PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
Facultad de Ingeniería
IIC2613 Inteligencia Artificial
2022-1

Tarea 4

Aspectos generales

Formato y plazo de entrega

El formato de entrega son *jupyter notebooks* (.ipynb)[1] en los directorios de cada pregunta. El lugar de entrega es en el repositorio de la tarea, en la *branch* por defecto, hasta el **jueves 30 de junio a las 23:59 hrs.** Para crear tu repositorio, debes entrar en el enlace del anuncio de la tarea en Canvas.

Replicabilidad de resultados

Muchas funciones de sklearn tienen componentes aleatorias, lo que hace que en cada ejecución puedas obtener resultados distintos. Para evitar esto, usaremos una *seed* de `numpy` igual a tu número de alumno[2]. Si tu número de alumno fuera `121212` la celda del *notebook* debería quedar de la siguiente manera:

```
# Establece una semilla para resultados replicables
import numpy as np
np.random.seed(121212)
```

Integridad Académica

Este curso se adhiere al Código de Honor establecido por la universidad, el cual tienes el deber de conocer como estudiante. Todo el trabajo hecho en esta tarea debe ser **totalmente individual**. La idea es que te des el tiempo de aprender estos conceptos fundamentales, tanto para el curso, como para tu formación profesional. Las dudas se deben hacer exclusivamente al cuerpo docente a través de las *issues* en GitHub.

Por otra parte, sabemos que estás utilizando material hecho por otras personas, por lo que es importante reconocerlo de la forma apropiada. Todo lo que obtengas de internet debes citarlo de forma correcta (ya sea en APA, ICONTEC o IEEE). Cualquier falta a la ética y/o a la integridad académica será sancionada con la reprobación del curso y los antecedentes serán entregados a la Dirección de Pregrado.

Comentarios adicionales

El objetivo de esta tarea es que comprendan los conceptos de Redes Neuronales y aprendizaje reforzado, aplicándolos en actividades prácticas. Es fundamental que pongan énfasis en las explicaciones y justificaciones de sus respuestas, cuidando la redacción, ortografía; manteniendo el código ordenado y comentado. Respuestas que solo presenten resultados o código no serán consideradas, mientras que tareas desordenadas pueden ser objeto de descuentos.

NOTA: Debes presentar un README explicando los detalles de tu entrega y todas las librerías que hayas usado. Adicionalmente, **todas las celdas del notebook deben estar ejecutadas**

[1] El nombre del archivo es irrelevante.
[2] Si tu número de alumno termina en J, reemplázala por un 0

1. Redes Neuronales (Total 3 pts.)

El *dataset* GTZAN consta de 10 carpetas, donde cada una de ellas está asociada a un género musical diferente que puede ser: Jazz, Classical, Metal, Disco, Reggae, Hip Hop, Pop, Rock, Country y Blues. Dentro de cada carpeta, hay 100 audios .WAV de 30 segundos cada uno, con música asociada al genero correspondiente.

Para facilitar el trabajo, se te entrega el archivo `mfcc_data.json`, el cual corresponde a un JSON con todos los audios del *dataset* original ya pre-procesados (con sus MFCC's). Para crear este archivo, cada audio fue dividido en 10 segmentos de 3 segundos cada uno y, a cada uno de estos segmentos, se le calculó el MFCC, tal como se muestra en la siguiente figura ilustrativa:

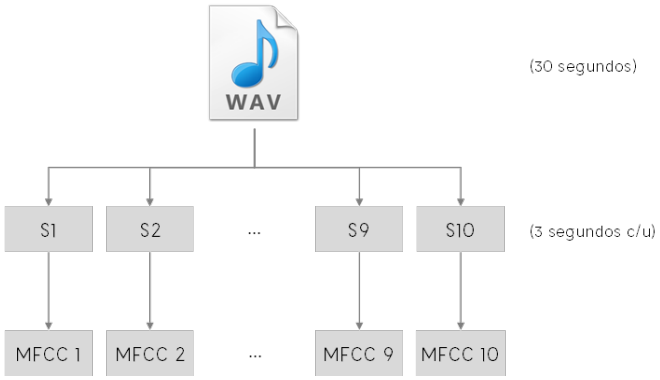

Figura 1: Diagrama de obtención de MFCC's para un audio

De este modo, se obtuvieron $100 \cdot 10 = 1000$ MFCC's para cada genero musical, llegando a $1000 \cdot 10 = 10000$ datos entre todo el *dataset*. Para esta parte, deberás programar redes neuronales densas y convolucionales que sean capaces de clasificar el género al que pertenece un audio usando sus MFCC's.

