Mathematical Foundation of DNN : HW 1

Jeong Min Lee

March 11, 2024

1

In this problem, I followed the notation given by the Petersen and Pedersen[1]. Also, I denote the element of matrix X in ith row and jth column as X_{ij}

a

$$\left(\frac{\partial}{\partial\theta} l_i(\theta)\right)_j = \frac{\partial l_i(\theta)}{\partial \theta_j} \tag{1}$$

$$= (X_i^T\theta - Y_i)\frac{\partial}{\partial\theta_j} X_i^T\theta \tag{2}$$

$$= (X_i^T\theta - Y)X_{ij} \tag{3}$$

By enumerating the last equation in column vector, one can get the following result.

$$\frac{\partial}{\partial\theta} l_i(\theta) = (X_i^T\theta - Y_i)X_i \tag{4}$$

Note that $(X_i^T\theta - Y_i) \in \mathbb{R}$ and thus, enumeration affects only to the last X_{ij}. (It results in X_i)

b

$$\mathcal{L}(\theta) = \frac{1}{2}\|X\theta - Y\|^2 \tag{5}$$

$$= \frac{1}{2}\sum_i \left(X_i^T\theta - Y_i\right)^2 \tag{6}$$

$$= \sum_i l_i(\theta) \tag{7}$$

From the obseravtion above and the result of the problem(a),

$$\nabla_\theta \mathcal{L}(\theta) = \sum_i \nabla_\theta l_i(\theta) \tag{8}$$

$$= \sum_i (X_i^T\theta - Y_i)X_i \tag{9}$$

$$= \sum_i X_i^T\theta X_i - X_iY_i \tag{10}$$

According to the hint given by the original problem statement, noting that the matrix consisting of the X_i as a column is X^T,

$$\nabla_\theta \mathcal{L}(\theta) = X^T \begin{pmatrix} X_1^T\theta \\ X_2^T\theta \\ \vdots \\ X_N^T\theta \end{pmatrix} - X^TY \tag{11}$$

$$= X^T \begin{pmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_N^T \end{pmatrix} \theta - X^TY \tag{12}$$

$$= X^TX\theta - X^TY \tag{13}$$

$$= X^T(X\theta - Y) \tag{14}$$

2

Since $f'(\theta) = \theta$,

$$\theta^{(k+1)} = \theta^{(k)} - \alpha f'(\theta^{(k)}) \tag{15}$$
$$= (1-\alpha)\theta^{(k)} \tag{16}$$

1

$$\frac{\theta^{(k+1)}}{\theta^{(k)}} = 1 - \alpha \tag{17}$$
$$\therefore \theta^{(k)} = \theta^{(0)}(1-\alpha)^k \tag{18}$$

If $\alpha > 2$, then $|1-\alpha| > 1$. This,from geometric sequence's convergence condition, results in $\theta^{(k)} \to \infty$ as $k \to \infty$

3

From problem 1, I showed the following.

$$\nabla f(\theta) = X^T(X\theta - Y) \tag{19}$$

Inserting it to the GD,

$$\theta^{(k+1)} = \theta^{(k)} - \alpha X^T(X\theta^{(k)} - Y) \tag{20}$$
$$= \theta^{(k)} - \alpha X^T X\theta^{(k)} + \alpha X^T Y \tag{21}$$
$$= \theta^{(k)} - \alpha X^T X\theta^{(k)} + \alpha X X^T \theta^\star \tag{22}$$

By substracting $\theta^\star$ on both side of equation (22), the following equation is derived.

$$\theta^{(k+1)} - \theta^\star = (I_p - \alpha X^T X)(\theta^{(k)} - \theta^\star) \tag{23}$$

Note that I_p denotes identity matrix whose dimension is $p \times p$. This recursive equation has the following closed form. Here I denoted $\theta^k - \theta^\star$ as $\zeta^{(k)}$.

$$\zeta^k = \zeta^{(0)}(I_p - \alpha X^T X) \tag{24}$$

According to SVD[2],

$$X^T X = V\Sigma^2 V^T \tag{25}$$

where $V \in O(p)$, orthonormal group $O(p)$ with dimension p. I assumed that diagonal entries of $\Sigma^2 = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \cdots, \sigma_p^2)$, the eigenvalues of $X^T X$ are ordered.(I denote diagonal matrix whose diagonal entries are $a_1, a_2, \cdots, a_n$ as $\mathrm{diag}(a_1, a_2, \cdot, a_n)$) This implies that $\rho(X^T X) = \sigma_1^2$. Since $V \in O(p)$, the equation (23) can be written as follows.

$$\zeta^k = \left(V I_p V^T - \alpha V\Sigma^2 V^T\right)^k \zeta^{(0)} \tag{26}$$
$$= V(I_p - \alpha\Sigma^2)^k V^T \zeta^{(0)} \tag{27}$$

$$\|\zeta^{(k)}\|^2 = [\zeta^{(0)}]^T V^T (I_p - \alpha\Sigma^2)^{2k} V\zeta^{(0)} \tag{28}$$
$$= [\zeta^{(0)}]^T V^T \mathrm{diag}(1-\alpha\sigma_1^2, 1-\alpha\sigma_2^2, \cdots .1-\alpha\sigma_p^2)^{2k} V\zeta^{(0)} \tag{29}$$
$$= [\zeta^{(0)}]^T V^T \mathrm{diag}((1-\alpha\sigma_1^2)^{2k}, (1-\alpha\sigma_2^2)^{2k}, \cdots, (1-\alpha\sigma_p^2)^{2k}) \tag{30}$$

Since $\alpha > 2/\rho(X^T X)$ implies $\left|1-\alpha\sigma_1^2\right| > 1$, $\zeta^{(k)}$,that is $\theta^{(k)}$, diverges.

