$P\left(s\right)=\sum\_{t}^{}P(t,s)$

$P\left(b\right)=\frac{P(a,b)}{P(b)}$

$P\left(x\_{2}\right)=\frac{P\left(x\_{1},x\_{2}\right)}{\sum\_{x\_{1}}^{}P\left(x\_{1},x\_{2}\right)}$

$P\left(y\right)P\left(y\right)=P\left(x,y\right)$

$P\left(x\_{1},x\_{2}, …, x\_{n}\right)=\prod\_{i}^{}P(x\_{i}|x\_{1}, …, x\_{i-1})$

$X⫫Y | Z iff P\left(z\right)=P\left(z\right)P\left(y\right) or P\left(y,z\right)=P(x|z)$

Bayes net CPT: $P(x\_{i}|parents\left(X\_{i}\right))$

Bayes net: $P\left(x\_{1},x\_{2}, …x\_{n}\right)=\prod\_{1\rightarrow n}^{}P(x\_{i}|parents\left(X\_{i}\right))$

Size of joint dist over N variables: $2^{N}$

Size of N-node net if nodes have up to k parents: $N\*2^{k+1}$

D-separation: independence guaranteed if no active paths from X to Y

Active: W🡪W🡪W, W🡨W🡪W, W🡪G🡨W, W🡪(W🡪G)🡨W

Inactive: W🡪G🡪W, W🡨G🡪W, W🡪W🡨W

Bayes nets compactly encode joint distributions

Guaranteed independencies of distributins can be deduced from BN graph structure

D-separation gives precise conditional independence guarantees from graph alone

A Bayes’ net’s joint distribution may have further (conditional) independence that is not detectable until you inspect its specific distribution

Inference by enumeration: Select entries consistent with evidence, sum out hidden variables to get joint of query and evidence, normalize

Inference by elimination: Track factors, Join factors, eliminate by summing out a variable to make a new factor, normalize

Ex. $\sum\_{u}^{}all factors with u$

Complexity determined by largest factor

Prior sampling: sample x­­I from P(Xi|Parents(Xi))

Rejection: Only keep samples consistent with evidence (given)

Likelihood weighting: $S\left(w,e\right)=\prod\_{i=1}^{l}P(z\_{i}|Parents\left(z\_{i}\right))$. $w\left(z,e\right)=\prod\_{i=1}^{m}P\left(Parents\left(E\_{i}\right)\right)$

Estimate probability based on weights.

Gibbs Sampling:

Procedure: keep track of a full instanJaJon x1, x2, …, xn.

Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a Jme, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.

Property: in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution

Rationale: both upstream and downstream variables condition on evidence.

In contrast: likelihood weighting only conditions on upstream evidence, and hence weights obtained in likelihood weighting can someJmes be very small. Sum of weights over all samples is indicative of how many “effective” samples were obtained, so want high weight.

Step 1: Fix evidence. Step 2: Initialize other variables. Step 3: Repeat: Choose non-evidence variable X, Resample X from P(X|all other variables)

Only CPTs with the resampled variable remain.

Decision networks. $EU\left(leave\right)=\sum\_{w}^{}P\left(w\right)\*P(leave,w)$

$MEU\left(\left\{\right\}\right)=\max\_{a}EU\left(a\right)=70$ (2 options for a: leave, take).

­$EU\left(bad\right)=\sum\_{w}^{}P\left(bad\right)U\left(leave,w\right)$ ­where a🡪U🡨w🡪Forecast=bad

$MEU\left(F=bad\right)=\max\_{a}EU(a|bad)$

$VPI(E^{'}|e)=(\sum\_{e'}^{}P\left(e^{'}\right)MEU(e,e^{'}))-MEU(e)$

VPI properties: nonnegative, nonadditive, order-independent

POMDPs add: observation function, observations

HMMs: Underlying Markov chain over states X, observe effects at each step

Defined by: Initial distribution P(X1), Transitions P(Xt|Xt-1), Emissions P(Et|Xt)

Passage of time: Belief B(Xt)=P(Xt|e1:t).

