Bayesian Structure Learning

R. B. Alexander

Bayesian structure learning is a combined structure learning and parameter learning task that involves learning the structure of a Bayesian network (a directed acyclic graph) G from a dataset D. The dataset contains n discrete random variables $\mathbf{X}_{1:n}$. Each of the variables has $\mathbf{r}_i$ possible instantiations and for a given graph G, each of the variables has $\mathbf{q}_i$ possible instantiations of its parents π_{ij}. The number of times $\mathbf{X}_i = \mathbf{k}$ given π_{ij} occurs in the dataset is $\mathbf{m}_{ijk}$ and the associated probability of $\mathbf{X}_i = \mathbf{k}$ given π_{ij} is $\mathbf{P}(\mathbf{X}_i = \mathbf{k} \mid \pi_{ij}) = \boldsymbol{\theta}_{ijk}$. Using the Bayesian score function, we can estimate the likelihood of a graph structure given the dataset. Once we have computed the Bayesian score, we must search the space of all Bayesian networks $\mathcal{G}$ to find the graph that maximizes the Bayesian score and is thus, the most probable graph.

Bayesian-Dirichlet Score Function

For this project, the Bayesian-Dirichlet scoring function was used, which assumes a Dirichlet prior over the Bayesian network parameters ($P(\boldsymbol{\theta}) \sim \text{Dir}(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$). The Bayesian-Dirichlet scoring function can be shown to take the following form, where Γ is the gamma function, $\Gamma(n) = (n-1)!$.

$$\ln P(G \mid D) = \ln P(G) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \ln \left(\frac{\Gamma(\alpha_{ij0})}{\Gamma(\alpha_{ij0} + m_{ij0})} \right) + \sum_{k=1}^{r_i} \ln \left(\frac{\Gamma(\alpha_{ijk} + m_{ijk})}{\Gamma(\alpha_{ijk})} \right)$$

with α_{ij0} and m_{ij0} defined as:

$$\alpha_{ij0} = \sum_{k=1}^{r_i} \alpha_{ijk}$$

$$m_{ij0} = \sum_{k=1}^{r_i} m_{ijk}$$

Since we have no information about which graph structures are more or less probable, we use a uniform graph prior, $P(G) = 1$. Under weak assumptions, we can assume a uniform Dirichlet prior over the Bayesian network parameters where $\alpha_{ijk} = \alpha$. Here, we set $\alpha = 1$, which gives the K2 scoring function:

$$\ln P(G \mid D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \ln \Gamma(r_i) - \ln \Gamma(r_i + m_{ij0}) + \sum_{k=1}^{r_i} \ln \Gamma(1 + m_{ijk})$$

Graph Search Algorithms

The space of directed acyclic graphs is superexponential with the number of nodes, so an efficient search strategy is critical in finding an optimal Bayesian network. Several algorithms for graph search exist and two relevant classes of these algorithms are directed graph search algorithms and partially-directed graph search algorithms. Two prominent algorithms for directed graph search are K2 search, which greedily adds parents to a node until no higher-scoring graphs are found, and local search, which starts from a graph structure and moves to the highest-scoring graph in its neighborhood (defined by elementary graph operations). Algorithms for partially-directed graph search search the space of Markov equivalence classes, which is smaller than the space of directed graphs. Some algorithms exploit the *score equivalence* of the scoring function to minimally search the space of partially-directed graphs, yielding robust searches.

K2 Search with Randomized Start

The K2 search strategy begins from a completely unconnected graph. For a given node ordering $X_{1:n}$, the first node in the ordering X_1 is examined. An edge is added from one node in the exclusive node ordering $X_{1:n\setminus 1}$ to the examined node (for example, $X_2 \rightarrow X_1$), the new graph is verified to be a directed acyclic graph, and the Bayesian score is computed. This process is repeated independently for each node in the exclusive node ordering after removing the previously added edge. Whichever of the new graphs gives the highest score is stored as the new best graph. The process repeats for the same examined node and parents are greedily added until none of the new graphs has a higher Bayesian score than the best graph. Then, this process is repeated by individually examining the next node in the node ordering X_i, and iteratively adding the parents from the new exclusive node ordering $X_{1:n\setminus i}$ until no improvement is made. Once the final node is examined and no improvements can be made, the K2 search is complete.

It is clear that the node ordering strongly affects the final graph that is located in the K2 search. Since we have no information about which ordering is best, we assume a uniform distribution over possible node orderings. In our code, we set the seed of a random number generator for reproducibility and performed K2 searches on random permutations of node orderings. We theorize that a random search over K2 directed graph searches is likely to produce reasonable results much faster than a random directed graph search or a single deterministic K2 directed graph search.

