Introducción del algoritmo de aprendizaje por refuerzo DQN para juegos de Atari

Angel Larreategui Castro
Universidad Nacional de Ingeniería
Lima, Perú
alarreateguic@uni.pe

Fernando Zambrano Altamirano
Universidad Nacional de Ingeniería
Lima, Perú
fzambranoa@uni.pe

José Reyes Gutiérrez
Universidad Nacional de Ingeniería
Lima, Perú
jreyesg@uni.pe

***Resumen*—**

Índice de Términos— Deep Learning, DQL, Atari, Atari 2600, Deep Mind, Agent57

I. Introducción

II. Objetivo del estudio

Analizar el desempeño de la IA y ver que alcance el mayor rendimiento posible en los juegos (Constantemente superándose).

III. Problema

Ya que los videojuegos son un desafío de varias tareas y de toma rápida de decisiones, se busca que un algoritmo pueda adaptarse a esto con miras a poder adaptarse a otras multitareas con la misma eficiencia.

IV. Marco Teórico

IV-A. Deep Learning

Es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. El aprendizaje profundo es parte de un conjunto más amplio de métodos de aprendizaje automático basados en asimilar representaciones de datos. Una observación (por ejemplo, una imagen) puede ser representada en algunas formas (por ejemplo, un vector de píxeles), pero algunas representaciones hacen más fácil aprender tareas de interés (por ejemplo, "¿es esta imagen una cara humana?") sobre la base de ejemplos, y la investigación en esta área intenta definir qué representaciones son mejores y cómo crear modelos para reconocer estas representaciones.

IV-B. Q-Learning

En el problema completo de aprendizaje por refuerzo, el estado cambia cada vez que ejecutamos una acción. Podemos representar el problema de la siguiente manera. El agente recibe el estado (**state**) en el que se encuentra el entorno (**environment**), el cual representaremos con la letra s (**state**). El agente ejecuta entonces la acción que elija, representada con la letra a (**action**). Al ejecutar esa acción, el entorno responde proporcionando una recompensa, representada con la letra r (**reward**), y el entorno se traslada a un nuevo estado, representado con s' (**next state**). Este ciclo se puede observar en la figura 1.

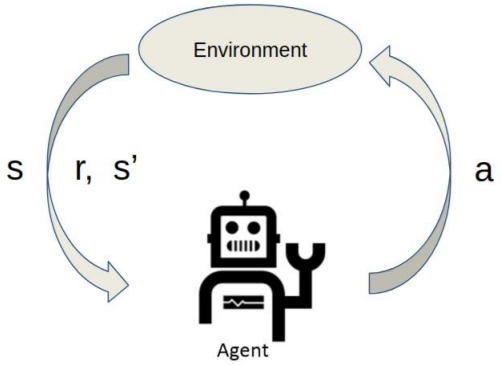

Figura 1. Representación del cambio de estado cada vez que se ejecuta una acción [1]

Por lo tanto, la acción que el agente escoja no debe sólo depender de la recompensa a que vaya a recibir a corto plazo. Debe elegir las acciones que a largo plazo le traerán la máxima recompensa (o retorno) posible en todo el episodio (**episode**). Este ciclo trae una secuencia de estados, acciones y recompensas, desde el primer paso del ciclo hasta el último: $s1, a1, r1; s2, a2, r2; \ldots; sT, aT, rT$. Aquí, T indica el fin del episodio.

IV-B1. Función de valor: Para cuantificar cuanta recompensa obtendrá el agente a largo plazo desde cada estado, introducimos la función de valor $V(s)$. Esta función produce una estimación de la recompensa que obtendrá el agente hasta el final del episodio, empezando desde el estado s. Si conseguimos estimar este valor correctamente, podremos decidir ejecutar la acción que nos lleve al estado con el valor más alto.

IV-B2. Q-Learning, resolviendo el problema: Para resolver el problema del aprendizaje por refuerzo, el agente debe aprender a escoger la mejor acción posible para

cada uno de los estados posibles. Para ello, el algoritmo Q-Learning intenta aprender cuanta recompensa obtendrá a largo plazo para cada pareja de estados y acciones (s, a). A esa función la llamamos la función de acción-valor (action-value function) y este algoritmo la representa como la función $Q(s, a)$, la cual devuelve la recompensa que el agente recibirá al ejecutar la acción a desde el estado s, y asumiendo que seguirá la misma política dictada por la función Q hasta el final del episodio. Por lo tanto, si desde el estado s, tenemos dos acciones disponibles, $a1$ y $a2$, la función Q nos proporcionará los valores-Q (Q-values) de cada una de las acciones. Por ejemplo, si $Q(s, a1) = 1$ y $Q(s, a2) = 4$, el agente sabe que la acción $a2$ es mejor y le traerá mayor recompensa, por lo que será la acción que ejecutará.

IV-B3. Ecuación de Bellman: Primero veamos como es la ecuación de Bellman:

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

Figura 2. la ecuación de Bellman [1]

La explicación para esta ecuación es la siguiente. El valor-Q del estado s y la acción a $(Q(s, a))$ debe ser igual a la recompensa r obtenida al ejecutar esa acción, más el valor-Q de ejecutar la mejor acción posible a' desde el próximo estado s', multiplicado por un factor de descuento γ (discount factor), que es un valor con rango $\gamma \in (0, 1]$. Este valor γ se usa para decidir cuánto peso le queremos dar a las recompensas a corto y a largo plazo, y es un hiperparámetro que debemos decidir nosotros.

