Inteligencia Artificial Avanzada

Inteligencia artificial en juegos

Aplicaciones de las redes bayesianas en juegos y cálculo del estado siguiente de un agente artificial.

David de León Rodríguez
Alberto González Álvarez
19/04/2018

Índice

Inteligencia Artificial Avanzada

• INTRODUCCIÓN A LA IA EN JUEGOS

La inteligencia artificial se define como la capacidad de un determinado sistema, de poder pensar y razonar de manera autónoma, en el caso de los juegos, se podría definir como la capacidad que tiene el sistema para emular patrones de comportamiento, pensamiento y razonamiento humano, de tal forma que, de cara al jugador, parezca que está jugando con otro humano más y no con un ente totalmente artificial.

A diferencia de los sistemas que normalmente se programan para tomar decisiones más críticas y precisas, los sistemas que se implementan para simular el comportamiento humano en un juego se hacen para que permitan cierto margen de error, es decir; no siempre se toma la mejor decisión para el agente artificial, pues para un jugador no resultaría algo lógico ni divertido que siempre ese agente adivinara siempre lo que está haciendo un jugador humano en un determinado momento. Por ejemplo, si estamos jugando un shooter, no resulta lógico que un bot (agente artificial) dispare siempre a la cabeza con total precisión y matara a todo aquel enemigo, es por ello que se da cierto margen a estos agentes para que no aciertien siempre, pero sin caer en que nunca acierten (pue eso también provocaría que el jugador no disfrutara tanto de la experiencia de juego). Normalmente, los márgenes de acierto o dificultad se pueden ajustar en la mayoría de juegos.

Esto produce que se tomen decisiones no tan óptimas en el juego, e implica que objetivos computacionales, como son el uso de memoria o el tiempo, no se tengan tanto en cuenta en favor de hacer un juego más entretenido y más *jugable*. Antiguamente esto resultaba bastante complicado y tan sólo se dedicaba un 1% o 2% de la potencia total del procesador, para el cálculo de las acciones que ejecutarían los personajes fictios, sin embargo, hoy en día con la llegada de procesadores cada vez más potentes, este porcentaje puede llegar hasta el 35%, pues se hace un cálculo más exhaustivo y más preciso en el que intervienen más variables, resultando por tanto, un videojuego mejor construido y que, sin duda, se acercará mucho más a lo que pasaría en el mundo real.

Alan Turing ya en 1950 adelantaba que un sistema se podrá designar como inteligente si el ser humano con el que interactua es incapaz de reconocer si se está interactuando con un ser humano o con un agente artificial. Esta es la idea que reside a la hora de programar un bot para un videojuego.

Un agente artificial necesita saber cómo actuar en el instante siguiente al que se encuentra, para ello lo que se hace es recoger información del entorno del juego y de cómo se está desarrollando la partida, pues no es lo mismo que un partido de fútbol se vaya perdiendo que ganando, ya que, en el primer caso, se supone que el agente artificial debe actuar de forma más agresiva y atacante para intentar meter un gol, sin embargo, en caso de ir perdiendo debe tomar una decisión más conservadora e intentar evitar que le marquen más goles. Es por ello que para tomar las decisiones

no sólo se tienen en cuenta los diferentes sensores sino, como mencionamos antes, el entorno o evolución del juego.

Evidentemente, todo esto lleva asociado por detrás un cálculo de lo que debe hacer el agente, para luego representarlo. Normalmente esto suele ir separado, es decir, se calculan las probabilidades teniendo en cuenta también la evolución del juego, para luego definir cómo será el movimiento del personaje en el juego y cuál será su objetivo.

También hay que tener en cuenta si estos objetivos se implementan en cada personaje de manera aislada, o en cambio forman parte de una estrategia colectiva que rige todos los personajes del juego. Por un lado, implementar la inteligencia de cada personaje de manera aislada mejora la experiencia añadiendo muchas más variables a la evolución del juego, imitando mejor a la naturaleza menos algorítmica que tiene el destino en la vida real, a diferencia de los objetivos tan concretos de la programación de éstos personajes. Por otro lado, esto complica de manera notable la implementación del videojuego, siendo a veces innecesaria dadas las características de cada juego. Por ejemplo, en un juego de ajedrez no se tiene todo esto en cuenta, mientras que en un juego de simulación social como Los Sims sería más interesante que cada personaje tuviera su propia inteligencia.

