Machine Learning & Data Mining
Homework 6

Caltech CS/CNS/EE 155
February 2022

Policies

- **Due 9 PM, March** $2^{\mathbf{nd}}$, via Gradescope.
- You are free to collaborate on all of the problems, subject to the collaboration policy stated in the syllabus.
- In this course, we will be using Google Colab for code submissions. You will need a Google account.
- This set uses PyTorch, a Python package for neural networks. We recommend using Google Colab, which comes with PyTorch already installed.

Submission Instructions

- Submit your report as a single .pdf file to Gradescope, under ”Set 6 Report”.
- In the report, **include any images generated by your code** along with your answers to the questions.
- Submit your code by **sharing a link in your report** to your Google Colab notebook for each problem (see naming instructions below). Make sure to set sharing permissions to at least ”Anyone with the link can view”. **Links that can not be run by TAs will not be counted as turned in.** Check your links in an incognito window before submitting to be sure.
- For instructions specifically pertaining to the Gradescope submission process, see `https://www.gradescope.com/get_started#student-submission`.

Google Colab Instructions

For each notebook, you need to save a copy to your drive.

1. Open the github preview of the notebook, and click the icon to open the colab preview.
2. On the colab preview, go to File → Save a copy in Drive.
3. Edit your file name to “lastname_firstname_set_problem”, e.g.”yue_yisong_set6_prob2.ipynb”

1 Class-Conditional Densities for Binary Data [25 Points, 8 EC 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 [9 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: *Using the chain rule of probability:*

$$p(x \mid y) = p(x_D \mid x_{1:D-1}, y = c)p(x_{D-1} \mid x_{1:D-2}, y = c)...p(x_1 \mid y = c) \tag{1}$$

Substituting the definition of θ_{xjc},

$$p(x \mid y) = \theta_{xDc}\theta_{x(D-1)c}...\theta_{x1c} = \prod_{j=1}^{D} \theta_{xjc} \tag{2}$$

Storing each θ_{xjc} requires $C * 2^{j-1}$ parameters because there are two possibilities for each $x_{1:j-1}$. The sum of the geometric series $\sum_{j=1}^{D} C * 2^{j-1} = C * 2^D$. Thus the number of parameters needed is of order $O(C \times 2^D)$.

Problem B [8 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: *Computing the joint probability distribution $p(x \mid y = c)$ also has complexity $O(C \times 2^D)$ because there are D binary features and C classes, which is the same order as the previous part.*

Problem C [4 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: *When the sample size is small, Naive Bayes is likely to give a lower test set error because conditional independence is a good assumption if there are fewer samples. Learning a full model could lead to overfitting when trying to learn a relationship between different features.*

Problem D [4 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: *When the sample size is large, the full model is likely to give a lower test error. Conditional independence is not a good assumption in this case and will lead to underfitting. Given enough data, the model should be able accurately to learn the feature dependencies on each other, which makes the full model a better choice.*

Problem E [8 EC 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? In justifying your answer for the full model, choose either the implementation in 1A or 1B and state your choice. 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: *Not attempted.*

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. We have also uploaded a note on HMMs to the github that might be helpful.

Sequence Prediction

These next few problems will require extensive coding, so be sure to start early!

- You will write an implementation for the hidden Markov model in the cell for `HMM Code` in the notebook given to you, within the appropriate functions where 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.
- You can (and should!) use the helper cells for each of the subproblems in the notebook, namely `2A`, `2Bi`, `2Bii`, `2C`, `2D`, and `2F`. These can be used to run and check your implementations for each of the corresponding problems. The cells provide useful output in an easy-to-read format. There is no need to modify these cells.
- Lastly, the cell for `Utility` contains some functions used for loading data directly from the class github repository. There is no need to modify this cell.

