Probability Theory

Probability Space $(\Omega, \mathcal{F}, P)$. where Ω is set of atomic events, and $\mathcal{F}$ is σ-algebra:

$$\mathcal{F} \subseteq 2^{\Omega},\ \Omega \in \mathcal{F}$$
$$A \in \mathcal{F} \Rightarrow \Omega \setminus A \in \mathcal{F}$$
$$A_1, ..., A_n \in \mathcal{F} \Rightarrow \bigcup_i A_i \in \mathcal{F}$$

Gaussian Distribution

1 r.v.: $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
Multivariate Gaussian:
$p(\underline{x}) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} exp(\frac{-1}{2}(\underline{x}-\mu)^T \Sigma^{-1} (\underline{x}-\mu))$
Marginalization is just indexing.
Conditioning:

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

Variance is always shrinking when conditioning happens, so gaussians never surprise!
Multiplication $X \sim \mathcal{N}(\mu_V, \Sigma_{VV}) \in R^d$ and $M \in R^{m \times d}$, multiplication of these two is also gaussian: $MX \sim \mathcal{N}(M\mu_V, M\Sigma_{VV}M^T)$

Prod, Sum and Bayes Rule

Product Rule: $P(A, B) = P(A|B)P(B)$
Sum Rule: $P(A) = \sum_b P(A, B = b)$
Bayes Rule: $P(X)$ is prior, $P(Y|X)$ is likelihood, $p(X|Y)$ is posterior:
$p(X|Y) = \frac{p(X)p(Y|X)}{\sum_x p(X=x)p(Y|X=x)}$

Conditional Independence Properties

$A \perp B|C$ means $P(A|B, C) = P(A|C)$ or $P(A, B|C) = P(A|C)P(B|C)$

$$X \perp Y, W|Z \Rightarrow X \perp Y|Z$$
$$(X \perp Y|Z) \wedge (X \perp W|Y, Z) \Rightarrow X \perp Y, W|Z$$
$$X \perp Y, W|Z \Rightarrow X \perp Y|W, Z$$
$$(X \perp Y|W, Z) \wedge (X \perp W|Y, Z) \Rightarrow X \perp Y, W|Z$$

Bayesian Networks

Every probability distribution *can* be described by a BN. $X \perp Y|Z$ but $X \neg \perp Y$:

- indirect causal effect: $X \to Z \to Y$
- indirect evidential effect: $X \leftarrow Z \leftarrow Y$
- common cause: $X \leftarrow Z \to Y$

$\neg(X \perp Y|Z)$ but $X \perp Y$:

- common effect: $X \to Z \leftarrow Y$

If X and Y are d-separated in a BN it means that they are independent in any distribution on that graph. But if X and Y are independent it does not mean that they are d-separated in any BN that we have. (Because in creating BN order matters).

Typical Queries

- Conditional/marginal query: (p-complete) $p(X_i|X_s = x_s)$? $\Rightarrow$ Variable Elimination
- MPE (Most probable explanation): (NP-complete) $\underset{x_{\bar{s}}}{argmax}\ p(X_{\bar{s}} = x_{\bar{s}}|X_s = x_s)$
- MAP (Maximum a posteriori): (NP^{PP}-complete) $\underset{x_A}{argmax}\ p(X_A = x_A|X_B)$

Conditional/marginal query

Exact: Variable elimination

- Cashing reduces the number of summations.
- Odering matters: use topological order for poly trees.$O(n)$ A DAG is a polytree iff dropping edge directions results in a tree. every polytree is a DAG but every DAG is not a poly tree. for other cases any ordering may cause factors to blow up! $(O(2^n))$
- Variable Elimination doesn't care about probabilistic interpretation of factors!

