TP: Q-learning

immediate

22 de noviembre de 2016

1. Introducción

En este trabajo exploramos el algoritmo de Q-learning para entrenar jugadores virtuales del juego *4 en línea*. El algoritmo, en su versión más básica, consiste en explorar la mayor cantidad de configuraciones posibles del juego, penalizando o recompensando las distintas acciones que los jugadores realizan. Al finalizar el entrenamiento los jugadores virtuales tienen una valorización de algunas de las acciones posibles para una fracción del total de configuraciones posibles del juego.

1.1. *4 en línea*

El *4 en línea* es un juego de mesa para dos jugadores que consiste en introducir fichas en un tablero vertical, con el objetivo de colocar cuatro fichas consecutivas del color correspondiente, ya sea en forma vertical, horizontal, o diagonal. Gana el primer jugador que alcanza esa configuración, y en caso que el tablero se complete antes de que algún jugador lo logre (ver figura 1 , se produce un empate. Típicamente el tablero tiene un tamaño de 6 filas por 7 columnas, lo que da un total de 4.531.985.219.092 configuraciones posibles.

2. Algoritmo de Q-learning

El algoritmo consiste en explorar la mayor cantidad de configuraciones posibles del juego. Durante el entrenamiento (etapa de exploración), los jugadores actualizan una magnitud $Q(s, a)$ que representa la ganancia o valorización de realizar la acción a en el estado s. Cuando un jugador realiza una acción ganadora (aquella que lo conduce a un estado con cuatro fichas

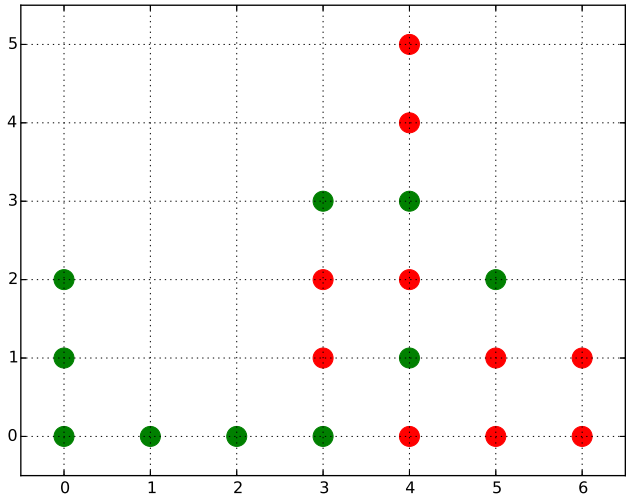


Figura 1: Posición donde el jugador verde gana.

alineadas) recibe una recompensa lo que lo lleva, en el transcurso del entrenamiento, a valorizar positivamente las acciones que lo acercaron a dicho estado.

Finalizado el entrenamiento, la mejor estrategia teórica consiste en, dado un estado s, elegir la acción a que maximize $Q(s, a)$ (etapa de explotación).

2.1. Estructura de datos

Almacenamos la ganancia $Q(s, a)$ en un diccionario cuyas entradas son un estado representado con una variable *string* de 42 caracteres de largo. Así por ejemplo el *string* '0021000....0' representa un estado donde el primer jugador tiene una ficha en la cuarta columna, el segundo jugador en la tercera, y el resto del tablero se encuentra vacío. Inicializamos los valores de $Q(s, a)$ con un número real aleatorio entre 0 y 1.

2.2. Representación del problema

Decidimos pensar el juego como la evolución de un único sistema que puede encontrarse en cualquiera de los estados mencionados en la introducción. Para ello en lugar de distinguir entre dos jugadores observamos que para cada estado queda inmediatamente determinado cuál de ambos con-

trincantes debe mover por la paridad del número de fichas del mismo; así si el sistema tiene una cantidad par de fichas, necesariamente juega el primer jugador en alguna de las columnas disponibles, caso contrario lo hace el segundo.

Por otro lado, consideramos que una correcta actualización de la ganancia $Q(s, a)$ viene dada por la ecuación 1, donde $R(s, a)$ es la recompensa de realizar una acción a en el estado s, s' es el estado obtenido al realizar a en s, y a' es cualquiera de la acciones posibles en el estado s'.

$$Q^{n+1}(s, a) = Q^n(s, a) + \alpha(R(s, a) - \gamma(max_{a'}Q^n(s', a') - Q^n(s, a)) \quad (1)$$

Dado un estado, la valorización de una acción:

- aumenta si al realizar una acción, de esta se obtiene una recompensa (en este caso se gana el juego)
- disminuye si la mejor acción del siguiente estado (que corresponde al otro jugador) está muy valorizada. Es decir cada jugador trata de tomar acciones que no lleven a un estado donde el otro jugador tenga una muy buena jugada.

Pensándolo de esta manera, en el algoritmo solo se introducen recompensas positivas. Cuando un jugador gane y reciba una recompensa, el otro jugador desvaluará la acción que llevó al sistema a ese estado, y valorará más las otras. Utilizamos $\alpha = 0{,}01$, $\gamma = 0{,}9$, y una recompensa $R = 100$.