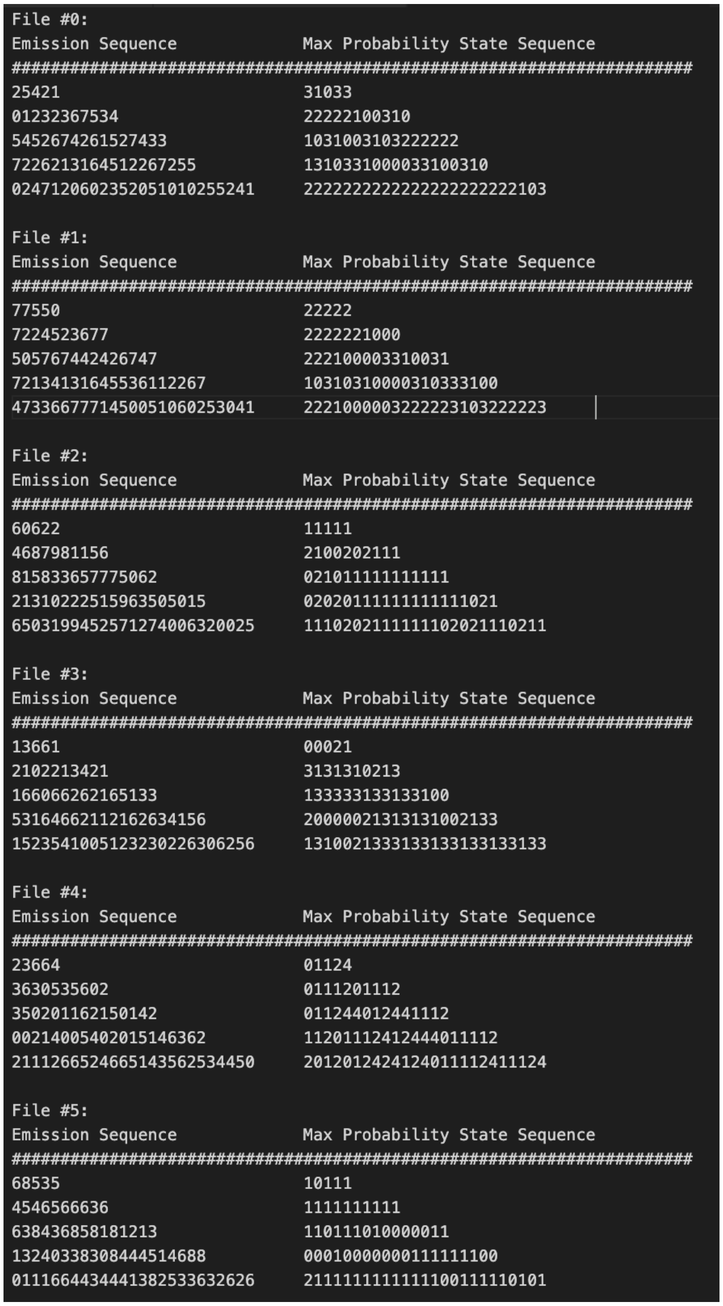
The supplementary data folder of the class github repository contains 6 files titled `sequence_data0.txt`, `sequence_data1.txt`, ... , `sequence_data5.txt`. Each file specifies a **trained** HMM. The first row contains two tab-delimited numbers: the number of states Y and the number of types of observations 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 (in `viterbi()` of the `HiddenMarkovModel` object). Write your implementation well, as we will be reusing it in a later problem. See the end of problem 2B for a big hint! Note that you do not need to worry about underflow in this part.

In your report, show your results on the 6 files. (Copy-pasting the results of the cell for `2A` suffices.)

Solution A: *Link to code*



```
File #0:
Emission Sequence             Max Probability State Sequence
######################################################################
25421                         31033
01232367534                   22222100310
5452674261527433              1031003103222222
7226213164512267255           1310331000033100310
0247120602352051010255241     2222222222222222222222103

File #1:
Emission Sequence             Max Probability State Sequence
######################################################################
77550                         22222
7224523677                    2222221000
505767442426747               222100003310031
72134131645536112267          10310310000310333100
4733667771450051060253041     2221000003222223103222223

File #2:
Emission Sequence             Max Probability State Sequence
######################################################################
60622                         11111
4687981156                    2100202111
815833657775062               021011111111111
21310222515963505015          02020111111111111021
6503199452571274006320025     1110202111111102021110211

File #3:
Emission Sequence             Max Probability State Sequence
######################################################################
13661                         00021
2102213421                    3131310213
166066262165133               133333133133100
53164662112162634156          20000021313131002133
1523541005123230226306256     1310021333133133133133133

File #4:
Emission Sequence             Max Probability State Sequence
######################################################################
23664                         01124
3630535602                    0111201112
350201162150142               011244012441112
00214005402015146362          11201112412444011112
2111266524665143562534450     2012012424124011112411124

File #5:
Emission Sequence             Max Probability State Sequence
######################################################################
68535                         10111
4546566636                    1111111111
638436858181213               110111010000011
13240338308444514688          00010000000111111100
0111664434441382533632626     2111111111111100111110101
```

Problem B [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 (the starting state transition probabilities are defined in `self.A_start` in `HiddenMarkovModel`). Again, write your implementation well, as we will be reusing it in a later problem.

