Métodos de Inteligencia Artificial

L. Enrique Sucar (INAOE)

esucar@inaoep.mx

ccc.inaoep.mx/esucar

Tecnologías de Información

UPAEP

Agentes que Aprenden: Aprendizaje por Refuerzo (RL)

- Introducción
- MDPs
- Algoritmos
- Aplicaciones

Introducción

- En aprendizaje por refuerzo (RL) el objetivo es aprender cómo mapear situaciones a acciones para maximizar una cierta recompensa.
- Promesa: *programar agentes mediante premio y castigo sin necesidad de especificar cómo realizar la tarea*
- El refuerzo puede darse en un estado terminal y/o en estados intermedios.

Introducción

- Diferencias con otros tipos de aprendizaje:
  - No se le presentan pares entrada - salida.
  - El agente tiene que obtener experiencia útil acerca de los estados, acciones, transiciones y recompensas de manera activa para poder actuar de manera óptima.
  - La evaluación del sistema ocurre en forma concurrente con el aprendizaje.

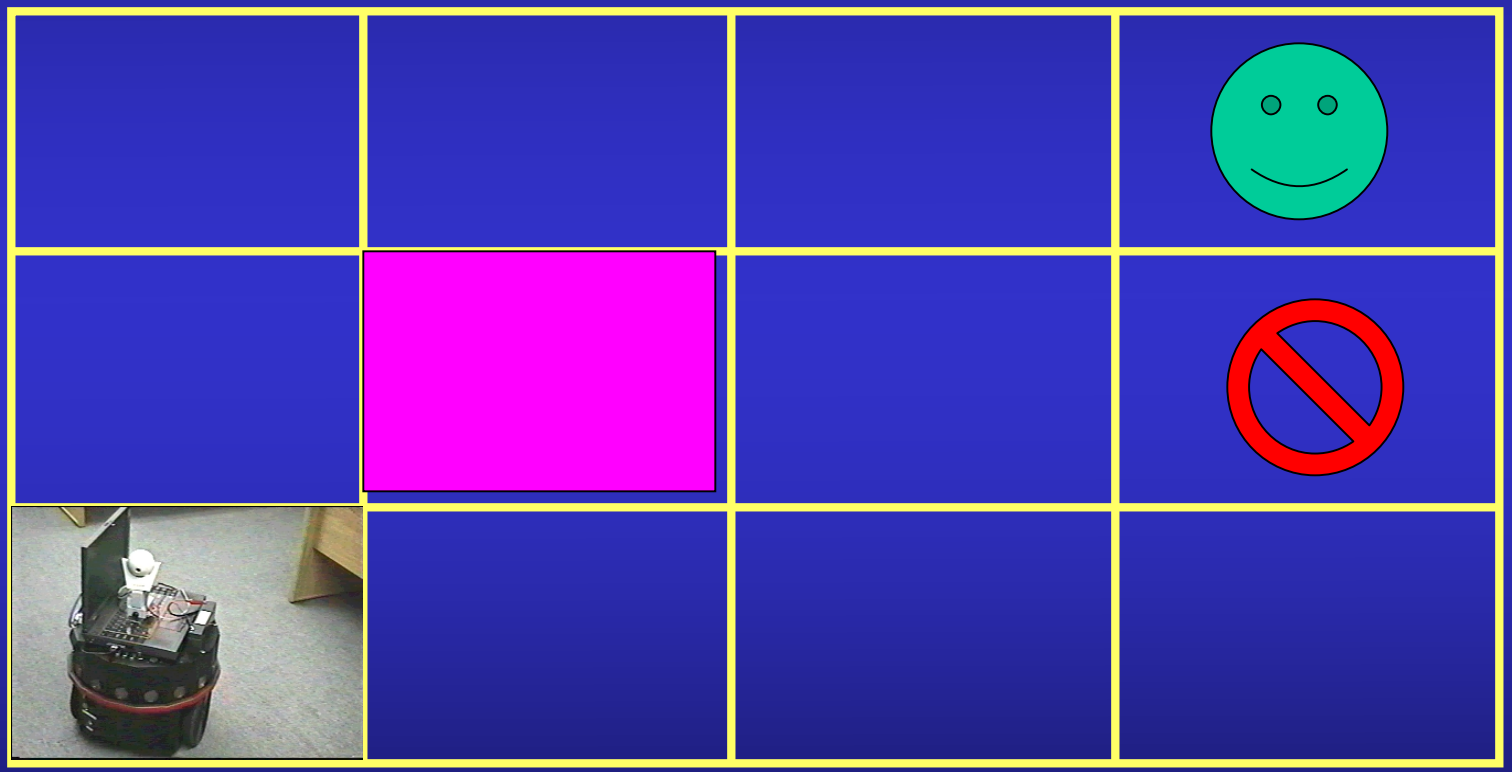
Introducción

- En RL un agente trata de aprender un comportamiento mediante interacciones de prueba y error en un ambiente dinámico e incierto
- En general, al sistema no se le dice qué acción debe tomar, sino que debe de descubrir que acciones dan el máximo beneficio
- En un RL estándar, un agente está conectado a un ambiente por medio de percepción y acción
- En cada interacción el agente recibe como entrada una indicación de su estado actual ($s \in S$) y selecciona una acción ($a \in A$).
- La acción cambia el estado y el agente recibe una señal de refuerzo o recompensa ($r \in$ R)

