Initialization Heuristics for Greedy Bayesian Network Structure Learning

Walter Perez Urcia   Denis Deratani Mauá

Universidade de São Paulo
Instituto de Matemática e Estadística
Departamento de Ciências da Computação

KDMiLE 2015

Contents

1. Motivation
2. Bayesian Networks
3. Learning Bayesian networks from data
4. Initializing Heuristics
5. Experiments

Motivation

Car Evaluation Dataset

- Buying price (B): v-high, high, med, low
- Maintain cost (M): v-high, high, med, low
- Doors (D): two, three, four, more
- Persons (P): two, four, more
- Luggage boot (L): small, medium, big
- Safety (S): low, medium, high

Represent:

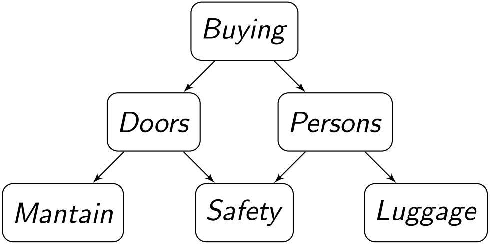
- Half of cars that have four doors have a medium luggage boot
- 15% of cars are low safety, 77% medium safety and 8% high safety

Using a probabilistic model of knowledge to represent all possible relations we have:

$$\mathbb{P}(B, M, D, P, L, S)$$

This requires $4 \times 4 \times 4 \times 3 \times 3 \times 3 = 1728$ probabilities hard to estimate, but we can drastically reduce this number by assuming (conditional) independences

For example:



- Doors and Persons are independent given Buying:
  $\mathbb{P}(D, P \mid B) = \mathbb{P}(D \mid B)\mathbb{P}(P \mid B)$
- Mantain and Safety are independent given Doors:
  $\mathbb{P}(M, S \mid D) = \mathbb{P}(M \mid D)\mathbb{P}(S \mid D)$

  $\vdots$

Bayesian Networks

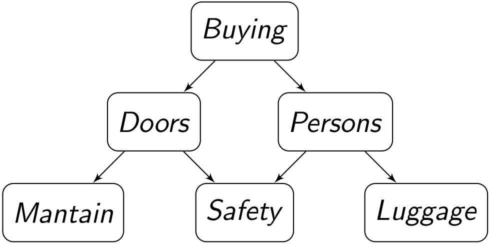
A Bayesian Network consists of

- A DAG G over a set of variables $X_1, \ldots, X_n$
- Markov Property: Given its parents, every variable is conditionally independent from its non-descendant non-parents
- Probability constraints: $\mathbb{P}(X_i = k \mid Pa(X_i) = j) = \theta_{ijk}$

Joint Probability Distribution

There is a unique probability function consistent with a BN:

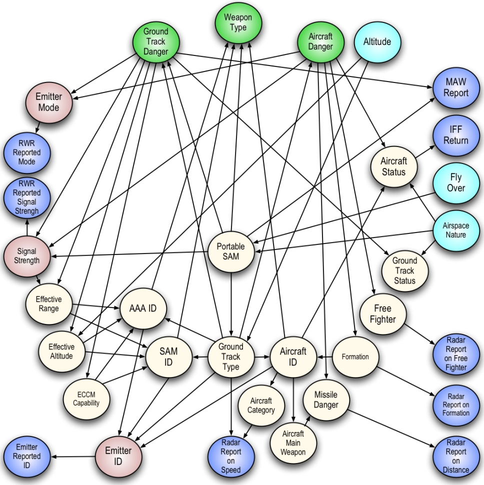
$$\mathbb{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid Pa(X_i)) = \prod_{i=1}^{n} \theta_{ijk}$$



$$\mathbb{P}(B, M, D, P, L, S) = \mathbb{P}(B)\mathbb{P}(D \mid B)\mathbb{P}(P \mid B)\mathbb{P}(M \mid D)\mathbb{P}(S \mid D, P)\mathbb{P}(L \mid P)$$

