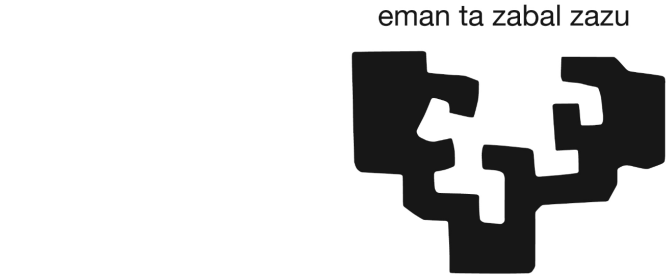
Dueling Network

Informe de trabajo



Técnicas Avanzadas de la Inteligencia Artificial

26/12/2022

David Bernal Gómez
Sara Martín Aramburu

Índice

Explicación

En caso de que el código no sea visible, se puede acceder al collab mediante este link.

En este trabajo se ha implementado la arquitectura “Dueling Network”. Esta nueva arquitectura supone una pequeña mejora al algoritmo original.

Dueling network consiste en separar la red neuronal original en dos sub redes. Una de las subredes devuelve un escalar, el V value para el estado actual V(s). La otra subred devolverá la *advantage function* A(s, a). (No devuelve una función como tal, devuelve el valor de ventaja para cada par estado-acción, en un tensor)

La advantage function devuelve una “valor de importancia” de cada acción, es decir, la ventaja que surge de elegir la acción “a”, en el estado “s”

$$A(s,a) = Q(s,a) - V(s)$$

Basándonos en esta ecuación, podemos definir los Q values como:

$$Q(s,a) = A(s,a) + V(s)$$

Siendo $A(s,\ a)$ y $V(s)$ valores obtenidos por las dos subredes mencionadas.

Sin embargo, esta ecuación tiene un pequeño problema: podemos predecir el mismo valor q teniendo distintos valores en $A(s,\ a)$ y $V(s)$ (Como se puede observar en el ejemplo propuesto en las transparencias)
La solución que se propone es forzar a la *advantage function* a tomar valores de 0. Eso puede lograrse restando el valor máximo o el valor medio de la *advantage function* de durante el cálculo del q value:

$$Q(s,a) = V(s) + \left(A(s,a) - \max_{a'} A(s,a') \right)$$

Recomendados por el profesor, hemos decidido restar el valor medio:

$$Q(s,\ a)\ =\ V(s)\ +\ (\ A(s,\ a)\ -\ mean_{a'}A(s,\ a'))$$

Cambios en el código

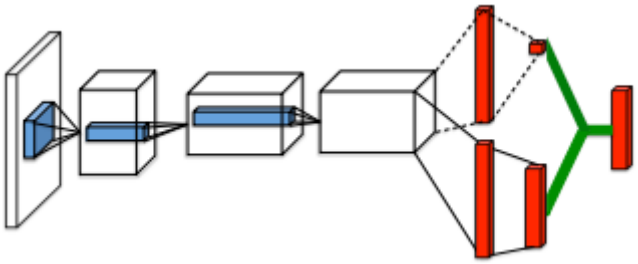
En la clase DQN, en lugar de usar una única red neuronal, usamos dos. En la primera (a), tiene un tamaño de salida `num_actions` (obtendremos un output para cada acción), mientras que en la segunda (v), tiene un tamaño de 1 (obtendremos el v value del estado)

```
#***********Añadido***********
#Usar dos redes, una para obtener v() y otra para a()
self.a = nn.Sequential(
    nn.Linear(num_inputs, 128),
    nn.ReLU(),
    nn.Linear(128, 128),
    nn.ReLU(),
    nn.Linear(128, num_actions)
)

self.v = nn.Sequential(
    nn.Linear(num_inputs, 128),
    nn.ReLU(),
    nn.Linear(128, 128),
    nn.ReLU(),
    nn.Linear(128, 1)
)
#*********************************
```

En la función *forward*, calculamos el output a partir de un input x recibido. Primeramente, el input pasa por las dos subredes, de las que obtendremos V(s) (output_v) y A(s, a) (output_a) Devolveremos el resultado aplicando la fórmula mencionada en el apartado anterior.

```
def forward(self, x):
    #return self.layers(x)

    #******Añadido**********
    output_a = self.a(x)
    output_v = self.v(x)

    return output_v + output_a - output_a.mean()
    #***************************
```

