Learning Bayesian Networks: Shortest Path Perspective

Walter Perez Urcia

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

Novembro 2015

Outline

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

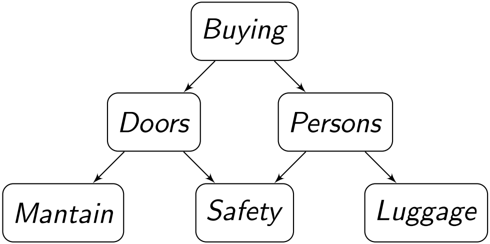
There is a unique probability function consistent with a BN:

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

Car Evaluation Dataset

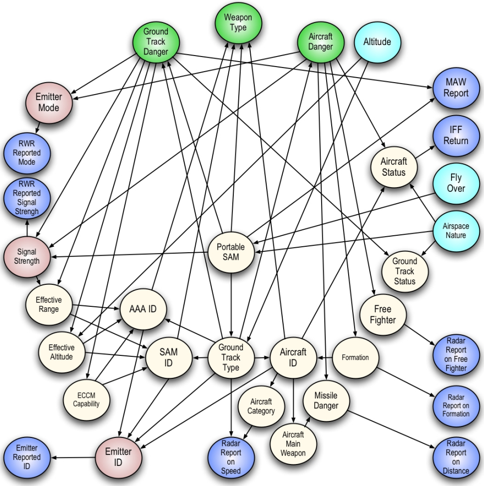
- Buying price (B): v-high, high, med, low
- Maintain cost (M): v-high, high, med, low
- Doors (D): two, three, four, more
- Persons (P): two, four, more
- Luggage boot (L): small, medium, big
- Safety (S): low, medium, high

We can construct manually



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

Example



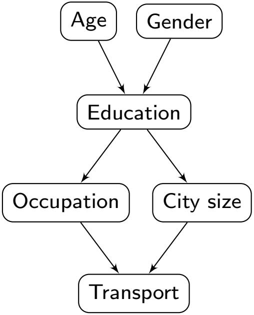
Ground Track Danger
Weapon Type
Aircraft Danger
Altitude
Emitter Mode
MAW Report
IFF Return
RWR Reported Mode
RWR Reported Signal Strengh
Aircraft Status
Fly Over
Airspace Nature
Signal Strength
Portable SAM
Ground Track Status
Effective Range
AAA ID
Free Fighter
Effective Altitude
SAM ID
Ground Track Type
Aircraft ID
Formation
Radar Report on Free Fighter
ECCM Capability
Aircraft Category
Missile Danger
Radar Report on Formation
Emitter Reported ID
Emitter ID
Radar Report on Speed
Aircraft Main Weapon
Radar Report on Distance

Learning Bayesian networks from data

Learning BN from data

Given a data set infers a Bayesian network structure

Age	Gender	City Size	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

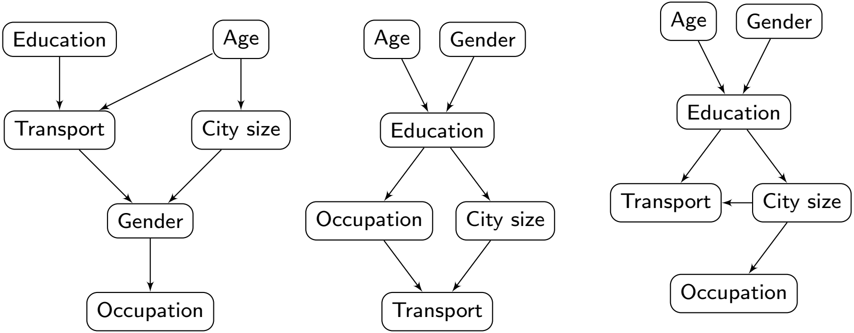
Given dataset D, select G that maximizes decomposable score function:

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

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

Most common scores are:

- BIC: $\psi(N) = log(N)/2$
- AIC: $\psi(N) = 1$



$sc(G) = -9508.34$ $sc(G) = -6917.23$ $sc(G) = -8891.52$

Score-based structure learning is an NP-Hard problem

Score-based structure learning is an NP-Hard problem

Suppose we have a dataset with n attributes

- Number of scores: $n2^{n-1}$
  - e.g. For $n = 20$, there are 10 million of scores

