

Aprendizaje por Refuerzos

Teoría y Aplicaciones en Robótica,
Neurociencia y Psicología

Carlos “Greg” Diuk

Department of Psychology
Princeton Neuroscience Institute
Princeton University



Resumen de ayer

- Repasamos MDPs y cómo resolverlos mediante VI cuando conocemos T y R .
- Tres tipos de complejidad en aprendizaje: experiencia, computacional, espacio.
- Aprendiendo cuando no conocemos T y R : model-based, model-free y policy search.

Resumen de ayer

- Algoritmos model-based construyen modelos de T y R.
 - Se puede hacer exploración inteligente: R_{\max} y el optimismo ante la incertidumbre.
 - Uso eficiente de la experiencia (bajo sample complexity).
 - Alto costo computacional porque hay que correr un algoritmo de planning para resolver el modelo (VI u otro).

Resumen de ayer

- Algoritmos model-free aproximan Q/V directamente.
 - Menos oportunidades de exploración inteligente.
 - Bajo costo computacional.
- Policy search usa métodos de gradiente para encontrar políticas directamente.
- Aplicaciones: AIBOs y Helicopter.

Resumen de ayer

- Aprendizaje en el cerebro:
 - Hábitos y *goal-directed*. Dos sistemas?
Corresponden a model-free y model-based.
 - Model-free en el cerebro:
 - Señal de dopamina en ventral striatum responde a errores de predicción.
 - Model-based en el cerebro:
 - Un poco más complicado y se sabe menos, pero hay modelos. DLPFC codifica valor? OFC representaciones del estado?

Model-free II

Q-learning

- Recordemos:

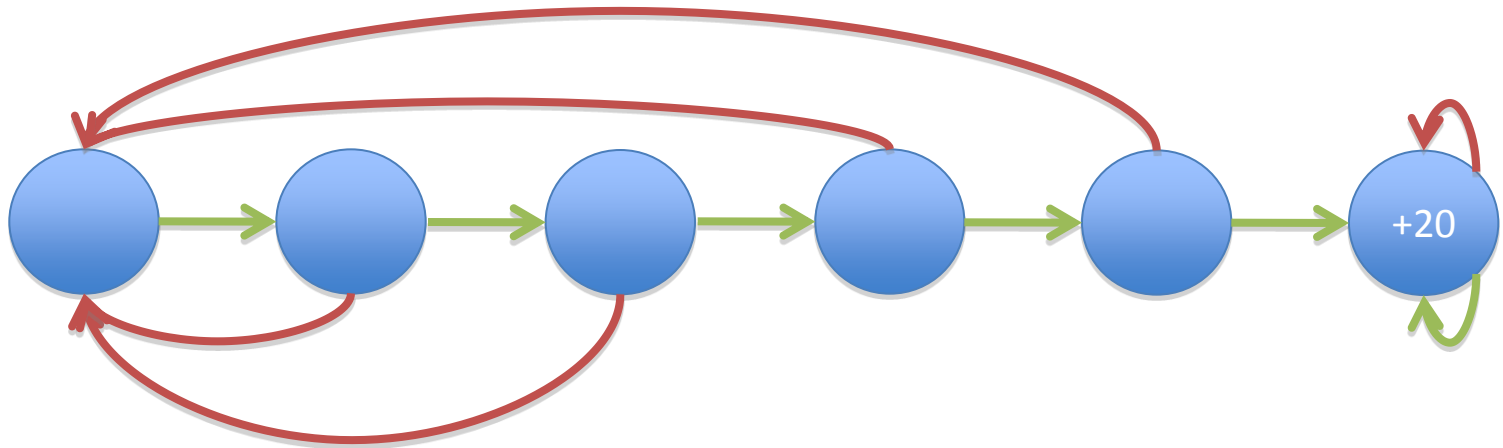
Observar s, a, r, s' y hacer:

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

- Watkins y Dayan (1992) demostraron que converge a Q^* .
- No quiere decir que lo haga rápido...

Q-learning: problema

- Exploración ϵ -greedy o softmax.
- Piensen qué pasa con la exploración de ese tipo acá:



$O(2^{|S|})$ hasta descubrir el estado con el refuerzo. Hasta que no lo alcanza, no se propaga ningún Q interesante.

Q-learning

- No es PAC (Probably Approximately Correct).
- R_{\max} y otros algoritmos model-based sí lo son.
- Strehl et al 2006 muestra una variación de Q-learning que sí es PAC, construyendo modelos *locales* en base a la experiencia reciente.

SARSA(λ)

- Vamos a mejorar un poco las cosas, al menos en la práctica.
- Q-learning actualiza sólo un par estado-acción por vez:

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

- ¿Por qué no actualizar más?

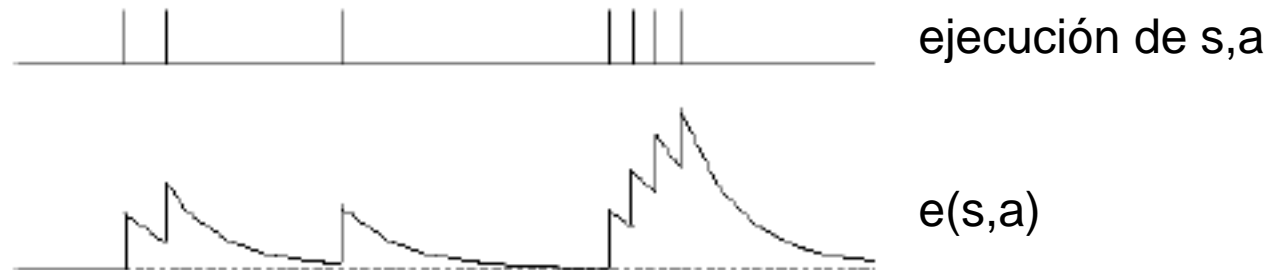
SARSA(λ)

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \delta_t e_t(s,a)$$

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

Eligibility trace... (intraducible)



SARSA(λ)

Initialize $Q(s, a)$ arbitrarily and $e(s, a) = 0$, for all s, a

Repeat (for each episode):

Initialize s, a

Repeat (for each step of episode):

Take action a , observe r, s'

Choose a' from s' using policy derived from Q (e.g., ϵ -greedy)

$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$

$e(s, a) \leftarrow e(s, a) + 1$

For all s, a :

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$

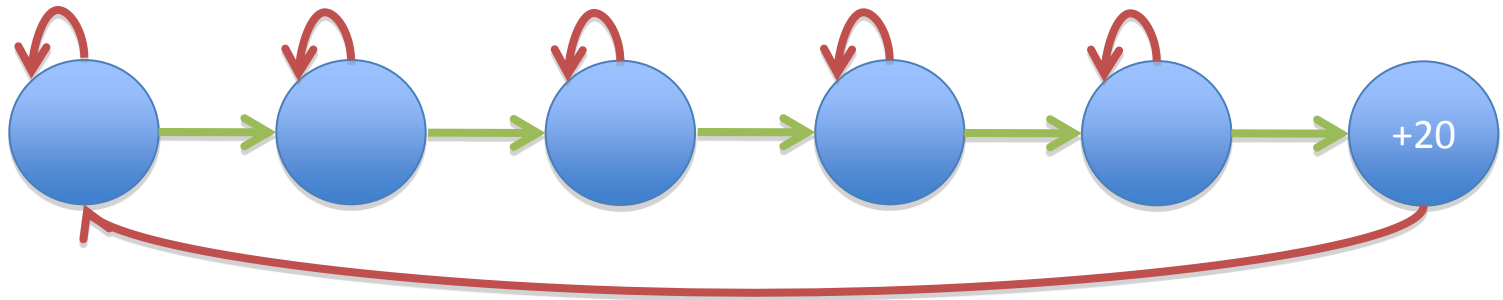
$e(s, a) \leftarrow \gamma \lambda e(s, a)$

