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]

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:

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

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

Solution A: *Since all the x_i's are conditionally independent, we can argue the following:*

$$p(x \mid y = c) = p(x_1 \mid y = c) \cdot p(x_2 \mid y = c) \cdots p(x_D \mid y = c) = p(x_1, \cdots, x_D \mid y)$$

And by the chain rule of probability, we get

$$p(x_1, \cdots, x_D \mid y) = p(x_1 \mid y = c) \cdot p(x_2, \cdots x_D \mid x_1, y = c)$$

$$= p(x_1 \mid y = c) \cdot p(x_2 \mid x_1, y = c) \cdot p(x_3 \mid x_2, x_1, y = c) \cdots p(x_D \mid x_{D-1}, \cdots, x_1, y = c)$$

And by substituting the variable given in the problem, we get

$$= \theta_{x1c} \cdot \theta_{x2c} \cdots \theta_{x(D-1)c}\theta_{xDc} = \prod_{j=1}^{D} \theta_{xjc}$$

To store each θ_{xjc}, we need $O(2^D C)$ number of parameters. We consider the permutations of previous features. In other words, $\sum_{i=0}^{D-1} 2^j = 2^D - 1 \approx 2^D$. And for C classes, we multiply this by C. Thus, we get $O(2^D C)$.

Problem B [5 points]: Assume we did no such factorization, and just used the conditional 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: *We will still have the same number of parameters as before ($O(2^D C)$). This is because all D features are binary. So, to represent all possibilities for D features, from x_1 to x_D, we need 2^D parameters. And since we have C classes, we multiply this by C to get $O(2^D C)$.*

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, Naive Bayes is more likely to give lower test error. Full model will have more features than Naive Bayes, and with a small sample size, it is much more likely to overfit to the sample data. The resulting model will be biased, and the model would thus perform worse with a test set.*

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: *If the sample size N is very large, full model is likely to give lower test set error. With a large sample, full model will have much lower chance of overfitting than with a small sample, and the larger set of features it has will better capture the underlying model. On the other hand, with the limited set of features, Naive Bayes has a higher chance of underfitting.*

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: *Native Bayes: Computing $p(x \mid y) = \frac{p(x|y)p(y)}{p(x)}$ will require $O(C \cdot D)$ complexity. We know that $p(y)$ is a uniform class prior. So, this is an O(1) operation. $p(x \mid y)$ is an O(D) operation, for each feature x_i. And $p(x)$ must be computed for each class. So, this is an O(C) operation. Thus, we have $O(CD)$ as our computational complexity.*
Full model: The computational complexity for full model will also be $O(CD)$. Converting a D-bit vector to an array index is an O(D) operation, and we do it for C number of classes. So, overall, we have O(CD). We can also think of this in terms of the product we were given for Problem A.

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

Computing probability for each x_j is an O(D) operation, and doing so for each class is an O(C) operation. Thus, overall, it's O(CD).

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 $0, 1, ..., X-1$). 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:

Code for problem 3

https://colab.research.google.com/drive/1C-qPAb9rr-l2urAjb98qJ9uWCj389XPP?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
72134131645536112267          10310310000310333100
4733667771450051060253041     2221000003222223103222223

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     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:

Forward:

```
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:

```
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
```

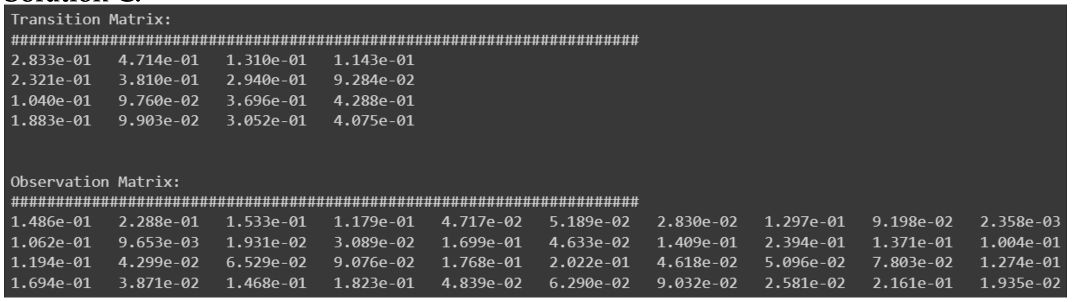
HMM Training

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?

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

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

Observation Matrix =

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

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).

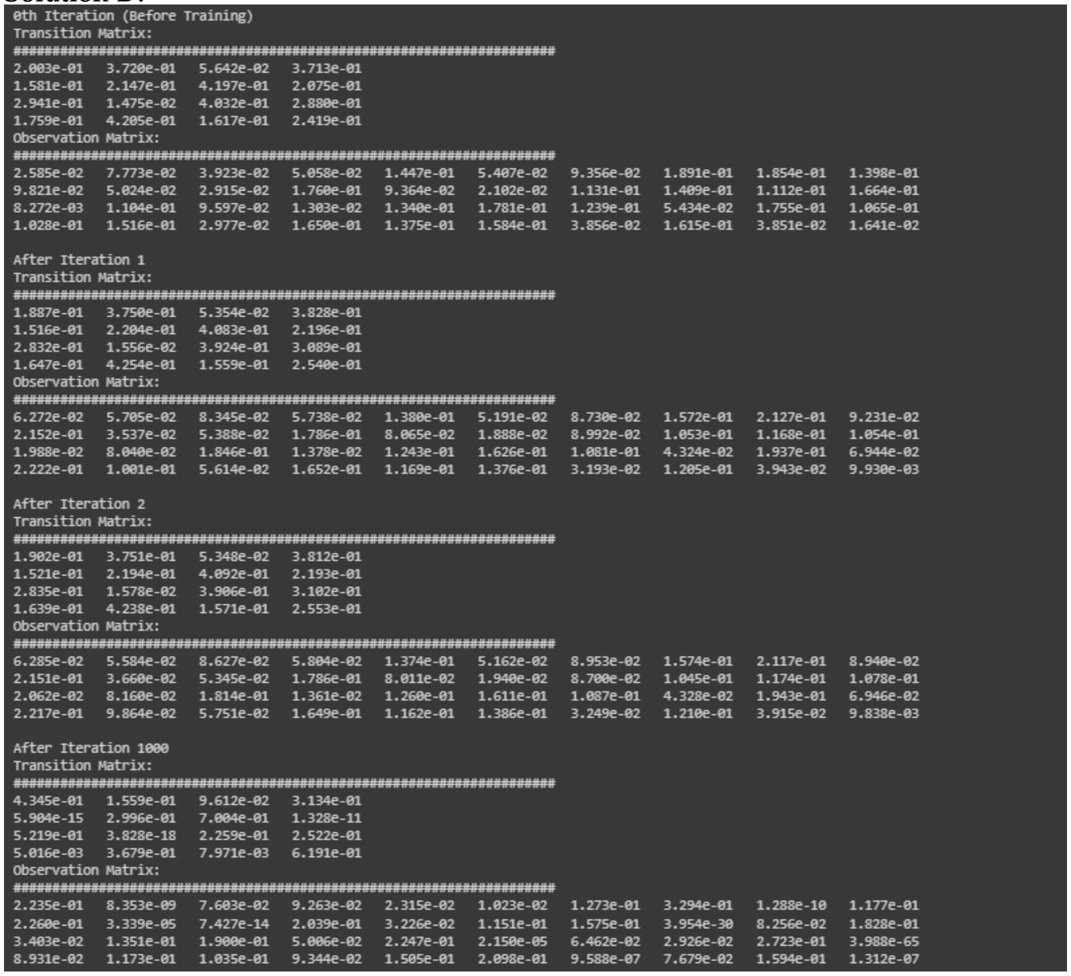
$$\text{Transition Matrix} = \begin{pmatrix} 4.345\text{e-}01 & 1.559\text{e-}01 & 9.612\text{e-}02 & 3.134\text{e-}01 \\ 5.904\text{e-}15 & 2.996\text{e-}01 & 7.004\text{e-}01 & 1.328\text{e-}11 \\ 5.219\text{e-}01 & 3.828\text{e-}18 & 2.259\text{e-}01 & 2.522\text{e-}01 \\ 5.016\text{e-}03 & 3.679\text{e-}01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

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:



```
0th Iteration (Before Training)
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

After Iteration 1
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

After Iteration 2
Transition Matrix:
######################################################################
1.902e-01   3.751e-01   5.348e-02   3.812e-01
1.521e-01   2.194e-01   4.092e-01   2.193e-01
2.835e-01   1.578e-02   3.906e-01   3.102e-01
1.639e-01   4.238e-01   1.571e-01   2.553e-01
Observation Matrix:
######################################################################
6.285e-02   5.584e-02   8.627e-02   5.804e-02   1.374e-01   5.162e-02   8.953e-02   1.574e-01   2.117e-01   8.940e-02
2.151e-01   3.660e-02   5.345e-02   1.786e-01   8.011e-02   1.940e-02   8.700e-02   1.045e-01   1.174e-01   1.078e-01
2.062e-02   8.160e-02   1.814e-01   1.361e-02   1.260e-01   1.611e-01   1.087e-01   4.328e-02   1.943e-01   6.946e-02
2.217e-01   9.864e-02   5.751e-02   1.649e-01   1.162e-01   1.386e-01   3.249e-02   1.210e-01   3.915e-02   9.838e-03