IV-C. El algoritmo Deep Q-Network o DQN

Vimos que el algoritmo Q-Learning funciona muy bien cuando el entorno es simple y la función $Q(s, a)$ se puede representar como una tabla o matriz de valores. Pero cuando hay miles de millones de estados diferentes y cientos de acciones distintas, la tabla se vuelve enorme, y no es viable su utilización. Por ello, Mnih et al. [1] inventaron el algoritmo Deep Q-Network o DQN. Este algoritmo combina el algoritmo Q-learning con redes neuronales profundas (Deep Neural Networks). Como es sabido en el campo de la IA, las redes neuronales son una fantástica manera de aproximar funciones no lineales. Por lo tanto, este algoritmo usa una red neuronal para aproximar la función Q, evitando así utilizar una tabla para representar la misma. En realidad, utiliza dos redes neuronales para estabilizar el proceso de aprendizaje. La primera, la red neuronal principal (main Neural Network), representada por los parámetros θ, se utiliza para estimar los valores-Q del estado s y acción a actuales: $Q(s, a; \theta)$. La segunda, la red neuronal objetivo (target Neural Network), parametrizada por $\theta^{'}$, tendrá la misma arquitectura que la red principal pero se usará para aproximar los valores-Q del siguiente estado $s^{'}$ y la siguiente acción $a^{'}$. El aprendizaje ocurre en la red principal y no en la objetivo. La red objetivo se congela (sus parámetros no se cambian) durante varias iteraciones (normalmente alrededor de 10000), y después los parámetros de la red principal se copian a la red objetivo, transmitiendo así el aprendizaje de una a otra, haciendo que las estimaciones calculadas por la red objetivo sean más precisas.

V. Organización del informe (secciones)

En el presente trabajo nosotros vamos a realizar las siguientes secciones:

VI. Estado del Arte

Un gran avance para los proyectos de este tipo sería el modelo de control de políticas basado en deep learning (Mnih et al., 2013) [4] el cual recibe una entrada de pixeles sin procesar y da como salida una estimación de recompensas futuras. Un modelo así es de mucha ayuda para el entrenamiento del algoritmo propuesto.
Otro modelo exitoso que también sirve de referencia es el Simulated Policy Learning (SimPLe) (Kaiser et al., 2020) [5], el cual está basado en modelos de predicción de video y presenta una comparación de muchas arquitecturas de modelo, incluyendo una nueva arquitectura que da los mejores resultados en su configuración.
Por último, cabe mencionar el estudio que usa aprendizaje por refuerzo poco profundo para juegos de Atari (Liang et al., 2016) [6], el cual plantea un conjunto de características computacionalmente prácticas que logran un rendimiento capaz de competir con el de algoritmos DQN en el Arcade Learning Environment (ALE).

VII. Metodología

En esta sección se describen las herramientas y la métodología que se usarán en el presente trabajo

VII-A. Keras

Keras es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano. [7]

Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo. Sus fuertes se centran en ser amigable para el usuario, modular y extensible.

Chollet explica que Keras ha sido concebido para actuar como una interfaz en lugar de ser una framework de machine learning standalone. Ofrece un conjunto de abstracciones más intuitivas y de alto nivel haciendo más sencillo el desarrollo de modelos de aprendizaje profundo independientemente del backend computacional utilizado. [8]

VII-B. TensorFlow

Es una biblioteca de código abierto tanto para Machine Learning como para la computación numérica a gran escala. También, esta biblioteca reúne una serie de modelos y algoritmos de Deep Learning y Machine Learning y los hace útiles mediante una metáfora común. [9]

TensorFlow puede entrenar y ejecutar redes neuronales profundas para la clasificación de dígitos escritos a mano, el reconocimiento de imágenes, la incrustación de palabras, las redes neuronales recurrentes, los modelos secuencia a secuencia para la traducción automática, el procesamiento del lenguaje natural y las simulaciones basadas en ecuaciones diferencias parciales. Lo mejor de todo es que TensorFlow admite predicción de producción a escala, con los mismos modelos utilizados para el entrenamiento.

VIII. Diseño del experimento

IX. Experimentación y resultados

X. Discusión de resultados

XI. Conclusiones

XII. Trabajos Futuros

Referencias

[1] Introduccion al aprendizaje por refuerzo q-learning Disponible en https://markelsanz14.medium.com/introducción-al-aprendizaje-por-refuerzo-parte-2-q-learning-883cd42fb48e.

[2] Introduccion al aprendizaje por refuerzo algoritmo DQN Disponible en https://markelsanz14.medium.com/introducción-al-aprendizaje-por-refuerzo-parte-3-q-learning-con-redes-neuronales-algoritmo-dqn-bfe02b37017f.

[3] Agent57: Outperforming the human Atari benchmark Disponible en https://deepmind.com/blog/article/Agent57-Outperforming-the-human-Atari-benchmark.

[4] Playing Atari with Deep Reinforcement Learning Disponible en https://arxiv.org/pdf/1312.5602.pdf

[5] MODEL BASED REINFORCEMENT LEARNING FOR ATARI Disponible en https://arxiv.org/pdf/1903.00374.pdf

[6] State of the Art Control of Atari Games Using Shallow Reinforcement Learning Disponible en https://arxiv.org/pdf/1512.01563.pdf

[7] About Keras Disponible en https://keras.io/about/

[8] Good news, Tensorflow chooses Keras! Disponible en https://github.com/keras-team/keras/issues/5050

[9] ¿Qué es TensorFlow? ¿Cómo funciona? Disponible en https://aprendeia.com/que-es-tensorflow-como-funciona/