CS 747, Autumn 2022: Lecture 22

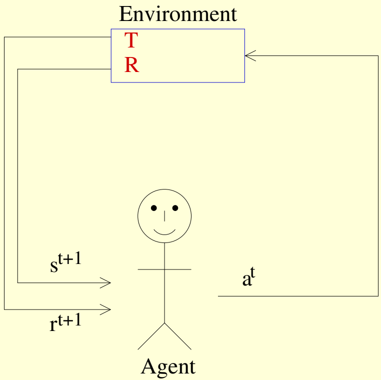
Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2022

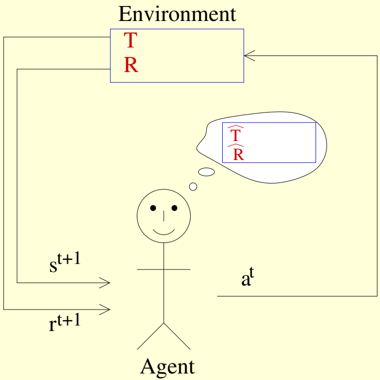
Recall What a "Model" is

- MDP (S, A, T, R, γ).



Recall What a “Model” is

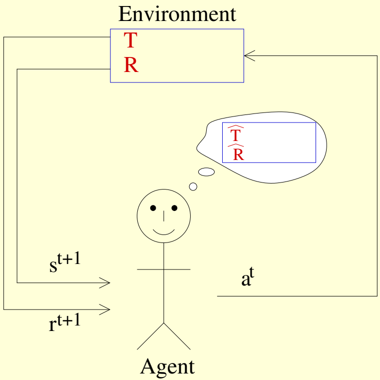
- MDP (S, A, T, R, γ).



- Model $(\hat{T}, \hat{R})$ is agent’s estimate of (T, R).

Recall What a “Model” is

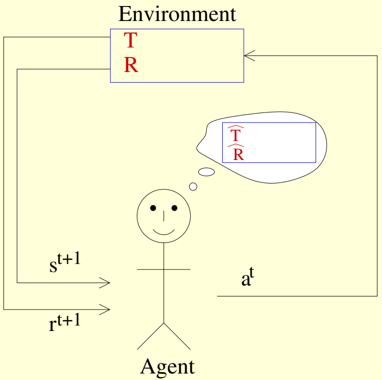
- MDP (S, A, T, R, γ).



- Model $(\hat{T}, \hat{R})$ is agent’s estimate of (T, R).
- Distributional models store $T(s, a, s')$ for $s, s' \in S, a \in A$.

Recall What a "Model" is

- MDP (S, A, T, R, γ).



- Model $(\hat{T}, \hat{R})$ is agent's estimate of (T, R).
- Distributional models store $T(s, a, s')$ for $s, s' \in S, a \in A$.
- Sample models generate $s' \sim T(s, a)$ for $s \in S, a \in A$.

Models in RL

1. Dyna-Q algorithm
2. Model-based RL for helicopter control

Models in RL

1. Dyna-Q algorithm
2. Model-based RL for helicopter control

Learning and Using Models

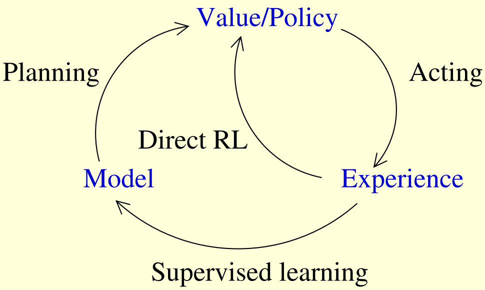


Figure from Section 8.2, Sutton and Barto (2018).

Learning and Using Models

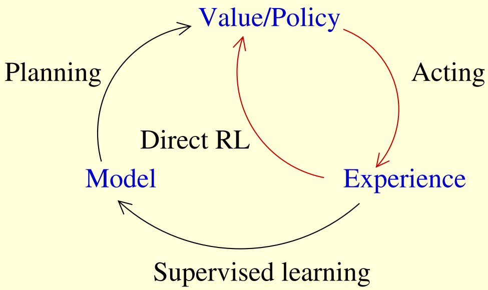


Figure from Section 8.2, Sutton and Barto (2018).

Learning and Using Models

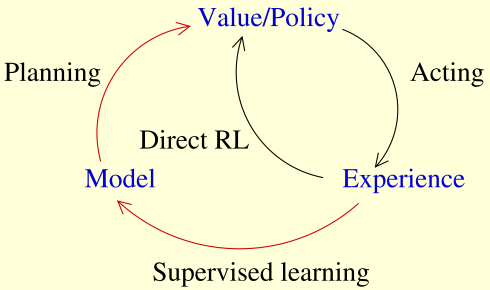


Figure from Section 8.2, Sutton and Barto (2018).

