Q-learning

Gustavo Landfried, Guillermo Pasqualetti, Sebastián Pinto

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

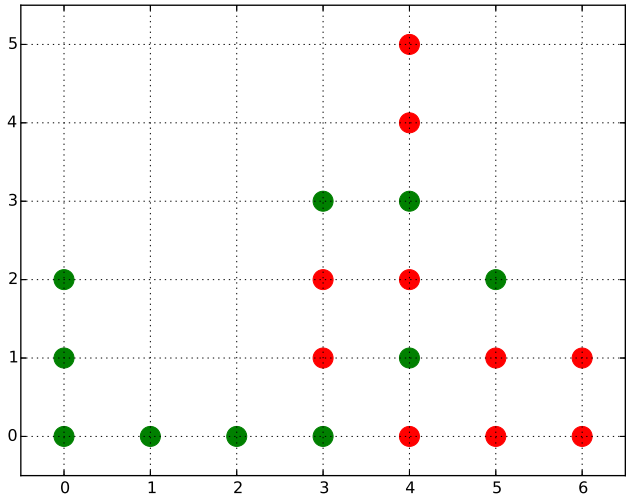


Figura 1: Posición donde el jugador verde gana.

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

Suponiendo que el algoritmo podía no converger, creamos una versión alternativa como estrategia para acelerar el proceso de aprendizaje. En este nueva versión dotamos a los jugadores de un grado de "visión", reconocen la presencia de 3 fichas en línea propias o agenas. En caso de que el jugador se encuentre a 3 fichas propias en línea, elegirá siempre la acción que deja 4 en línea y lo lleva a ganar. Si no tuviera la posibilidad de ganar, revisa si hay 3 en línea del oponente, y elegirá siempre la acción que bloquea al oponente la posibilidad de hacer 4 en línea.

En un juego de estas características los jugadores solo ganan si logran generar con una acción dos 3 en línea. En estos casos el oponente no puede impedir que el jugador gane. Cuando bloquea un 3 en línea, queda otro disponible.

En esta nueva versión los jugadores deben aprenden el valor de los miles de millones de acciones que no llevan a hacer 4 en línea o bloquear un 4 en línea. Tienen que aprender a ganar, es decir, a armar las "trampas" en las que una única acción genera "varios" posibles 4 en línea.

Con temperatura 0 los jugadores eligen la accion a que en el estado s maximiza $Q(s, a)$. Por defecto las acciones $a*$ que llevan a hacer un 4 en línea tienen un valor positivo de 1000 puntos, y las acciones $a**$ que llevan a no bloquear un 4 en línea de un oponente tiene un valor negativo de -1000 puntos. El resto de las acciones se inicializan con un valor aleatorio entre 0 y 1. Alcanzar un 4 en línea da un premio de 100 puntos. Cuando un jugador juega al azar no evalúa los valores $Q(s, a)$.

3. Resultados

El algoritmo alternativo que dotó a los jugadores con un grado de visión, alcanzó un buen nivel de aprendizaje luego de correr durante 3 días. El experimiento corrió con tempreatura 0. Esto quiere decir que durante todo el proceso de aprendizaje los juagores decidieron qué acción tomar basándose siempre en los valores $Q(s, a)$ aprendidos. Esto que fue producto de un error tuvo un resultado positivo.

Para evelauar si hubo aprendizaje corrimos varios experimento. En el primero hicimos que el jugador 1 eligiera acciones al azar mientras que el jugador 2 se base siempre en los valor $Q()$. Durante la competecia los jugadores no tienen ningún grado de visión. El jugador 2 sólo puede basarse en los valores aprendidos.