Message Passing on Factor Graphs(Belief Propagation)

Fast, but may not converge, or converge to wrong dist in loopy graphs

- Initialize all messages as uniform distribution
- Until converged do
  1. Pick some ordering on the factor graph edges (+directions)
  2. Update messages according to this ordering
     Messages from variable v to factor u:
     $\mu_{v\to u}^{(t+1)}(x_v) = \prod_{u' \in N(v)\setminus\{u\}} \mu_{u'\to v}^{(t)}(x_v)$
     Messages from factor u to variable v:
     $\mu_{u\to v}^{(t+1)}(x_v) = \sum_{x_u \sim x_v} f_u(x_u) \prod_{v' \in N(u)\setminus\{v\}} \mu_{v'\to u}^{(t)}(x_{v'})$
  3. Break once all messages change by at most ϵ

Intention: $\hat{}(X_v = x_v) \propto \prod_{u \in N(v)} \mu_{u\to v}(x_v)$
$\hat{}(X_u = x_u) \propto f_u(x_u) \prod_{v \in N(u)} \mu_{v\to u}(x_u)$ For polytrees, like variable elimination we first send messages from leaves to the root and then backwards from root to the leaves and then like chains we can show the convergence.
if overshoot: they got more confident$\to$MPE mostly is correct they may not converge: if range of prior becomes smaller(more deterministic) the chance that the algorithms doesn't converge is higher.

MPE query

Like variable elimination we can find the max probability and cash the intermediate results. And instead of sum, we have max now. And we need a backward pass to find the argmax.
We can apply belief propagation(message passing) just like what we did for marginals and just change factor to node massages sum to max!

Mean-Field Variational Inference

Fast and converges in loopy graphs but it's not exact.
$Q^* \in \underset{Q \in \mathcal{Q}}{argmin}\ KL(Q \parallel P)$
$\mathcal{Q} = \{Q : Q(X_{1:n}) = \prod_{i=1}^n Q_i(X_i)\}$
Intuition: posterior often much better approximated by a simple distribution than the prior.
KL distance $KL(Q \parallel P) = \sum Q(x) log\frac{Q(x)}{P(x)}$
Entropy $H(Q) = -\sum Q(x) log(Q(x))$ for discrete and finite range: uniform is most uncertain distribution, for continuous infinite range: gaussian is most uncertain(with max entropy)
Evidence Lower Bound (ELBO):
$\underset{Q}{argmin}\ KL(Q \parallel P) = \underset{Q}{argmax} \sum_{j=1}^n H(Q_j) + \sum_i \sum_{x_{A_i}} (\prod_{j \in A_i} Q_j(x_j)) log\Phi_i(x_{A_i})$
How to optimize? Gradient descent and coordinate ascent work. Coordinate ascent fix all variables except one and optimize that variable.
$Q^*(x_i) \propto E_{x-i\sim Q^{(t)}} log\ p(x_{1:n})$

Hoeffding's Inequality

if f is bounded $[0, C]$
$P(|E_P[f(X)] - \frac{1}{N}\sum_i f(x_i)| > \epsilon) \leq 2exp(-2N\epsilon^2/C^2)$

Sampling

- Forward sampling even works for loopy models efficiently.
- Use forward sampling to sample from BN and then use counts to compute conditional probabilities leads to *rejection sampling*. This has the problem with rate events. so we should take care of **relative error in Hoeffding's inequality**.
- relative error in Hoeffding's inequality: $P(\hat{p}(x) \notin p(x)(1 \pm \epsilon)) \leq 2exp(-Np(x)\epsilon^2/3)$
- So as long as we are sampling from a prior we are ok, but using rejection sampling to sample for the posterior is not a good idea and we need gibbs sampling for that.

