Homework5: parameter learning with incomplete data

Zeming Fang
661958922

1. Introduction

In homework 4, we have extensively discussed parameter learning with complete data, using both MLE and MAP criterion. A challenge is, what if the data is not completed? Or some values are lost in the data? An intuitive idea is to fill the missing gap in the data and use the "complete" data to learn the parameters. The algorithm that formalizes this idea is *Expected-maximization* (EM) algorithm.

2. Expectation-maximization algorithm

2.1 General EM: formalization and solution

Given training data $D = \{D_1, D_2, ..., D_M\}$ of M i.i.d samples. For the $m\text{th}$ sample $D_m = X_m = \{Y_m, Z_m\}$, where Y_m is the observed variables with complete values and Z_m represent the latent variables with missing values. With these notations, a parameter learning problem can be defined as maximizing the MLE of the observed variables in a Bayesian Network(BN):

$$\Theta^* = \arg\max_{\Theta} \sum_{m=1}^{M} \log \sum_{Z_m} p(Y_m, Z_m|\Theta) \tag{1.1}$$

Here, I see the solution to this objective in an alternative optimization perspective.

$$
\begin{aligned}
&\max_{\Theta} J \\
&= \max_{\Theta} \sum_{m=1}^{M} \log \sum_{Z_m} p(Y_m, Z_m|\Theta) \\
&= \max_{\Theta,\Phi} \sum_{m=1}^{M} \log \sum_{Z_m} q(Z_m|Y_m,\Phi) \frac{p(Y_m, Z_m|\Theta)}{q(Z_m|Y_m,\Phi)} \\
&\geq \max_{\Theta,\Phi} \sum_{m=1}^{M} \sum_{Z_m} q(Z_m|Y_m,\Phi) \log \frac{p(Y_m, Z_m|\Theta)}{q(Z_m|Y_m,\Phi)} \\
&= \max_{\Theta,\Phi} \sum_{m=1}^{M} \sum_{Z_m} q(Z_m|Y_m,\Phi) \log p(Y_m, Z_m|\Theta) \\
&\qquad - q(Z_m|Y_m,\Phi) \log p(Z_m|Y_m|\Phi)
\end{aligned}
\tag{1.2}
$$

We can see this J as an optimization problem of each parameter, alternatively.

- Step 1: $\max_{\Theta} J$

$$
\max_{\Theta} \sum_{m=1}^{M} \sum_{Z_m} q(Z_m|Y_m,\Phi^{t-1}) \log p(Y_m, Z_m|\Theta) \tag{1.3}
$$

- Step 2: $\max_{\Phi} J$

$$
\max_{\Phi} -KL(q(Z_m|Y_m,\Phi)||p(Y_m, Z_m|\Theta^t) \tag{1.4}
$$

In the EM algorithm, both E-step and M-step are discussing the solution to step 1. Before discussing EM, I would like to add a brief discussion about step2. The closed-form solution of step2 is:

$$
\begin{aligned}
q(Z_m|Y_m,\Phi) &\propto p(Y_m, Z_m|\Theta^t) \\
\Rightarrow q(Z_m|Y_m,\Phi) &\propto p(Z_m|Y_m,\Theta^t) p(Y_m,\Theta^t) \\
\Rightarrow q(Z_m|Y_m,\Phi) &= p(Z_m|Y_m,\Theta^t) \\
\Rightarrow \Phi &= \Theta^t
\end{aligned}
\tag{1.5}
$$