Learning and Using Models

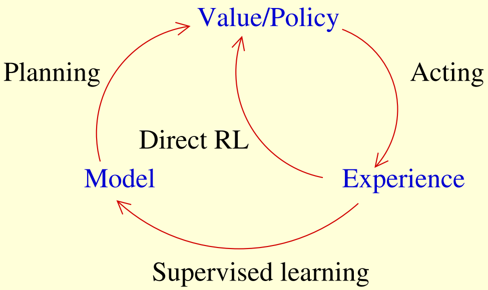


Figure from Section 8.2, Sutton and Barto (2018).

Learning and Using Models

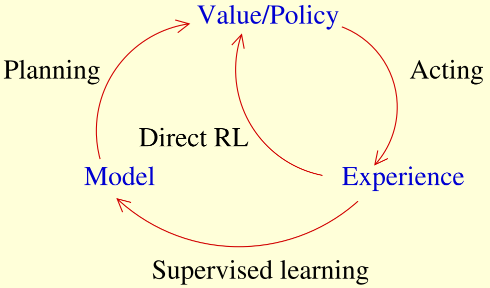


Figure from Section 8.2, Sutton and Barto (2018).

- What are pluses and minuses of model-based learning?

Learning and Using Models

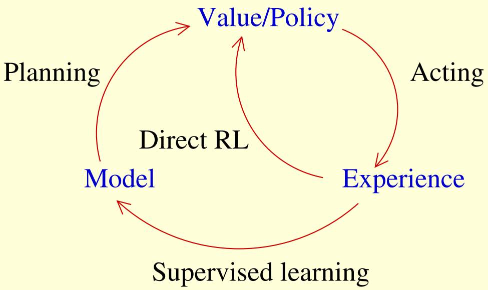


Figure from Section 8.2, Sutton and Barto (2018).

- What are pluses and minuses of model-based learning?
- \+ Fewer environmental interactions (but more computation).
- \+ Adapting to changes in the environment.
- \- Being misled by an incorrect/biased model.

Dyna-Q Algorithm

Initialise Q, *Model*.
Loop forever:
  $s \leftarrow$ current state.
  $a \leftarrow \epsilon$-greedy(s, Q).
  Take action a; get next state s', reward r.
  $Q(s, a) \leftarrow Q(s, a) + \alpha\{r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)\}$.
  $UpdateModel(Model, s, a, r, s')$.
  Loop N times:
    $\bar{s} \leftarrow$ Random previously observed state.
    $\bar{a} \leftarrow$ Random previously taken action from $\bar{s}$.
    $\bar{s}', \bar{r} \sim Model(\bar{s}, \bar{a})$.
    $Q(\bar{s}, \bar{a}) \leftarrow Q(\bar{s}, \bar{a}) + \alpha\{\bar{r} + \gamma \max_{\bar{a}' \in A} Q(\bar{s}', \bar{a}') - Q(\bar{s}, \bar{a})\}$.

Dyna-Q Algorithm

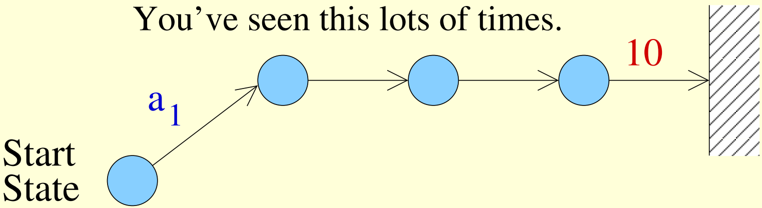
Initialise Q, $Model$.
Loop forever:
    $s \leftarrow$ current state.
    $a \leftarrow \epsilon$-greedy(s, Q).
    Take action a; get next state s', reward r.
    $Q(s, a) \leftarrow Q(s, a) + \alpha\{r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)\}$.
    $UpdateModel(Model, s, a, r, s')$.
    Loop N times: //Simulation using model.
        $\bar{s} \leftarrow$ Random previously observed state.
        $\bar{a} \leftarrow$ Random previously taken action from $\bar{s}$.
        $\bar{s}', \bar{r} \sim Model(\bar{s}, \bar{a})$.
        $Q(\bar{s}, \bar{a}) \leftarrow Q(\bar{s}, \bar{a}) + \alpha\{\bar{r} + \gamma \max_{\bar{a}' \in A} Q(\bar{s}', \bar{a}') - Q(\bar{s}, \bar{a})\}$.

