Universidad del Valle de Guatemala
Departamento de Ingeniería Mecatrónica
MT3006 - Robótica 2
MSc. Miguel Zea

Laboratorio 3

Redes neuronales en TensorFlow

Objetivos

- Aprender a utilizar Keras para describir redes de aprendizaje automático basadas en TensorFlow.
- Emplear TensorFlow y Keras para entrenar una red neuronal que permita la identificación de un sistema dinámico no lineal y caótico.

Procedimiento

Tome nota que para esta práctica deberá redactar y subir un reporte a Canvas respondiendo a las preguntas que se planterán durante el desarrollo del laboratorio.

En la práctica de esta semana usted tendrá como tarea emplear la combinación de TensorFlow y Keras dentro de Python para resolver dos problemas de machine/deep learning (con redes de poca profundidad, tal que aún puedan entrenarse sólo con CPU) con base en data previamente seleccionada y/o generada. Para lograr esto, realice lo siguiente:

1. Preste atención a la introducción básica a TensorFlow y Keras que brindará el catedrático del laboratorio. Adicionalmente, revise los recursos proporcionados para tener una idea general de cómo emplear las herramientas de software que utilizará para resolver los problemas que se planterán más adelante.

2. Descargue de Canvas el archivo `mt3006lab3.zip` y extraiga sus contenidos a una carpeta de su selección. Dentro de la carpeta usted encontrará no sólo la data que deberá emplear para los problemas sino que también un par de scripts en Python los cuales puede emplear como base para sus soluciones.

3. Emplee como base el script `lab3p1.py` para resolver el problema de **clasificación de imágenes de perros y gatos**.

    a) Se tiene una colección de 160 imágenes en escala de grises, de 32x32 cada una, que describen perros y gatos. El código para cargar las mismas, en forma de vectores de 1024 dimensiones, ya se encuentra dentro del script provisto en Python y se separan en un set de entrenamiento (60 imágenes) y de validación (20 imágenes). Dentro del código también se encuentra la generación automática de las etiquetas de las imágenes: 0 para un perro y 1 para un gato, y la visualización de un ejemplo de cada categoría.

En el contexto de aprendizaje supervisado, se le llama set de entrenamiento a la data que se emplea para obtener los parámetros del modelo mediante la solución al problema de optimización. Por otro lado, el set de validación corresponde a una colección de data desconocida para el modelo, la cual se emplea para obtener el rendimiento del modelo entrenado y ajustar los *hiper-parámetros* del entrenamiento para evitar el *overfitting* y obtener una exactitud adecuada (típicamente dada como un porcentaje de éxito de la clasificación). En algunos casos, se emplea adicionalmente un set de prueba para evitar posibles sesgos introducidos durante el ajuste de parámetros en la validación. Un split típico de entrenamiento-validación-prueba es un 70-20-10 por ciento del total de la data.

b) Entrene un perceptrón con función de activación sigmoide para resolver el problema de clasificación. Emplee una función de pérdida de entropía cruzada binaria (*binary cross-entrophy*), gradient descent estocástico (corresponde al optimizer *SGD*) para resolver el problema de optimización, con una tasa de aprendizaje de 0.01 y un parámetro de momentum de 0.9 (puede encontrar más información sobre este tipo de algoritmo de descenso por gradiente en el siguiente enlace), indique que se recolectará una métrica de exactitud. Adicionalmente, utilice 200 *epochs* (corridas completas al dataset de entrenamiento) para el entrenamiento.

c) Corra el script completo de Python y preste atención a los gráficos generados y a la información impresa en la consola. En los gráficos verá la evolución de la función de pérdida y la exactitud del modelo, tanto durante el entrenamiento como durante la validación. En la consola observará el entrenamiento del modelo, epoch por epoch, la exactitud final del entrenamiento y la validación, y la *matriz de confusión* (esta se emplea para la evaluación de modelos de clasificación y brinda el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos, encontrados al aplicar el modelo en un dataset de interés) generada para el dataset de validación. **NOTA:** puede emplear `model.predict(x)` para evaluar el modelo en una observación en particular `x`.

d) Dentro de su reporte, incluya la figura que muestra la evolución de la pérdida y la exactitud del modelo en función del número de epochs. Adicionalmente, responda a las siguientes preguntas: ¿Existe una discrepancia entre la exactitud del modelo para la data de entrenamiento y la de validación? ¿Según la evolución de la pérdida, cree que el modelo obtenido le hace un overfit a la data de entrenamiento? ¿Qué le dice la matriz de confusión con respecto al rendimiento del modelo en el set de validación? ¿Qué ocurre si cambia la función de activación del modelo a una lineal?

