STRUCTURE LEARNING FOR BAYESIAN NETWORK

★ Score - Based Approach :-

→ Define a criterion to evaluate how well the bayesian network fits the data

→ Searches over the space of DAGs for a structure achieving the maximal score.

* Score Metrics :

$$\text{Score}(G:D) = LL(G:D) - \phi(|D|)\,||G||$$

Regularization term

G - Graph , D - Data

LL - log likelihood

$|D|$ - # datapoints

$||G||$ - # parameters

$\phi(t) = 1$ ← Akaike Information Criterion

$\phi(t) = \log(t)/2$ ← Bayesian Information Criterion

Bayesian Dirichlet score = $\log P(D|\theta_G) + \log P(\theta_G)$

$$P(D|G) = \int P(D|G,\theta_G) \cdot P(\theta_G|G) \cdot d\theta_G$$

↓ prob. of the data given the network structure & params

↓ prior prob of the params

$$P(D|\theta_G) = \prod_i \prod_{\pi_i} \left[\frac{\Gamma(\sum_j N'_{i,\pi_i,j})}{\Gamma(\sum_j N'_{i,\pi_i,j} + N_{i,\pi_i,j})} \prod_j \frac{\Gamma(N'_{i,\pi_i,j} + N_{i,\pi_i,j})}{\Gamma(N'_{i,\pi_i,j})} \right]$$

π_i - parent configuration of var i

$N_{i,\pi_i,j}$ - count of var i taking value j with parent config π_i

N' - counts in prior

* Chow-Liu Algorithm :

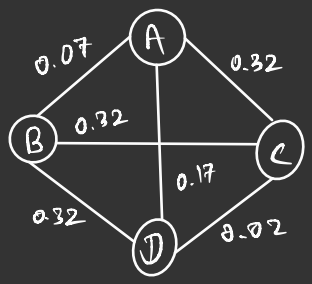
→ Score based approach which finds the maximum-likelihood tree-structured graph.

step-1 :

- compute Mutual Information for all pairs of vars and form a complete graph

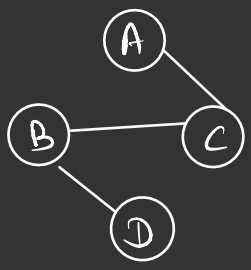
$$MI(X,U) = \sum_{x,u} \hat{P}(x,u) \log \left[\frac{\hat{P}(x,u)}{\hat{P}(x) \cdot \hat{P}(u)} \right]$$

$$\hat{P}(x,u) = \frac{\text{count}(x,u)}{\#\text{data points}}$$



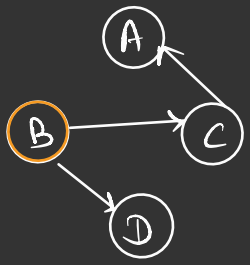
Step-2:

- Find the maximum weight spanning tree. using Kruskal or Prim algorithms.



Step-3:

- Pick any node to be the root variable and assign directions radiating outward from this node.



Proof:

$$P_t(x) = \prod_{i=1}^{n} P(x_i \mid x_{\pi(i)})$$

$$KL(P, P_t) = \sum_x P(x) \log\left(\frac{P(x)}{P_t(x)}\right)$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1, \neq root}^{n} \log P(x_i \mid x_{\pi(i)})$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1, \neq root}^{n} \log \frac{P(x_i, x_{\pi(i)})}{P(x_{\pi(i)})}$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \sum_{i=1, \neq n}^{n} \log \frac{P(x_i, x_{\pi(i)})}{P(x_i)\,P(x_{\pi(i)})} - \sum_x P(x) \sum_i \log P(x_i)$$

$$\sum_x P(x) \sum_i \log \frac{P(x_i, x_{\pi(i)})}{P(x_i)\,P(x_{\pi(i)})} = \sum_{x_i, x_{\pi_i}} P(x_i, x_{\pi_i}) \log \frac{P(x_i, x_{\pi_i})}{P(x_i, x_{\pi_i})} = I(x_i, x_{\pi_i})$$

$$KL(P, P_t) = -\sum_{i=1}^{n} I(x_i, x_{\pi_i}) + \sum_{i=1}^{n} H(x_i) - H(x)$$

↓ Mutual Information > 0

Independent of the dependence tree

So, Minimizing $I(P, P_t)$ is the same as maximizing the total branch weight: $\sum_{i=1}^{n} I(x_i, x_{\Pi_i})$

* Search Algorithms :

→ Local Search Algorithm :
- starts with an empty graph or a complete graph
- At each step, attempt to change the graph structure by adding / removing / reversing an edge.
- If the score increases, then it adopts the change

→ Greedy search (K3 Algorithm) :
- Assume a topological order of the graph
- For each variable, we restrict its parent set to the variables with a higher order.
- While searching for parent set for each variable, it takes a greedy approach by adding the parent that increases the score most until no improvement can be made.

→ The graph space is highly non-convex and both algorithms might get stuck at some sub-optimal regions.

✱ Constraint Based Approach :

→ Employs the independence test to identify a set of edge constraints for the graph and then finds the best DAG that satisfies the constraints.

→ Works well with some prior information of structure but requires lot of data samples.

→ Order Search Approach :

- Conducts a search over topological orders & graph space
- Swaps the order of two adjacent variables from topological order at each step and employs the K3 algorithm as a sub-routine.

→ Integer Linear Programming Approach :

- Encodes the graph structure, scoring and the acyclic constraints into a linear programming problem.
- Requires a bound on the maximum number of parents any node in the graph can have.