Machine Learning & Data Mining
Homework 6

Caltech CS/CNS/EE 155
February 2022

Dallas Taylor

Google Colab Links

- Problem 2 : https://colab.research.google.com/drive/15r_bA9PD9JrJNl8VHZYo5IvERAVXYigk?usp=sharing

Machine Learning & Data Mining
Homework 6

Caltech CS/CNS/EE 155
February 2022

Dallas Taylor

---

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: *Here, we use the chain rule of probability to factorize $p(x \mid y)$. The chain rule allows us to factorize this as the following:*

$$\begin{aligned} p(x \mid y) &= p(x_1 \mid y = c) \cdot p(x_2 \mid x_1, y = c) \cdot \ldots \cdot p(x_D \mid x_{1,\ldots,D-1}, y = c) \\ &= \theta_{x1c} \cdot \theta_{x2c} \cdot \ldots \cdot \theta_{xDc} \\ &= \prod_{j=1}^{D} \theta_{xjc} \end{aligned}$$

From the above, we can clearly determine that if we wish to store each θ_{xjc}, then we are required to store 2 parameters for each of $j-1$ prefixes (since we are considering binary features). Thus we store a total of $\sum_{j=1}^{D} 2^{j-1}$ parameters. Thus, considering each $c \in C$, we have a space complexity of $\mathcal{O}(C * 2^D)$ parameters.

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: *When considering an arbitrary x, c, we would need to store parameters for each possible value of x. We know that x has a maximum of D binary features, and thus we store $\mathcal{O}(2^D)$ parameters for each x, and furthermore we clearly store $\mathcal{O}(C * 2^D)$ parameters when considering each $c \in C$. Thus, in order to compute $p(x \mid y = c)$ without factorization, we have a space complexity of $\mathcal{O}(C * 2^D)$ parameters. This is the same order*

of parameters as described in Problem A.

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 we have a small sample size N, then we can see that our Naive Bayes model will likely give lower test set error because of the smaller number of parameters. Although the Naive Bayes and full models both had the same order of parameters, the Naive Bayes had an overall lower number of parameters. Thus, when considering a small N, we can avoid potential overfitting due to excessive parameters by utilizing the Naive Bayes model. Additionally, we can logically conclude that since the Naive Bayes model utilizes conditional independence, we should also be able to perform better when utilizing a small N.*

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 we have a large sample size N, then we can see that our full model will likely give lower test set error because of the larger number of parameters. Thus, we are able to utilize a large amount of parameters in order to maximize learning of conditional dependence with a sample size that properly represents the data. Additionally, we can logically conclude that since the full model utilizes conditional dependence, we should be able to perform better when utilizing a large N, whereas Naive Bayes will most likely underfit the data.*

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: *The computational complexity of making a prediction using Naive Bayes for a single test case is _.*

The computational complexity of making a prediction using the full model for a single test case is _.

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:

Dataset	*Emission Sequence*	*Max-probability State Sequence*
0	*25421*	*31033*
0	*01232367534*	*22222100310*
0	*5452674261527433*	*1031003103222222*
0	*7226213164512267255*	*1310331000033100310*
0	*0247120602352051010255241*	*222222222222222222222103*
1	*77550*	*22222*
1	*7224523677*	*2222221000*
1	*505767442426747*	*222100003310031*
1	*72134131645536112267*	*10310310000310333100*
1	*4733667771450051060253041*	*22210000032222231032222 23*
2	*60622*	*11111*
2	*4687981156*	*2100202111*
2	*815833657775062*	*021011111111111*
2	*21310222515963505015*	*02020111111111111021*
2	*65031994525712740063200 25*	*11102021111111020211102 11*
3	*13661*	*00021*
3	*2102213421*	*3131310213*
3	*166066262165133*	*133333133133100*
3	*53164662112162634156*	*20000021313131002133*
3	*15235410051232302263062 56*	*13100213331331333133131 33133*
4	*23664*	*01124*
4	*3630535602*	*0111201112*
4	*350201162150142*	*011244012441112*
4	*00214005402015146362*	*11201112412444011112*
4	*21112665246651435625344 50*	*20120124241240111124112 411124*
5	*68535*	*10111*
5	*4546566636*	*1111111111*
5	*638436858181213*	*110111010000011*
5	*13240338308444514688*	*00010000000111111100*
5	*01116644344413825336326 26*	*21111111111111001111101 01*

