

Structure learning for bayesian networks

Structure learning for Bayesian networks

The task of structure learning for Bayesian networks refers to learn the structure of the directed acyclic graph (DAG) from data. There are two major approaches for the structure learning: score-based approach and constraint-based approach .

Score-based approach

The score-based approach first defines a criterion to evaluate how well the Bayesian network fits the data, then searches over the space of DAGs for a structure with maximal score. In this way, the score-based approach is essentially a search problem and consists of two parts: the definition of score metric and the search algorithm.

Score metrics

The score metrics for a structure \mathcal{G} and data D can be generally defined as:

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

Here $LL(\mathcal{G} : D)$ refers to the log-likelihood of the data under the graph structure \mathcal{G} . The parameters in Bayesian network \mathcal{G} are estimated based on MLE and the log-likelihood score is calculated based on the estimated parameters. If we consider only the log-likelihood in the score function, we will end up with an overfitting structure (namely, a complete graph.) That is why we have the second term in the scoring function. $|D|$ is the number of sample and $\|\mathcal{G}\|$ is the number of parameters in the graph \mathcal{G} . With this extra term, we will penalize the over-complicated graph structure and avoid overfitting. For AIC the function

$\phi(t) = 1$, while for BIC $\phi(t) = \log(t)/2$. It is important to note that in BIC, the influence of model complexity will decrease as M grows, allowing the log-likelihood term to eventually dominate the score.

There is another family of Bayesian score function called BD (Bayesian Dirichlet) score. For BD score, it first defines the probability of data D conditional on the graph structure \mathcal{G} as

$$P(D|\mathcal{G}) = \int P(D|\mathcal{G}, \Theta_{\mathcal{G}})P(\Theta_{\mathcal{G}}|\mathcal{G})d\Theta_{\mathcal{G}},$$

where $P(D | \mathcal{G}, \Theta_{\mathcal{G}})$ is the probability of the data given the network structure and parameters, and $P(\Theta_{\mathcal{G}} | \mathcal{G})$ is the prior probability of the parameters. When the prior probability is specified as a Dirichlet distribution,

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

Here π_i refers to the parent configuration of the variable i and $N_{i,\pi_i,j}$ is the count of variable i taking value j with parent configuration π_i . N' represents the counts in the prior respectively.

With a prior for the graph structure $P(\Theta_{\mathcal{G}})$ (say, a uniform one), the BD score is defined as

$$\log P(D|\Theta_{\mathcal{G}}) + \log P(\Theta_{\mathcal{G}}).$$

Notice there is no penalty term appending to the BD score due to that it will penalize the overfitting implicitly via the integral over parameter space.

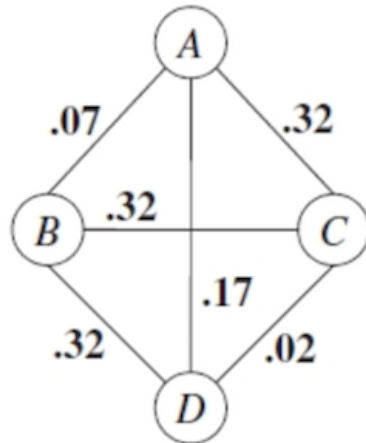
Chow-Liu Algorithm

The Chow-Liu Algorithm is a specific type of score based approach. The Chow-Liu algorithm finds the maximum-likelihood tree structure where each node has at most one parent. Note that here our score is simply the maximum likelihood, we do not need to penalize the complexity since we are already limiting complexity by restricting ourselves to tree structures.

The algorithm has three steps:

1) Compute the mutual information for all pairs of variables X, U , and form the mutual information graph where the edge between variables X, U has weight $MI(X, U)$:

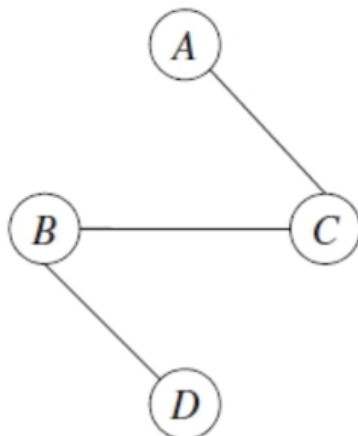
$$MI(X, U) = \sum_{x, u} \hat{p}(x, u) \log \left[\frac{\hat{p}(x, u)}{\hat{p}(x)\hat{p}(u)} \right]$$



Remember that from our empirical distribution

$$\hat{p}(x, u) = \frac{\text{Count}(x, u)}{\# \text{ data points}}.$$

2) Find the **maximum** weight spanning tree: the maximal-weight tree that connects all vertices in a graph. This can be found using Kruskal or Prim Algorithms.



