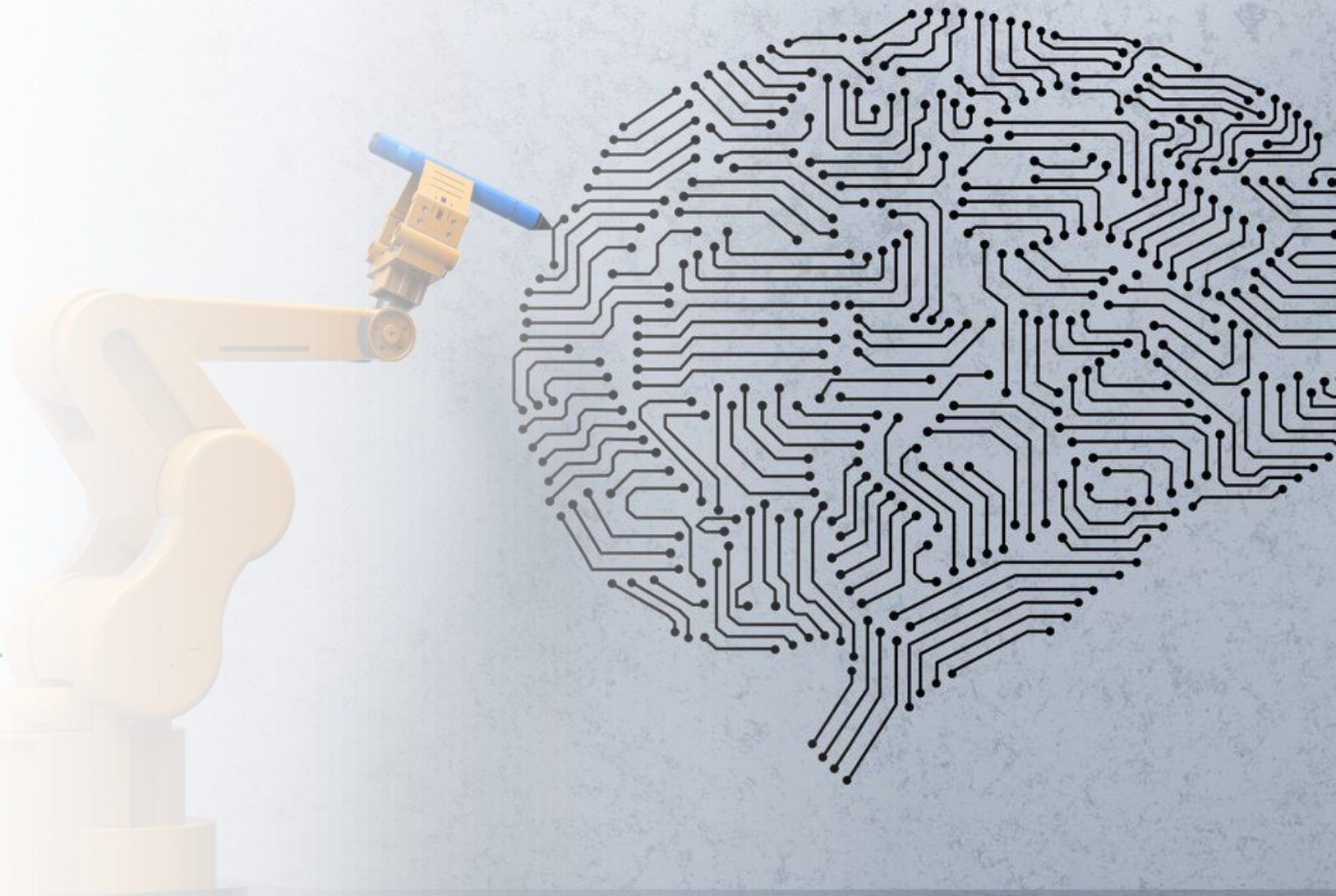
Aprendizaje por refuerzo

Clase 9: Integrando aprendizaje y planeación



Para el día de hoy…

- Aprendizaje por refuerzo basado en modelo
- Arquitecturas
- Búsqueda basada en simulación

Tarea 2

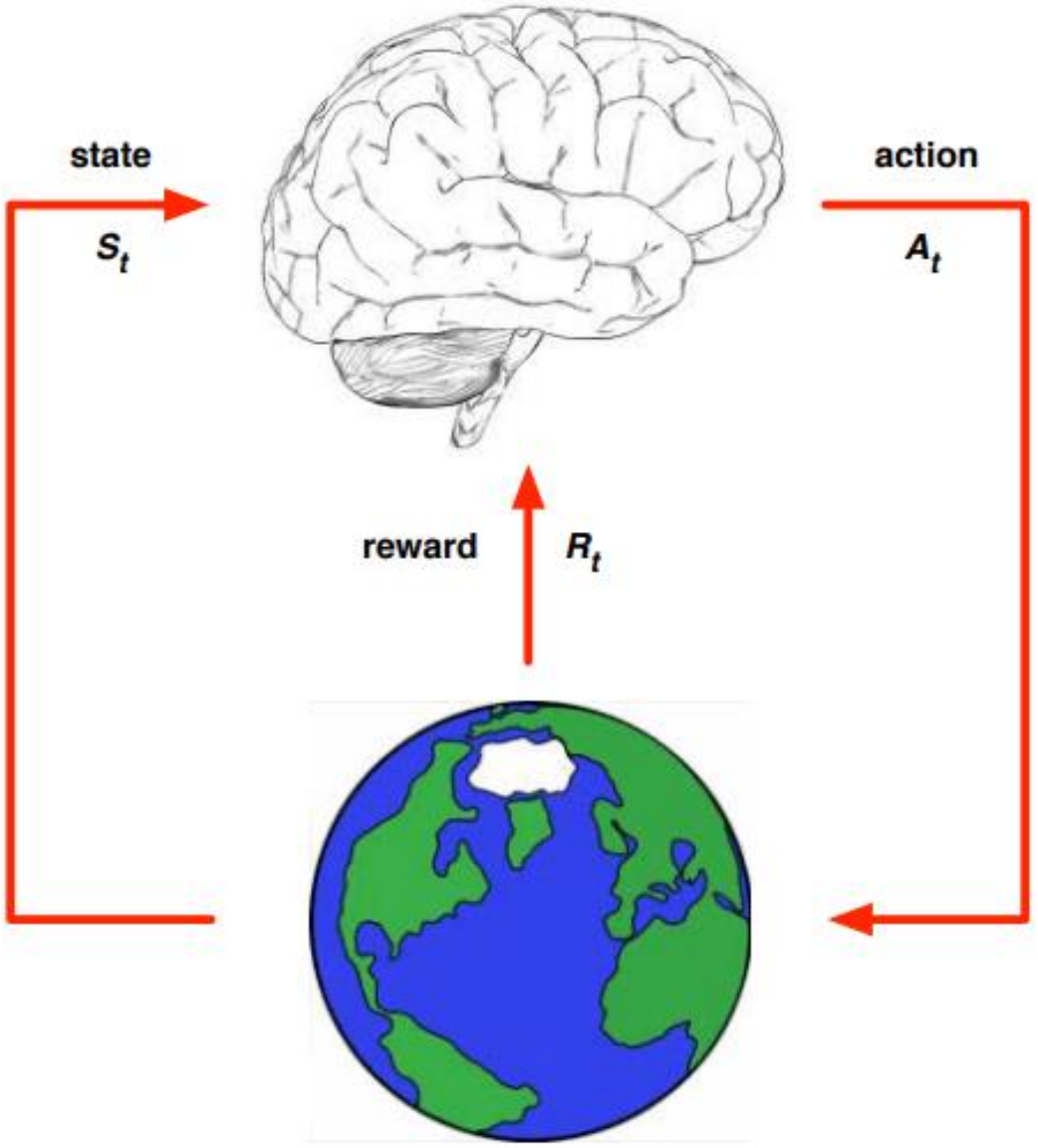
- Implementar en Python
  - (20 pts) MC
  - (20 pts) SARSA λ
  - (20 pts) Aprendizaje Q
- Código para entrenar y probar algoritmos en
  - (10 pts) Tic tac toe con oponente aleatorio
  - (10 pts) Blackjack (https://gymnasium.farama.org/environments/toy_text/blackjack/)
  - (10 pts) Mountain car (https://gymnasium.farama.org/environments/classic_control/mountain_car_continuous/) (Con función de aproximación lineal en batch)
  - (10 pts) CartPole (https://gymnasium.farama.org/environments/classic_control/cart_pole/)
- Reportar (puntos negativos en caso de no entregar)
  - (-20 pts) Parámetros: número de episodios, α, política utilizada, otros parámetros relevantes
  - (-20 pts) Entrenamiento: gráfica de convergencia episodios vs recompensa obtenida para cada algoritmo
  - (-20 pts) Prueba: gráfica de convergencia episodios vs recompensa obtenida para cada algoritmo para 10 experimentos