$s \leftarrow s'; a \leftarrow a'$

until s is terminal

SARSA(λ)

- Un caso malo para SARSA:

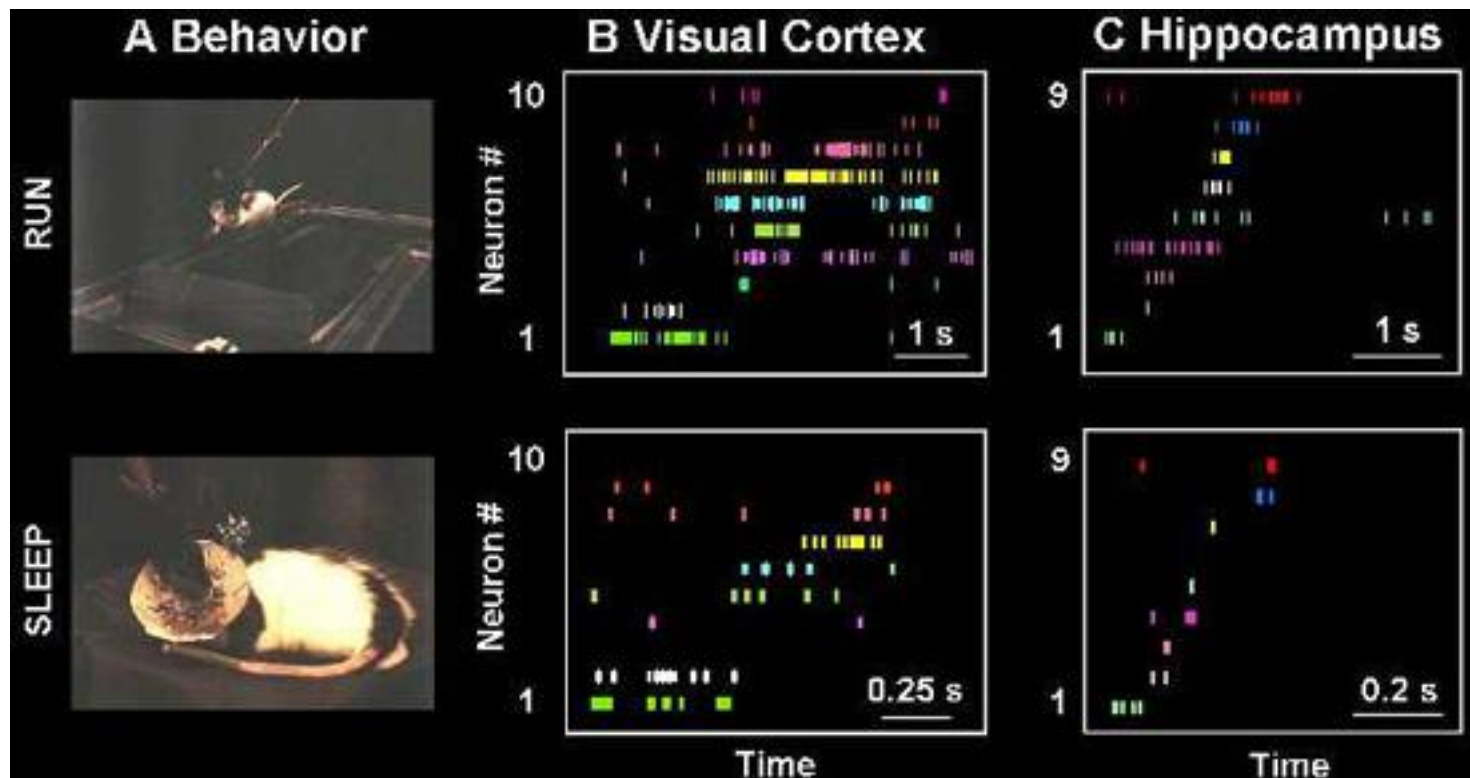


Replay de experiencia

- Acumular experiencia (s,a,s,a,...) durante un episodio, o un rato.
- Al llegar a un estado de fin de episodio, hacer “replay” de esa experiencia.
- Mejor aún si se hace de atrás para adelante.
- De nuevo, ventajas empíricas en ciertos casos.

Replay de experiencia

- Ocurre algo parecido en el hipocampo durante el sueño.



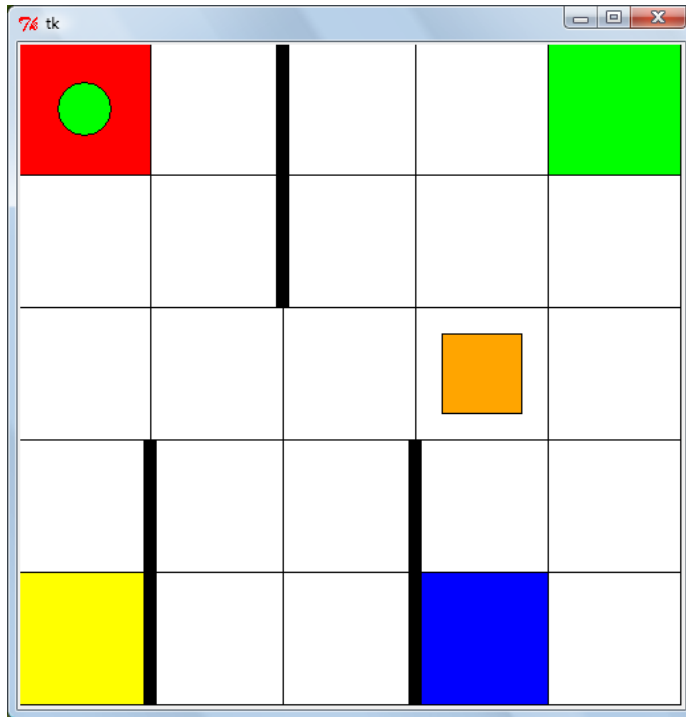
DYNA

- Más model-based que model-free, la idea es:
 - Construye un modelo de T.
 - En lugar de hacer planning (VI), usa el modelo para “proyectar” o “imaginar” experiencias.
- También pasa algo parecido en el hipocampo.

Representaciones

Juguemos un juego

Vamos a ganarle a los algoritmos de RL...

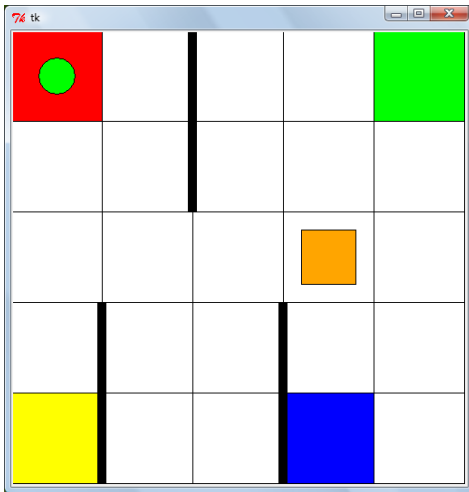


- 6 acciones:
 \uparrow , \downarrow , \rightarrow , \leftarrow , A, B
- Cuando se cierra el juego, es porque ganaron.

