Bayesian Networks II

Advanced Artificial Intelligence: Workshop

Heriberto Cuayáhuitl
http://staff.lincoln.ac.uk/hcuayahuitl

School of Computer Science, University of Lincoln

16 October 2019

Prior Sampling

- Download (from Blackboard) and run the following program: **PriorSampling.py**

- The program above implements the *PRIOR-SAMPLE* algorithm below in order to return an event of the form [*true*, *false*, *true*, *true*] — use the Sprinkler and Burglary networks[1] for testing your program

function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
  inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return x

[1] Note that you have two versions of the Burglary network, one from the book and the other one from your own calculations (last week)

Implemention of Approximate Inference: Today

- Implement the Rejection Sampling algorithm and use the Sprinkler and Burglary networks for testing your program[2]

function REJECTION-SAMPLING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network
    N, the total number of samples to be generated
  local variables: $\mathbf{N}$, a vector of counts for each value of X, initially zero

  for j = 1 to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    if $\mathbf{x}$ is consistent with $\mathbf{e}$ **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}$)

- Test your program varying the value N (e.g. 10, 10^2, 10^3, 10^4, 10^5). How many samples would you use and why?

[2] Feel free to reuse the code in PriorSampling.py

Implemention of Approximate Inference: Homework

Implement the algs. below and test them with the Sprinkler/Burglary nets

```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns an estimate of P(X|e)
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)
          N, the total number of samples to be generated
  local variables: W, a vector of weighted counts for each value of X, initially zero

  for j = 1 to N do
      x, w ← WEIGHTED-SAMPLE(bn, e)
      W[x] ← W[x] + w where x is the value of X in x
  return NORMALIZE(W)
```

```
function WEIGHTED-SAMPLE(bn, e) returns an event and a weight

  w ← 1; x ← an event with n elements initialized from e
  foreach variable X_i in X_1, ..., X_n do
      if X_i is an evidence variable with value x_i in e
          then w ← w × P(X_i = x_i | parents(X_i))
          else x[i] ← a random sample from P(X_i | parents(X_i))
  return x, w
```

```
function GIBBS-ASK(X, e, bn, N) returns an estimate of P(X|e)
  local variables: N, a vector of counts for each value of X, initially zero
                   Z, the nonevidence variables in bn
                   x, the current state of the network, initially copied from e

  initialize x with random values for the variables in Z
  for j = 1 to N do
      for each Z_i in Z do
          set the value of Z_i in x by sampling from P(Z_i|mb(Z_i))
          N[x] ← N[x] + 1 where x is the value of X in x
  return NORMALIZE(N)
```