDdf2FUeEDbDG6ad97xYNj?usp=sharing>

As the number of hidden states increased, the sentence started making more logical/grammatical sense. When there's only one hidden state, we know that our start states are just uniform distribution across all the words. Thus, the words are literally selected at random because there's no preference in choosing one word or another. They have equal probability of being chosen. As a result, the sentence does not make sense. Yes, allowing more hidden states would increase the training data likelihood because it gives the model more classifications for the observations. So it can more accurately predict the sentence structure. Although, this is to a certain extent, like for grammar, there's nouns, verbs, adjectives, adverbs, etc. But we don't want to increase the hidden states to too large of a number because then we are wasting space and training time when most states will have a low probability anyway due to sparsity.

Solution I:

State 1



17