Examen Inteligencia Artificial

Kevin Hernández Rostrán
Tecnológico de Costa Rica

12 de julio de 2020

Parte 1

1. ¿Que son few shot learners? ¿One shot learners?

En problemas de categorización en los modelos de aprendizaje, los "one-shot learners" tienen como objetivo aprender información sobre categorías de objetos de una, o solo unas pocas, muestras/imágenes durante el entrenamiento. En el paper usan el término "meta-learning" para demostrar la estructura del bucle interno / bucle externo del método general, y el término "in context-learning" para referirse al bucle interno del "meta-learning". Por lo que los autores describen a "zero-shot", "one-shot", o "few-shot" dependiendo de cuántas demostraciones se brinden en el momento de la inferencia.

2. Comente el modelo GPT-3 que utilizan, ¿Cómo es?

El modelo de Pre-entrenamiento Generativo (GPT-3) utiliza modelado de lenguaje de probabilidad condicional con una arquitectura de red neuronal transformadora que se basa en mecanismos de auto atención (inspirados en mecanismos de atención de tareas de procesamiento de imágenes) en lugar de recurrencia o convolución. Según los autores este es un modelo de lenguaje autorregresivo de 175 mil millones de parámetros.

3. Comente alguna de las aplicaciones para la que lo utilizaron

- Utilizaron el modelo GPT-3 para evaluarlo en más de dos docenas de conjuntos de datos NLP, así como en varias tareas diseñadas para probar la adaptación rápida a tareas que probablemente no estén contenidas directamente en el conjunto de entrenamiento.
- Compararon el modelo GPT-3 con configuraciones de "one-shot" y "few-shot" con resultados de F1 en conjuntos de datos: CoQA DROP QuACS-

QuADv2 RACE-h RACE-m, LAMBADA y otros con datos limpios y sucios.

- Probaron el modelo en varias medidas GPT-3 Small (con 125M de parámetros), GPT-3 Medium (con 350M de parámetros), GPT-3 Large (con 760M de parámetros), así hasta llegar al "GPT-3" (con 175.0B de parámetros).

Parte 2

La Siguiente Gran Revolución: NLP

El aprendizaje profundo o "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.

A lo largo de 10 años se ha visto un crecimiento sustancial en el campo de la Inteligencia Artificial, aunque desde muy temprano se acuñaban los términos de red neuronal artificial. Todo esto ha desarrollado un revolución tecnológica que el día de hoy nos ha cambiado la vida.

En el Deep learning viéndolo desde una perspectiva general y en función de las tareas que queramos resolver y los tipos de datos que analicemos nos moveremos en diferentes campos como la robótica, visión artificial "computer visión" o el procesamiento del lenguaje natural "Natural Language Processing".

En setiembre de 2012, AlexNet compitió en el Desafío de reconocimiento visual a gran escala ImageNet. La red logró un error de top 5 del 15.3%, más de 10.8 puntos porcentuales menos que el del segundo lugar. Lo que hizo que se valorará el uso de redes neuronales para las tareas basadas en imágenes. En 2014, los modelos generativos obtuvieron un gran impacto para la evolución generativa artificial.

GPT-3 tiene 175 mil millones de parámetros. Un parámetro es un cálculo en una red neuronal que aplica una ponderación mayor o menor a algún aspecto de los datos, para darle mayor o menor importancia a ese aspecto en el cálculo general de los datos. Son estos pesos los que dan forma a los datos y le dan a la red neuronal una perspectiva aprendida sobre los datos, es decir este modelo se usa para predecir la siguiente palabra a partir de palabras anteriores. Lo sorprendente de GPT-3 es que es capaz de generar texto muy realista sobre poesía, diálogo, juegos de palabras, parodias literarias y narración de cuentos.

Es impresionante cómo se publican cada vez más artículos y experimentos sobre IA, como el ChatBot Blender de Facebook, un chatbot que aprende a combinar varias habilidades de conversación, incluidas: la capacidad de asumir una personalidad, discutir casi cualquier tema y mostrar empatía. Como diría el locutor: ¡Es brutal!.

Es impresionante lo que se puede lograr con el procesamiento del lenguaje natural, desde consolas con programación inteligente hasta traductores de código. En Junio de 2020, Facebook AI Research anunció el lanzamiento de

TransCoder, un sistema que utiliza el aprendizaje profundo sin supervisión para convertir el código de un lenguaje de programación a otro. TransCoder recibió un entrenamiento en más de 2.8 millones de proyectos de código abierto y supera los sistemas de traducción de código existentes que utilizan métodos basados en reglas.

Lo que es un poco preocupante son las noticias como las de Microsoft que despidió a sus colaboradores para reemplazarlos por máquinas entrenadas. Lo que parece curioso también es cómo hacen los seres humanos para aprender y saber que con tener un panorama visual poco detallado y algunos inputs puedan concluir con resultados coherentes. Por esto es importante profundizar en estos temas, e aprender más sobre la teoría.