IMPORTANTE: No debes subir el JSON al repositorio o se podrían aplicar descuentos.

Actividad 1: Comprendiendo los datos (0.4 pts.)

En cápsulas se vio las dificultades de representar de forma numérica datos más complejos, como texto, imágenes o, en este caso, audio. También se vio sobre los MFCC's como una forma de representar audios de forma bastante completa. Para esta actividad deberás escoger una canción cualquiera del *dataset* y, utilizando la librería `librosa`, obtener su MFCC y graficarlo. Además, deberás investigar y responder:

- ¿Qué son y para qué sirven los MFCC's? **Nota:** Se espera una respuesta breve y general. Basta con una noción a grandes rasgos de lo que son este tipo de datos.
- Si tuviéramos pocos audios para entrenar un modelo, ¿qué técnicas podrías usar para enriquecer el set de datos? **Nota:** Investiga sobre el aumento de datos (puede ser más fácil ver ejemplos con imágenes y luego extrapolarlo).

Actividad 2: Optimizador Adam (0.4 pts.)

Investiga sobre el optimizador Adam. Explica cómo funciona y qué lo diferencia de SGD (*Stochastic Gradient Descent*) ¿Qué beneficios tiene y por qué podría ser de utilidad en este contexto?

Actividad 3: Red Neuronal Densa (0.6 pts.)

Para esta actividad, deberás construir una red neuronal densa multi-capa, como las vistas en clases/ayudantías. Esta debe ser capaz de recibir el set de datos con los MFFC's y entregar un *output* de 10 dimensiones, donde cada entrada puede tomar un valor entre 0 y 1, con la probabilidad que tenga de pertenecer a la clase representada por su posición. Por ejemplo, en la siguiente figura se recibe un MFCC y se obtiene la probabilidad de pertenecer a cada categoría (las etiquetas y valores son solo de referencia):

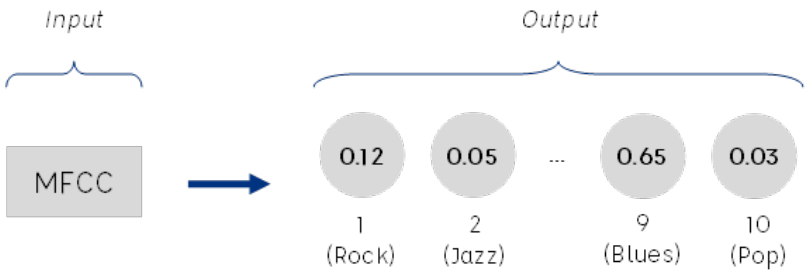

Figura 2: Diagrama de la red neuronal

La estructura de tu red, esto es, la cantidad de capas densas, la cantidad de neuronas por capa y las funciones de activación quedan a elección libre. Sin embargo, el optimizador debe ser Adam. También puedes utilizar distintas técnicas que consideres útiles para llegar a un mejor resultado, pero recuerda explicarlas. **Nota:** Como se trata de una red de clasificación multi-clase, debería usarse una función de activación específica en la capa de salida.

Se espera que además de entrenar tu modelo, hagas lo siguiente:

- Grafica cómo varía el *accuracy* y la función de pérdida (*loss function*) con las épocas[3] e interpreta lo que ves (ej. ¿hay *overfitting*?)
- Evalúa el *accuracy* en el set de test y comenta los resultados brevemente

Para que tu respuesta sea considerada válida, deberás obtener **al menos un 40 % de *accuracy*** en el set de entrenamiento y de test.

Actividad 4: Introducción a las CNN's (0.3 pts.)

Investiga qué es una Red Neuronal Convolucional o CNN (*Convolutional Neural Network*) y responde en palabras sencillas[4] las siguientes preguntas:

- ¿En qué consisten, qué hacen y cuál es su utilidad al trabajar con audio?
- ¿Cuáles son sus ventajas respecto a redes convencionales para el procesamiento de audio?

[3] Antes de graficar, puedes hacerte una idea utilizando la extensión de `TensorBoard` de GoogleColab vista en ayudantías.
[4] No se espera excesiva rigurosidad, pero sí que se entiendan las ideas más importantes.

