2020-14281 이서용 HW1..

#1.

(a)

- $$\frac{\partial l_i(\theta)}{\partial \theta_j} = \frac{\partial \frac{1}{2}(X_{i1}\theta_1 + X_{i2}\theta_2 + \cdots + X_{ip}\theta_p - Y_i)^2}{\partial \theta_j} = (X_{i1}\theta_1 + X_{i2}\theta_2 + \cdots + X_{ip}\theta_p - Y_i)X_{ij}$$

$$= (X_i^T\theta - Y_i)X_{ij}. \quad \text{Thus, } \nabla_\theta l_i(\theta) = (X_i^T\theta - Y_i)X_i$$

$* \; X_i^T = [X_{i1}, X_{i2}, \ldots, X_{ip}]$

- $$X^T(X\theta - Y) = (X_1^T\theta - Y_1)X_1 + (X_2^T\theta - Y_2)X_2 + \cdots + (X_N^T\theta - Y_N)X_N$$

$$L(\theta) = \frac{1}{2}\left((X_1^T\theta - Y_1)^2 + (X_2^T\theta - Y_2)^2 + \cdots + (X_N^T\theta - Y_N)^2\right)$$

$$\frac{\partial L(\theta)}{\partial \theta_j} = \frac{\partial l_1(\theta)}{\partial \theta_j} + \frac{\partial l_2(\theta)}{\partial \theta_j} + \cdots + \frac{\partial l_N(\theta)}{\partial \theta_j}$$

$$= (X_1^T\theta - Y_1)X_{1j} + (X_2^T\theta - Y_2)X_{2j} + \cdots + (X_N^T\theta - Y_N)X_{Nj}$$

$$\therefore \nabla_\theta L(\theta) = (X_1^T\theta - Y_1)X_1 + (X_2^T\theta - Y_2)X_2 + \cdots + (X_N^T\theta - Y_N)X_N = X^T(X\theta - Y)$$

#2 $f(\theta) = \theta^2/2$

$f'(\theta^k) = \theta^k$. * θ^k: kth iterate of Gradient descent

$$\theta^1 = \theta^0 - \alpha f'(\theta^0) = \theta^0 - \alpha\theta^0$$

$$|\theta^1| = |1-\alpha||\theta^0|$$

$$|\theta^2| = |1-\alpha|^2|\theta^0|$$

$$\vdots$$

$$|\theta^n| = |1-\alpha|^n|\theta^0|.$$

If $\alpha > 2$ and $|\theta^0| > 0$, $|1-\alpha| > 1$ Thus $|\theta^n| \to \infty$ as $n \to \infty$.

It diverges if $\alpha > 2$!

#3 $f(\theta) = \frac{1}{2}\|X\theta - Y\|^2$

$\nabla f(\theta^k) = X^T(X\theta^k - Y)$ holds (proved in #1).

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

Let $\theta^* = (X^TX)^{-1}X^TY$. then,

$$\theta^{k+1} - \theta^* = \theta^k - (X^TX)^{-1}X^TY - \alpha X^TX\theta^k + \alpha X^TY$$

$$= (I - \alpha X^TX)\theta^k - (I - \alpha X^TX)(X^TX)^{-1}X^TY$$

$$= (I - \alpha X^TX)(\theta^k - \theta^*)$$

$$\theta^n - \theta^* = (I - \alpha X^TX)(\theta^{n-1} - \theta^*) = (I - \alpha X^TX)^2(\theta^{n-2} - \theta^*) = \cdots = (I - \alpha X^TX)^n(\theta^0 - \theta^*)$$

X^TX 의 largest eigenvalue 를 ρ라 하자. (절대값이 최대인 수)

$$(\alpha X^TX)\mathbb{V} = (\alpha\rho)\mathbb{V} \;\Rightarrow\; (I - \alpha X^TX)\mathbb{V} = \mathbb{V} - \alpha\rho\mathbb{V} = (1 - \alpha\rho)\mathbb{V}$$

만약 $\alpha > \frac{2}{\rho}$ 라면, $|1 - \alpha\rho| > 1$ 이다 $I - \alpha X^TX$ 의 eigenvalue 중 하나의 절대값은 1을 초과함을 알 수 있다.

