T	S	P
warm	sun	0.4
warm	rain	0.1
cold	sun	0.2
cold	rain	0.3

Select $\rightarrow$

$P(T,r)$

T	P
warm	0.1
cold	0.3

Normalize $\rightarrow$

$P(T|r)$

T	P
warm	0.25
cold	0.75

ALWAYS REMEMBER TO NORMALISE

Inference by enumeration. Space Complexity: O(d^n) to store the joint distribution. Worst case time complexity: O(d^n)

2. INDEPENDENCE

Two variables are independent iff for all values in the domain of X and Y.

P(X|Y) = P(X) and vversa. Helps reduce complexty more instead of full joint.

X is *conditionally independent* of Y given Z iff for all values in the domain of X, Y and Z:

$$P(X|Y,Z) = P(X|Z)$$
$$P(X,Y|Z) = P(X|Z)P(Y|Z)$$

CPT: conditional probability table for each node.

2.5 CHAIN RULE

Obtain the joint prob by multiplying the conditional probabilities by chain

Generalization of product rule P(X1, X2, X3,..Xn)

= P(X1) * P(X2 |X1) P(X3 |X2, X1)*...*(Xn |Xn-1…X1)

Bayes Rule P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X)

Big-O notation tells us how algorithm's run-time behave as input size n asymptotically approaches infinity

BIG O ABSTRACTS out constant factors, only keeps the complexity

And ONLY KEEP MOST DOMINANT TIME FACTOR

Querying in Bayesian Networks using joint by chain rule all the CPTs.

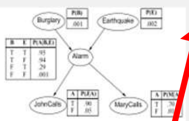
X is independent of all the non-descendants, given all its parents (Ui). It is conditionally independent of Zs given its parents.

Note also that when summed across a row, probability is 1. Observe

Σ_M P(M|A=T) = P(M=T|A=T)+P(M=F|A=T)=1;

for given fixed A value

$P(b,e,j) = ?$ Note that A, M are hidden variables



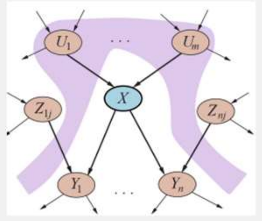
$$= \sum_{A,M} P(b,e,A,j,M)$$

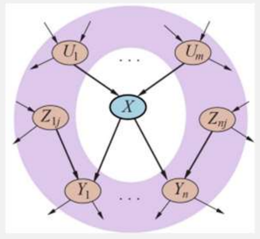
$$= \sum_{A,M} P(b)P(e)P(A|b,e)P(j|A)P(M|A) = P(b)P(e)\sum_{A,M} P(A|b,e)P(j|A)P(M|A)$$

$$= P(b)P(e)\sum_{A,} P(A|b,e)P(j|A)\sum_{M} P(M|A) = P(b)P(e)\sum_{A} P(A|b,e)P(j|A)$$

$$= P(b)P(e)[P(a|b,e)p(j|a) + P(\neg a|b,e)P(j|\neg a)]$$

Activate Windows





CPT of Node B

A	P(B|A)
a_1	$\theta_B(b_1\|a_1)$
a_2	$\theta_B(b_1\|a_2)$

4. BN DSEP

For Markov Blanket rules, variable is independent of other nodes given its direct Markov Blanket -> Parents + Children + Parent of same children. if two nodes are d-seperated, they are independent

PATH CASE CHECKS. Find P(A,C) vs P(A,C |B).

Sum out unseen nodes not wanted in query.

D-separation does not guarantee dependence as they can be numerically independent

Analyze the graph by breaking down all the paths connecting the two variables into the 3 cases we have seen so far. If all possible paths are cut off or inactive, the two nodes are called dseparated or independent

MORALISATION

1.Keep ALL ancestor or direct parents of all variables in concern.

2.For each pair of variables with common child, draw a line between them, between the parents. If there are more than two parents, draw edge between each one of them 3.Delete Z (evidence nodes) and its edges.

Independent is guaranteed if: ▸ X and Y are disconnected completely. No possible path to trace.

5. MLE Finds the set of parameters (Θ) that maximise the likelihood (probability) of seeing the data D. argmax L(Θ) wrt = argmax P(D|Θ)

Obtain the likelihood using the assumption that observations are independent and identically distributed Prob of R.V seeing the value and the times it happens, all independently and multiplied together.

$$\text{argmax}_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta}) + \lambda\left(\sum_{i=1}^{m} \theta_i - 1\right)$$

$$\sum^{\text{domain } d} \Theta_i^{ni}$$

2. To find the value that maximise L(Θ), apply log transformation. //+lagrange

3. Use the probability constraint of all probs sum =1 and sub it in.

4. Obtain derivatives of L(Θi) with respect to Θi and equate to 0

If many probability parameters Θn, use LAGRANGE.

