1 Prepare

1.1 Conditional independence

1. $\Pr[a|b] = \Pr[a] \Rightarrow \Pr[a|b,c] = \Pr[a|c]$ False. Independence doesn't imply conditional independence.
2. $X \perp Y|Z \Rightarrow X \perp Y$. False. Conditional independence doesn't imply independence.

1.2 Active trail

(informally: path along which information can flow) Two kinds of active trails:
- $\neg X \perp Z, X \perp Z|Y$ with Y unobserved.
— $X \to Y \to Z$
— $X \leftarrow Y \leftarrow Z$
— $X \leftarrow Y \to Z$
- $X \perp Z, \neg(X \perp Z|Y)$ with y or decendant(Y) observed: $X \to Y \leftarrow Z$

Theorem Given observations O No active trail between $X, Y \Leftrightarrow d-sep(X,Y;O) \Rightarrow X \perp Y|O$

1.3 Information theory

Mutual information $I(\boldsymbol{X}_A;\boldsymbol{X}_B) = \sum_{\boldsymbol{X}_A,\boldsymbol{x}_B} P(\boldsymbol{X}_A,\boldsymbol{X}_B)\log\frac{P(\boldsymbol{X}_A,\boldsymbol{X}_B)}{P\boldsymbol{X}_A)P(\boldsymbol{X}_B)}$.
- $I(\boldsymbol{X}_A;\boldsymbol{X}_B) \geq 0$, and $I(\boldsymbol{X}_A;\boldsymbol{X}_B) = 0 \Leftrightarrow \boldsymbol{X}_A \perp \boldsymbol{X}_B$
- monotonicity: $\forall B \subseteq C : I(\boldsymbol{X}_A;\boldsymbol{X}_B) \leq I(\boldsymbol{X}_A,\boldsymbol{X}_C)$

1.4 MAP VS MLE

$\hat{\theta}_{MLE} = \arg\max \log \prod_{i=1}^N P(x_i;\theta) = \arg\min -\sum_{i=1}^N \log P(x_i;\theta)$
$\hat{\theta}_{MAP} = \arg\min -\sum_{i=1}^N \log P(x_i;\theta) - \log P(\theta)$ (trade-off between likelihood and prior)

1.5 Linear regression

Known: prior: $P(\theta) \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, $\{(x_i,y_i)\}$, $\epsilon \sim \mathcal{N}(0,\sigma_y^2)$ and $y = \theta x + \epsilon$.
$p(\theta^*|X,y) \sim \mathcal{N}(\mu', \Sigma')$ with $\Sigma' = (\Sigma_\theta^{-1} + \frac{1}{\sigma_y^2}X^TX)^{-1}$ and $\mu' = \Sigma'(\Sigma_\theta^{-1}\mu_\theta + \frac{1}{\sigma_y^2}X^Ty)$

2 Inference

2.1 Bayesian Network

Bayesian network (G,P)**:** DAG with conditional probability distribution $\Pr[X_i|Pa(X_i)]$. Joint distribution $\Pr[X_{1:n}] = \prod_i \Pr[X_i|Pa(X_i)]$

Specifying a BN Given variables $X_1,\cdots,X_n$. Pick ordering. For all $i \in [n]$ find the minimum parent set $A \subseteq \{X_1,\ldots,X_{i-1}\}$ s.t. $X_i \perp X_{\bar{A}}|X_A$. 2.Specify/learn $\Pr[X_i|A]$. BN defined this way are sound. Ordering matters a lot for compact.

Numbers $|X|$: the cardinality of variables; N the number of free variables; $|X|^N$: the number of states.

2.2 Typical Queries

Conditional Distribution, MPE ($\arg\max_{e,b,a}\Pr[e,b,a|J = t, M = f]$), MAP ($\arg\max_e \Pr[e|J = t, M = f]$)

2.3 Variable Elimination

Algorithm: condition query Input: BN, query variable(s) X, observed values e for variable(s) E
Output: $\Pr[X|E = e]$
1. Choose ordering for all variables: $X_1,\cdots,X_n$
2. For $i = 1:n$: create initial factors $F = \{f_i = \Pr[X_i|Parents(X_i)]\}$
3. For $i = 1:n$ If $X_i \notin \{X,E\}$:
3.1 multiply all factors $\{f_{i_1},\cdots,f_{i_m}\}$ that include X_i
3.2 marginalize out $X_i \Rightarrow g = \sum_{x_i}\prod_{i_k} f_{i_k}$
3.3 add g to set of factors
4. Renormalize $\Pr(x,e)$ to get $\Pr[x|e] \propto \prod_i f_i \prod_j g_j$, the product of left factors including only variables in $X \cup E$