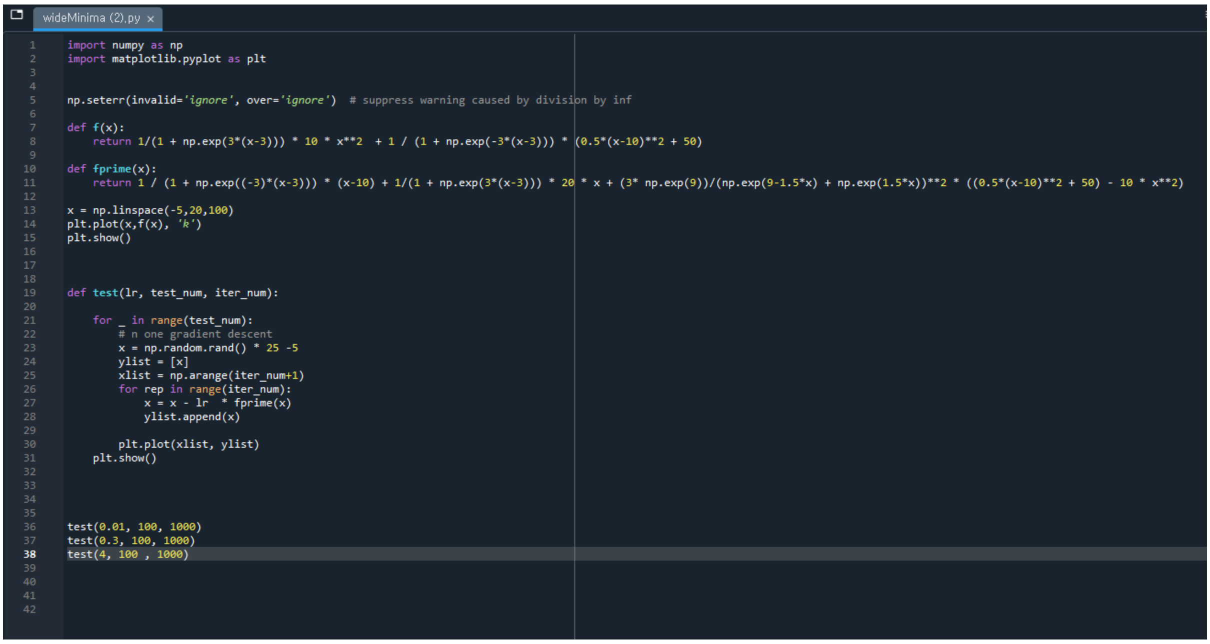
$I - \alpha X^TX$ 는 Real Symmetric 이므로 diagonalizable 하다. 즉 $I - \alpha X^TX = PDP^{-1}$.

$$\therefore (\theta^n - \theta^*) = (PD^nP^{-1})(\theta^0 - \theta^*) = P\begin{pmatrix} \lambda_1^n & & \\ & \lambda_2^n & \\ & & \ddots & \\ & & & \lambda_p^n \end{pmatrix}P^{-1}(\theta^0 - \theta^*)$$