Generically the solution is -> Θi = ni / total n over all ni.

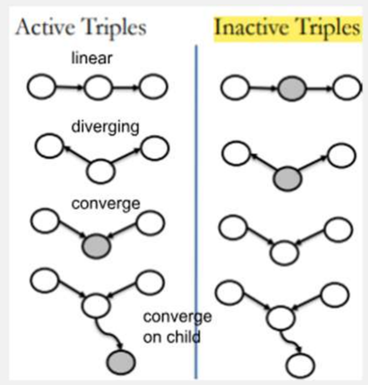
FOR CPT MLE estimate, include all parameters within a CPT for each given probability. **Be careful of whether its full or least params CPT.**

need to write out the entire log likelihood function across row and down all columns to get all the parameters. **∑∑log(Θ(bi | ai) =**

$$l(\theta_B) = n_{BA}(b1,a1)\cdot log\theta_B(\text{b1}|\text{a1}) + n_{BA}(b2,a1)\cdot log\theta_B(\text{b2}|\text{a1}) + n_{BA}(b1,a2)\cdot log\theta_B(\text{b1}|\text{a2}) + n_{BA}(b2,a2)\cdot log\theta_B(\text{b2}|\text{a2})$$

Laplace Smoothing/Add-one Smoothing. **Add λ= 1 to all the freq counts**

Higher values of λ pushes the distribution towards uniform! λ values diminishes with more data and higher counts. ***Remember that plus 1 affects the DENOMINATOR ALSO***



B	#	P(B)
b_1	4+1	5/7
b_2	1+1	2/7

B	D	#	P(D|B)
b_1	d_1	1+1	2/6
b_1	d_2	3+1	4/6
b_2	d_1	1+1	2/3
b_2	d_2	0+1	1/3

6. Markov Models —- Value or the state of a random variable (Xt) in each time step only depends on the previous time step.

p(Xt+1 | Xt … X2 X1) = only P(X+1 | Xt) **Stationarity assumption: transition probabilities between the states are always the same**

Joint distribution for a basic markov chain Y1->Y2->...->Yn

P(Y1, Y2, …Yn) = P(Y1) ∘ ∏ P(Yi | Yi-1)

For markov, need INITIAL STATE PROB, TRANSITION PROBs

Probability distribution at any point itself like P(X3) can be derived

$$\begin{aligned} &Example \ -> \ P(V3 = h) = \Sigma_{V2}\, P(V3 = h, V2) \\ &= \Sigma_{V2}\, P(V3 = h \mid v2) \circ P(v2) \\ &= [P3(h\,|h) + P3(h \mid l)] \circ (\Sigma_{V1}\, P(v2 \mid v1)) \circ P1(v1) \\ &= P3(h|h) * [\Sigma_{V1} P(v2 = h|v1) * P(v1)] \\ &\quad + P3(h|l) * [\Sigma_{V1} P(v2 = h|v1) * P(v1)] \end{aligned}$$

For majority of the markov chains, the stationary distribution is independent of the initial distribution. Satisfies distribution at infinity. Remember sub back prob constraints = 1.

Infinity stationary probs →NOT depend on initial state probs,Only dep on transit probs

$P_\infty(X) = P_{\infty+1}(X)$

$P_\infty(X) = \Sigma_x P(X|x)P_\infty(x)$ (eq. I)

$\Sigma_x P_\infty(x) = 1$ (eq. II)

$$P_{\infty+1}(V) = \Sigma_v P(V|v)P_\infty(v) = P_\infty(V)$$

CPT assumed to not change

$P_\infty(high) = P(high|high)P_\infty(high) + P(high|low)P_\infty(low)$

▸ $P_\infty(high) = 0.9 * P_\infty(high) + 0.3 * P_\infty(low)$

2. $\Sigma_v P_\infty(V) = 1 \Rightarrow P_\infty(low) + P_\infty(high) = 1$

$\Sigma_{t=0}^{\infty} \gamma^t R(s_t)$

12. REINFORCE LEARNING write out all T(S, A, S") & R(S,A, S")

The utility function over a path s0,s1,s2… gives the sum of discounted rewards

Goal to learn Optimal policy (*π(s)): a function that specifies the action to take in state s that results in the highest expected utility.

