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. $P(X_1, \dots, X_n) = \prod_i P(X_i | \boldsymbol{Pa}_{X_i})$

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:

$X \to Y \to Z$ or $X \leftarrow Y \to Z$ blocks information if Y is observed

$X \to Y \leftarrow Z$ 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: $dsep(A; B|O) \Rightarrow A \perp B | O$

Linear time alg: check if $dsep(X; Y|Z)$

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 $O(b^{d+1})$ b … branching factor

Exact Inference (Only exact for trees)

Typical Queries

Marginal: PDF for RVs in a subset - $P(E|J = T) = \frac{1}{Z} P(E, J = T)$

MPE: Given values for some RVs compute most likely assignment to all remaining RVs $argmax_{e,b,a} P(e, b, a | J = T, M = F)$

MAP: Most likely assignment to some RV

$argmax_{e,b} P(e, b | J = T)$

Variable Elimination

Algorithm: Given BN and Query $P(X|E = e)$

Choose ordering $X_1, \dots, X_n$; Set up initial factors: $f_i = P(X_i | Pa_i)$

For i=1:n, $X_i \notin \{X, E\}$

  Collect and multiply all factors f that include X_i

  Generate new factor by marginalizing out X_i: $g = \sum_{x_i} \prod_j f_j$

  Add g to set of factors

Renormalize $P(x, e)$ to get $P(x|e)$

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 $P(X_i, A = a | E = e)$ for each cutset; Then $P(X_i | E = e) = \sum_a P(X_i, A = a | E = e)$

Belief Propagation/Factor Graphs

Msg. from node v→factor u: $\mu_{v \to u}(x_v) = \prod_{u' \in N(v) \setminus \{u\}} \mu_{u' \to v}(x_v)$

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

Factor→node: $\mu_{u \to v}(x_v) = \sum_{x_u \sim x_v} f_u(x_u) \prod_{v' \in N(u) \setminus \{v\}} \mu_{v' \to u}(x_{v'})$

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

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 ϵ.

After convergence we have correct values for all marginals:

$P(X_v = x_v) \propto \prod_{u \in N(v)} \mu_{u \to v}(x_v)$ v…node

$P(X_u = x_u) \propto f_u(x_u) \prod_{v \in N(u)} \mu_{v \to u}(x_v)$ 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 $X_1, \dots, X_n$. For i=1 to n: Sample variables in given order.

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

Marginals: $P(X_i = T) = Count(X_i = T)/N = \hat{P}(X_i = T)$

Conditionals $P(X_i = T | X_j = T) = \frac{P(X_i = T, X_j = T)}{P(X_j = T)} = \frac{Count(X_i = T, X_j = T)}{Count(X_j = T)}$

Hoeffding's inequality Relative error gets very high for rare events

$P\left(\left|\mu - \frac{1}{n} \sum_i X_i\right| \geq \epsilon\right) \leq 2\exp(-2n\epsilon^2) = \rho$ if $X_1, \dots, X_n$ i.i.d. samples from Bernoulli Dist.

Gibbs Sampling

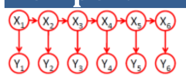
Gibbs Sampling: Start with initial assignment $x^{(0)}$ to all vars.

Fix observed variables to their observed values. • For t=1 to ∞ do:

Set $x^{(t)} = x^{(t-1)}$ • For each unobserved variable X_i, Resample $x_i^{(t)} \sim P(X_i | \boldsymbol{v}_i) = \frac{1}{Z} \prod_{j: i \in f_j} f_j(\boldsymbol{v}_i)$ based on all other variables.

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

Temporal Models



$X_1, \dots, X_T$: Unobserved (hidden) states.

$Y_1, \dots, Y_T$: Observations

Bayesian Filtering (HMM)

At time t, assume we have $P(X_t | Y_{1 \dots t-1})$

Conditioning (Measurement update): Complexity $O(k)$

$$P(X_t | Y_{1:t}) = 1/Z \underbrace{P(X_t | Y_{1:t-1})}_{prior} \underbrace{P(Y_t | X_t)}_{measurement\ model}$$

Prediction (Prior update): Complexity $O(k^2)$

$$P(X_{t+1} | Y_{1:t}) = \sum_{x_t} P(X_t | Y_{1:t}) \underbrace{P(X_{t+1} | X_t)}_{Process\ Model}$$

General Kalman Update

•Motion Model: $P(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t) = N(\boldsymbol{x}_{t+1}; \boldsymbol{F}\boldsymbol{x}_t, \Sigma_x)$ • Sensor Model: $P(\boldsymbol{y}_t | \boldsymbol{x}_t) = N(\boldsymbol{y}_t; \boldsymbol{H}\boldsymbol{x}_t, \Sigma_y)$ • State at time t: $P(\boldsymbol{x}_t | \boldsymbol{y}_{1:t}) = N(\mu_t, \sigma_t^2)$ •**Kalman Update:** $\mu_{t+1} = \boldsymbol{F}\mu_t + \boldsymbol{K}_{t+1}(\boldsymbol{y}_{t+1} - \boldsymbol{H}\boldsymbol{F}\mu_t)$ • $\Sigma_{t+1} = (\boldsymbol{I} - \boldsymbol{K}_{t+1})(\boldsymbol{F}\Sigma_t \boldsymbol{F}^T + \Sigma_x)$ •**Kalman Gain:** $\boldsymbol{K}_{t+1} = (\boldsymbol{F}\Sigma_t \boldsymbol{F}^T + \Sigma_x)\boldsymbol{H}^T \left(\boldsymbol{H}(\boldsymbol{F}\Sigma_t \boldsymbol{F}^T + \Sigma_x)\boldsymbol{H}^T + \Sigma_y\right)^{-1}$

Dynamic Bayesian Networks/Particle Filtering

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

Particles: $\delta_x(x') = \begin{cases} 1 & if\ x = x' \\ 0 & otherways \end{cases}$ $P(X_t | y_{1:t}) \approx \frac{1}{N} \sum_{i=1}^N \delta_{x_{i,t}}$

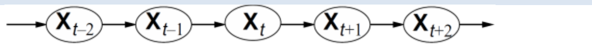
Predict: Propagate particl through process model $x_i' \sim P(X_{t+1} | x_{i,t})$

Conditioning (Measurement Update): Weigh particles based on how well they predict the observation: $w_i = 1/Z\ P(y_{t+1} | x_i')$

Resample N particles: $x_{i,t+1} \sim \sum_{i=1}^N w_i \delta_{x_i'}$ (without resampling all weight concentrates on one particle with time)

Probabilistic Planning

Markov Chains



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

$P(X_{t+1} = x | X_t = x') = P(X_{t'+1} = x | X_{t'} = x') \forall t, t'$

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.

$Z_t = [X_{t-1}, X_t] \in DxD$ • $\frac{1}{Z} Q(\boldsymbol{x}) P(\boldsymbol{x}' | \boldsymbol{x}) = \frac{1}{Z} Q(\boldsymbol{x}') P(\boldsymbol{x} | \boldsymbol{x}')$

Markov Decision Processes

MDP≙Controlled Markov chain; on edges write: a: P(x'|x,a) (r(x,a))

Specified by: *States* $X = \{1, \dots, n\}$; *Actions* $A = \{1, \dots, m\}$;

Reward Function: $r(x, a)$ (average reward for a certain action)

Transition Probabilities:

$P(x' | x, a) = Prob(next\ state = x' | Action\ a\ in\ state\ x)$

Discounted Rewards: infinite horizon, discount future rewards.

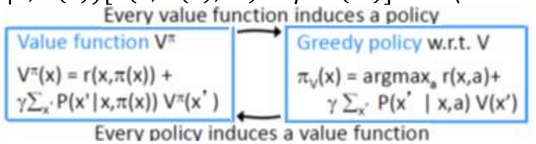
Initialize with R=0, and state x. *For t=0 to ∞:*

Choose action a; Obtain discounted reward $R = R + \gamma^t r(x, a)$;

End up in state x' according to $P(x' | x, a)$; Update $x \leftarrow x'$

Deterministic Policy: $\pi: X \to A$; Induces a Markov Chain with transition prob.: $P(X_{t+1} = x' | X_t = x) = P(x' | x, \pi(x))$

Value Function: $V^\pi(x) = \mathbb{E}[\sum_{(t=0)}^{\infty} \gamma^t r(X_t, \pi(X_t)) | X_0 = x] = \sum_{x'} P(x'|x,\pi(x))[r(x,\pi(x),x') + \gamma V^\pi(x')]$ (recursive)



Policy Iteration: Exact sol.; Complexity per iteration: $n^3 + nm\Delta$

Start with an arbitrary policy π. Until converged, do:

Compute $V^\pi(x)$; Compute greedy policy π_G w.r.t. V^π; Set $\pi \leftarrow \pi_G$

→Monotonically converges to an optimal policy π^* in $O^*(\frac{n^2 m}{1-\gamma})$ iter

Bellman Theorem: Policy optimal⇔greedy w.r.t its induced value function $V^*(x) = \max_a[r(x,a) + \gamma\sum_{x'} P(x'|x,a)V^*(x')]$

