Modified from Daniel Eckert’s ZFG

## Bayesian Networks

Directed, acyclic grapah with directed edges from (immediate) causes to (immediate) effects. Each vertex is interpreted as a random variable.

Every PDF can be described by a BN (see chain rule).

#### Naïve Bayes

Suppose we have multiple effects with the same cause (e.g. flu causes fever, runny nose, cough, …).

**Assumption:** Effects are conditionally independent given cause

#### Active Trails/D-Separation

**2 RVs are independent if all paths between them are blocked:**

or blocks information if Y is observed

blocks if Y and all its descendants are unobserved

**Active Trail:** Undirected path at which information is not blocked

**d-separation:** If there is *no active trail for observation O, two nodes* are called d-separated by O:

**Linear time alg:** check if   
Find all nodes reachable from X (careful with implem. details)

**1.)** Mark **Z** and its ancestors **2.)** Do breadth-first starting from X; stop if path is blocked; Mem&Time b … branching factor

### Exact Inference (Only exact for trees)

#### Typical Queries

**Marginal:**PDF for RVs in a subset -

**MPE:** Given values for some RVs compute most likely assignment to all remaining RVs

**MAP:**Most likely assignment to some RV

#### Variable Elimination

**Algorithm:** Given BN and Query

Choose ordering ; Set up initial factors:

For i=1:n,

Collect and multiply all factors f that include

Generate new factor by marginalizing out :

Add g to set of factors

Renormalize to get

**Ordering for Polytrees:**Pick root; Orient edges towards root; Eliminate in topological ordering (from outside to inside).

**For non-Polytrees:**Pick subset of variables A (‘cutset’) such that remaining variables form a polytree; Calculate for each cutset; Then

#### Belief Propagation/Factor Graphs

*Msg. from node vfactor u:*

(Multiply the msgs from all neighbor nodes except the target u)

*Factornode:*

(Multiply the messages from all neighbor factors except target v with the factor value, and sum up over all possible values of the RVs that are consistent with )

**Algorithm:** Initialize all messages as uniform distribution (1).

*Until converged:* Pick some ordering on the factor graph edges (+directions); Update messages according to this ordering; Break once all messages change by at most .

*After convergence we have correct values for all marginals:*

v…node

u…factor

**Convergence:** BP converges if graph is acyclic & connected (tree)

### Approximate Inference

**Loopy belief propag.:** In general doesn’t converge (can oscillate). Often overconfident (multiplies same factors multiple times).

#### Sampling based inference

**Monte Carlo Sampling:** Sort variables in topological ordering . For i=1 to n: Sample variables in given order.

Repeat this process N times. (Works even with loopy models)

*Marginals:*

*Conditionals*

**Hoeffding’s inequality** Relative error gets very high for rare events if i.i.d. samples from Bernoulli Dist.

#### Gibbs Sampling

**Gibbs Sampling:** Start with initial assignment to all vars.

Fix observed variables to their observed values. • For t=1 to do: Set • For each unobserved variable , Resample based on all other variables.

**Advantage:**re-sampling only requires multiplying factors containing it (and renormalizing).

## Temporal Models

: Unobserved (hidden) states.

: Observations

#### Bayesian Filtering (HMM)

At time t, assume we have

**Conditioning (Measurement update):** Complexity

**Prediction (Prior update):** Complexity

#### General Kalman Update

•Motion Model: • Sensor Model: • State at time t: •**Kalman Update:** • •**Kalman Gain:**

#### Dynamic Bayesian Networks/Particle Filtering

**Advantage:** Can deal with non-gaussian distributions ( arbitrary) & handle very complex/loopy networks.

*Particles:*

**Predict:** Propagate particl through process model

**Conditioning (Measurement Update):** Weigh particles based on how well they predict the observation:

Resample N particles: (without resampling all weight concentrates on one particle with time)

## Probabilistic Planning

#### Markov Chains

****

**Stationarity/Markov Assumption:** Transition prob. Independ. of t

**Ergodicity:** There exists a finite t such that every state can be reached from every state in exactly t steps.

**Higher-order dependencies:** Can always reduce MC to first order.

 •

### Markov Decision Processes

*MDPControlled Markov chain;* on edges write: a: P(x’|x,a) (r(x,a))

