Parte03

October 21, 2018

1 Introducción al Aprendizaje por Refuerzos

1.0.1 Curso Aprendizaje por Refuerzos, Diplomatura en Ciencia de Datos, Aprendizaje Automático y sus Aplicaciones

1.0.2 FaMAF, 2018

Agenda Clase 1 Parte 3

1.1 Aprendizaje por Refuerzos basado en Modelos.

- Los modelos del entono representan cualquier elemento que permita al agente predecir cómo el entorno responderá ante las acciones que ejecuta.
  Dado un estado y una acción, el modelo produce una una predicción del siguiente estado y el reward asociado.

- Si el modelo es estocástico, entonces existen distintos posibles estados siguientes/rewards, cada uno con distinta probabilidad de ocurrir.

- Algunos modelos producen una descripción de todas las posibilidades con sus probabilidades asociadas: los mismos se denominan **modelos de distribución**.
  Otros modelos producen solo una de las posibilidades, sampleadas en función de su probabilidad (la cuál es desconocida a priori por el agente); estos se denominan **modelos basados en ejemplos**.

- Ambos tipos de modelos pueden ser empleados para replicar o generar experiencia.
  Dado un estado-acción inicial, un modelo basado en ejemplos produce una posible transición, y un modelo de distribución genera todas las posibles transiciones pesadas por su probabilidad de ocurrir.
  De esa manera, un modelo basado en ejemplos podría producir un episodio completo, y un modelo de distribución, todos los posibles episodios y sus posibilidades.

- En ambos casos, decimos que el modelo es utilizado para producir **experiencia simulada**.

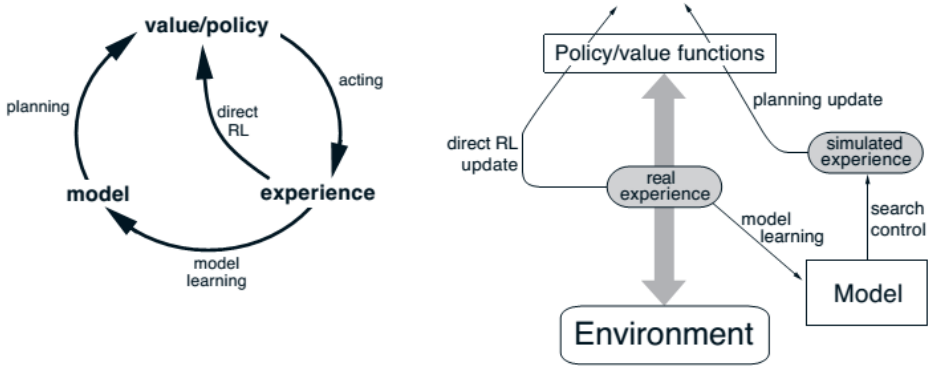
1.2 Arquitectura Dyna: Integración de Planning, Acting, y Learning.

- El término *planning* en entornos RL se utiliza para referirse a cualquier proceso computacional que emplea el modelo como entrada y produce o mejora una política de interacción con un entorno determinado.

- Cuando el *planning* es realizado on-line, mientras se interactúa con un entorno, surgen cuestiones importante que es necesario resolver.
  Por ejemplo, la nueva información proveniente de la interacción podría impactar en el modelo y de esa manera modificar el proceso de *planning*.

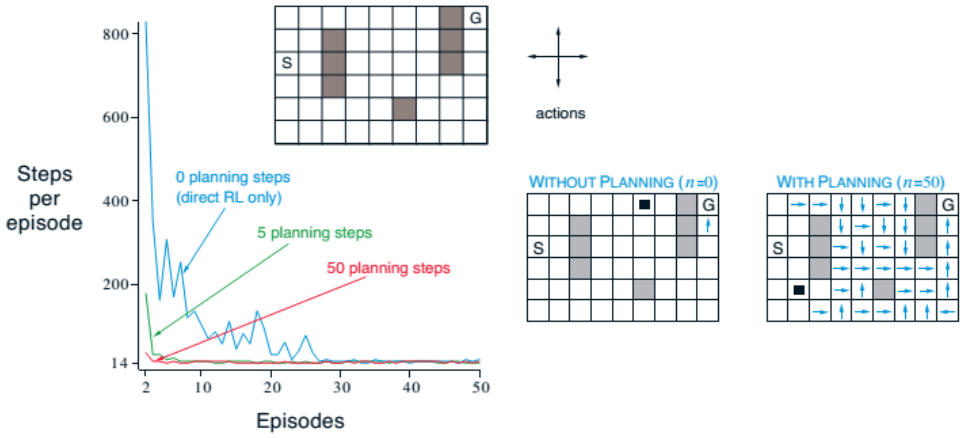
- Si la toma de decisiones y el aprendizaje de modelos (*decision-making y model-learning* respectivamente) son procesos basados en computación intensiva, entonces los recursos computacionales deberían ser divididos entre ambos.

- Dyna-Q es una arquitectura que integra las funciones más importantes para un agente que realiza planificación (*planning*) on-line.
  La misma, establece al menos dos roles para la experiencia real generada por el agente: el primero, establece que la misma puede ser empleada para mejorar el modelo (*model learning*, hacer que refleje la realidad lo mejor posible), o bien mejorar la función de valor y por ende la política, empleando los métodos basados en Aprendizaje por Refuerzo (*direct-RL*).