Actividad 5: Creando una CNN clasificadora (0.7 pts.)

Construye una red neuronal convolucional que clasifique las canciones por género musical como en la actividad 3. Al igual que antes, la estructura de la red y las técnicas que uses quedan a tu criterio, pero debes usar el optimizador Adam.

Se espera que hagas lo siguiente:

- Grafica cómo varía el *accuracy* y la función de pérdida (*loss function*) con las épocas e interpreta lo que ves (ej. ¿hay *overfitting*?)
- Evalúa el *accuracy* en el set de test y comenta los resultados brevemente

Para que tu respuesta sea considerada válida, deberás obtener **al menos un 50 % de *accuracy*** en el set de entrenamiento y de test.

Actividad 6: Comparación de modelos (0.4 pts.)

Compara el rendimiento de ambas redes neuronales y grafica la matriz de confusión en el set de test para cada una de ellas. Además, responde brevemente:

- ¿Cuáles son las clases que más se confunden en cada una de las redes neuronales? ¿Por qué crees que ocurre?
- ¿Son las mismas clases las que más se confunden en ambas redes?

Actividad 7: Probando con audios propios (0.2 pts.)

La función `get_mfccs` que está en el código base del *notebook* recibe un *string* con la ruta a un archivo de audio en formato WAV que tenga al menos 30 segundos de duración y la etiqueta a la que corresponde (eso dependerá de cada implementación). Al ejecutarla, carga el archivo y trunca su duración a los primeros 30 segundos, luego lo divide en 10 segmentos de 3 segundos cada uno, calcula el MFCC a cada segmento y retorna un diccionario con los 10 MFCC's calculados y la etiqueta ingresada.

Para esta actividad deberás escoger una canción que te guste y que claramente pertenezca a una de las 10 categorías del *dataset*, luego utiliza la función `get_mfccs` para obtener sus MFCC's. Posteriormente, entrega los MFCC's calculados a a tu red neuronal convolucional entrenada para que clasifique el género musical. Evalúa si el resultado coincide con el esperado.

IMPORTANTE

La canción que escojas debe ser descargada desde YouTube o algún otro medio y no debes subirla al repositorio, pero debes adjuntar un enlace público que permita acceder al recurso.

2. Aprendizaje Reforzado (Total 3 pts.)

En esta parte ocuparás aprendizaje reforzado (*Reinforcement Learning*) para entrenar una IA que sea capaz de aprender a jugar al clásico juego Pong por sí misma:

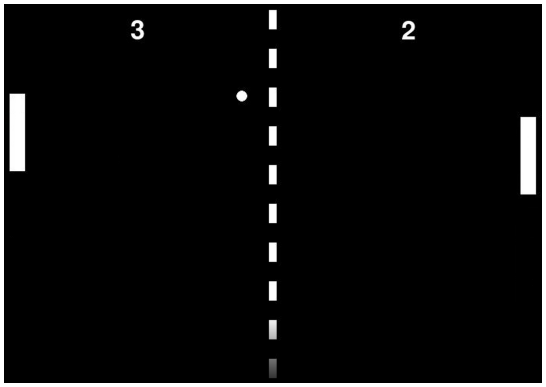

Figura 3: GUI del juego Pong

En tu repositorio se te da el código del juego Pong, pero el pádel (paleta móvil que controla el jugador) se mueve de manera aleatoria. Deberás implementar el algoritmo *Q-Learning* para que pueda moverse de forma "inteligente", respondiendo la mayor cantidad de pelotas posibles.

La interfaz usada para entrenar al agente es diferente a la que se utiliza para jugar el juego. Para simplificar el proceso de entrenamiento, el pádel no juega contra otro pádel, sino que juega solo, recibiendo pelotas lanzadas en posiciones aleatorias desde el fondo de la cancha.

Los archivos contenidos en el código base son:

- `PongAI.py`: Código con la interfaz gráfica y funcionamiento general del juego. **No modificar**.
- `QAgent.py`: Este archivo contiene el modelo de un agente que se mueve de forma aleatoria. Es aquí donde deberás trabajar y aplicar tus conocimientos de aprendizaje reforzado.