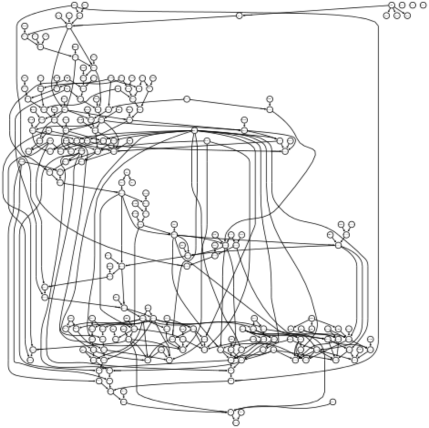
This requires
$4 + (4 \times 4) + (3 \times 4) + (4 \times 4) + (3 \times 4 \times 3) + (3 \times 3) = 93$
probabilities instead of 1728

Consider each variable has k values:
We requires k^{33} probabilities without independences.



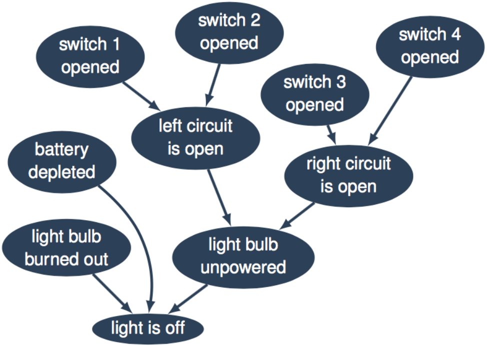
- Elicitation from expert knowledge
- Direct translation
- Learning from data

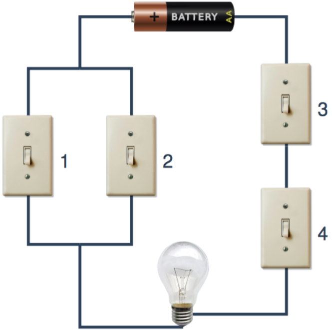
Elicitation



ANDES: Intelligent Tutoring System to teach Newtonian Physics

Direct Translation





Learning Bayesian networks from data

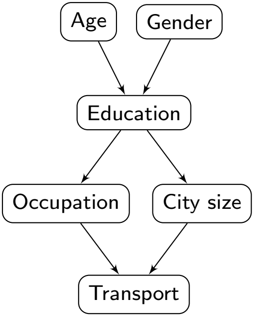
Learning BN from data

Given a data set infers a Bayesian network structure

Learning BN from data

Given a data set infers a Bayesian network structure

Age	Gender	City Education	Occupation	Transport	
adult	F	big	high	employee	car
adult	M	small	uni	employee	car
adult	F	big	uni	employee	train
young	M	big	high	self-emp	car
adult	M	big	high	employee	car
⋮	⋮	⋮	⋮	⋮	⋮



Constraint-based approaches

Perform multiple conditional independence hypothesis testing in order to build a DAG

Score-based approaches

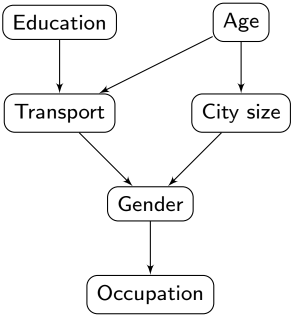
Associate every DAG with a polynomial-time computable score value and search for structure with high score values