Nota: leyendo por encima el paper propuesto, Antes de separar las redes en dos, se preprocesan los datos por una red convolucional. Y el resultado se pasa a las dos subredes, como puede verse en esta imagen:



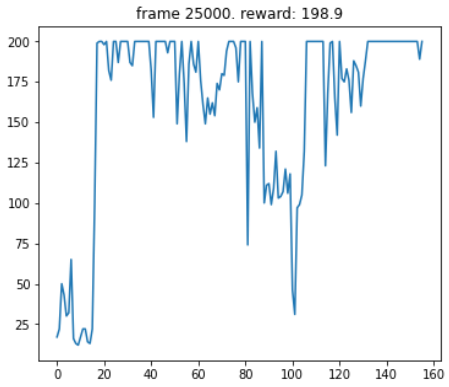
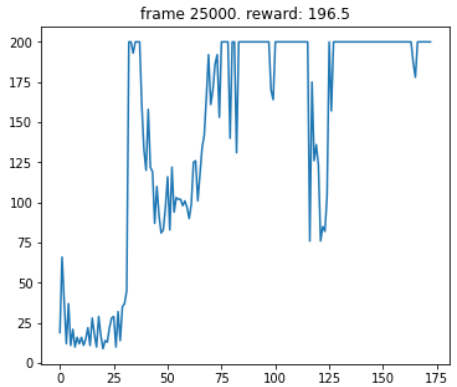
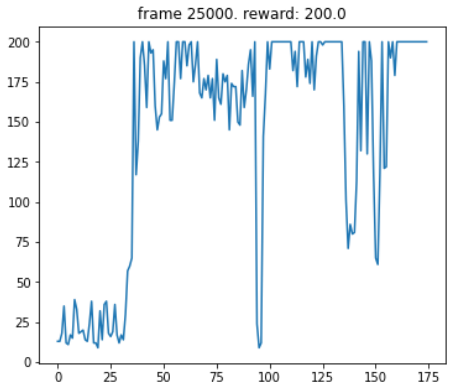
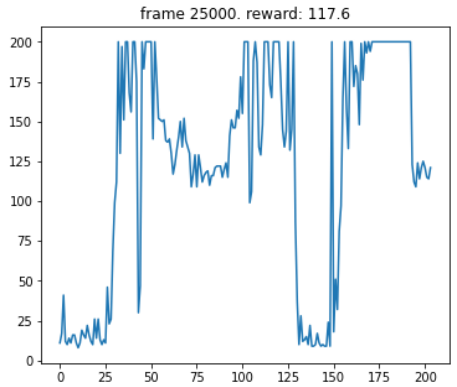
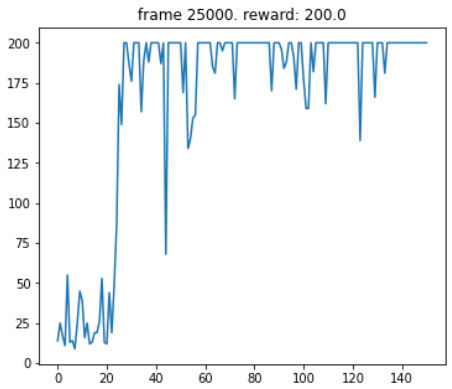
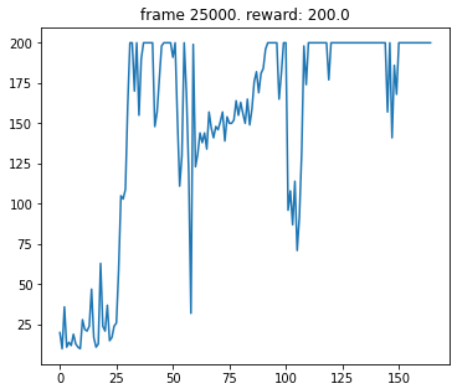
Nosotros no tenemos ninguna capa previa a las dos subredes. Lo hemos implementado tal y como se propone en la imagen de las transparencias, pero no sabemos si con una o varias capas anteriores para pre procesar los inputs se obtendrían mejores resultados