Score-based structure learning is an NP-Hard problem

Suppose we have a dataset with n attributes

- Number of scores: $n2^{n-1}$
  - e.g. For $n = 20$, there are 10 million of scores
- Number of possible DAGs: $O(n2^{n(n-1)})$
  - e.g. For $n = 5$, there are 5 million DAGs

- Limit the maximum number of parents for each variable

Theorem 1

In an optimal Bayesian network based on the BIC scoring function, each variable has at most $d = \lfloor log(\frac{2N}{logN}) \rfloor$ parents, where N is the number of instances.

- Limit the maximum number of parents for each variable

Theorem 1

In an optimal Bayesian network based on the BIC scoring function, each variable has at most $d = \lfloor log(\frac{2N}{logN}) \rfloor$ parents, where N is the number of instances.

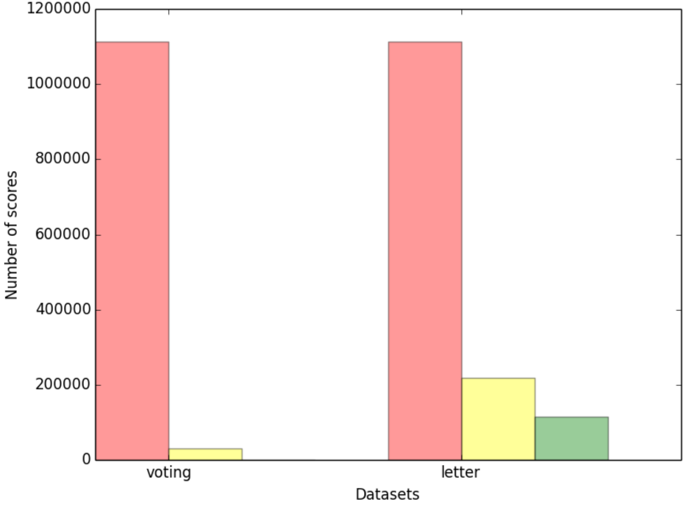
- Prune non-optimal candidate parent sets

Theorem 2

Let U and S be two candidate parent sets for X such that $U \subset S$, and $sc(X, U)$ is better than $sc(X, S)$. Then S is not the optimal parent set of X for any candidate set.

Dataset	n (#attributes)	N (#instances)
Census	15	30168
Voting	17	232
Letter	17	20000
Hepatitis	17	80
Image	20	2310
Heart	23	80
Mushroom	23	8124
Parkinsons	23	195
Autos	25	159
Flag	28	194

Table 1: Data sets characteristics


1200000
1000000
800000
600000
400000
200000
0
Number of scores
voting
letter
Datasets

Dataset	d	Total Scores	Limited Scores	Pruned Scores
Census	8	245K	45K (18.33%)	3.5K (1.44%)
Voting	4	1.1M	31K (2.78%)	939 (0.08%)
Letter	8	1.1M	218K (19.64%)	115K (10.38%)
Hepatitis	3	1.1M	9.5K (0.85%)	174 (0.02%)
Image	6	10M	542K (5.18%)	6.2K (0.06%)
Heart	3	96M	35K (0.04%)	327 (0.00034%)
Mushroom	7	96M	3.9M (4.07%)	13K (0.014%)
Parkinsons	4	96M	168K (0.17%)	2.4K (0.003%)
Autos	4	419M	265K (0.06%)	3K (0.0007%)
Flag	4	3.7B	491K (0.01%)	776 (0.00002%)

Table 2: Number of scores

A Star Approach

Main idea of algorithm

Evaluate nodes with function f and expand to the one with lowest f value.

