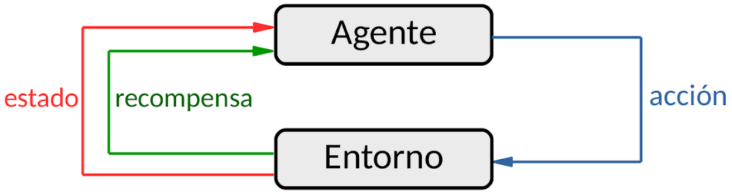
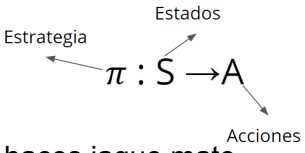
Reinforcement learning = APRENDIZAJE POR REFUERZO

El aprendizaje por refuerzo es muy parecido a como aprendemos los seres vivos.



Hay un **AGENTE** que tiene sensores para observar el **estado** de su entorno, puede realizar **acciones** sobre el estado y modificarlo y cada accion tiene una **recompensa** numérica

ESTRATEGIA: acciones que aprende el agente para maximizar la recompensa



Un **estado final es** donde el agente termina, por ejemplo te hacen jaque mate o haces jaque mate

Cada decisión que toma el agente lo va llevando a distintos estados (maquina de Turing).
Las acciones serían lo que me lleva de un nodo a otro. Los estados son los nodos. A través de la estrategia que aprende va tomando las acciones que más le convengan.

<u>Diferencias con el aprendizaje supervisado</u>

Las recompensas vienen con **demora**. Viene cuando se acaba con el episodio.
EPISODIO: empezar y terminar la tarea que tiene que realizar (ejemplo: un juego, la recompensa viene cuando gano o pierdo)

Debe aprender a discernir cuáles acciones de una secuencia fueron meritorias.

<u>Exploración</u>: nuestro agente va explorando todas las acciones posibles y así entrena. No hay set de datos, sino que el agente va explorando todas las acciones posibles.

No tiene que recolectar datos (ventaja) pero puede ser inabarcable la cantidad de estados (desventaja).

<u>Proceso de decisión de Markov</u>

Conjunto de **estados S**: son todos los estados en que se puede terminar.
Conjunto de **acciones A**: para realizar en cada estado, no en todos los estados vamos a tener las mismas acciones.
Función de transición T: dado un estado y una acción me devuelve el siguiente estado.
Función de recompensa R: en un estado realizo una acción y obtengo una recompensa.

FUNCION DE VALOR: dada una política, me devuelve la **recompensa acumulada** que se consigue al seguir la política π a partir del estado S. $V^{\pi}(s) = r_0+\gamma r_1+\gamma^2 r_2+ .. = \sum^{\infty}_{t=0}\gamma^t r_t$

Gamma: Es el **factor de descuento**. Cada estado que tenes que pasar hasta llegar al final le va descontando un poquito de recompensa, porque cuanto más alto se te haga el camino hasta llegar menos recompensa tenes.
Además de maximizar la recompensa, queremos que el camino sea el más corto.

Definición recursiva:

Ecuación de Bellman: define nuestra recompensa acumulada. la cuenta como la recompensa inmediata + el siguiente estado. Es una forma recursiva de definirlo.

Funcion Q(s, a): Nos dice la máxima ganancia esperada a partir del estado s ejecutando la accion a

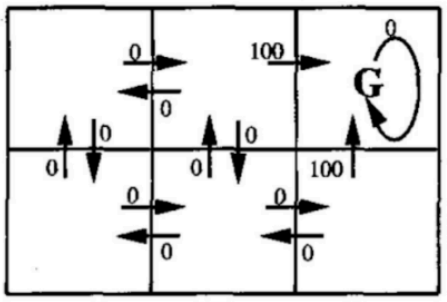
¿Cómo funciona?

Inicializar Q(*s*,*a*)
Repetir (para cada episodio):
    Inicializar *s*
    Repetir:
        Elegir la acción *a* en *s* y ejecutarla
        observar la recompensa r en el nuevo estado 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
Definimos $\pi(s) = \text{argmax}_a\, Q(s,a)$

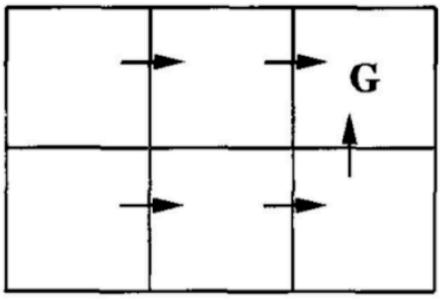
Alpha es el **learning rate**. Esta para decir que tanto quiero aprender de lo que logré, porque si lo que hice no está bien no quiero que se actualice demasiado.

Cuando terminan los episodios, la política es el mejor Q (lo que conocio como mejor opcion en cada caso).

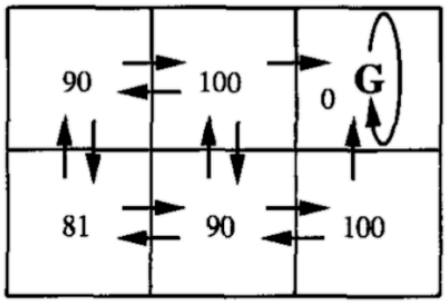
Función de recompensa *R*:



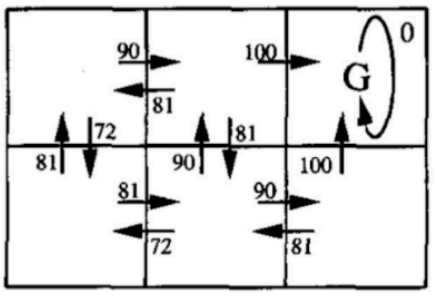
Política óptima π^*:



Función de valor V^*:



$Q(s, a)$:



La diferencia entre V y Q es que V te dice si arrancas en un slot , usando la politica, cuanta seria la ganancia. y Q te dice para cualquier accion que tomes, cual es la ganancia que tendrias si haces esa accion y despues seguis con la politica

Dilema exploración-explotación

En los casos que ya había tomado una acción que funcionó bien, ¿que me conviene? Explorar mejores opciones o explotar lo que ya sabemos? Tenemos episodios limitados. Esto es un nuevo trade-off.

Ejemplo: K-armed bandits. Aparecen distintas estrategias. Heurísticas:

- **ε- greedy:** Con cierta probabilidad ε elige al azar, y con 1-ε elige lo que ya conocía.
- **ε- first**: primero explorar y luego explotar. En los primeros (1-ε)% elegimos al azar y en los restantes ε% elegimos el mejor brazo conocido.

Las computadoras le ganan a los profesionales porque los profesionales no tienen todo el grafo con todas las acciones posibles en su cabeza.

Deep Q learning: Aprendizaje por refuerzos a través de deep learning.
Se mezcló redes neuronales convolucionales con la idea de Q-learning.

La capa convolucional está al principio y se usa para alimentar a la red neuronal con la imagen del juego. No se le pone el estado, sino que directamente el frame. En la capa de salida se pone que acción tomo para que se produzca ese movimiento.
Además se cambió la función de costo para que sea una función de costo parecida a la del Q-learning.

¿Cuándo es más dificil que se logre aprender bien en deep q learning? cuadno hay que representar un estado muy largo para atras. por ejemplo en un juego donde una palanca que moviste muchas jugadas antes, tiene un efecto en algo mucho despues.