After Iteration 1000
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
```

Problem E [5 points]: How do the transition and emission matrices from 3C and 3D 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.

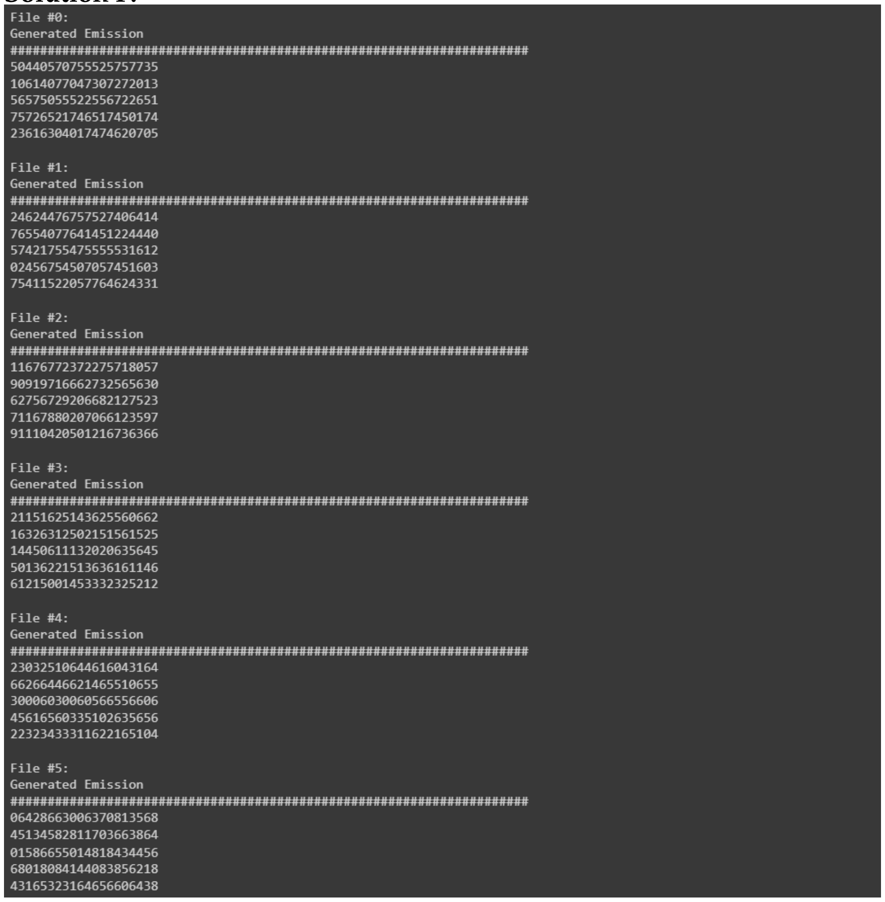
Solution E: *I believe the matrices from 3C are more of an accurate representation of Ron's moods and music-related behavior than the matrices from 3D. The 3D matrices have entries over a much larger range with some of the entries being almost zero ($< 10^{-10}$) while the matrices from 2C have entries that are spread over a much smaller range, a relatively balanced distribution, and no almost zero entries. In practical terms, it wouldn't make much sense for any transition or emission probability to be so close to zero when considering Ron's choices since we know his moods and music choices were relatively balanced. This is what's reflected in the matrices from 2C, evincing how training with Ron's mood as labels, produced a more accurate representation. One way we could* *make the unsupervised training more accurate would be to train with more data or leverage the fact that we know the data is balanced to force the matrix to be less sparse by making sure all entries are above a certain decimal value during training.*

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
######################################################################
50440570755525757735
10614077047307272013
56575055522556722651
75726521746517450174
23616304017474620705

File #1:
Generated Emission
######################################################################
24624476757527406414
76554077641451224440
57421755475555531612
02456754507057451603
75411522057764624331

File #2:
Generated Emission
######################################################################
11676772372275718057
90919716662732565630
62756729206682127523
71167880207066123597
91110420501216736366

File #3:
Generated Emission
######################################################################
21151625143625560662
16326312502151561525
14450611132020635645
50136221513636161146
61215001453332325212

File #4:
Generated Emission
######################################################################
23032510644616043164
66266446621465510655
30006030060566556606
45616560335102635656
22323433311622165104

File #5:
Generated Emission
######################################################################
06428663006370813568
45134582811703663864
01586655014818434456
68018084144083856218
43165323164656606438
```