Ejemplo

- Un robot móvil que tiene que llegar a la meta
- Ambiente: estado (posición actual), recompensas (meta, obstáculos)
- Acciones: izquierda, derecha, arriba, abajo
- Objetivo: encontrar las acciones que lo lleven en forma “óptima” a la meta

Ejemplo – robot móvil

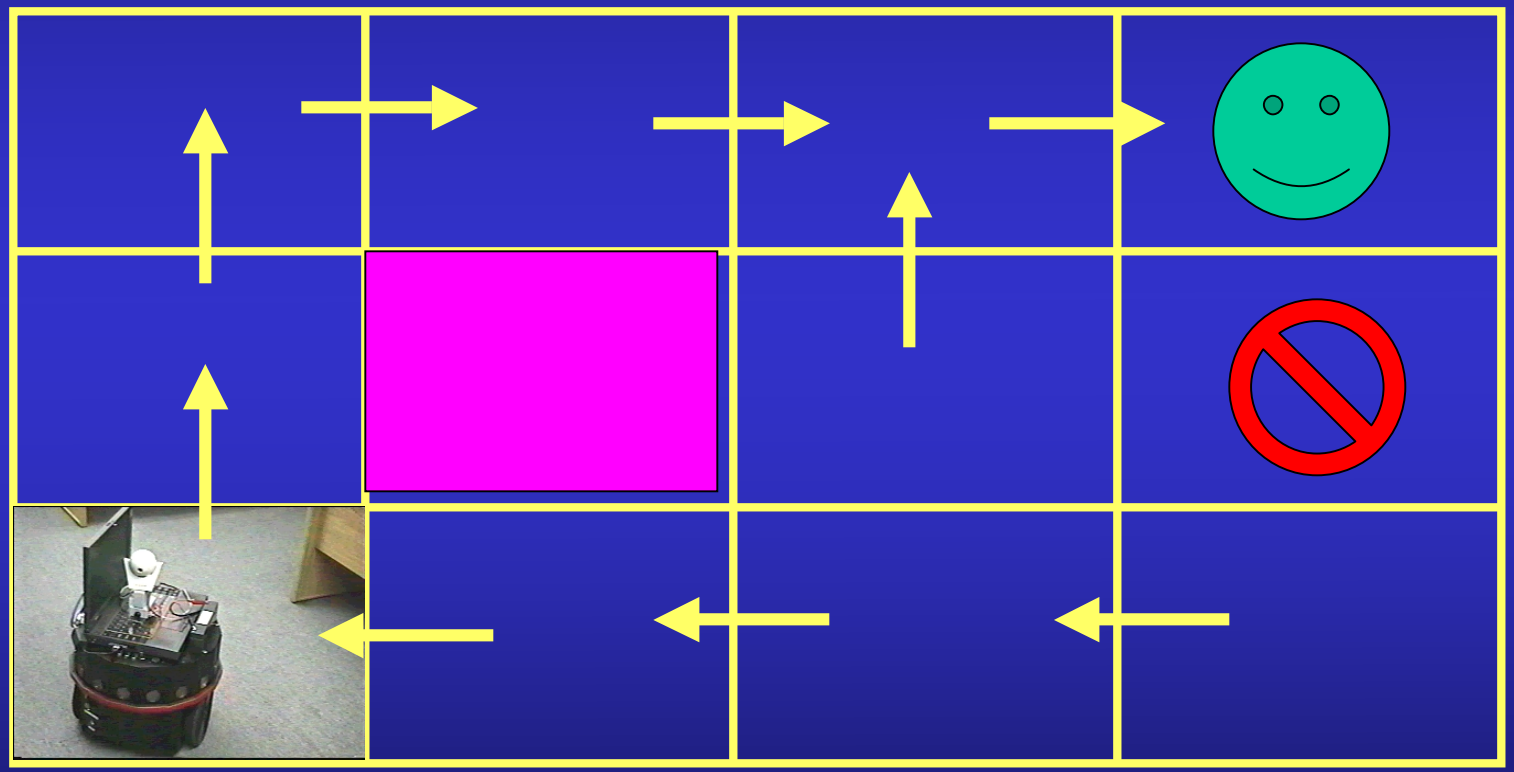


Inicio

Principios

- El comportamiento del agente debe de ser tal que escoja acciones que tiendan a incrementar a largo plazo la suma de las recompensas totales
- El objetivo del agente es encontrar una política (π), que mapea estados a acciones que maximice a largo plazo el refuerzo
- En general el ambiente es no determinista (tomar la misma acción en el mismo estado puede dar resultados diferentes)
- Sin embargo, se asume que el ambiente es estacionario (las probabilidades de cambio de estado no cambian o cambian muy lentamente)

Ejemplo de Política



Inicio

Exploración vs. Explotación

- Aspectos importantes:
  - se sigue un proceso de prueba y error
  - la recompensa puede estar diferida
- Existe un balance entre exploración y explotación
- Para obtener buena ganancia uno prefiere seguir ciertas acciones, pero para saber cuáles, se tiene que hacer cierta exploración
- Muchas veces depende de cuánto tiempo se espera que el agente interactúe con el medio ambiente

Procesos de Decisión de Markov (MDPs)

- En RL se tiene que decidir en cada estado la acción a realizar
- Este proceso de decisión secuencial se puede caracterizar como un procesos de decisión de Markov o MDP
- Un MDP modela un problema de decisión secuencial en donde el sistema evoluciona en el tiempo y es controlado por un agente
- La dinámica del sistema esta determinada por una función de transición de probabilidad que mapea estados y acciones a otros estados

