# Lecture 19, 20: Learning structure with incomplete data

- Two problems: missing values and missing (hidden) variables
- Scoring structures with missing values
- Structural EM
- Dealing with hidden variables



## Two distinct problems

1. You know all the variables, but some values are missing in some instances, e.g.,

| X | Y | Z |
|---|---|---|
| 0 | 1 | 1 |
| 1 | ? | 0 |
| 0 | ? | ? |
| ... | ... | ... |

This makes the search problem a lot harder, but still doable

2. There are hidden (latent) variables which you *never* observe, e.g.

| X | Y | Z | H |
|---|---|---|---|
| 0 | 1 | 1 | |
| 1 | 1 | 0 | |
| 0 | 0 | 0 | |
| ... | ... | ... | |

## Scoring structure using MDL

- Recall that for a graph $G$, the MDL score has the form:

$$\text{score}_{\text{MDL}} = m \sum_i MI_{\hat{p}}(X_i, \text{Parents}(X_i)) - \text{Penalty}(G)$$

- To get $\hat{p}$, we need to compute the parameters of the graph $G$, from our incomplete data
- Simple idea: use gradient descent or EM to compute (as best we can) max. likelihood parameters given the data
- The penalty term depends on the size of the graph, *not* the parameters, so it will not be affected.



## A simple algorithm

1. Start with a graph structure $G$
2. Repeat as long as desired:
   (a) Consider all graphs $G'$ that can be obtained by adding or deleting an arc from $G$ (these are $G$'s *successors*)
   (b) For each structure $G'$, run EM (or gradient ascent) to fit its parameters.
   (c) Compute $\text{score}_{\text{MDL}}(G')$ for each $G'$
   (d) Pick a $G'$ out of the candidates using your favorite method (e.g., greedily or using simulated annealing)
   (e) $G \leftarrow G'$

## The simple algorithm is too slow!

- If we have $n$ random variables, *in each search step* there are $n^2$ possible successors for $G$ (we can pick any pair of variables and add an arc, if none is there, or remove an arc, if they are connected
- Of course, this is a worst-case estimate, because some of the resulting structures may be illegal
- Finding the parameters of the network requires some number of EM iterations
- Then to compute the score, we need to compute the likelihood of the data, which is basically a step of inference
- We need a better idea!

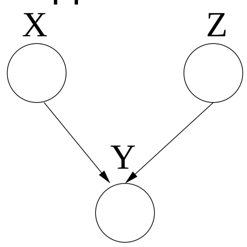


## Structural EM (Friedman, 1997)

- Recall the interpretation of the EM algorithm in parameter estimation
  - Start with a guess for the parameters
  - *Complete the data* by assigning the most likely values to the missing variables.
  - Improve the parameter guess based on the completed data, and iterate
- So let's *use our current network* $G$ to complete the data!
- In our previous algorithm, we completed the data separately using each successor $G'$
- But *$G$ and $G'$ differ only by one arc!*
- So using $G$ to complete the data cannot be too bad...

## Example

- Suppose we have the following data, and $G$ is a v-structure




| X | Y | Z |
|---|---|---|
| 0 | 1 | 1 |
| 1 | ? | 0 |
| 0 | 1 | ? |
| ... | ... | ... |

- We need to complete the value of $Y$ in instance 2, $Y^2$, and the value of $Z$ in instance 3, $Z^3$
- To complete $Y^2$, we need to compute $p(Y = 1|X = 1, Z = 0, \langle G, \theta\rangle)$ ($p(Y = 0|X = 1, Z = 0, \langle G, \theta\rangle)$ can be obtained given that probabilities must sum to 1)

  This requires *inference*!
- Likewise, for $Z^3$, we need to compute $p(Z|X = 0, Y = 1, \langle G, \theta\rangle)$



## Example: Two versions of the algorithm

- Hard EM: pick the *most likely values* for $Y^2$ and $Z^3$, then install them and use the resulting data set to score the successors $G'$
- Soft EM
  - Consider *all possible* assignments of values for $Y^2$ and $Z^3$, which gives us *several completed data sets*
  - The score for the successors $G'$ is obtained as an *expected value*, by averaging the scores obtained from each data set

## Example: Soft EM

- Consider all possible combinations of values for $Y^2$ and $Z^3$: $\langle Y^2 = 0, Z^3 = 0\rangle$, $\langle Y^2 = 1, Z^3 = 0\rangle$, ...
- This gives us 4 data sets, call them $D_{00}$, $D_{01}$, $D_{10}$, $D_{11}$
- Because the data is i.i.d., the likelihood of each data set is: $p(D_{ij}) = p(Y = i|X = 1, Z = 0, \langle G, \theta\rangle)p(Z = j|X = 0, Y = 1, \langle G, \theta\rangle)$
- For every $G'$, evaluate 4 scores, score$_{ij}(G')$, *one corresponding to each completion* of the data, $D_{ij}$

$$\text{score}_{MDL}(G') \approx \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} p(D_{ij})\text{score}_{ij}(G')$$

- Note that the number of data sets created in the “soft” version is *exponential* in the number of missing values



## Making the algorithm more efficient

- Recall from lecture 16 that the likelihood of the (complete) data can be *decomposed* based on the network structure
- Likewise, the MDL score can be computed by looking at the mutual information of a node and it parents, which *can be computed locally* at each node, using counts
- So we keep sufficient statistics (counts) at each node
- The fact that there are missing values only means we need to keep alternate counts *at the nodes* for which values are missing.
- When going from $G$ to a successor $G'$, we recompute the score *only for the families that are affected*
- Every $k$th search step, we have to do EM again to compute a new completion of the data set

## Theoretical properties

- For any two graphs $G_1$ and $G_2$, we have:

$$\text{score}_{MDL}(G_2) - \text{score}_{MDL}(G_1)$$
$$\geq E[\text{score}_{MDL}(G_2)|\text{completed data}] - E[\text{score}_{MDL}(G_1)|\text{completed data}]$$

- So if SEM moves from graph $G_1$ to a graph $G_2$ that seems to have a better expected MDL score (according to the possible data completions), then the true MDL score of $G_2$ is also better than the true MDL score of $G_1$
- The difference between the two MDL scores is at least as big as predicted by SEM
- Hence, the score is guaranteed to converge to a local maximum
- Of course, like in regular EM, multiple restarts will help get a better network in the end.



## What about Bayesian scoring?

- Recall the Bayesian score:

$$\text{score}_{\text{Bayes}} = p(G|D) = p(G)p(D|G) = p(G)\int p(D|G,\theta)p(\theta|G)d\theta$$

- We have to evaluate the integral for all graphs $G$!
- Evaluating the integral can also be quite expensive!
  - We can pick a few graphs that are most likely, and evaluate it only for those
  - Alternatively, use stochastic integration, but it turns nasty...

## Computational hardship

- The computation of parameters for every candidate is very expensive
- We cannot tell beforehand whether it’s really worth doing it (how good will a candidate be?)
- Works only if we limit the search space to a small number of networks



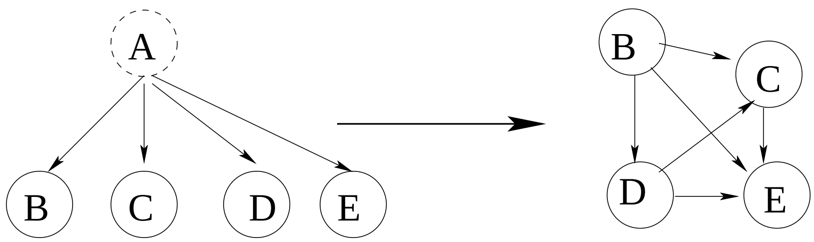
## Dealing with hidden variables

This is much harder!

- How can we tell there is something hidden?
- How many hidden variables should be introduce?
- How should they link to the rest of the network?

## How can we tell there is something hidden?

- We are doing structure learning and all networks have very low score
- Based on prior knowledge of the domain, the obtained structures do not make sense
- There are big cliques of nodes that are strongly connected
- Example: consider what happens below if we consider removing node A:



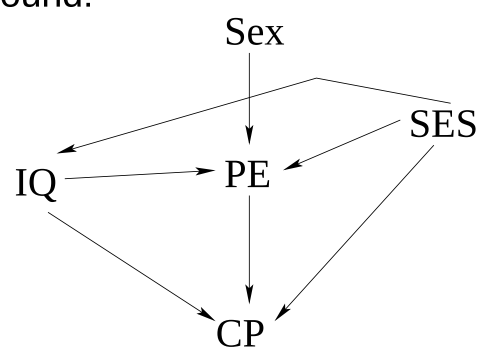



## How do we get the structure with hidden variables?

- How many hidden variables should be introduce?
  - As few as possible! Most applications introduce at most one....
- How should they link to the rest of the network?
  - Make a guess for an the structure (e.g. by looking at large cliques or strongly connected subsets of nodes)
  - Then use EM to estimate parameters!

## Hidden variables: Case study (Heckerman)

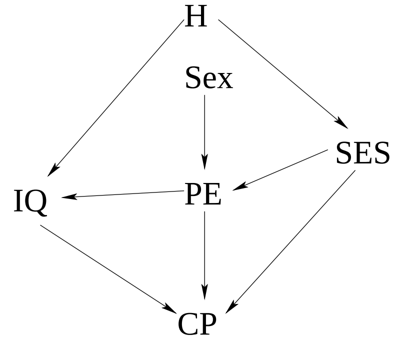
- Complete data from over 10000 Wisconsin high school graduates: sex (2 values), socio-economic status (4 values), IQ (4 values), parental encouragement (2 values), college plans (2 values)
- Goal is to find causal relationships between the variables
- Best structure found:






## Hidden variables: Case study (2)

- They considered adding 1-4 hidden variables, each with between 2-6 possible values.
- Best structure has one hidden variable, $H$,with two possible values




- This is $2 \cdot 10^{10}$ more likely than the previous best!
- In general, bushy networks are an indication of potential hidden variables