RL basado en modelo

- En las últimas estudiamos como utilizar la experiencia para
  - Aprender las funciones de valor $\hat{v}(s, w)$ y acción $\hat{q}(s, a, w)$
  - Aprender la política $\hat{\pi}(s, a, \theta)$
- Ahora aprenderemos el modelo directamente de la experiencia
- También a usarlo para planear
- E… integrarlo en una sola arquitectura

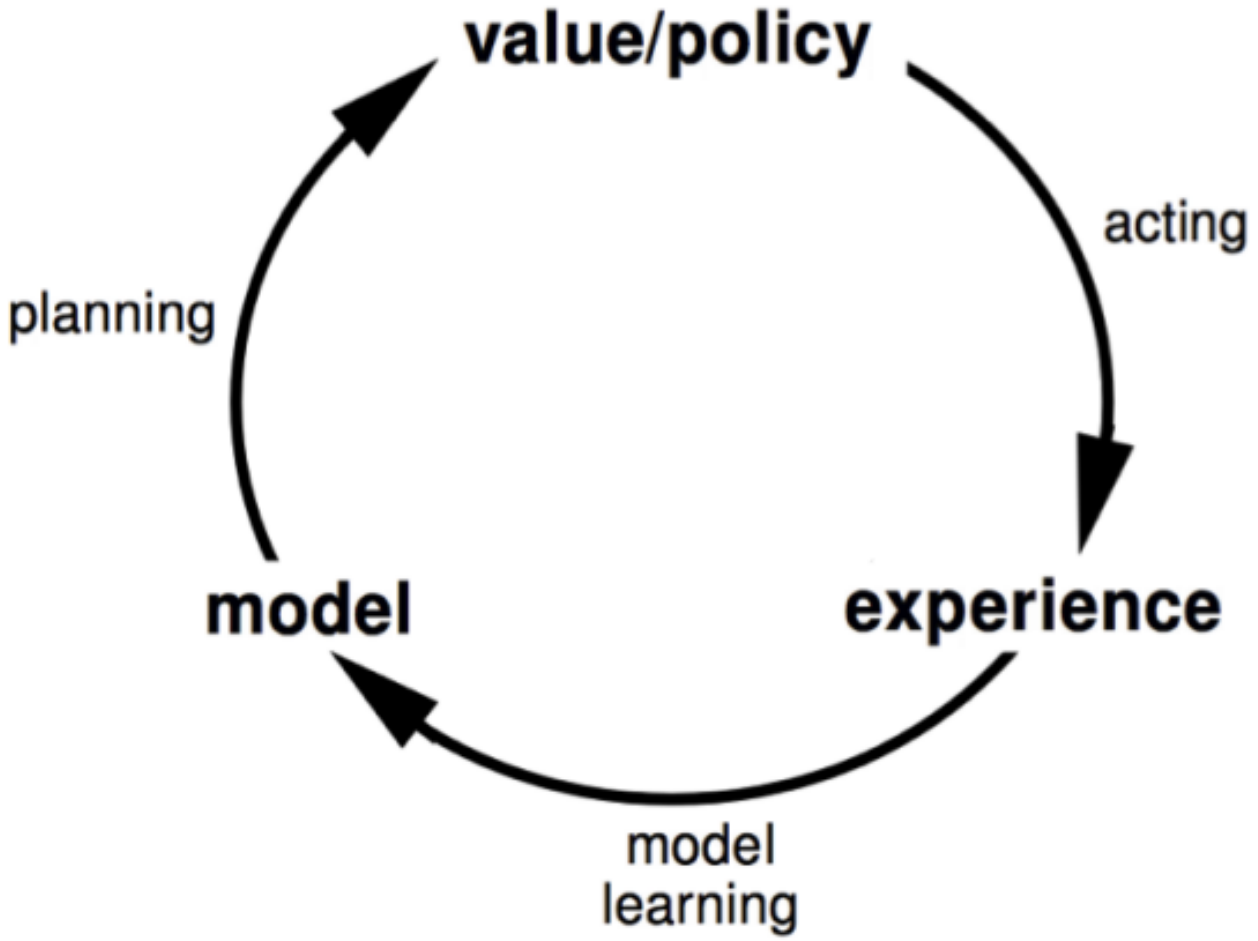
RL basada en modelo y libre de modelo

- RL basada en modelo
  - Aprende el modelo de la experiencia
  - Planea función valor y/o política a partir del modelo
- RL libre de modelo
  - No existe un modelo
  - Aprende la función de valor y/o política a partir de la experiencia



RL basa en modelo

- Ventajas
  - Puede ser aprendido eficientemente por métodos de aprendizaje supervisado
  - Puede razonar acerca de la incertidumbre del modelo
- Desventajas
  - Primero aprende el modelo, después construye una función de valor
  - Existen dos fuentes de error



¿Qué es un modelo?