Como ya se mencionó anteriormente, el archivo sobre el cual debes trabajar e implementar *Q-Learning* es `QAgent.py`[5]. Este archivo contiene la clase `Agent` con los siguientes métodos:

- `__init__()`: Este método inicializa los atributos del agente. En el caso del agente aleatorio, solo se inicializan en 0 las partidas jugadas por el agente.
- `get_state(game)`: Este método consulta al juego por el estado actual del agente y lo retorna como un vector o tupla de 5 dimensiones, donde cada una de las entradas representa la siguiente información (en orden):
  1. **Velocidad en el eje Y:** Toma un valor entero entre -5 y 5 correspondiente a la velocidad actual de la pelota en el eje Y. Los valores negativos indican un movimiento hacia arriba de la cancha (se redondea al entero más cercano).
  2. **Proximidad al pádel:** Toma un valor entero entre 0 y 5 que representa la distancia entre el pádel y el cuadrante en el que se encuentra la pelota (son 6 cuadrantes), tal como se muestra en la figura:

[5]Este archivo se encuentra muy bien comentado, por lo que se recomienda leerlo como parte del enunciado.

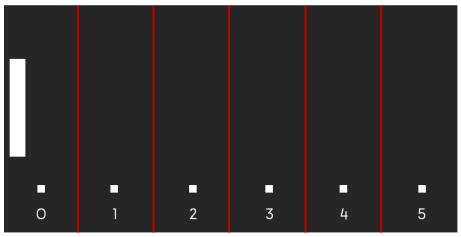

Figura 4: Proximidad del pádel y cuadrante de la pelota

3. **Altura del balón respecto al extremo superior del pádel:** Es un entero que puede tomar los valores 0, 1, 2, 3 y 4. Si la pelota se encuentra arriba del extremo superior del agente, entonces toma el valor 0 si está a menos de un pádel de distancia de dicho extremo, o 1 si está más lejos. Por el contrario, si la pelota está debajo del extremo superior del agente, toma el valor 2 si está a menos de un pádel de distancia y el valor 3 si está más lejos.
4. **Altura del balón respecto al extremo inferior del pádel:** Es un entero que puede tomar los valores 0, 1, 2, 3 y 4. Si la pelota se encuentra arriba del extremo inferior del agente, entonces toma el valor 0 si está a menos de un pádel de distancia de dicho extremo, o 1 si está más lejos. Por el contrario, si la pelota está debajo del extremo inferior del agente, toma el valor 2 si está a menos de un pádel de distancia y el valor 3 si está más lejos.

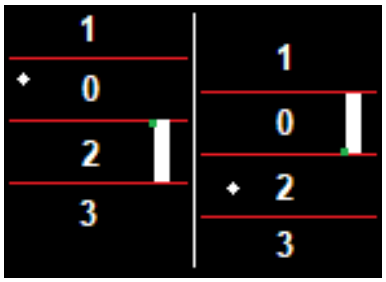

Figura 5: Referencia respecto al extremo superior (izq.). Referencia respecto al extremo inferior (der.)

5. **Número de rebotes:** Es un entero correspondiente al número de rebotes que ha dado la pelota desde el golpe del rival. Por físicas del juego, se encuentra entre 0 y 2.

- `get_action(state)`: Este método recibe el estado del agente y retorna un entero representando la acción que debe tomar el agente. Las acciones posibles son:
  - `0`: Moverse hacia arriba.
  - `1`: No moverse.
  - `2`: Moverse hacia abajo.

  Nota: En el caso del agente aleatorio, este no utilizará la información que entrega el estado y retornará una acción al azar.

Fuera de esta clase, sólo existe la función `train()`. Esta función es la encargada de entrenar al agente y es la que se llama al correr el archivo de ejecución principal.

Actividad 1: Comprendiendo los hiperparámetros (0.4 pts.)

En esta actividad tendrás que pensar el problema antes de comenzar a programarlo, para ello, responde lo siguiente:

- ¿Cuál es la función del *exploration rate*?
- ¿Cuál es el problema de tener un *exploration rate* mínimo con un valor muy bajo y un *exploration decay rate* con un valor muy alto?
- ¿Qué pasa si el *exploration decay rate* es demasiado bajo?

En qué tipo de situaciones o cosas se podrían reflejar los cambios en estos hiperparámetros, contextualizado en este problema (juego Pong).

Actividad 2: Implementando Q-Learning (1.8 pts.)

