Introduction to Graphical Models: Mini Project

Due on December 8, 2017 at 11:55 pm

Taught by Assisitant Professor Umut Simsekli

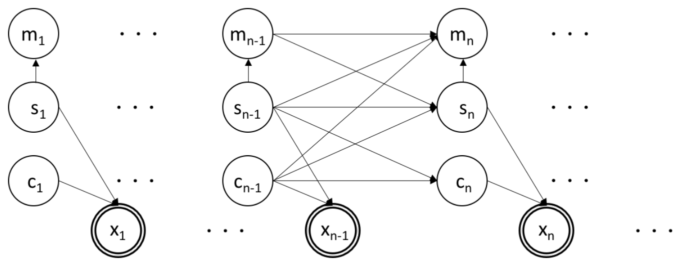
Pengfei MI, Rui SONG, Yanting LI

This document is the report for the Mini-Project of the course "Introduction to Graphical Models" of Master 2 Data Science. This project is done in group of three, with group members: Pengfei MI, Rui SONG, and Yanting LI.

Question 1

Draw the directed graphical model for the described model.

Solutions:
The directed graphical model for our model can be drawn like below



Question 2

Construct the transition matrix.

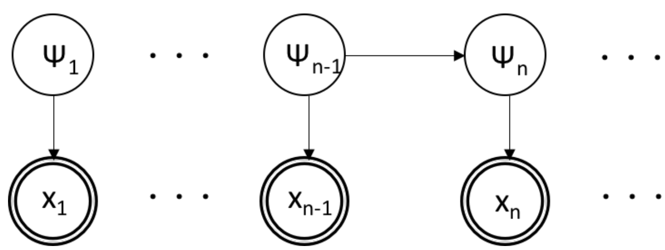
Solutions:
Define a variable $\Psi_n \equiv [s_n, m_n, c_n]$. The set of all states can be listed in a vector Ω then the state of the system at pixel n can be represented as $\Psi_n = \Omega(j)$ where $j \in \{1, 2, ..., (S \times M \times C)\}$ and $C = max_{s,k} l(s)$ ($= 7$ in our model). Then the transition matrix is:

$$
\begin{aligned}
A(i,j) &= p(\Psi_{n+1} = \Omega(i) | \Psi_n = \Omega(j)) \\
&= p(c_{n+1} = c_i, s_{n+1} = s_i, m_{n+1} = m_i | c_n = c_j, s_n = s_j, m_n = m_j) \\
&= p(c_{n+1} = c_i | c_n = c_j, s_n = s_j) p(s_{n+1} = s_i, m_{n+1} = m_i | c_n = c_j, s_n = s_j, m_n = m_j) \\
&= p(c_{n+1} = c_i | c_n = c_j, s_n = s_j) p(s_{n+1} = s_i | c_n = c_j, s_n = s_j, m_n = m_j) p(m_{n+1} = m_i | s_{n+1} = s_i, c_n = c_j, s_n = s_j) \\
&= \begin{cases} \delta(c_{n+1} - c_n - 1)\delta(s_{n+1} - s_n)\delta(m_{n+1} - m_n) & \text{if } c_n \neq l(s_n) \\ \delta(c_{n+1} - 1)\tau(s_{n+1}|s_n, m_n)\tau(m_{n+1}|s_{n+1}, m_n) & \text{if } c_n = l(s_n) \end{cases}
\end{aligned} \tag{1}
$$

where $\tau(s_{n+1}|s_n, m_n)$ and $\tau(m_{n+1}|s_{n+1}, m_n)$ are defined in the description.
Then the observation model can be written like:

$$
p(x_n|\Psi_n) = \prod_{i=0}^{1} \mathcal{N}(x_n; \mu_i, \sigma_i^2)^{\mathbb{1}(f(\Psi_n)=i)}
$$

Then the graphical model becomes:



Question 3

Simulate the HMM and visualize the simulated data.

Solutions:
The general idea to simulate the HMM model is: for each states $j \in \{1, 2, ..., (S \times M \times C)\}$, we generate a random number r according to a uniform distribution in the interval $[0, 1]$. Then we find the state k such that $\sum_{i=1}^{k} A(i, j) < r$ and $\sum_{i=1}^{k+1} A(i, j) >= r$. Then we will transfer from state j to state i.
For the code, please look at file *template_code.ipynb*. The barcode simulated is : As we can see this looks

[1 8 4 5 2 5 9 5 0 0 7 9]

very much like a real scanline. And the barcode string is ”184525950079”.

Question 4

Fill code part 1, 2, and 3

Solutions:
Please refer to file *template_code.ipynb*.