- Un modelo $\mathcal{M}$ es una representación de un MDP $(\mathcal{S}, \mathcal{A}, \hat{p})$ alternativamente$(\mathcal{S}, \mathcal{A}, \hat{p}, \hat{r})$
  - Por ahora, los estados y acciones son los mismo que los del problema real
  - La dinámica $\hat{p}_\eta$ se encuentra parametrizada por pesos η
  - El modelo aproxima las transiciones y recompensas $\hat{p}_\eta \approx p$

$$R_{t+1}, S_{t+1} \sim \hat{p}_\eta(r, s' | S_t, A_t)$$

  - Típicamente suponemos

$$\mathbb{P}[S_{t+1}, R_{t+1} | S_t, A_t] = \mathbb{P}[S_{t+1} | S_t, A_t] \mathbb{P}[R_{t+1} | S_t, A_t]$$

Aprendizaje del modelo

- Objetivo: estimar el modelo $\mathcal{M}_\eta$ a partir de experiencias $\{S_1, A_1, R_2, \dots, S_T\}$
- Este es un problema supervisado

$$S_1, A_1 \to R_2, S_2$$
$$S_1, A_1 \to R_2, S_2$$
$$\vdots$$
$$S_{T-1}, A_{T-1} \to R_T, S_T$$

- Aprender $s, a \to r$ es un problema de regresión
- Aprender $s, a \to s'$ es un problema de estimación de densidad
- Elegir alguna función de pérdida
- Encontrar los parámetros η que minimicen la pérdida empírica

Ejemplos de modelos

- Normalmente se descompone la dinámica p_η en funciones separadas para transiciones y recompensas
- Para cada una se puede considerar
  - Modelo tabular
  - Modelo de esperanza lineal
  - Procesos Gaussianos
  - ⋮

Modelos tabulares

- Es un MDP explicito
- Cuenta las visitas $N(s,a)$ para cada par estado acción

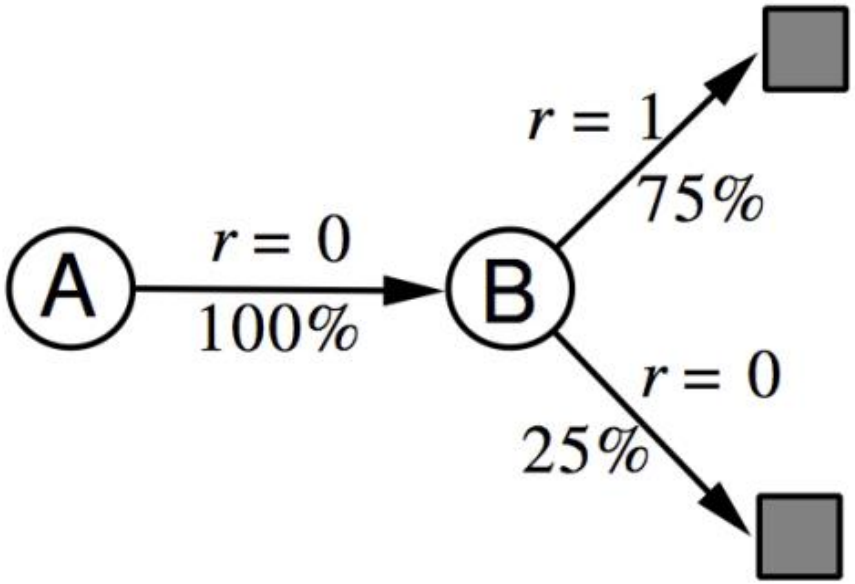
$$\hat{p}_t(s'|s,a) = \frac{1}{N(s,a)} \sum_{k=0}^{t-1} 1(S_k = s, A_k = a, S_{k+1} = s')$$

$$\mathbb{E}_{\hat{p}_t}[R_{t+1}|S_t = s, A_t = a] = \frac{1}{N(s,a)} \sum_{k=0}^{t-1} 1(S_k = s, A_k = a) R_{k+1}$$

Ejemplo

Datos dos estados A, B; $\gamma = 1$; 8 episodios de experiencia

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



Planeación con el modelo

- Dado un modelo $\mathcal{M}_\eta$
- Resolver el MDP
- Utilizando
  - Iteración de valor
  - Iteración de política
  - Búsqueda en árbol
  - ...

Planeación basada en muestras

- Utilizar el modelo solo para generar muestras
- Generamos experiencia a partir de

$$S, R \sim \hat{p}_\eta(\cdot, s, a)$$

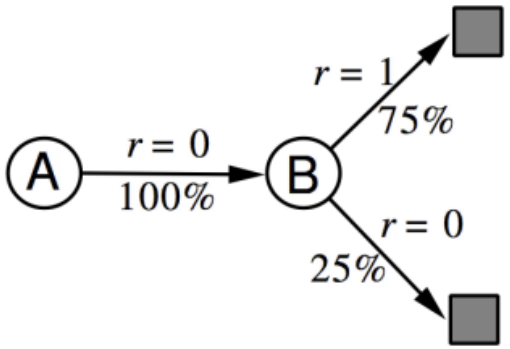
- Aplicar RL libre de modelo
  - Monte-Carlo control
  - SARSA
  - Q-learning

De regreso a nuestro ejemplo

- Construir un modelo tabular a partir de la experiencia real
- Aplicar RL libre de modelo a partir de la experiencia muestreada
- Con MC
  - $V(A) = 1$
  - $V(B) = 0.75$

Real experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



Sampled experience

B, 1
B, 0
B, 1
A, 0, B, 1
B, 1
A, 0, B, 1
B, 1
B, 0

Planeando con un modelo inexacto

Dado un modelo imperfecto $\hat{p}_\eta \neq p$

- El proceso de planeación puede calcular una política subóptima
- El desempeño está limitado a la política optima para el MDP aproximado $(\mathcal{S}, \mathcal{A}, \hat{p}_\eta)$
- RL basada en modelo es solo tan buena como el modelo estimado

¿Cómo podemos lidiar con esos problemas inevitables?

