Problem Set #8

Bennett Hellman
6.689 - Advances in Computer Vision
MIT

April 26, 2022

Problem 1 *Markov Network*

Since node a is only connected to node b:

$$P(a) = km_{ba}(a)$$

and we know since node b is connected to $c\,\&\,d$:

$$m_{ba}(a) = \Psi(a,b)m_{db}(b)m_{cb}(b)$$

moving right in the graph:

$$m_{db}(b) = \Psi(b,d)m_{ed}(d)$$

$$m_{ed}(d) = \Psi(d,e)m_{fe}(e)$$

$$m_{fe}(e) = \Phi(e,f)\begin{bmatrix}1\\0\end{bmatrix}$$

and the bottom node:

$$m_{cb}(b) = \Phi(b,c)\begin{bmatrix}0\\1\end{bmatrix}$$

In summary:

$$P(a) = k \times \Psi(a,b) \cdot (\Phi(b,c) \cdot \begin{bmatrix}0\\1\end{bmatrix} \times \Psi(b,d) \cdot \Psi(d,e) \cdot \Phi(e,f) \cdot \begin{bmatrix}1\\0\end{bmatrix})$$

When $\alpha = 0.9$:

$$\begin{bmatrix}P(a=1)\\P(a=0)\end{bmatrix} = \begin{bmatrix}0.30487805\\0.69512195\end{bmatrix}$$

When $\alpha = 0.6$:

$$\begin{bmatrix}P(a=1)\\P(a=0)\end{bmatrix} = \begin{bmatrix}0.42118227\\0.57881773\end{bmatrix}$$

First, let's note that b will always likely look like c (in isolation) and e to f. Lets simplify and say that e is always likely to be 0. When α is large, all nodes are similar to ones they are connected with. Since b is connected both a nodes likely to be $0 \,\&\, 1$, as $\lim_{\alpha \longrightarrow \infty} b = 0.5$ and thus $a = 0.5$. However, with an $\alpha = 0.9$, the probability of $b = 0$ deteriorates along the chain from f making c the stronger influence.

When α is smaller, it is a little more complicated. At $\alpha = 0.6$, connections affiliated with α are a near coin-flip. It is a coin-flip for d and b from the right side of the network. However, b is still very much like c. Therefore, b should have a higher probability of being 1 than the previous scenario. However, it is a coin-flip from b to a, lowering the probability.

```
a=0.9
phi_bc = np.array([[0.9,0.1],[0.1,0.9]])
phi_ef = phi_bc
psi_ab = np.array([[a,1-a],[1-a,a]])
psi_bd = psi_ab
psi_de = psi_bd
unnorm = psi_ab@((phi_bc@np.array([[0],[1]]))
    *(psi_bd@psi_de@phi_ef@np.array([[1],[0]])))
norm = unnorm/np.sum(unnorm)
print(f'When $\alpha$={a}, P(a=0) = {norm[0]} & P(a=1)={norm[1]}')
a=0.6
phi_bc = np.array([[0.9,0.1],[0.1,0.9]])
phi_ef = phi_bc
psi_ab = np.array([[a,1-a],[1-a,a]])
psi_bd = psi_ab
psi_de = psi_bd
unnorm = psi_ab@((phi_bc@np.array([[0],[1]]))
    *(psi_bd@psi_de@phi_ef@np.array([[1],[0]])))
norm = unnorm/np.sum(unnorm)
print(f'When $\alpha$={a}, P(a=0) = {norm[0]} & P(a=1)={norm[1]}')
```

```
When $\alpha$=0.9, P(a=0) = [0.30487805] & P(a=1)=[0.69512195]
When $\alpha$=0.6, P(a=0) = [0.42118227] & P(a=1)=[0.57881773]
```

Problem 2 *Progress Report*

The introduction, literature review, and approach are all on the subsequent page in CVPR format. In general, we have a well mapped out plan. Our results now are working for masking an image within an image. However, our model architecture is set up to mask it in a black and white image which is not our intent. Once we fix the architecture, we are using multiple different loss functions and peripheral encryption. We have the tasks divided and have a clear plan for creating a quality project on time.