Question 5

Compute the filtering distribution and marginals

Solutions:
The filtering distribution $p(\Psi_n|x_{1:n})$ can be got from the Bayesian rule:

$$p(\Psi_n|x_{1:n}) = \frac{p(\Psi_n, x_{1:n})}{p(x_{1:n})}$$

where $p(\Psi_n, x_{1:n})$ is the message $\alpha_{n|n}(\Psi_n)$ which can be got by forward recursion. Since $x_{1:n}$ is our observation, $p(x_{1:n})$ is therefore a constant. So $p(\Psi_n|x_{1:n})$ is the normalization of $p(\Psi_n, x_{1:n})$.

To compute the marginals $p(s_n|x_{1:n})$, $p(c_n|x_{1:n})$ and $p(m_n|x_{1:n})$, since they are not independent, we need to sum up all $p(\Psi_n|x_{1:n})$ with the same s_n for computing $p(s_n|x_{1:n})$, and the same process for $p(c_n|x_{1:n})$ and $p(m_n|x_{1:n})$.

Please refer to file *template_code.ipynb* part 4.

Question 6

Compute the smoothing distribution and marginals.

Solutions:
Similar with the previous question, the smoothing distribution $p(\Psi_n|x_{1:N})$ can be got from:

$$p(\Psi_n|x_{1:N}) = \frac{p(\Psi_n, x_{1:N})}{p(x_{1:N})} \propto p(\Psi_n, x_{1:N})$$

by message $\beta_{n|n+1}(\Psi_n)\alpha_{n|n}(\Psi_n)$ via Forward-Backward recursion.

The computation of the smoothing distribution of s_n, c_n and m_n are also the sum of $p(\Psi_n|x_{1:N})$ with the same s_n, c_n and m_n respectively.

Please refer to file *template_code.ipynb* part 5.

Question 7

Compute the most-likely path by using the Viterbi algorithm.

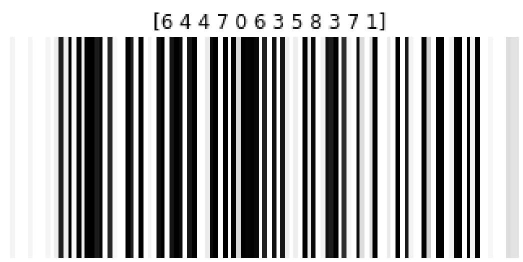
Solutions:
We apply the Viterbi algorithm in backward pass.
That is, for $t \in \{1, 2, ..., T-1\}$, when we compute $\beta_{t|t}(x_t)$ using $\beta_{t|t+1}(x_t)$, instead of using the sum of terms of all x_{t+1}, we keep only the maximum term. At the same time, we also record the argmax in a matrix. After we get the argmax of x_1, we generate the most likely path by the argmax matrix.
Please refer to file *template_code.ipynb* "Viterbi algorithm" part.

Question 8

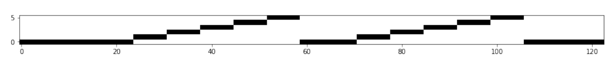
Run algorithms on different random barcodes.

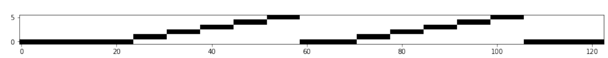
Solutions:
In this question, we tested the program with observation noise 20 and 50. Set $\mu_0 = 255$, $\mu_1 = 0$. and $\sigma_0^2 = \sigma_1^2 = 1$.

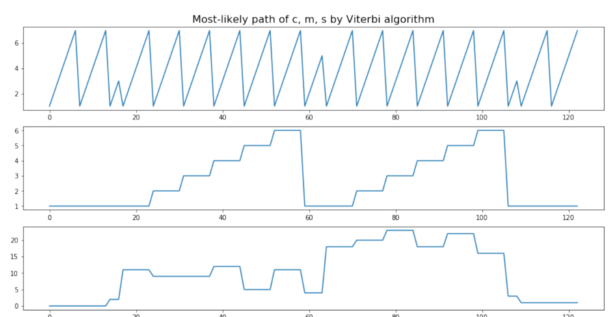
With obs_noise = 20, we generated the following barcode:



The filtering distribution, smoothing distribution and the most-likely path are shown below:







With obs_noise = 80:
When the noise is increased, in both filtering and smoothing distribution, the result becomes more "ambiguous". In each state, a specific result is got "later" than the previous case with a smaller noise.

[0 3 8 0 5 6 6 1 7 6 0 6]