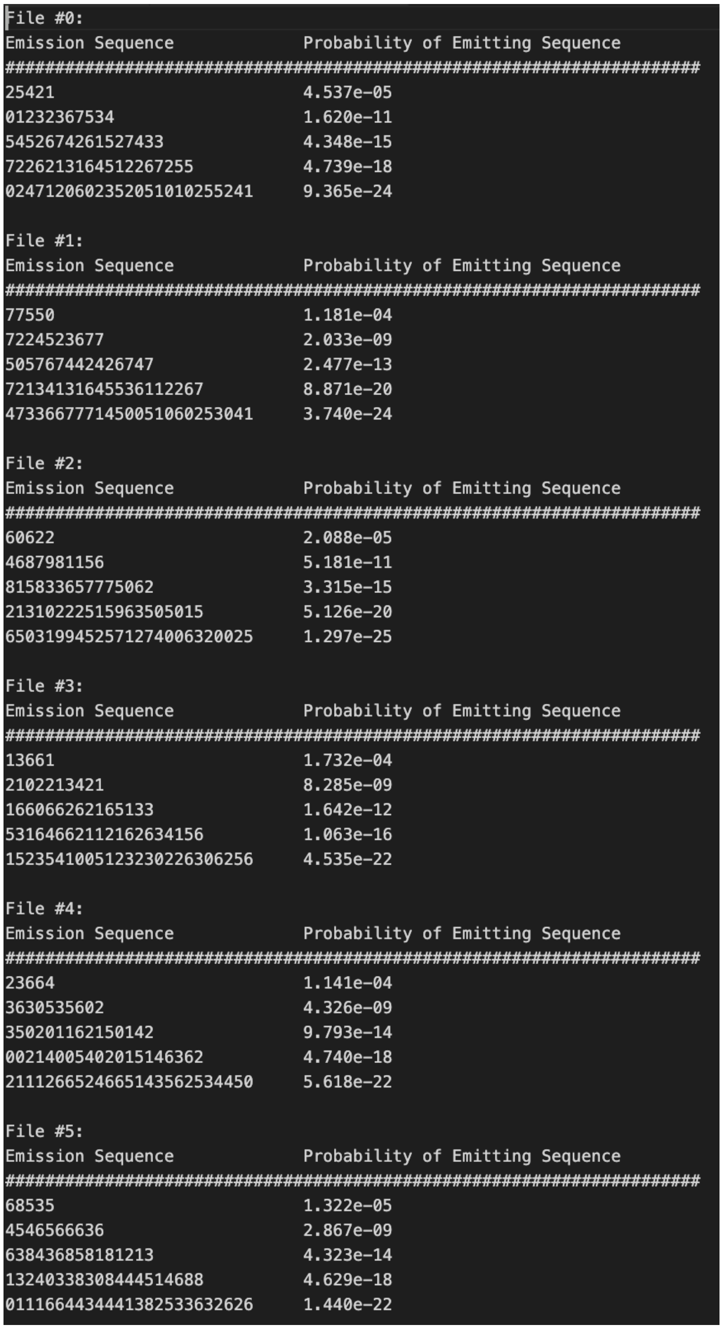
Note that the probability of emitting an input sequence can be found by using either the α vectors from the Forward algorithm or the β vectors from the Backward algorithm. You don't need to worry about this, as it is done for you in `probability_alphas()` and `probability_betas()`.

Implement the Forward algorithm. In your report, show your results on the 6 files.
Implement the Backward algorithm. In your report, show your results on the 6 files.

