1 Dependencies in a Directed Graphical Model

1.1 In the graphical model of Figure 1, is $W_{d,n} \perp W_{d,n+1} \mid \theta_d, \beta_{1:K}$?

Yes.

1.2 In the graphical model of Figure 1, is $\theta_d \perp \theta_{d+1} \mid Z_{d,1:N}$?

No.

1.3 In the graphical model of Figure 1, is $\theta_d \perp \theta_{d+1} \mid \alpha, Z_{1:D,1:N}$?

Yes.

1.4 In the graphical model of Figure 2, is $W_{d,n} \perp W_{d,n+1} \mid \Lambda_d, \beta_{1:K}$?

No.

1.5 In the graphical model of Figure 2, is $\theta_d \perp \theta_{d+1} \mid Z_{d,1:N}, Z_{d+1,1:N}$?

No.

1.6 In the graphical model of Figure 2, is $\Lambda_d \perp \Lambda_{d+1} \mid \Phi, Z_{1:D,1:n}$?

No.

2 Likelihood of a Tree Graphical Model

2.1 Implement a dynamic programming algorithm that for a given T, Θ, β computes $p(\beta \mid T, \Theta)$

This algorithm is based on the derivation given in video 7.6b, module 7.

We start by defining a binary-tree shaped DGM, with root node A, and nodes B, C being children of A, and O_1, O_2 being both leaves and children of B.

Now, we can define a function

$$s(u, i) = p(O \cap \downarrow u \mid u = i)$$

Here O refers to the set of all observations (located on the leaves) and $\downarrow u$ denotes the set of nodes that are descendants of u.

We know that all nodes (and thus all observations) are descended from the root, thus $s(A, i)$ denotes the probability of all observations, given that the root takes on value i. We know that we can get the probability of only the observations from the conditional probability using the product rule and marginalization

$$s(A, i) = p(O \mid A = i)$$

$$p(O \mid A = i) = \frac{p(O, A = i)}{p(A = i)}$$

$$p(O) = \sum_A \frac{p(O, A = i)}{p(A = i)} p(A = i)$$

$$p(O) = \sum_i s(A, i) p(A = i)$$

Furthermore, we can define a recurrence relationship for s, using the conditional independence properties encoded in the DGM

$$s(A,i) = p(O\cap\downarrow B \mid A=i)p(O\cap\downarrow C \mid A=i)$$

"Unmarginalizing" either of these terms results in

$$p(O\cap\downarrow B \mid A=i) = \sum_j p(O\cap\downarrow B, B=j \mid A=i)$$

Using the product rule and conditional independence

$$\sum_j p(O\cap\downarrow B, B=j \mid A=i) = \sum_j p(O\cap\downarrow B \mid B=j)p(B=j \mid A=i)$$

Now, we recognize that

$$p(O\cap\downarrow B \mid B=j) = s(B,j)$$

Putting this together

$$s(A,i) = \sum_j s(B,j)p(B=j \mid A=i) \times \sum_k s(C,k)p(C=k \mid A=i)$$

We can also see that when we reach node B, who has children that are leaves, we reach our base case and the recursion ends.

$$s(B,i) = p(O_1 \mid B=i)p(O_2 \mid B=i)$$

Now, in order to calculate $p(O)$, we will need to calculate $s(A,i)$ with i taking on all K values. We also see that as the recursion becomes a layer removed from A we will calculate $s(B,j)$, whose computation does not depend on i. Thus we will do the same computations K times. As $s(B,j)$ is itself a recurrent function call with the same problem, these inefficiencies quickly add up. If we instead make use of dynamic programming, and save the result of each $s(u,i)$ that we compute in a map with (u,i) as the key, we will only need to calculate each item once. This makes our algorithm time complexity $O(Kn)$ where $n = ||nodes||$ as opposed to K^d where d is depth of the tree, a great improvement.

Code that implements this algorithm can be found in Appendix 1.