2.3. Exploración de los estados: temperatura

Durante la fase de entrenamiento no existe *a priori* ninguna restricción sobre las acciones que el sistema decida explorar; estas pueden ser escogidas totalmente al azar. Siguiendo esta estrategia, y al cabo de cierto tiempo, la exploración recorrerá múltiples veces todos los pares (s, a) logrando una valorización adecuada de todas las acciones posibles para todos los estados. Sin embargo dada la ingente cantidad de estados posibles mencionada en la introducción, aun en un tablero pequeño esta estrategia resulta impracticable en un tiempo razonable. Resulta entonces útil introducir una magnitud (temperatura) que permita focalizar la búsqueda en aquellas ramas del árbol de jugadas que, a la luz del entrenamiento transcurrido, parezcan la más promisorias. Cuando la temperatura sea máxima el sistema elegirá en forma equiprobable entre las acciones posibles (exploración pura), mientras que cuando sea mínima escogerá aquella acción que maximize Q (explotación pura).

En suma, la disminución progresiva de este parámetro nos permite variar la estrategia de entrenamiento desde una puramente exploratoria y abarcativa hacia otra focalizada sobre las ramas más promisorias del árbol de jugadas.

2.4. Modificaciones agregadas

ACA IRIAN LAS COSAS QUE LE PUSIMOS DESPUES, DE QUE GANE EL JUEGO SI TIENE 3, ETC..

Suponiendo que el algoritmo no iba a converger, generamos una versión paralalea "con visión". En esta versión los jugadores no dejen pasar un 4 en línea, y que no elijan una jugada que deja un 4 en lińea del oponente. Con saben un jugar a un paso de distancia del objetivo. A temperatura 0 no se dejan perder y no desaprovechan una victoria cuando pueden. Sin embargo los jugadores tienen que aprender el valor del resto de las acciones. Al empezar solo entienden cuando la situación está en frente suyos. Lo que tienen que aprender es a llevar el juego hacia esos estados, no llegar por azar ahí. En un contexto donde los jugadores tienen este tipo de visión la única forma de ganar es aprendan a armar las "trampas.[en] las que el oponente no puede bloquear, esto es, dejar dos posibles 4 en línea.

3. Resultados

Dejamos corriendo una versión del qleraning desde el sábado a la mañana hasta la un último backup del Q el mártes a la mañana.

Con este Q corremos una competencia entre los jugadores. Dudamos de que haya aprendido bien. Es enspecial el jugador 2. Por eso corremos una version de competenecia con el jugador 1 al azar y el 2 basado en los valores de Q. Lo que obtenemos es lo siguiente.

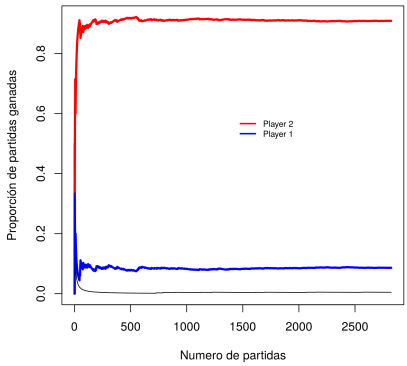


Figura 2: Jugador 1 al azar. Jugador 2 basado en los valores Q

En este experimiento dejamos la "temperatura". Ahora, hacemos un experimiento simiolar, perto variando la temperatura de un solo jugador, y dejando el otro jugando al azar. Hacemos esto para los dos jugadores. Lo que obtenemos es lo siguiente.

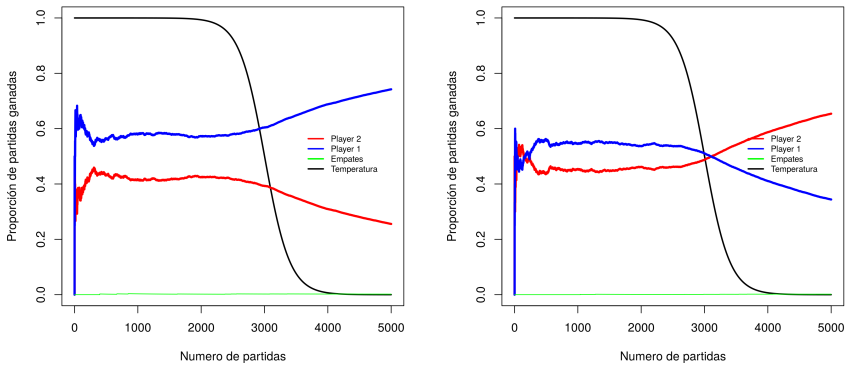
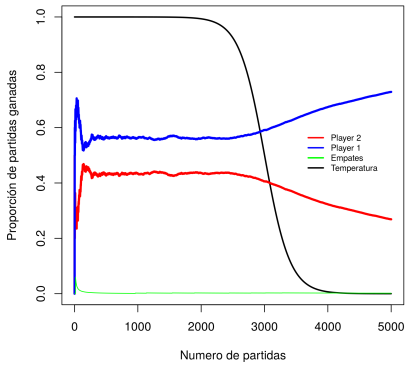


Figura 3: Un jugador al azar y un jugador bajando la temperatura

La versión modificada de actualización de Q "con visión" también aprendió. Esta versión la dejamos corriendo del sábado a la mañana hasta el domingo a las 23.59hs. Usando el Q resultante corrimos una competencia de los jugadores, ahora sin visión, basados solo en el Q. En el experimiento en el que se varía la temperatira de un jugador y se el otro jugador fijo en el azar, los resultados son los siguientes.