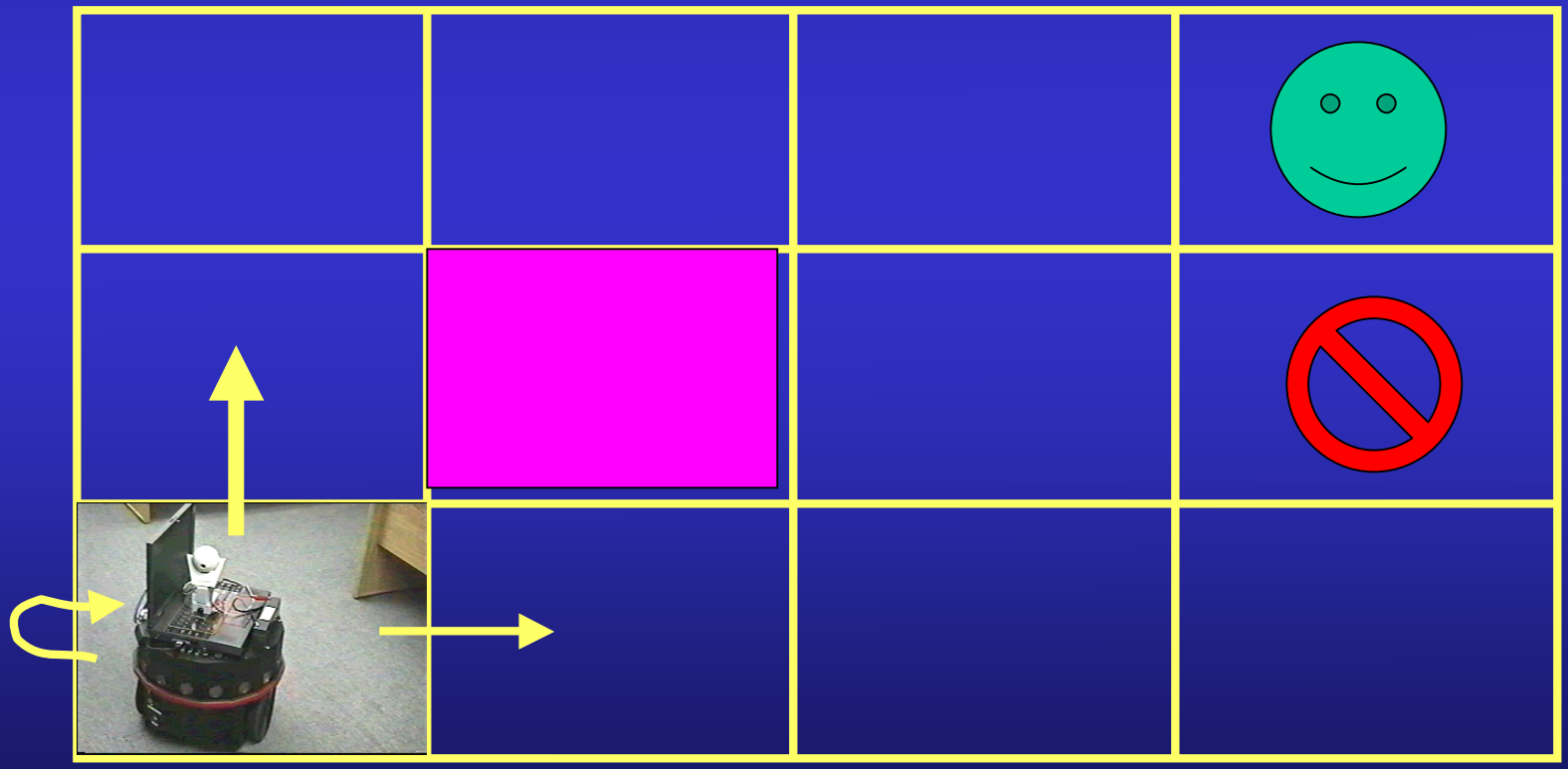
Modelo de Transición

- Normalmente existe incertidumbre respecto a los resultados de una decisión (acción)
- Esta incertidumbre se modela como una probabilidad de llegar al estado “j” dado que se encuentra en el estado “i” y se realizá la acción “a”:

$$P_{ij}^{a}$$

Modelo de Transición

- Probabilidad dirección deseada – Pij=0.8
- Probabilidad 2 direcciones vecinas – Pik=0.1



Procesos de Decisión de Markov

- Problema de obtener la política óptima en un ambiente observable – MDP
- El método clásico para resolver estos problemas se conoce como “iteración de valor” (*value iteration*)
- La idea básica es calcular la utilidad de cada posible estado y usar éstas para seleccionar la acción óptima en cada estado
- Otra forma de solucionar el problema es mediante al aprendizaje de la política a través de interactuar con el ambiente - aprendizaje por refuerzo

Procesos de Decisión de Markov

- Formalmente, un MDP (discreto) se define por:
  - Un conjunto finito de estados, S
  - Un conjunto finito de posibles acciones, A
  - Un modelo de transición, que especifica la probabilidad de pasar a un estado dado el estado presente y la acción, P(s | s', a)
  - Una función de recompensa, que especifica el "valor" de ejecutar cierta acción *a* en el estado *s*, r(s, a)