- Idea 1: cuando el modelo esté equivocado, usar RL libre de modelo
- Idea 2: razonar sobre la incertidumbre sobre η (métodos Bayesianos)
- Idea 3: combinar métodos basados en modelo y libres de modelo

Experiencia real y simulada

- Consideramos dos fuentes de experiencia
- Experiencia real: muestreada en el ambiente

$$r, s' \sim p$$

- Experiencia simulada: muestreada del modelo

$$r, s' \sim \hat{p}_\eta$$

Integrando aprendizaje y planeación

RL basada en modelo

- Aprende el modelo de la experiencia
- Planea función valor y/o política a partir del modelo

RL libre de modelo

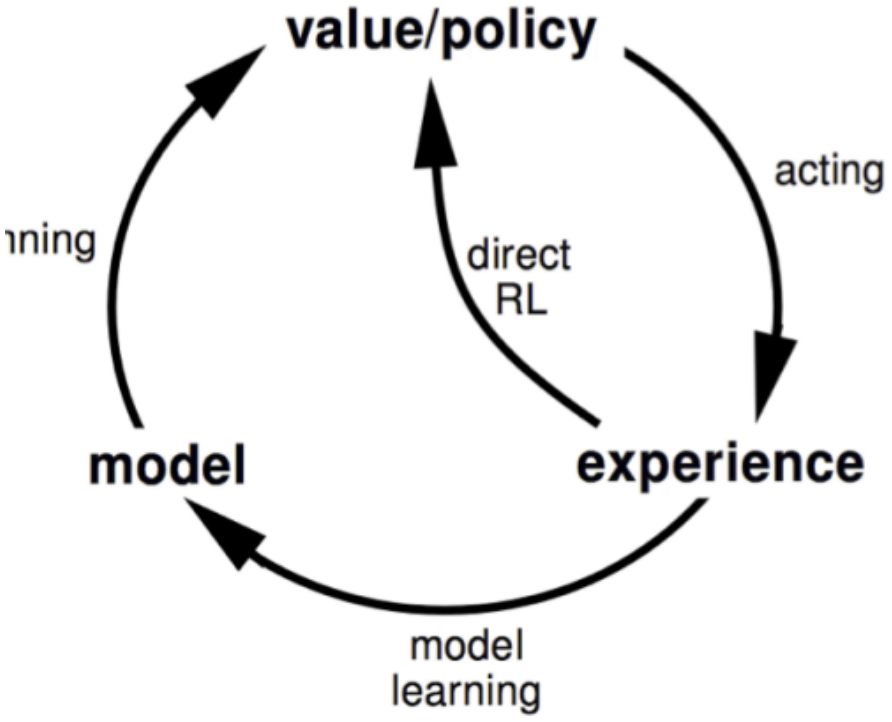
- No existe un modelo
- Aprende la función de valor y/o política a partir de la experiencia

Dyna

- Aprender el modelo de experiencia real
- Aprender y planear la función de valor y/o política de la experiencia real y simulada
- Tratar las experiencias reales y simuladas de forma equivalente. Conceptualmente, las actualizaciones no son distinguibles

Algoritmo Dyna-Q

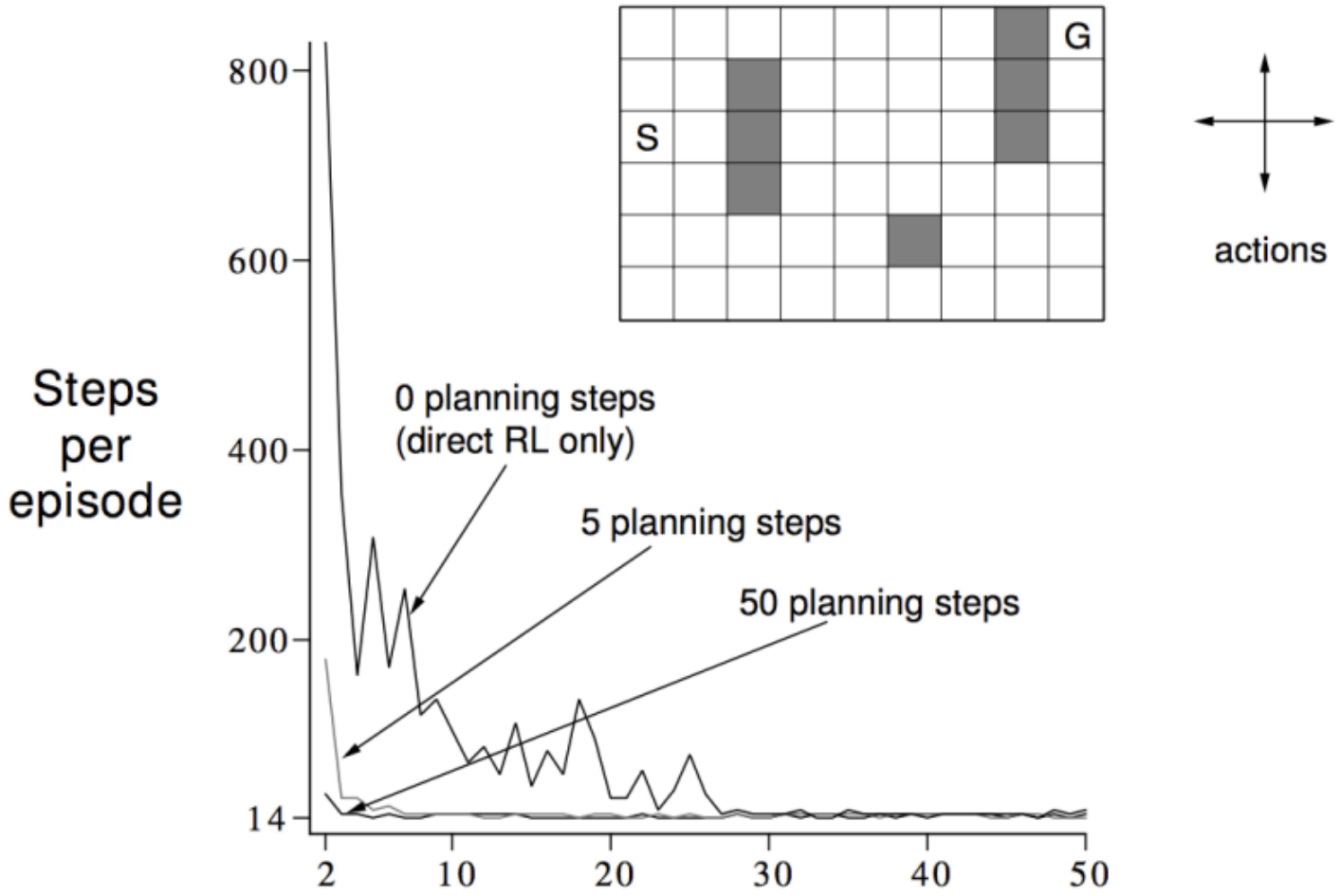
- Podemos realizar más cálculos para aprender más eficientemente
- Importante cuando recolectar datos es
  - Caro/lento
  - Peligroso



Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

¿Y funciona?