Parte 4

Para las siguientes secciones es importante saber la máquina sobre la que se corrieron los problemas:

```
Model Identifier:       MacBookPro14,1
Processor Name:         Intel Core i5
Processor Speed:        2.3 GHz
Number of Processors:   1
Total Number of Cores:  2
Memory:                 8 GB
```

Cartpole

```
Python      3.8.0
pip         20.1.1
numpy       1.19.0
gym         0.17.2
Keras       2.4.3
matplotlib  3.2.2
tensorflow  2.2.0
```

El problema del péndulo invertido "cartpole" es un péndulo con un centro de gravedad por encima de su punto de pivote. Es inestable, pero se puede controlar moviendo el punto de pivote debajo del centro de masa. El objetivo es mantener el carro equilibrado mediante la aplicación de fuerzas apropiadas a un punto de pivote.

La figura 1 muestra las más de 100 corridas que se realizaron para el experimento, donde se logra evidenciar los puntajes van incrementando.

La figura 2 muestra la interfaz gráfica generada por el Cartpole que se logró correr satisfactoriamente.

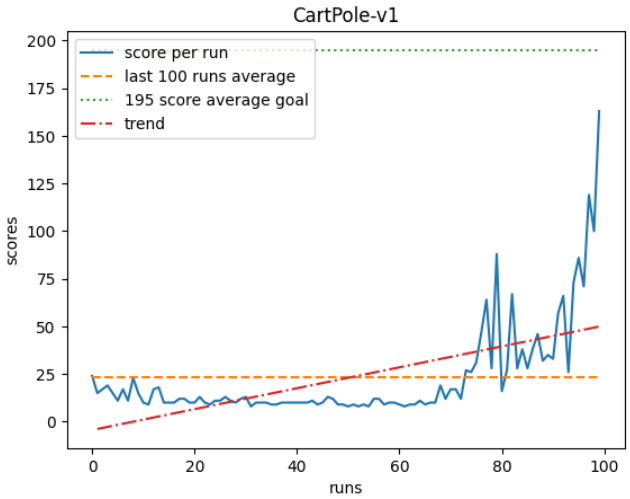

Figura 1: Puntuaciones Cartpole

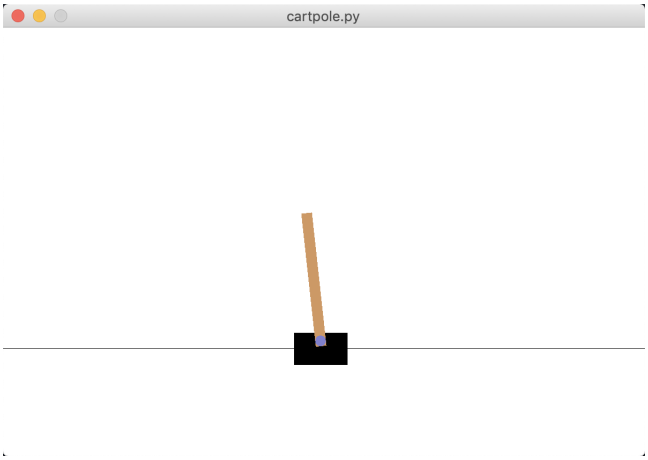

Figura 2: Interfaz Gráfica de Cartpole

Parte 5

Atari

`Python` `2.7.17`

```
pip             19.3.1
numpy           1.16.6
gym             0.16.0
Keras           2.4.3
matplotlib      2.2.5
tensorflow-gpu  1.1.0
opencv-python   4.2.0.32
Pillow          6.2.2
```

El problema tiene como propósito crear un agente que aprenda imitando el cerebro humano y generalice lo suficiente como para jugar múltiples juegos distintos.

Las figuras 3, 4, 5, 6, 7 y 8 muestran los resultados de la implmentación.