Back to step 1, its solution is the M-step in EM algorithm:

$$
\begin{aligned}
&\max_{\Theta} \sum_{m=1}^{M} \sum_{Z_m} q(Z_m|Y_m, \Phi^{t-1}) \log p(Y_m, Z_m|\Theta) \\
&= \max_{\Theta} \sum_{m=1}^{M} \sum_{z_1^m, z_2^m, \ldots, z_N^m} \prod_{i=1}^{N} q(z_i^m|\pi(z_i^m), \Phi^{t-1}) \log \prod_{i=1}^{N} p(Y_m, z_i^m|\Theta) \\
&= \max_{\Theta} \sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{z_i^m} q(z_i^m|\pi(z_i^m), \Phi^{t-1}) \log p(Y_m, z_i^m|\Theta) \qquad (1.6) \\
&= \max_{\Theta} \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{z_i^m} q(z_i^m|\pi(z_i^m), \Phi^{t-1}) \log p(Y_m, z_i^m|\Theta) \\
&= \max_{\Theta} \sum_{i=1}^{N} \sum_{j=1}^{J} \sum_{m=1}^{M} \sum_{z_i^m} q(z_i^m|\pi(z_i^m) = j, \Phi^{t-1}) \log p(z_i^m|\pi(z_i^m) = j, \Theta)
\end{aligned}
$$

The solution to this equation is:

$$\theta_{njk}^{t+1} = \frac{N^{ijk}}{\sum_k N_{ijk}}$$

Where $N_{ijk} = \sum_{m=1}^{W} q(z_i^m = c|\pi(z^m) = j, \theta^{t-1})$ for unobserved data.

The convergence speed of EM heavily relies on the initialization distribution. If the initialized distribution happens to be close to the optimal distribution, the EM algorithm may converge within a few iterations. Otherwise, it may take a much longer time.

2.2 Hard EM

In the general EM algorithm, we infer the data according to the proposal distribution. To be more specific, we transversed all possible states for the current node and meanwhile assigned a weight to this sample. In hard EM, the algorithm fills the missing data with the most probable explanation instead of enumerating all possible states. Thus, the inferred list in the hard EM will be much smaller than that in the general EM but biased.

In terms of implementation, the only difference between hard-EM and EM locates at the E-step. Different from equation 1.4, the step1 in hard-EM is:

$$\max_{\Theta} \sum_{m=1}^{M} \log p(Y_m, \hat{Z}_m|\Theta) \tag{1.6}$$

Where $\hat{Z}_m = \arg\max_{z_m} p(z_m|Y_m, \Theta^{t-1})$.

Theoretically, hard-EM is a lower bound of the general EM but a tight one.

3. Experiment setting

In this project, we are instructed to learn the parameter of a BN when given structure (see figure1 for the structure) but from incomplete data.

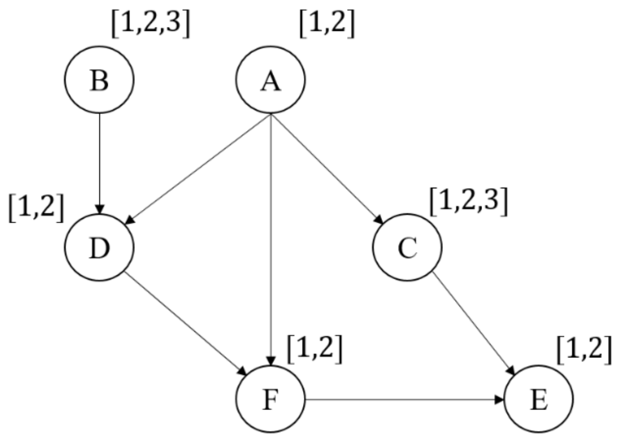

Figure1 (Adopted and modified from HW4 instruction) known BN structure

To allow learning, we need to infer the missing data. Thus, we were instructed to use the EM algorithm for learning. However, the performance of the EM algorithm is sensitive to its hyperparameter, including its data generation method and its initialization distribution. In this project, we briefly discuss the impact of the choices of these hyperparameters.

To be more specific, we:

- Compared the performance of different data generation methods: EM method, which uses sum-product inference to generate data; hard-EM method, which

uses MPE inference to generate data.
- Compared the performance of different initialization methods: uniform initialization, random initialization, and use the distribution learned from complete data as initialization of EM.

4. Results and discussions

4.1 Learned CPT for EM and hard-EM using uniform initilaization

Table 1 shows the learned CPT for this experimental setting using the EM algorithm with uniform initialization. This algorithm converged at the seventh iteration with tolerance as $\lambda = 1e - 3$.

