

Markov Library: A Tool for Computing Ideals Associated to Bayesian Networks

Changsung Kang and Luis David Garcia

The Markov library contains methods to compute the ideals and primary decomposition associated to Bayesian networks described in [1]. This library is implemented in a computer algebra system, Singular ([2]). This document describes how to use this library for the various computations. In Section 1, we show how to compute various ideals associated to Bayesian networks without hidden variables. In Section 2, we describe how to compute the polynomial constraints for Bayesian networks with hidden variables.

1 Markov Ideals of Bayesian Networks

Consider the Bayesian network G in Figure 1 which has 4 vertices. Assume that all the random variables take binary values and the probability distribution associated to the Bayesian network is represented by indeterminates $P_{1111}, \dots, P_{2222}$. The command `probring(d)` where `d` is a list of integers that represent the dimensions of variables creates a polynomial ring associated to the probability distribution.

```
intvec d = 2,2,2,2;
int n = size(d);
def pdR = probring(d);
setring pdR;
pdR;
// characteristic : 0
// number of vars : 16
//      block 1 : ordering dp
//      : names  p1111 p1112 p1121 p1122 p1211
p1212 p1221 p1222 p2111 p2112 p2121 p2122 p2211 p2212 p2221
p2222
//      block 2 : ordering C
```

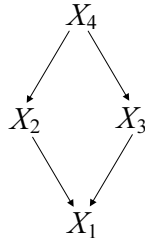


Figure 1: A Bayesian network on four observable variables.

We use an $n \times n$ matrix M to represent the Bayesian network structure with n variables. We assume a topological ordering $X_n > \dots > X_1$. $M[i, j] = 1$ implies the existence of an edge (i, j) . The command `bnet(n, v)` generates an $n \times n$ matrix whose lower triangle is given by the list `v`. Thus, the network structure in Figure 1 is represented as follows.

```

intvec v = 1,1,0,0,1,1;
intmat m = bnet(n,v);
m;
0,0,0,0,
1,0,0,0,
1,0,0,0,
0,1,1,0

```

The commands `localMarkov(m)`, `pairMarkov(m)` and `globalMarkov(m)` generate local, pairwise and global Markov property respectively for the Bayesian network represented by `m`. If `l=localMarkov(m)`, then `l[i]` corresponds to the i th conditional independence statement `l[i][1] ⊥ l[i][2] | l[i][3]`.

```

list l = localMarkov(m);
list pw = pairMarkov(m);
list g = globalMarkov(m);
l;
[1]:
  [1]:
    [1]:
      1
    [2]:

```

```

[1]:
  4
[3]:
  [1]:
    2
  [2]:
    3
[2]:
  [1]:
    [1]:
      2
  [2]:
    [1]:
      3
  [3]:
    [1]:
      4

```

The result of the command `localMarkov` shows that the local Markov property for G is the set $\{X_1 \perp\!\!\!\perp X_4 \mid \{X_2, X_3\}, X_2 \perp\!\!\!\perp X_3 \mid X_4\}$.

The command `MarkovIdeal(L, d)` gives the ideal of the independence model represented by a list L . We can get the codimension, degree and number of minimal generators of an ideal I by the command `info(I)`.

```

ideal I = MarkovIdeal(l,d);
info(I);
// ** I is no standard basis
// ** I is no standard basis
// dimension (proj.) = 10
// degree (proj.) = 24
// ** right side is not a datum, assignment ignored
[1]:
  5
[2]:
  0
[3]:
  6

```

Let $p_{++\dots+u_{r+1}\dots u_n}$ denote the marginalization over the first r random variables and \mathbf{p} denote the product of all of these linear forms. The command `torideal(I, d)`

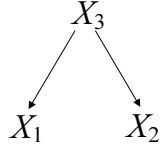


Figure 2: A Bayesian network on two observable variables X_1 and X_2 and one hidden variable X_3 .

computes the ideal $I : \mathbf{p}^\infty$.

```
ideal T = torideal(I,d);
```

We can test if $I_{\text{local}(G)} = I_{\text{global}(G)}$ by executing `quotient(I,G)`.

```
quotient(I,G);
_[1]=1
```

Since $I_{\text{local}(G)} \subseteq I_{\text{global}(G)}$, the result above implies that $I_{\text{local}(G)} = I_{\text{global}(G)}$.