$$f(U) = g(U) + h(U)$$

- States: sets of variables
- Initial state: $\emptyset$
- Final state: $\{X_1, X_2, \ldots, X_n\}$
- $g(U \cup \{X\}) = sc(U) + BestScore(X, U)$

$$BestScore(X, U) = \min_{P \subset U} sc(X, P)$$

Algorithm 1: A^* Search Algorithm

Input : Sparse parent graphs containing $BestScore(X, U)$

Output: an optimal Bayesian network G

```
start ← ∅
sc(start) ← 0
push(pqueue, start, h(start))
while not_empty(pqueue) do
    U ← pop(pqueue)
    if U is goal then
        return network(U)
    end
    for each X ∈ V \ U do
        g ← sc(U) + BestScore(X, U)
        if pqueue not contains U ∪ {X} or g < sc(U ∪ {X}) then
            sc(U ∪ {X}) = g
            if pqueue contains U ∪ {X} then
                update(pqueue, U ∪ {X}, g + h(U ∪ {X}))
            else
                push(pqueue, U ∪ {X}, g + h(U ∪ {X}))
            end
        end
    end
end
```

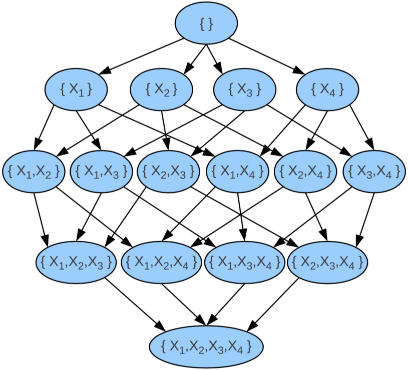


Figure 1: Order graph for four variables

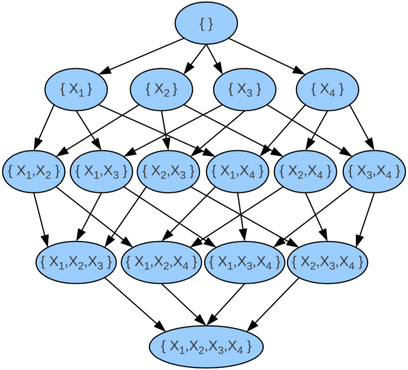


Figure 1: Order graph for four variables

What about the heuristics?

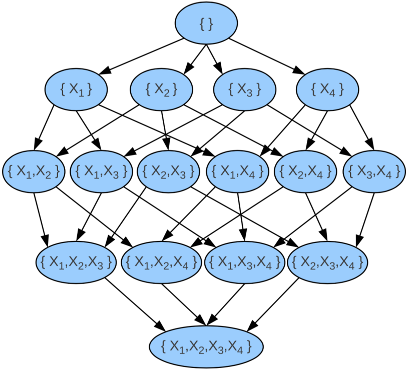


Figure 1: Order graph for four variables

What about the heuristics?

We will discuss only three heuristics:

- Simple heuristic
- Dynamic K-Cycle
- Static K-Cycle

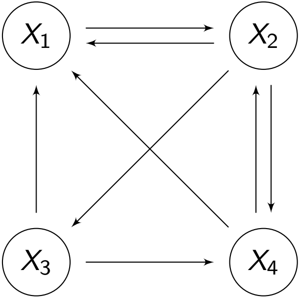
Simple heuristic

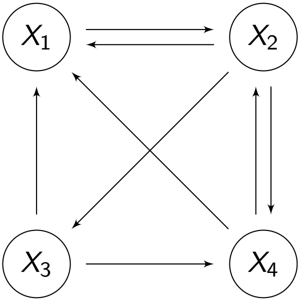
$$h_{simple}(U) = \sum_{X \in V \setminus U} BestScore(X, V \setminus \{X\})$$

Simple heuristic

$$h_{simple}(U) = \sum_{X \in V \setminus U} BestScore(X, V \setminus \{X\})$$

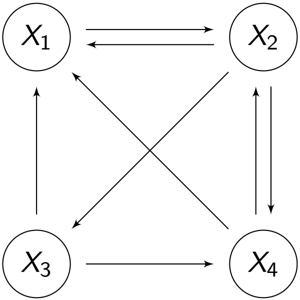
For example for *start* node:





Cycle between X_1 and X_2

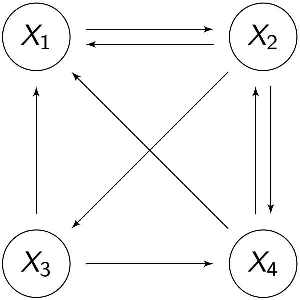
- Delete arc $X_1 \rightarrow X_2$, $BestScore(X_2, \{X_3, X_4\})$
- Delete arc $X_2 \rightarrow X_1$, $BestScore(X_1, \{X_3, X_4\})$



Cycle between X_1 and X_2

- Delete arc $X_1 \rightarrow X_2$, $BestScore(X_2, \{X_3, X_4\})$
- Delete arc $X_2 \rightarrow X_1$, $BestScore(X_1, \{X_3, X_4\})$

What about cycle between X_1, X_2 and X_4?