MCMC

- converges to the exact marginals but we should wait maybe for a looong time.
- stationary distributions: $\underset{N\to\infty}{lim} P(X_N = x) = \pi(x)$ which is in dependant of $p(X_1)$
- if detailed balance $\to$ markov chain has stationary distribution of $\frac{1}{Z}Q(x)$, detail balance is for any state x and x': $Q(x)p(x'|x) = Q(x')p(x|x')$
- proposal distribution: given $X_t = x$ same x' from $R(x'|x)$
- acceptance distribution: accept the proposal with probability $\alpha = min\{1, \frac{Q(x')R(x|x')}{Q(x)R(x'|x)}\}$

Gibbs Sampling

Random order

- randomly initialize variables
- fix observed variables
- in each iteration pick one random variable uniformly random from unobserved variables, say x_i
- sample x_i from $P(x_i|v_i)$ which is conditioned on all other variables except x_i. **Sampling from X_i given all other variables can be done efficiently and only depends on the factors that x_i participated in.**
- this satisfies detailed balance and leads to true stationary distribution

Practical variant

In each iteration we update all of the variables. This does not satisfies detailed balance but leads to true stationary distribution.

Ergodic Theorem

$\underset{N\to\infty}{lim} \frac{1}{N}\sum_i f(x_i) = E_{x\sim\pi}[f(x)]$ How many samples? we don't know. One can actually throw away "burn-in" period samples and compute the average on the rest of the samples. How to pick t_0 as burn-in period? run multiple mcmc, compare variance of samples within one chain to variance across the chains, after the burn-in period these two must roughly be the same.

Inference Tasks

- Filtering $P(X_t|Y_{1:t})$
- Prediction $P(X_{t+\Delta}|Y_{1:t})$
- Smoothing $P(X_t|Y_{1:T})$, $1 \leq t \leq T$
- MPE: $\underset{X_{1:T}}{argmax} P(X_{1:T}|Y_{1:T})$

Bayesian Filtering

Assume we have $P(X_t|Y_{1:t-1})$

- Conditioning: $P(X_t|Y_{1:t}) = \frac{1}{Z}P(X_t|Y_{1:t-1})P(Y_t|X_t)$ and $Z = \sum_x P(X_t = x|Y_{1:t-1})P(Y_t|X_t = x)$
- Prediction: $P(X_{t+1}|Y_{1:t}) = \sum_x P(X_t = x|Y_{1:t})P(X_{t+1}|X_t = x)$

Kalman Filter

Motion Model $P(X_{t+1}|X_t)$:
$X_{t+1} = FX_t + \epsilon_t$ where $\epsilon_t \in \mathcal{N}(0, \Sigma_x)$
Sensor Model $P(Y_t|X_t)$:
$Y_t = HX_t + \eta_t$ where $\eta_t \in \mathcal{N}(0, \Sigma_y)$
Kalman Update:
$\mu_{t+1} = F\mu_t + K_{t+1}(y_{t+1} - HF\mu_t)$
$\Sigma_{t+1} = (I - K_{t+1})(F\Sigma_t F^T + \Sigma_x)$
Kalman Gain: $K_{t+1} = (F\Sigma_t F^T + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_y)^{-1}$

Particle Filtering

approximate inference technique for dynamical models(dynamic bayesian networks, non-linear kalman filters). Given:
$P(X_t|Y_{1:t}) \approx \frac{1}{N}\sum_i \delta_{x_{i,t}}$
Prediction:(sample according to out motion model)
$x'_i \sim P(X_{t+1}|x_{i,t})$
Conditioning:
Weigh particles: $w_i = \frac{1}{Z}p(y_{t+1}|x'_i)$
Resample N particles: $x_{i,t+1} \sim \sum_i w_i \delta_{x'_i}$
If no resampling? only one particle with weight 1 $\rightarrow$ starvation

Assumed Density, EKF, UKF

Assumed Density Filter Use forward KL to estimate the posterior which leads to moment matching. **Extended Kalman Filters** next state assumed to be gaussian: propagate the mean function through the non-linear dynamics. propagates the covariance through a linearization of the system. **Unscented Kalman Filter (UKF)** propagates some weighted points. approximates the posterior by a gaussian by moment matching. Kalman filters do not assume the gaussain distribution for every state. The Extended Kalman Filter is not an Assumed Density Filter. The Unscented Kalman Filter is an Assumed Density Filter. The Unscented Kalman Filter is not a Particle Filter.

