Laboratorio No. 4

Gradient Descent Variants y Newton's Method

Problema 1 – GD Variants

En este primer problema del laboratorio, consideraremos el problema de ajustar una recta de regresión a un conjunto de datos. El modelo de regresión esta dado por la ecuación $Ax = b$, en donde A representa el conjunto de entradas de dimensión $n \times d$ y b representa el conjunto de observaciones de dimensión $n \times 1$. El objetivo es determinar el *vector de coeficientes* x de dimensión $d \times 1$ y que *minimiza* la suma de errores al cuadrado, i.e.

$$\min_{x \in \mathbb{R}^d} \quad f(x) \triangleq \sum_{i=1}^{n} f_i(x), \tag{1}$$

en donde, $f_i(x) = (x^T a_i - b_i)^2$ denota el cuadrado del error de la i-ésima observación $a_i \in \mathbb{R}^d$ y $b_i \in \mathbb{R}$.

El conjunto de datos a utilizar en esta parte del laboratorio deberán ser generados a partir del siguiente código:

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

Del código anterior es claro que, para el training set, se seleccionan n puntos $a_1, \dots a_n$ extraídos independientemente de una distribución Gaussiana d-dimensional, luego seleccionamos el *verdadero* vector de coeficientes x^* (a partir de una distribución Gaussiana d-dimensional). El vector b se forma asignándole a cada observación a_i la etiqueta $(x^*)^T a_i$ más una componente de ruido obtenida de una distribución Gaussiana unidimensional. Utilizaremos dicho código para estudiar cuatro formas diferentes de calcular el vector x^*, estas son: 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:

$$x^* = (A^T A)^{-1} A^T b$$

Recuerde que A es una matriz de dimensión $n \times d$ con una observación por fila, mientras que b es un vector n-dimensional con las etiquetas respectivas. Utilizando la data generada con el código anterior, calcule el valor de x^* así como el valor de la función objetivo f. ¿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 (o learning rates) constantes 0.00005, 0.0005

y 0.0007. Inicialice el vector x con el vector nulo y seleccione un criterio adecuada para detener el algoritmo. Luego, realice una gráfica del valor de la función objetivo f del problema (1) versus el índice de la iteración para cada uno de los 3 step sizes dados. Las tres gráficas deben estar en el mismo plano y mostrar los resultados de (al menos) las primeras 20 iteraciones. Finalmente, comente cómo el step size afecta la convergencia del algoritmo 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 el algoritmo SGD para resolver aproximadamente el problema (1). Ejecute este algoritmo 3 veces, una para cada uno de los siguientes step sizes (o learning rates) constantes 0.0005, 0.005 y 0.01. Inicialice el vector x con el vector nulo y realice 1000 iteraciones para cada step size. Realice un gráfico del valor de la función objetivo f versus el número de iteraciones 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 ahora implemente el algoritmo MBGD para resolver aproximadamente el problema (1). Para ello, utilice batches de tamaño 25, 50 y 100, considerando para cada uno de estos casos los siguientes step sizes (o learning rates) constantes: 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. Realice una lista de posiciones ordenando los métodos de mejor a peor desempeño. Para ello, considere el valor óptimo de la función objetivo f^* alcanzado, el número de iteraciones requeridas y por supuesto, el error en el x^* obtenido versus el x real (`x_true`).

Problema 2 – Método de Newton

Considere de nuevo el problema de optimización

$$\min_{(x_1,x_2)\in\mathbb{R}^2} \; f(x_1,x_2) = 100\,(x_2 - x_1^2)^2 + (1-x_1)^2. \tag{2}$$

Recuerde que la función objetivo de este problema es conocida como *Rosenbrock's Function* y es utilizada como benchmark en la evaluación de algoritmos.

Parte 1 – GD con Bactracking Line Search

En el laboratorio anterior, se aplicó el método de GD (con un step size constante) para resolver el problema (2). Ahora, utilice *Backtracking Line Search* para determinar el step size. Fije los parámetros de este algoritmo de acuerdo a los valores dados en clase. Compare los resultados de ambas formas de determinar el step size, para ello utilice $x_0 = (0,0)^T, (0.6,0.6)^T, (-0.5,1)^T$ y $(-1.2,1)^T$.

Parte 2 – Método de Newton con Bactracking Line Search

Aplique el *método de Netwon* para resolver este problema de optimización. Al igual que en el Problema 2, utilice $x_0 = (0, 0)^T$ y un step-size unitario. ¿Qué sucede si varía el punto inicial a $x_0 = (0.6, 0.6)^T, (-0.5, 1)^T$ y $(-1.2, 1)^T$ y si el step size lo calcula utilizando *Backtracking Line Search*? Experimente con estos parámetros y reporte sus hallazgos. Detenga la ejecución del algoritmo cuando $\|\nabla f(x_k)\| < 10^{-8}$ o bien cuando el número de iteraciones exceda 3000. Su output debe ser mostrado en una tabla con las cuatro columnas siguientes:

a) el número k de la iteración,

b) x_k,

c) la dirección p_k, y

d) $\|\nabla f(x_k)\|$.