Algorithm: MPE/MAP Similar to condition query, with the change $g = \max_{x_i}\prod_{i_k} f_{i_k}$
To get the argmax, in the end, add the step: For $i = n:-1:1.\hat{x}_i = \arg\max_{x_i} g_i(x_i,\hat{x}_{i+1:n})$

3 Approximate Inference

3.1 Variational Inference

Idea: use distribution in specific families to approximate the unknown distribution.
$KL(Q||P) = \sum_x Q(x)\log\frac{Q(x)}{P(x)}$ (reverse)
If $Q(X_{1:n}) = \prod_{i=1}^n Q_i(X_i)$ and $P(X) = \frac{1}{Z}\prod_{i=1}^m \Phi_i(X_{A_i}) \Rightarrow$
$\arg\min_{Q\in\mathcal{Q}} KL(Q||P) = \arg\max_Q \sum_{i=1}^n H(Q_i) + \sum_i^m \sum_{x_{A_i}} \prod_{j\in A_i} Q_j(x_j)\log\Phi(x_{A_i})$ (ELBO(Q))
(CAVI)$Q_j \arg\max_{Q_j} ELBO_j(Q) \propto \exp(\sum_{x_{j-}}\prod_{i\neq j} Q_i(x_i)\log\Phi(x))$
$ELBO_j(Q) = \sum_{x_j} Q_j(x_j)\sum_{x_{j-}}\prod_{i\neq j} Q_i(x_i)\log\Phi(x) - \sum_x Q_j(x_j)\log Q_j(x_j)$

3.2 MCMC

Idea: create Markov chain from unnormalized distribution $Q(X)$, which has the stationary distribution $P(X)$ which we want to sample from.

Forward sampling 1.Sort variables in topological ordering $X_1,...,X_n$
2. For $i = 1$ to n do: Sample $x_i \sim P(X_i|X_1 = x_1,...,X_{i-1} = x_{i-1})$
Probability computing: Marginals: $P(w = t) \approx \frac{1}{N}\sum_{i=1}^N [w = t](x^{(i)} = \frac{Count(w=t)}{N}$
Conditionals: $P(C = t|W = t) = \frac{P(C=t,W=t)}{P(w=t)} \approx \frac{Count(W=t,C=t)}{Count(W=t)}$

Metropolis Hastings MCMC Algorithm Given $X_t = x$,
1. Proposal distribution $R(X',X)$. Sample 'proposal' $x' \sim R(X',X = x)$ (Note: The performance of the algorithm will strongly depend on R.)
2. Acceptance distribution:Suppose $X_t = x$ With probability $\alpha = \min\{1,\frac{Q(x')R(x,x')}{Q(x)R(x',x)}$ set $X_{t+1} = x'$; Otherwise, set $X_{t+1} = x$ (again to the current/same state).

Gibbs Sampling (practical variant)
1. Start with initial assignment $x^{(0)}$ to all variables
2. Fix observed variables X_B to their observed values x_B
3. For $t \leftarrow 1$ to ∞:
3.1 set $x^{(t)} = x^{(t-1)}$
3.2 For each variable X_i except those in B:
3.2.1 fix all the other variables' values as v_i
3.2.2 $x_i^{(t)} \leftarrow \Pr[X_i|v_i]$

Transition matrix Between two adjective states (n-dim), the transition probability=
- the probability of selecting certain variable (the i-th dimension) times
- the probability of that variable changes in a certain direction:
fix the other variables, calculate the marginal via BN, then renormalize.

4 Sequential model

$X_1,\cdots,X_T$: unobserved variables (states), $Y_1,\cdots,Y_T$: observations.
This factorizes as: $\Pr[X_{1:T},Y_{1:T}] = \Pr[X_1]\prod_{t=2}^T \Pr[X_t|X_{t-1}]\prod_{t=1}^T \Pr[Y_t|X_t]$
- $\Pr[X_1]$: the initial distribution
- $\Pr[X_t|X_{t-1}]$: the transition model
- $\Pr[Y_t|X_t]$: the measurement model.