Value of a policy and policy of a value

Given the policy:
$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_{x'} p(x'|x, \pi(x))V^\pi(x')$
which is equal to solving this linear system:
$V^\pi = r^\pi + \gamma T^\pi V^\pi$
Given the value function:
$\pi_V(x) = \underset{a}{argmax}\ r(x, a) + \gamma \sum_{x'} p(x'|x, a)V(x')$

Bellman Theorem

A policy is optimal if and only if it is greedy w.t.r to it's induced value function.
$V^*(x) = \underset{a}{max}[r(x, a) + \gamma \sum_{x'} p(x'|x, a)V^*(x')]$

Policy Iteration

Policy iteration start with a random policy, compute it's value and update the policy greedily. In polynomial iterations $\sim O(n^2m)$ converges to optimal policy, complexity of each iteration $O(n^3 + n^2m)$. Monotonically improve $V^{\pi_{t+1}} \geq V^{\pi_t}$. Problem ? solving the linear system per iter.

Value Iteration

Define $V_0(x) = max_a r(x, a)$, in iteration t:
$Q_t(x, a) = r(x, a) + \gamma \sum_{x'} p(x'|x, a)V_{t-1}(x')$,
$V_t(x) = max_a Q_t(x, a)$. break if $\underset{x}{max}|V_t(x) - V_{t-1}(x)| \leq \epsilon$.
Complexity per iteration: $O(n^2m)$ but $O(nms)$ for sparse MDPs where number of states reachable from any given state and action is $\leq s$
Bellman update is a contraction:
$BV = max_a r(x, a) + \sum_{x'} p(x'|x, a)v(x')$ and
$\|BV - BV'\|_\infty \leq \gamma\|V - V'\|_\infty$
Using linear programming for solving bellman equation is useful if we have a structured MDP and we want to encode them as well.

POMDP

- POMDPs are much more powerful than MDPs but they are intractable in general. POMDP can be transformed to Belief-state MDP
- Stochastic observation: $P(Y_{t+1} = y|b_t, a_t) = \sum_x b_t(x)P(Y_{t+1} = y|X_t = x, a_t)$
- State update: $b_{t+1}(x') = \frac{1}{Z}\sum_x b_t(x)P(X_{t+1} = x'|X_t = x, a_t)P(y_{t+1}|x')$
- Reward function: $r(b_t, a_t) = \sum_x b_t(x)r(x, a_t)$
- The problem is we have infinitely many beliefs.But after a finite horizon time steps, there are finite possible beliefs and we can solve for that.

MLE for bayesian nets

$\theta^*_{j|par_j} = \frac{Count(X_j, X_{par_j})}{Count(X_{par_j})}$

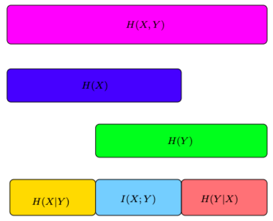
Learning as inference

Given $D = \{(f^{(i)}, w^{(i)}), i \in \{1, ..., n\}\}$
MAP estimation
$\hat{\theta} \in \underset{\theta}{argmax}\ P(\theta|f^{(i)}, w^{(i)})$
$\Rightarrow P(F|w, D) \approx P(F|w, \hat{\theta})$
Bayesian Learning
$P(Y(x')|D) = \int P(w|D)P(Y(x')|x', w)dw$
So if we are so sure about the w and $P(w|D)$ is peaky, bayesian learning is more similar to using the MAP estimate, but if we are unsure(e.g. bimodal distribution of w) bayesian estimate is so different from MAP.

