DQN

Q-LEARNING

- $Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{k} \mid s,a\right]$
- $Q^{\pi}(s,a) = \mathbb{E}_{s',a',r}[R(s,a) + \gamma Q^{\pi}(s',a') \mid s,a]$
- $Q^{*}(s,a) = \mathbb{E}_{s',r}\left[R(s,a) + \max_{a'} Q^{*}(s',a') \mid s,a\right]$

- $\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')$
- $\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha[r(s,a) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a)]$

Q-LEARNING: OFF POLICY

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using ~~policy derived from~~ Q (e.g., ε-greedy) Exploration
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$ Exploitation
    $S \leftarrow S'$
  until S is terminal

Q-LEARNING -> DQN

Pros

- Same principal but use function approximation
- Large scale problems, huge (Continuous) state space
- More flexible data(e.g. Images, signals)
- Stability boost: Replay buffer & Target network
- Efficiency boost: Output array of Qs

Issues

- Limited to discrete action space
- Bad for large action space

DQN IMPLEMENTATION

Algorithm 4 DQN

procedure DQN
  Initialize network Q_ω *and* Q_{target} *as a clone of* Q_ω
  Initialize replay buffer R *and burn in with trajectories followed by random policy*
  repeat for E **training episodes:**
    Initialise S_0
    for $t = 0, 1, \ldots, T-1$:

$$a_t = \begin{cases} \arg\max_a Q_\omega(s_t, a) & \text{with probability } 1 - \epsilon \\ \text{Random action} & \text{otherwise} \end{cases}$$

      Take a_t *and observe* r_t, s_{t+1}
      Store (s_t, a_t, r_t, s_{t+1}) *in* R
      Sample minibatch of (s_i, a_i, r_i, s_{i+1}) *with size* N *from* R

$$y_i = \begin{cases} r_i & s_{i+1} \text{ is terminal} \\ r_i + \gamma \max_a Q_{\text{target}}(s_i, a) & \text{otherwise} \end{cases}$$

$$L(\omega) = \frac{1}{N} \sum_{i=0}^{N-1} (y_i - Q_\omega(s_i, a_i))^2$$

      Update Q_ω *using* `Adam` $(\nabla_\omega L(\omega))$
      Replace Q_{target} *with current* Q_θ *if* $t \mod 50 = 0$
end procedure

Loss reduced from 2D data

ONLY update on ONE ACTION, NO CHANGE to other Qs

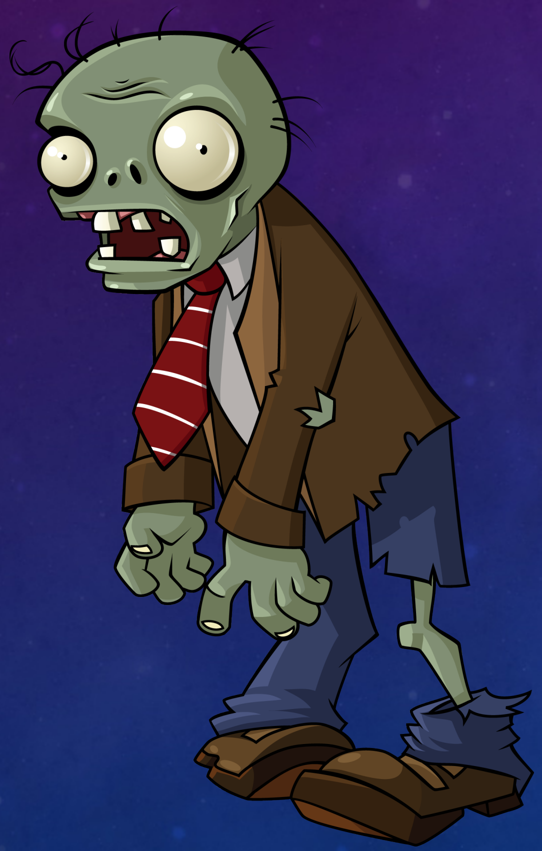
MODEL-BASED RL

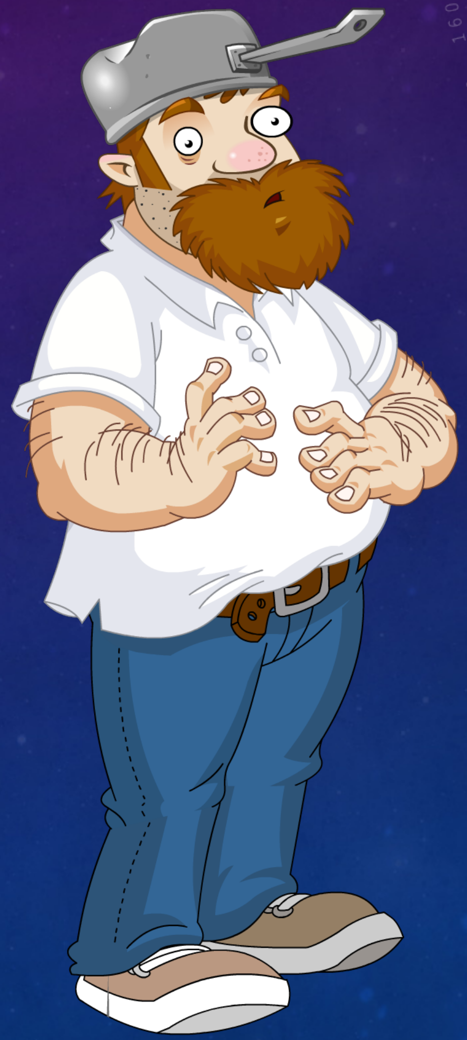
IDEA

- Learns an environment/dynamics model/network
- Function Approximation: $P_\varphi: S \times A \rightarrow R \times S'$
- Model rollout: $S_t \rightarrow A_t | \pi_\theta(S_t) \rightarrow R_t | P_\varphi(S_t, A_t) \rightarrow S_{t+1} | P_\varphi(S_t, A_t) \rightarrow \cdots$
- (Probabilistic) Ensemble networks

MBRL VS MODEL FREE: INTUITION

- Psychology: Unconscious vs Conscious
- Biology: Habitual vs Goal-directed





MBRL VS MODEL FREE

- More sample efficient(Short term)
- Compounding error(Stochasticity & Environment variety)
- Challenge: State representation & Use of prior domain knowledge

- Legged robot application: MBRL to stand and Model free RL to walk(Surrounding obstacles)