Resolviendo el juego

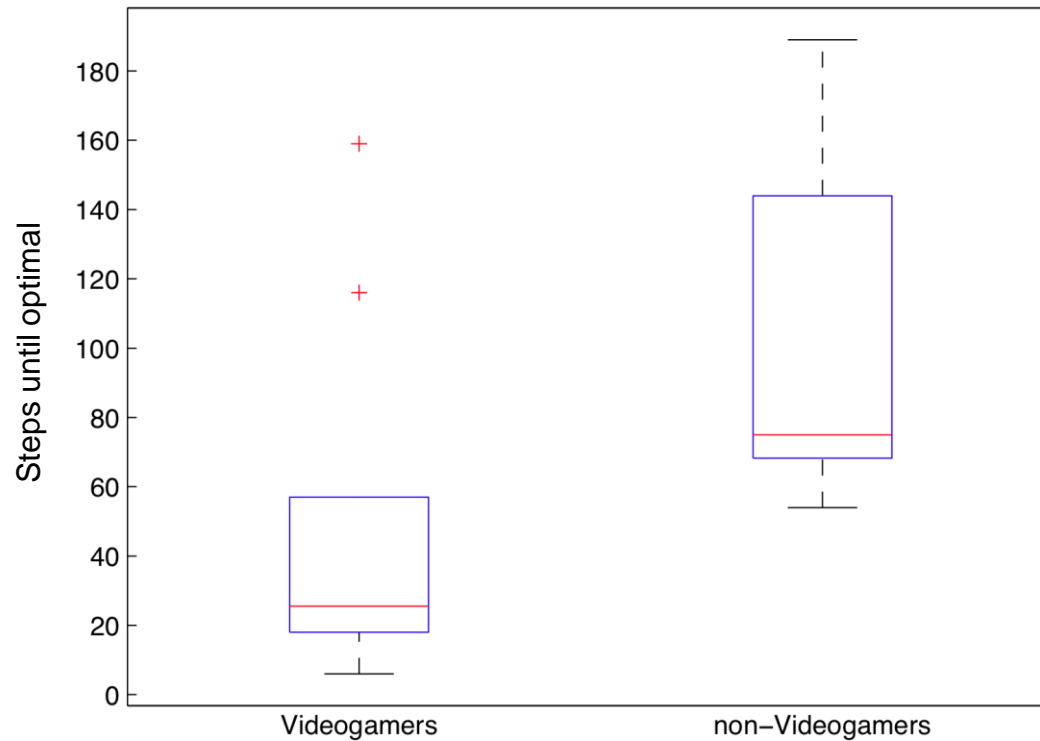
Usamos muchas claves visuales:

- Resolvemos navegación en 1 paso.
- Identificamos las líneas gruesas como paredes impasables.
- Identificamos ubicaciones salientes (son de color!).
- Identificamos eventos salientes: se achica el círculo, se mueve con la caja.
- Etc, etc...



The Taxi problem
[Dietterich 2000]

Performance de humanos



34 participantes: 17 nunca lo resolvieron, 17 sí.

Otro juego

- Ahora perdamos...

<http://www.humanfactors.com/downloads/mousemaze.asp>

Generalización y sesgos inductivos

- A veces ciertos sesgos (como nuestra noción de *arriba y abajo*) nos perjudican si no se aplican.
- Suponemos que tenemos sesgos adecuados (innatos y adquiridos) para lidiar con el mundo.
- Somos buenos generalizadores, somos buenos para abstraer.

Algunas cosas que podemos hacer en RL

- Cambiar la representación de estado.
- Incorporar cierto conocimiento previo.
- Aprender jerárquicamente.
- Usar métodos de generalización y aproximación.

Representations planas

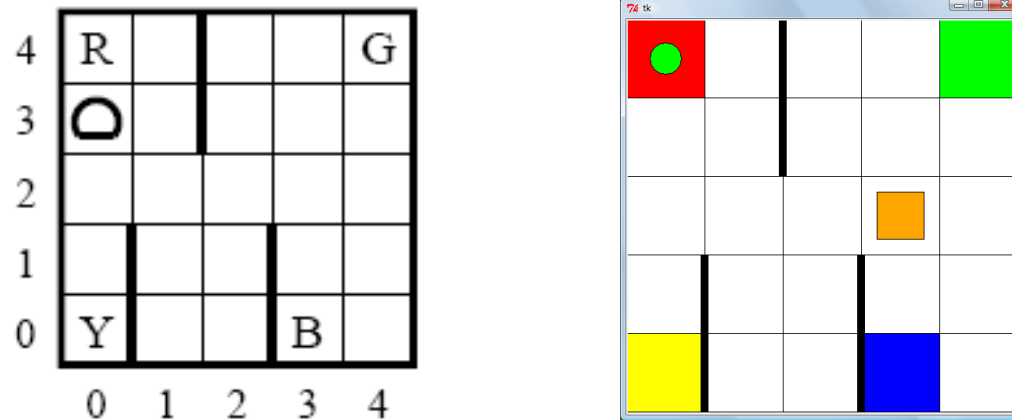


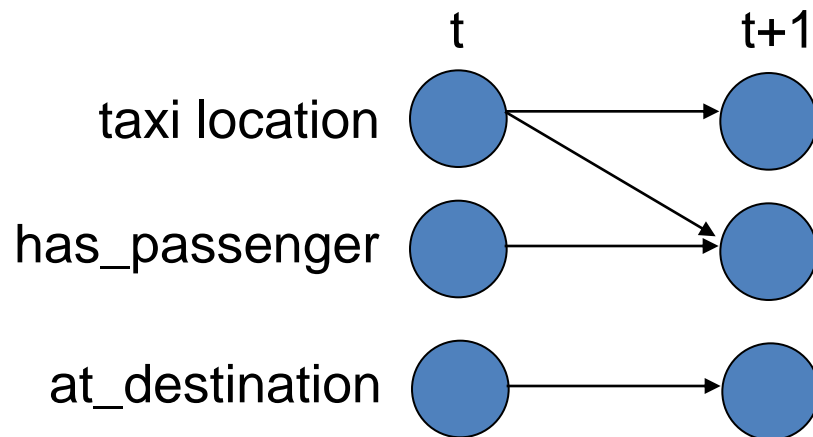
Figure 1: The Taxi Domain.