2.2 Report $p(\beta \mid T, \Theta)$ for each given data

2.2.1 Small tree

Sample: 0 Beta: [nan 2. nan 0. 2.] Likelihood: 0.009786075668602368

Sample: 1 Beta: [nan 3. nan 0. 0.] Likelihood: 0.015371111945909397

Sample: 2 Beta: [nan 1. nan 0. 0.] Likelihood: 0.02429470256988136

Sample: 3 Beta: [nan 0. nan 3. 4.] Likelihood: 0.005921848333806081

Sample: 4 Beta: [nan 3. nan 3. 3.] Likelihood: 0.016186321212555956

2.2.2 Medium tree

Sample: 0
Beta: [nan nan nan nan nan nan nan 1. 3. nan nan nan nan 4. nan nan 1. nan nan nan 0. nan nan 3. nan nan 0. nan 0. 4. nan nan 1. 0. 4. 0. 1. 1. nan 3. 3. 1. 0. 0. 2. 4. nan 1. 2. 3. 0.]
Likelihood: 1.7611947348905713e-18

Sample: 1
Beta: [nan nan nan nan nan nan nan 0. 0. nan nan nan nan 4. nan nan 3. nan nan nan 0. nan nan 3. nan nan 1. nan 3. 1. nan nan 4. 3. 1. 1. 4. 3. nan 3. 1. 0. 2. 4. 3. 2. nan 4. 4. 0. 0.]
Likelihood: 2.996933026124685e-18

Sample: 2
Beta: [nan nan nan nan nan nan nan 1. 3. nan nan nan nan 2. nan nan 2. nan nan nan 1. nan nan 4. nan nan 4. nan 4. 4. nan nan 2. 1. 2. 3. 0. 3. nan 3. 2. 2. 2. 1. 4. 2. nan 4. 3. 3. 4.]
Likelihood: 2.891411201505415e-18

Sample: 3
Beta: [nan nan nan nan nan nan nan 0. 0. nan nan nan nan 2. nan nan 4. nan nan nan 1. nan nan 3. nan nan 2. nan 4. 2. nan nan 4. 4. 2. 3. 0. 3. nan 3. 4. 1. 2. 3. 4. 2. nan 1. 0. 3. 0.]
Likelihood: 4.6788419411270006e-18

Sample: 4
Beta: [nan nan nan nan nan nan nan 4. 1. nan nan nan nan 0. nan nan 1. nan nan nan 1. nan nan 0. nan nan 1. nan 0. 1. nan nan 1. 1. 2. 4. 1. 0. nan 3. 0. 0. 3. 3. 3. 1. nan 2. 3. 3. 1.]
Likelihood: 5.664006737201378e-18

2.2.3 Large tree

Sample: 0
Likelihood: 3.63097513075208e-69
Sample: 1
Likelihood: 3.9405421986921234e-67
Sample: 2
Likelihood: 5.549061147187144e-67
Sample: 3
Likelihood: 9.89990102807915e-67
Sample: 4
Likelihood: 3.11420969368965e-72

3 Super Epicentra - Expectation-Maximization

3.1 Derive an EM algorithm for the model

The outline of the expectation maximization algorithm is given in section 9.3 in Pattern Recognition and Machine Learning, by Bishop [1]. The outline is summarized as follows:

Given a joint distribution $p(Y, Z \mid \theta)$ the goal is to maximize the likelihood $p(Y \mid \theta)$ w.r.t. θ.

1. Choose initial parameters θ
2. Evaluate $p(Z \mid Y, \theta^{old})$
3. Evaluate θ^{new} where

$$\theta^{new} = argmax_{\theta} Q(\theta, \theta^{old})$$

and

$$Q(\theta, \theta^{old}) = \sum_Z p(Z \mid Y, \theta^{old}) ln(p(Y, Z \mid \theta))$$

4. Check for convergence, we deem the algorithm to have converged if $p(Y \mid \theta^{new}) - p(Y \mid \theta^{old}) < \varepsilon$. If not converged, repeat from step 2. Otherwise θ^{new} are our parameter values.

In the outline above, Y denotes the set of observed variables, Z denotes the set of latent variables, and θ denotes the model parameters. Thus in the context of our model $Y = \{X, S\}$, $Z = \{Z\}$, and $\theta = \{\mu, \tau, \lambda, \pi\}$.