Ejemplo – recompensas

0	0	0	 +1
0		0	 -1
0	0	0	0

Utilidad

- La utilidad de un estado depende de la secuencia de acciones tomadas a partir de dicho estado (i) de acuerdo a la política establecida (π)
- En principio, se puede obtener como la utilidad esperada de todas las posibles secuencias de acciones (Hi) y la utilidad resultante para c/u:

$$U(i) = UE(\ Hi(\pi)\ ) = \Sigma\ P(Hi(\pi))\ U_h\ Hi(\pi)$$

Utilidad

- Si la utilidad es separable, se puede estimar como la utilidad del estado presente y la utilidad de los siguiente estados
- La forma más sencilla es que sea una función aditiva:

$$U[s_0, s_1, \ldots s_n] = R(s_0) + U[s_1, \ldots s_n]$$

- Donde *R* se conoce como la función de recompensa

Programación Dinámica

- Dada la condición de separabilidad, la utilidad de un estado se puede obtener en forma iterativa maximizando la utilidad del siguiente estado:

$$U(i) = R(i) + \max_a \sum_j P(s_j \mid s_i,a)\, U(j)$$

- La política óptima esta dada por la acción que de mayor utilidad:

$$\Pi^*(i) = \arg\max_a \sum_j P(s_j \mid s_i,a)\, U(j)$$

Horizonte finito vs. infinito

- Los problemas con un número finito de pasos se conocen como MDP de horizonte finito
- Los problemas en que puede haber un número infinito de pasos se conocen como MDP de horizonte infinito
- Muchos problemas, como el ejemplo del robot, son de horizonte infinito

Solución

- Los métodos principales para resolver MDPs son:
- Iteración de valor (Bellman, 57),
- Iteración de política (Howards, 60),
- Aprendizaje por refuerzo

MDPs – ecuaciones fundamentales

- Función de valor (ecuación de Bellman):

$$V^*(s) = \max_a \{ R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \}$$

- Política:

$$\pi^*(s) = \arg\max_a \{ R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \}$$

Donde γ es un factor de descuento

Solución

Función de valor

- Una política para un MDP es una asociación π:S $\rightarrow$A (acción por estado).
- Dada la política, el valor para horizonte finito es: V_n^{π}: S $\rightarrow \Re$

$$V_n^{\pi}(i) = R(i, \pi(i)) + \Sigma P(\pi(i) \mid i,j)\ V_{n-1}(j)$$

- Para horizonte infinito, generalmente se considera un *factor de descuento*, $0<=\gamma<1$:

$$V^{\pi}(i) = R(i, \pi(i)) + \gamma\Sigma P(\pi(i) \mid i,j)\ V(j)$$

Solución

Política óptima

- La solución a un MDP da una política óptima.
- Esto es, la política que maximiza la ecuación de Bellman:

$$\pi^*(i) = \max \left[R(i, a) + \gamma \sum P(a \mid i,j)\, V^*(j) \right]$$

Iteración de Valor

- En el caso de horizonte infinito, se puede obtener la utilidad de los estados –y la política óptima, mediante un método iterativo
- En cada iteración (t+1), se estima la utilidad de cada estado basada en los valores de la iteración anterior (t):

$$U_{t+1}(i) = R(i) + \max_a \sum_j P(s_j \mid s_i,a)\, U_t(j)$$

- Cuando t→inf, los valores de utilidad convergen a un valor estable

Iteración de Valor

Algoritmo:

- Inicializar: $U_t = U_{t+1} = R$
- Repetir:
  - $U_t = U_{t+1}$
  - $U_{t+1}(i) = R(i) + \max_a \sum_j P(s_j \mid s_i, a)\, U_t(j)$
- Hasta: $| U_t - U_{t+1} | < \varepsilon$