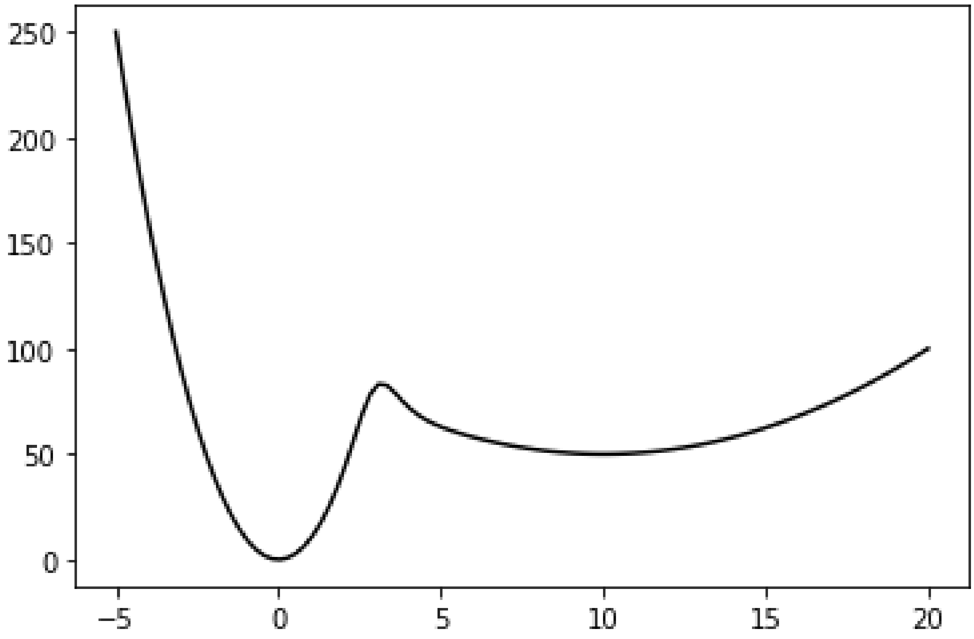
$\lambda^* = \max\{|\lambda_1|, \ldots, |\lambda_p|\} > 1$ 이므로 $(\lambda^*)^n \to \infty$ as $n \to \infty$ 이고

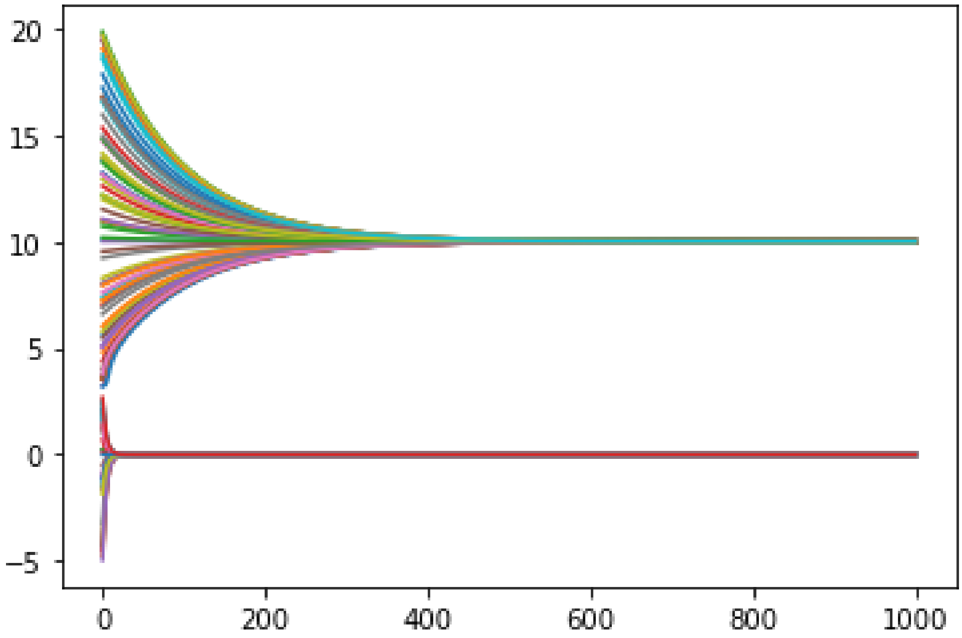
따라서 $\alpha > \frac{2}{\rho}$ 라면 대부분의 θ^0 에 대해 $\theta^n - \theta^*$ 가 발산, 즉 θ^n 이 발산한다.