Dyna-Q Algorithm

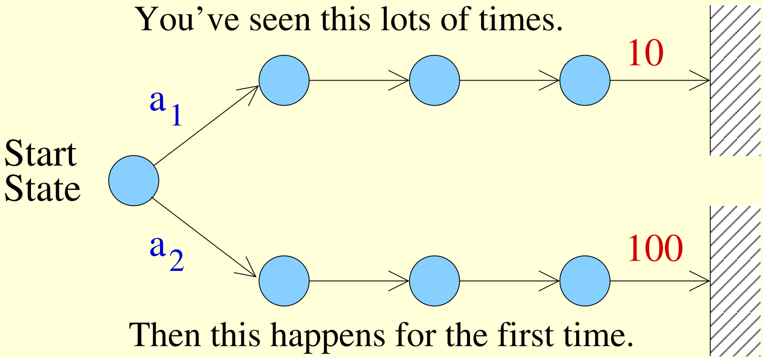
Initialise Q, *Model*.
Loop forever:
    $s \leftarrow$ current state.
    $a \leftarrow \epsilon$-greedy(s, Q).
    Take action a; get next state s', reward r.
    $Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma \max_{a' \in A} Q(s', a') - Q(s,a)\}$.
    $UpdateModel(Model, s, a, r, s')$.
    Loop N times: //Simulation using model.
        $\bar{s} \leftarrow$ Random previously observed state.
        $\bar{a} \leftarrow$ Random previously taken action from $\bar{s}$.
        $\bar{s}', \bar{r} \sim Model(\bar{s}, \bar{a})$.
        $Q(\bar{s},\bar{a}) \leftarrow Q(\bar{s},\bar{a}) + \alpha\{\bar{r} + \gamma \max_{\bar{a}' \in A} Q(\bar{s}', \bar{a}') - Q(\bar{s},\bar{a})\}$.

In practice, model and Q implemented using function approximator, rules.

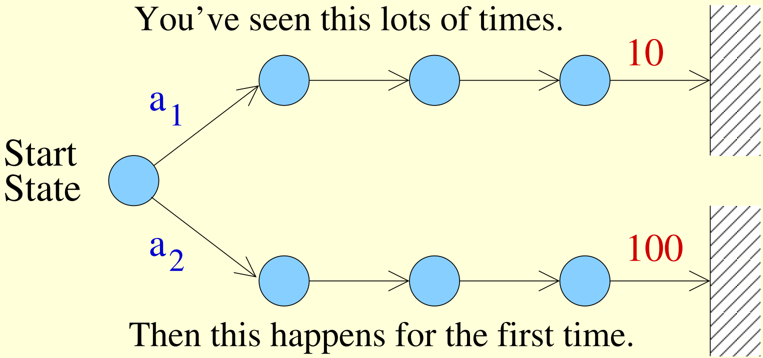
Effect of Model



Effect of Model



Effect of Model



- Models can lead to more efficient exploration.
- Model uncertainties can also be maintained.
- Dyna-Q can be augmented with prioritised sweeping to expedite reconciliation of Q-function with model.

Models in RL

1. Dyna-Q algorithm
2. Model-based RL for helicopter control

Models in RL

1. Dyna-Q algorithm
2. Model-based RL for helicopter control

   Autonomous helicopter flight via Reinforcement Learning.

   Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry, Advances in Neural Information Processing Systems 16, pp. 799-806, MIT Press, 2003.

Controlling a Helicopter

[1]

- State described by position (x, y, z), orientation (ϕ, θ, ω), velocity $(\dot{x}, \dot{y}, \dot{z})$, and angular velocity $(\dot{\phi}, \dot{\theta}, \dot{\omega})$.
- Actions: 4-dimensional control of rotor tilts, speeds.
- Task: hover in place, or follow a trajectory.

1. https://www.publicdomainpictures.net/pictures/20000/velka/police-helicopter-8712919948643Mk.jpg.

Controlling a Helicopter

[1]

- Episodic or continuing? What are T, R, γ?

1. https://www.publicdomainpictures.net/pictures/20000/velka/police-helicopter-8712919948643Mk.jpg.

Controlling a Helicopter

[1]

- Episodic or continuing? What are T, R, γ?
- How to learn to fly? By trial and error?!

1. `https://www.publicdomainpictures.net/pictures/20000/velka/police-helicopter-8712919948643Mk.jpg`.

Approach of Ng et al. (2003)

- Have a human pilot fly the helicopter; record trajectory.
- Learn a model using supervised learning on gathered data.
- Run policy search on the model.
- Evaluate learned policy on (real) helicopter.

Data Gathering

- Human pilot flies helicopter for a few minutes.
- $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots$ trajectory recorded at 50Hz.
- Trajectory split into separate train (339s) and test (140s) segments.
- Domain knowledge applied to simplify model learning (use of body coordinates, accounting for symmetries, etc.).

Learning the Model

- Given query x, output y is computed as a linear function of state features as well as actions:

$$y = \beta x + \eta,$$

  where parameters β and η (noise) are determined mainly by training points in the vicinity of x.

- Example of an instance-based approach yielding a non-linear, distributional model, which is subsequently used as a sample model.