P(Xt+1|e1:t)=$\sum\_{x\_{t}}^{}P\left(X\_{t}\right)P(x\_{t}|e\_{1:t})$. $B^{'}\left(X\_{t+1}\right)=\sum\_{x\_{t}}^{}P\left(x\_{t}\right)B(x\_{t})$

Observation: B’(Xt+1)=P(Xt+1|e1:t)

$P\left(e\_{1:t+1}\right)∝P\left(X\_{t+1}\right)P(X\_{t+1}|e\_{1:t})$ . $B(X\_{t+1}∝P\left(X\_{t+1}\right)B'(X\_{t+1})$

Forward algorithm: $B\_{t}\left(X\right)=P(x\_{t}|e\_{1:t})∝P\left(x\_{t}\right)\sum\_{x\_{t-1}}^{}P\left(x\_{t-1}\right)P(x\_{t-1}, e\_{1:t-1})$

Particle filtering elapse time: $x^{'}=sample(P\left(x\right))$

observe: $w\left(x\right)=P\left(x\right), B(x)∝P\left(X\right)B'(X)$

Viterbi algorithm: $m\_{t}\left[x\_{t}\right]=\max\_{x\_{1:t-1}}P\left(x\_{1:t-1},x\_{t},e\_{1:t}\right)=P\left(x\_{t}\right)\max\_{x\_{t-1}}P\left(x\_{t-1}\right)m\_{t-1}[x\_{t-1}]$

General naïve bayes: |Y| parameters

$P\left(Y,F\_{1}, …F\_{n}\right)=P(Y)\prod\_{i}^{}P(F\_{i}|Y)$ if Fi are children of Y

$P\_{ML}\left(x\right)=\frac{count(x)}{total samples}, L\left(x,θ\right)=\prod\_{i}^{}P\_{θ}(x\_{i})$

Maximum likelihood: $θ\_{ML}=argmax\_{θ}P\left(θ\right)=argmax\_{θ}\prod\_{i}^{}P\_{θ}(X\_{i})$

$θ\_{MAP}=argmax\_{θ}P\left(θ\right)P(θ)$

Laplace smoothing: $P\_{LAP,k}\left(x\right)=\frac{c\left(x\right)+k}{N+k|X|}$ N=number of sample, |X|=# values input can take

Tune hyperparameters(i.e. k) on held-out data

Baselines: determine how hard the task is, know what “good” accuracy is (ex. previous work)

Confidence: $confidence\left(x\right)=\max\_{y}P(y|x)$

$activation\_{w}\left(x\right)=\sum\_{i}^{}w\_{i}∙f\_{i}=w∙f(x)$ If positive, output +1 if negative -1

Binary decision rule: examples are points, weight vector is hyperplane, sides correspond to +/-1

$y=+1 if w∙f\left(x\right)\geq 0,-1 if w∙f\left(x\right)<0$

start with weights 0. if correct, no change. else, $w=w+y\*∙f$

Multiclass decision roles: weight vector for each class wy, score = $w\_{y}∙f(x)$

Prediction highest score wins: $y=argmax\_{y} w\_{y}∙f(x)$

Start with weights 0. if correct no change. else, lower score of wrong, raise score of right.

$w\_{y}=w\_{y}-f\left(x\right), w\_{y\*}=w\_{y\*}+f(x)$

Properties of perceptrons:

Seperability: true if some parameters get the training set perfectly correct

Convergence: if the training is seperable, perceptron will eventually converge (binary case)

Mistake Bound: max number of mistakes (binary case) related to margin/degree of seperability

$mistakes< \frac{k}{δ^{2}}$

Problems: noise (if data not seperable), mediocre generalization (barely separating), overtraining

MIRA: adjust weight update $\min\_{w}\frac{1}{2}\sum\_{y}^{}\left|\left|w\_{y}-w\_{y}^{'}\right|\right|^{2}, w\_{y\*}∙f\left(x\right)\geq w\_{y}∙f\left(x\right)+1$

$w\_{y}=w\_{y}-τf\left(x\right), w\_{y\*}=w\_{y\*}+τf(x)$ , $τ=\frac{\left(w\_{y}^{'}-w\_{y\*}^{'}\right)∙f+1}{2f∙f}$

Maximum step size: $τ^{\*}=min⁡(\frac{\left(w\_{y}^{'}-w\_{y\*}^{'}\right)∙f+1}{2f∙f},C)$

SVM: maximize the margin $\min\_{w}\frac{1}{2}\left|\left|w\right|\right|^{2}, ∀i,y w\_{y\*}∙f\left(x\_{i}\right)\geq w\_{y}∙f\left(x\_{i}\right)+1$

Naïve bayes: builds a model training data, gives prediction probabilities, strong assumptions about feature independence, one pass through data (counting)

Perceptron/MIRA: less assumptions about data, mistake-driven learning, multiple passes through data (prediction), often more accurate

Changing bayes nets: careful not to create cycles/independences not present in the original

Summing out connects neighbors, observing severs edges unless active