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	22222222222222222222222103	9.365e-24

Solution B:

Dataset	*Emission Sequence*	*Probability of Sequence (F)*	*Probability of Sequence (B)*
0	*25421*	*4.537e-05*	*4.537e-05*
0	*01232367534*	*1.620e-11*	*1.620e-11*
0	*5452674261527433*	*4.348e-15*	*4.348e-15*
0	*7226213164512267255*	*4.739e-18*	*4.739e-18*
0	*0247120602352051010255241*	*9.365e-24*	*9.365e-24*
1	*77550*	*1.181e-04*	*1.181e-04*
1	*7224523677*	*2.033e-09*	*2.033e-09*
1	*505767442426747*	*2.477e-13*	*2.477e-13*
1	*72134131645536112267*	*8.871e-20*	*8.871e-20*
1	*4733667771450051060253041*	*3.740e-24*	*3.740e-24*
2	*60622*	*2.088e-05*	*2.088e-05*
2	*4687981156*	*5.181e-11*	*5.181e-11*
2	*815833657775062*	*3.315e-15*	*3.315e-15*
2	*21310222515963505015*	*5.126e-20*	*5.126e-20*
2	*6503199452571274006320025*	*1.297e-25*	*1.297e-25*
3	*13661*	*1.732e-04*	*1.732e-04*
3	*2102213421*	*8.285e-09*	*8.285e-09*
3	*166066262165133*	*1.642e-12*	*1.642e-12*
3	*53164662112162634156*	*1.063e-16*	*1.063e-16*
3	*1523541005123230226306256*	*4.535e-22*	*4.535e-22*
4	*23664*	*1.141e-04*	*1.141e-04*
4	*3630535602*	*4.326e-09*	*4.326e-09*
4	*350201162150142*	*9.793e-14*	*9.793e-14*
4	*00214005402015146362*	*4.740e-18*	*4.740e-18*
4	*2111266524665143562534450*	*5.618e-22*	*5.618e-22*
5	*68535*	*1.322e-05*	*1.322e-05*
5	*4546566636*	*2.867e-09*	*2.867e-09*
5	*638436858181213*	*4.323e-14*	*4.323e-14*
5	*13240338308444514688*	*4.629e-18*	*4.629e-18*
5	*0111664434441382533632626*	*1.440e-22*	*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:

State transition matrix:

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

Output emission matrix:

$$O = \begin{matrix} \\ \\ \\ \\ \end{matrix} \begin{matrix} 1.486e-01 & 2.288e-01 & 1.533e-01 & 1.179e-01 & 4.717e-02 & 5.189e-02 \\ 1.062e-01 & 9.653e-03 & 1.931e-02 & 3.089e-02 & 1.699e-01 & 4.633e-02 \\ 1.194e-01 & 4.299e-02 & 6.529e-02 & 9.076e-02 & 1.768e-01 & 2.022e-01 \\ 1.694e-01 & 3.871e-02 & 1.468e-01 & 1.823e-01 & 4.839e-02 & 6.290e-02 \end{matrix}$$

$$\begin{bmatrix} 2.830e-02 & 1.297e-01 & 9.198e-02 & 2.358e-03 \\ 1.409e-01 & 2.394e-01 & 1.371e-01 & 1.004e-01 \\ 4.618e-02 & 5.096e-02 & 7.803e-02 & 1.274e-01 \\ 9.032e-02 & 2.581e-02 & 2.161e-01 & 1.935e-02 \end{bmatrix}$$

