PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
Facultad de Ingeniería
IIC2613 Inteligencia Artificial
2022-2

Tarea 4

Aspectos generales

Formato y plazo de entrega

El formato de entrega son: un archivo .ipynb [1] para la parte de redes neuronales y completar el código pedido en la parte de aprendizaje reforzado. Es importante que a la hora de entregar el notebook este se encuentre **con las celdas ejecutadas**. Para la parte de redes neuronales, las respuestas a las preguntas teóricas deben incluirse en celdas de texto adyacentes al código que utilices, de manera tal que el texto y las celdas de código sigan una estructura lógica que evidencie tu trabajo. El lugar de entrega es en el repositorio de la tarea, en la *branch* por defecto, hasta el **jueves 1 de diciembre** Para crear tu repositorio, debes entrar en el enlace del anuncio de la tarea en Canvas. Por último, recuerda que los cupones de atraso son días **no hábiles** extra.

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]:

```
# 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.

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

1. DCClasificador de sonidos (Total 3 pts.)

El archivo *UrbanSound8k.csv*, contiene el nombre de 8732 audios .WAV, los cuáles pueden ser de 10 categorías distintas. Estos audios se encuentran repartidos en 10 carpetas, donde en cada una de ellas se pueden encontrar audios de las distintas clases. Estas clases son las siguientes: Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren y Street Music. En la base estas clases se encuentran en orden, es decir que un 0 representa a Air Conditioner y así para las demás clases.

Para facilitar el trabajo, en el código base del notebook se te entrega una celda que contiene todo lo necesario para cargar los audios (cada línea de la celda tiene un comentario explicando lo que hace, por si se quiere saber a detalle cómo se están cargando los datos). Lo único que debes hacer es correr esta celda y obtendrás tres arrays:

- **X**: Contiene las medias por fila de los MFCC´s de los 8732 audios, quedando de dimensión 8732 x 40. Listo para hacer una red neuronal densa pues ya se encuentra "aplanado".
- $\boldsymbol{X_{cnn}}$: Contiene los MFCC´s sin "aplanar" y redimensionados, para quedar de dimensiones 8732 x 173 x 40. Este array lo debes usar para la parte de CNN.
- **y**: Es el vector que contiene las etiquetas de cada clase de sonido en la base de datos. Deberás hacer unos cambios a este vector para que sea numérico y lo pueda recibir la red neuronal.

Nota: Es normal que la celda mencionada se demore 5 minutos o más en correr, y el entrenamiento de las redes también toma su tiempo, por lo que recomendamos fuertemente que ocupen notebooks de *google colab* para realizar la tarea.

Importante: Recuerda que debes separar las bases en entrenamiento, testeo y validación. Los porcentajes que elijas para cada set de datos quedan a tu criterio.

Actividad 1: Comprendiendo los datos (0.2 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 un audio cualquiera del dataset, usar alguna función para que se pueda reproducir en el archivo .ipynb y, utilizando la librería librosa, obtener su MFCC y graficarlo (esto se hace de manera independiente a la carga de datos comentada anteriormente). Además, deberás investigar y responder lo siguiente:

- ¿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: Estructura de las redes neuronales (0.5 pts.)

- Explica, en términos generales, en qué consiste una red neuronal densa.

- Explica qué es una función de activación, y comenta sobre qué propiedades debe cumplir dicha función para poder ser utilizada en una red neuronal. Menciona un ejemplo en particular de función de activación, y fundamenta por qué podría ser utilizada para construir una red neuronal.
- Investiga sobre el problema de vanishing gradient en el entrenamiento de redes neuronales, y cómo la función de activación ReLU proporciona ventajas sobre otras funciones de activación ante este problema.
- Investiga en qué consiste una capa softmax, especificando qué hace y cuál su utilidad en los problemas de clasificación entre varias categorías.