4.1 Inference tasks

- Marginalization (Smoothing): $\Pr[X_t|y_{1:T}], \forall 1 \leq t \leq T$. - Filtering: $\Pr[X_t|y_{1:t}]$ - Prediction: $\Pr[X_{t+\Delta}|y_{1:t}]$
- MPE: $\arg\max_{x_{1:T}}\Pr[X_{1:T} = x_{1:T}|y_{1:T}]$

4.2 HMM

Number of Parameters Totally, N-1+N(N-1)+N(M-1)
- N-1: the probability distribution of the initial space with N states.
- N(N-1): the transition matrix
- N(M-1): the likelihood of observations given states.

Computational cost : Filtering: given the prior on initial state distribution $\Pr[X_1 = s|\phi]$, totally $O(Tn^2)$
$\Pr[X_1 = s_1|Y_1 = o_1] = \frac{\Pr[Y_1=o_1|X_1=s_1]\Pr[X_1=s_1]}{\sum_s \Pr[Y_1=o_1|X_1=s]\Pr[X_1=s]}$: $O(n)$ for all s,
$\Pr[X_2 = s_2|Y_1 = o_1] = \sum_s \Pr[X_2 = s_2|X_1 = s]\Pr[X_1 = s|Y_1 = o_1] : O(n), O(n^2)$ for all s_2
Have $\Pr[X_2 = s_2|Y_1 = o_1]$, compute $\Pr[X_2 = s_2|Y_1 = o_1, Y_2 = o_2] : O(n)$

4.3 Kalman Filter

Computational cost Number of modes grow exponentially. Use assumed density function to solve this problem.

Limitation 1. only describe Guassian distributions (unimodal); 2. only for linear transformation.

4.4 Particle Filtering

Application : Suppose the true distribution (could be continuous) is $P(X)$, we get N i.i.d samples from it: $x_1,\cdots,x_N$.
- represent: we can approximate $P(x)$ the distribution by $\frac{1}{N}\sum_{i=1}^N \delta_{x_i}(x)$
- get expectations: for an interesting objective function $f(X)$: we can approximate $\mathbb{E}_P[f(X)]$ by $\frac{1}{N}\sum_i f(x_i)$.

Algorithm Given the belief of prior distribution $P_0(X)$, we draw N i.i.d. samples from it: $x_{1,0},\cdots,x_{N,0}$ (in this way, we can approximate $P_0(x) \simeq \frac{1}{N}\sum_i \delta_{x_{i,0}}(x)$)
1. (prediction/propagate) $x_i' \sim P(X_{t+1}|X_t = x_{i,t})$
2. (conditioning) reweight the importance y_{t+1}: $w_i = \frac{1}{Z}P(y_{t+1}|x_i')$
3. resample N articles as the simulation of newly believed distribution according to the new weight: $x_{i,t+1} \sim \sum_i w_i\delta_{x_i'} \Rightarrow P(X_{t+1}|y_{1:t+1}) \simeq \frac{1}{N}\sum_i \delta_{x_{i,t+1}}(x)$

Reasons for resampling avoid starvation: all weight concentrate on a single particle; well represent the distribution.

4.5 Assumed Density Filtering

The general idea: instead of keeping track of the true marginals, using distributions from simple families (assumed density) to approximate it (like VI)

Definition 1. assume the prior and posterior are from the same parametric family; 2. the objective is to minimize forward KL: $Q^* \arg\min_{Q\in\mathcal{Q}} KL(P||Q)$

Kalam Filter is not an assumed density filtering. Because we don't assume the posterior is Gaussian.

4.6 Approximate non-linear system

- Transition: $x_{t+1} = f(x_t) + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0,\Sigma_x)$
- Challenge: Even if $P(X_t|y_{1:t})$ is Gaussian, $P(X_{t+1}|y_{1:t})$ is not.
- Approximation: $P(x_{t+1}|y_{1:t}) \simeq Q = \mathcal{N}(\mu,\Sigma)$

Extended Kalman Filter $\mu = f(\mu_t), \Sigma = \hat{F}\Sigma_t\hat{F}^T + \Sigma_x$, where $\hat{F} = \frac{\partial f(x)}{\partial x}|_{x=x_t}$