25 (ubicaciones taxi) x 5 (ubicaciones pasajero + “en taxi”) x 4 (destinos)
=> **500 estados**

El conjunto de estados es un conjunto de identificadores: $s_0 \dots s_{499}$

Representaciones factorizadas

- Incorporando estructura. Taxi es un conjunto:
 <x,y>-location, has_passenger, at_destination
- Función de transición modelada como Dynamic Bayes Network.
- Factores que determinan *PICKUP*:

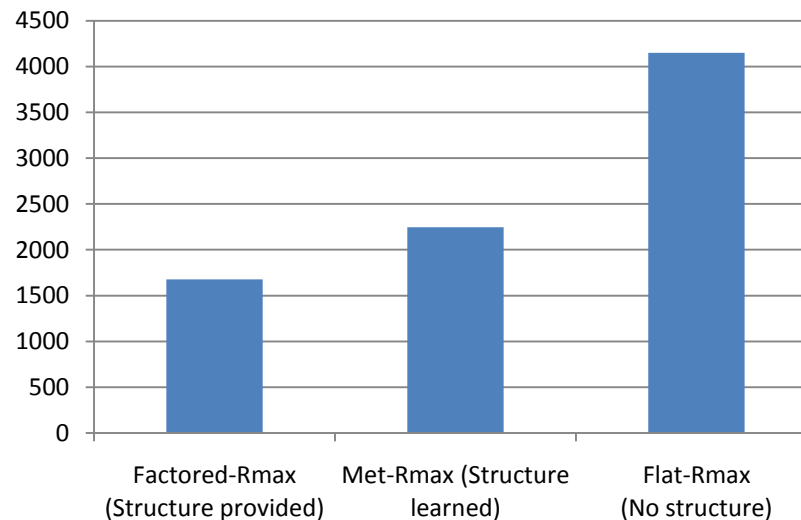


- Factored-Rmax: la DBN viene como input, aprende parámetros.
 [Guestrin *et al.* 2002]

¿Y si queremos aprender el DBN?

Vamos a invertir un rato en entender un método que nos va a servir para un par de cosas.

Adelantamos el resultado en Taxi:



El problema de los meteorólogos

Te acabás de mudar a una nueva ciudad con muchas radios y canales de TV. Cada mañana, querés saber si va a llover o no.

Los primeros días, prendés todos los canales, escribís el pronóstico, y luego observás qué pasa.

¿Qué canal o radio tiene el mejor meteorólogo?

Los k-Meteorólogos

Meteorologists make probabilistic predictions



$p(\text{rain})=0.1$



$p(\text{rain})=0.6$



$p(\text{rain})=0.9$

Each day, you observe the predictions and then observe the true outcome (rain=0 or 1)

In the long-run, which is the best meteorologist?

Los k-Meteorólogos

Each meteorologist uses a different prediction rule



$$p(\text{rain})=0.1$$

h_1



$$p(\text{rain})=0.6$$

h_2



$$p(\text{rain})=0.9$$

h_3

h^* is the correct hypothesis

We show that if $h_i=h^*$, then squared error of h_i is smallest after enough experience is gathered.

Probabilistic concepts

- Probabilistic concepts generalize deterministic concepts to capture uncertainty in many real-life problems.

Deterministic concept		Probabilistic concept	
X	Y	X	Y
[0,0,1,1,0]	+	[0,0,1,1,0]	0.9
[1,0,1,0,0]	-	[1,0,1,0,0]	0.4
[0,1,1,1,1]	-	[0,1,1,1,1]	0.1
[0,0,0,0,0]	+	[0,0,0,0,0]	0.8
...		...	

Kearns and Schapire (1994) show how probabilistic concepts can be learned in PAC setting: not suitable for non-iid samples (for example, in the context of exploration in RL).

Los k-Meteorologos adaptativos

- Cada meteorologo *aprende* su hipótesis de la experiencia.
- Asumamos 2 meteorologos
 - Observar input x_t
 - Pedirle a cada meteorologo una predicción.
 - Si alguno responde \perp , observar output z_t y dárselo a los meteorólogos para que aprendan
 - Si los dos meteorólogos tienen predicción y coinciden, usar esa predicción.
 - Si no coinciden, eliminar al meteorólogo con mayor error cuadrático medio.

Analysis

- If each meteorologist observes $O(1/\varepsilon^2 \ln k/\delta)$ samples, the total number of \perp s is bounded by

$$\Theta(k/\varepsilon^2 \ln k/\delta)$$

Where:

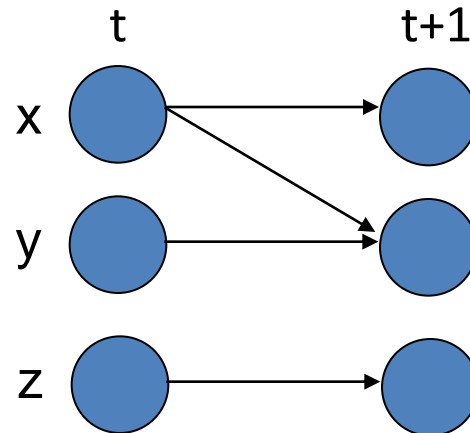
- k is the number of meteorologists.
 - ε is a measure of accuracy expected.
 - δ is the probability with which accuracy is achieved.
-
- Polynomial convergence (matching lower and upper bounds)!

Los meteorólogos aprenden la DBN

- Factored-state representations: state represented as vector of state variables:

$$s = \langle x, y, z \rangle$$

- Transition probabilities modeled as Dynamic Bayes Network:



Los meteorólogos aprenden la DBN

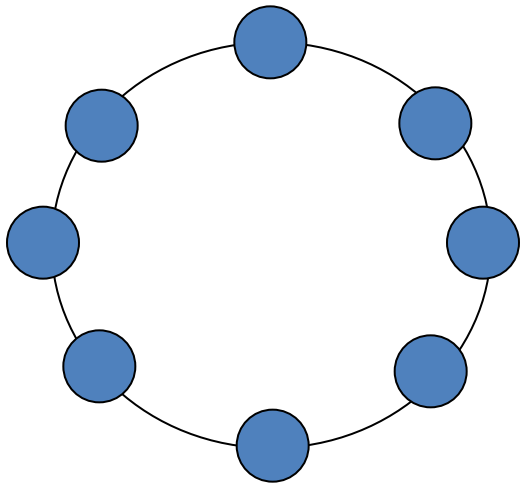
– Met-Rmax:

- Asume in-degree máximo k del DBN es conocido.
- Construye un meteorólogo por cada subconjunto de k padres.
- Si cualquiera de los mets dice “ \perp ”, asume transición optimista.
- Funciona y tiene cotas polinomiales (bueno, exponenciales en k asumiendo $k \ll n$)