- 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 (0.8 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):

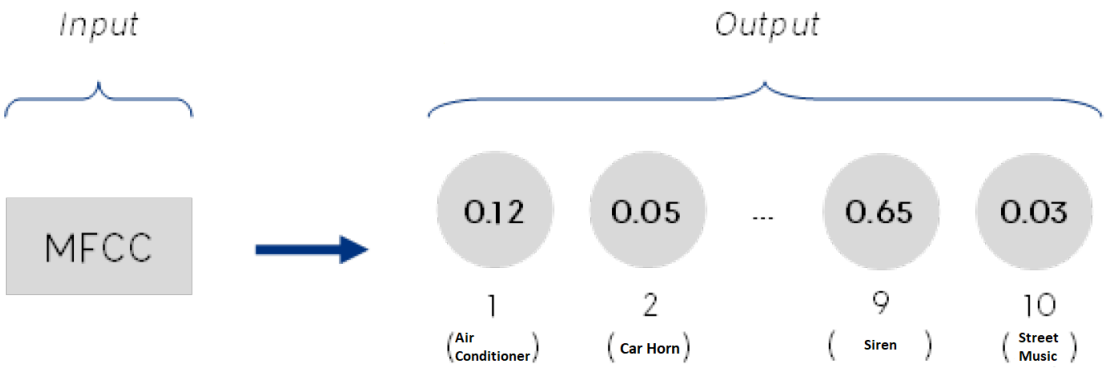

Figura 1: Diagrama de la red neuronal

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], para las bases de entrenamiento y validación, e interpreta lo que ves (ej. ¿hay overfitting?)
- Evalúa el accuracy en el set de test y comenta los resultados brevemente.

[3]Antes de graficar, puedes hacerte una idea utilizando la extensión de TensorBoard de GoogleColab vista en ayudantías.

Para que tu respuesta sea considerada válida, deberás obtener **al menos un 65 % de *accuracy*** en el set de entrenamiento y de testeo, y para obtener todo el puntaje deberás tener **sobre un 85 % de *accuracy*** en estos mismos sets.

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 las siguientes preguntas:

- Lee el **siguiente artículo** y explica brevemente en qué consisten las redes neuronales convolucionales (CNN). En tu respuesta, explica la función de los siguientes componenetes de una CNN:
  - Convolutional layer
  - Pooling layer
  - Fully connected layer
- ¿Cuáles son sus ventajas respecto a redes convencionales para el procesamiento de audio?

Actividad 5: Creando una CNN clasificadora (0.8 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, para las bases de entrenamiento y validación, 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 70 % de *accuracy*** en el set de entrenamiento y de testeo, y para obtener todo el puntaje deberás tener **sobre un 85 % de *accuracy*** en estos mismos sets.

Actividad 6: Comparación de modelos (0.2 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?
- ¿Cuáles clases son las que peor predicen los modelos? ¿A qué podría deberse esto?

Actividad 7: Probando la CNN (0.2 pts.)

Ahora que tenemos nuestra red neuronal convolucional lista, la probaremos con algunos audios para notar su desempeño. En la celda respectiva a esta pregunta se les entrega una función llamada ***transform***, la cual recibe el path a un audio y devuelve un valor representando el audio de la misma manera que se entrenó la red convolucional. Para esta actividad deberán usar su CNN para predecir la clase de 5 audios que ustedes decidan y deberán:

- Mostrar el nombre de la clase predicha para este audio y mostrar el audio en su notebook. [4]
- Si la clase predicha no coincide con el sonido del audio explicar a qué cree que podría deberse. ¿Hay ruido de fondo? ¿El sonido se parece a otra clase? etc.

[4]Para dejar el audio reproducible en su notebook pueden utilizar la función **Audio** de **IPython.display**.

