Structure Learning in Bayesian Networks (mostly Chow-Liu)

Sue Ann Hong
11/15/2007

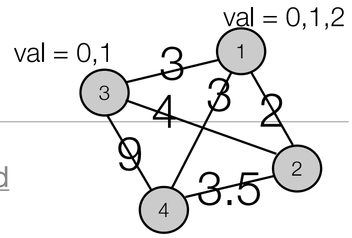
Chow-Liu

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
- Give directions to edges in MST

Chow-Liu: how-to



- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)

$$I(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

$$\hat{P}(x_i, x_j) = \frac{Count(x_i, x_j)}{m}$$

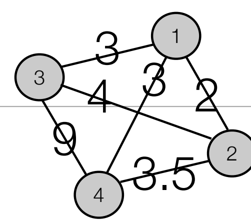
"empirical distribution" # examples

- e.g. (1) & (3)

$$I(X_1, X_3) = \sum_{x_1=0}^{1} \sum_{x_2=0}^{2} \hat{P}(X_1 = x_1, X_2 = x_2) \log \frac{\hat{P}(X_1 = x_1, X_2 = x_2)}{\hat{P}(X_1 = x_1)\hat{P}(X_2 = x_2)}$$

e.g. $\hat{P}(X_1 = 0, X_2 = 1) = \frac{Count(X_1 = 0, X_2 = 1)}{m}$

Chow-Liu: how-to



- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

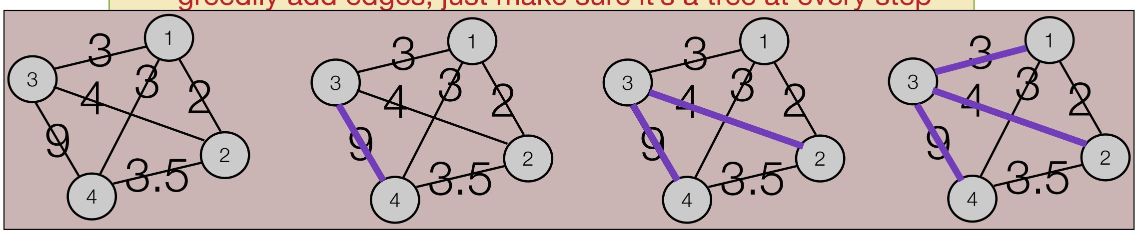
- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST) ← must reach all nodes
  - tree with the greatest total weight $\sum_{(X_i, X_j) \in E} I(X_i, X_j)$
  - greedily add edges, just make sure it's a tree at every step
  - e.g. Kruskal, Prim

Chow-Liu: how-to

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
  - tree with the greatest total weight $\sum_{(X_i,X_j)\in E} I(X_i,X_j)$
  - greedily add edges, just make sure it's a tree at every step

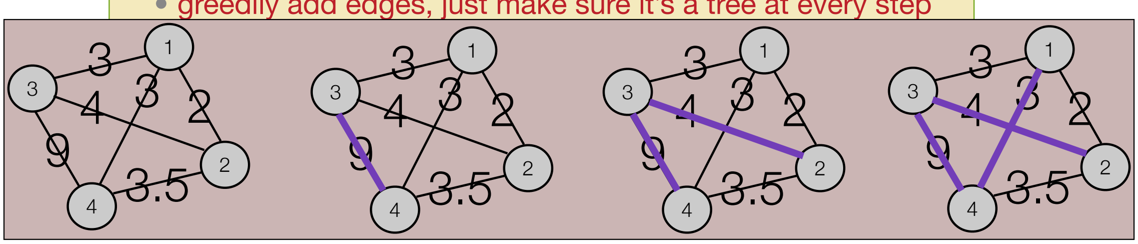


Chow-Liu: how-to

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
  - tree with the greatest total weight $\sum_{(X_i,X_j)\in E} I(X_i,X_j)$
  - greedily add edges, just make sure it's a tree at every step



Chow-Liu: how-to

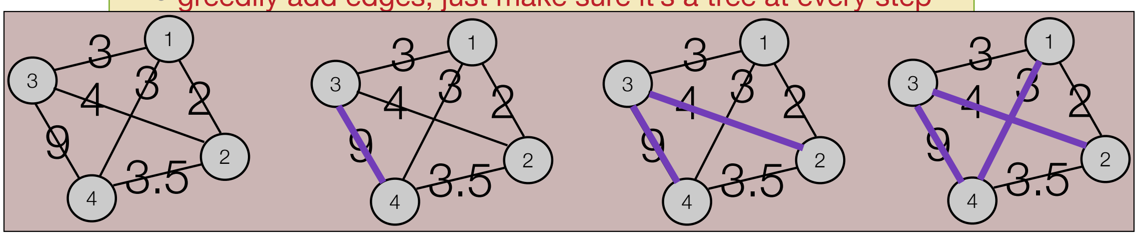
what if there were negative edges?
the algorithm still works
but do you want them?