**Specified by:** *States* ; *Actions* ;

*Reward Function:* (average reward for a certain action)

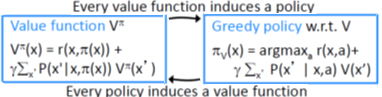
*Transition Probabilities:*

**Discounted Rewards:** infinite horizon, discount future rewards.

Initialize with R=0, and state x. *For t=0 to :*   
Choose action a; Obtain discounted reward ; End up in state according to ; Update

**Deterministic Policy:** ; Induces a Markov Chain with transition prob.:

**Value Function:**(recursive)



**Policy Iteration:** Exact sol.; Complexity per iteration:

Start with an arbitrary policy . Until converged, do:

Compute ; Compute greedy policy w.r.t. ; Set

Monotonically converges to an optimal policy in iter

*Bellman Theorem:* Policy optimalgreedy w.r.t its induced value function

**Value Iteration:** -optimal sol.; Complexity per iteration:

Initialize ; For to do:

For each x, a:

For each x:

Break if

Then choose greedy policy w.r.t (guaranteed to converge!)

### POMDP (Controlled HMM)

Have only noisy observations of the hidden state

**Idea:** Interpret POMDP as MDP. New states correspond to beliefs in original POMDP

**Belief State MDP:** *States:* Beliefs over states for original POMDP

*Actions:* Same as original MDP • *Transition Model:*

Stochastic observations

State update (Bayesian Filtering) - Given :

*Reward function:*

**Policy Gradient Method** parametric form of policy

For each parameter the policy induces a Markov chain. Can compute expected reward by sampling. Find optimal parameters through search (gradient ascend)

## Learning

### Bayesian Net Learning

**Parameter Learning:** Given net structure G and Data set D of complete observ. globally optimal MLE, Requires complete data

*For each RV estimate:*

*Pseudo counts:*To deal with missing data, assume that we’ve seen a number of occurrences.

**Structure Learning:** Scoring Function quantifies for each structure G the fit to the data D.

*Use Maximum Likelihood to score BN:* •Measure of dependence between 2 RV’s:

• indep => I = 0 •Fully deterministic => I = Entropy(X) •  •••

*Problem:* Optimal solution is always the fully connected graph

Bayesian information criterion: ‘Prefer simpler models’  
• •Finds corr. structure (consistent) for N -> inf

(n=#RV’s, |G|=#Param(G), N=#Training Ex.)

### Reinforcement Learning

Learning MDP by obs./estimating state transitions & rewards

#### Model-based RL

*Data Set:*

*Estimate transitions:*

*Estimate rewards:*

**greedy:** Solution to Exploration-Exploitation Dilemma.

With probability… Pick random action; ( pick best

**Algorithm:** *Fairy Tale state:*

*Input:* Starting state , discount factor

*Initially:* Add fairy tale state to MDP; Set for all states x and actions a; Set for all states x and actions a; Choose optimal policy for r and P

*Repeat:* Execute policy ; For each visited state action pair x, a, update ; Estimate transition probabilitfies ; If observed “enough” transitions/rewards, recompute policy according to current model P and r

**Problem w. model based RL:** High comput./memory demand.

#### Model-free RL (Q-learning)

Suppose we have initial estimate and observe transition (cur, action, next) = (x, a, x‘) with reward r. Then:

where

*Theorem:* If , and a’s are chosen at random, then Qlearning converges to optimal

*Theorem:* With prob., optimistic Q-learn. obtains -optimal policy after #time steps that is pol. In |X|, |A|, ,

• Mem C: (Indep of # states)

#### C:\Users\stege\AppData\Local\Microsoft\Windows\INetCacheContent.Word\screenshot-20170205-155352.pngHeuristic Search

**Reinforce Algorithm:** Input: ; **1.)** Init policy weights **2.)** Repeat: **a)** Generate an episode (rollout) **b)** For t=1,…T: Set to the return from step t Update :

**Monte Carlo Esimate as Surrogate:**

## **Probability**

Sum rule

Prod. Rule

Bayes

Indep. RV • • • • if P() > 0

**Bayes rule:**

**Law of large numbers:**

#### Gaussians

**Gaussian:** =std. dev

**Multivariate Gaussian:**

**Conditional Distributions:** For two Gaussian RV’s and

and