After you complete problems 2A and 2B, you can compare your results (the probabilities from the forward and backward algorithms should be the same) for the file titled `sequence_data0.txt` with the values given in the table below:

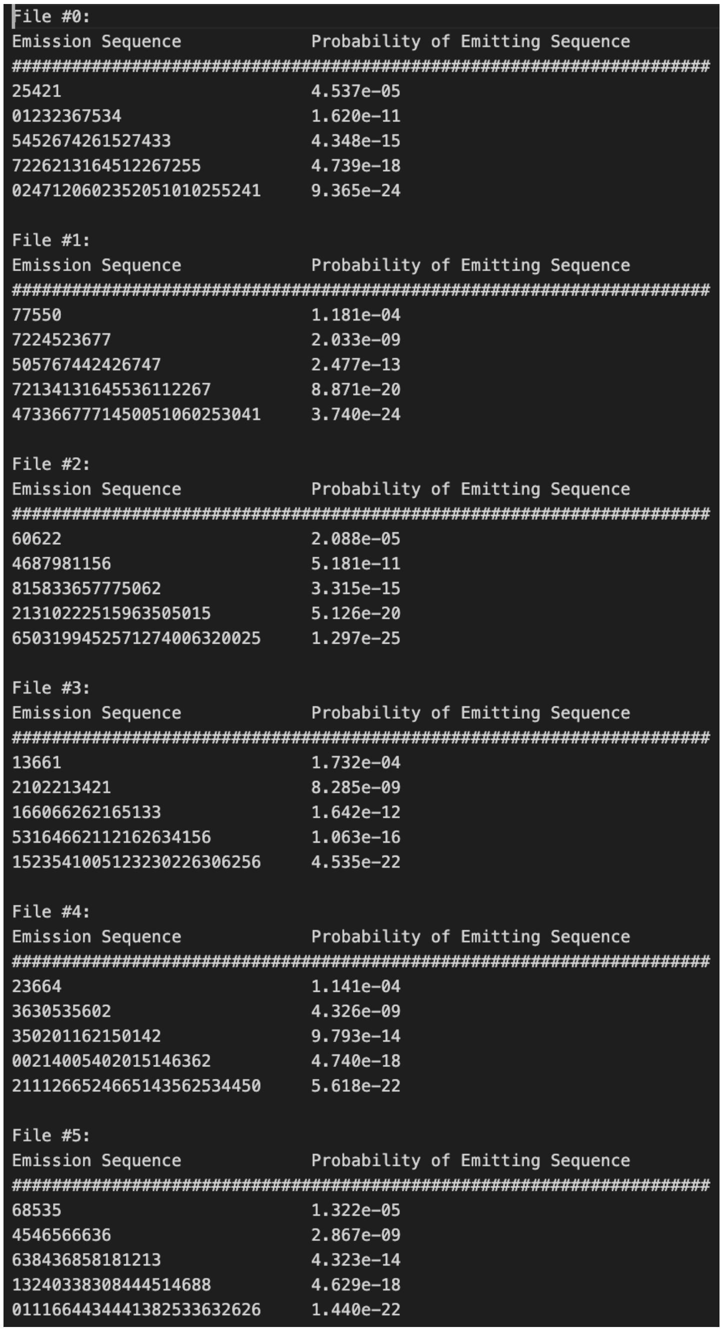
Dataset	Emission Sequence	Max-probability State Sequence	Probability of Sequence
0	25421	31033	4.537e-05
0	01232367534	22222100310	1.620e-11
0	5452674261527433	1031003103222222	4.348e-15
0	7226213164512267255	1310331000033100310	4.739e-18
0	0247120602352051010255241	2222222222222222222222103	9.365e-24