Results

In our implementation, we used K2 search and added randomized starts to identify several most-probable graphs, with one graph having the maximal Bayesian score. In particular, we examined three datasets: the `small` dataset, containing 8 variables; the `medium` dataset, containing 12 variables; and the `large` dataset containing 50 variables. The time for each search is provided in Table 1.

Table 1 Runtimes in Seconds for K2 Search with Randomized Start

	Iterations			
Dataset	*1*	*10*	*100*	*1000*
`small`	3.1187	16.341	104.40	1002.7
`medium`	29.534	249.30	2516.8	
`large`	8219.9			

Since the m_{ijk} counts only change marginally between graphs based on the assignment of new parents, it is possible to efficiently update only the necessary m_{ijk} counts, leading to significant computational savings, especially on larger datasets. We did not implement efficient caching or recomputation of the counts, so the searches were quite computationally expensive. As a result, we limited the number of parents of any node to 8 to minimize the associated computational cost. It should be noted that in a K2 search, as parents are added, the Bayesian counts become more complex, but the total number of counts remains unchanged. At the same time, as more parents are added, the possibility of directed cycles increases and thus the space of valid Bayesian networks decreases, so the number of required computations tends to decrease as the search progresses. Figure 1 shows the maximum Bayesian score and total search time as a function of the number of randomized search iterations. We observe only a marginal improvement in score with the logarithm of the number of iterations, but the computations scale linearly with time as expected.

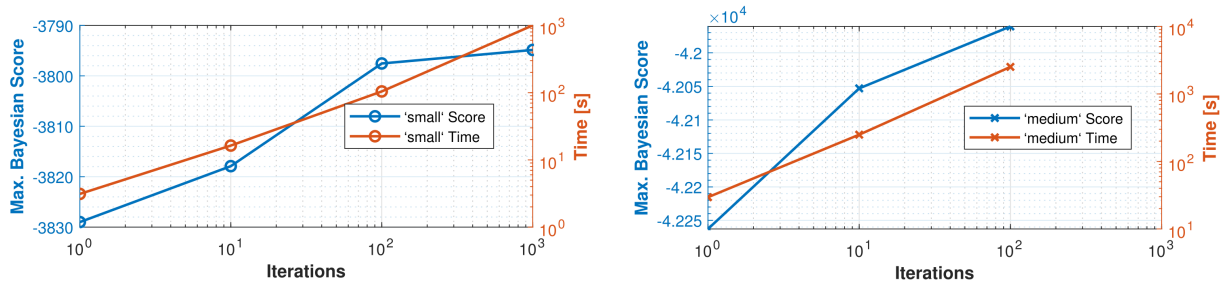

Figure 1 Bayesian score improvement using K2 search iterated over randomly-permuted variable orderings for the `small` (*left*) and `medium` (*right*) datasets.

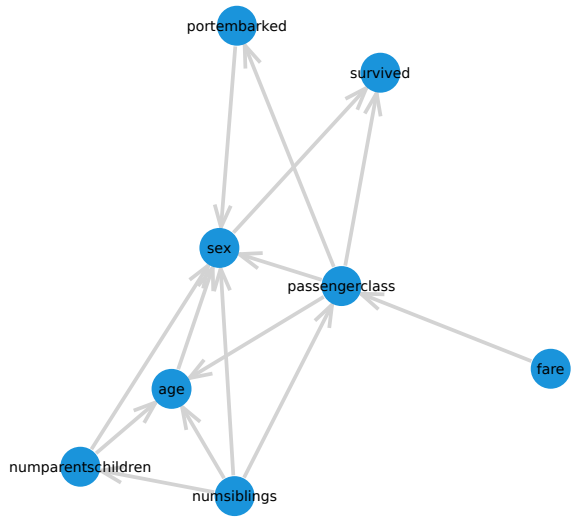

Figure 2 Bayesian network learned from the `small` dataset (8 variables) using a K2 search of the space of directed acyclic graphs with 1000 randomized starts. ($\ln P(G \mid D) \approx -3795$)

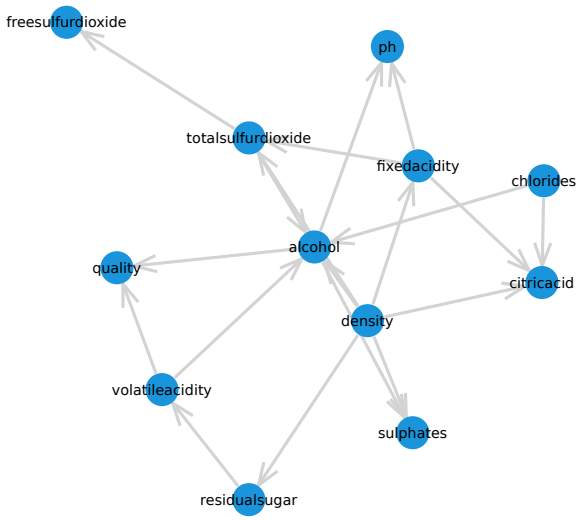

Figure 3 Bayesian network learned from the `medium` dataset (12 variables) using a K2 search of the space of directed acyclic graphs with 100 randomized starts. ($\ln P(G \mid D) \approx -41961$)

The results of the graph searches produced the graphs displayed in Figures 2, 3, and 4. In the `small` and `medium` graphs, we observe limited connectivity, whereas in the `large` graph, we can see a high degree of connection between the nodes. This demonstrates one of the core strengths of the Bayesian structure learning approach - ***balancing model complexity with availability of data***.