Learning as optimization

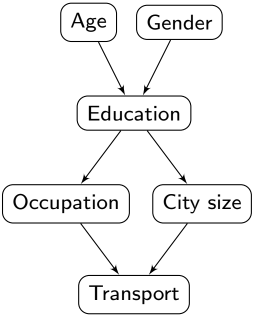
Given dataset D, select G that maximizes decomposable score function:

$$sc(G, D) = LL(D \mid G) + \psi(N) \times |G|$$

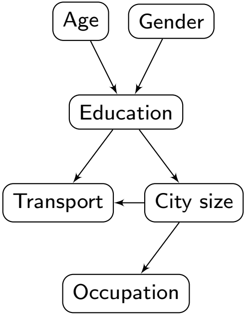
$$sc(G) = \sum_i sc(X_i, Pa(X_i))$$



$$sc(G) = -9508.34$$



$$sc(G) = -6917.23$$



$$sc(G) = -8891.52$$

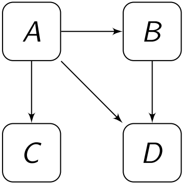
Greedy Search is a popular approach to find an approximate solution

```
GreedySearch( Dataset D , Solution G_0 ) : return a BN G
  G = G_0
  For a number of iterations K
    best_neighbor = find_best_neighbor(G)
    if score(best_neighbor) > score(G) then
      G = best_neighbor
  Return G
```

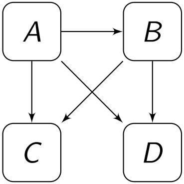
Greedy Search approaches for learning Bayesian networks can be classified as:

- Equivalence-based
- Structure-based
- Order-based

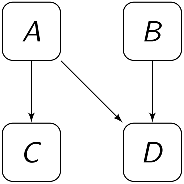
Consider incumbent solution is



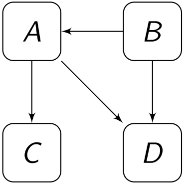
Neighboorhood:



Add an edge



Remove an edge



Revert an edge's direction

Based on the observation that the problem of learning a Bayesian network can be written as

$$G^* = \arg\max_{<} \max_{G \text{ consistent with } <} \sum_{i=1}^{n} sc(X_i, Pa(X_i))$$

$$G^* = \arg\max_{<} \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\}} sc(X_i, P)$$

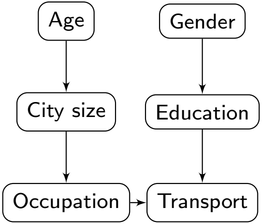
An optimal DAG can be found by maximizing the local scores independently given an order of the variables

```
OrderBasedGreedySearch( Dataset D , Order L_0 ) :
return a BN
  L = L_0
  For a number of iterations K
    current_sol = L
    For each i = 1 to n-1 do
      L_i = swap(L, i, i+1)
      if score(L_i) > score(current_sol)
        current_sol = L_i
    if score(current_sol) > score(L) then
      L = current_sol
  Return network(L)
```

where $swap(L, i, i+1)$ swaps the values $L[i]$ and $L[i+1]$

Consider incumbent solution is

$$[A, G, C, E, O, T]$$



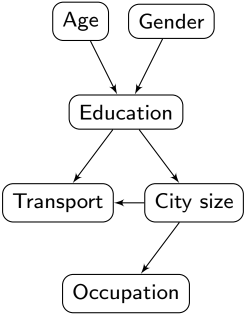
$$sc = -13192.33$$

Neighborhood:

- $[G, A, E, C, O, T]$
  $sc = -10593.82$
- $[A, C, G, E, O, T]$
  $sc = -10891.48$
- $[A, G, E, C, O, T]$
  $sc = -8991.52$
- $[A, G, C, O, E, T]$
  $sc = -9917.23$
- $[A, G, C, E, T, O]$
  $sc = -9158.42$

Now, incumbent solution is

$$[A, G, E, C, O, T]$$

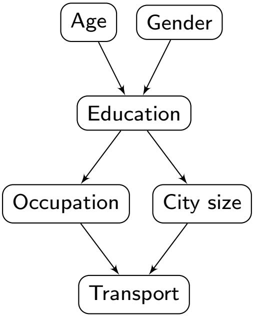


$$sc = -8991.52$$

- $[G, A, E, C, O, T]$
  $sc = -7593.82$
- $[A, E, G, C, O, T]$
  $sc = -8891.48$
- $[A, G, C, E, O, T]$
  $sc = -13192.33$
- $[A, G, E, O, C, T]$
  $sc = -6917.23$
- $[A, G, E, C, T, O]$
  $sc = -6999.99$