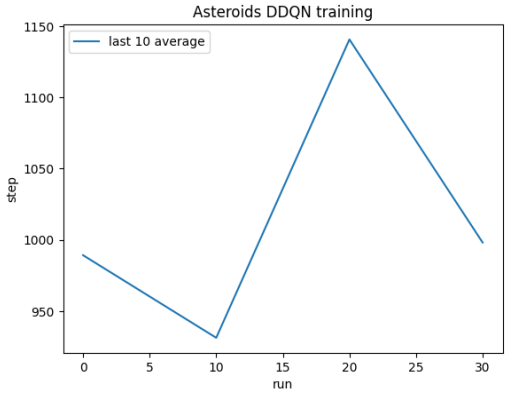

Figura 3: Asteroids DDQN pasos

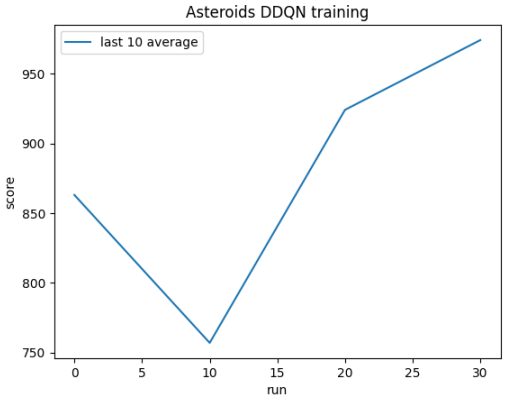

Figura 4: Asteroids DDQN puntuaciones

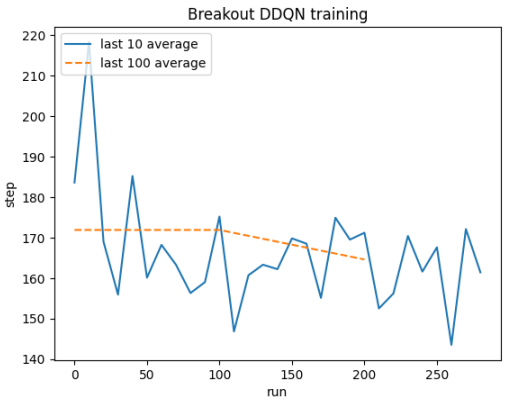

Figura 5: Breakout DDQN pasos

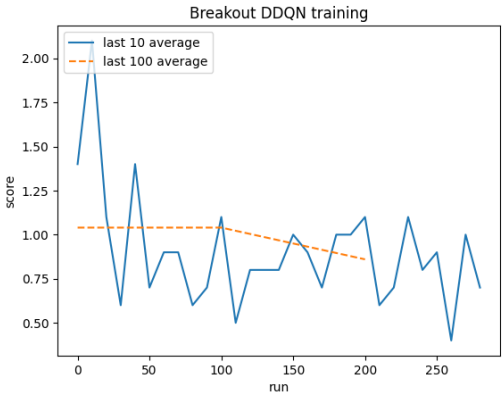

Figura 6: Breakout DDQN puntuaciones

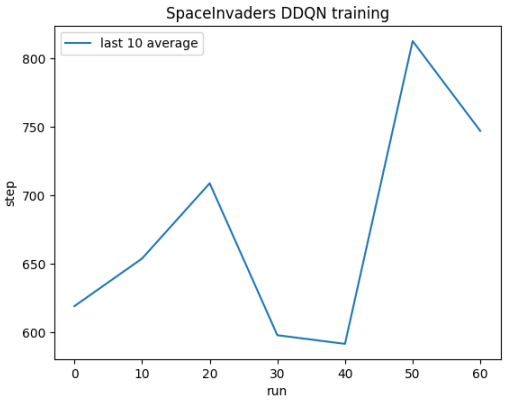

Figura 7: Space Invaders DDQN pasos

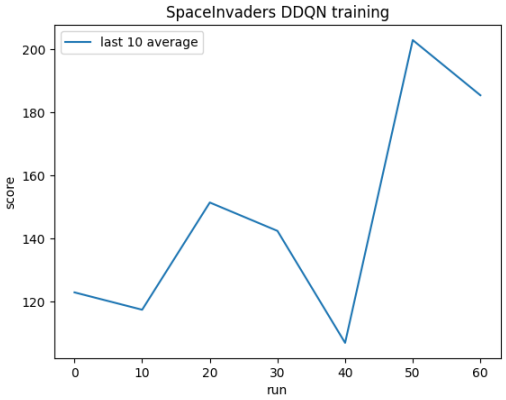

Figura 8: Space Invaders DDQN puntuaciones

Parte 6

¿Como se comparan los metodos de aprendizaje por refuerzos en ambas implementaciones? ¿Cuales son sus diferencias?

Para el Cartpole se calcula la nueva q tomando la q máxima para una acción dada (valor predicho de un mejor estado siguiente), multiplicándola por el factor de descuento (GAMMA) y finalmente agregándola a la recompensa del estado actual. Aunque parezca que a la hora de converger está tratando de predecir su propia salida, es fácil porque, en otras palabras, se está actualizando el valor Q con las recompensas futuras acumuladas con descuento. Pero no se garantiza la convergencia en algoritmos simples de Deep Q-Learning (DQN). Para Cartpole es "fácil" aprender debido al espacio de acción discreto muy limitado.

Sin embargo, la convergencia no siempre es tan "fácil" y en problemas más complejos surge la necesidad de técnicas más avanzadas que estabilicen la capacitación. Estas técnicas son, por ejemplo, Double DQN's.