2 Bayesian Networks with Hidden Variables

In this section, we show how to compute the polynomial constraints for a Bayesian network with hidden variables. Let G be a Bayesian network on n discrete random variables. Assume that the variables X_1, \dots, X_r are observed and X_{r+1}, \dots, X_n are hidden. Let $\mathbb{R}[D]$ denote the ring generated by the indeterminates for the probability distribution $P(X_1, \dots, X_n)$ and $\mathbb{R}[D']$ denote the subring of $\mathbb{R}[D]$ generated by the indeterminates for the observable probability distribution $P(X_1, \dots, X_r)$. Our goal is to compute the ideal

$$Q_G = P_G \cap \mathbb{R}[D']$$

where P_G is the distinguished component for G . See [1] for more details.

We will compute Q_G for the Bayesian network G in Figure 2. Assume that X_1 and X_2 are ternary and X_3 is binary. We first declare the polynomial rings $\mathbb{R}[D]$ and $\mathbb{R}[D']$. We use names `pdR` and `H` for $\mathbb{R}[D]$ and $\mathbb{R}[D']$ respectively.

```
intvec d = 3,3,2;
int n = size(d);
```

```

int r = 2;
def pdR = probring(d);
intvec d2 = d[1..r];
def H = probring(d2);
pdR;
// characteristic : 0
// number of vars : 18
//      block 1 : ordering dp
//      : names  p111 p112 p121 p122 p131 p132 p211
p212 p221 p222 p231 p232 p311 p312 p321 p322 p331 p332
//      block 2 : ordering C
H;
// characteristic : 0
// number of vars : 9
//      block 1 : ordering dp
//      : names p11 p12 p13 p21 p22 p23 p31 p32 p33
//      block 2 : ordering C

```

The command `map_observable(H, d, r)` generates the ring map from H to basering induced by the inclusion of H in basering.

```

setring pdR;
def Phi = map_observable(H, d, r);
Phi;
Phi[1]=p111+p112
Phi[2]=p121+p122
Phi[3]=p131+p132
Phi[4]=p211+p212
Phi[5]=p221+p222
Phi[6]=p231+p232
Phi[7]=p311+p312
Phi[8]=p321+p322
Phi[9]=p331+p332

```

$P_G = I_{\text{global}(G)} : \mathbf{p}^\infty$ is computed by the command `torideal` as in Section 1.

```

intvec v = 0,1,1;
intmat m = bnet(n,v);
list g = globalMarkov(m);
ideal G = MarkovIdeal(g,d);

```

```
ideal T = torideal(G,d);
```

Finally, $Q_G = P_G \cap \mathbb{R}[D']$ can be computed by the command `preimage`.

```
setring H;  
ideal Q = preimage(pdR,Phi,T);  
Q;  
Q[1]=p13*p22*p31-p12*p23*p31-p13*p21*p32+p11*p23*p32+p12*p21*p33  
-p11*p22*p33
```

The result above shows that the ideal Q_G is generated by the determinant of the 3×3 matrix (p_{ij}) .

References

- [1] L. D. Garcia, , M. Stillman, and B. Sturmfels: Algebraic geometry of Bayesian networks, *Journal of Symbolic Computation* **39/3-4** (2005) 331-355.
- [2] G.-M. Greuel, G. Pfister and H. Schönemann: Singular 2.0: A computer algebra system for polynomial computations, University of Kaiserslautern, 2001, <http://www.singular.uni-kl.de>.