Now, incumbent solution is

$$[A, G, E, O, C, T]$$



$$sc = -6917.23$$

- $[G, A, E, O, C, T]$
  $sc = -8593.82$
- $[A, E, G, O, C, T]$
  $sc = -7289.48$
- $[A, G, O, E, C, T]$
  $sc = -9145.13$
- $[A, G, E, C, O, T]$
  $sc = -8991.52$
- $[A, G, E, O, T, C]$
  $sc = -6991.08$

Problems

- Too many possible orders: $n!$
- Slow convergence
- Poor local maxima

C

Initializing Heuristics

We propose two different approaches to reduce the space of possible orders:

- DFS-based approach
- FAS-based approach

We can have an upper bound for $sc(G^*)$ by getting $sc(\overline{G})$

$$\overline{G} = \arg \sum_i \max_{Pa(X_i)} sc(X_i, Pa(X_i))$$

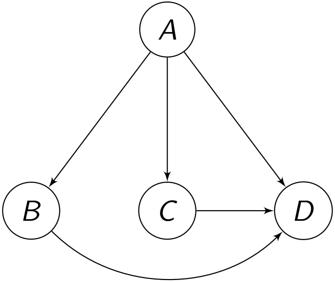
Best Parent Set

The parents of a variable X_i in graph $\overline{G}$

- We can exploit the information provided by $\overline{G}$ and avoid generating orders which are guaranteed sub-optimal
- Assume best parent set is unique
- Consider two variables X_i and X_j in $\overline{G}$, where X_j is parent of X_i, but there is no arc from X_i to X_j
- No optimal ordering can have X_i preceding X_j

The number of these orderings can be much smaller than the full space of orderings

Example



Graph $\overline{G}$

ABDC	CDBA
ACDB	DABC
ADBC	DACB
ADCB	DBAC
BDAC	DBCA
BDCA	DCAB
CDAB	DCBA

Possible orders from $\overline{G}$

The algorithm

- Take as input $\overline{G}$ and mark all nodes unvisited
- Start with an empty list L
- While there is an unvisited node
  - Select an unvisited X_i uniformly random
  - Perform a depth-first search (DFS) rooted at X_i and add to L the visited nodes
- Return L

Disadvantage of DFS approach

This approach can be seen as removing edges from $\overline{G}$ such as to make it a DAG and then extract a topological order. But not all edges are equally relevant in terms of avoiding poor local maxima.

Estimating the relevance

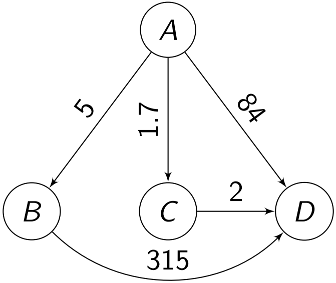
We can estimate the relevance of an edge $X_j \to X_i$ by

$$W_{ji} = sc(X_i, Pa^*(X_i)) - sc(X_i, Pa^*(X_i) \setminus \{X_j\})$$

where $Pa^*(X_i)$ represents the best parent set for X_i.

We then wish to find a topological ordering of $\overline{G}$ that violates the least cost of edges.