Essential -> argmax choose highest util action pick the action that gives the best util outcome. for EACH ACTION write sum all probs and the outcomes of actions, since action can have more than one outcome + reward

*Can be like **Find all iterations 1 or more V1*(S={0,1,2,3}) -> find for each iteration each V1*(S) for all states.***

For next iterations, the V2* will use V1*(s) for action that results in state s

▸ $V_i^*(s)$ is the estimate of $V^*(s)$ in iteration i

1. Start with $V_0^*(s) = 0 \quad \forall s \in S$
2. Compute V_{i+1}^* given V_i^*:

$$V_{i+1}^*(s) \leftarrow \max_a \Sigma_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i^*(s')] \quad \textit{(Bellman equation)}$$

3. Repeat until convergence
   ▸ i.e., $V_{i+1}^*(s)$ is very close to $V_i^*(s)\ \forall s$

- For true optimal V^*, $V^* = R(s) + \gamma V^*$
- Subtracting the above two equations:

$$V_{i+1}^* - V^* = \gamma(V_i^* - V^*)\ (\gamma < 1)$$

- Distance between the estimated and optimal would decrease by a factor γ for each iteration

7.HMM includes evidences into Markov
three types of parameters for HMM. Initial P(X1). Transition P(Xi |Xi-1). Emission P(Ei |Xi). Evidence observation independent of other evidences. (DEPENDS ON STRUCTURE OF HMM VARIATION)

$$P(X_1,\dots,X_n,Y_0,Y_1,\dots,Y_n,Y_{n+1})$$
$$= P(Y_0,Y_1,\dots,Y_n,Y_{n+1})P(X_1,\dots,X_n|Y_0,Y_1,\dots,Y_n,Y_{n+1})$$
$$= P(Y_0)\prod_{i=1}^{n+1}P(Y_i|Y_{i-1})\prod_{i=1}^{n}P(X_i|Y_i)$$

$P(Y_i|Y_{i-1})$: transition distribution
$P(X_i|Y_i)$: output/emission/observation distribution

Based on counting. a_ij is the times a for **value i transit to value j** for a random variable. Number of times value i appears **does not require it to start at value i, just seeing how many times IT APPEARS.**

$$a_{i->j} = \frac{count(i->j)}{count(i)},\ s.t\ count(i)\ is\ just\ total\ number\ of\ times\ value\ i\ appear,$$
$$b_i(o) = \frac{count(i->o)}{count(i)}$$

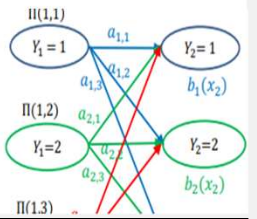
8. VITERBI Algo. for (y1, y_n) sequences, N hidden states -> $O(N^n)$
N possible sequences, n sequence lengths

$$\Pi(t,1) = \max_{i=\{1,2,3\}}\Pi(t-1,i)a_{i,1}b_1(x_t)$$

- Recursive Equation!

E.g., For $k = 2$ and three states

- $\Pi(2,1) =$
  $\max(\Pi(1,1)a_{1,1}b_1(x_2), \Pi(1,2)a_{2,1}b_1(x_2), \Pi(1,3)a_{3,1}b_1(x_2))$
- $= \max_{i=\{1,2,3\}}\Pi(1,i)a_{i,1}b_1(x_2)$
- $\Pi(2,2) =$
  $\max(\Pi(1,1)a_{1,2}b_2(x_2), \Pi(1,2)a_{2,2}b_2(x_2), \Pi(1,3)a_{3,2}b_2(x_2))$
- $= \max_{i=\{1,2,3\}}\Pi(1,i)a_{i,2}b_2(x_2)$



Use $\Pi(k,v)$ to represent the max probability over all length k sequences ending in state v

Base case:

- $\Pi(1,v) = a_{0,v}b_v(x_1)$

Also MUST ***Backpointers*** keep track of the t-1 hidden state value that gives max prob

Recursive step (moving forward):

- for $k \in \{2,\dots,n\}$:
  - for $v \in \{1,\dots,N-1\}$:
    $$\Pi(k,v) = \max_{u\in\{1,\dots,N-1\}}\{\Pi(k-1,u)\ a_{u,v}\ b_v(x_k)\}$$