e) Añada al código, exactamente antes de la visualización de los ejemplos de cada una de las categorías, un re-arreglo aleatorio de las columnas de los arrays `dogWave` y `catWave` (puede emplear un random generator de numpy junto con la función shuffle del mismo para hacer esto). Esto ayudará a cambiar los sets de entrenamiento y validación para cada corrida del script (una forma más sofisticada de hacer esto se conoce como validación cruzada, *cross-validation*, en caso que desee investigar un poco más al respecto). Adicionalmente, ajuste la tasa de aprendizaje, el parámetro de momentum y el número de epochs hasta lograr encontrar un modelo con poco overfit. ¿Cuál es el valor de los hiper-parámetros que mejor le funcionaron? Incluya de nuevo las mismas figuras que para el inciso anterior y la respuesta a las preguntas.

f) Emplee el método `model.get_weights()` para obtener los pesos encontrados por el modelo. Descarte el término de bias, re-arregle los 1024 pesos restantes en un array de 32x32 y grafíquelo como una imagen. ¿Qué puede decir sobre el modelo con base en esta imagen? ¿Qué factores son determinantes para el modelo al momento de efectuar

la clasificación entre gatos y perros? Incluya la respuesta a estas preguntas dentro de su reporte.

4. Emplee como base el script `lab3p2.py` para resolver el problema de **identificación no lineal del sistema de Lorenz**.

   a) Se quiere emplear una red neuronal densa multi-capa para identificar la dinámica del sistema Lorenz

   $$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}\sigma(y-x)\\ x(\rho-z)-y\\ xy-\beta z\end{bmatrix}}_{\mathbf{f}(\mathbf{x})},$$

   cuando sus parámetros se seleccionan para que el sistema presente un atractor extraño (caos). Dentro del script provisto en Python, se generan y grafican las trayectorias de este sistema para 100 condiciones iniciales aleatorias. La data se prepara tal que pueda emplearse tanto como la entrada y salida de una red neuronal. Recordemos que, la simulación de un sistema dinámico incluye de manera implícita una discretización de la dinámica de tal forma que se obtiene un mapa iterado

   $$\mathbf{x}_{k+1} = \mathbf{F}\left(\mathbf{x}_k\right)$$

   a partir del sistema continuo. La idea es emplear los x_k's generados como la entrada a una red neuronal y los x_{k+1}'s como salidas, para obtener un modelo que pueda predecir el siguiente estado si se le coloca en la entrada el estado actual, en esencia, identificando la dinámica discreta en el tiempo que representa al sistema continuo bajo cierto esquema de integración numérica.

   b) Proponga, defina y entrene una red neuronal densa multi-capa que permita la identificación del sistema Lorenz. Se recomienda que inicie con una arquitectura de regresión de una capa oculta y que vaya añadiendo capas según los resultados del aprendizaje. Adicionalmente, se recomienda diversidad en las funciones de activación para que el modelo pueda capturar correctamente las no linealidades de la dinámica. Usted tiene la tarea de establecer hiper-parámetros y métricas de rendimiento adecuadas para la aplicación.

   c) A partir de una condición inicial aleatoria, genere la trayectoria del sistema de Lorenz empleando integración numérica (puede emplear el módulo de `integrate` de `scipy` o hacer su propia rutina basada en RK4) y también la red neuronal entrenada. Grafique ambas soluciones en la misma figura (puede emplear esto junto con las métricas de rendimiento del inciso anterior para validar el modelo).

   d) Con base en los resultados obtenidos en los incisos anteriores, ¿Cuál fue la arquitectura final que empleó para resolver el modelo (*especifique*)? ¿Qué tan exacto fue el modelo entrenado? ¿Qué tan cara, computacionalmente hablando, es la evaluación de la red neuronal entrenada en comparación a la evaluación del campo vectorial del sistema Lorenz? ¿Cómo puede establecer que el modelo no está presentando overfitting? Incluya la respuesta a estas preguntas en su reporte, junto con las figuras que considere pertinentes para respaldar su discusión.

5. Suba a Canvas el reporte con las figuras y respuestas a las preguntas indicadas durante el desarrollo de la práctica.

Recuerde que entregas tardías representan una penalización del 25 % por semana.