Unscented Kalman Filter suppose $P(X_t|y_{1:t}) \sim \mathcal{N}(\mu_t, \Sigma_t)$ and propagate points with dynamics $x_i' \sim f(x_i) + w$, approximate μ and Σ via moment matching ($\mu = \mathbb{E}_{x\sim p(\cdot)}[x], \Sigma = \mathbb{E}_{x\sim p(\cdot)}[(x-\mu)(x-\mu)^T]$)

5 Planning

5.1 Markov Decision Process

A MDP is specified by $< X, A, P(\cdot|\cdot,\cdot), r(\cdot), \gamma >$:
- X: a finite set of states (not variables, variables can be infinite)
- A: a finite set of actions
- $P(x'|x,a)$: transition probability (matrix)
- $r(x,a)$ or $r(x)$ or $r(x,a,x')$: reward function
- $\gamma \in [0,1]$: discount factor

5.2 Value function

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

$$\boldsymbol{v}^\pi = \begin{pmatrix} V^\pi(1) \\ V^\pi(2) \\ \vdots \\ V^\pi(n) \end{pmatrix}, \quad \boldsymbol{r}^\pi = \begin{pmatrix} r(1,\pi(1)) \\ r(2,\pi(2)) \\ \vdots \\ r(n,\pi(n)) \end{pmatrix},$$

$$\boldsymbol{T}^\pi = \begin{pmatrix} P(1|1,\pi(1)) & \cdots & P(n|1,\pi(1)) \\ P(1|2,\pi(2)) & \cdots & P(n|2,\pi(2)) \\ \vdots & \ddots & \vdots \\ P(1|n,\pi(n)) & \cdots & P(n|n,\pi(n)) \end{pmatrix}$$

$\Rightarrow \boldsymbol{v}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{T}^\pi)^{-1} \boldsymbol{r}^\pi . \boldsymbol{v}^\pi : X^N \to \mathbb{R}^N$

Q function $Q(x,a) = r(x,a) + \gamma \sum_{x'} P(x'|x,a) V(x')$

5.3 Bellman Theorem

Policy optimal $\Leftrightarrow$ Greedy w.r.t. its induced function.

Greedy policy w.r.t V: $\pi_V(x) := \arg\max_a r(x,a) + \gamma \sum_{x'} P(x'|x,a) V(x')$

Bellman Equation For the optimal policy π^* it holds that $V^*(x) := \max_a r(x,a) + \gamma \sum_{x'} P(x'|x,a) V^*(x')$

5.4 Value iteration

Algorithm (Init) For each $x \in \mathcal{X}$: $V_0(x) \leftarrow \max_a r(x,a)$
compute value function: For($t \leftarrow 1$ to ∞):
– For each $x \in \mathcal{X}$:
—- For each $a \in \mathcal{A}$: $Q_t(x,a) = r(x,a) + \gamma \sum_{x'} P(x'|x,a) V_{t-1}(x')$
—- $V_t(x) \leftarrow \max_a Q_t(x,a)$ (Bellman operator)
– If $(\|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\|_\infty < \epsilon')$:break
choose greedy policy π_G w.r.t. V_t

Convergence #iterations: exponential for $\gamma < 1$:
- $\forall V, \|BV - BV^*\|_\infty \leq \gamma \|V - V^*\|_\infty$ with $BV^* = V^*$
- (init) $\|V_0 - V^*\|_\infty \leq \frac{2R^{max}}{1-\gamma}$

$\gamma^N \frac{2R^{max}}{1-\gamma} < \epsilon \Leftrightarrow N = \lceil \log_{\frac{1}{\gamma}} \frac{2R^{max}}{\epsilon(1-\gamma)} \rceil$

In each iteration: $O(n^2 m)$

5.5 Policy iteration

Algorithm Initialize: policy π' (and reward vector, transition matrix)
Repeat:(1) $\pi \leftarrow \pi'$; (2) Value determination (solve linear system): $V^\pi(x) = r(x,\pi(x)) + \sum_{x'} P(x'|x,\pi(x))$; (3)Policy improvement for each state (via greedy policy); Until $\pi = \pi'$

Convergence #iterations: depends a lot on initial policy and polynomial on epsilon.

In each iteration: the complexity is $O(n^2 \cdot m)$.