hint hint

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
  - tree with the greatest total weight $\sum_{(X_i,X_j) \in E} I(X_i, X_j)$
  - greedily add edges, just make sure it's a tree at every step

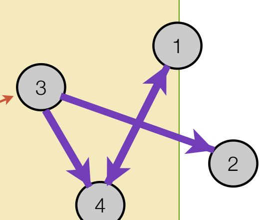


Chow-Liu: how-to

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
- Give directions to edges in MST
  - pick your favorite node (e.g. sinus??)
  - draw arrows going away from it (e.g. BFS, DFS)



Chow-Liu: why it works

- Goal: find a **tree** that **maximizes** the data likelihood

Algorithm

- Compute weight I(Xi,Xj) of each (possible) edge (Xi,Xj)
- Find a **maximum** weight spanning tree (MST)
- Give directions to edges in MST

Just two questions:
1. why can we represent data likelihood as sum of I(Xi,Xj) over edges?
2. why can we pick any direction for edges in the tree?*

*as long as it's a tree

1. why can we represent data likelihood as sum of I(Xi,Xj) over edges?

2. why can we pick any direction for edges in the tree?

- data likelihood given (directed) edges

$$logP(D|G,\theta_G) = \sum_{j=1}^{m} \sum_{i=1}^{n} logP(x_i|pa_{X_i})$$

- information theoretic quantity

$$logP(D|G,\theta_G) = m \sum_{i=1}^{n} (I(X_i, Pa_{X_i}) - H(X_i))$$

- max only part that matters

$$\text{argmax}_G logP(D|G,\theta_G) = \text{argmax}_G \sum_{i=1}^{n} I(X_i, Pa_{X_i})$$

- tree! (Pa_Xi = just one other node) => I(Xi,Pa_Xi) = I(Xi,Xj)

$$\text{argmax}_G logP(D|G,\theta_G) = \text{argmax}_G \sum_{(X_i,X_j)\in E} I(X_i, X_j)$$

  - directed edges? nah. I(Xi,Xj) = I(Xj,Xi)

$$I(X_i, X_j) = \sum_{x_i,x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i,x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

1. why can we represent data likelihood as sum of I(Xi,Xj) over edges?

2. why can we pick any direction for edges in the tree?

- data likelihood given (directed) edges

$$logP(D|G,\theta_G) = \sum_{j=1}^{m}\sum_{i=1}^{n} logP(x_i|pa_{X_i})$$

- information theoretic quantity

$$logP(D|G,\theta_G) = m\sum_{i=1}^{n}(I(X_i,Pa_{X_i}) - H(X_i))$$

- max only part that matters

$$\text{argmax}_G logP(D|G,\theta_G) = \text{argmax}_G \sum_{i=1}^{n} I(X_i,Pa_{X_i})$$

- tree! (Pa_Xi = just one other node) => I(Xi,Pa_Xi) = I(Xi,Xj)

$$\text{argmax}_G logP(D|G,\theta_G) = \text{argmax}_G \sum_{(X_i,X_j)\in E} I(X_i,X_j)$$

  - directed edges? nah. I(Xi,Xj) = I(Xj,Xi)

$$I(X_i,X_j) = \sum_{x_i,x_j} \hat{P}(x_i,x_j)\log\frac{\hat{P}(x_i,x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$$

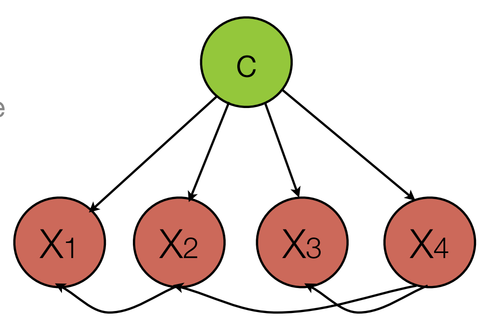
  - so directions don't matter

    - **as long as no v-structures**

break

TAN::Tree-Augmented Naive Bayes

- NB + Chow-Liu
  - Same old Chow-Liu on features, but with I(Xi,Xj|c) instead of I(Xi,Xj)
  - Then learn P(Xi | Pa(Xi), c) as before
- **Remember** this algorithm for the future



the usual difficulties

- In general, NP-hard to learn structure with #parents > 1

try e.g.

- BIC score: approximation of Bayesian score → "regularization"
  → maximizing still NP-hard

- Trees - "easy" to learn:
  - one parent - no v-structures
    - can do this greedy search with completely uncoupled scores

Announcing

- no recitation next week - happy thanksgiving!
  - better sleep...!