Cycle between X_1 and X_2

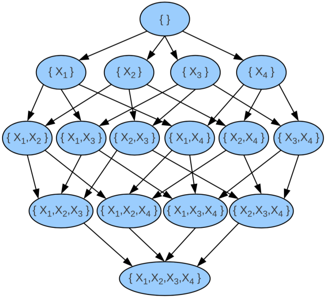
- Delete arc $X_1 \rightarrow X_2$, $BestScore(X_2, \{X_3, X_4\})$
- Delete arc $X_2 \rightarrow X_1$, $BestScore(X_1, \{X_3, X_4\})$

What about cycle between X_1, X_2 and X_4?

h_{simple} is a special case with $K = 1$

Main idea

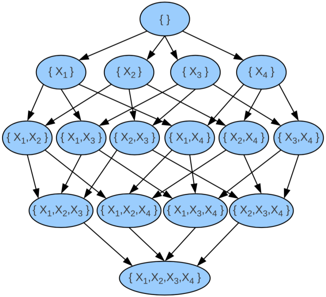
Compute the costs for all groups of variables with size up to K and store them in a single pattern database



Theorem 3

The cost of the pattern U, $c(U)$, is equal to the shortest distance from $V \setminus U$ to the goal node in the order graph.

$$cost(U) = shortest\_distance(V \setminus U)$$



The algorithm is as follows:

- Perform a breadth-first search in the last K layers of order graph
  - In each node S, update $shortest\_distance(S)$
  - Calculate $diff(S)$, the difference between the cost and h_{simple}
  - Prune S if does not improve related to h_{simple}
  - Set pattern $cost(V \setminus S) = shortest\_distance(S)$
- Sort patterns based on $diff(S)$ in descending order

The algorithm is as follows:

- Perform a breadth-first search in the last K layers of order graph
  - In each node S, update $shortest\_distance(S)$
  - Calculate $diff(S)$, the difference between the cost and h_{simple}
  - Prune S if does not improve related to h_{simple}
  - Set pattern $cost(V \setminus S) = shortest\_distance(S)$
- Sort patterns based on $diff(S)$ in descending order

How do I calculate the heuristic $h_{dynamic}$?

Algorithm 2: $h_{dynamic}(U)$

$h \leftarrow 0$
$R \leftarrow U$
for *each* $S \in PD$ **do**
    if $S \subset R$ **then**
        $R \leftarrow R \setminus S$
        $h \leftarrow h + PD(S)$
    end
end
return h

Algorithm 3: $h_{dynamic}(U)$

```
h ← 0
R ← U
for each S ∈ PD do
    if S ⊂ R then
        R ← R \ S
        h ← h + PD(S)
    end
end
return h
```

Compute $h_{dynamic}$ is more expensive than h_{simple}

Main idea

To partition the variables into several static exclusive groups V_i, and create a separate pattern database PD^i for each group

Main idea

To partition the variables into several static exclusive groups V_i, and create a separate pattern database PD^i for each group

Algorithm 5: $h_{static}(U)$

```
h ← 0
for each V_i ∈ V do
    h ← h + PD^i(U ∩ V_i)
end
return h
```

