# ML

Alternative Loss:

$$l_{0/1} = 1_{yh_\theta(x)<0}$$
$$l_{hinge} = \max(0, 1 - yh_\theta(x))$$
$$l_{logistic} = \log(1 + e^{-yh_\theta(x)})$$
$$l_{squared} = \exp(-yh_\theta(x))$$

SVM: $\min_\theta \sum_{i=1}^n \max(0, 1 - y_i h_\theta(x_i))$

Logistic regression: $\min_\theta \sum_{i=1}^n \log(1 + e^{-y_i h_\theta(x_i)})$

# Search

admissible heuristic: $0 \le h(n) \le h^*(n)$, guarantees optimality for tree search

Graph Search: never expand a state twice

---

Consistency of Heuristic: $h(A) - h(B) \le c(A, B)$, the $f$ value along a path is non-decreasing

Define $h(s) = \text{FutureCost}^j(s)$ for some relaxed problem $j$, then $h(s)$ is consistent heuristic.

Theta* Search: if parent($x$) is visible from $y$, insert $y$ with estimate $f(y) = g(\text{parent}(x)) + c(\text{parent}(x), y) + h(y)$

weighted A*: $f(n) = g(n) + \epsilon \cdot h(n)$, $\epsilon > 1$ bias towards states that are closer to goal. cost(solution) $\le \epsilon$ cost(optimal)

# Value iteration

define value function recursively via Bellman equation

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^\pi(s')$$

so

$$V^*(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s'|s, a) V^*(s')$$
$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s'|s, a) V^*(s')$$

value iteration converges to $V^*$ because define $B\hat{V}(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s'|s, a)\hat{V}(s')$, then

$$\|BV_1 - BV_2\|_\infty = \gamma \|V_1 - V_2\|_\infty$$

hence

$$\|BV^{(n)} - V^*\|_\infty \le \gamma \|V^{(n)} - V^*\|_\infty$$

Greedy policy: $\pi_g(s) = \arg\max_a \sum_{s'} P(s'|s, a) V(s')$, if $\|V - V^*\|_\infty < \lambda$, then $|V_g - V^*| \le 2\gamma\lambda/(1-\gamma)$

# Policy iteration

Policy evaluation: $V^\pi = r + \gamma P^\pi V^\pi$, so $V^\pi = (I - \gamma P^\pi)^{-1} r$

Generalized policy iteration: any interleaving of policy evaluation and improvement $V \leftarrow V^\pi, \pi \leftarrow \pi_g$

complexity: $O(|A|^n)$ policies at most, $O(n)$ iterations required to converge

# DP

Asynchronous: in place DP, prioritized sweeping, real-time DP

Full width: every successor state is considered

Sample backup: use sample rewards and sample transitions $\langle S, A, R, S' \rangle$

Approximate DP: approximate value function using NN

# MDP extension

Ergodic Markov Process: aperiodic, recurrent, $\varrho^\pi = \lim_{T \to \infty} \frac{1}{T} \mathrm{E}[\sum_{t=1}^T P(s_t = s)]$, and

$$\tilde{v}_\pi(s) = \mathrm{E}_\pi[\sum_{k=1}^{\infty} (R_{t+k} - \varrho^\pi)|S_t = s]$$
$$= \mathrm{E}_\pi[R_{t+1} - \varrho^\pi + \tilde{v}_\pi(S_{t+1})|S_t = s]$$

# RL1

Passive Learning: learn value function given fixed policy; Active Learning: find optimal policy

# Passive RL

Passive RL: estimate $V^\pi$

- direct estimation: sample trajectory following $\pi$, model free
- Adaptive DP: model-based, estimate transition model based on observation, then use $V^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s')$, model-based

---

- TD learning: $V(s) \leftarrow V(s) + \alpha(R + \gamma V(s') - V(s))$, $s'$ is the next state, model free