Learn BN structure

MLE score is the best possible likelihood we could achieve under the best choice of parameters which is θ_G. Which is equal to:
$log(D|\theta_G, G) = N\sum_i \hat{I}(X_i, Par_{X_i}) + const$
BIC score
$S_{BIC}(G) = \sum_i \hat{I}(X_i, Par_{X_i}) - \frac{logN}{2N}|G|$ where $|G|$ is number of rows in G CPTs. It can be shown that BIC is consistent.
If we cosider graphs where each node can only have one parent, the problem reduces to finding maximum spanning tree which can be solved in $O(|E|log|E|)$ (Chow-Liu algorithm)

Mutual Information



$I(X_A; X_B) = \sum_{x_A, x_B} P(x_A, x_B)log\frac{P(x_A, x_B)}{P(x_A)P(x_B)}$
$I(x_A; x_B) \geq 0$ and 0 iff X_A, X_B are independent.
MI is symmetric and monotonic means:
$\forall B \subseteq C : I(X_A; X_B) \leq I(X_A; X_C)$

MLE for logistic regression

$\hat{y}_i = \sigma(w^T x_i) = \frac{1}{1+exp(-w^T x_i)}$
$\hat{R}(w) = \sum_i log(1 + exp(-y_i w^T x_i))$
This cost function is convex.
SGD for logistic regression:
$\hat{P}(Y = -y|w, x) = \frac{1}{1+exp(yw^T x)}$
$w \leftarrow w + \eta_t yx\hat{P}(Y = -y|w, x)$

Aleatoric vs. epistemic uncertainty

Aleatoric uncertainty: Noise in observations given perfect knowledge of the model. **Epistemic uncertainty:** Uncertainty about parameters due to the lack of data (i.e., diffuse posterior).
Bayesian learning captures Epistemic uncertainty which MAP estimators do not!

RL

RL vs supervised learning? in RL data is not i.i.d and depends on our actions.
Model Based: first estimate transision model and reward function then use value or policy iteration to plan. **Model Free** Estimate value function directly like policy gradient methods and Q learning.
model-based vs Q learning? model based: store $O(|A||X|^2)$ and we should do the computation for optimal policy more than one time(e.g. for R max for each state action pair). model free: memory: $O(|X||A|)$, computation each iteration $O(|A|)$
On policy agent has control which actions to pick. **Off policy** we have the data already!
ϵ **greedy** with probability ϵ_t picks random action and with probability $1 - \epsilon_t$ pick best action. problem? Doesn't quickly eliminate clearly suboptimal actions.

R-max Algorithm

There is no excplicit distinction between explore and exploit, you always act optimally.
Eet for all action and state pair $R(x, a) = R_{max}$ and $P(x^*|x, a) = 1$ where x^* is the fairy tale state: $R(x^*, a) = R_{max}$, $P(x^*|x^*, a) = 1$
In each timesteps R_{max} either Obtains near-optimal reward, or Visits at least one unknown state-action pair.

Q-Learning

Estimate
$Q^*(x, a) = r(x, a) + \gamma \sum_{x'} P(x'|x, a)V^*(x')$ and $V^*(x) = max_a Q^*(x, a)$.
Observation: (x, a, r, x'), Initial estimate: $Q(x, a)$:
$Q(x, a) \leftarrow (1-\alpha_t)Q(x, a)+\alpha_t(r+\gamma max_{a'} Q(x', a'))$
Similar to R-max we can initialize optimistically:
$Q(x, a) = \frac{R_{max}}{1-\gamma}\prod_{t=1}^{T_{init}} \frac{1}{1-\alpha_t}$

DQN

$Q(x, a; \theta) = \theta^T\Theta(x, a)$ Try to minimize $L(\theta) = \sum_{(x,a,r,x') \in D}(r + \gamma max_{a'} Q(x', a'; \theta^{old}) - Q(x, a; \theta))^2$ Double DQN: use new estimate parameter θ to pick actions.

Policy Search-REINFORCE

Initialize policy weights θ. Generate an episode (rollout): $X_0, A_0, R_0, ..., X_t, A_t, R_t$. for each t update policy w.t.r reward:
$\theta = \theta + \alpha\gamma^t G_t \partial_\theta log\pi(A_t|X_t; \theta))$