

Actividad 2

Pytorch

Profesor: Pablo Estévez

Auxiliar: Pablo Cornejo

Ayudantes: Diego Castillo, Andrés González, Sebastián Guzmán, Francisco Soto

Instrucciones Generales

- **Importante:** Todos los gráficos realizados **deben** tener una etiqueta en el eje X y en el eje Y, además de un título. Las etiquetas y los títulos son libres a ser determinados por cada uno, sin embargo, deben estar relacionados con el contexto del gráfico.
- Por favor comentar los códigos. Si hay subsecciones, se recomienda comentar los lugares donde comienza cada subsección.
- Utilicen nombres significativos para variables, funciones y clases. Esto les ayudará principalmente a ustedes cuando lean sus códigos en un futuro y también al resto de la gente con la que trabajen luego.

Instrucciones Específicas

- Se recomienda ver el material del auxiliar 2 antes de hacer esta actividad.
- Para la pregunta 1 no se pueden utilizar los módulos `nn.Linear`, `nn.Sigmoid` y `nn.Sequential`. En la pregunta 2 sí pueden utilizarlos.
- Para la pregunta 1 utilice como base el código que se encuentra en el archivo `P1.py`. Entregue este archivo modificado con su código agregado.
- Para la pregunta 2 utilice como base el código que se encuentra en el archivo `P2.ipynb` y entregue este mismo archivo con su código agregado. En este se definen rutinas de entrenamiento y ustedes sólo deben completar algunas partes. No es necesario programar todo desde cero, pero si quieren hacerlo también se acepta, siempre y cuando se obtenga lo que se pide.
- Para la generación de datos de la pregunta 2 pueden utilizar el mismo código utilizado en la actividad 1, pero tengan cuidado con que en esta actividad se agrega un paso extra de normalización de los datos y creación de dataset de pytorch.

P1: Programación de módulos de pytorch

En esta pregunta se programarán algunos módulos de pytorch preexistentes. Cada módulo debe heredar de la clase `nn.Module` de pytorch, definiendo el comportamiento de la capa en el método `forward`. No pueden utilizar los módulos que ya definen estos comportamientos de forma directa, pero sí pueden utilizar otras funciones de pytorch básicas como multiplicaciones matriciales y función exponencial.

1. Cree una capa lineal desde cero (capa que imite el comportamiento de `nn.Linear`). Esta capa debe recibir el número de neuronas de entrada y el número de neuronas de salida, y calcular la multiplicación matricial entre la entrada y los pesos, sumando finalmente los parámetros llamados bias (ver auxiliar 2).
2. Cree un módulo que calcule la función sigmoide. Puede utilizar sólo la función `torch.exp` (capa que imite el comportamiento de `nn.Sigmoid`).
3. Cree un módulo que reciba un número arbitrario de módulos de pytorch (clases que heredan de `nn.Module`) y corra estos módulos de forma secuencial alimentando al siguiente módulo con la salida del anterior (capa que imite el comportamiento de `nn.Sequential`).

P2: Modelo lineal con datos sintéticos

1. **Creación de features sintéticas:** Genere una matriz con 1000 muestras generadas de una distribución [normal bivariada](#) con medias $[-1, -4]$ y matriz de covarianza:

$$\begin{pmatrix} 1 & 0.6 \\ 0.6 & 1 \end{pmatrix}$$

Realice lo mismo en una matriz aparte, cambiando las medias a $[3, -7]$ y utilizando la misma matriz de covarianza anterior. Concatene ambas matrices para obtener una matriz final de tamaño 2000x2. Normalice estos datos utilizando [StandardScaler](#) de `sklearn`.

Creación de las clases: Genere un nuevo vector de 2000 elementos tal que los primeros 1000 elementos sean ceros y los últimos 1000 sean unos.

Creación del dataset: Cree un `TensorDataset` de `pytorch` con los datos anteriores (ver auxiliar 2).

2. Defina un modelo lineal con tamaño de entrada igual al número de *features* de los datos y tamaño de salida igual a 1. Para esto, utilice la clase `nn.Linear` de `pytorch`. Agregue una capa con función de activación sigmoide luego de la capa lineal. Se recomienda utilizar el módulo `nn.Sequential` en su modelo.
3. Entrene este modelo sobre los datos anteriores (tarea de clasificación). No es necesario generar un dataset de validación ni reportar un gráfico del *loss*.

4. Su modelo está compuesto de distintos parámetros. En este caso son 2 pesos y un *bias*. Muestre estos valores y sus formas.
5. Grafique los datos y el hiperplano separador obtenido luego de entrenar en un mismo gráfico. Para esto, utilice la función auxiliar entregada en el material de la tarea (lea la documentación de esta función). Para realizar este gráfico no invierta la normalización de los datos del inicio.