2. DCCanario (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 Flappy Bird por sí misma:

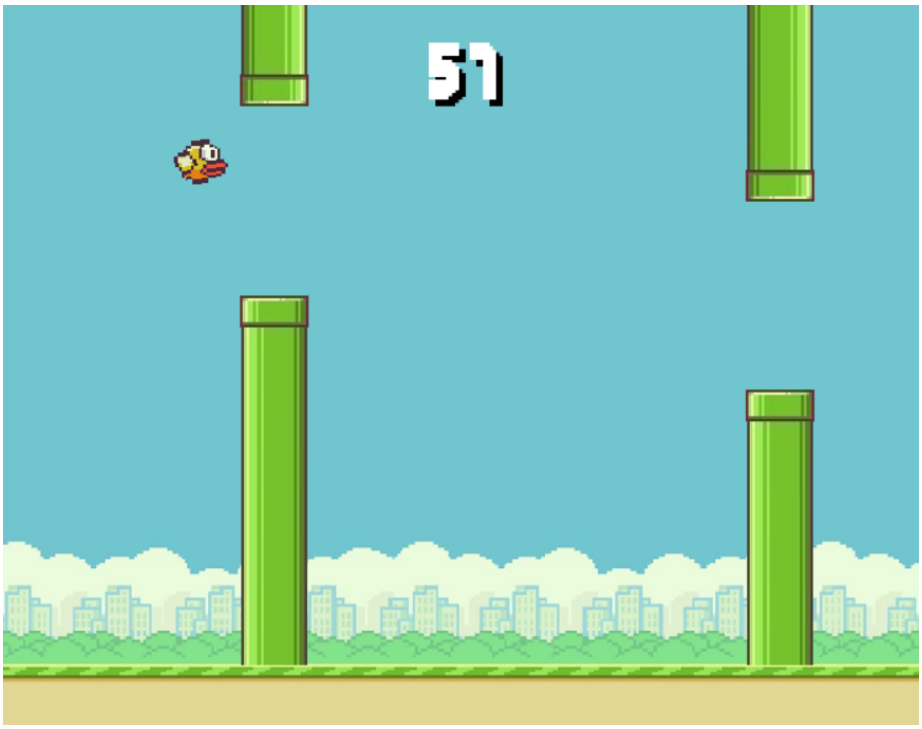

Figura 2: GUI del juego Flappy Bird

En tu repositorio se te da el código del juego Flappy Bird, pero el personaje (un pájaro que salta a lo largo el eje Y) decide si saltar o no de manera aleatoria. Deberás implementar el algoritmo *Q-Learning* para que pueda moverse de forma "inteligente", tratando de lograr el mayor puntaje posible (atravesando el mayor número de tuberías consecutivamente).

Los archivos contenidos en el código base son:

- `FlappyBirdAI.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):

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

1. **Distancia en el eje X:** Toma un valor entero entre 0 y 50 correspondiente a la distancia entre el jugador y la siguiente tubería en el eje X (corresponde a una subdivisión del espacio posible en 51 partes iguales).

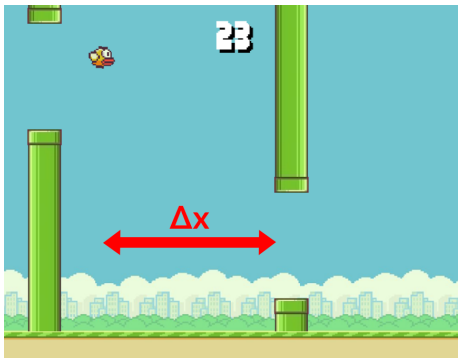

Figura 3: Proximidad entre el jugador y la siguiente tubería

2. **Distancia en el eje Y:** Toma un valor entero entre 0 y 27 que representa la distancia entre el jugador y el centro del agujero de la siguiente tubería en el eje Y, tal como se muestra en la figura:

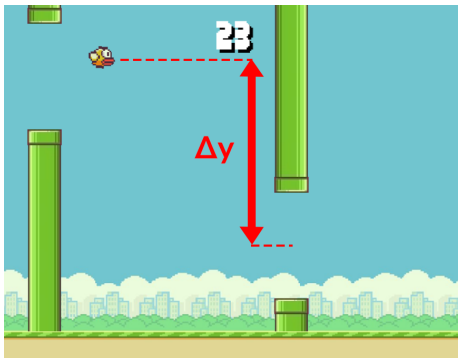

Figura 4: Proximidad del jugador y el agujero siguiente

3. **Sentido de la próxima tubería:** Es un entero correspondiente a la dirección en que se encontrará el próximo agujero (0 si se encuentra encima del jugador y 1 si se encuentra debajo de él).
