CS 188 Notes 2

LEC 16: Bayes Nets: Approximate Inference

Sampling

* Draw N samples from a sampling distribution S
* Compute an approximate posterior probability
* Show this converges to the true probability P
* Why sample?
  + Often very fast to get decent approx. answers
  + Algorithms are very simple and general (easy to apply to fancy models)
  + Require very little memory: O(n)

Sampling basics: discrete (categorical) distribution:

* To simulate a biased d-sided coin:
  + **Step 1:** 
    - Get sample u from uniform distribution over [0, 1)
      * E.g. random() in python
  + **Step 2:** 
    - Convert this sample u into an outcome for the given distribution by associating each outcome x with a P(x)-sized sub-interval of [0,1)

Sampling in Bayes Nets:

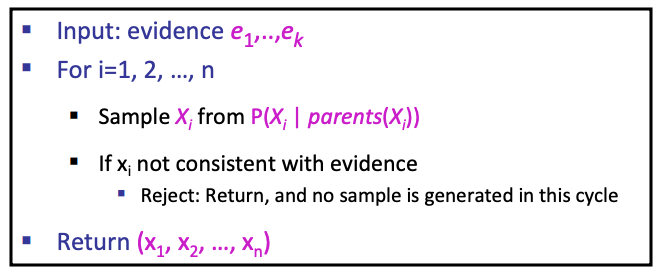
1. Prior Sampling
2. Rejection Sampling
3. Likelihood Weighting
4. Gibbs Sampling

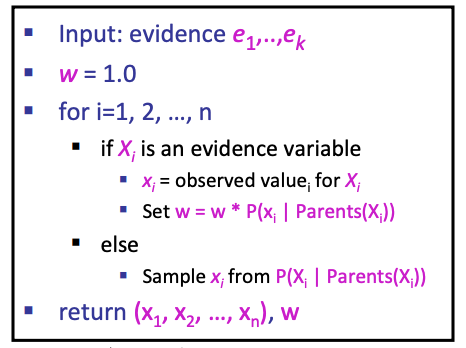
Prior Sampling:

* For i=1, 2, …, n (in topological order)
  + Sample Xi from P(Xi | parents(Xi ))
* Return (x1, x2, …, xn)
* This sample generates samples w/ probability:

Rejection Sampling:

* Simple modification of prior sampling for conditional probabilities
* Say we want P(C| r, w)
  + Count the C outcomes, but ignore (reject) samples that don’t have R=true, W=true
* Very inefficient (rejects a lot of samples)



Likelihood Weighting:

* **Idea to deal with Rejection sampling inefficiency:** fix evidence variables, sample the rest
  + **Problem:** sample distribution not consistent
  + **Solution:** weight each sample by probability of evidence variables given parents
* Likelihood weighting is good
  + All samples are used, the values of **downstream** variables are influenced by **upstream** evidence
* **Weakness:**
  + Values of **upstream** variables are unaffected by **downstream** evidence

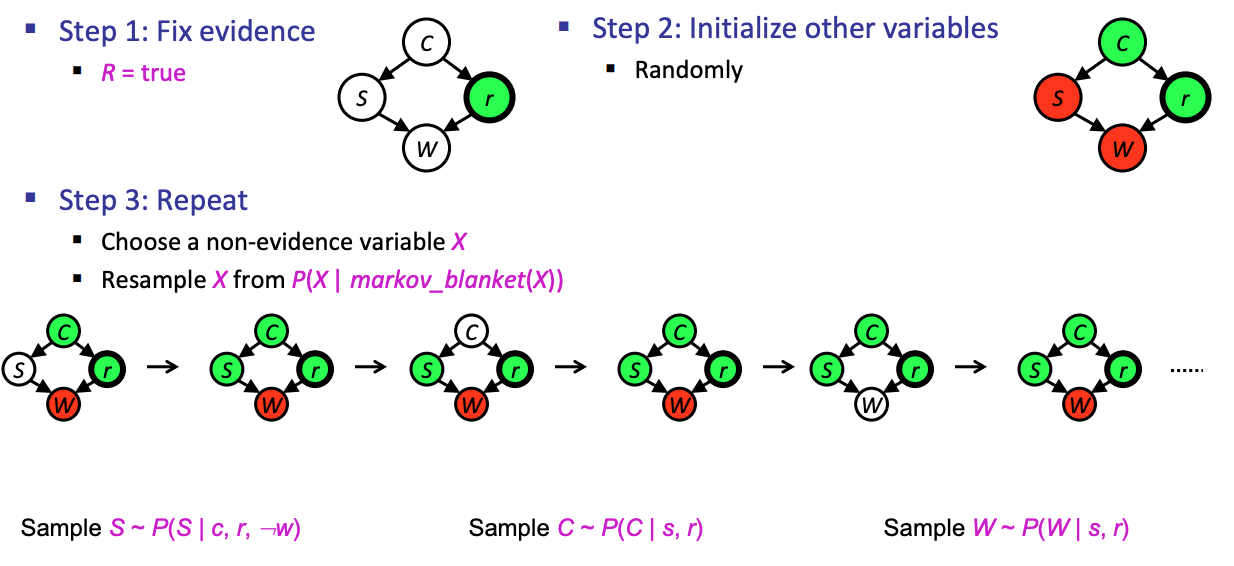
Gibbs Sampling:

* **Markov Chain Monte Carlo**
  + A family of randomized algorithms for algorithms for approximating some quantity of interest over a very large state space
  + **Markov chain**: a sequence of randomly chosen states (“random walk”), where each state is chosen conditioned on the previous state
  + **Monte Carlo**: an algorithm (usually based on sampling) that has some probability of producing an incorrect answer
  + **MCMC** = wander around for a bit, average what you see
* **Gibbs Sampling** is a particular kind of MCMC
  + States are complete assignments to all variables
    - Cf local search: closely related to min-conflicts, simulated annealing
  + Evidence variables remain fixed, other variables change
  + To generate the next state, pick a variable and sample a value for it conditioned on all the other variables (Cf min-conflicts!)