Last step: $\max_{v\in\{1,\dots,N-1\}}\{\Pi(n,v)a_{v,STOP}\}$

$$= \max_{y_1 y_2 y_3,\dots,y_n} P(Y_1)P(x_1|Y_1)P(Y_2|Y_1)P(x_2|Y_2)\dots P(Y_n|Y_{n-1})P(X_n|Y_n)\,P(Y_{n+1}|Y_n)$$
$$\Pi(1,Y_1=v) = P(Y_1=v|Y_0)P(x_1|Y_1=v)\ \text{(base case)}$$
$$\Pi(k,Y_k=v) = \max_{y_{k-1}}\Pi(k-1,Y_{k-1})P(Y_k=v|Y_{k-1})P(x_k|Y_k=v)$$
(recurrence equation)

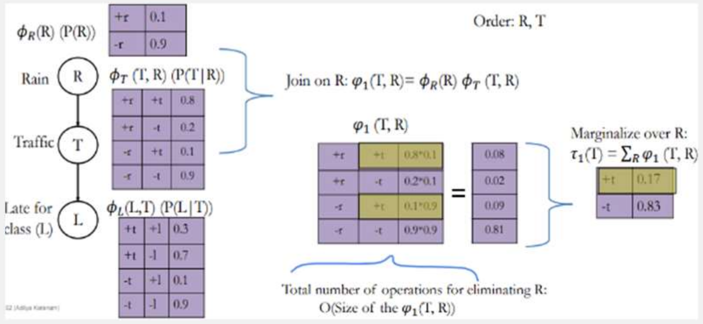
9. Variable elimination for networks, By grouping and transforming, compute lesser so we compute once and reuse

$$P(D) = \textstyle\sum_C\sum_B\sum_A P(D|C)P(C|B)P(A)P(B|A)$$
$$= \textstyle\sum_C P(D|C)\underbrace{\textstyle\sum_B P(C|B)\underbrace{\textstyle\sum_A P(A)P(B|A)}_{\tau_1(B)}}_{\tau_2(C)}$$

Each factor has a scope ▸ all Random variables included in the factor for CPT. 1. Here you see for P(C|E,F) the factor ϕC|EF contains scope C,E,F → 2. Ψ(X,Y) = ϕX * ϕY. Scope Ψ is union of scope of all multiplied factors.
3. τ (X) = sumout Y Σ Ψ(X,Y). Scope τ (X) is exclude eliminated var. Need not be valid prob distributions

- Eliminate D $\quad \Phi^1 = \{\phi_L(L,G), \phi_s(S,I), \phi_I(I), \phi_G(G,D,I), \phi_D(D)\}$
  - Join on D: $\varphi_1(G,D,I) = \phi_G(G,D,I)\,\phi_D(D)$
  - Sum out D: $\tau_1(G,I) = \sum_D \varphi(G,D,I)$
  - Update remaining factors: $\Phi^2 = \{\phi_L(L,G), \phi_s(S,I), \phi_I(I), \cancel{\phi_G(G,D,I), \phi_D(D)}, \tau_1(G,I)\}$

Eg Order of elimination maybe given//selected, compared to find fastest way of eliminating.After elim allthe variables***, normalise final table ϕfinal***.



Order: R, T
Rain
R
Traffic
T
Late for class (L)
L
Join on R: φ₁(T, R)= ϕR(R) ϕT (T, R)
Marginalize over R: τ₁(T) = Σ_R φ₁ (T, R)
Total number of operations for eliminating R: O(Size of the φ₁(T, R))

$\phi_R(R)$ (P(R))

+r	0.1
-r	0.9

ϕ_T (T, R) (P(T|R))

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

ϕ_L(L,T) (P(L|T))

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

φ_1 (T, R)

+r	+t	0.8*0.1	0.08
+r	-t	0.2*0.1	0.02
-r	+t	0.1*0.9	0.09
-r	-t	0.9*0.9	0.81

+t	0.17
-t	0.83

avoid exponential complexity or have m <<< n ? Dep on order of elimination, largest factor during elim? depend on largest table for interm factor found ϕ.
VE complexity = O(n*Nmax), Nmax = k^m , m vars with k values
VE better than ENUMERATION of O(n* k^n), since Nmax lesser usually
Dep on size of largest interim factor

10. Approx inference by SAMPLING.
Sampling from multinom distribution. Draw a U value from dist. Convert U to a variable outcome base on subinterval of prob table

Example: Sampling from P(A)

- $0 \le u < 0.6, \Rightarrow a_0$
- $0.6 \le u < 0.7, \Rightarrow a_1$
- $0.7 \le u < 1, \Rightarrow a_2$

A	P(A)
a_0	0.6
a_1	0.1
a_2	0.3

Prior sampling Draw several samples from the Bayes' net by visiting each variable in topology order. Prob of sample is consistent as LLN. Weakness for many variables (eg100s), hard to get enough samples of rare events

Rejection Sampling: Intuition can calculate the conditional probability as: ▸ Sample from prior distribution (prior sampling) ▸ **Ignore (reject) samples that don't match evidence** ▸ Use counting to obtain the probability. Only counting amongst those relevant to evidence of query However it can: ▸ Rejects many samples if the evidence is unlikely! ▸ Fraction of samples consistent with evidence decreases as the #evidence variables grows.