G
S
actions
Steps per episode
800
600
400
200
14
2
10
20
30
40
50
0 planning steps
(direct RL only)
5 planning steps
50 planning steps

Modelos paramétricos

vs

repetición de experiencia l

Modelos paramétricos vs repetición de experiencia II

Cómputo

- Consultar repetición de experiencia es muy barato
- Generar una muestra del modelo aprendido puede ser caro

Memoria

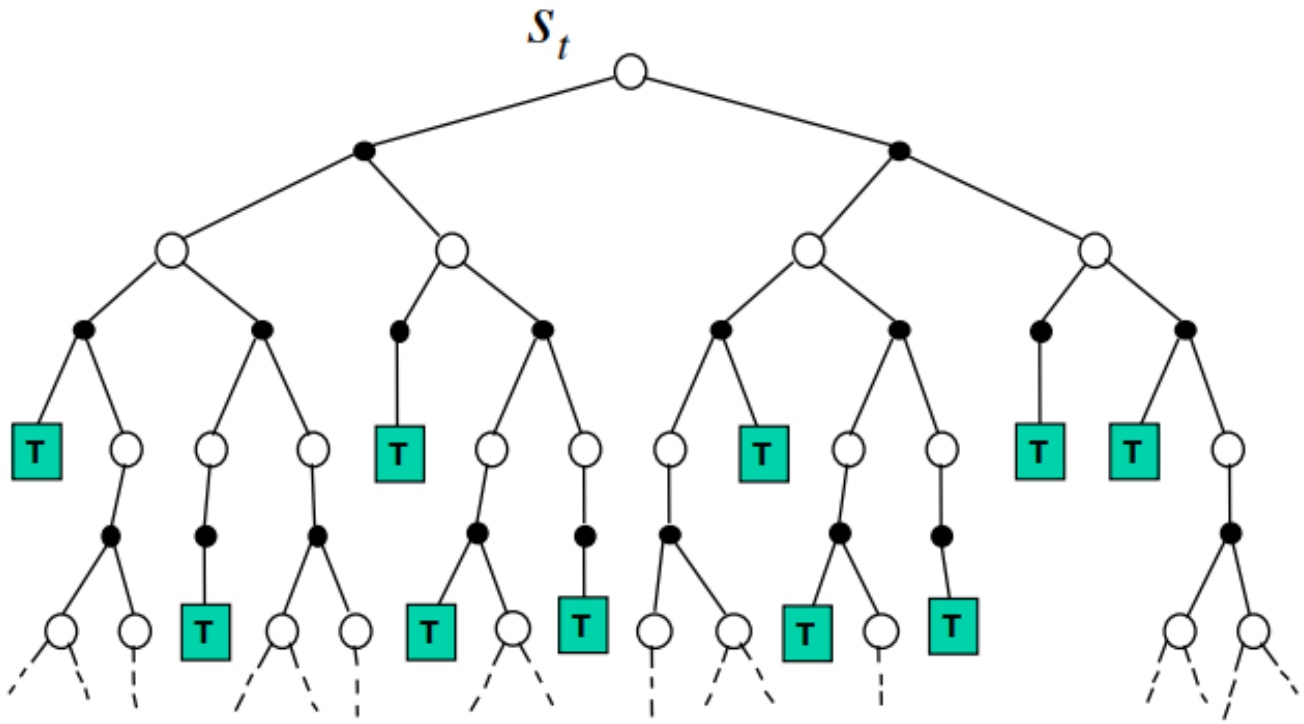
- La repetición de memoria escala linealmente con su capacidad
- Un modelo paramétrico tiene memoria constante

Planeación para selección de acción

- Hemos considerado el caso donde la planeación se usa para mejorar la función de valor global
- Ahora consideraremos planeación en el futuro cercano para seleccionar la siguiente acción
- El agente podría hacer una función de valor local
- La inexactitud del modelo puede resultar en exploración en lugar de malas actualizaciones

Búsqueda hacia adelante

- Los algoritmos de búsqueda hacia adelante seleccionan la mejor acción "viendo hacia adelante"
- Construyen un árbol de búsqueda con el estado actual s_t en la raíz
- Usan el modelo del MDP para ver hacia adelante
- No hay necesidad de resolver todo el MDP, solo lo que parte de s_t



Predicción vía simulación de Monte-Carlo

- Dado un modelo paramétrico $\mathcal{M}_\eta$ y una política π
- Simula K episodios de experiencia iniciando en el ahora S_t

$$\left\{S_t^k = S_t, A_t^k, R_{t+1}^k, \dots, S_t^K\right\}_{k=1}^K \sim \hat{p}_\eta, \pi$$

- Evaluar el estado por medio de la media del retorno

$$v(S_t) = \frac{1}{K}\sum_{k=1}^{K} G_t^k \approx v_\pi(S_t)$$

Control vía simulación de Monte-Carlo

- Dado un modelo paramétrico $\mathcal{M}_\eta$ y una política π
- Simula K episodios de experiencia iniciando en el ahora s

$$\left\{S_t^k = s, A_t^k, R_{t+1}^k, \ldots, S_t^K\right\}_{k=1}^K \sim \hat{p}_\eta, \pi$$