Solution B: *Forward 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:



```
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

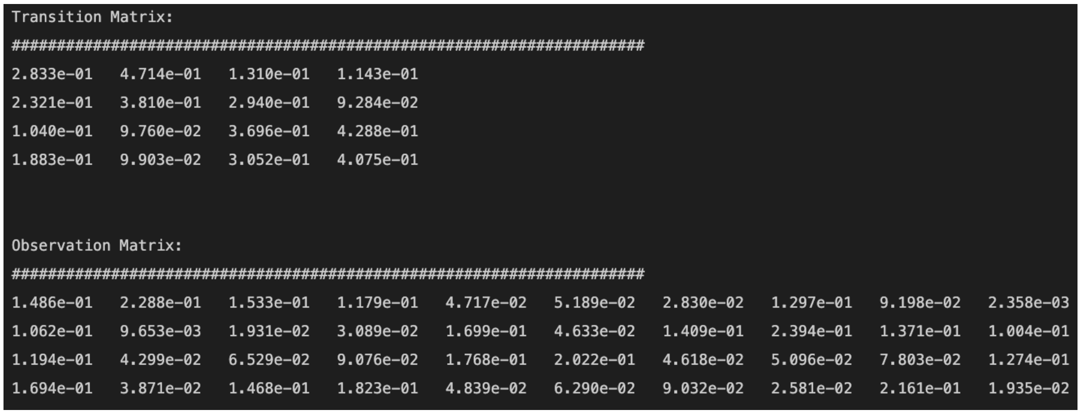
Ron is an avid music listener, and his genre preferences at any given time depend on his mood. Ron's possible moods are happy, mellow, sad, and angry. Ron experiences one mood per day (as humans are known to do) and chooses one of ten genres of music to listen to that day depending on his mood.

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?

Tip: the $(1, 1)$ entry of your transition matrix should be `2.833e-01`, and the $(1, 1)$ entry of your observation matrix should be `1.486e-01`.

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.

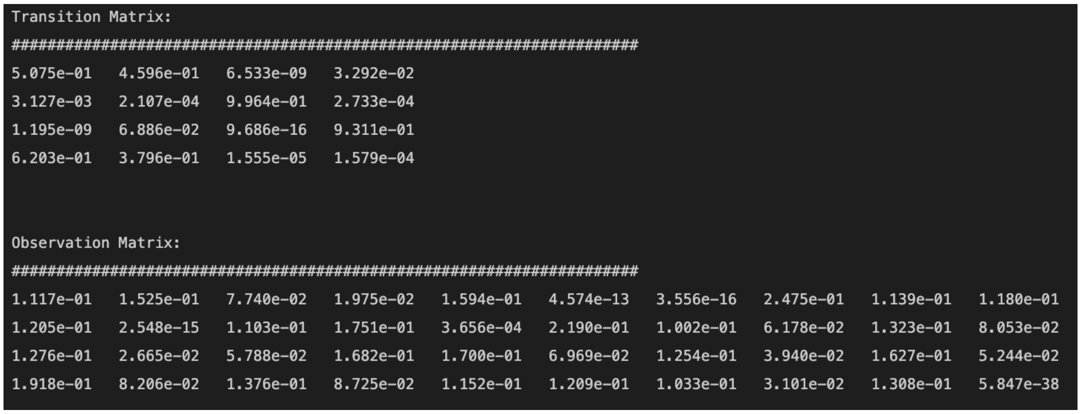
In this problem, we will train an unsupervised Hidden Markov Model on this data. Recall that unsupervised HMM training is done using the Baum-Welch algorithm and will require repeated EM steps. For this problem, we will use 4 hidden states and run the algorithm for 1000 iterations. The transition and 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? Please report the result using random seed state 1, as is done by default in the notebook.

Tips for debugging:

- The rows of the state transition and output emitting matrices should sum to 1.
- Your matrices should not change drastically every iteration.
- After many iterations, your matrices should converge.
- If you used random seed 1 for this computation (as is done by default in the notebook), the $(1, 1)$ entry of the state transition matrix should be `5.075e-01`, and the $(1, 1)$ entry of the output emission matrix should be `1.117e-01`.

Solution D:



```
Transition Matrix:
######################################################################
5.075e-01   4.596e-01   6.533e-09   3.292e-02
3.127e-03   2.107e-04   9.964e-01   2.733e-04
1.195e-09   6.886e-02   9.686e-16   9.311e-01
6.203e-01   3.796e-01   1.555e-05   1.579e-04


