Tarea 4: Redes Neuronales

Profesor: Felipe Tobar
Auxiliares: Mauricio Araneda, Alejandro Cuevas y Mauricio Romero
Consultas: Todo el cuerpo docente.
Fecha entrega: 23/6/2019
Formato entrega: Entregue un informe en formato PDF con una extensión de a lo más 2 páginas presentando y analizando sus resultados, detalle la metodología utilizada y adicionalmente debe entregar un jupyter notebook con los códigos que creó para resolver la tarea.

Esta es una de las 2 versiones de la tarea4, deben elegir solo una de las versiones (SVM o redes).

P1. Clasificación con redes neuronales

Se pretende realizar la tarea de clasificación sobre el dataset MNIST, usando Redes neuronales como clasificador, para esto:

a) (1 pts.) Cargue los datos (Vea el demo para cargarlos) y sepárelos de forma aleatoria en conjuntos de entrenamiento $5/7$, validación $1/7$ y prueba $1/7$ (si carga con pytorch el conjunto de prueba ya esta definido).

Normalice los datos de forma que el valor de cada pixel se encuentre en el rango $[0, 1]$ en vez de $[0, 255]$, para esto divida todo el dataset por 255.

En caso de cargar los datos con Pytorch: Transforme sus datos de tal forma que cada imagen de 28×28 sea un vector de tamaño 784.

Observación: Le puede ser util train_test_split de sklearn.

b) (3 pts.) Investigue al menos 3 algoritmos de optimización utilizados para el entrenamiento de redes neuronales y compárelos con gradiente descendente vainilla en términos algorítmicos, hiperparámetros, etc. Seleccione un algoritmo para utilizarlo en su tarea y fundamente su elección.

c) (3 pts.) Construya una clase MLP que le permita agregar capas y regularizadores en el constructor. Es decir, al instanciar la clase debe poder ingresar número de capas, neuronas por capas y opciones extras como regularización o dropout.

d) (5 pts.) Entrene y pruebe las siguientes arquitecturas:

1. MLP 1 capa.
2. MLP 2 capas.
3. MLP 1 capa con regularización L2.
4. MLP 1 capa con reg. dropout en capa oculta.

Comente lo que sucede durante el entrenamiento con la función de costo, muestre como cambia en función de las épocas y analice el resultado de clasificación en base a la matriz de confusión.

¿Qué puede decir sobre la regularización en cada caso? ¿Cómo afecta el número de parámetros de los modelos bajo estudio?

Utilizando el conjunto de validación guarde* el mejor modelo obtenido durante el entrenamiento y compare las métricas de desempeño en el conjunto de test, en base al mejor modelo en cada caso.

Observaciones: En cuanto a los demás parámetros de red, tales como función de activación, cantidad de neuronas por cada y tazas de aprendizaje se recomienda:

- Utilice alrededor de 100 neuronas por capa.
- Entrene un número cercano a 20 épocas.
- Se recomienda utilizar funciones de activación ReLu.
- Dada esa función de activación, se romienda una tasa de aprendizaje del orden de $lr = 2 \cdot 10^{-4}$. (Tomar en cuenta que con otra función de activación puede variar el orden de magnitud de la tasa a usar)
- Se recomienda un tamaño de batch $batch_{size} = 100$.
- La regularización L2 se puede agregar en el optimizador a través del argumento *weight_decay*.
- $\star$ Para guardar y cargar modelos en Pytorch revise `https://pytorch.org/tutorials/beginner/saving_loading_models.html` .