* + - Will tend to move toward states of higher probability, but can go down too
    - **In Bayes Net:**

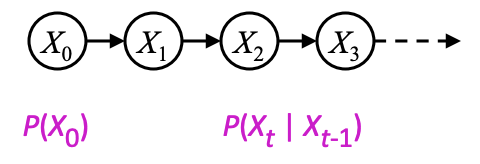




LEC 17: Markov Models

Markov Models (aka Markov Chain):

* Value of X at a given time is called the state (usually discrete, finite)

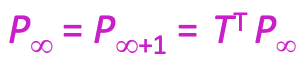


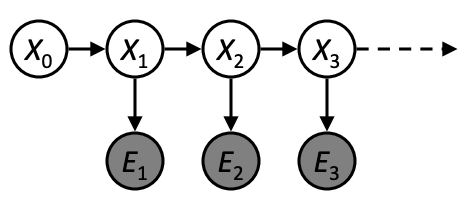
* The **transition model** P(Xt | Xt-1) specifies how the state evolves over time
* **Stationarity** assumption: transition probabilities are the same at all times
* **Markov assumption**: “future is independent of the past given the present”
  + Xt+1 is independent of X0,…, Xt-1 given Xt
  + ****This is a **first-order** Markov model (a kth-order model allows dependencies on k earlier steps)
* **Joint Distribution:**

Forward Algorithm:

* What is the state at time t?
  + P(Xt ) = ∑xt-1 P(Xt ,Xt-1=xt-1)
  + = ∑xt-1 P(Xt-1=xt-1) P(Xt | Xt-1=xt-1)
* Iterate this update starting at t=0

Stationary Distribution:

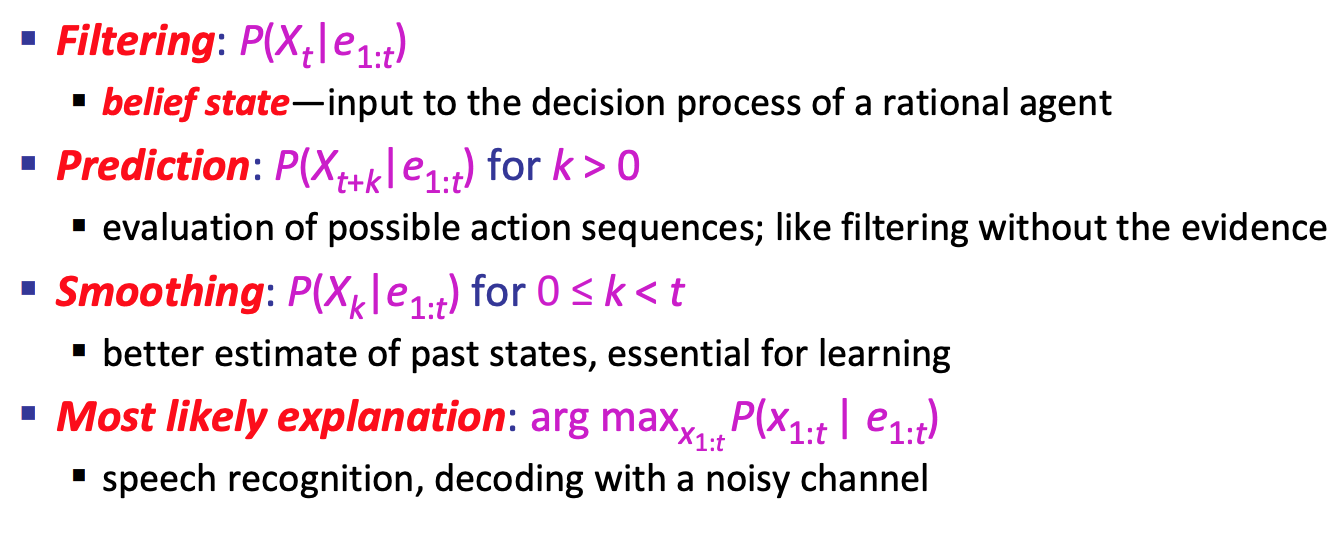
* Satisfies:



Hidden Markov Models:

* Underlying Markov chain over states X
* You observe evidence E at each time step
* X\_t is a single discrete variable; Et may be continuous and may consist of several variables
* Joint distribution for hidden Markov model:
  + Future states still independent of past given the present
  + Current Evidence is independent of everything else given the current state

Inference Tasks:



Filtering / Monitoring:

* Filtering, or monitoring, or state estimation, is the task of maintaining the distribution **f 1:t = P(Xt |e1:t )** over time
* Start with f0 in an initial setting, usually uniform