3) Pick any node, and assign directions radiating outward from node (arrows go away from it).



The Chow-Liu Algorithm has a complexity of order n^2 , as it takes $O(n^2)$ to compute mutual information for all pairs, and $O(n^2)$ to compute the maximum spanning tree.

Now that we have described the algorithm, let's explain why this works. It turns out that the likelihood score decomposes into mutual information and entropy terms:

$$\log p(\mathcal{D} \mid \theta^{ML}, G) = |\mathcal{D}| \sum_i MI_{\hat{p}}(X_i, X_{pa(i)}) - |\mathcal{D}| \sum_i H_{\hat{p}}(X_i)$$

We would like to find a graph G that maximizes this log-likelihood. Since the entropies are independent of the dependency ordering in the tree, the only terms that change with choice of G are the mutual information terms. So we want

$$\arg \max_G \log P(\mathcal{D} \mid \theta^{ML}(G), G) = \arg \max_G \sum_i MI(X_i, X_{pa(i)})$$

Now if we assume $G = (V, E)$ is a tree where each node has at most one parent, we get

$$\arg \max_{G: G \text{ is tree}} \log P(\mathcal{D} \mid \theta^{ML}(G), G) = \arg \max_{G: G \text{ is tree}} \sum_{(i,j) \in E} MI(X_i, X_j)$$

Note that the orientations of edges do not matter because mutual information is symmetric. Thus we can see why the Chow-Liu algorithm finds the best approximate tree structure where nodes are restricted to have at most one parent.

Search algorithms

The most common choice for search algorithms are local search and greedy search.

For local search algorithm, it starts with an empty graph or a complete graph. At each step, it attempts to change the graph structure by a single operation of adding an edge, removing an edge or reversing an edge. (Of course, the operation should preserve the acyclic property.) If the score increases, then it adopts the attempt and does the change, otherwise it makes another attempt.

For greedy search (namely the K3 algorithm), we first 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.

A former CS228 student has created an interactive web simulation for visualizing the K3 learning algorithm. Feel free to play around with it and, if you do, please submit any feedback or bugs through the Feedback button on the web app.

Although both approach are computational tractable, neither of them have a guarantee of the quality of the graph that we end up with. The graph space is highly “non-convex” and both algorithm might get stuck at some sub-optimal regions.

Constraint-based approach

The constraint-based case employs the independence test to identify a set of edge constraints for the graph and then finds the best DAG that satisfies the constraints. For example, we could distinguish V-structure and fork-structure by doing an independence test for the two variables on the sides conditional on the variable in the middle. This approach works well with some other prior (expert) knowledge of structure but requires lots of data samples to guarantee testing power. So it is less reliable when the number of sample is small.

Recent Advances

In this section, we will briefly introduce two recent algorithms for graph search: order-search (OS) approach and integer linear programming (ILP) approach.

The OS approach, as the name refers, conducts a search over the topological orders and the search over graph space at the same time. The K3 algorithm assumes a topological order in advance and do the search only over the graphs that obey the topological order. When the order specified is a poor one, it may end with a bad graph structure (with a low graph score). The OS algorithm resolves this problem by doing search over orders at the same time. It shifts two adjacent variable in an order at each step and employs the K3 algorithm as a sub-routine.

The ILP approach encodes the graph structure, scoring and the acyclic constraints into a linear programming problem. Thus it can utilize the state-of-art integer programming solver. But this approach requires a bound on the maximum number of node parents in the graph (say to be 4 or 5). Otherwise, the number of constraints in the ILP will explode and the computation will be intractable.

[Index](#)[Previous](#)[Next](#)