4. Estrategias exploradas

Utilizando el algoritmo presentado en la ecuación 1 exploramos la competencia entre dos jugadores, donde uno de ellos juega toma siempre una acción al azar, mientras que el otro adopta alguna estrategia, que en nuestro caso siempre será el jugador número 2. El jugador que comienza es elegido al azar.

4.1. E-greedy

Una de las estregias exploradas es la *e-greedy*, en la cual el jugador número 2, elije con probabilidad ϵ una acción al azar, y con una probabilidad $1-\epsilon$, la acción con la mayor valorización. Exploramos para valores de $\epsilon = 0{,}1$, $0{,}5$, $1{,}00$, y $0{,}00$. Las dos últimas se corresponden, respectivamente, al caso donde ambos jugadores juegan al azar, y cuando el jugador número 2 siempre juega a la acción de mayor valorización, es decir, prácticamente no explora. En la figura 5 observamos la evolución de las victorias acumuladas de cada jugador en función del número de jugadas.

De la figura 5 concluímos que un valor bajo ϵ, pero no nulo, ayuda al jugador 2 a sacar cierta ventaja del 1. En este caso, el jugador explora el $10\,\%$ de las veces y el resto juega a la mejor jugada. Observamos que a partir del paso 10000, la exploración inicial ya es suficiente, y comienza a sacar ventaja de la información ganada durante ese lapso.

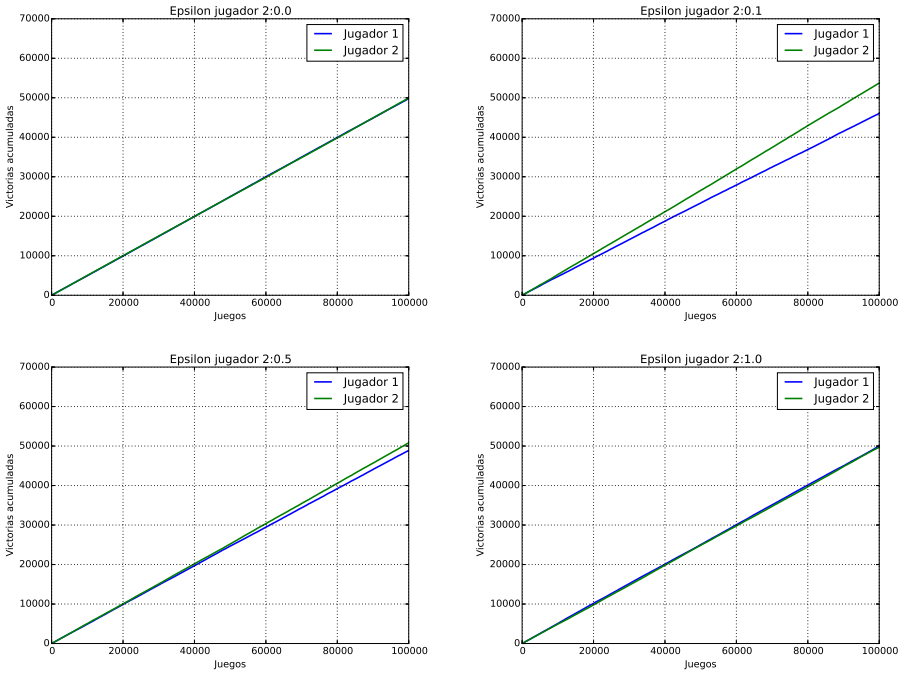


Figura 5: Victorias acumuladas en función del número de juegos realizados. Se observa que el jugador 2 saca ventaja al implementar una estrategia *e-greedy* con ϵ bajos. De izquierda a derecha, y de arriba hacia abajo, $\epsilon = 0{,}00$, $0{,}10$, $0{,}50$ y $1{,}00$.

4.2. Softmax

El siguiente paso que buscamos explorar fue la de implementar una estrategia softmax, en la cual la elección de la acción se da con una probabilidad dada por la ecuación 2, donde T es la temperatura del sistema. La idea original era la implementación de una temperatura alta e ir disminuyendola gradualmente, de esta forma el jugador que adopte esta estrategia explorará al principio la mayor cantidad de estados con probabilidad casi uniforme, y a medida de que la temperatura sea más baja, elegirá con mayor probabilidad los estados con Q mayor. Sin embargo, en el caso anterior observamos que los valores Q no estaban acotados, tanto superior como inferiormente, por lo tanto en el cálculo de la exponencial tuvimos errores númericos, por lo que nos resultó muy dificil explorar este algortimo.

$$P(s,a) = \frac{\exp(Q(s,a)/T)}{\sum_{a'} \exp(Q(s,a')/T)} \tag{2}$$

5. Conclusiones