OR LIKELIHOOD WEIGHTING ▸ Idea: fix evidence variables and sample the rest. **Do this when generation values are provided then you can fix evidences**. **If samples pregiven only consider those consistent with evidences, recall tutorial qn**
Example: weight is 0.8 for the sample [b, a] just weight it by 0.8 rather than explicitly sampling. ***Multiply weights only by the evidences var values***. LLweight also consistent sampling LLN

Ex: P(C|S =s, W = w)
Sampling Process (topological order). set initial weight =1

- *Sample* from P(C), value = c
- Sprinkler is an *evidence* variable:
  - Value = s, weight= 1*0.1
- *Sample* from P(R|C=c), value=r
- Wet grass is an *evidence* variable:
  - Value = w, weight = 0.1*0.99
- Sample returned: [c, s, r, w] with weight = 0.099

LIKELIHOOD WEIGHTING better than rejection sampling? ▸ Yes, better because uses all samples. Weakness: ▸ Performance bad with large number of evidence variables ▸ **Most samples very small weights as the number of evidence variables increases** ▸ The sampling process pays attention to evidence only in ancestor variables. Highly improbable event because ▸ Weights are low ▸ Problem LL weighting works well when evidence upstream

11. BN STRUCTURE LEARNING. Entropy: Measures the uncertainty we have about the value of a random variable. BE CAREFUL CONDITIONAL ENTROPY
P(X,Y) for cond entropy is from prob of full joint table
ENTROPY ALWAYS POSITIVE BECAUSE WE NEGATIVE LOG

- $Entropy\ (X) = -\sum_x p(X)log_2 p(X)$

Boolean
p(x1) = p
p(x2) = 1-p

= -(p log p + (1-p) log(1-p))

$$ENT(X|Y) = -\textstyle\sum_y p(Y=y)ENT(X|Y=y) = -\sum_{x,y}p(X,Y)log_2 p(X|Y)$$

Structure Learning two components:
▸ Objective score func: score higher for better fit data, Least negative based on LogLLH
▸ Optimization or Search: ▸ Search structure with the highest score
Bayes nets are directed acyclic graphs. – Steps for calculating Log-likelihood score
1. Get the CPTs of each node using data (MLE estimates)
2. Calculate the entropy of each node. 3. Use the entropies get the log-likelihood score
p(x,y) is from the entire joint data probability for CONDITIONAL entropy.
By adding more parents to a variable, decreases the uncertainty(entropy) ▸never increase its entropy term ▸ We will never decrease the log-likelihood of the resulting structure. Highest LLHOOD score is always the COMPLETE DAG —> so need penalty

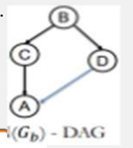
G* is the result of adding edges to G,

- LL(G*|D) ≥ LL(G|D)

AIC penalty is constant, use 1 unless
BIC pen is log(N)/ 2
Scores can be decomposed
Can find BIC of a node also

ENT should be positive value. So logLL = -ENT, loglikehood will be negative value.
Scoring should be based on logLL and find highest value(least negative).
N size of DATASET

$$LL(G|Data) = -N * (ENT\ B + ENT\ C|B + ENT\ D|B + ENT\ A|C,D) = \text{--ve value}$$



Scoring measure function:

- Score(G|D) = LL(G|D) - $\varphi(N)$ dim(G)

dim(G): dimension of BN(G)
N: *Size of the dataset*

First component: log-likelihood function

- Enhances the fit with data, and prefers models with more parameters

Second component: $\varphi(N)$ dim(G)

- Penalty term that favors simpler (smaller number of parameters) models
- dim(G) : model complexity
- $\varphi(N)$: is a function of the data size N, weight for model complexity

search over the space of graph structures? Move to the neighbor DAG that has the highest score ▸ **Neighbors are obtained through three operations by 1 step: Adding, removing, or reversing edge in one step**