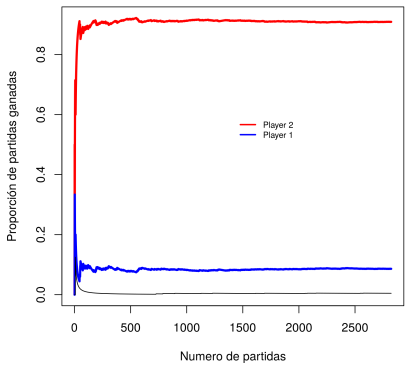
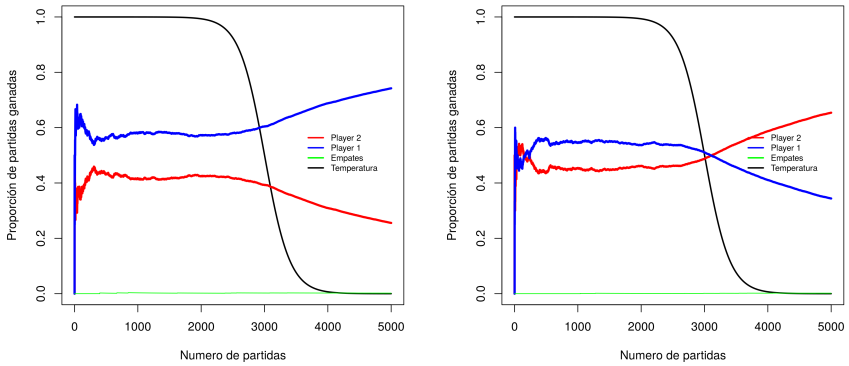


Figura 2: Jugador 1 al azar. Jugador 2 basado en los valores Q

En este caso ocurre lo que se esperaba. El jugador 2 ganara casi siempre. No gana siempre dado que el jugador 1 recorre ahora caminos al azar, algunos de los cuales no fueron explorados durante la fase de entrenamiento.

En el segundo experimento dejamos a un jugador eligiendo las acciones al azar y al otro le fuimos variando la proporción de veces que se basa en los valores $Q()$ o al azar. Hicimos esto para los dos jugadores. Lo que obtuvimos es lo siguiente:



(a) Jugador 2 100 % al azar. **(b)** Jugador 1 100 % al azar

Figura 3: Un jugador 100 % al azar. El otro con azar variable.

Se puede ver en ambos experimentos que cuando los dos jugadores juegan al azar tiene rendimientos similares, aunque siempre son mayores para el jugador que empieza primero. Cuando uno de los jugadores reduce la proporción de veces que se basa en el azar para elegir una acció se puede ver como empieza a mejorar su rendimiento. Esto pasa para ambos jugadores.

4. Estrategias exploradas

Utilizando el algoritmo presentado en la ecuación 1 exploramos el impacto de la estrategia adoptada sobre el desempeño de juego. Convenimos en entrenar siempre al jugador 2 y medir su desempeño al enfrentarse con un contendiente (jugador 1) que elije siempre acciones al azar (temperatura infinita). Para anular cualquier posible impacto de la iniciativa sobre el desempeño definimos de manera equiprobable qué jugador mueve primero en cada partido, así en promedio ambos jugadores comenzarán la misma cantidad de veces.

4.1. E-greedy

Una de las estregias de aprendizaje que estudiamos fue la *e-greedy* en la cual el jugador número 2 elije con probabilidad ϵ una acción al azar y con probabilidad $1 - \epsilon$ la acción con la mayor valorización. Se trata de una estrategia que combina exploración y explotación en proporción constante. Profundizamos para los siguientes valores de $\epsilon = 0{,}1$, $0{,}5$, $1{,}00$, y $0{,}00$.

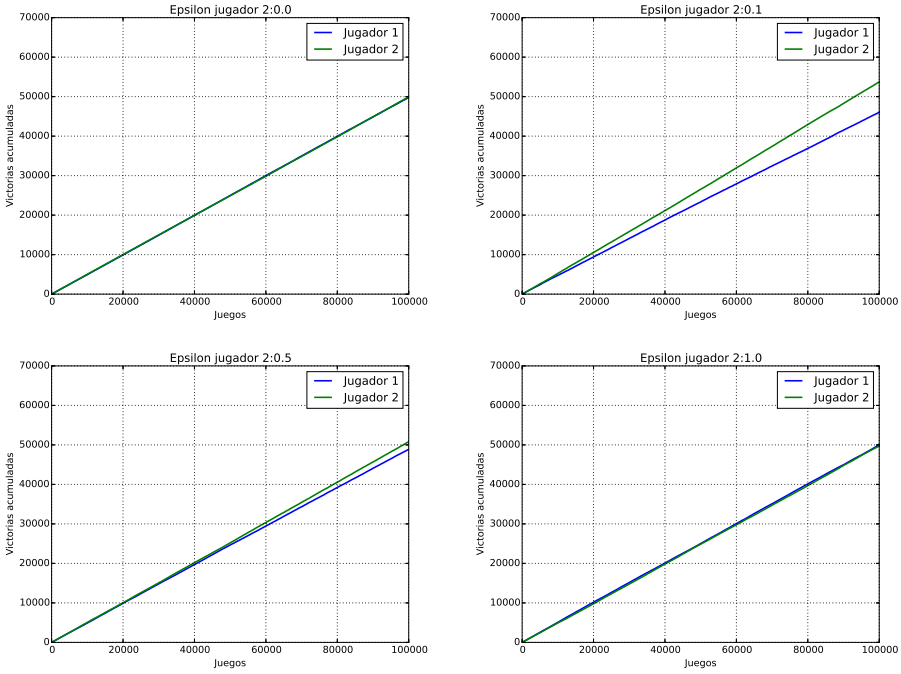


Figura 4: Victorias acumuladas en función del número de juegos realizados. Se observa que el jugador 2 saca ventaja al implementar una estrategia *e-greedy* con ϵ bajos. De izquierda a derecha, y de arriba hacia abajo, $\epsilon = 0{,}00$, 0,10, 0,50 y 1,00.

Las dos últimas se corresponden, respectivamente, con los casos puramente exploratorio y puramente explotatorio. En la figura 4 observamos la evolución de las victorias acumuladas de cada jugador en función del número de jugadas.

De la figura 4 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. Cabe mencionar que solo entrenamos durante 100000 partidas, lo que da lugar a un número de estados visitados que podemos acotar superiomente por 42 x100000, que es del orden de la millonésima parte del total de estados, mencionados en la introducción.

4.2. Softmax

Otra estrategia que exploramos fue *Softmax*, en la cual la acción es escogida con una probabilidad dada por la ecuación 2, donde T es la temperatura del sistema. La idea original fue la implementación de una temperatura inicial alta que disminuimos gradualmente con el número de partidos disputados, de esta forma el jugador se centrará al comienzo en la exploración pero a medida de que la temperatura descienda variará hacia una mayor explotación. Sin embargo observamos que los valores Q no estaban acotados ni superior ni inferiormente por lo que los cálculos de la exponencial arrojaron errores númericos insalvables. Nos resultó muy dificil explorar este algoritmo.

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

5. Conclusiones

Implementamos un algoritmo basado en Q-learning para entrenar jugadores virtuales del juego 4 en línea. El algoritmo basicamente consistió en explorar con diferentes estrategias, la mayor cantidad de estados posibles del juego y valorizar la acciones que se toman en cada uno de ellos.

Observamos el comportamiento de un jugador que toma decisiones con una dada estrategia en contra de uno que toma acciones aleatoriamente. En particular, empleamos la estrategia *e-greedy*, donde observamos que el jugador entrenado saca una leve ventaja para valores de ϵ pequeños pero no nulos, es decir, en escenarios donde la exploración aleatoria es poco frecuente. Para otras estrategias, nos enfrentamos a problemas númericos que no nos permitieron avanzarcon la investigación.

Por otro lado, exploramos una versión alternativa que llamamos "con visión". En esta versión obtuvimos que los jugadores aprendieran a jugar sin agregar complejidad computacional al algoritmo. Esta modificación permitió acelarar el proceso de aprendizaje.