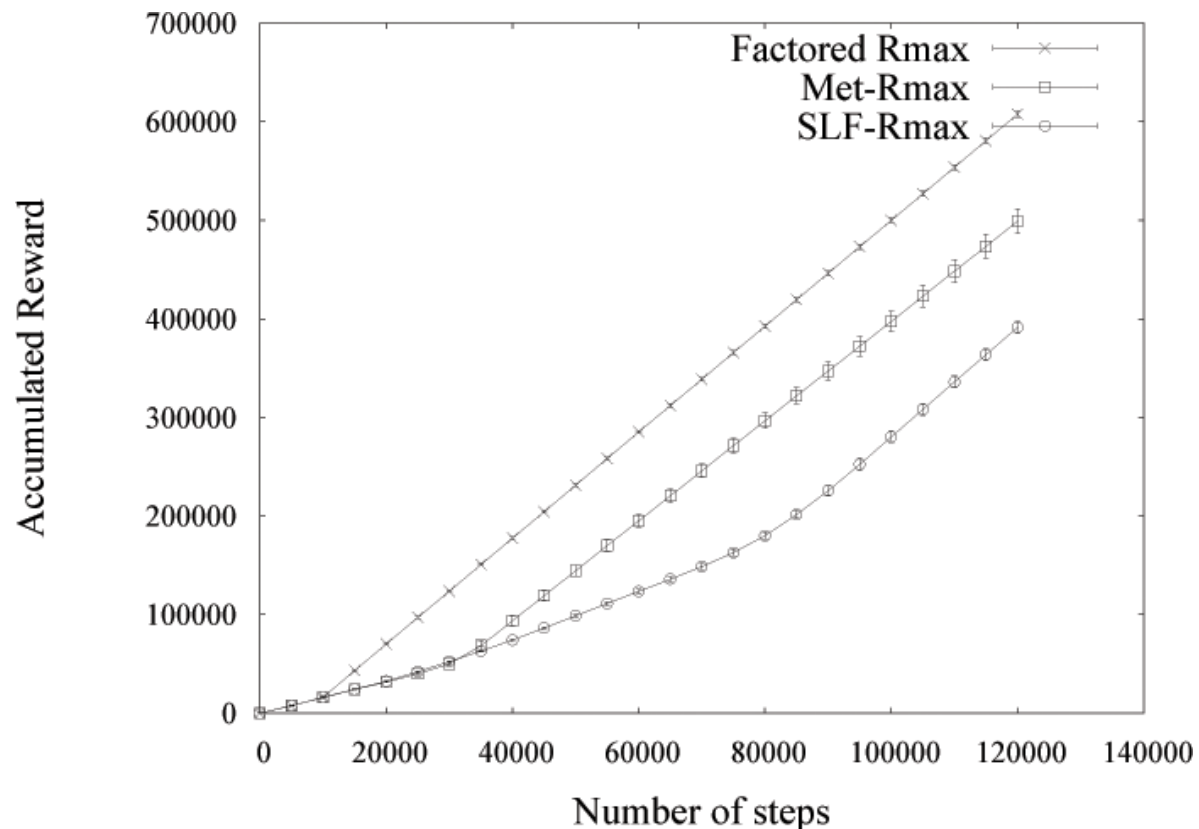
SysAdmin

- Problema canónico [Guestrin et al 2003]

Bi-ring network
Topology (with 8 machines)



8 state variables: 2^8 states
9 actions: reboot machine i +
“do_nothing”

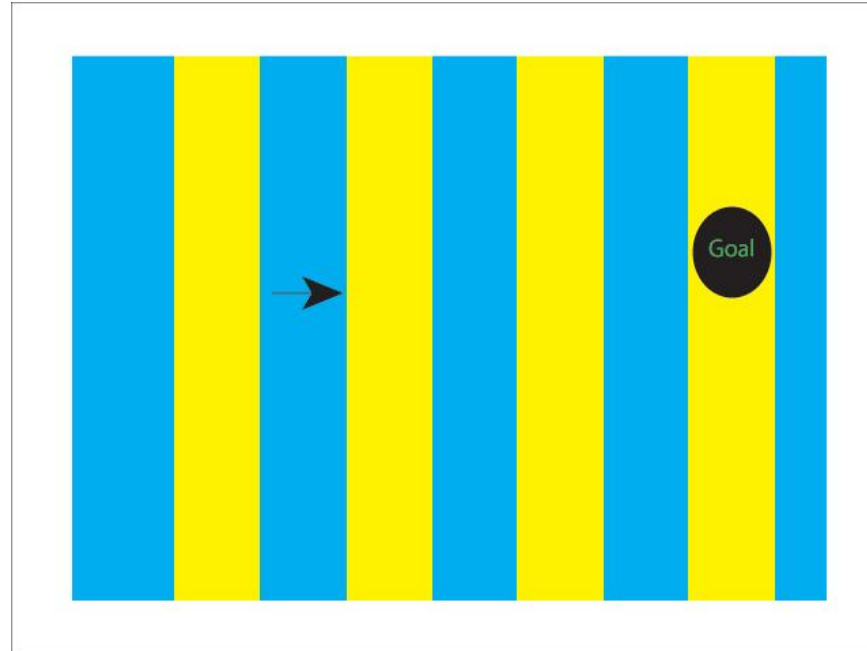


Application II

- Navegación de un robot LEGO.
- Qué sensor predice dinámica mejor:
 - Robot sensa el terreno y lo clasifica en base a:
 - Color
 - Textura
 - Color y textura

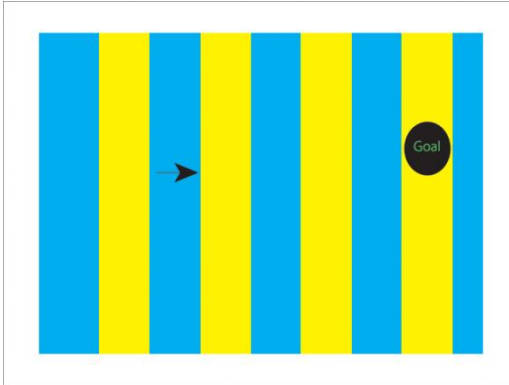


Artificial Dynamics



	Action 0	Action 1	Action 2	Action 3
Yellow surface	Left	Right	Forward	Backward
Blue surface	Right	Left	Backward	Forward