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        for child-node in EXPAND(STATE[node], problem) do
            fringe ← INSERT(child-node, fringe)
        end
    end
```

# Active RL

Exploration vs Exploitation

- Greedy in the limit of infinite exploration (GLIE): select random action with probability $p(t)$, and $p(t) \to 0$ as $t \to \infty$. In practice, $p(t)$ converges to $\epsilon$.
- Boltzmann Exploration: $\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}$, where $\tau$ is the temperature parameter
- Optimistic Exploration: if $N(s, a) < N_e$, set $V(s) \leftarrow R(s) + \gamma V_{max}$, where $N(s, a)$ is the number of times action $a$ has been selected in state $s$, $N_e$ is the exploration threshold, $V_{max}$ is the maximum possible value

Rmax Algorithm:

1. Start with an optimistic model (assign largest reward to unexplored states, transition from unexplored states only to themselves)
2. solve for optimal policy
3. take greedy action
4. update optimistic estimated model (if a state becomes explored, use its true statistics)
5. repeat

TD-based active RL: take action from exploration/exploitation policy, update $V(s) = V(s) + \alpha(R + \gamma V(s') - V(s))$

Bellman constraint on optimal $Q$ function: $Q^*(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a')$

Q-learning: $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma \max_{a'} Q(s', a') - Q(s, a))$

SARSA: $Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma Q(s', a') - Q(s, a))$, where $Q$ is updated based on the next action $a'$. $a$ and $a'$ is chosen from $\epsilon$-greedy policy.

# RL2

Function approximation: $\hat{V}_\theta(s) = \theta_0 + \sum_{i=1}^d \theta_i \phi_i(s)$, where $\phi_i$ is the feature function.

---

TD-based RL for Linear Approximator: $\theta \leftarrow \theta + \alpha(R + \gamma \hat{V}_\theta(s') - \hat{V}_\theta(s)) \nabla \hat{V}_\theta(s)$, action is chosen from $\epsilon$-greedy policy.

Q-function approximation: $\hat{Q}_\theta(s, a) = \theta_0 + \sum_{i=1}^d \theta_i \phi_i(s, a)$

Q-learning with linear approximator: $\theta \leftarrow \theta + \alpha(R + \gamma \max_{a'} \hat{Q}_\theta(s', a') - \hat{Q}_\theta(s, a)) \nabla \hat{Q}_\theta(s, a)$

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

## DQN

Stable solution to deep value-based RL

1. Experience replay: store experience $\langle s, a, r, s' \rangle$ in replay buffer, sample minibatch from replay buffer
2. Fixed Target Q-network
3. Reward/Value clipping

Loss function: $L(\theta) = \mathrm{E}[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$

Double Q-Learning: select action $\epsilon$-greedy w.r.t $Q_1 + Q_2$, update $Q_1$ using $Q_2$, and vice versa

Double DQN: current $Q$ network for action selection, older $Q$ network for value estimation

$$L = (r + \gamma Q(s', \arg\max_{a} Q(s', a'; \theta); \theta^-) - Q(s, a; \theta))^2$$

Prioritized Experience Replay: DQN error $|r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)|$, stochastic prioritized $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$

Dueling network: $Q(s, a; \theta) = V(s; \theta) + A(s, a; \theta) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta)$

Multi-step Returns: $L = (r + \gamma^k \max_{a'} Q(s^{(k)}, a'; \theta^-) - Q(s, a; \theta))^2$, where $r = \sum_{i=0}^{k-1} \gamma^i r_{t+i}$

# RL3

## Policy Gradient

$$J_1 = V^{\pi_\theta}(s_1)$$
$$J_{avgV} = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$
$$J_{avgR} = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) R(s, a)$$

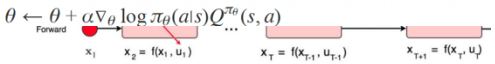
take $J = J_1, J_{avgR}, \frac{1}{1-\gamma} J_{avgV}$, policy gradient theorem:

$$\nabla_\theta J = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$$

where $\mathrm{E}_{\pi_\theta}$ is the expectation w.r.t the state distribution $d^{\pi_\theta}$, $d^{\pi_\theta}(s) = \sum_{t=1}^{\infty} \gamma^{t-1} P(s_t = s)$

Monte-Carlo Policy Gradient:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)$$




for $t = T$ to 1:

$$\mathbf{Q}_t = \mathbf{C}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{F}_t$$
$$\mathbf{q}_t = \mathbf{c}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{f}_t + \mathbf{F}_t^T \mathbf{v}_{t+1}$$
$$Q(\mathbf{x}_t, \mathbf{u}_t) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{q}_t$$
$$\mathbf{u}_t \leftarrow \arg\min_{\mathbf{u}_t} Q(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$
$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t}$$
$$\mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{q}_{\mathbf{u}_t}$$
$$\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t, \mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{K}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{K}_t$$
$$\mathbf{v}_t = \mathbf{q}_{\mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{k}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t} + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{k}_t$$
$$V(\mathbf{x}_t) = \text{const} + \frac{1}{2} \mathbf{x}_t^T \mathbf{V}_t \mathbf{x}_t + \mathbf{x}_t^T \mathbf{v}_t$$

Forward recursion

for $t = 1$ to $T$:

$$\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$
$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

## Actor-Critic

use a critic to estimate the value function, and an actor to select actions

1. sample reward $r$, transition $s' \sim P(\cdot|s, a)$, action $a' \sim \pi_\theta(\cdot|s')$
2. update critic: $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a))$
3. update actor: $\theta \leftarrow \theta + \beta \nabla_\theta \log \pi_\theta(a|s) Q(s, a)$

Reduce variance: advantage function $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$

Estimating Advantage Function:

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

and

$$E[\delta^{\pi_\theta}|s, a] = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s) = A^{\pi_\theta}(s, a)$$

so

$$\nabla_\theta J(\theta) = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s, a)] = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \delta^{\pi_\theta}]$$

In Actor critic, we have $\nabla_\theta J(\theta) = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$, however we use $\nabla_\theta J(\theta) = \mathrm{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s, a)]$ to estimate the gradient.

Unbiased if $\nabla_w Q_w(s, a) = \nabla_\theta \log \pi_\theta(a|s)$, and value function is learned with $\mathrm{E}_{\pi_\theta}[(Q_w(s, a) - Q^{\pi_\theta}(s, a))^2]$. (Set $Q_w = w^T \Phi, \pi_\theta = \theta^T \Phi$)

## Model-based RL

Deterministic Environment: Cross entropy method to compute $A = \arg\max_A J(A)$

1. sample $A_i \sim p(A)$
2. evaluate $J(A_i)$
3. pick top $k$ samples
4. update $p(A)$ to fit the top $k$ samples

Discrete case: Monte Carlo Tree Search (MCTS): Selection, Expansion, Simulation, Backpropagation

- Two policies: selection policy and rollout policy, for selection and simulation
- Upper Confidence Bound (UCB): $Q(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}}$
- Final Action Selection: max (highest weight), robust (highest visit count), both

Continuous case:

$$\min_{u_1, \ldots, u_T} \sum_{t=1}^{T} c(x_t, u_t) s.t. x_{t+1} = f(x_t, u_t)$$
$$\min_{u_1, \ldots, u_T} c(x_1, u_1) + c(f(x_1, u_1), u_2) + \cdots + c(f(f(\ldots)\ldots), u_T)$$

Linear case: $f(x_t, u_t) = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_t$, $c(x_t, u_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + c_t^T \begin{bmatrix} x_t \\ u_t \end{bmatrix}$

minimize from $u_T$ to $u_1$. If not linear, use taylor expansion to approximate the cost function.

## Adversarial Search

Alpha-Beta Search:

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, min-value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, max-value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

MCTS policies: tree policy, default policy, backup policy

Upper Confidence Bound for Trees (UCT): $R(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}}$, where $R(s, a)$ is the average reward of taking action $a$ in state $s$

Expectimax Search: compute average score under optimal play, max node as in minimax, chance node as average of children

Principle of maximum expected utility: A rational agent should chose the action that maximizes its expected utility, given its knowledge

## Probabilistic Models

Check whether two variables are independent given a set of variables -> wxb algorithm
(逆着走可以变正着走，正着走需要遇到一个黑点变成逆着走)

Sampling

- Prior Sampling: ignore evidence, sample from joint probability, do inference by counting the right samples
- Rejection Sampling: sample from joint probability, reject samples that do not match evidence
- Likelihood Weighting: when evidence is observed, sample from joint probability, weight the samples by the likelihood of evidence
- Gibbs Sampling: each time sample a variable given the rest, repeat $N$ times for each variable

HMM: given observation $e_{1:T}$, find $p(x_t|e_{1:t})$

- Forward Algorithm: $p(x_t|e_{1:t}) = \alpha p(e_t|x_t) \sum_{x_{t-1}} p(x_t|x_{t-1}) p(x_{t-1}|e_{1:t-1})$
- Time complexity: $O(TN^2)$, space complexity: $O(N)$
- Use Particle Filter to approximate the posterior distribution, each particle is a sample of the state, first sample $x' \sim P(x|x_{t-1})$, then $w \sim P(e_t|x')$, then resample particles based on the weights
- MLE: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t}) = \arg\max_{x_{1:t}} P(x_{1:t}, e_{1:t})$

Dynamic Bayesian Network: repeat a fixed Bayesian network over time, variables are connected to the previous time step