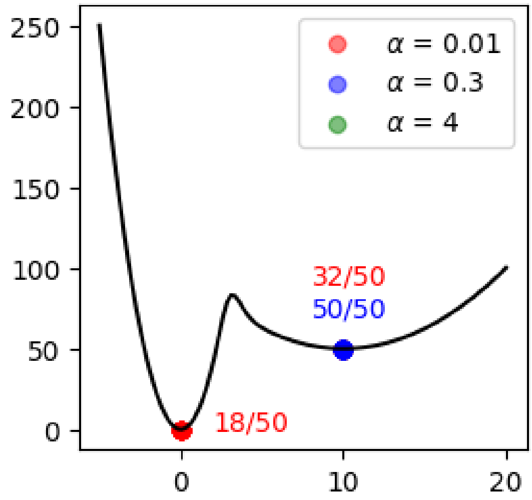

Figure 1: The result of gradient descent of $f(x)$ is described. The proportion of initial points that converges to each local minimum is depicted. The color of given ratio matches to the each α. Note that there is no scatter in this plot corresponding to $\alpha = 4$ since none of them converged.

4

I implement my own gradient descent algorithm to solve this problem. As mentioned in homework document, I use three learning rate, $\alpha = [0.01, 0.3, 4]$. Furthermore, for the robust analysis, I randomly sampled 50 initial point of x for each α. The result agrees to the general notion that GD with small learning rate temps to converge both of local minma, while that with intermediate learning rate converges only to the wide local minimum. The following python code is the code used to solve this problem. The result of these code is described on Figure 1.

```
import numpy as np
import matplotlib.pyplot as plt
alpha_lst = [0.01, 0.3,4]
iter_num = 50 # number of sampling
epsilon = 1e-4 # acceptable level
result = dict() # contain the result of experiment
max_step = 1000 # maximum step not to run while loop infintely

for alpha in alpha_lst:
    print(f"--------------------------alpha = {alpha}---------------------------------------")
    result[alpha] = []
    for i in range(iter_num):
        x = 25*np.random.random_sample()-5
        x_init = x
        step = 0
        while fprime(x) > epsilon or step < max_step:
            print(f"step : {step}/{max_step}")
            step +=1
            x = x - alpha * fprime(x) # Gradient Descent
        result[alpha].append((x,x_init))
```

5

Since there is no theoretical analysis requirment on this probelm. I briefly introduce my code. The following box is the code I implemented.

```
    import numpy as np

    class Convolution1d :
        def __init__(self, filt) :
            self.__filt = filt
            self.__r = filt.size
```

[1]To resolve the confusion due to the notation between kth element and power of k, I used parenthesis to denote the kth element

[2]Note that the Spectral Theorem of symmetric matrix guarantees SVD. The diagonal entries, which is the distinct eigenvalues, are positive. Also, thanks to the form of $X^T X$, it is postive semi-definite.

```
        self.T = TransposedConvolution1d(self.__filt)

    def __matmul__(self, vector) :
        r, n = self.__r, vector.size

        return np.asarray([sum([self.__filt[i] * vector[i+j] for i in range(r)]) for j in
                                                range(n-r+1)])  # IMPLEMENT THIS

class TransposedConvolution1d :
    '''
    Transpose of 1-dimensional convolution operator used for the
    transpose-convolution operation A.T@(...)
    '''
    def __init__(self, filt) :
        self.__filt = filt
        self.__r = filt.size

    def __matmul__(self, vector) :
        r = self.__r
        n = vector.size + r - 1
        vector = np.concatenate([np.zeros((r-1,)), vector, np.zeros((r-1,))]) # padding
        return  np.asarray([sum(self.__filt[::-1] * vector[i:i+3]) for i in range(len(
                                                vector)-r+1)]) # IMPLEMENT THIS

def huber_loss(x) :
    return np.sum( (1/2)*(x**2)*(np.abs(x)<=1) + (np.sign(x)*x-1/2)*(np.abs(x)>1) )
def huber_grad(x) :
    return x*(np.abs(x)<=1) + np.sign(x)*(np.abs(x)>1)


r, n, lam = 3, 20, 0.1

np.random.seed(0)
k = np.random.randn(r)
b = np.random.randn(n-r+1)
A = Convolution1d(k)
#from scipy.linalg import circulant
#A = circulant(np.concatenate((np.flip(k),np.zeros(n-r))))[r-1:,:]


x = np.zeros(n)
alpha = 0.01
for _ in range(100) :
    x = x - alpha*(A.T@(huber_grad(A@x-b))+lam*x)

print(huber_loss(A@x-b)+0.5*lam*np.linalg.norm(x)**2)
```

For **Convolution1d.__matmul__()**, I implemented it by using simple list comprehension. ith iteration calculate the product of each elements in k and vector and **sum()** function, python native function, calcaulate their sum. This repeat until the every row of A is seen. However, for **TransposeConvlution1d.__matmul()__**, I padded the target vector. I strongly believe implementing this function in list comprehension with single line is possible, but I think it is not that intuitive to read. Thus, to maintain the readability of my code, I add additional single line that pads zeros on the target vector. As a result, I got 0.4587586843129764 for the result of execution, which matches to that of the naive implementation of A.

References

[1] K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. Version 20081110.