- Some parameters hard-coded based on domain knowledge.

- Design and choices validated by visualising divergence between predicted and actual trajectories.

Policy Search

- Policy template: feed-forward neural networks with state (and derived) features as input, and one output for each of four action dimensions ($[-1, 1]$). Few tens of parameters.
- For given policy π, define $U(\pi)$ to be the expected long-term reward from start state. Need to find

$$\operatorname*{argmax}_{\pi \in \Pi} U(\pi).$$

- Instead find $\operatorname{argmax}_{\pi \in \Pi} \hat{U}(\pi)$, estimated using rollouts of π on model.
- Search based on hill-climbing or gradient ascent.
- "PEGASUS" trick used to reduce variance across rollouts.

Hovering, Trajectory-following

- Hovering at $(x^\star, y^\star, z^\star)$:

$$
\begin{aligned}
R(s, a) &= R(s) + R(a), \text{ where} \\
R(s) &= -[\alpha_x (x - x^\star)^2 + \alpha_y (y - y^\star)^2 + \alpha_z (z - z^\star)^2 + \\
&\qquad \alpha_{\dot{x}} \dot{x}^2 + \alpha_{\dot{y}} \dot{y}^2 + \alpha_{\dot{z}} \dot{z}^2 + \alpha_{\dot{\omega}} \dot{\omega}^2], \\
R(a) &= -[\alpha_{a_1} (a_1)^2 + \alpha_{a_2} (a_2)^2 + \alpha_{a_3} (a_3)^2 + \alpha_{a_4} (a_4)^2].
\end{aligned}
$$

Hovering, Trajectory-following

- Hovering at $(x^\star, y^\star, z^\star)$:

$$
\begin{aligned}
R(s, a) &= R(s) + R(a), \text{ where} \\
R(s) &= -[\alpha_x(x - x^\star)^2 + \alpha_y(y - y^\star)^2 + \alpha_z(z - z^\star)^2 + \\
&\qquad \alpha_{\dot{x}}\dot{x}^2 + \alpha_{\dot{y}}\dot{y}^2 + \alpha_{\dot{z}}\dot{z}^2 + \alpha_{\dot{\omega}}\dot{\omega}^2], \\
R(a) &= -[\alpha_{a_1}(a_1)^2 + \alpha_{a_2}(a_2)^2 + \alpha_{a_3}(a_3)^2 + \alpha_{a_4}(a_4)^2].
\end{aligned}
$$

- Flying along trajectory $(x_t^\star, y_t^\star, z_t^\star)_{t=0}^{T}$:
  - "Obvious" idea of using $(x_t^\star, y_t^\star, z_t^\star)$ in place of $(x^\star, y^\star, z^\star)$ can be problematic.
  - Instead decouple deviation and progress.
  - Uses more parameters/connections in neural network-based policy than for hovering.

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \to A$ mapping using supervised learning?

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \rightarrow A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \rightarrow A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.
- (Tabular) model has $\theta(|S|^2|A|)$ float entries, Q function has $\theta(|S||A|)$ float entries, policy has $\theta(|S| \log |A|)$ bits.

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \to A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.
- (Tabular) model has $\theta(|S|^2|A|)$ float entries, Q function has $\theta(|S||A|)$ float entries, policy has $\theta(|S| \log |A|)$ bits.
- Batch RL and model-based RL related. Both involve more computation, but typically improve sample complexity.

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \to A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.
- (Tabular) model has $\theta(|S|^2|A|)$ float entries, Q function has $\theta(|S||A|)$ float entries, policy has $\theta(|S| \log |A|)$ bits.
- Batch RL and model-based RL related. Both involve more computation, but typically improve sample complexity.
- Models can benefit from domain knowledge (physics, games, etc.).

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \to A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.
- (Tabular) model has $\theta(|S|^2|A|)$ float entries, Q function has $\theta(|S||A|)$ float entries, policy has $\theta(|S| \log |A|)$ bits.
- Batch RL and model-based RL related. Both involve more computation, but typically improve sample complexity.
- Models can benefit from domain knowledge (physics, games, etc.).
- Gaussian Processes often used for model-learning in many robotic tasks (where samples are expensive).

Discussion

- Why not imitate the human pilot's policy: that is, learn $S \to A$ mapping using supervised learning?

  Sometimes works! Sometimes does not.
- (Tabular) model has $\theta(|S|^2|A|)$ float entries, Q function has $\theta(|S||A|)$ float entries, policy has $\theta(|S| \log |A|)$ bits.
- Batch RL and model-based RL related. Both involve more computation, but typically improve sample complexity.
- Models can benefit from domain knowledge (physics, games, etc.).
- Gaussian Processes often used for model-learning in many robotic tasks (where samples are expensive).

> *"Essentially, all models are wrong, but some are useful."*
>
> —George Box