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: *State transition matrix:*

$$A = \begin{bmatrix} 5.075e-01 & 4.596e-01 & 6.533e-09 & 3.292e-02 \\ 3.127e-03 & 2.107e-04 & 9.964e-01 & 2.733e-04 \\ 1.195e-09 & 6.886e-02 & 9.686e-16 & 9.311e-01 \\ 6.203e-01 & 3.796e-01 & 1.555e-05 & 1.579e-04 \end{bmatrix}$$

Output emission matrix:

$$O = \begin{bmatrix} 1.117e-01 & 1.525e-01 & 7.740e-02 & 1.975e-02 & 1.594e-01 & 4.574e-13 \\ 1.205e-01 & 2.548e-15 & 1.103e-01 & 1.751e-01 & 3.656e-04 & 2.190e-01 \\ 1.276e-01 & 2.665e-02 & 5.788e-02 & 1.682e-01 & 1.700e-01 & 6.969e-02 \\ 1.918e-01 & 8.206e-02 & 1.376e-01 & 8.725e-02 & 1.152e-01 & 1.209e-01 \end{bmatrix}$$

$$\begin{bmatrix} 3.556e-16 & 2.475e-01 & 1.139e-01 & 1.180e-01 \\ 1.002e-01 & 6.178e-02 & 1.323e-01 & 8.053e-02 \\ 1.254e-01 & 3.940e-02 & 1.627e-01 & 5.244e-02 \\ 1.033e-01 & 3.101e-02 & 1.308e-01 & 5.847e-38 \end{bmatrix}$$

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: *Both the transition and emission matrices from 2C has a much tighter range of values than that of 2D ($[e-02, e-01]$ and $[e-03, e-01]$ vs $[e-16, e-01]$ and $[e-38, e-01]$). We can additionally see that there are many values in both A and O for 2D that are very very small.*

I think that the supervised learning matrix provides a more accurate representation of Ron's mood and how they affect his music choices because of the consistency of values that are found in A and O. This consistency is most likely due to the fact that we are utilizing labeled data.

Thus, we can clearly see that one way we might be able to improve unsupervised learning is by increasing the volume of data that it is utilized in our learning.

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
30275324625466777310
12674540440200457677
65127570275777613572
33271771605315156154
52155156400525275160

File 1:
Generated Emission
42241166625457042510
07435207007450050275
01620520257047664536
57242257747004251454
75741077254777235530

File 2:
Generated Emission
86717153525423029785
03252537990319769129

45269848326305046269
73859765756277325723
74667023086576045839

File 3:
Generated Emission
43261316252113120153
66360635115351614426
65223525262413306262
42026266566233166036
63462155115414130206

File 4:
Generated Emission
03251043162503236110
43656361111150224220
46414106612054622215
54615016610346503303
62434150113514652524

File 5:
Generated Emission
08864624822644131653
06383807288633656001
53440708688455601534
02413566234643518222
64180208530841446134

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: *The sparsity of the trained A and O matrices indicate that a majority of our data points within them are 0. The clear outliers are also of low value, but are clearly distinguishable from the surrounding data points due to the sparsity.*

This sparsity affects the behavior such that it is very limited. The large volume of 0 or near 0 entries in A and 0 *allow for each state to only "contribute" to computation a limited amount.*

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, we can see that the sample emission sentences from the HMM start to make more grammatical and contextual "sense". The sentences begin to form understandable and coherent declarations. When there is only one hidden state, the sample emission sentence is very difficult to comprehend and does not make grammatical sense. When the number of hidden states is unknown for a fixed observation set, we can increase the training data likelihood by allowing more hidden states, and thus increasing sparsity and the strength of learned connections.*

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

Solution I: *I chose state 2.*

This state represents major political vocabulary. It is also notable that most of the words in this cloud are nouns, and are thus probably often the subject of sentences.

This state differs from the other states such that it mainly includes nouns such as president, member, time, etc. Additionally, we can see that we have key political vocabulary such as legislature, state, senate, etc.