Table1 learned CPT using hard-EM with uniform initialization

Node			**CPT**	
Node A				
		A=1	A=2	
	prior	0.3748	0.6252	
Node B				
		B=1	B=2	b=3
	prior	0.2819	0.4311	0.2870
Node C				
		C=1	C=2	C=3
	A=1	0.3365	0.2069	0.4567
	A=2	0.0583	0.1887	0.7531
Node D				
		D=1	D=2	
	A=1,B=1	0.0604	0.9397	
	A=2,B=1	0.1903	0.8097	
	A=1,B=2	0.8811	0.1189	
	A=2,B=2	0.4468	0.5532	

	A=1,B=3	0.4704	0.5296	
	A=2,B=3	0.2831	0.7169	
Node E				
		E=1	E=2	
	C=1,F=1	0.0031	0.9968	
	C=2,F=1	0.2889	0.7111	
	C=3,F=1	0.1413	0.8587	
	C=1,F=2	0.3145	0.6855	
	C=2,F=2	0.9001	0.0999	
	C=3,F=2	0.6350	0.3650	
Node F				
		F=1	F=2	
	A=1,D=1	0.9483	0.0517	
	A=2,D=1	0.3781	0.6219	
	A=1,D=2	0.1104	0.8896	
	A=2,D=2	0.3106	0.6894	

Table2 listed the learned CPT using hard-EM with uniform initialization. The algorithm converges at 4th iteration with tolerance $\lambda = 1e - 3$.

Table2 learned CPT using EM with uniform initialization

Node			**CPT**	
Node A				
		A=1	A=2	
	prior	0.366	0.634	
Node B				
		B=1	B=2	b=3
	prior	0.324	0.424	0.252
Node C				

	C=1	C=2	C=3
A=1	0.3224	0.1913	0.4863
A=2	0.0473	0.1640	0.7886

Node D

	D=1	D=2
A=1,B=1	0.0377	0.9623
A=2,B=1	0.1284	0.8716
A=1,B=2	0.9213	0.0787
A=2,B=2	0.4553	0.5447
A=1,B=3	0.4878	0.5122
A=2,B=3	0.2588	0.7412

Node E

	E=1	E=2
C=1,F=1	0.0000	1.0000
C=2,F=1	0.2381	0.7619
C=3,F=1	0.1129	0.8871
C=1,F=2	0.2581	0.7419
C=2,F=2	0.9333	0.0667
C=3,F=2	0.7070	0.2930

Node F

	F=1	F=2
A=1,D=1	0.9712	0.0288
A=2,D=1	0.3696	0.6304
A=1,D=2	0.0886	0.9114
A=2,D=2	0.2978	0.7022

Though close, these two CPTs were not the same. Intuitively, the EM method should be closer to the ground truth distribution because the CPT learned by hard EM has some probability as 0, an almost impossible event in PGM learning. (If the ground truth CPT in HW4 is also the ground truth in this homework, we can claim that CPT learned by EM is better than that learned by hard-EM quantitatively.

4.2 Convergence performance of EM and hard-EM.

Figure2 showed the different performance between EM and hard-EM. As we mentioned in section 4.1, when the tolerance is $1e-3$, the EM converged at the 7th iteration and the hard-EM converged at the 4th iteration. Despite sounding like the EM is much worse than the hard-EM, Figure2 showed that their convergence performances are alike. In fact, the convergence curve of EM is steeper than that of hard-EM at the early stage of learning.