In general, we were able implement a Bayesian structure learning framework with 1) a Bayesian scoring function and 2) directed graph search algorithms.

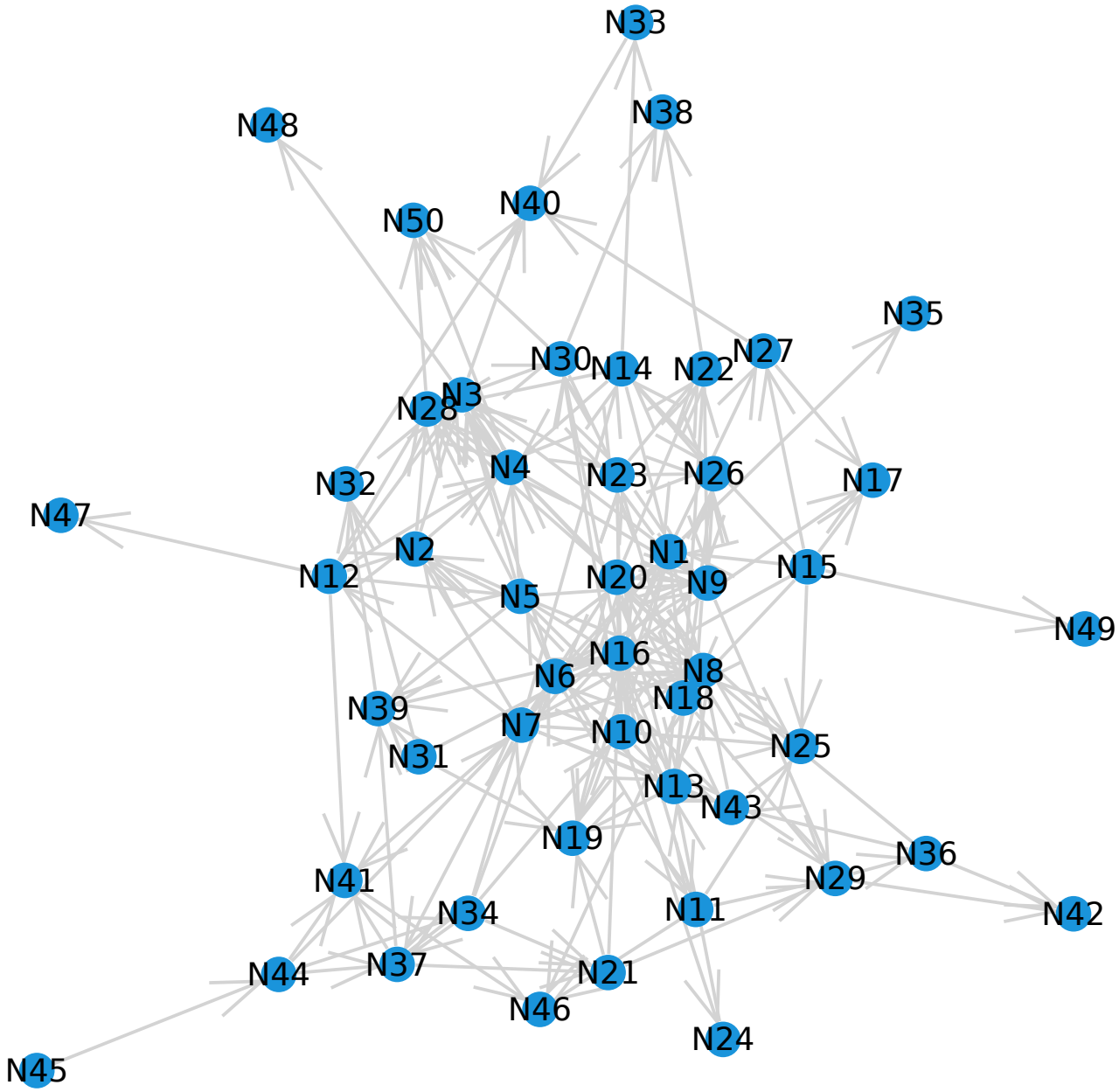

Figure 4 Bayesian network learned from the `large` dataset (50 variables) using a K2 search of the space of directed acyclic graphs with 1 randomized start. ($\ln P(G \mid D) \approx -427612$)