- Evaluar las acciones por medio de la media del retorno

$$q(s,a) = \frac{1}{K}\sum_{k=1}^{K} G_t^k \approx q_\pi(s,a)$$

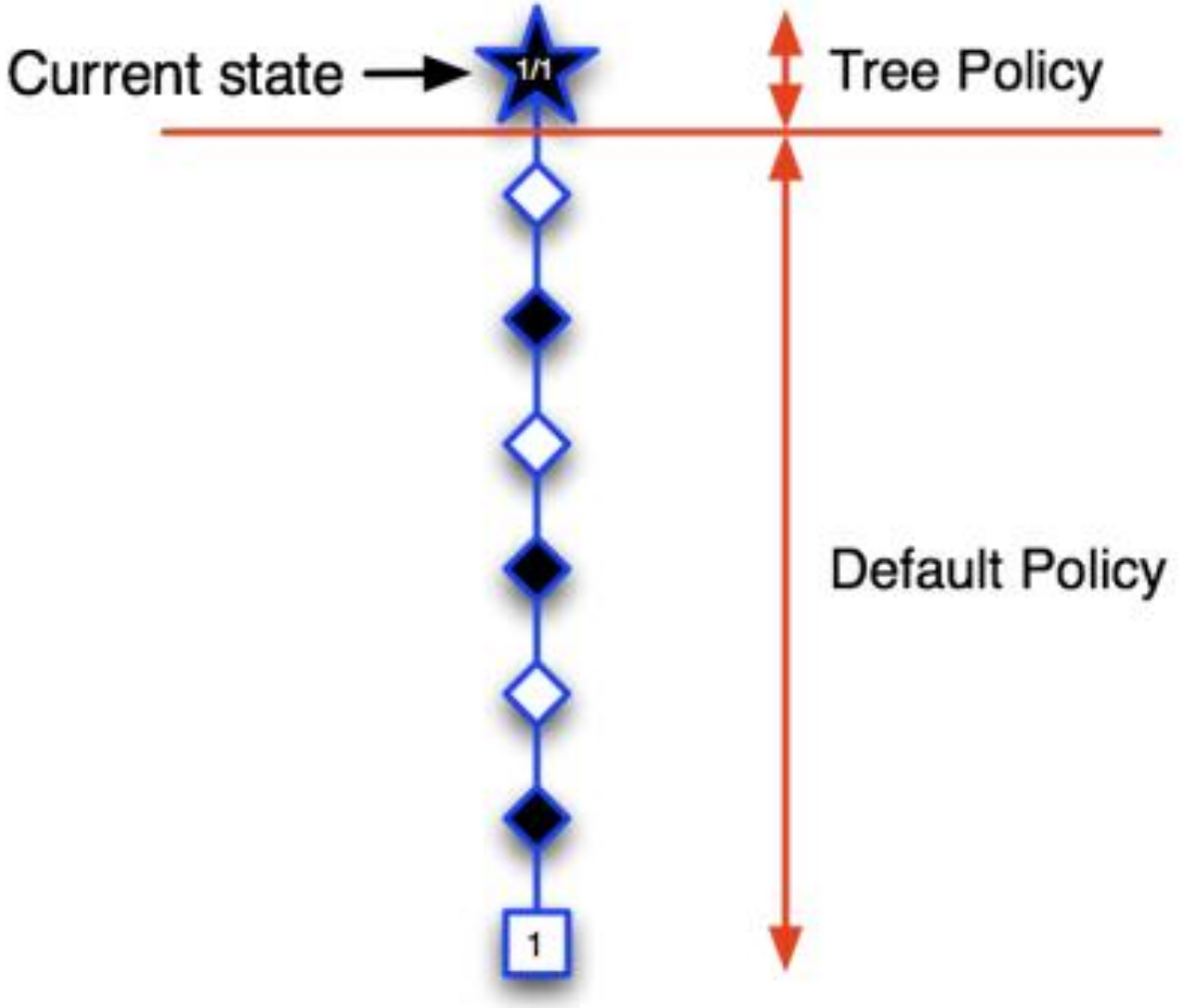
- Seleccionar la acción real con el valor máximo

$$A_t = \arg\max_{a \in \mathcal{A}} q(S_t, a)$$

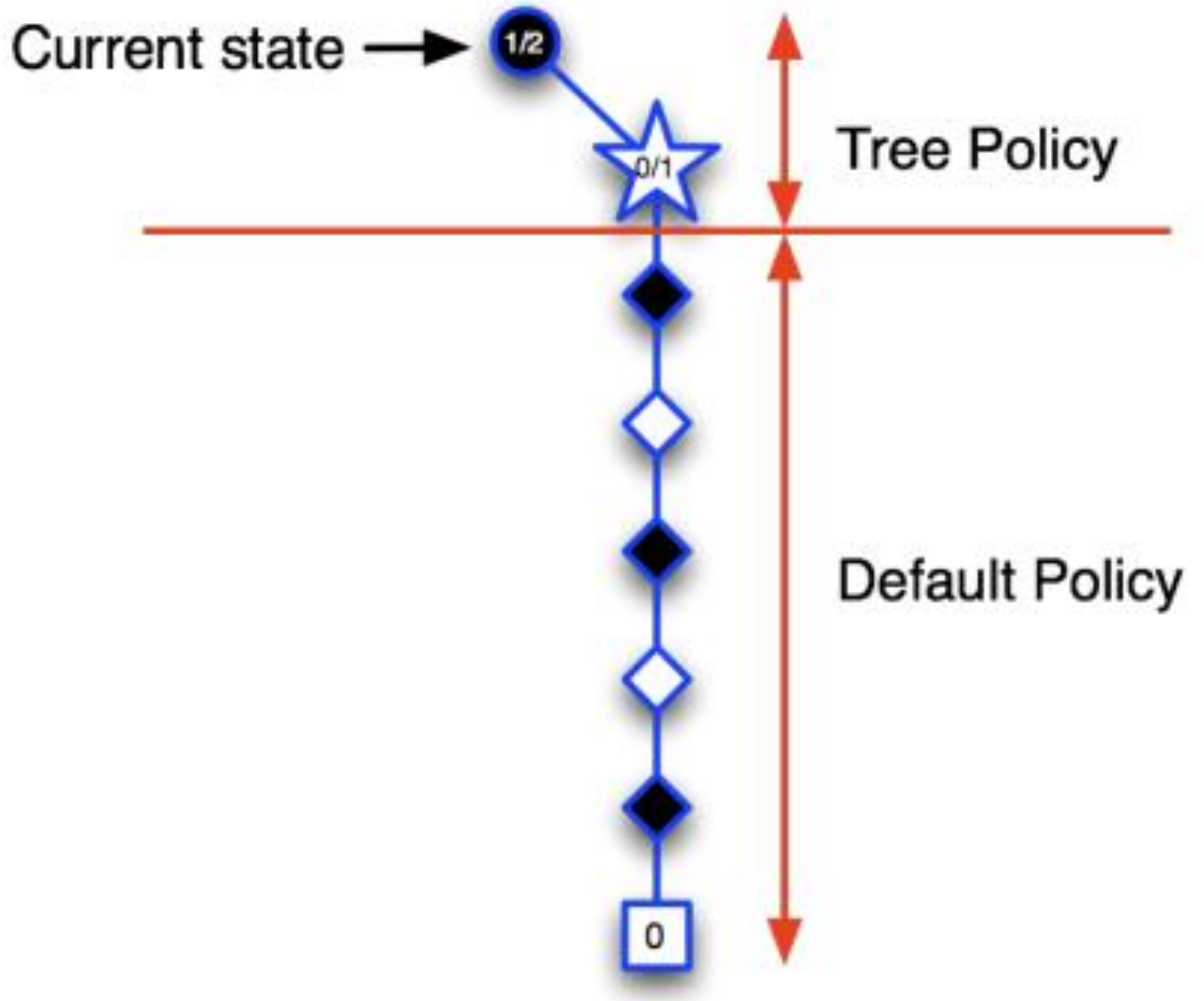
Monte-Carlo Tree Search (MCTS)

- En MCTS, incrementalmente se construye un árbol de búsqueda que contiene los estados y acciones visitados junto con los valores estimados $q(s,a)$ para cada par
  - Repetir para cada episodio simulado
    - Seleccionar: hasta que se llegue a una hoja, elegir acciones acorde a $q(s,a)$
    - Expandir: buscar en el árbol por un nodo
    - Rollout: hasta la terminación del episodio con una política fija
    - Actualizar: valores acción $q(s,a)$ para todos los pares en el árbol
    - $q(s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{u=t}^{T} 1(S_u^k, A_u^k = s,a) G_u^k \approx q_\pi(s,a)$
  - Regresar la mejor acción de acuerdo a $q(s,a)$ en la raíz cuando se termine el tiempo

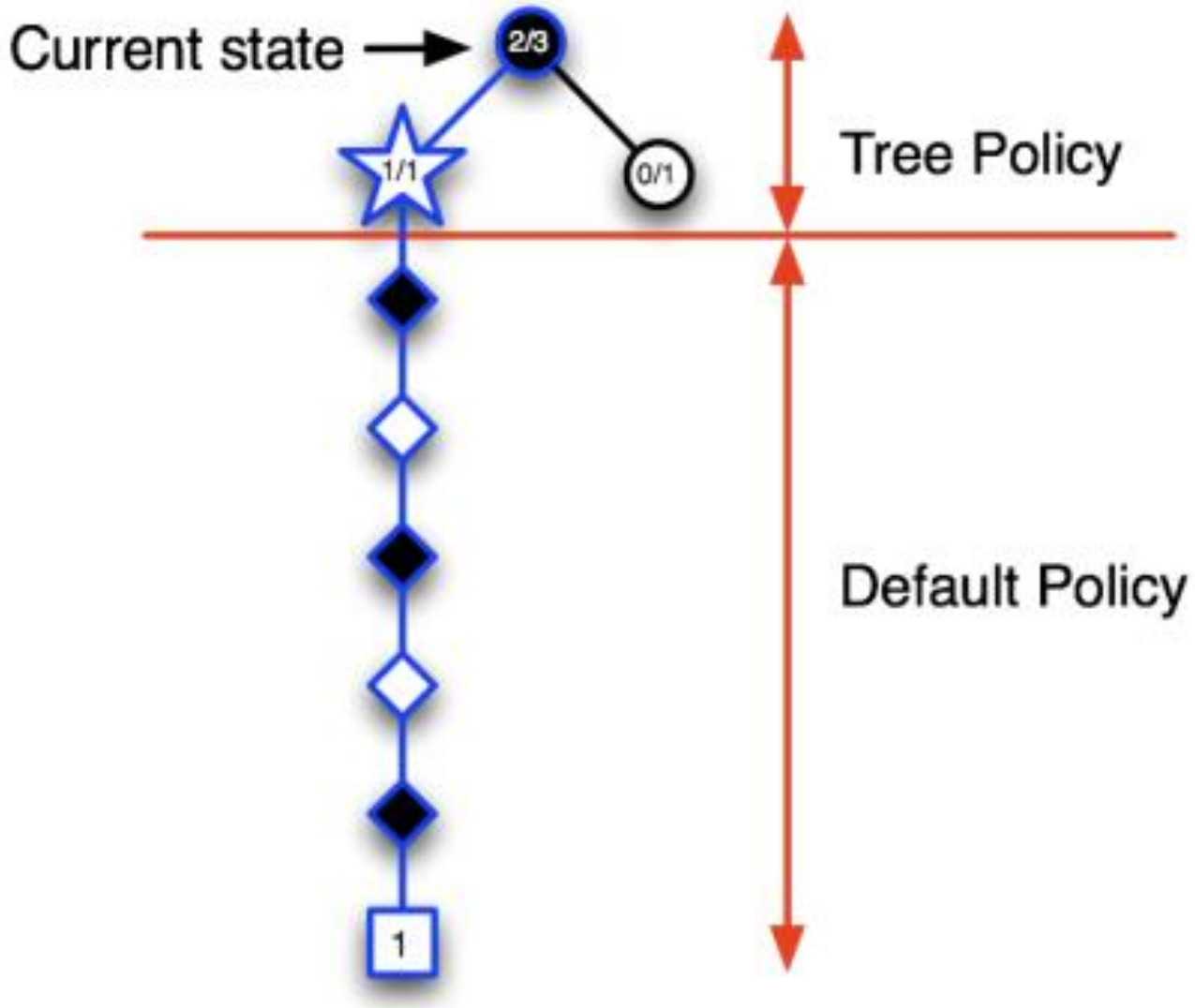
Ejemplo I



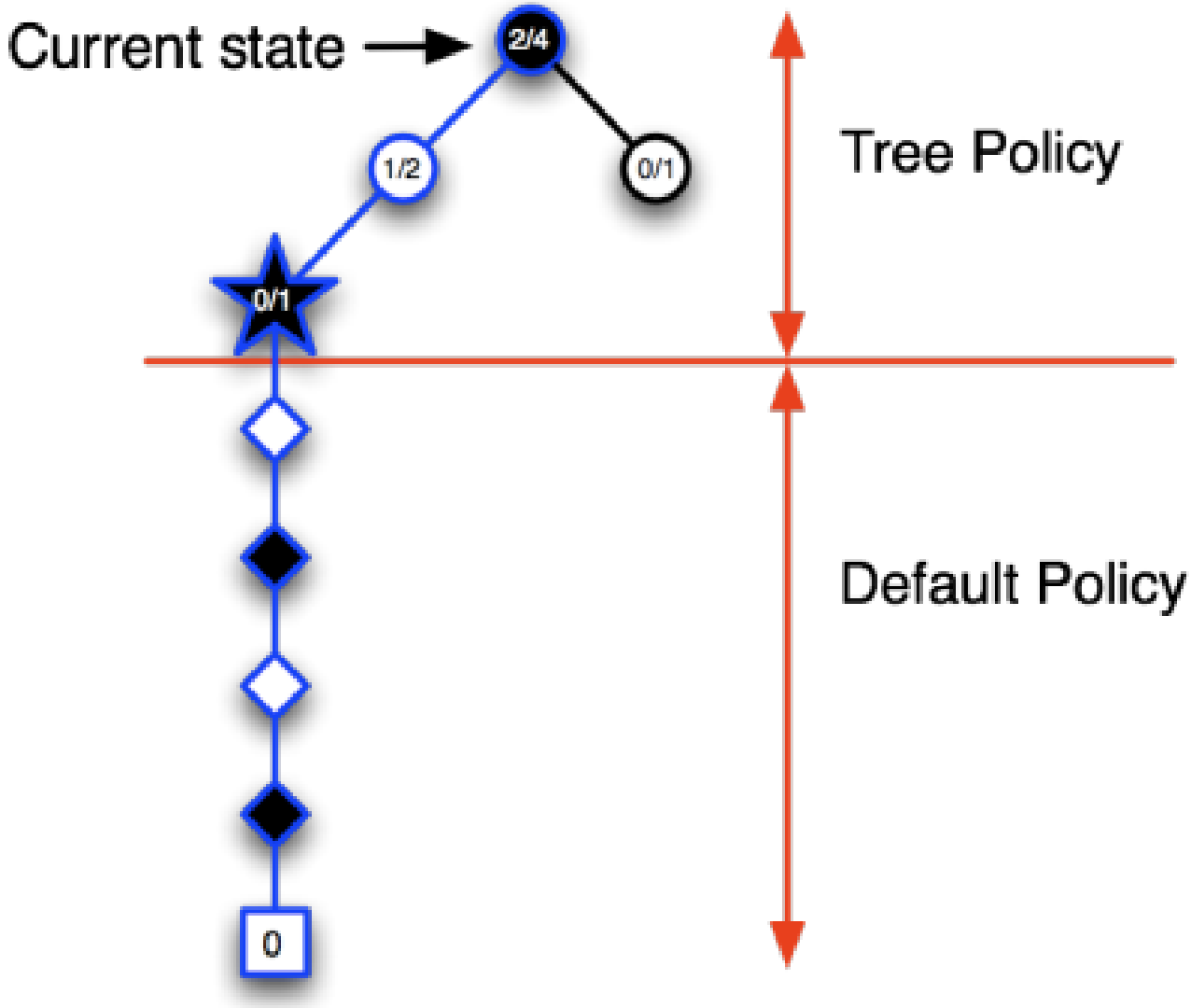
Ejemplo II



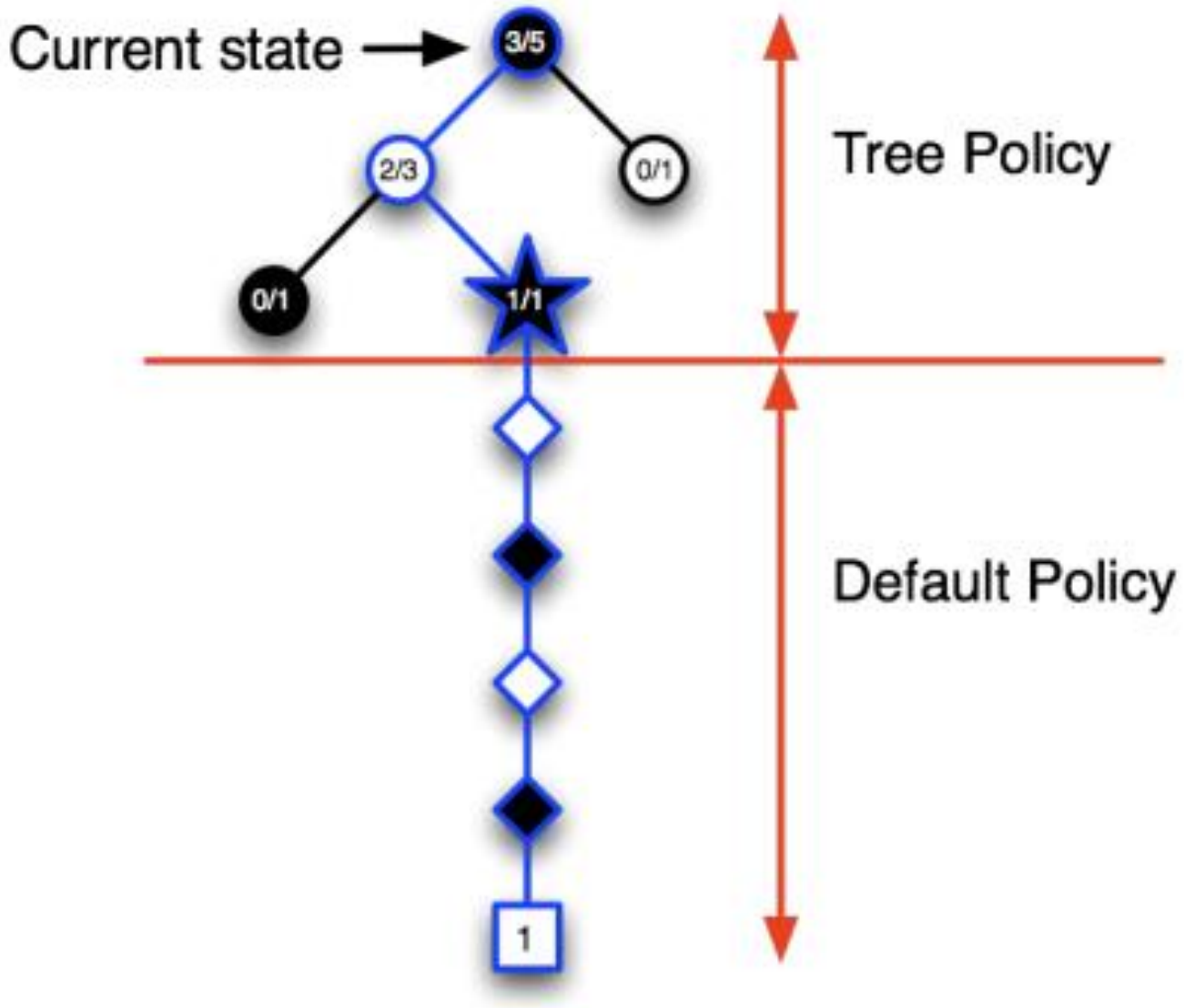
Ejemplo III



Ejemplo IV



Ejemplo V



Ventajas de MCTS

- Best-first search altamente selectivo
- Evalúa los estados dinámicamente
- Usa muestras para romper la maldición de la dimensionalidad
- Funciona con modelos de caja negra
- Eficiente computacionalmente
- También se puede usar TD ;)

Búsqueda en árbol y aproximación de funciones

La búsqueda en árbol es un enfoque tabular

Se basa en una tabla parcial

Para RL libre de modelo, la búsqueda en tabla es ingenua

- No se pueden almacenar valores para todos los estados
- No generaliza similitudes entre estados

Para búsqueda basada en simulación, la búsqueda en tabla es menos ingenua

- La búsqueda almacena valores para estados alcanzables
- No generaliza
- En espacios grandes, una función de aproximación puede ser útil

Para la otra vez…

- Exploración y explotación

Antes de empezar… el proyecto I

Anteproyecto

- Categoría (tesis, aplicación, reproducción, investigación)
- Integrantes
- El problema
  - ¿Qué tarea o problema se estudiará?
  - ¿Dónde se obtendrán los datos, simulador o sistema del mundo real?
  - ¿Cuál es la principal hipótesis que se investigará?
  - ¿Cómo se relaciona con RL?
- Metodología
  - Describir la clase de métodos que se utilizarán (de acuerdo al curso)
  - ¿Qué literatura se utilizará para evaluar los resultados? cualitativa y cuantitativa
- Contribuciones
- Referencias

Máximo 2 paginas

Entrega 30 de marzo

Existe oportunidad de discutir ideas

Antes de empezar… el proyecto II

- RLDM: Multi-disciplinary Conference on Reinforcement Learning and Decision Making
- AAMAS: International Conference on Autonomous Agents and MultiAgent Systems
- NIPS: Neural Information Processing Systems
- ICML: International Conference on Machine Learning
- ICLR: International Conference on Representations
- Kaggle: An online machine learning competition website
- https://www.kaggle.com/c/google-football
- https://www.cs.mcgill.ca/~jpineau/ICLR2018-ReproducibilityChallenge.html