From inspection of the DGM, and taking note of the conditional independence properties encoded by it, we can see that the joint probability distribution of a single data point $n \in N$ is given by:

$$p(X_n, S_n, Z_n \mid \theta) = \prod_k p(Z_n) p(X_n \mid Z_n) p(S_n \mid Z_n)$$

Assuming that our data points are sampled independently, the probability of observing the entire dataset, or likelihood, is given by

$$p(X, S, Z \mid \theta) = \prod_n \prod_k p(Z) p(X \mid Z) p(S \mid Z)$$

$$p(X, S, Z \mid \theta) = \prod_n \prod_k \pi_k^{z_{nk}} (N(x_n \mid \mu_k, \tau_k^{-1}) P(s_n \mid \lambda_k))^{z_{nk}}$$

Where N and P denote the Normal and Poisson distribution respectively. The log-likelihood is thus given by

$$ln(p(X, S, Z \mid \theta)) = \sum_n \sum_k z_{nk} (ln(\pi_k) + ln(N(x_n \mid \mu_k, \tau_k^{-1})) + ln(P(s_n \mid \lambda_k))$$

In practice, a problem arises due to the fact that Z is not observed. This means that we cannot directly use $p(Z \mid Y, \theta^{old})$, but we instead use its expectation w.r.t its posterior distribution. Thus we introduce

$$\gamma(z_{nk}) = \mathbb{E}(z_{nk}) = \frac{\pi_k N(x_n \mid \mu_k, \tau_k^{-1}) P(s_n \mid \lambda_k)}{\sum_j \pi_j N(x_n \mid \mu_j, \tau_j^{-1}) P(s_n \mid \lambda_j)}$$

We then want to maximize

$$\mathbb{E}_{\mathbb{Z}}(ln(p(X, S, Z \mid \theta)) = \sum_n \sum_k \gamma(z_{nk}) (ln(\pi_k) + ln(N(x_n \mid \mu_k, \tau_k^{-1})) + ln(P(s_n \mid \lambda_k))$$

Comparing this expression with the one given for only a Gaussian mixture model we can see that the difference is given by the added $ln(P(\lambda_k)$ term. Thus, when we derive the expression with respect to μ and σ, this term will disappear. This means that we can use the same solutions for approximating μ_{new}, σ_{new} as given by equations 9.19 and 9.21 in [1]. An analogous argument can be made for the estimation of π.

Using the same method as in the book for maximizing λ:

$$\lambda_k^{new} = argmax_\lambda \sum_n \gamma(z_{nk}) (ln(\pi_k) + ln(N(x_n \mid \mu_k, \tau_k^{-1})) + ln(P(s_n \mid \lambda_k))$$

$$0 = \frac{\partial}{\partial \lambda_k} \sum_n \gamma(z_{nk}) (ln(\pi_k) + ln(N(x_n \mid \mu_k, \tau_k^{-1})) + ln(P(s_n \mid \lambda_k))$$

$$0 = \sum_n \gamma(z_{nk}) \frac{\partial}{\partial \lambda_k} (ln(\pi_k) + ln(N(x_n \mid \mu_k, \tau_k^{-1})) + ln(P(s_n \mid \lambda_k))$$

$$0 = \sum_n \gamma(z_{nk}) \frac{\partial}{\partial \lambda_k} ln(P(s_n \mid \lambda_k))$$

Note that we are able to drop the summation over k due to the fact that if we reverse order of the summation we note that all the k terms are independent equally expressed additions to a quantity we wish to maximize (save for the value of λ_k), and thus we want to maximize all of them independently.

The Poisson distribution is defined as

$$P(\lambda) := f(s_n) = \frac{\lambda^{s_n} e^{-\lambda}}{s_n!}$$

Therefore

$$ln(P(\lambda) = s_n ln(\lambda) - \lambda - ln(s_n!)$$

and

$$\frac{\partial}{\partial \lambda_k}(P(\lambda) = \frac{\partial}{\partial \lambda}(s_n ln(\lambda) - \lambda - ln(s_n!)$$

$$\frac{\partial}{\partial \lambda}(P(\lambda) = \frac{s_n}{\lambda} - 1$$

Plugging this into our derivative of the expected log-likelihood

$$0 = \sum_n \gamma(z_{nk})(\frac{s_n}{\lambda_k} - 1)$$

$$0 = \sum_n (\frac{\gamma(z_{nk}) s_n}{\lambda_k} - \gamma(z_{nk}))$$

$$\sum_n \gamma(z_{nk}) = \sum_n \frac{\gamma(z_{nk}) s_n}{\lambda_k}$$

$$\lambda_k = \frac{\sum_n \gamma(z_{nk}) s_n}{\sum_n \gamma(z_{nk})}$$

Observe that this equation is of the same form as the equation for updating mu, except that we use s_n instead of x_n.

References

[1] *Pattern Recognition and Machine Learning.* Springer, 2006.