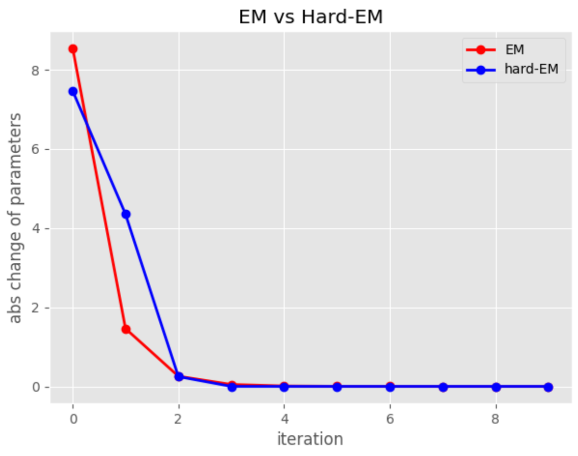

Figure2 EM versus hard-EM with uniform distribution

However, in terms of computation complexity, hard-EM is much lower than EM due to much fewer inferred data.

4.3 EM with different initialization method.

As we mentioned in the class, the EM algorithm's performance is sensitive to the initialization method. In this section, apart from the uniform distribution, we proposed two simple strategies: adding small perturbations to uniform distribution and using the complete data distribution.

In general, the uniform distribution is the maximum entropy distribution that is close to every possible distribution. However, in the complex nonlinear condition, we wish to have a better start point. Though not reliable, adding small perturbation may bring the algorithm to the neighbor of a local minimum, accelerating learning. The *random1* and *random2* were two experiments using the perturbating method. We found that even for the better one, it was only slightly better than the uniform initialization. The worse one is worse than the uniform benchmark.

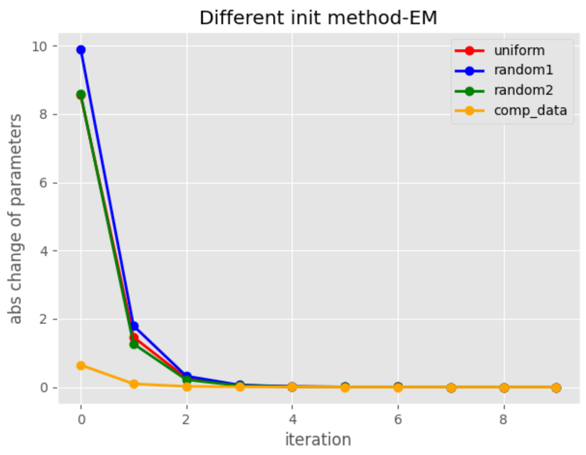

Figure3 EM with different initialization methods distribution

A more efficient method is first to learn a CPT from the complete data and use the learned CPT as the initialization for the EM. This method works well because the learned CPT from completed data is part of our objective CPT. Thus, the cross-entropy between this initialization and the target objective is much lower than that of uniform initialization. We can see in Figure3 that the EM algorithm with this initialization was much better than other methods. It converged at the second iteration.

I also implemented these two initialization on hard-EM algorithm, see figure4. Similarly, we may drew the conclusion that using the complete data distribution as the initialization is best initialization methods against the other two alternatives.

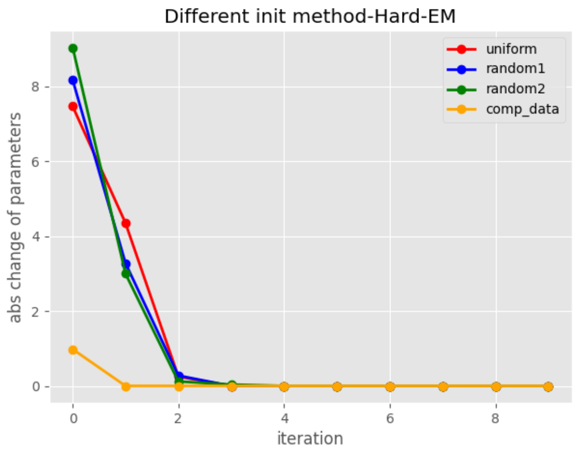

Figure4 Hard-EM with different initialization methods distribution

5. Conclusions

In this homework, we implemented the both EM and hard-EM algorithms to learn a toy PGM with incomplete data. We showed that both algorithms have good convergence properties. We also proposed an initialization method: learn a distribution from the complete data and use it as the initialization for the EM algorithm. We showed this initialization that greatly accelerates EM learning.