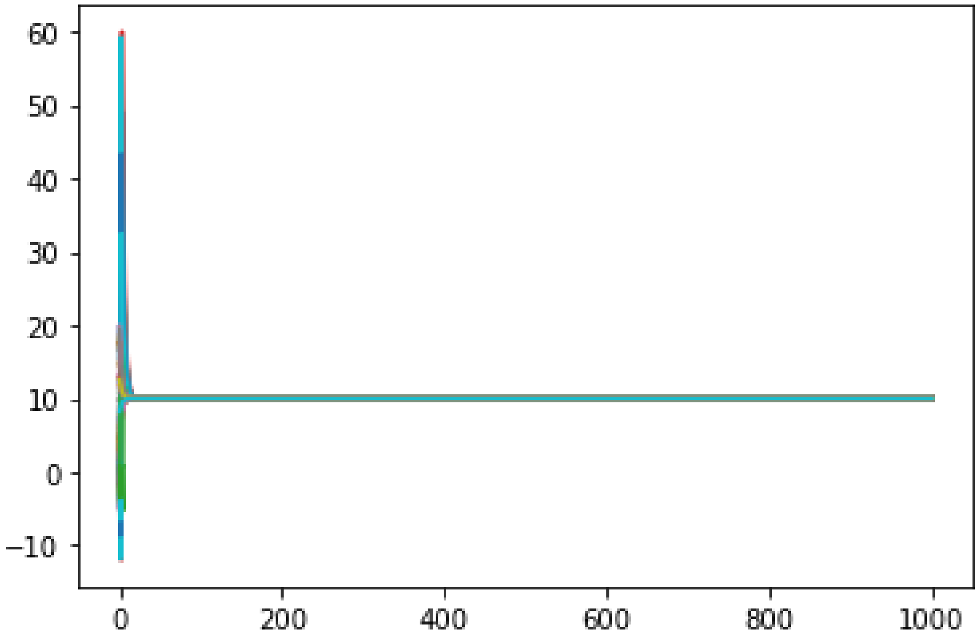
#4.

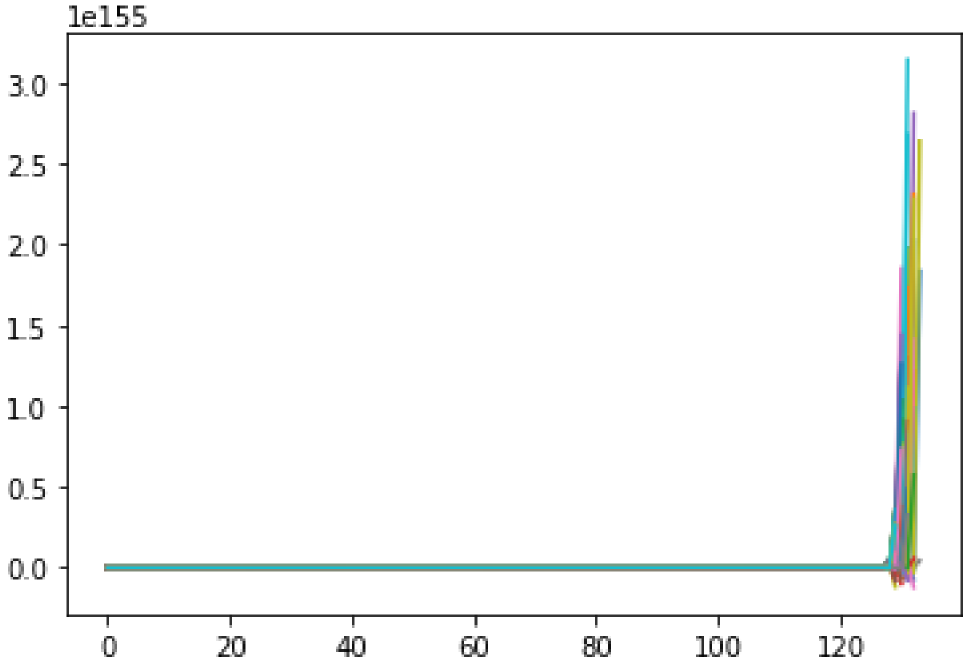


```python
import numpy as np
import matplotlib.pyplot as plt


np.seterr(invalid='ignore', over='ignore')  # suppress warning caused by division by inf

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)

x = np.linspace(-5,20,100)
plt.plot(x,f(x), 'k')
plt.show()



def test(lr, test_num, iter_num):

    for _ in range(test_num):
        # n one gradient descent
        x = np.random.rand() * 25 -5
        ylist = [x]
        xlist = np.arange(iter_num+1)
        for rep in range(iter_num):
            x = x - lr  * fprime(x)
            ylist.append(x)

        plt.plot(xlist, ylist)
    plt.show()




test(0.01, 100, 1000)
test(0.3, 100, 1000)
test(4, 100 , 1000)
```