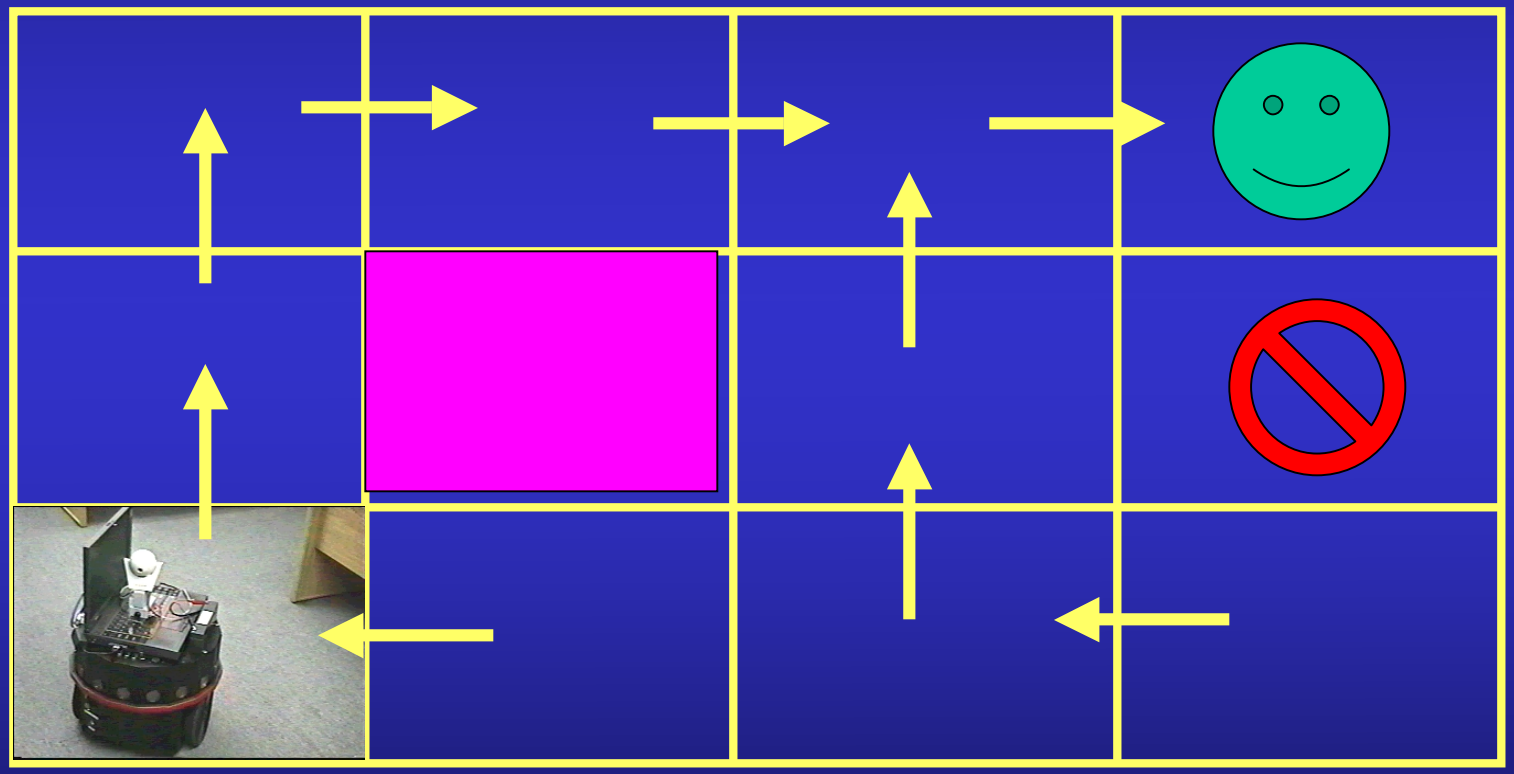
Iteración de Valor

- ¿Cuántas veces repetir la iteración?
- Normalmente el número de iteraciones para obtener la política óptima es menor que el requerido para que las utilidades converjan
- En la práctica, el número de iteraciones es relativamente *pequeño*

Ejemplo – utilidades de los estados

0.812	0.868	0.912	
0.762		0.660	
0.705	0.655	0.611	0.338

Ejemplo – política óptima



Solución sin modelo

- Iteración de valor requiere de un modelo (función de transición) del problema
- Lo métodos de aprendizaje no requieren de un modelo y se basan en explorar el ambiente:
  - Monte Carlo
  - Diferencias Temporales
  - Q-learning

Monte Carlo

- Los métodos de Monte Carlo, solo requieren de experiencia y la actualización se hace por episodio más que por cada paso.
- El valor de un estado es la recompensa esperada que se puede obtener a partir de ese estado.
- Para estimar V podemos tomar estadísticas haciendo un promedio de las recompensas obtenidas

Monte Carlo

- Repite
  - Genera un episodio usando π
  - Para cada estado *s* en ese episodio:
    - *R* ← recompensa después de la primera ocurrencia de *s*
    - Añade *R* a *recomp*(*s*)
    - *V* (*s*) ← promedio(*recomp*(*s*))

Políticas de exploración

- e–*greedy*: en donde la mayor parte del tiempo se selecciona la acción que da el mayor valor estimado, pero con probabilidad ϱ se selecciona una acción aleatoriamente.
- *softmax*, en donde la probabilidad de selección de cada acción depende de su valor estimado.

Mejorar políticas

- Con Monte Carlo podemos alternar entre evaluación y mejoras en base a cada episodio
- La idea es que después de cada episodio las recompensas observadas se usan para evaluar la política y la política se mejora para todos los estados visitados en el episodio

Algoritmos on y off-policy

- Los algoritmos *on-policy*: Estiman el valor de la política mientras la usan para el control. Se trata de mejorar la poítica que se usa para tomar decisiones.
- Los algoritmos *off-policy*: Usan la política y el control en forma separada. La estimación de la política puede ser por ejemplo *greedy* y la política de comportamiento puede ser ε-*greedy*. Es decir que la política de comportamiento está separada de la política que se quiere mejorar. Esto es lo que hace Q-learning, lo cual simplifica el algoritmo.

Diferencias Temporales

- Los algoritmos son incrementales y fáciles de computar
- Actualizan las funciones de valor usando el error entre lo estimado y la suma de la recompensa inmediata y lo estimado del siguiente estado
- El más simple TD(0) es:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]$$

Algoritmo TD(0)

Inicializa $V(s)$ arbitrariamente y π a la política a evaluar
Repite (para cada episodio):
    Inicializa s
    Repite (para cada paso del episodio):
        $a \leftarrow$ acción dada por π para s
        Realiza acción a; observa la recompensa, r,
            y el siguiente estado, s'
        $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s') - V(s)]$
        $s \leftarrow s'$
    hasta que s sea terminal

Q-Learning

- Uno de los desarrollos más importantes en aprendizaje por refuerzo fue´ el desarrollo de un algoritmo *off-policy* conocido como Q-learning.
- La idea principal es realizar la actualización de la siguiente forma (Watkins, 89):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Algoritmo

Inicializa $Q(s, a)$ arbitrariamente
Repite (para cada episodio):
    Inicializa s
    Repite (para cada paso del episodio):
        Selecciona una a de s usando la politica dada por Q (e.g., ϵ–greedy)
        Realiza acción a, observa r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma max'_a Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$;
    hasta que s sea terminal

Aprendiendo modelos

- Con un modelo podemos predecir el siguiente estado y la recomepensa dado un estado y una acción
- La predicción puede ser un conjunto de posibles estados con su probabilidad asociada o puede ser un estado que es muestreado de acuerdo a la distribución de probabilidad de los estados resultantes
- Lo interesante es que podemos utilizar los estados y acciones simulados para aprender. Al sistema de aprendizaje no le importa si los pares estado-acción son dados de experiencias reales o simuladas.