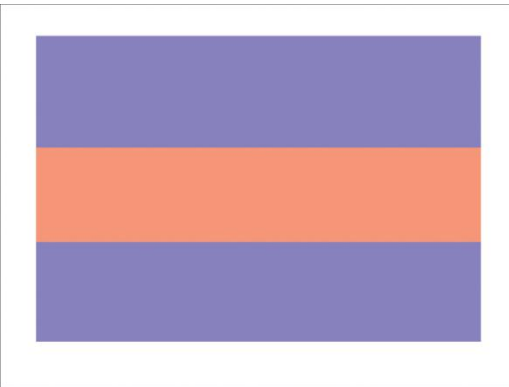
Features tested



Correct



Incorrect 1



Incorrect 2

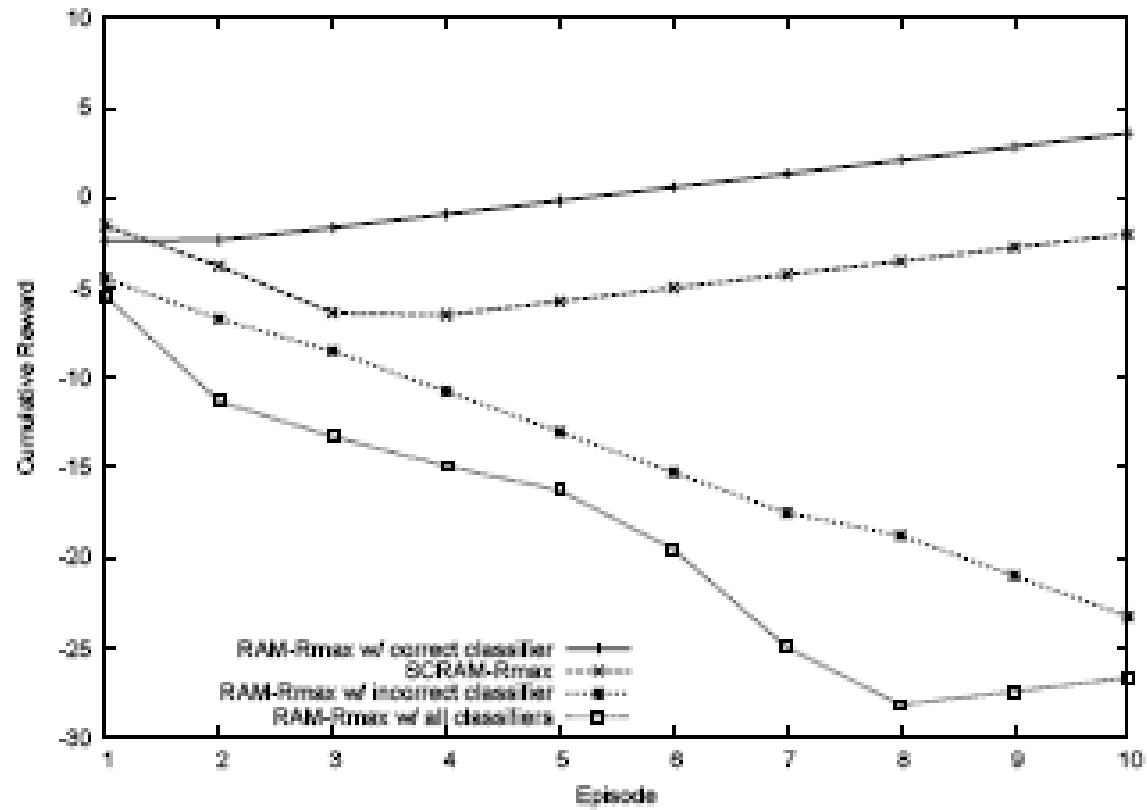


Incorrect 3



All together

Results



Taxi problem leaderboard

	# of steps	Planning time per step	What we have to tell it
Q-learning	29350	<1ms	#states, #actions
Flat Rmax	4151	~70ms	#states, #actions, Rmax (max possible reward)
Factored Rmax	1676	~90ms	DBN structure
Met-Rmax [Diuk et al 09]	2246	~70ms	Max in-degree of DBN

Aprendizaje jerárquico (mañana)

- Decompose target task into a hierarchy of smaller tasks.
- Hierarchies restrict policy space.

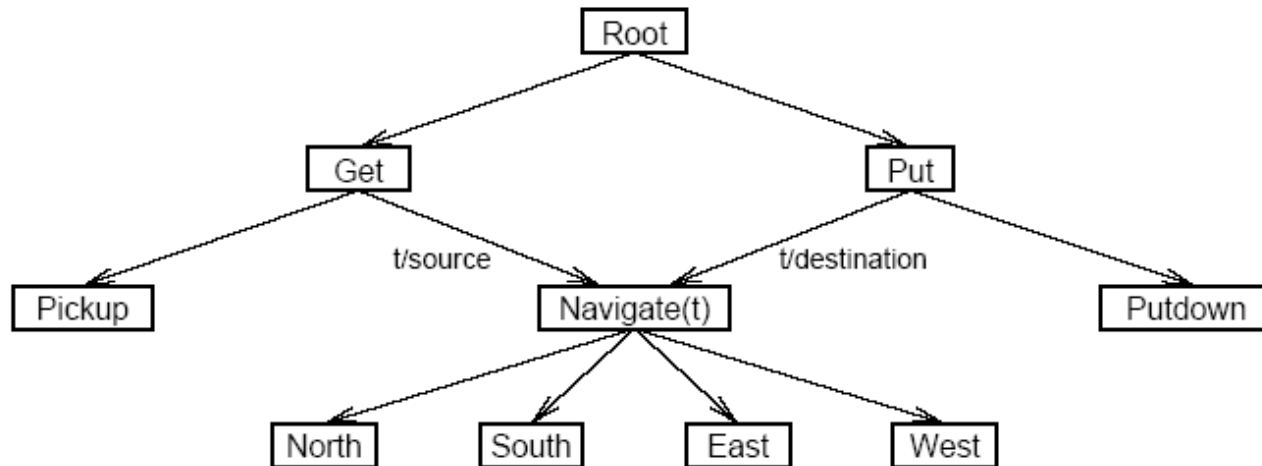


Figure 2: A task graph for the Taxi problem.

Taxi problem leaderboard

	# of steps	Planning time per step	What we have to tell it
Q-learning	29350	<1ms	#states, #actions
Flat Rmax	4151	~70ms	#states, #actions, Rmax (max possible reward)
Factored Rmax	1676	~90ms	DBN structure
Met-Rmax [Diuk et al 09]	2246	~70ms	Max in-degree of DBN
MaxQ	6298	9.6ms	Task hierarchy, DBNs for each task
DSHP [Diuk et al 06]	329	16ms	Task hierarchy, DBNs for each task

RL Relacional

- Representaciones en lógica de primer orden (FOL).
- El estado es un conjunto de predicados de FOL, codificados por el diseñador.
- rQ-learning en Taxi:
 - [Morales 03]
 - Codifica conocimiento parecido a representación jerárquica.
 - Aprende un poco más rápido que Factored-Rmax (>1300 steps)

La representación “correcta”?

- State space is too big, **flat** representations inadequate.
- We know how to learn **factored-state** representations (but they don't really represent state as *we* see it).
- **Hierarchical representations** are efficient, but we might be solving the problem for the agent.
- **Relational RL** first-order logic approaches are very expressive, but often very hard to learn.

Demos un paso atrás...

Cómo jugarían este juego?



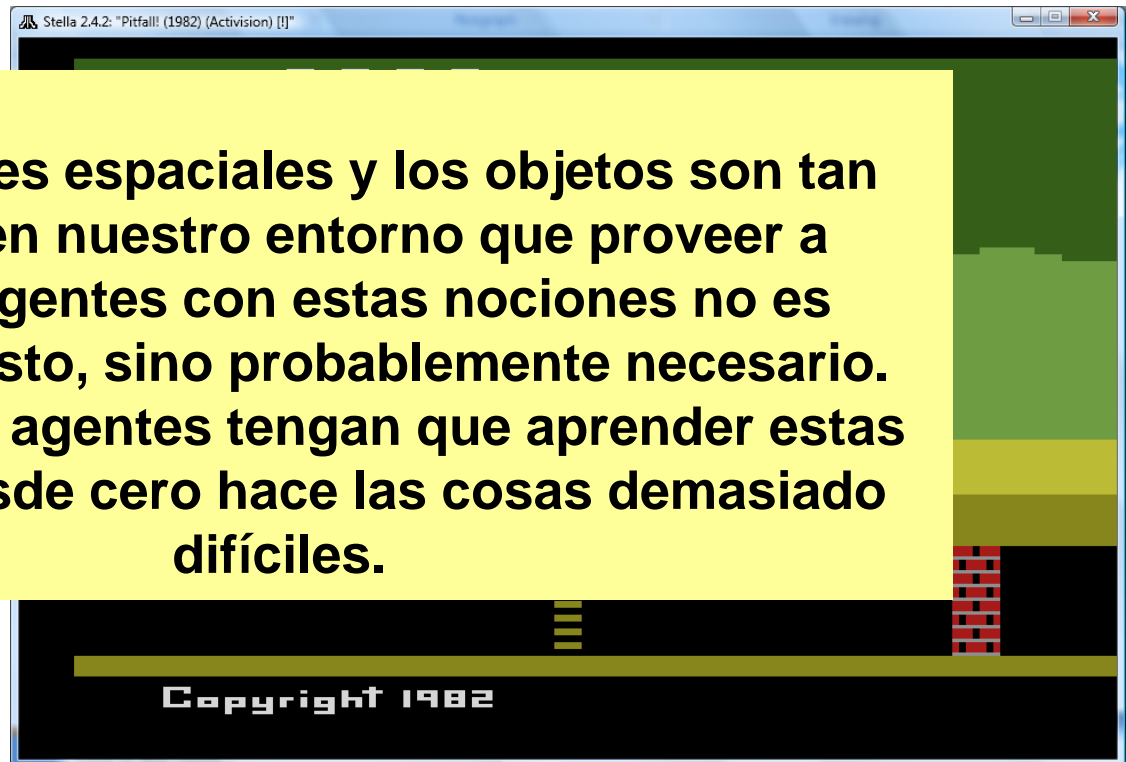
Qué es un estado?