Value Iteration: ϵ-optimal sol.; Complexity per iteration: $nm\Delta$

Initialize $V_0(x) = max_a\ r(x,a)$; For $t = 1$ to ∞ do:

For each x, a: $Q_t(x,a) = \sum_{x'} P(x'|x,a)[r(x,a,x') + \gamma V_{t-1}(x')]$

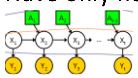
For each x: $V_t(x) = max_a\ Q_t(x,a)$

Break if $\sum_x |V_t(x) - V_{t-1}(x)| < \epsilon$

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

POMDP (Controlled HMM)

Have only noisy observations Y_t of the hidden state X_t



Idea: Interpret POMDP as MDP. New states correspond to beliefs $P(X_t|y_{1:t})$ in original POMDP

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

$\mathcal{B} = \Delta(\underbrace{\{1,\dots,n\}}_{X}) = \{b: \{1,\dots,n\} \to [0,1], \sum_x b(x) = 1\}$

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

Stochastic observations $P(Y_t|b_t) = \sum_{x=1}^n P(Y_t|X_t = x) b_t(x)$

State update (Bayesian Filtering) - Given b_t, y_t, a_t:

$$b_{t+1}(x') = \frac{1}{Z}\sum_x b_t(x) P(y_t|x) P(X_{t+1} = x'|X_t = x, a_t)$$

Reward function: $r(b_t, a_t) = \sum_x b_t(x) r(x, a_t)$

Policy Gradient Method parametric form of policy $\pi(b) = \pi(b;\theta)$

For each parameter θ the policy induces a Markov chain. Can compute expected reward $J(\theta)$ by sampling. Find optimal parameters through search (gradient ascend) $\theta^* = \arg\max_\theta J(\theta)$

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 X_i estimate: $\hat\theta_{X_i|Pa_i} = Count(X_i, Pa_i)/Count(Pa_i)$

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

$logP(D|\theta_G, G) = N\sum_{i=1}^n \hat I(X_i; \boldsymbol{Pa}_i) + const.$

Structure Learning: Scoring Function $S(G;D)$ quantifies for each structure G the fit to the data D. ⇒ $G^* = \arg\max_G S(G;D)$

Use Maximum Likelihood to score BN: $S(G;D) = max log_\theta P(D|\theta, G)$ •Measure of dependence between 2 RV's:

$\hat I(X_i, X_j) = \sum_{x_i,x_j} \hat P(x_i,x_j) \log\frac{\hat P(x_i,x_j)}{\hat P(x_i)\hat P(x_j)}$ $\quad \hat P(x_i,x_j) = \frac{Count(x_i,x_j)}{N}$

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

• $I(X_a, X_B) = I(X_B, X_A)$•$B$ $suset(C)$: $I(X_A; X_B) \le I(X_A; X_C)$•$I(X_A; X_B) = H(X_A) - H(X_A|X_B)$ •$H(X_I) = -\sum_{X_i} P(x_i) log P(x_i)$

Problem: Optimal solution is always the fully connected graph

→ Bayesian information criterion: 'Prefer simpler models'

•$S_{BIC}(G;D) = \sum_{i=1}^n \hat I(X_i; Pa_i) - \frac{\log N}{2N}|G|$ •Finds corr. structure (consistent) for N -> inf

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

$n = \#RV's, |G| = \#Param(G), N = \#Training\ Examples$

Reinforcement Learning

Learning MDP by obs./estimating state transitions & rewards

Model-based RL

Data Set: $D = \{(x_1, a_1, r_1, x_2), \dots, (x_n, a_n, r_n, x_{n+1})\}$

Estimate transitions: $\hat P(X_{t+1}|X_t, A_t) = \frac{Count(X_t, A_t, X_{t+1})}{Count(X_t, A_t)}$

Estimate rewards: $\hat r(X_t = x, A_t = a) = \frac{\sum_{t: X_t = x, a_t = a} r_t}{Count(X_t = x, A_t = a)}$

ϵ_t greedy: Solution to Exploration-Exploitation Dilemma.

With probability… ϵ_t Pick random action; $(1-\epsilon_t)$ pick best

R_{max} Algorithm: *Fairy Tale state:* $\forall a: r(x^*, a) = R_{max}$
$P(X_{t+1} = x^*|X_t = x^*, a) = 1$

Input: Starting state x_0, discount factor γ

Initially: Add fairy tale state x^* to MDP; Set $r(x,a) = R_{max}$ for all states x and actions a; Set $P(x^*|x,a) = 1$ for all states x and actions a; Choose optimal policy for r and P

Repeat: Execute policy π; For each visited state action pair x, a, update $r(x,a)$; Estimate transition probabilitfies $P(x'|x,a)$; If observed "enough" transitions/rewards, recompute policy π 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 $Q_0(x,a)$ and observe transition (cur, action, next) = (x, a, x') with reward r. Then:

$$V^*(x) = \max_a Q^*(x,a)$$
where $Q^*(x,a) = r(x,a) + \gamma\sum_{x'} P(x'|x,a)V^*(x')$
$$Q_{t+1}(x,a) = \alpha_t[r + \gamma\max_{a'} Q_t(x',a')] + (1-\alpha_t)Q_t(x,a)$$

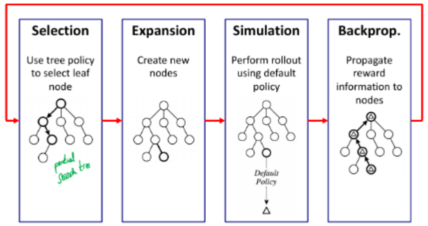
Theorem: If $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$ and a's are chosen at random, then Qlearning converges to optimal Q^*

Theorem: With prob.$1-\delta$, optimistic Q-learn. obtains ϵ-optimal policy after #time steps that is pol. In |X|, |A|, $1/\epsilon$, $\log(1/\rho)$

• Mem $O(|X||A|)$C: $sel\ |A|, update\ O(|A|)$, (Indep of # states)

Heuristic Search

$MSVE(\boldsymbol\theta) = \sum_{s\in X} d(s)\left(V^\pi(s) - \hat V(s;\boldsymbol\theta)\right)^2$



Reinforce Algorithm:

Input: $\pi(a|s;\theta)$; **1.)** Init policy weights θ **2.)** Repeat: **a)** Generate an episode (rollout) $S_0, A_0, R_0, S_1, A_1, R_1, \dots, S_T, A_T, R_T$ **b)** For t=1,…T: Set $G_t$to the return from step t Update $\boldsymbol\theta$: $\boldsymbol\theta = \boldsymbol\theta + \boldsymbol{\alpha\gamma^t} G_t \Delta_{\boldsymbol\theta}\log(\pi(A_t|S_t;\theta))$

Monte Carlo Esimate as Surrogate: $(\mathbf{1}.)init\ \boldsymbol\theta\ (eg = 0)(\mathbf{2a}.)Generate\ S_0, A_0, R_0, \dots, S_T, A_T, R_T\ (\mathbf{2b})For\ t = 1..T: G_0 = R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots; G_1 = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots; Update\ \theta = \theta + \alpha[G_t - \hat V(S_t;\theta]\nabla\hat V(S_t;\theta)$

Probability

Sum rule $P(X_i) = \sum_{x_1,\dots,x_{i-1},x_{i+1},\dots,x_n} P(x_1,\dots,x_{i-1},X_i,x_{i+1},\dots,x_n)$

Prod. Rule $P(X_1,\dots,X_n) = P(X_1)P(X_2|X_1)\dots P(X_n|X_1,\dots,X_{n-1})$

Bayes $P(X|Y) = P(X)P(Y|X)/\sum_{X=x} P(X=x)P(Y|X=x)$

Indep. RV $X \perp Y|Z \Rightarrow Y \perp X|Z$ • $X \perp Y, W|Z \Rightarrow X \perp Y|Z$ • $(X \perp Y|Z) \wedge (X \perp W|Y,Z) \Rightarrow X \perp Y, W|Z$ • $X \perp Y, W|Z \Rightarrow X \perp Y|Z, W$• $(X \perp Y|W,Z) \wedge (X \perp W|Y,Z) \Rightarrow X \perp Y, W|Z$ if P() > 0

Bayes rule: $P(C|T) = \frac{P(C)P(T|C)}{P(T)} = \frac{P(C)P(T|C)}{\sum_c P(C=c)P(T|C=c)}$

Law of large numbers: $\mathbb{E}_P[f(X)] = lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^N f(x_i)$

Gaussians

Gaussian: $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$ $\quad\sigma$=std. dev

Multivariate Gaussian: $\Sigma = \begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{21} & \sigma_2^2\end{pmatrix}$ $\quad \mu = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}$

$$p(\vec x) = \frac{1}{2\pi\sqrt{|\Sigma|}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

Conditional Distributions: For two Gaussian RV's X_A and X_B
$$P(X_A|X_B = x_B) = N(\mu_{A|B}, \Sigma_{A|B})$$

$\mu_{A|B} = \mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B)$ and $\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$