Acá es donde se encuentra la diferencia entre los dos proyectos, porque en Atari se utiliza el DDQN - Double Deep Q-Learning. Para esto los autores desacoplan las opciones de acción de la generación de valor Q objetivo. Para hacerlo, crean dos redes separadas, de ahí el nombre: Doble Q-Learning. Se utiliza un red principal para seleccionar una acción y una red objetivo para generar un valor Q para esa acción. Para sincronizar las dos redes, se copian los pesos de la red primaria a la objetivo cada n (generalmente alrededor de 10k) pasos de entrenamiento.

Otra mejora importante fue la implementación de la red neuronal convolucional diseñada por Deep Mind en el proyecto de Atari.

Parte 7

¿Como se compara el aprendizaje que ud. logra en la implementacion de juegos de Atari con la que esta en el articulo?

Es poco relevante la verdad, para la implementación del artículo tanto para *Breakout* como *SpaceInvaders* fueron entrenados con 5M de pasos con los mismos hiperparámetros:

```
GAMMA = 0.99
MEMORY_SIZE = 900000
BATCH_SIZE = 32
TRAINING_FREQUENCY = 4
TARGET_NETWORK_UPDATE_FREQUENCY = 40000
MODEL_PERSISTENCE_UPDATE_FREQUENCY = 10000
REPLAY_START_SIZE = 50000
```

```
EXPLORATION_MAX = 1.0
EXPLORATION_MIN = 0.1
EXPLORATION_TEST = 0.02
EXPLORATION_STEPS = 850000
```

Y en general, tardaron aproximadamente 40h en la GPU Tesla K80 o 90h en la CPU Intel i7 Quad-Core de 2.9 GHz.

En cambio mi implementación con los mismos hiperparámetros duró un corto período de tiempo debido al siguiente error:

```
tensorflow.python.framework.errors_impl.UnimplementedError:
The Conv2D op currently only supports the NHWC tensor format
on the CPU.
The op was given the format: NCHW
```

Lo que resultó en mediciones muy pobres como se nota en las figuras 3, 4, 5, 6, 7 y 8.

Parte 8

La implementación se encuentra en la carpeta ***Implementacion cuadricula ventosa***.

1. Técnica utilizada

SARSA

El algoritmo SARSA es una diferencia temporal que examina el valor de una transición entre un estado y el siguiente. El algoritmo fue similar al de la figura 9

2. Tiempo de convergencia en episodios, duracion de episodios.

Initialize, for all $s \in S, a \in A(s)$:
    $Q(s,a) \leftarrow$ arbitrary
    $\pi(s) \leftarrow$ arbitrary
Repeat for each episode:
    Initialise s
    Choose a from s using an ϵ-greedy policy on Q
    Repeat for each state in the episode:
        Take action a, observe r and s'
        Choose a' from s' using an ϵ-greedy policy on Q
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
        $s \leftarrow s', a \leftarrow a'$

Figura 9: Reinforcement Learning, Richard Sutton y Andrew Barto

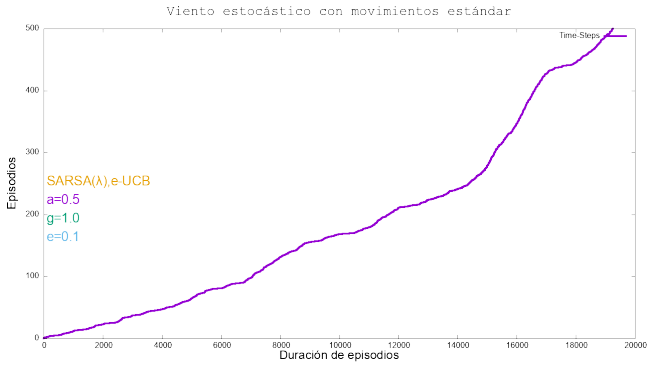

Figura 10: Episodios y su duración en cada episodio

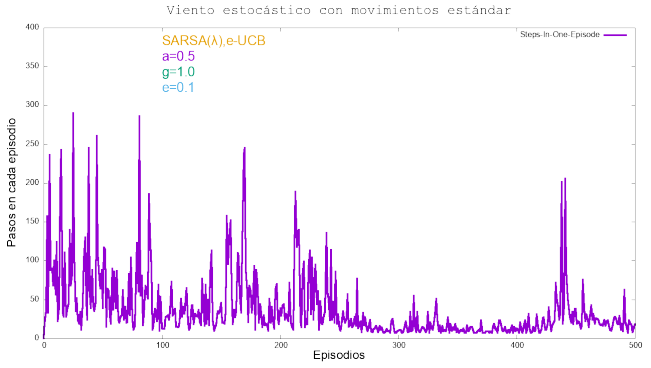

Figura 11: Episodios y los pasos en cada episodio