Example



Graph $\overline{G}$

- C is not very relevant as parent to D
- B is the most relevant parent of D

Min-Cost Feedback Arc Set

Given a weighted directed graph $G = (V, E)$, a set $F \subseteq E$ is called a Min-Cost Feedback Arc Set (min-cost FAS) if every (directed) cycle of G contains at least one edge in F and the sum of weights is minimum.

$$F = \min_{G-F \text{ is a DAG}} \sum_{X_j \to X_j \in E} W_{ij}$$

Finding FAS F

The following algorithm finds an approximate solution:

```
MinimumCostFAS( Graph G ) : Return FAS F
  F = empty set
  While there is a cycle C on G do
    W_min = lowest weight of all edges in C
    For each edge (u,v) ∈ C do
      W_uv = W_uv − W_min
    If W_uv = 0 add to F
  For each edge in F, add it to G if does not build a
      cycle
  Return F
```

The algorithm

- Take the weighted graph $\overline{G}$ with weights W_{ij} as input
- Find min-cost FAS F
- Remove the edges in F from $\overline{G}$
- Return a topological order from $\overline{G} - F$

Experiments

Configuration

For each dataset considered:

- Limit parent set size to 3
- Perform 1000 runs of Order-based Greedy Search
- For each run at most 100 iterations ($K = 100$)
- Use BIC score
- Find best parent sets by exhaustive search

Datasets

Dataset	n (#attributes)	N (#instances)	Density of $\overline{G}$
Census	15	30168	2.85
Letter	17	20000	2.41
Image	20	2310	2.45
Mushroom	23	8124	2.91
Sensors	25	5456	3.00
SteelPlates	28	1941	2.18
Epigenetics	30	72228	1.87
Alarm	37	1000	1.98
Spectf	45	267	1.76
LungCancer	57	27	1.44

Table 1: Data sets characteristics

Small Domains

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
Census	Random	**-212186.79**	-213074.18 ± 558.43	-212342.26 ± 174.21	7.26 ± 2.90
	DFS-based	-212190.05	-212736.80 ± 379.96	-212339.83 ± 152.26	5.90 ± 2.61
	FAS-based	-212191.64	**-212287.99 ± 92.54**	**-212222.12 ± 70.99**	**3.28 ± 1.67**
Letter	Random	-138652.66	-139774.54 ± 413.74	-139107.13 ± 329.15	6.07 ± 2.50
	DFS-based	-138652.66	-139521.38 ± 396.61	**-138999.84 ± 310.06**	5.75 ± 2.35
	FAS-based	-138652.66	**-139050.43 ± 70.55**	-139039.26 ± 87.97	**2.24 ± 0.96**
Image	Random	**-12826.08**	-13017.13 ± 44.35	-12924.24 ± 41.39	7.59 ± 2.71
	DFS-based	-12829.10	-12999.09 ± 38.56	-12921.13 ± 37.88	7.10 ± 2.47
	FAS-based	-12829.10	**-12930.63 ± 20.83**	**-12882.30 ± 26.43**	**5.05 ± 1.72**
Mushroom	Random	**-55513.38**	-58450.72 ± 1016.54	-56563.84 ± 616.59	7.59 ± 2.76
	DFS-based	**-55513.38**	-58367.11 ± 871.25	-56472.72 ± 546.19	7.75 ± 2.58
	FAS-based	-55574.71	**-56450.49 ± 154.54**	**-56198.66 ± 174.64**	**4.65 ± 1.63**
Sensors	Random	**-62062.13**	-63476.33 ± 265.46	-62726.60 ± 251.26	9.22 ± 2.94
	DFS-based	-62083.21	-63392.60 ± 255.90	-62711.50 ± 257.79	9.65 ± 3.12
	FAS-based	-62074.88	**-62530.26 ± 133.44**	**-62330.94 ± 121.82**	**5.17 ± 2.24**