Value determination: n^3 Policy update: for each state n, for each action m: calculate $Q(x,a)$: n. Totally $O(n^2 m + n^3)$

5.6 Summary

In practice, which of policy or value iteration works better depends on application. Policy iteration is monotonically improve $V^{\pi_{t+1}}(x) \geq V^{\pi_t}(x) \forall x, t$. Not value iteration.

5.7 POMDP

A POMDP (controlled HMM) is specified by $< X, Y, A, P(\cdot|\cdot,\cdot), P_S(\cdot|\cdot), r(\cdot), \gamma >$:
- X: a finite set of states (not variables, variables can be infinite)
- Y: observations
- A: a finite set of actions
- $P(x'|x,a)$: transition probability (matrix)
- $P(y|x)$: sensor(observation) model
- $r(x,a)$ or $r(x)$ or $r(x,a,x')$: reward function
- $\gamma \in [0,1]$: discount factor

Transform to MDP $< B, A, P(\cdot|\cdot,\cdot), r(\cdot), \gamma >$:
- B: beliefs over the original states. $B = \{b : b \in [0,1]^n, \sum_{x\in X} b(x) = 1\}, b_t(x) = P(x_t = x|a_{1:t}, y_{1:t})$
- A: a finite set of actions
- $P(b'|b,a)$: transition model: $P(b'|b,a) = \sum_y P(b'|y,a,b) \sum_{x'} P(y|x') \sum_x P(x'|x,a) b(x)$
— 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: Given a_t, b_t, y_{t+1} : $b_{t+1}(x') \propto P(y_{t+1}|X_{t+1} = x') \sum_x P(X_{t+1} = x'|x, a_t) b_t(x)$
- $r(b,a)$ or $r(x)$ or $r(x,b,x')$: reward function: $r(b,a) = \sum_x b(x) r(x,a)$
- $\gamma \in [0,1]$: discount factor **Policy** on POMDP maps from **belief state** to actions. Optimal policy on this MDP is also optimal on POMDP.

solutions Standard MDP methods are not suitable: (1)exponential in T,(2) many states never been reached.

Finite Forward search : For finite horizons T:
function ActionSearch(b,T):
If T=0, return [None,r(b)]
$[a^*, v^*] \leftarrow [None, -\infty]$
for $a \in A$ do:
– $v \leftarrow r(b,a)$
– For $y \in Y$ do:(1) $b' \leftarrow$ UpdateBelief(b,a,y);
(2) $[a', v'] \leftarrow$ ActionSearch(b,T-1)(3) $v \leftarrow v + P(y|b,a) v'$
– if $v > v^*$ then $[a^*, v^*] \leftarrow [a, v]$
return $[a^*, v^*]$

$Q(b,a) = r(b,a) + \sum_y P(y|b,a) r(b_{a,y}, a)$ where $b_{a,y}$ =updateBelief(b,a,y). Find optimal action:$\arg\max_a Q(b,a)$

Policy gradient method 1. use parameters to represent policy 2. for each parameter, sample and average to compute the expected value. 3. find optimal parameters.

6 Learning

6.1 Parameter

MLE for BN $\hat{\theta}_{X_i|Pa(X_i)} = \frac{COunt(X_i, Pa(X_i))}{Count(Pa(X_i))}$
- globally optimal maximum estimation
- requires complete data (EM-algorithm for unobserved variables.)

MAP Inference $\theta^* \in \arg\max_\theta P(\theta|data)$. Suppose $D = \{(f^{(i)}, w^{(i)}), i \in 1 : N\}$, for a new data point w, to predict the label $P(F|w,D) \simeq P(F|w,\hat{\theta})$. Can be solved via numeric methods, e.g. (mini-)SGD.

Bayesian learning $P(F|w,D) = \int P(F,\theta|w,D) d\theta = \int P(F|w,\theta) P(\theta|w,D) d\theta$

Regularize for BN deal with few samples case: (1)pseudo-count $\theta_{F=c} = \frac{Count(F=c)+\alpha_c}{N+\alpha_c+\alpha_l}$; (2) Beta prior over parameters $Beta(\theta; \alpha_c, \alpha_l)$ (equivalent)

6.2 Structure

Scoring (1) MLE score: a score of BN (G,P) is $\max_\theta \log P(D|\theta,G) = \log P(D|\theta_G, G) = N \sum_{i=1}^n \hat{I}(X_i; Pa(X_i)) + const$. According to the monotonicity of mutual information, MLE OPT is fully connected graph.

