Instituto Tecnologico de Costa Rica (ITCR)
Sede Interuniversitaria de Alajuela
Escuela de Computación
Curso: Inteligencia Artificial
Profesora: Maria Auxiliadora Mora

Entrega: en el TecDigital de un archivo .zip con un cuaderno jupyter en Pytorch, **debidamente documentado**, con una función definida por ejercicio.

Modo de trabajo: Grupos de una o dos personas.

Introduccion

En este trabajo programado se aplicarán conceptos básicos de mínimos cuadrados y aprendizaje automático utilizando redes neuronales convolucionales para resolver problemas de clasificación de imágenes todo utilizando el lenguaje Python, con la libreria Pytorch.

El objetivo del trabajo es poner en práctica el conocimiento adquirido en clase sobre estos temas por medio de ejercicios que permitan al estudiante experimentar con ejemplos de uso.

A. Minimos cuadrados con tensores de Pytorch (6 puntos)

El presente ejercicio se va a realizar utilizando el conjunto de datos ≪Default of Credit Card Clients Dataset. Default Payments of Credit Card Clients in Taiwan from 2005≫. Se adjunta un archivo con los datos (defaultofcredit). Una descripción completa del conjunto de datos está disponible en

https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset.

1. Cargue el conjunto de datos defaultofcredit.csv, los targets (b) corresponden a la columna: default_payment_next_month.

2. (1 punto) Explore el conjunto de datos, visualice algunas estadísticas y verifique que no existan valores faltantes.

3. (2 puntos) Implemente en python la función wopt = estimateOptimunW(X,b) la cual recibe los datos y sus targets (X y b), y estima el vector de pesos óptimo w_opt usando mínimos cuadrados. Debe programar en python a nivel de operaciones matriciales y/o vectoriales, no debe invocar a una función pre-construida que calcule mínimos cuadrados. Utilice la funcion estimateOptimumW para calcular el wopt para los datos.

4. (2 puntos) Implemente función forward, la cual estima las salidas del modelo al hacer T=f(X, wopt). Donde la función f(x) se refiere a la función de activación, que decide a cual clase pertenece cada muestra, segun el resultado del producto punto de las muestra y los pesos óptimos, en este caso simplemente usando la funcion signo o escalón.

5. (1 punto) Evalúe el error de predicción utilizando la distancia euclideana.

B. Perceptrón de una capa con tensores de Pytorch (15 puntos)

1. (10 puntos) Implemente el algoritmo del perceptrón de una capa rescindiendo al máximo de estructuras de tipo for, usando en su lugar operaciones matriciales. Debe implementarlo sin utilizar ninguna biblioteca, es decir en Pytorch no se puede usar ninguna clase o funcionalidad desarrollada por PyTorch o alguna otra biblioteca.

2. Reporte los resultados del clasificador: Para probar su clasificador utilice cúmulos de datos generados con la funcion createData disponible en el cuaderno de Jupyter adjunto, la cual implementa la generación de datos aleatorios. Tales datos serán utilizados como datos de prueba. Parametrice la cantidad de muestras, matriz de covarianza y medias. La meta es lograr que el perceptrón clasifique bien los datos generados. Es decir construir un separador lineal de las clases.

2.1. (3 puntos) Realice 2 pruebas con distintas distancias de separación entre las muestras de las clases, con una prueba linealmente separable, y otra no, y documente el numero de muestras mal clasificadas. Defina el conjunto de muestras de entrenamiento como el 70 % de las muestras aleatoriamente seleccionadas, y el resto utilícelas como muestras de prueba.

2.2. (2 puntos) Grafique el error o pérdida de entrenamiento de al menos dos corridas, con todas las iteraciones de esas corridas. Ejemplo de gráfica de error por época:

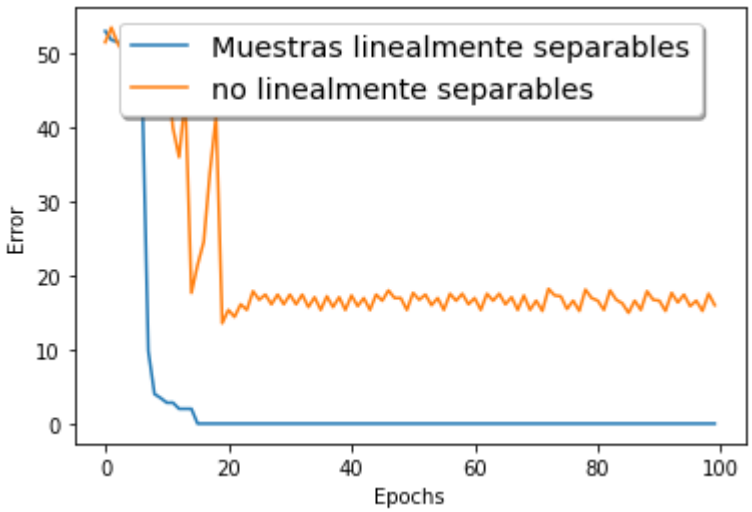

Figura 1. Error de entrenamiento para muestras linealmente separables generadas de forma aleatoria

C. Redes neuronales convolucionales (16 puntos)

Utilice PyTorch para implementar una red neuronal profunda para clasificar imágenes utilizando capas convolucionales. Realice las siguientes actividades vistas en clase:

1. (2 puntos) Cargar y normalizar los datos.
2. (2 puntos) Explorar los datos
3. (2 puntos) Definir la red convolucional
4. (2 puntos) Definir los hiperparametros, por ejemplo, funcion de perdida, el optimizador, entre otros.
5. (2 puntos) Entrenar la red
6. (3 puntos) Evaluar el modelo resultante Accuracy, Precision, Recall y F1 (investigue cómo se utilizan estas medidas en clasificación)
7. (3 puntos) Presentar al menos cuatro conclusiones.

Para el ejercicio **seleccione alguno de los siguientes conjuntos de datos:**

1. A Large Scale Fish Dataset: https://www.kaggle.com/crowww/a-large-scale-fish-dataset
2. 300 Bird Species - Classification: https://www.kaggle.com/gpiosenka/100-bird-species
3. Skin Cancer MNIST: HAM10000: https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000
4. Pokemon Image Dataset: https://www.kaggle.com/vishalsubbiah/pokemon-images-and-types