Basándote en lo visto en clases y ayudantías, implementa el algoritmo *Q-Learning*[6] para el agente del juego Pong, esto es, el pádel. Para esto deberás realizar los siguientes pasos[7] (recuerda comentar todo lo que hagas en tu código):

1. Inicializar la *Q-Table* del agente en el método `__init__` con sus dimensiones correctas **(0.3 pts.)**.
2. Modificar el método `get_action` para que el agente sea capaz de explorar el estado actual o explotar la mejor acción conocida hasta el momento **(0.6 pts.)**.
3. Actualizar la *Q-Table* una vez que se ejecute la acción seleccionada por el agente **(0.5 pts.)**.
4. En caso de terminar una partida, actualizar el *exploration rate* del agente **(0.4 pts.)**.

IMPORTANTE Y OBLIGATORIO

Para facilitar la corrección de tu tarea, te pediremos subir la *Q-Table* de tu agente ya entrenado como un archivo en formato `q_table.csv` sin *headers* (filas/columnas con títulos o nombres).

El archivo .csv debe poseer 8 columnas y 3168 filas. Las primeras 5 columnas corresponden a los posibles valores de las 5 dimensiones de los estados en el orden que se mencionan en este enunciado. Por otra parte, las siguientes 3 columnas corresponden al *Q-Value* de las acciones posibles del pádel también en el orden del enunciado. De esta forma, cada fila representa un posible estado y los valores de las acciones posibles para ese estado. Una fila de este archivo se podría ver así:

$$[-4,\ \ 1,\ \ 1,\ \ 0,\ \ 1,\ \ 1.9878,\ \ 0.243,\ \ 99.346]$$

Este formato también es el mismo requerido para poder participar en el bonus, así que estás invitado/a/e a participar. **Nota:** Puedes encontrar un ejemplo del archivo aquí.

Actividad 3: Análisis de parámetros del agente (0.4 pts.)

Una vez hayas programado tu modelo y este sea capaz de aprender, juega con los hiper-parámetros y responde las siguientes preguntas:

- ¿Qué rol cumple la tasa de descuento en *Q-Learning*?
- ¿Qué tasa de descuento te dio mejores resultados? ¿Por qué crees que fue así?
- ¿Para qué sirve el *learning rate*? ¿Qué valor te entregó mejores resultados? Comenta los resultados.

[6] Si tienes dudas con *Q-Learning*, puedes revisar este artículo.
[7] Se recomienda revisar el código base, específicamente los `#IMPLEMENTAR` que hay en el código

Actividad 4: Nueva política de recompensas (0.4 pts.)

Actualmente, el juego otorga recompensas al agente calculando la posición en donde terminará la pelota; asignando 1 punto si se acerca hacia esta posición (o se encuentra en ella) o restando 1 punto si se está alejando de esta.

Para esta actividad deberás diseñar (no es necesario implementar) una política para recompensa distinta a la implementada en el archivo `Pong.py` para poder entrenar al agente y responde:

- ¿Por qué crees que sería una buena forma de recompensar al agente? Argumenta.

Bonus: DCCampeonato IIC2613 2022-1

Para finalizar el semestre de forma especial, celebraremos la primera edición del DCCampeonato IIC2613, en donde puedan poner a prueba sus agentes de aprendizaje reforzado y competir con sus demás compañeros de curso en un torneo de pádel virtual. Quienes ganen este evento **podrán ganar hasta 10 décimas** en su tarea.

Para participar, deberán inscribirse en un formulario que se anunciará en conjunto con todas las reglas del campeonato a través de Canvas. Contamos con su motivación y participación para hacer de esta primera edición una instancia pedagógica y recreativa que pueda mantenerse a futuro.

Referencias

[1] Olteanu A. (2020). *GTZAN Dataset - Music Genre Classification* [Fichero de datos]. Recuperado el 25 de mayo del 2022 en: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[2] Tec With Tim. (2022, 15 de febrero). *Make Pong With Python!* [Video]. YouTube. Recuperado el 18 de mayo del 2022 en: https://www.youtube.com/watch?v=vVGTZlnnX3U

Material Recomendado

[1] Ringa Tech. (2022, 25 de mayo). *Funciones de activación a detalle (Redes neuronales)* [Video]. YouTube. Recuperado el 28 de mayo del 2022 en: https://www.youtube.com/watch?v=$_0wdproot34$