Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

T.U.I.A

Aprendizaje Automático II

Trabajo Práctico 2

Redes neuronales recurrentes y Q-learning

1er cuatrimestre 2025

Autor/es:

Grupo Nº	
Nombre y Apellido	Nº de Legajo

Corrigió	Calificación

Problema 1 - Audio MNIST

Descripción:

En este problema, se presenta un conjunto de datos que contiene clips de audio correspondientes a dígitos hablados del 0 al 9.

Dataset:

spoken_digit | TensorFlow Datasets

El dataset proporcionado incluye un total de 2500 clips de audio correspondientes a 5 locutores distintos, 50 clips por dígito por locutor.

Objetivo:

Utilizando el dataset proporcionado, el objetivo es construir un modelo de clasificación utilizando redes neuronales que pueda inferir con precisión el dígito correspondiente dado un clip de audio. Se deben entrenar y evaluar modelos utilizando técnicas adecuadas de validación y métricas de evaluación de clasificación.

Se solicita entrenar dos modelos de distintas arquitecturas y comparar los resultados:

- Modelo convolucional sobre los espectrogramas de los clips.
- Modelo recurrente sobre los espectrogramas de los clips.

Ver notebook como ejemplo de obtención de espectrogramas a partir de clips de audio.

Entrega:

La entrega debe incluir:

Código fuente de la solución implementada en Google Colab, que incluya:

- Análisis previo y preprocesamiento del set de datos.
- Definición y entrenamiento del modelo.
- Resultados de la evaluación de los modelos, incluyendo métricas de desempeño y visualizaciones relevantes.

Nota: el código debe estar debidamente documentado con comentarios explicativos para que el trabajo sea fácilmente comprensible para otros revisores.

Problema 2 - Flappy Bird

Descripción:

Entrenar agentes que puedan jugar Flappy Bird usando diferentes enfoques con PyGame Learning Environment.

Objetivo:

El objetivo de este ejercicio es entrenar agentes para resolver videojuegos sencillos usando Q-Learning y la librería PLE. En primer lugar, usar Q-learning para entrenar a un agente para jugar Flappy Bird. Luego, entrenar a otro agente usando Deep Q-learning y la Q-table del agente provisto.

Entrega:

Entrega: La entrega debe incluir el código fuente de la solución, completando los archivos y scripts provistos en el template del proyecto.

1. Agente Q-Learning (Corresponde al Ejercicio A del README.md):
    a. Implementación del Agente.
    b. Entrenamiento del Agente.
    c. Prueba del Agente Entrenado
2. Agente Basado en Red Neuronal (Corresponde al Ejercicio B del README.md):
    a. Entrenamiento de la Red Neuronal.
    b. Implementación del Agente Neuronal.
    c. Prueba del Agente Neuronal

Deben poder utilizarse los distintos tipos de agentes usando el parámetro `--agent` al ejecutar `test_agent.py`.

Dentro del repositorio se debe incluir un archivo **conclusiones.md** (**usando Markdown**) con:

- Descripción de la ingeniería de características sobre el estado del juego (discretización).
- Análisis y comparación de los resultados obtenidos para los diferentes agentes.