Dyna-Q

- El algoritmo Dyna-Q combina experiencias con planificación para aprender más rápidamente una política óptima.
- La idea es aprender de experiencia, pero también usar un modelo para simular experiencia adicional y así aprender más rápidamente

Algoritmo

Inicializa $Q(s, a)$ y $Modelo(s, a)$ $\forall s \in S, a \in A$

DO forever

$s \leftarrow$ estado actual

$a \leftarrow \epsilon$−$greedy(s, a)$

realiza acción a observa s' y r

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

$Modelo(s, a) \leftarrow s', r$

Repite N veces:

$s \leftarrow$ estado anterior seleccionado aleatoriamente

$a \leftarrow$ acción aleatoria tomada en s

$s', r \leftarrow Modelo(s, a)$

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

Extensiones

El principal problema de los MDPs es el crecimiento de la complejidad al aumentar el número de estados y acciones (el tamaño del problema crece exponencialmente con el número de variables de estado y acciones). Para ello se han planteado:

- Representaciones factorizadas
- Representaciones abstractas (agregación de estados)
- Modelos jerárquicos (serie / paralelo)

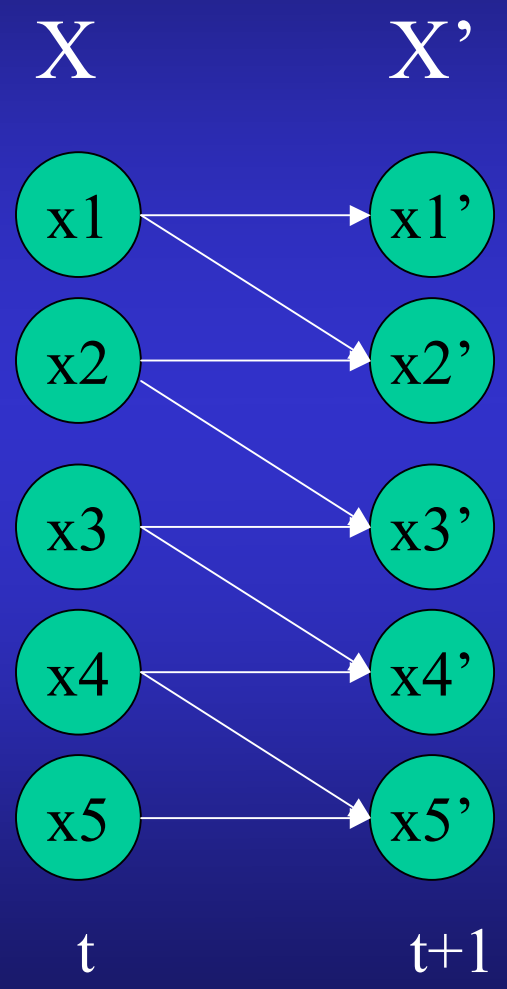
MDPs factorizados

- El estado de descompone en un conjunto de variables o factores
- Esto permite utilizar representaciones basadas en modelos gráficos para reducir la complejidad del modelo de transición y de recompensa:
  - El modelo de transición se representa usando RBD (una RBD de 2 etapas por acción)
  - La función de recompensa se representa usando árboles de decisión (considerando sólo las variables relevantes)

MDP factorizado

$X = \{x1, x2, x3, x4, x5\}$

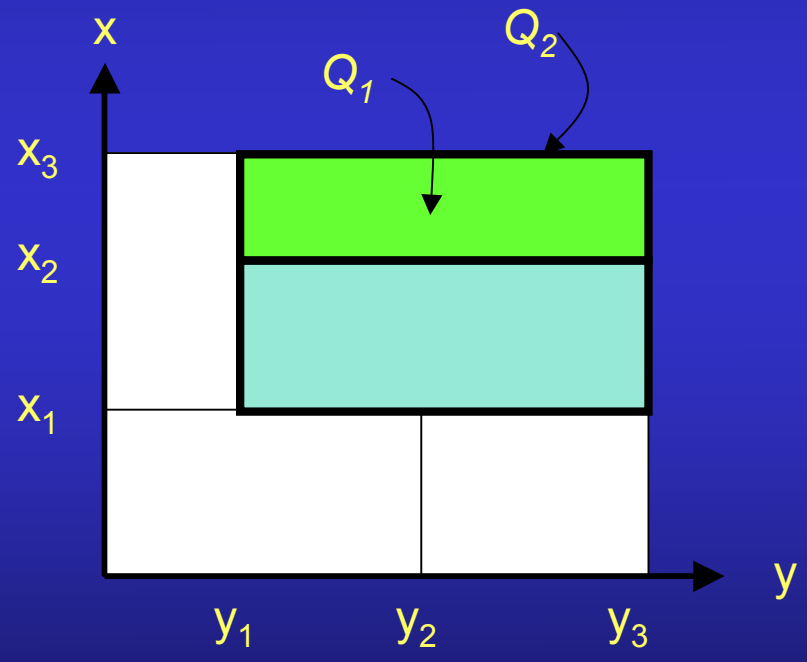
Se tiene una RBD por acción



MDPs abstractos

- Otra alternativa para reducir la complejidad es reducir el número de estados, agrupando estados con propiedades similares (recompensa, probabilidades de transición)
- Esto también se puede aplicar a dominios continuos
- La desventaja es que la solución puede ya no ser óptima