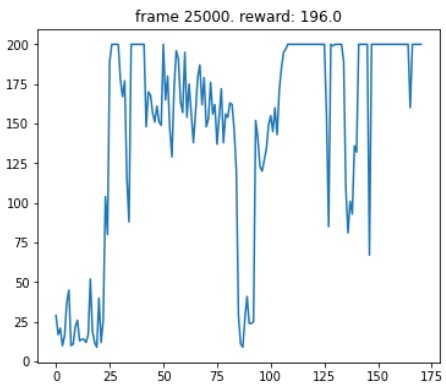
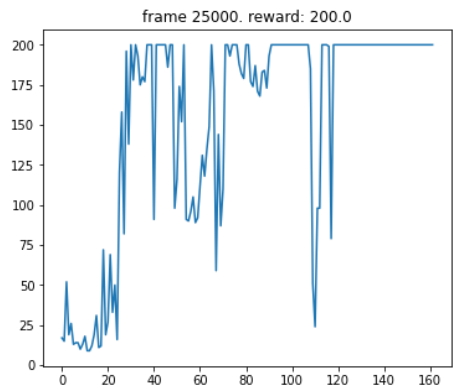
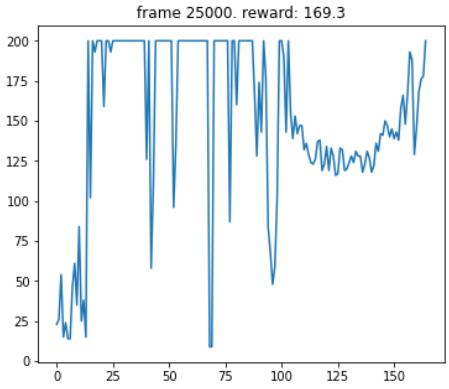
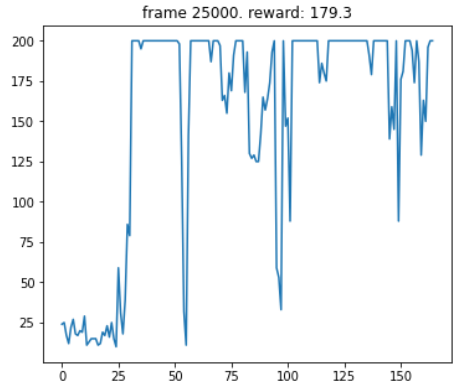
Otra opción (en el forward):

```
#definir en _init_ self.capaPreprocesado
x = self.capaPreprocesado(x)

output_v = self.v(x)
output_a = self.a(x)

return = output_v + (output_a - output_a.mean())
```

Resultados

La forma en la que vamos a evaluar y comparar es la siguiente:
Entrenaremos 5 veces cada modelo para posteriormente comparar las gráficas resultantes. Las gráficas muestran la recompensa obtenida (eje y) a lo largo de 25.000 iteraciones (eje x). Solo se representan los valores de recompensa cada 200 iteraciones, (por ejemplo, cuando x=20, nos encontramos en la iteración 20*200 = 4000)

Entrenamiento	Dueling network	DQN
1		
2		
3		

4		
5		

Con estos resultados podemos comparar:

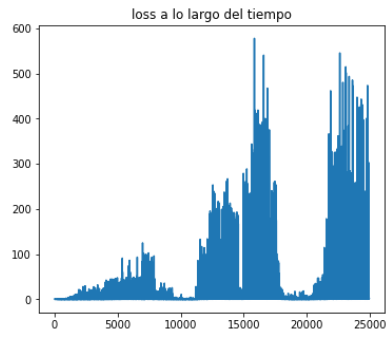
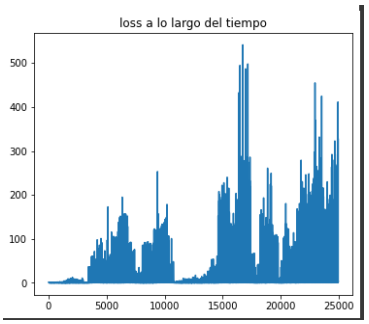
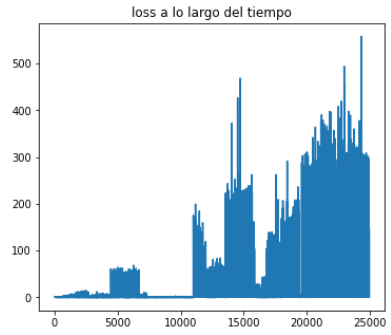
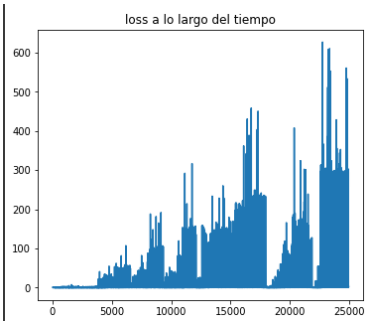
- Las veces que finaliza con la recompensa máxima:
  - Dueling network: 2/5 pruebas
  - DQN: 2/5 pruebas
- El momento en el que la recompensa se dispara:
  - Dueling network: entre las iteraciones 0 y 5000 (25*200)
  - DQN: entre las iteraciones 0 y 6000 (30*200)
- Cuánto tiempo se mantienen los modelos con una recompensa máxima:
  - Ambos modelos sufren altibajos en las iteraciones intermedias.
- Recompensa media a lo largo de los entrenamientos:
  - Dueling network: media de 152 puntos.
  - DQN: media de 140 puntos, ligeramente inferior.