Sin embargo, la inteligencia artificial no solo se aplica a los contrincantes con los que interactúa el usuario, existen otras variables dentro de los videojuegos que también requieren de cierta inteligencia para aportar una experiencia de juego cohesiva y creíble, consiguiendo aumentar la calidad de la experiencia de juego y la eficacia a la hora de imitar la realidad. Por ejemplo, imaginemos que en un juego, en un determinado momento se encuentra soleado, no resulta lógico que de un momento a otro cambie el tiempo y llueva, deberá llevarse una progresión hasta que se produzca ese fenómeno, tal y como pasaría en el mundo real.

Ademas, también se puede mejorar la experiencia del usuario al usar un determinado juego almacenando información acerca de su comportamiento para que el juego actúe acorde a los gustos del usuario. Esto se puede lograr utilizando minería de datos extrayendo la información almacenada para que, por ejemplo, se den un tipo de situaciones antes que, o en lugar de otras, para que el usuario disfrute más. Por ejemplo, si el videojuego está dotado de una historia que se va desarrollando a medida que el usuario cubre ciertos objetivos, los personajes del juego involucrados en la historia deberán actuar acorde a lo que ha ido ocurriendo.

- ## APLICACIONES DE LAS REDES BAYESIANAS EN JUEGOS

El artículo propone un modelo basado en redes bayesianas para determinar y aprender los comportamientos de un videojuego sencillo. Se trata de un personaje al que llamaremos bot que puede pasar a seis distintos estados según los valores de una serie de atributos y el estado en el que se encontraba en el momento anterior. El videojuego en sí consiste en que los personajes se muevan por un entorno tridimensional haciendo uso de recursos esparcidos por el universo para combatir otros a personajes.

Entre los objetivos del uso de redes bayesianas se encuentran: que el programa sea fácil de interpretar y también de expresar, adecuarse a los requerimientos computacionales como son el tiempo y el uso de recursos del sistema, la separación del diseño y el desarrollo del videojuego y la capacidad de los bots de tener diferentes comportamientos posibles. Como explicamos en el punto anterior, también se incluye como objetivo la humanidad del bot, así como el aprendizaje de nuevos comportamientos.

Las redes bayesianas utilizan la teoría de probabilidad para funcionar, constando de conocimientos a priori y preguntas para determinar el comportamiento del bot. El conocimiento que tenemos se trata de una serie de variables con las que se define una distribución de probabilidad. Con esta distribución preguntamos a los datos, lo que separará las variables en observables, de interés, de las que se busca respuesta, y residuales, las que no son ni observables ni tienen que ver con la pregunta. La pregunta se responde haciendo uso del teorema de Bayes con la distribución de probabilidad.

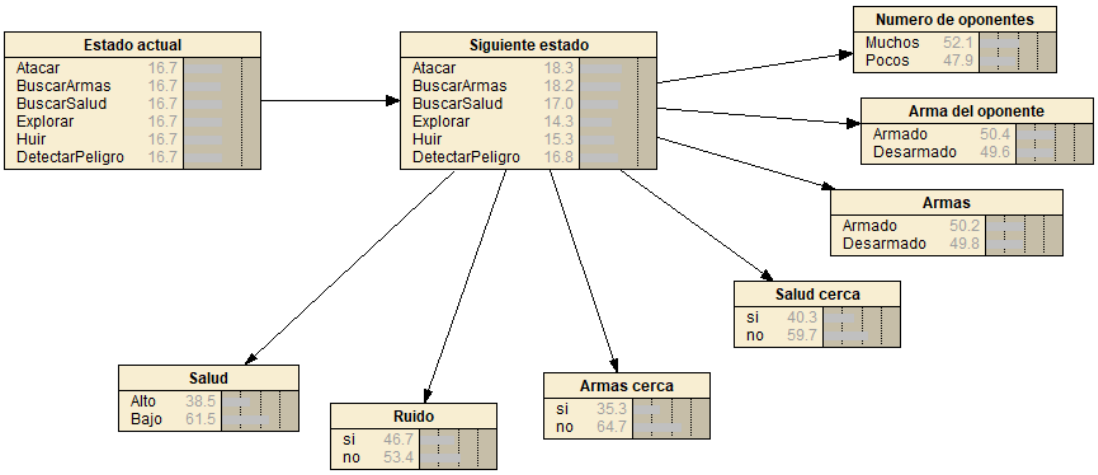
La evolución del bot y de su comportamiento tiene que ver con el instante en el que se encuentre y la información que se reciba del entorno del bot, con esto, se determinará cuál será la siguiente acción que ejecutara el agente ficticio.