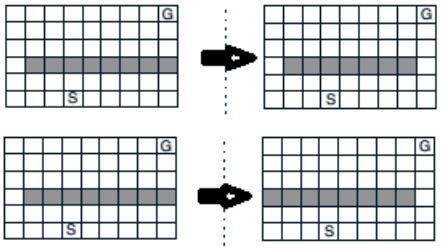
Tabular Dyna-Q

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
  $R, S' \leftarrow Model(S,A)$
  $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$



1.3 Corrección de los modelos

- En muchos entornos determinísticos, una vez aprendido el modelo, puede utilizarse de manera directa para propagar cambios producidos por las Diferencias Temporales.
- En entornos no determinísticos, o en aquellos en los cuales se de el hecho de que en algún momento de la interacción agente-entorno este último cambia, es necesario incorporar mecanismos de auto-corrección de modelos para evitar obtener políticas sub-óptimas.



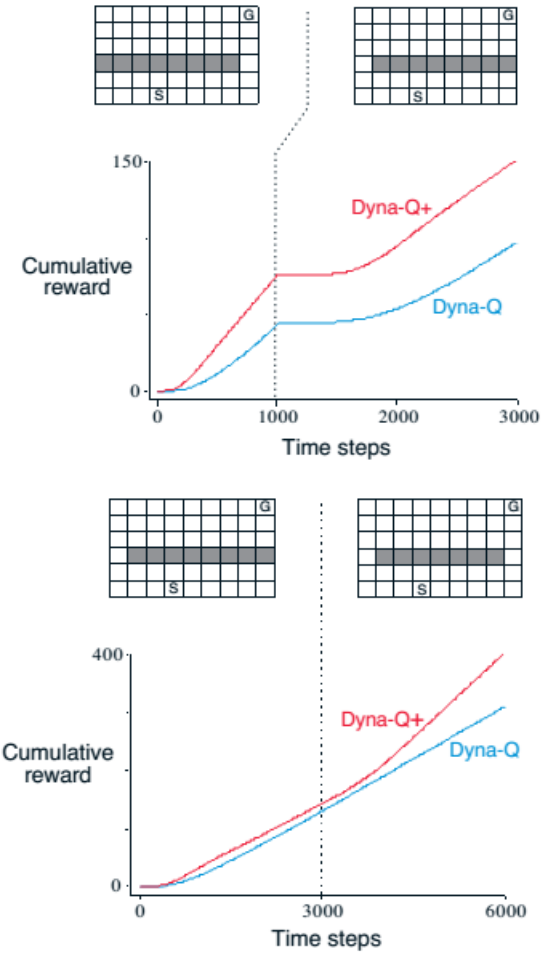
- El problema general de corrección de modelos es otra versión del trade-off exploración-explotación.
- En el contexto del *planning* en RL, la exploración significa ejecutar acciones que mejoran el modelo, mientras que la explotación significa comportarse de manera óptima de acuerdo a lo establecido por el modelo.
  Es decir, se busca que el agente explore para detectar cambios en el entorno, pero sin degradar la performance del mismo de manera determinante.

1.3.1 Dyna-Q+

- El método *Dyna-Q+* permite resolver el trade-off mencionado empleando una heurística basada en la cantidad de tiempo que un par estado-acción no es visitado de manera real por el agente.
- Mientras más tiempo haya pasado desde una interacción real, mayor es la probabilidad de que las dinámicas en determinados pares estado-acción hayan cambiado, y que por lo tanto, el modelo sea incorrecto.

- Para motivar el testeo de acciones que hace largo tiempo no se ejecutan en determinado estado, se otorga un *bonus especial* a las acciones simuladas que tienen en cuenta dichas acciones.

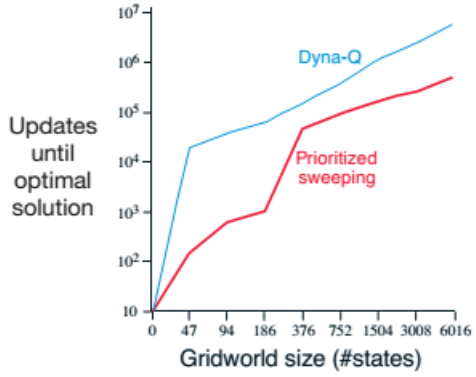
- En particular, si la recompensa para una transición determinada es R, y dicha transición no ha sido ejecutada realmente en time steps, entonces las actualizaciones basadas en *planning* son llevadas a cabo como si la misma produjera una recompensa de $R +$, para un comprendido entre 0 y 1.
  El parámetro se denomina **curiosidad computacional**.



1.3.2 Prioritized Sweeping

Prioritized sweeping for a deterministic environment

Initialize $Q(s,a)$, $Model(s,a)$, for all s,a, and $PQueue$ to empty
Loop forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow policy(S,Q)$
- (c) Take action A; observe resultant reward, R, and state, S'
- (d) $Model(S,A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$.
- (f) if $P > \theta$, then insert S,A into $PQueue$ with priority P
- (g) Loop repeat n times, while $PQueue$ is not empty:
  - $S,A \leftarrow first(PQueue)$
  - $R,S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
  - Loop for all $\bar{S},\bar{A}$ predicted to lead to S:
    - $\bar{R} \leftarrow$ predicted reward for $\bar{S},\bar{A},S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S,a) - Q(\bar{S},\bar{A})|$.
    - if $P > \theta$ then insert $\bar{S},\bar{A}$ into $PQueue$ with priority P



In [ ]: