MathDNN Homework 1

Department of Computer Science and Engineering
2021-16988 Jaewan Park

Problem 1

(a) Let $X_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}$ and $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}$, then for $j = 1, \cdots, p$,

$$\begin{aligned} \frac{\partial}{\partial \theta_j} l_i(\theta) &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2} (X_i^\intercal \theta - Y_i)^2 \right] \\ &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2} (x_{i1}\theta_1 + \cdots + x_{ip}\theta_p - Y_i)^2 \right] \\ &= (x_{i1}\theta_1 + \cdots + x_{ip}\theta_p - Y_i) x_{ij} \\ &= (X_i^\intercal \theta - Y_i) x_{ij}. \end{aligned}$$

Therefore

$$\begin{aligned} \nabla_\theta l_i(\theta) &= \begin{bmatrix} \frac{\partial}{\partial \theta_1} l_i(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_p} l_i(\theta) \end{bmatrix} \\ &= \begin{bmatrix} (X_i^\intercal \theta - Y_i) x_{i1} \\ \vdots \\ (X_i^\intercal \theta - Y_i) x_{ip} \end{bmatrix} = (X_i^\intercal \theta - Y_i) \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \\ &= (X_i^\intercal \theta - Y_i) X_i. \end{aligned}$$

(b) Let $X_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}$ and $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}$, then for $j = 1, \cdots, p$,

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \mathcal{L}(\theta) &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2} \left\| X\theta - Y \right\|^2 \right] \\ &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{2} \left\| \begin{matrix} X_1^\intercal \theta - Y_1 \\ \vdots \\ X_N^\intercal \theta - Y_N \end{matrix} \right\|^2 \right] = \frac{\partial}{\partial \theta_j} \left[\sum_{i=1}^N \frac{1}{2} (X_i^\intercal \theta - Y_i)^2 \right] = \frac{\partial}{\partial \theta_j} \left[\sum_{i=1}^N l_i(\theta) \right] \\ &= \sum_{i=1}^N \frac{\partial}{\partial \theta_j} l_i(\theta) = \sum_{i=1}^N (X_i^\intercal \theta - Y_i) x_{ij}. \end{aligned}$$

Therefore

$$
\begin{aligned}
\nabla_\theta \mathcal{L}(\theta) &= \begin{bmatrix} \sum_{i=1}^N (X_i^\intercal \theta - Y_i) x_{i1} \\ \vdots \\ \sum_{i=1}^N (X_i^\intercal \theta - Y_i) x_{ip} \end{bmatrix} \\
&= \begin{bmatrix} (X_1^\intercal \theta - Y_1) x_{11} + (X_2^\intercal \theta - Y_2) x_{21} + \cdots + (X_N^\intercal \theta - Y_N) x_{N1} \\ \vdots \\ (X_1^\intercal \theta - Y_1) x_{1p} + (X_2^\intercal \theta - Y_2) x_{2p} + \cdots + (X_N^\intercal \theta - Y_N) x_{Np} \end{bmatrix} \\
&= \sum_{i=1}^N X_i (X_i^\intercal \theta - Y_i) = \sum_{i=1}^N (X^\intercal)_{:,i} (X\theta - Y)_i \\
&= X^\intercal (X\theta - Y).
\end{aligned}
$$

Problem 2

Since $f(\theta) = \theta^2/2$, we can get $f'(\theta) = \theta$ and the iteration equation can be written as

$$\theta^{k+1} = (1 - \alpha)\theta^k.$$

Thus the nth iteration gives

$$\theta^n = (1-\alpha)\theta^{n-1} = \cdots = (1-\alpha)^n \theta^0.$$

Therefore if $\alpha > 2$ and $\theta^0 \neq 0$, θ^n diverges since it is a geometric sequence where its ratio is less than -1.

Problem 3

Since $f(\theta) = \frac{1}{2}\|X\theta - Y\|^2$, we can get $\nabla f(\theta) = X^\intercal (X\theta - Y)$ and the iteration equation can be written as

$$\theta^{k+1} = \theta^k - \alpha X^\intercal \big(X\theta^k - Y\big).$$

Assume $X^\intercal X$ is invertible and let $\theta^* = (X^\intercal X)^{-1} X^\intercal Y$, then

$$
\begin{aligned}
\theta^{k+1} - \theta^* &= \theta^k - \alpha X^\intercal \big(X\theta^k - Y\big) - (X^\intercal X)^{-1} X^\intercal Y \\
&= \theta^k - \alpha X^\intercal X \theta^k + \alpha X^\intercal Y - (X^\intercal X)^{-1} X^\intercal Y \\
&= (I - \alpha X^\intercal X)\theta^k + (\alpha X^\intercal X - I)(X^\intercal X)^{-1} X^\intercal Y = (I - \alpha X^\intercal X)\theta^k + (\alpha X^\intercal X - I)\theta^* \\
&= (I - \alpha X^\intercal X)\big(\theta^k - \theta^*\big)
\end{aligned}
$$

where $I \in \mathbb{R}^{p \times p}$ is an identity matrix. Thus the nth iteration gives

$$\theta^n - \theta^* = (I - \alpha X^\intercal X)\big(\theta^{n-1} - \theta^*\big) = \cdots = (I - \alpha X^\intercal X)^n \big(\theta^0 - \theta^*\big).$$

Since $X^\intercal X$ is a symmetric matrix and a positive matrix, it has p real positive eigenvalues and is diagonalizable by them. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ the eigenvalues and M the matrix with the eigenvectors as its columns, then

$$M^{-1}X^\intercal XM = \mathrm{diag}(\lambda_1, \cdots, \lambda_p), \quad X^\intercal X \sim \mathrm{diag}(\lambda_1, \cdots, \lambda_p)$$

and gradually

$$\begin{aligned} I - \alpha X^\intercal X &= I - \alpha M \mathrm{diag}(\lambda_1, \cdots, \lambda_p) M^{-1} \\ &= MIM^{-1} - M(\alpha\, \mathrm{diag}(\lambda_1, \cdots, \lambda_p))M^{-1} \\ &= M(I - \alpha\, \mathrm{diag}(\lambda_1, \cdots, \lambda_p))M^1 = M \mathrm{diag}(1 - \alpha\lambda_1, \cdots, 1 - \alpha\lambda_p)M^{-1} \end{aligned}$$

so $I - \alpha X^\intercal X \sim \mathrm{diag}(1 - \alpha\lambda_1, \cdots, 1 - \alpha\lambda_p)$. Therefore

$$\begin{aligned} (I - \alpha X^\intercal X)^n &= M \mathrm{diag}(1 - \alpha\lambda_1, \cdots, 1 - \alpha\lambda_p)^n M^{-1} \\ &= M \mathrm{diag}((1 - \alpha\lambda_1)^n, \cdots, (1 - \alpha\lambda_p)^n)M^{-1}. \end{aligned}$$

If $\alpha > 2/\rho(X^\intercal X)$, we can obtain $1 - \alpha\lambda_1 < -1$ since $\rho\big(X^T X\big) = \lambda_1 \geq 0$. Therefore $(1 - \alpha\lambda_1)^n$ diverges, and consequently $(I - \alpha X^\intercal X)^n$, $\theta^n - \theta^*$, and θ^n also diverge.

Problem 4

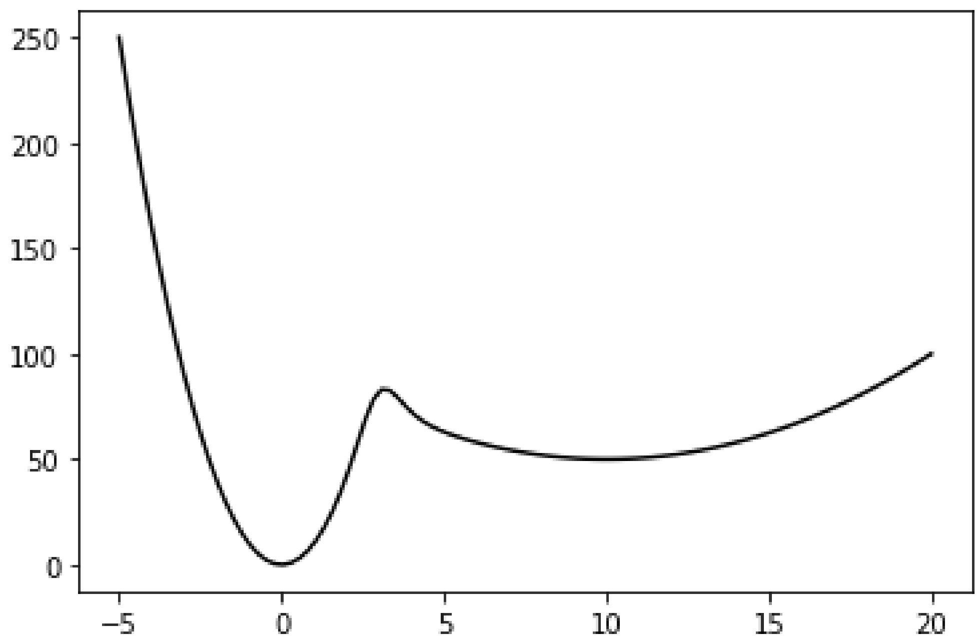
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random as rd
%matplotlib inline
np.seterr(invalid='ignore', over='ignore')  # suppress warning caused by division by␣
 ↪inf
```

[1]: {'divide': 'warn', 'over': 'warn', 'under': 'ignore', 'invalid': 'warn'}

[2]:
```
def f(x):
    return 1/(1 + np.exp(3*(x-3))) * 10 * x**2  + 1 / (1 + np.exp(-3*(x-3))) * (0.
 ↪5*(x-10)**2 + 50)

def fprime(x):
    return 1 / (1 + np.exp((-3)*(x-3))) * (x-10) + 1/(1 + np.exp(3*(x-3))) * 20 * x +␣
 ↪(3* np.exp(9))/(np.exp(9-1.5*x) + np.exp(1.5*x))**2 * ((0.5*(x-10)**2 + 50) - 10 *␣
 ↪x**2)
```

[3]:
```
x = np.linspace(-5,20,100)
plt.plot(x,f(x), 'k')
plt.show()
```

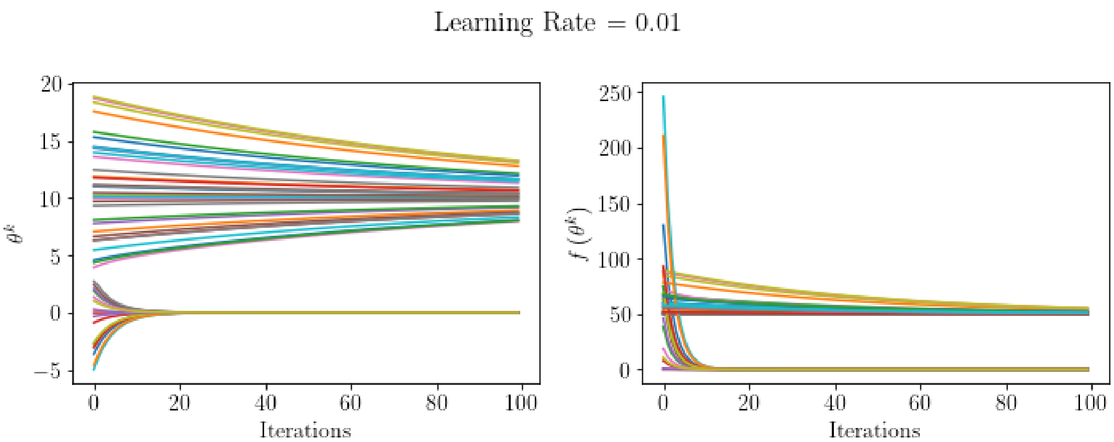


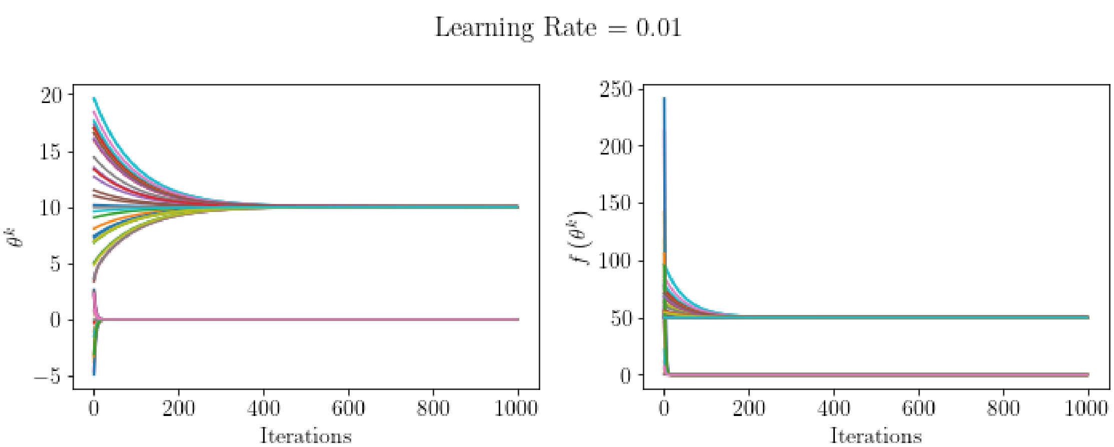
[4]:
```
def GD(alpha, start_x, iteration_count):
    x = [start_x]
    y = [f(start_x)]
    for _ in range(iteration_count - 1):
        x.append(x[-1] - alpha * fprime(x[-1]))
        y.append(f(x[-1]))
    return x, y

def GD_plot_alpha(alpha, sample_count, iteration_count):
    plt.rc('text', usetex=True)
    plt.rc('font', family='serif')
    plt.rc('font', size = 14)
    plt.rcParams["figure.figsize"] = [10, 4]
    plt.rcParams["figure.autolayout"] = True
    fig, axs = plt.subplots(1, 2)
    fig.suptitle('Learning Rate = {alpha}'.format(alpha=alpha))
    for _ in range(sample_count):
        x, y = GD(alpha, rd.uniform(-5, 20), iteration_count)
        axs[0].plot(list(range(iteration_count)), x)
        axs[0].set(xlabel='Iterations', ylabel=r'$\theta^k$')
        axs[1].plot(list(range(iteration_count)), y)
        axs[1].set(xlabel='Iterations', ylabel=r'$f\left(\theta^k\right)$')
    plt.show()
```

In the case of $\alpha = 0.01$, starting points that are relatively large converge to the wide minimum, and others converge to the sharp minimum.

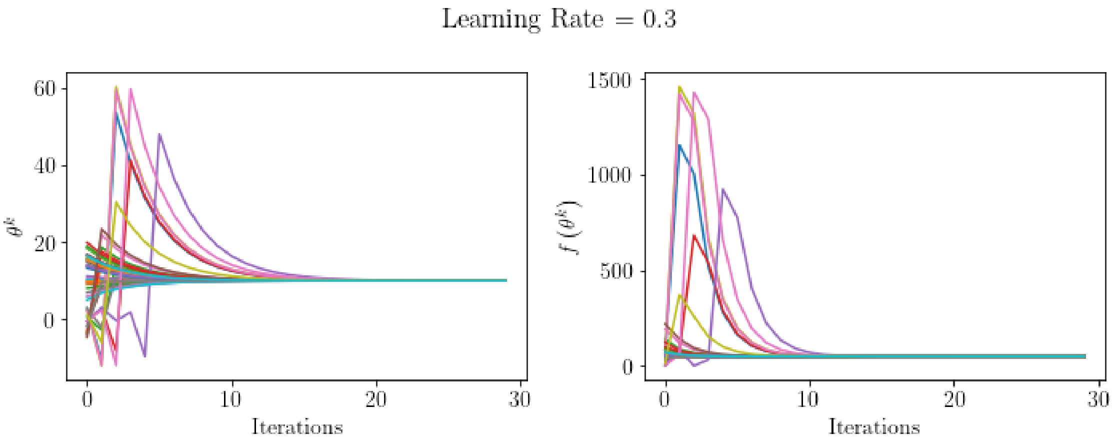
```
[5]: GD_plot_alpha(0.01, 50, 100)
     GD_plot_alpha(0.01, 50, 1000)
```

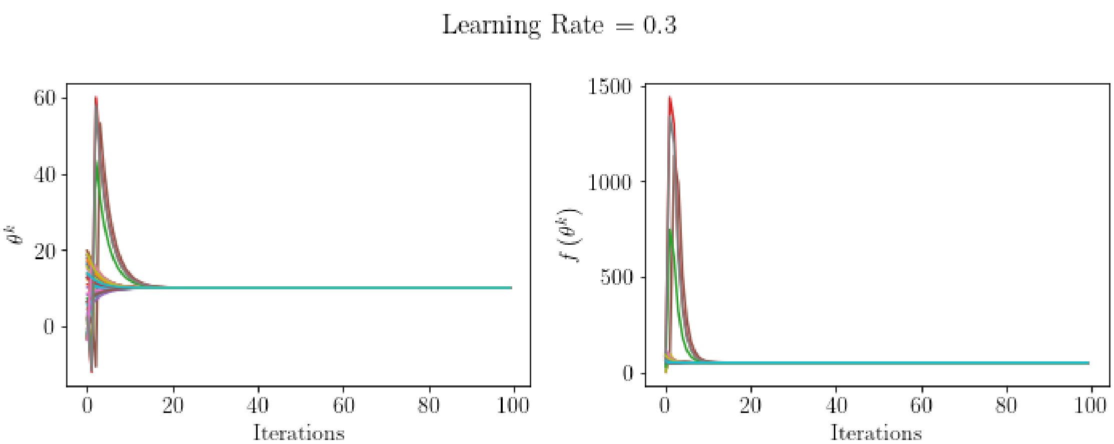




In the case of $\alpha = 0.3$, all cases converge to the wide minimum.

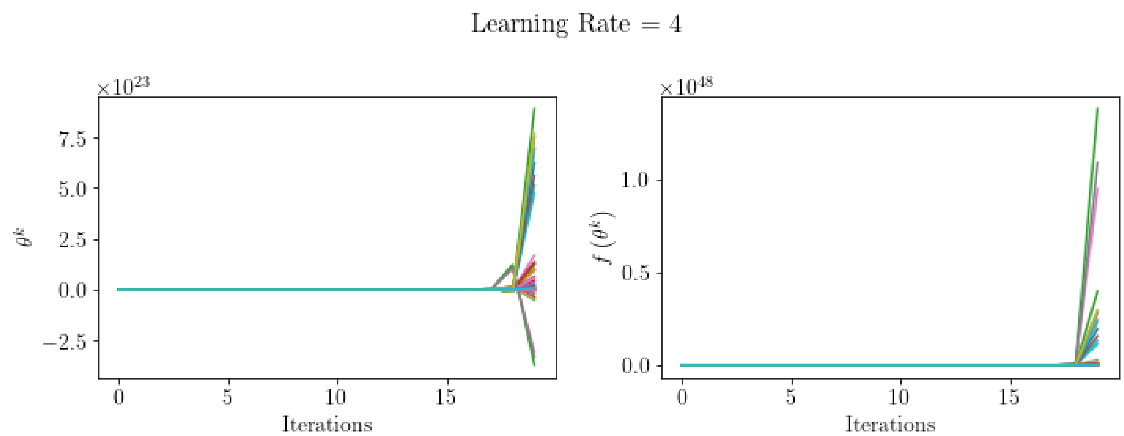
[6]:
```
GD_plot_alpha(0.3, 50, 30)
GD_plot_alpha(0.3, 50, 100)
```

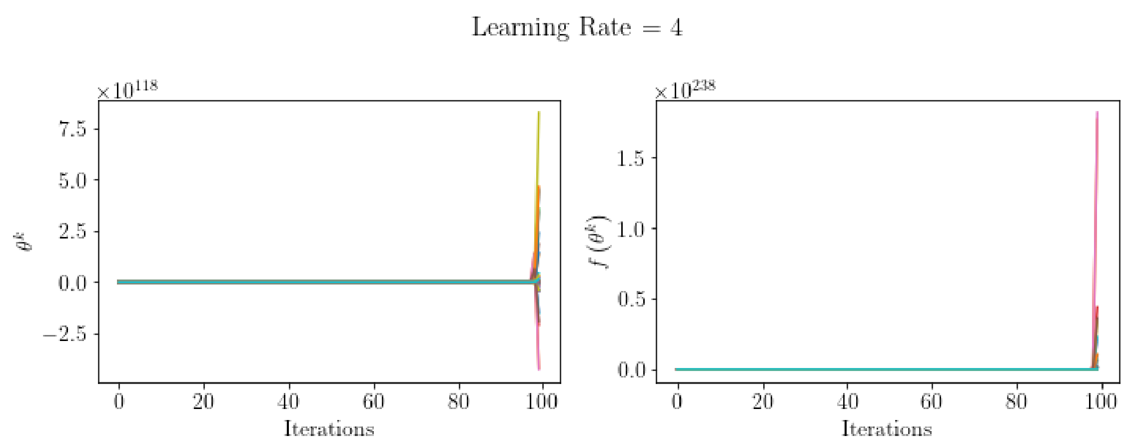




In the case of $\alpha = 4$, most cases do not converge.

[7]:
```
GD_plot_alpha(4, 50, 20)
GD_plot_alpha(4, 50, 100)
```





Problem 5

[8]:
```python
import numpy as np
```

[9]:
```python
class Convolution1d :
    def __init__(self, filt) :
        self.__filt = filt
        self.__r = filt.size
        self.T = TransposedConvolution1d(self.__filt)

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

        return np.asarray([sum([self.__filt[v] * vector[h + v] for v in range(r)]) for h in range(n - r + 1)])

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

        return np.asarray([sum([self.__filt[v] * vector[h - v] for v in range(max(0, h - n + r), min(r, h + 1))]) for h in range(n)])
```

[10]:
```python
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)
```

[11]:
```python
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)

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)
```

```
0.4587586843129764
```