En un principio se da libertad para completar las diferentes tablas de probabilidades para pasar de una acción a otra y qué *sensores* influyen en un cambio de comportamiento en el bot, es por ello que nosotros seremos los que decidamos cuál será el comportamiento para el bot.

De cara a implementar esto en un juego, se descartará completamente el uso de sentencias if-else, pues resultará un código *espagueti*, pues no sabremos donde se sitúa cada cosa ni tampoco favorece que se puedan realizar futuros cambios, además de ser una tarea tediosa en el caso de que se tengan un número considerable de sensores (que suele ser lo habitual cuando hablamos de un juego actual).

- # APLICACIÓN PRÁCTICA PARA EL CASO PROPUESTO

Para elaborar nuestro personaje y cómo este actuará según lo que reciba a través de los sensores, hemos diseñado la siguiente red bayesiana, en la que el estado siguiente dependerá del estado en el que se encuentre y de los valores de los sensores (Salud, Ruido, Armas cerca, Salud cerca, Armas, Arma del oponente, Numero de oponentes).



En nuestro caso, hemos diseñado un bot *kamikaze*, es decir, que siempre que pueda atacar ataque, priorizando eso sobre el resto de acciones, sin embargo no es un bot 'tonto' pues en caso de encontrarse desarmado y no tener demasiada salud, priorizará la búsqueda de botiquines para recuperar su salud y, en segundo lugar, la búsqueda de armas. Teniendo esto en cuenta rellenamos las tablas que influirán en el estado que se encontrará el bot en el siguiente instante. Lo que hemos mencionado anteriormente se ve reflejado en la siguiente tabla:

Siguiente estado	Alto	Bajo
Atacar	50	50
BuscarArmas	50	50
BuscarSalud	0	100
Explorar	60	40
Huir	10	90
DetectarPeligro	60	40

Esta es la tabla que mide el estado de salud de nuestro bot y se interpretaría de la siguiente manera:

- Si nuestro bot estaba atacando previamente entonces deducimos que el bot tiene un arma y al ser un bot *kamikaze*, nuestro bot no le importará si tiene la salud alta o baja, seguirá atacando.
- Si nuestro bot, por ejemplo, se encontraba buscando armas, seguirá haciéndolo independientemente del nivel de salud que tenga, pues hemos considerado que la salud no influye en que pare de buscar armas y pase otro estado o en que siga buscando más armas.

- Si nuestro bot se encontraba buscando salud y tiene la salud alta, dejará de buscarla, sin embargo, si tiene la salud baja, seguirá inequívocamente, buscando más salud.
- Si nuestro bot se encontraba explorando y tenía salud alta hemos deducido que es porque no tenía ningún enemigo, así que lo más probable es que siga en el mismo estado. Si por el contrario el bot tenía la salud baja, hemos dado la posibilidad de que cambie de estado e intente buscar salud, pero no lo hará de forma rápida porque tampoco se encuentra amenazado por ningún enemigo.
- Si se encontraba huyendo y la salud es alta, lo más probable es que deje de hacerlo, porque nuestro bot no es un cobarde y hará frente a los enemigos que sean necesarios.
- La última variable la hemos interpretado como "buscar enemigos", es decir, si anteriormente el bot se encontraba buscando enemigos, si tiene la salud alta seguirá buscándolos, aunque si no es así, lo hará con menos probabilidad, aunque sin llegar a ser 0, pues recordemos que nuestro bot es *kamikaze* y no es ningún cobarde.

Para hacer nuestro bot más coherente, y menos arbitrario en cuánto a comportamiento se refiere, haremos que los cambios de un estado a otro sean poco probables. Esto lo conseguimos teniendo en cuenta la tabla del estado siguiente y conociendo la tabla del estado anterior

Estado actual	Atacar	BuscarArmas	BuscarSalud	Explorar	Huir	DetectarPeligro
Atacar	90	4	4	0	2	0
BuscarArmas	5	90	1	1	0	3
BuscarSalud	4	5	90	0	0	1
Explorar	6	1	1	85	0	7
Huir	0	5	5	0	90	0
DetectarPeligro	5	4	1	0	0	90

Para este caso, en la diagonal hemos puesto los mayores valores, pues queremos que si nuestro bot se encuentra atacando, en el siguiente instante se encuentre también atacando. El único estado que hemos considerado que puede variar más es el de Explorar, pues al ser nuestro bot *kamikaze*, hará lo posible para atacar si sabe que puede atacar a algún oponente.

Participación de los integrantes del grupo

- David de León Rodríguez: Red e informe. (50%)
- Alberto González Álvarez: Red, programa e informe. (50%)