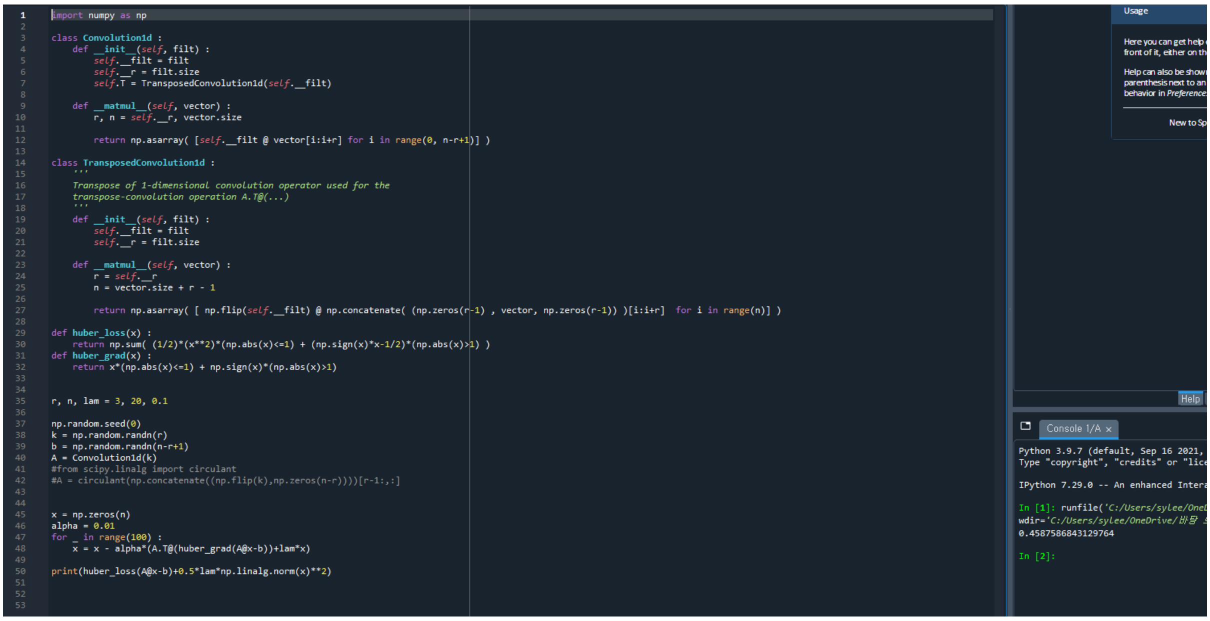

20
15
10
5
0
−5
0
200
400
600
800
1000


60
50
40
30
20
10
0
−10
0
200
400
600
800
1000



볼 수 있듯이, learning rate(lr)이 0.01 일때에는 sharp/wide minimum 중에 하나로 수렴하는 모습이 보여졌으며, lr이 0.3 일때는 wide minimum 으로 수렴하는 모습이 보여졌고, 마지막으로 lr이 4 일때는 발산하는 경향이 보여졌다.

#5.



```python
import numpy as np

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( [self.__filt @ vector[i:i+r] for i in range(0, 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( [ np.flip(self.__filt) @ np.concatenate( (np.zeros(r-1) , vector, np.zeros(r-1)) )[i:i+r]  for i in range(n)] )

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)
```

Here you can get help
front of it, either on th

Help can also be show
parenthesis next to an
behavior in *Preference*

New to Sp

```
Python 3.9.7 (default, Sep 16 2021,
Type "copyright", "credits" or "lice

IPython 7.29.0 -- An enhanced Intera

In [1]: runfile('C:/Users/sylee/OneD
wdir='C:/Users/sylee/OneDrive/바탕 
0.4587586843129764

In [2]:
```

위와 같이 slicing 과 concatenation을 적절히 사용하면 처리 가능하다. 출력으로 0.4587586.... 이 나오는 것을 확인할 수 있다. 코드를 살펴보면, Convolution1d의 경우 0을 하나도 사용하지 않고 처리한 것을 볼 수 있으며 Transpose의 경우 0을 $O(nr)$개 추가로 사용했음을 볼 수 있다. 이는 원래 행렬의 0의 개수인 $O(n^2)$ 개보다 적은 값이며 따라서 보다 효율적이다.