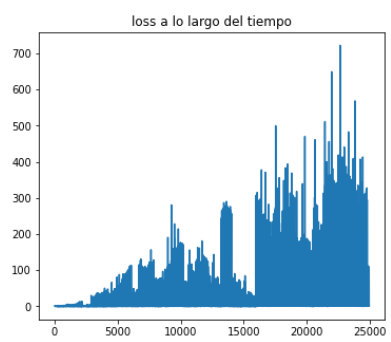
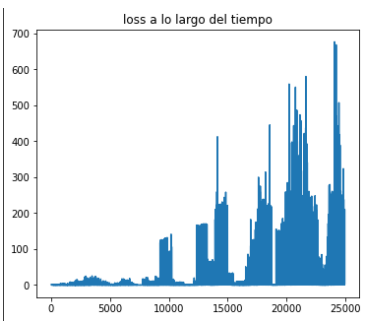
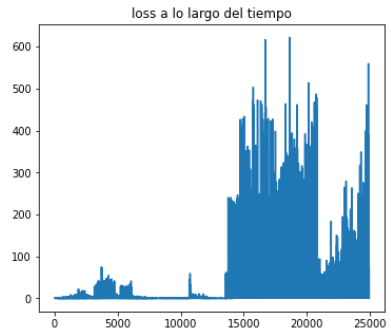
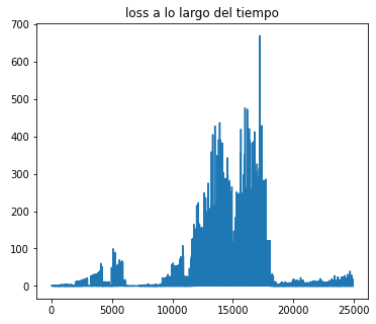
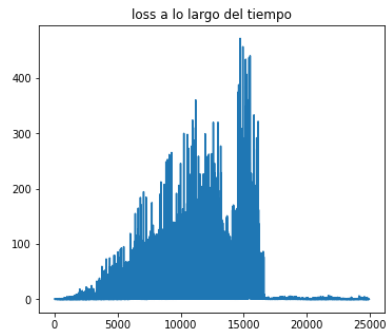
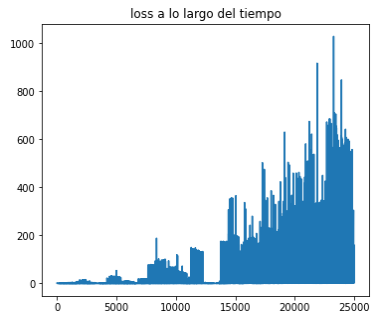
Además, comparamos también las recompensas medias en cada prueba, así como la recompensa media de cada algoritmo:

Entrenamiento	Dueling network	DQN
1	160	144
2	142	135
3	164	142
4	145	134
5	150	146
Media total	152	140

Podemos ver que la media total del algoritmo dueling network es 12 puntos superior a la versión DQN, lo cual indica cierta mejoría en el resultado. Aún así, 5 entrenamientos no son suficientes para afirmarlo con seguridad.

Por otro lado, obtenemos las siguientes gráficas de loss:

Entrenamiento	Dueling network	DQN
1		
2		

3		
4		
5		

Como podemos ver, en muchas ocasiones el loss se mantiene mucho más tiempo en valores bajos en el Dueling Network. Pero el valor del loss no nos sirve de mucho para determinar qué algoritmo es mejor, porque no todos los episodios son iguales en duración.