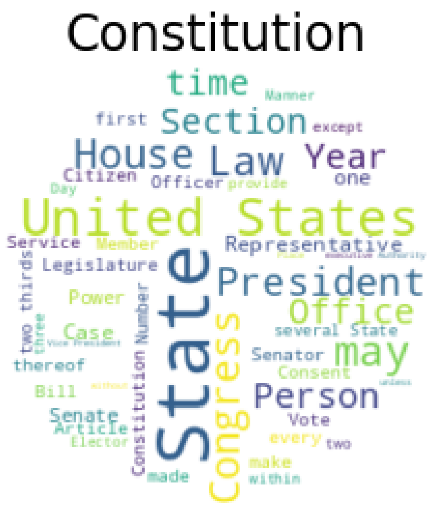
Visualization & Analysis

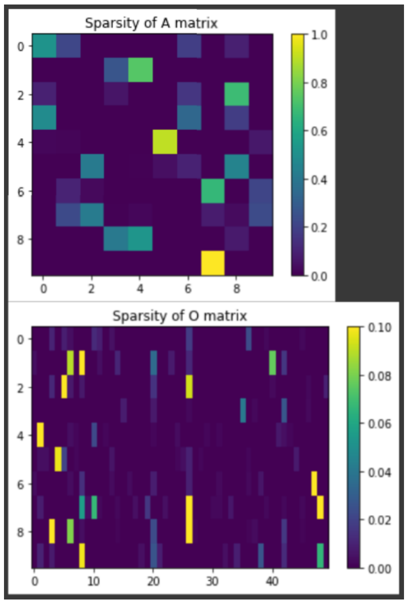
Once you have implemented the above, load and run `2_notebook.ipynb`. In this notebook, you will apply the HMM you have implemented to the Constitution. There is no coding required for this part, only analysis. To run the notebook, however, you will likely need to install the `wordcloud` package. Please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

