Laboratorio No. 3

Regresiones Lineales vía GD, SGD y Minibatch

Instrucciones:

- Para este laboratorio debe realizar un pequeño documento que incluya las respuestas y gráficos correspondientes a cada una de las partes detalladas a continuación. Recuerde que respuesta no justificada no recibirá calificación.
- Resuelva cada una de los ejercicios que se le solicitan a continuación utilizando Python, no olvide subir su código con extensión *.py* o adjuntar el link de colab o respositorio de trabajo.

En este laboratorio, consideraremos el problema de ajustar una linea de regresión a un conjunto de datos. El modelo de regresión esta dado por la ecuación $Y = \beta X$, en donde X representa el conjunto de entradas de dimensión $n \times d$ y Y representa el conjunto de observaciones de dimensión $n \times 1$. El objetivo es determinar el vector de coeficientes β de dimensión $d \times 1$ y que minimiza la suma de errores al cuadrado, i.e.

$$\min \quad f(\beta) = \sum_{i=1}^{n} f_i(\beta), \tag{1}$$

en donde $f_i(\beta) = (\beta^T x_i - y_i)^2$ denota el cuadrado del error de la i-esima observación $x_i \in \mathbb{R}^d$ y $y_i \in \mathbb{R}$.

Los datos de este laboratorio, van a ser generados a partir del siguiente código:

```
d = 100 #cantidad de columnas para el dataset.
n = 1000 #cantidad de observaciones para el dataset.
X = np.random.normal(0,1, size=(n,d))
beta_true = np.random.normal(0,1, size=(d,1))
y = X.dot(beta_true) + np.random.normal(0,0.5,size=(n,1))
```

Del código anterior es claro que para el training set, se seleccionan n puntos $x_1, \dots x_n$ extraidos independientemente de una distribución Gaussiana d-dimensional, luego seleccionamos el *verdadero* vector de coeficientes β^* (a partir de una distribución Gaussiana d-dimensional). El vector Y se forma asignándole a cada observación x_i la etiqueta $(\beta^*)^T x_i$ más una componente de ruido obtenida de una distribución Gaussiana 1-dimensional. Utilizaremos dicho código para estudiar cuatro formas diferentes de calcular el vector β^* utilizando: una Solución Cerrada, el algoritmo Gradient Descent (GD), el algoritmo de Stochastic Gradient Descent (SGD) y Mini Batch Gradient Descent (MBGD).

Parte 1: Solución Cerrada

En clase probamos que la solución cerrada para el problema [1] viene dada por:

$$\beta^* = (X^T X)^{-1} X^T Y$$

Recuerde que X es una matriz de dimensión $n \times d$ con una observación por fila, y Y es un vector n-dimensional con las etiquetas respectivas. Utilizando la data generada con el código anterior calcule el valor de β^* así como el valor de la función objetivo correspondiente. ¿Por qué en la práctica no se utiliza este método? Justifique su respuesta.

Parte 2: GD
En esta parte resolveremos el problema [1] utilizando el algoritmo GD. Ejecute este algoritmo 3 veces, una para cada uno de los siguientes step sizes constantes (learning rates) 0.00005, 0.0005 y 0.0007. Inicialice el vector β con el vector nulo y seleccione un criterio adecuada para detener el algoritmo. Luego realice una gráfica del valor de la función objetivo del problema (1) con cada uno de los step sizes dados para 20 iteraciones. Las tres gráficas deben estar en el mismo plano. Finalmente, comente cómo el step size afecta la convergencia del GD, puede experimentar con otros step sizes para justificar su respuesta. ¿Con cuál step size constante obtuvo el "mejor" resultado?

Parte 3: SGD
En esta parte deberá implementar SGD para resolver aproximadamente el problema (1) ejecute este algoritmo 3 veces, una para cada uno de los siguientes step sizes constantes (learning rates) 0.0005, 0.005 y 0.01. Inicialice el vector β con el vector nulo y realice mil iteraciones por cada step size. Realice un gráfico del valor de la función objetivo versus el número de la iteración para cada uno de los 3 step sizes dados. Las tres gráficas deben estar en el mismo plano. Finalmente, comente cómo el step size afecta la convergencia del SGD, puede experimentar con otros step sizes para justificar su respuesta. ¿Con cuál step size constante obtuvo el "mejor" resultado?

Parte 4: MBGD
Repita la parte 3 pero implementando el algoritmo MBGD con batches de tamaño 25 , 50 y 100, considerando para cada uno de estos casos los siguientes step sizes: 0.0005, 0.005 y 0.01. ¿Cómo el tamaño del batch afecta la convergencia de este algoritmo? ¿Con cuál combinación de batch size y step size obtuvo el mejor resultado? Justifique su respuesta.

Parte 5: Comparación
Compare el desempeño de cada uno de los métodos implementados en este laboratorio. Realice una lista de posiciones ordenando los métodos de mayor a menor desempeño; para ello, considere el valor óptimo de la función objetivo f^*, el número de iteraciones y el error en el β^* obtenido versus el β real (`beta_true`).