4	R			
3	○			
2				
1				
0	Y			
0				

Figure 1: T

A simple hash code that tells you if you've been "there" before.

Las relaciones espaciales y los objetos son tan comunes en nuestro entorno que proveer a nuestros agentes con estas nociones no es solamente justo, sino probablemente necesario. Que nuestros agentes tengan que aprender estas nociones desde cero hace las cosas demasiado difíciles.



What we (the agent) can actually "see": objects, interactions, spatial relationships.

OO representation

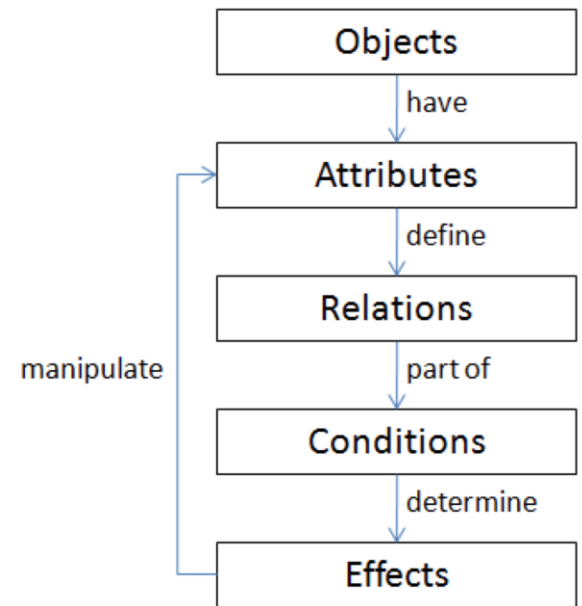
- Problem defined by a set of objects and their attributes.
- Example: Objects in Pitfall defined by a bounding box on a set of pixels based on color.



- **State** is the union of all objects' attribute values.

OO representation

- For any given state s , there is a function $pred(s)$ that tells us which relations occur under s .
- Dynamics defined by **conditions** and **effects**.
- **Conditions** are conjunctions of *terms*:
 - Relations between objects:
 - $touch_{N/S/E/W}(object_i, object_j)$
 - $on(object_i, object_j)$
 - Any (boolean) function on the attributes.
 - Any other function encoding prior knowledge.
- Actions have *effects* that determine how objects' attributes get modified. Effects have types: arithmetic, assignment, etc.



OO representation

- A state s (set of objects and their attribute values) induces a truth-value assignment to all terms in the world:

$\text{on}(\text{Harry}, \text{Ladder})$

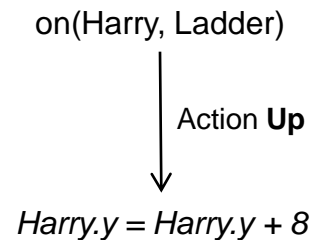
$\sim\text{touch}_N(\text{Harry}, \text{Log})$

$\sim\text{touch}_E(\text{Harry}, \text{Wall})$

...

- If $\text{pred}(s)$ satisfies a condition c , under action a , then the associated effect e is applied to the attributes of state s :

$\text{Harry}.y = \text{Harry}.y + 8$



Example – Pac-Man

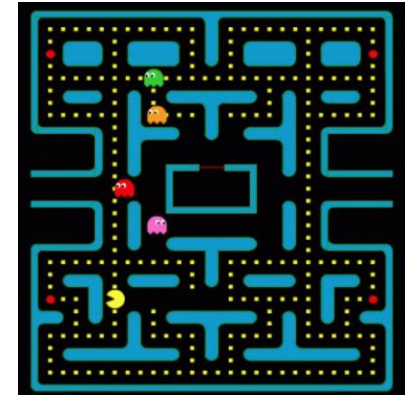
Objects: *Pac-Man, Ghosts, Pac-Dots, PowerPellets, Wall*

Actions: *North(o), South(o), East(o), West(o)*

Relations: $\text{touch}_{N/S/E/W}(\text{<class>, <class>})$

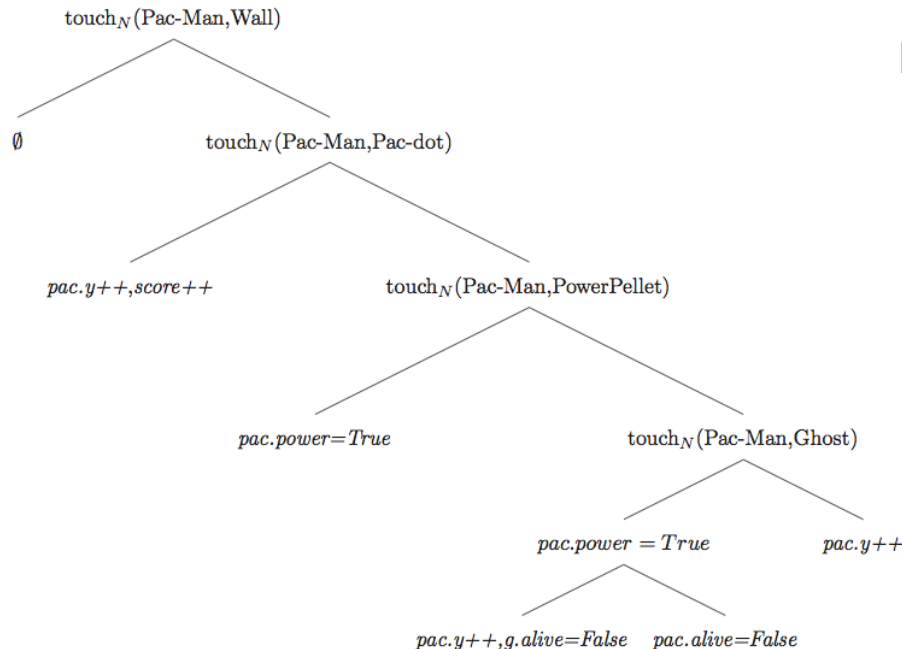
State: all objects' locations, *Pac.power, Pac.alive, Ghost.alive*

Effects types: arithmetic, assignment, etc.