Estados abstractos (cualitativos)



x_1, x_2, x_3, y_1, y_2, y_3 son valores de referencia o corte

Q_1=*pos(x, x2), ~pos(x,x3), pos(y, y1), ~pos(y,y3).*

Q_2=*pos(x, x1), ~pos(x,x2), pos(y, y1), ~pos(y,y3).*

Aplicaciones

- Juego de Damas
- Backgammon - TD-gammon
- Balanceo del péndulo invertido
- Control de aviones y helicópteros
- Control de procesos industriales
- Robótica
- …

Aplicaciones

- Basado en este enfoque se han desarrollado diversas tareas para un robot de servicio:
  - **Robot mensajero**: lleva mensajes u objetos de una persona a otra
  - **Robot anfitrión**: recibe visitantes y los guía en una institución
  - **Navegación** (Robocup@home): va a diferentes lugares en una casa comandado por voz
  - **Busca y encuentra** (Robocup@home): busca un objeto
  - **Seguimiento** (Robocup@home): sigue a una persona

Markovito: hardware

- Robot PeopleBot
- Cámara Pan/tilt
- Micrófono direccional
- 2 anillos de sonares
- Pinza
- Laser
- 2 computadoras (interna & laptop)

Robot Mensajero

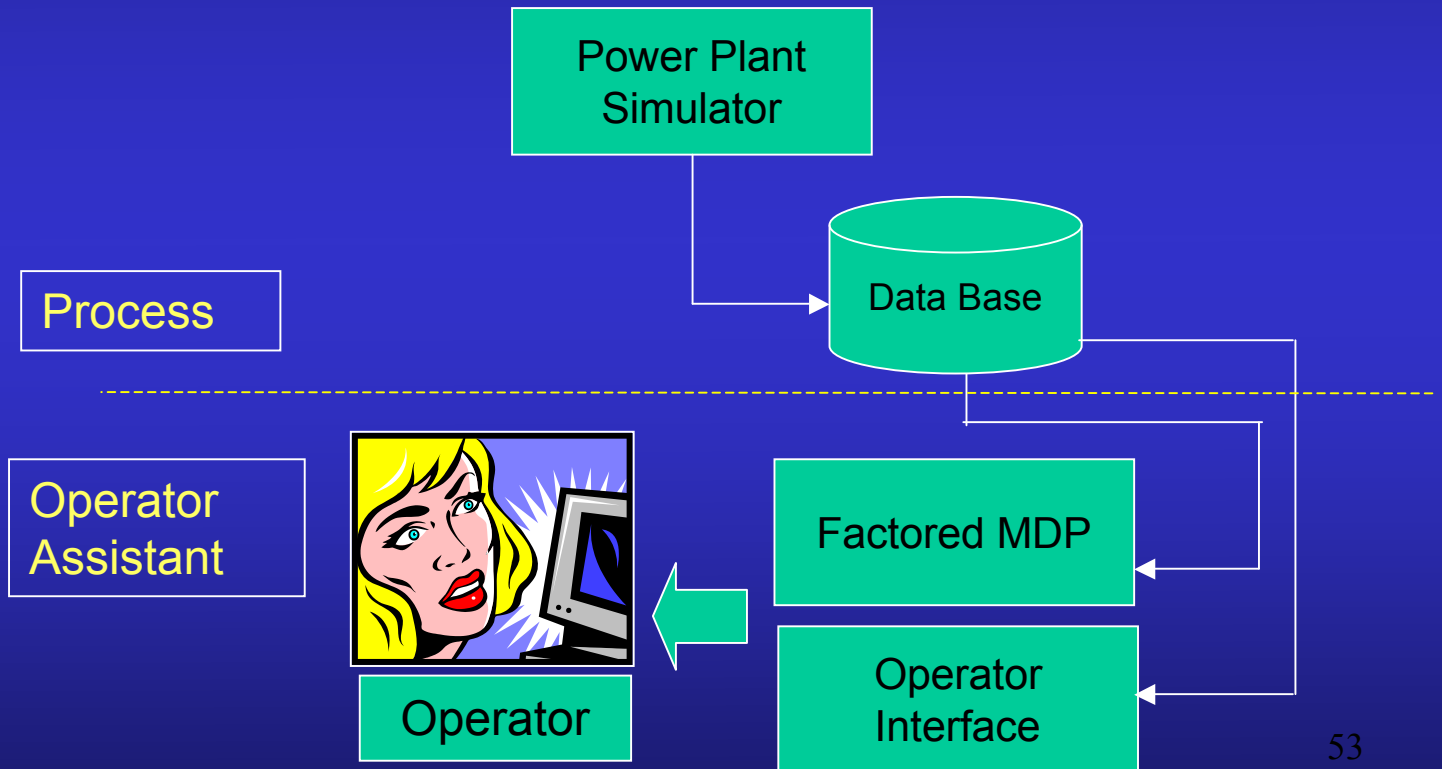
Otras tareas

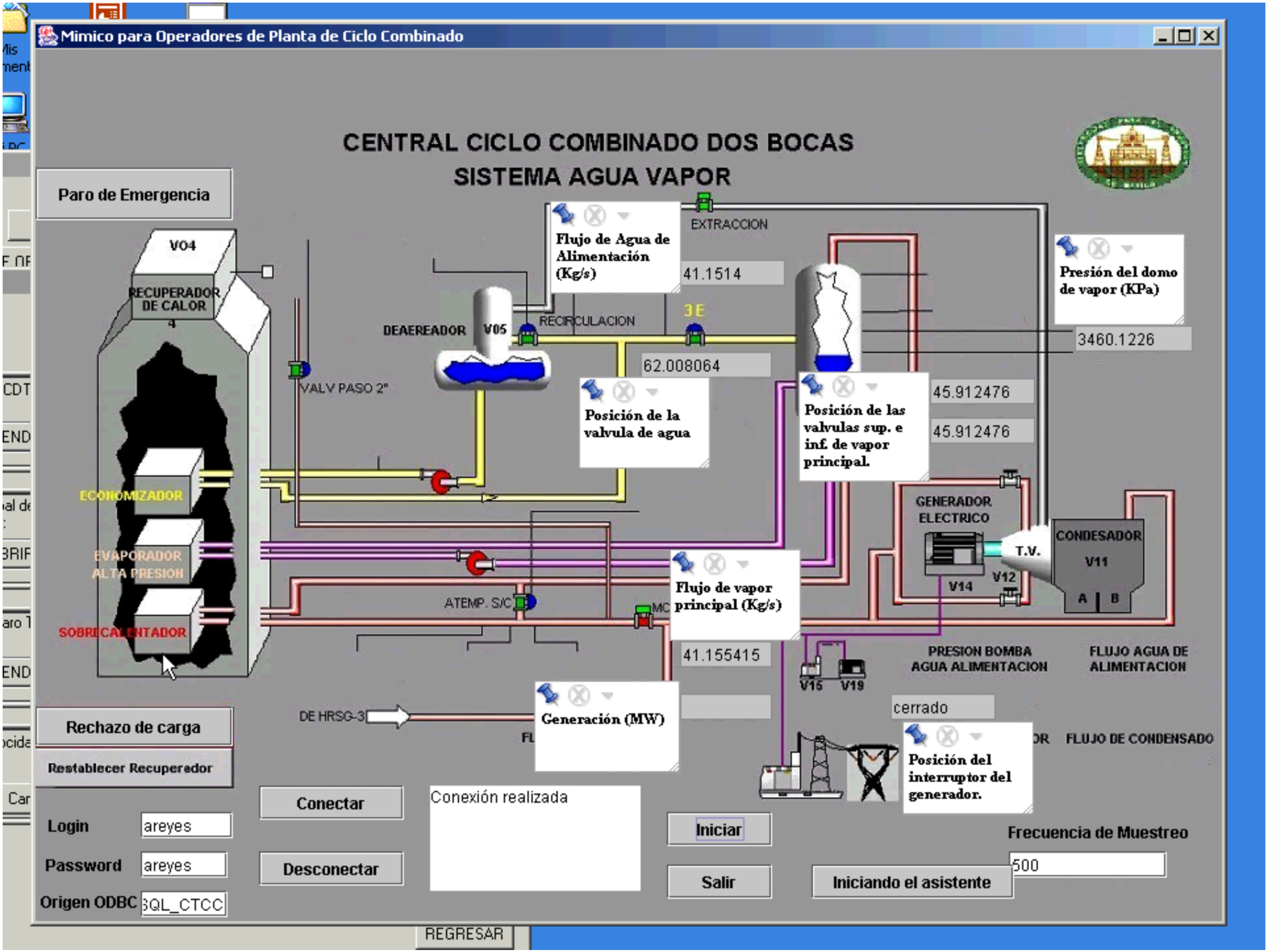
Navegación en un ambiente de “casa”