Experiments

Configuration

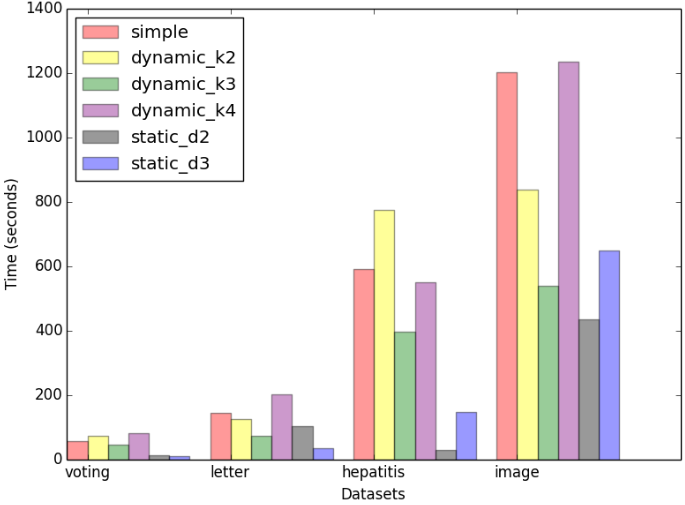
Configuration

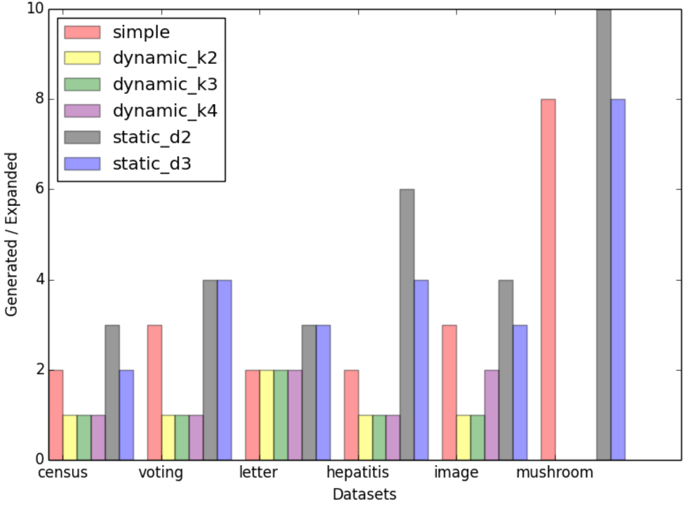
For each dataset and heuristic:

- Use BIC score
- Time limit: 2,5 hours
- For $h_{dynamic}$ considers $K = 2, 3, 4$
- For h_{static} considers two and three groups

Dataset	n (#attributes)	N (#instances)
Census	15	30168
Voting	17	232
Letter	17	20000
Hepatitis	17	80
Image	20	2310
Heart	23	80
Mushroom	23	8124
Parkinsons	23	195
Autos	25	159
Flag	28	194

Table 3: Data sets characteristics


simple
dynamic_k2
dynamic_k3
dynamic_k4
static_d2
static_d3
Time (seconds)
0
200
400
600
800
1000
1200
1400
voting
letter
hepatitis
image
Datasets


simple
dynamic_k2
dynamic_k3
dynamic_k4
static_d2
static_d3
Generated / Expanded
10
8
6
4
2
0
census
voting
letter
hepatitis
image
mushroom
Datasets

What happened with approximation methods?

Order-based Greedy Search

- Random approach
- DFS-based approach
- FAS-based approach

Dataset	Static D2	Static D3	Random	DFS-based	FAS-based
Census	5.873282	3.008102	18.516816823	10.804413507	2.80409026
Voting	13.692846	10.288065	0.263474565	0.276129374	0.096767351
Letter	103.601706	36.600822	72.033615582	79.929216127	80.7415335826
Hepatitis	30.095161	147.230065	0.146427219	0.143075645	0.077747351
Image	435.832472	648.232549	2.771176306	2.88687619	2.44088269
Mushroom	102.140279	255.284945	9.673048986	9.652037496	0.635835907
Parkinsons	3484.049183	553.567477	0.776882477	0.651769457	0.583941459
Autos	406.32017	2487.784721	1.173008457	1.266957928	1.182414249
Flag	1363.055556	TLE	0.686072783	0.608696292	0.493342731

Table 4: Comparison of times (seconds) with approximation methods

- Static K-Cycle heuristic reduces the total time to find the optimal Bayesian network compared to other heuristics
- Some approximation methods find Bayesian networks with an optimal score on considerable less time in most cases for the datasets

Bibliography

- *Learning Optimal Bayesian Networks: A Shortest Path Perspective*. Changhe Yuan and Brandon Malone
- *Initialization Heuristics for Greedy Bayesian Network Structure Learning*. Walter Perez and Denis D. Mauá

Thanks!

Learning Bayesian Networks: Shortest Path Perspective

Walter Perez Urcia

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

Novembro 2015