(2) BIC score: $S_{BIC}(G) = \sum_{i=1}^n \hat{I}(X_i, Pa(X_i)) - \frac{\log N}{2N}|G|$ (n:#vars,N:#training) (consistent, identify the correct structure with $N \to \infty$). NP hard.

Reduce overfitting - prior over parameters/structures;
- constraint optimization (e.g. bound #parents, e.g. as 1, the tree)
- complexity penalty: BIC score.

Approximate - use local search, may get stuck in local OPT.
- find the OPT tree. (1)For each edge $e = (X_i, X_j)$ compute $w_e = \hat{I}(X_i, X_j)$ with given data D. (2) Find the maximum spanning tree. Time complexity $O(|E| \log |E|)$ ($\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log \frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}$, with $\hat{P}(x_i, x_j) = \frac{Count(x_i, x_j)}{N}$)

7 Bayesian Learning

Principle: $p(\theta^*|X,y) \propto p(y|X, \hat{f}_{\theta^*}) p(\theta^*)$

7.1 Gaussian Process

A prior over functions $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, where $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ and $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x} - m(\boldsymbol{x}))^T)(f(\boldsymbol{x}' - m(\boldsymbol{x}'))]$. The predictive posterior for a new given point $\boldsymbol{x}^*$ is $P(\boldsymbol{y}^*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}^*) = \mathcal{N}(\mu_y^*, \Sigma_y^*)$ with $\mu_y^* = k(X, x^*)^T [K(X,X) + \sigma_y^2 \boldsymbol{I}]^{-1} \boldsymbol{y}$ and $\Sigma_y^* = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - k(X, x^*)^T [K(X,X) + \sigma_y^2 \boldsymbol{I}]^{-1} k(X, x^*) + \sigma_y^2 \boldsymbol{I}$

Uncertainty Aleatoric: noise, perfect know the model; Epistemic: uncertainty about parameters, due to lack of data.

8 Reinforcement Learning

on-policy RL: agent has full control over actions; off-policy: no control over actions.

8.1 Model-based

learning the transition model $P(s_{t+1}|s_t, a_t)$ and the value function $V(s)$.

Memory: $O(|X|^2|A|)$ for transition, $O(|X||A|)$ for reward

Time: solving MDP once: $poly(|X|, |A|, \frac{1}{\epsilon}, \log\frac{1}{\delta})$, need to do often.

Estimation $D = \{(x_i, a_i, r_i, x_{i+1})\}$, $P(X_{t+1}|X_t, A) \simeq \frac{Count(X_{t+1}, X_t, A)}{Count(X_t, A)}$, $r(x,a) \simeq \frac{1}{N_{x,a}} \sum_{t: x_t = x, a_t = a} R_t$

Approaches (1) random: explore: eventually good, exploit: bad; (2) greedy: explore: bad, stuck on suboptimal, exploit: quickly good; (3)ϵ_t greedy: $prob = \epsilon_t$:random, $1 - \epsilon_t$: greedy. (4)R_{max}: optimism in the face of uncertainty.

8.2 Model-free

learn the policy $\pi(a_t|s_t)$ or the state-action value function $Q(s_t, a_t)$. **Cheaper.**

Memory: $O(|X||A|)$ for $Q(x,a)$. Time: per iteration: pick $a_t = \arg\max Q(x,a)$ needs $O(|A|)$

Q-learning 1. initially estimate $Q(x,a)$; 2. keep updating with observation (x,a,x') and reward r: $Q(x,a) \leftarrow (1 - \alpha_t) Q(x,a) + \alpha_t (r + \gamma \max_{a'} Q(x', a'))$

Optimistic Q-learning: in initialization: $Q(x,a) = \frac{R_{max}}{1-\gamma} \prod_{t=1}^T (1 - \alpha_t)^{-1}$

Parametric Q-function $\theta^* = \arg\min L(\theta) = \sum_{(x,a,r,x')\in D} [r + \gamma \max_{a'} Q(x', a'; \theta^{old}) - Q(x, a; \theta)]^2$

DQN : two networks, use old parameters to evaluate Q function, new parameters for action selection. $L(\theta) = \sum_{(x,a,r,x')\in D} [r + \gamma Q(x', \hat{a}(x,\theta); \theta^{old}) - Q(x, a; \theta)]^2$