4. **Distancia en el eje Y de la subsiguiente tubería:** Toma un valor entero entre 0 y 27 que representa la distancia entre el jugador y el centro del agujero de la tubería que vendrá después de la siguiente, tal como se muestra en la figura:

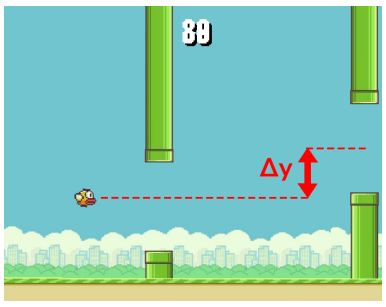

Figura 5: Proximidad entre el jugador y el agujero subsiguiente

5. **Sentido de la próxima tubería subsiguiente:** Es un entero correspondiente a la dirección en que se encontrará el agujero subsiguiente (0 si se encuentra encima del jugador y 1 si se encuentra debajo de él).

- `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`: No moverse.
  - `1`: Saltar.

  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 Flappy Bird).

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 Flappy Bird, esto es, el pájaro. 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.)**.

[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

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 7 columnas y 1599936 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 2 columnas corresponden al *Q-Value* de las acciones posibles del agente también en el orden del enunciado (primero, no moverse y segundo, saltar). 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:

$$[41,\ 17,\ 1,\ 9,\ 0,\ 1.9878,\ 99.346]$$

Este formato también es el mismo requerido para poder participar en el bonus, así que estás invitado/a/e a participar. Si todavía tienes dudas al respecto, puedes generar una issue en este enlace.

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.

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

Actualmente, el juego otorga recompensas al agente si se acerca hacia el agujero de la tubería, sumando 1 punto, o bien, restando 1 punto si se está alejando de esta. Además, castiga al agente con -1 punto si pierde la partida y lo premia con 50 cada vez que pasa a traves de una tubería.

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

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

Bonus: DCCampeonato IIC2613 2022-2

Para finalizar el semestre de forma especial, celebraremos la segunda edición del DCCampeonato IIC2613 en aprendizaje reforzado, en donde puedan poner a prueba sus agentes de aprendizaje reforzado y competir con sus demás compañeros de curso en un torneo de Flappy Bird. Quienes participen en este torneo recibirán **1 décima extra** en su tarea, mientras que quienes ganen este evento **podrán ganar hasta 2 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 instancia pedagógica y recreativa una tradición para el futuro del curso.

Comentarios

Como podrás notar, buena parte de esta tarea involucra respuestas de desarrollo escrito, donde debes transparentar tu razonamiento y explicar las decisiones que tomas. Por este motivo, para que una respuesta se considere correcta, debes tener cuidado de fundamentar apropiadamente lo que digas, aludiendo a referencias confiables y a la documentación de las librerías que utilices. Por ejemplo, si te pedimos explicar un hiperparámetro en particular, o calcular una métrica de desempeño, se espera que demuestres un dominio general de lo que hace dicho hiperparámetro, o de la información que entrega la métrica solicitada. Para esto, es esperable que busques recursos (libros, artículos, documentación oficial, etc.) para fundamentar tus respuestas, donde debes indicar claramente la fuente de tal recurso.

Al mismo tiempo, es posible que al intentar ejecutar operaciones costosas (como entrenar un modelo sobre un conjunto de datos), la ejecución sea más lenta de lo que esperas. Esto es un escenario cotidiano al trabajar con modelos de aprendizaje de máquina, de modo que se espera que seas capaz de lidiar con tales situaciones, y que transparentes y fundamentes las decisiones que tomes en el proceso.