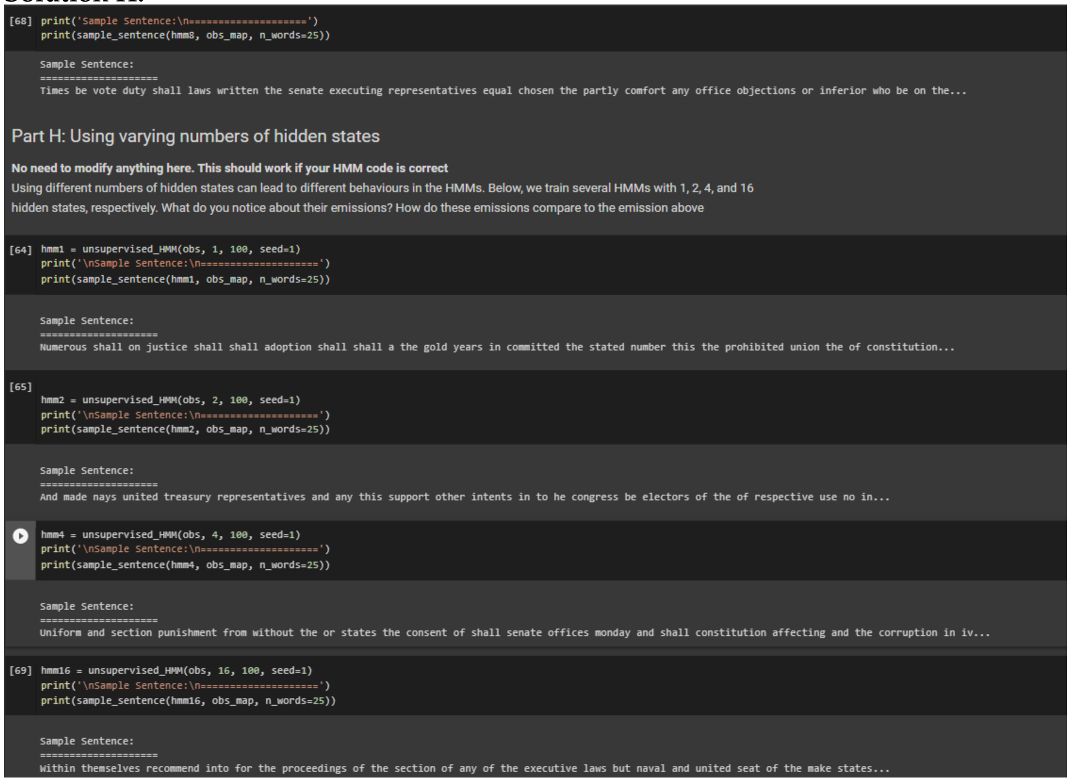




Both the trained A and O matrices are very sparse with few entries over 0.5. The more sparse the matrices, the more "consistent" sentences will be in terms of adjacent words. For example, if the matrix is very sparse, there may be only 2 emissions that are even somewhat likely for some given state such that any time we get to that state, either of the same 2 emissions will come next every single time. This happening for most states would make sentences relatively "consistent".

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:



```
[68] print('Sample Sentence:\n====================')
     print(sample_sentence(hmm8, obs_map, n_words=25))
```

```
Sample Sentence:
====================
Times be vote duty shall laws written the senate executing representatives equal chosen the partly comfort any office objections or inferior who be on the...
```

Part H: Using varying numbers of hidden states

No need to modify anything here. This should work if your HMM code is correct
Using different numbers of hidden states can lead to different behaviours in the HMMs. Below, we train several HMMs with 1, 2, 4, and 16 hidden states, respectively. What do you notice about their emissions? How do these emissions compare to the emission above

```
[64] hmm1 = unsupervised_HMM(obs, 1, 100, seed=1)
     print('\nSample Sentence:\n====================')
     print(sample_sentence(hmm1, obs_map, n_words=25))
```

```
Sample Sentence:
====================
Numerous shall on justice shall shall adoption shall shall a the gold years in committed the stated number this the prohibited union the of constitution...
```

```
[65]
     hmm2 = unsupervised_HMM(obs, 2, 100, seed=1)
     print('\nSample Sentence:\n====================')
     print(sample_sentence(hmm2, obs_map, n_words=25))
```

```
Sample Sentence:
====================
And made nays united treasury representatives and any this support other intents in to he congress be electors of the of respective use no in...
```

```
hmm4 = unsupervised_HMM(obs, 4, 100, seed=1)
print('\nSample Sentence:\n====================')
print(sample_sentence(hmm4, obs_map, n_words=25))
```

```
Sample Sentence:
====================
Uniform and section punishment from without the or states the consent of shall senate offices monday and shall constitution affecting and the corruption in iv...
```

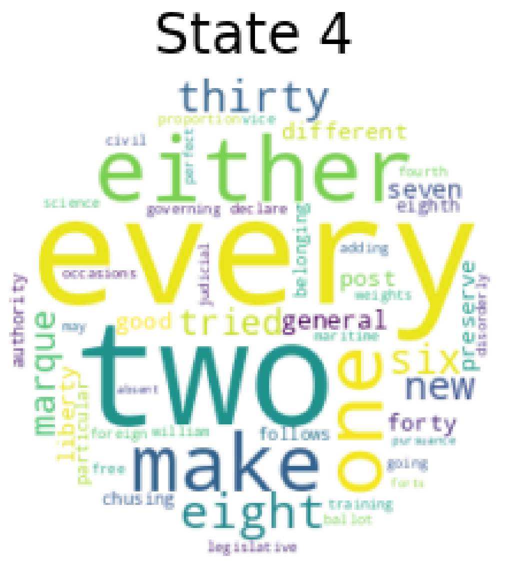
```
[69] hmm16 = unsupervised_HMM(obs, 16, 100, seed=1)
     print('\nSample Sentence:\n====================')
     print(sample_sentence(hmm16, obs_map, n_words=25))
```

```
Sample Sentence:
====================
Within themselves recommend into for the proceedings of the section of any of the executive laws but naval and united seat of the make states...
```

Although the sentences still make little sense, increasing the number of hidden states seems to make the sentences more grammatically correct. As we increase the number of hidden states, it seems the words depend more on the other words around them. When there is only one hidden state, adjacent words seem to be unrelated and the word choices for each sentence seem random, but when we increase the states, connections between words seem to appear and get stronger, producing the better grammar. In general, for a fixed observation set, we can increase the training data likelihood by allowing more hidden states as this would make more defined connections in our model, but this may not be desirable.

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:



State 4 seems to represent descriptors with its most representative words being "every", "two", and "either". Specifically, it has many descriptive numbers as representative words like "two", "one", "eight", "thirty", and "six". The other states seem to be more noun oriented or mixed/miscellaneous, but this state, for the most part, seems to capture most of the counters and specifiers/descriptors.