Enseñando a Markovito a Reconocer un objeto

Entregando una cerveza!

Asistente del Operador de Plantas Eléctricas




Mimico para Operadores de Planta de Ciclo Combinado
CENTRAL CICLO COMBINADO DOS BOCAS
SISTEMA AGUA VAPOR
Paro de Emergencia
V04
RECUPERADOR DE CALOR
4
ECONOMIZADOR
EVAPORADOR ALTA PRESION
SOBRECALENTADOR
DEAEREADOR
V05
RECIRCULACION
VALV PASO 2"
ATEMP. S/C
MC
DE HRSG-3
EXTRACCION
Flujo de Agua de Alimentación (Kg/s)
41.1514
3E
62.008064
Posición de la valvula de agua
Posición de las valvulas sup. e inf. de vapor principal.
45.912476
45.912476
Presión del domo de vapor (KPa)
3460.1226
GENERADOR ELECTRICO
T.V.
CONDESADOR
V11
A
B
V14
V12
Flujo de vapor principal (Kg/s)
41.155415
PRESION BOMBA AGUA ALIMENTACION
FLUJO AGUA DE ALIMENTACION
V15
V19
cerrado
Generación (MW)
FLUJO DE CONDENSADO
Posición del interruptor del generador.
Rechazo de carga
Restablecer Recuperador
Conectar
Conexión realizada
Login
areyes
Password
areyes
Origen ODBC
SQL_CTCC
Desconectar
Iniciar
Salir
Iniciando el asistente
Frecuencia de Muestreo
500
REGRESAR

Tarea

- Leer Capítulo 20 de Russell
- Presentación avance proyecto: modelo basado en redes bayesianas para VIH