Observation Matrix:
######################################################################
1.117e-01   1.525e-01   7.740e-02   1.975e-02   1.594e-01   4.574e-13   3.556e-16   2.475e-01   1.139e-01   1.180e-01
1.205e-01   2.548e-15   1.103e-01   1.751e-01   3.656e-04   2.190e-01   1.002e-01   6.178e-02   1.323e-01   8.053e-02
1.276e-01   2.665e-02   5.788e-02   1.682e-01   1.700e-01   6.969e-02   1.254e-01   3.940e-02   1.627e-01   5.244e-02
1.918e-01   8.206e-02   1.376e-01   8.725e-02   1.152e-01   1.209e-01   1.033e-01   3.101e-02   1.308e-01   5.847e-38
```

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.

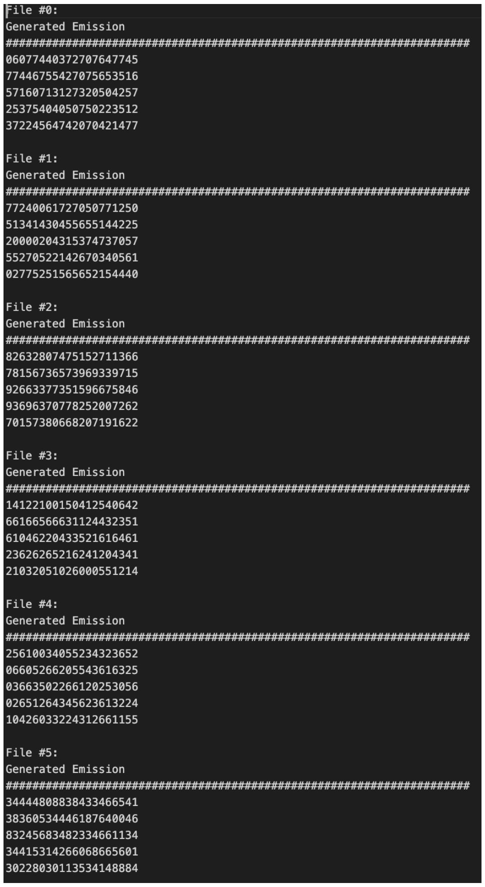
Solution E: *The transition matrix for 2D generally has smaller values than that from 2C, which menas this matrix is more sparse. The matrices from 2C are a likely a better representation of Ron's moods because it is actually trained with the moods, which makes the transition and observation matrices optimal. However, the unsupervised method is not able to learn from these moods. We could improve the unsupervised model by providing a meaningful prior for the transition or observation matrix, based on how Ron's moods might be* *distributed.*

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]: Run the cell for this problem. The code in the cell loads the trained HMMs from the files titled `sequence_data0.txt,...,sequence_data5.txt` and uses the six models to probabilistically generate five sequences of emissions from each model, each of length 20. In your report, show your results.

Solution F:



```
File #0:
Generated Emission
######################################################################
06077440372707647745
77446755427075653516
57160713127320504257
25375404050750223512
37224564742070421477

File #1:
Generated Emission
######################################################################
77240061727050771250
51341430455655144225
20000204315374737057
55270522142670340561
02775251565652154440

File #2:
Generated Emission
######################################################################
82632807475152711366
78156736573969339715
92663377351596675846
93696370778252007262
70157380668207191622

File #3:
Generated Emission
######################################################################
14122100150412540642
66166566631124432351
61046220433521616461
23626265216241204341
21032051026000551214

File #4:
Generated Emission
######################################################################
25610034055234323652
06605266205543616325
03663502266120253056
02651264345623613224
10426033224312661155

File #5:
Generated Emission
######################################################################
34444808838433466541
38360534446187640046
83245683482334661134
34415314266068665601
30228030113534148884
```

Visualization & Analysis

Once you have implemented the HMM code part of the notebook, load and run the cells for the following subproblems. Here you will apply the HMM you have implemented to the Constitution. There is no coding required for this part, only analysis.

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: *A and O are both sparse matrices, but O is slightly more sparse since it is larger. The sparsity of A implies that very few states are likely to be achieved from a given state. The sparsity of O implies that very few observations are likely to be generated from a given state.*

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 is increased, the emission sentences become more and more comprehensible, and resembling sentences from the constitution. For example, one hidden state produces a nonsensical sentence: "To the and meet their extend same states to from in any the time any elected states of the by proper they the pursuance for..." but once we have 4 hidden states, the emitted sentences make more sense: "Be of established inhabitant employed and proposing to such seat provided their test faithfully at committed or no one may witnesses of three executive of..." It appears that there may be some overfitting with 16 hidden states, as the sentence is less comprehensible and more skewed to unique words in the constitution. In the special case of one hidden state, the words are basically ordered, because they have the same probability of appearing anywhere in the sentence. In general, we can fit better to the training data by allowing more hidden states because this allows for more possible associations in the A and O matrices, which can only help on the training set.*

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 3 is semantically meaningful from a grammatical standpoint. While many of the other states are grouped around a certain concept, state 3 seems to be grouped around comparison words. "two," "three," and "every" are the most important words in this state in addition to other numbers and comparison words such as "different."*