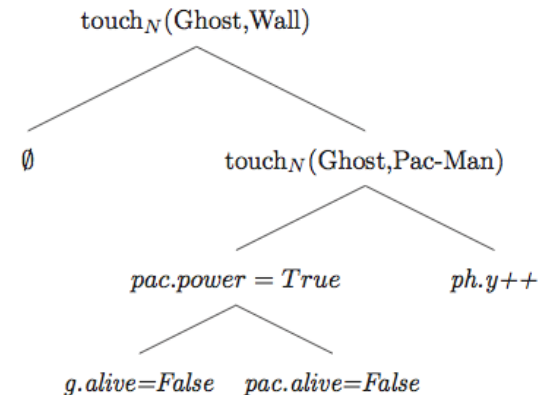


Input

Dynamics for action *North(Pac-Man):*



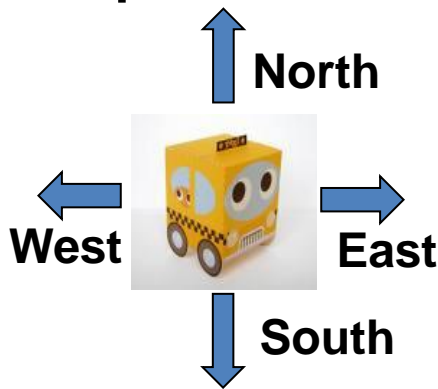
Dynamics for action *North(Ghost):*



Learned

Can we learn action effects?

- Simple model: effects have no preconditions and are **deterministic**.



Observe transition:

s: Taxi.<x,y> = <0,0>

a: *North*

s': Taxi.<x,y> = <0,1>

Observe another transition:

s: Taxi.<x,y> = <0,1>

a: *North*

s': Taxi.<x,y> = <0,2>

Hypotheses about effect of action *North*:

Taxi.y \leftarrow Taxi.y + 1

~~Taxi.y \leftarrow 1~~ ~~Taxi.y \leftarrow 2~~

And conditional effects?

- Single condition/effect per action (still deterministic)



$\langle 0,0 \rangle \Rightarrow t_N(T,W) \ t_S(T,W) \ t_E(T,W) \ t_W(T,W) \ \sim t_N(T,W) \ \sim t_S(T,W) \ \sim t_E(T,W) \ \sim t_W(T,W)$

North

$\langle 0,1 \rangle \Rightarrow t_N(T,W) \ t_S(T,W) \ t_E(T,W) \ t_W(T,W) \ \sim t_N(T,W) \ \sim t_S(T,W) \ \sim t_E(T,W) \ \sim t_W(T,W)$

North

$\langle 0,2 \rangle \Rightarrow t_N(T,W) \ t_S(T,W) \ t_E(T,W) \ t_W(T,W) \ \sim t_N(T,W) \ \sim t_S(T,W) \ \sim t_E(T,W) \ \sim t_W(T,W)$

⋮

$\langle 1,1 \rangle \Rightarrow t_N(T,W) \ t_S(T,W) \ t_E(T,W) \ t_W(T,W) \ \sim t_N(T,W) \ \sim t_S(T,W) \ \sim t_E(T,W) \ \sim t_W(T,W)$

North

$\langle 1,2 \rangle \Rightarrow$

Hypotheses about condition / effect of action *North*:

$t_N(T,W) \ t_S(T,W) \ t_E(T,W) \ t_W(T,W) \ \sim t_N(T,W) \ \sim t_S(T,W) \ \sim t_E(T,W) \ \sim t_W(T,W)$

 $\text{Taxi.y} \leftarrow \text{Taxi.y} + 1$
 ~~$\text{Taxi.y} \leftarrow 1$~~

DOORMax [Diuk, Cohen, Littman 2008]

- Un algoritmo eficiente para entornos determinísticos.
- Asume que hay hasta m efectos por acción.
- Exploración inteligente:
 - Guía la exploración para producir interacciones entre objetos que no sabe predecir.

Pitfall video



Flat state space size (not all reachable): $6^{320 \times 210}$

Learns policy in 494 game actions (4810 game frames)

Optimal policy requires 94 actions (905 game frames)

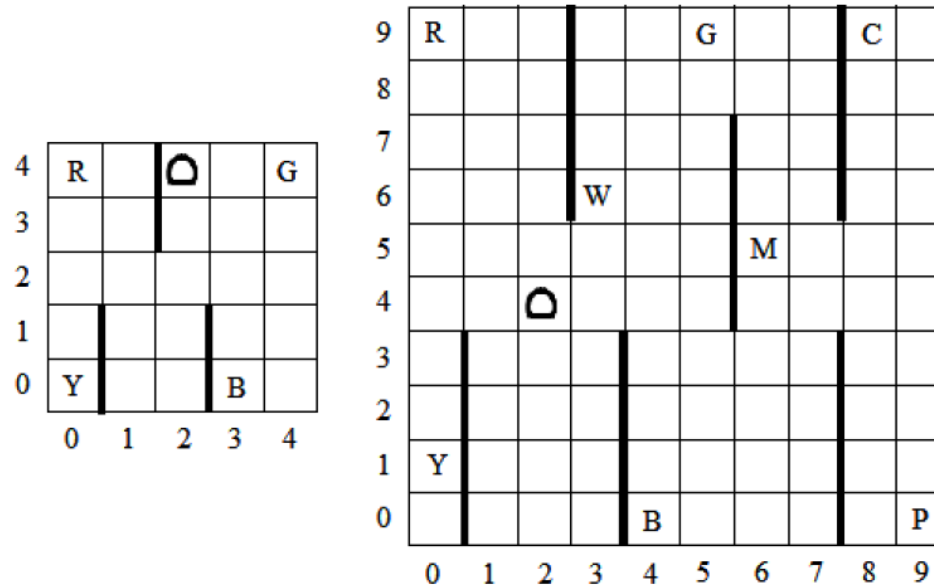
Taxi problem leaderboard

	# of steps	Planning time per step	What we have to tell it
Q-learning	29350	<1ms	#states, #actions
Flat Rmax	4151	~70ms	#states, #actions, Rmax (max possible reward)
Factored Rmax	1676	~90ms	DBN structure
Met-Rmax [Diuk et al 09]	2246	~70ms	Max in-degree of DBN
MaxQ	6298	9.6ms	Task hierarchy, DBNs for each task
DSHP [Diuk et al 06]	329	16ms	Task hierarchy, DBNs for each task

Taxi problem leaderboard

	# of steps	Planning time per step	What we have to tell it
Q-learning	29350	<1ms	#states, #actions
Flat Rmax	4151	~70ms	#states, #actions, Rmax (max possible reward)
Factored Rmax	1676	~90ms	DBN structure
Met-Rmax [Diuk et al 09]	2246	~70ms	Max in-degree of DBN
MaxQ	6298	9.6ms	Task hierarchy, DBNs for each task
DSHP [Diuk et al 06]	329	16ms	Task hierarchy, DBNs for each task
DOOR_{Max} [Diuk et al 08]	529	~45ms	Object representation

Taxi más grande



	Taxi 5x5	Taxi 10x10	Ratio
# States	500	7200	14.40
Factored – R_{\max}	1676 steps	19100 steps	11.39
$DOOR_{\max}$	529 steps	821 steps	1.55
$DOOR_{\max}$ with Transfer from 5x5	529 steps	0 extra steps	1

Análisis de DOORMax

- Mientras observemos efectos, DOORMax aprende los pares condición-efecto que determinan la dinámica del ambiente en $O(nk)$.
- **Muy mal peor caso**, cuando se observan muchos *no-efectos*: $O(2^n)$ – de nuevo el caso del candado.

DOORMax

- DOORMax aprende muy rápido en muchas situaciones.

Pero:

- Asume efectos determinísticos.
- Asume 1 condición por efecto.
- Mal cota en peor caso.