Large Domains

Dataset	Approach	Best Score	Avg. Initial Score	Avg. Best Score	Avg. It.
SteelPlates	Random	-13336.14	-13566.50 ± 65.80	-13429.13 ± 52.14	8.96 ± 3.43
	DFS-based	**-13332.91**	-13572.77 ± 81.12	-13432.30 ± 57.57	9.30 ± 3.38
	FAS-based	-13341.73	**-13485.26 ± 38.27**	**-13397.08 ± 29.53**	**7.77 ± 2.24**
Epigenetics	Random	-56873.76	-57722.30 ± 228.44	-57357.60 ± 222.12	5.89 ± 2.67
	DFS-based	**-56868.87**	**-57615.36 ± 189.17**	**-57308.93 ± 165.18**	6.42 ± 2.47
	FAS-based	**-56868.87**	-57660.09 ± 146.45	-57379.59 ± 148.42	**5.33 ± 2.28**
Alarm	Random	-13218.22	-13324.52 ± 30.49	-13245.43 ± 15.63	10.92 ± 3.24
	DFS-based	**-13217.97**	-13250.72 ± 17.70	-13236.71 ± 12.02	**4.32 ± 2.32**
	FAS-based	-13220.55	**-13249.77 ± 2.57**	**-13233.98 ± 6.19**	6.34 ± 1.74
Spectf	Random	-8176.81	-8202.03 ± 5.23	-8189.69 ± 4.65	7.20 ± 2.17
	DFS-based	**-8172.37**	-8200.04 ± 4.08	-8187.29 ± 4.91	7.86 ± 2.49
	FAS-based	-8172.51	**-8176.98 ± 2.01**	**-8176.07 ± 2.05**	**2.27 ± 1.11**
LungCancer	Random	**-711.23**	-723.79 ± 2.69	-718.03 ± 2.84	5.46 ± 1.78
	DFS-based	-711.36	-720.47 ± 2.51	**-715.29 ± 1.86**	5.02 ± 1.50
	FAS-based	-711.39	**-716.13 ± 0.89**	-715.67 ± 1.19	**2.73 ± 1.79**

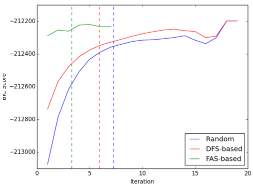


Figure 1: Census

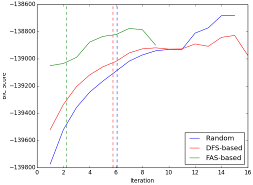


Figure 2: Letter

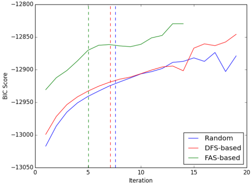


Figure 3: Image

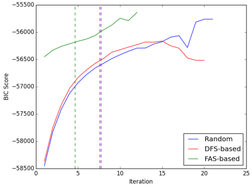


Figure 4: Mushroom

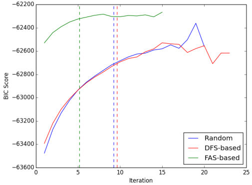


Figure 5: Sensors

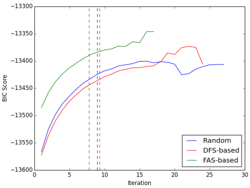


Figure 6: SteelPlates

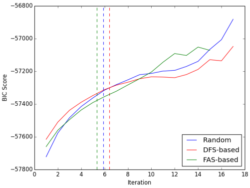


Figure 7: Epigenetics

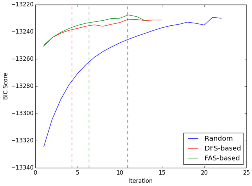


Figure 8: Alarm

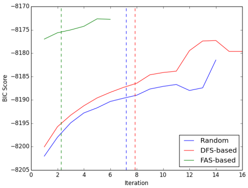


Figure 9: Spectf

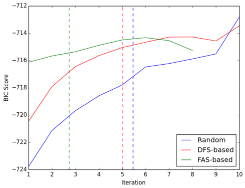


Figure 10: LungCanc

- The proposed heuristics lead to better solutions on average, and increase the convergence of the search with only a small overhead
- Larger diferences for datasets withs more variables are expected
- Our proposed techniques could return directed acyclic graphs instead of node orderings to be used for Structure- and Equivalence-based search approaches
- Employ the proposed heuristics in branch-and-bound solvers for finding optimal solutions

Thanks!