Assignment 1

#1 (a) $\nabla_\theta l_i(\theta) = \nabla_\theta \frac{1}{2}(x_i^T\theta - y_i)^2 = \left(\frac{\partial}{\partial\theta_1}\frac{1}{2}(x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{ip}\theta_p - y_i)^2, \cdots\right)^t$

$= (x_{i1}(x_{i1}\theta_1 + x_{i2}\theta_2 + \cdots + x_{ip}\theta_p - y_i), \cdots)^t$

$= (x_{i1}(x_i^T\theta - y_i), \cdots)^t$

$= (x_i^T\theta - y_i)x_i$

(b) $\nabla_\theta l(\theta) = \nabla_\theta \cdot \frac{1}{2}\|X\theta - y\|^2 = \nabla_\theta \cdot \frac{1}{2}\sum_{i=1}^n (x_i^T\theta - y_i)^2 = \sum_{i=1}^n (x_i^T\theta - y_i)x_i$

$= [x_1\ x_2\ \cdots\ x_N]\begin{pmatrix} x_1^T\theta - y_1 \\ \cdots \\ x_N^T\theta - y_N \end{pmatrix} = X^T(X\theta - y)$

#2 $\theta_{k+1} = \theta_k - \alpha f'(\theta_k)$, $f(\theta) = \frac{1}{2}\theta^2$ 이 convex, $\theta_0 \neq 0$, $\alpha > 2$ 일때 $\{\theta_k\}$ diverges

(pf) $\theta_{k+1} = \theta_k - \alpha\theta_k = (1-\alpha)\theta_k \Rightarrow |\theta_k| = |1-\alpha|^k|\theta_0|$, $|1-\alpha| > 1$ 이므로 θ_k diverges □

#3 $\theta_{k+1} = \theta_k - \alpha\nabla f(\theta_k)$, $f(\theta) = \frac{1}{2}\|X\theta - y\|^2$ 이 convex 일 때 a.e. $\forall \theta_0 \in \mathbb{R}^m$, $\alpha > 2/\rho(X^TX)$ 이면 $\{\theta_k\}$ diverges

(pf) $\nabla f(\theta) = X^T(X\theta - y)$ 이고, $X^TX \in \mathbb{R}^{p\times p}$ 는 symmetric matrix이므로 orthogonally diagonalizable한 matrix이자 positive matrix이므로, given eigenvalue가 모두 $\mathbb{R}_0^+$ 의 원소이고, 이를 크기순으로 $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ 이라고 하자.

한편 $f(\theta)$ 를 최소화하는 θ 는 $X^TX\theta = X^Ty$ 를 만족해야 하는데, 그 중 하나의 원소를 θ^* 라고 하자. 그러면,

$\exists U \in O(p)$ 를 만족하는 $U^TX^TXU = \mathrm{diag}(\lambda_1, \cdots, \lambda_p)$ 가 존재하고,

$I - \alpha X^TX \sim \mathrm{diag}(1-\alpha\lambda_1, \cdots, 1-\alpha\lambda_p)$ 임을 쉽게 확인할 수 있다.

$\theta_{k+1} = \theta_k - \alpha\nabla f(\theta_k) = \theta_k - \alpha X^T(X\theta_k - y) = \theta_k - \alpha X^TX\theta_k + \alpha X^Ty = \theta_k - \alpha X^TX\theta_k + \alpha X^TX\theta^*$

$\theta_{k+1} - \theta^* = (I - \alpha X^TX)(\theta_k - \theta^*) = (I - \alpha X^TX)^k(\theta_0 - \theta^*)$. 이때 (B)를 $[U]_1, \cdots, [U]_p$ 로 나타내자.

$[\theta_k - \theta^*]_B = \mathrm{diag}(1-\alpha\lambda_1, \cdots, 1-\alpha\lambda_p)^k[\theta_0 - \theta^*]_B$ (B) 좌표계에 대한

$\lim_{k\to\infty}(\theta_k - \theta^*)$ 가 수렴하려면 $\lim_{k\to\infty}[\theta_k - \theta^*]_B$ 가 수렴해야하는데, $[\theta_k - \theta^*]_B$ 의 1st comp. 의 값은 첫번째 좌표값

$(1-\alpha\lambda_1)^k[\theta_0 - \theta^*]_B$ 의 1st comp. 이다. $\lim_{k\to\infty}(1-\alpha\lambda_1)^k$ 를 생각하면, $\lim_{k\to\infty}|(1-\alpha\lambda_1)^k| = \infty$ 이므로

1st component가 수렴하려면 $[\theta_0 - \theta^*]_B$ 의 1st component가 0이어야 한다.

이는 a.e. 인 θ_0 에 대해 $\{\theta_k\}$ diverges. □

MATHDNN_1_4

September 9, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random as r
```

[2]:
```
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
```

[3]:
```
np.seterr(invalid='ignore', over='ignore')  # suppress warning caused by␣
↪division by inf
```

[3]:
```
{'divide': 'warn', 'invalid': 'warn', 'over': 'warn', 'under': 'ignore'}
```

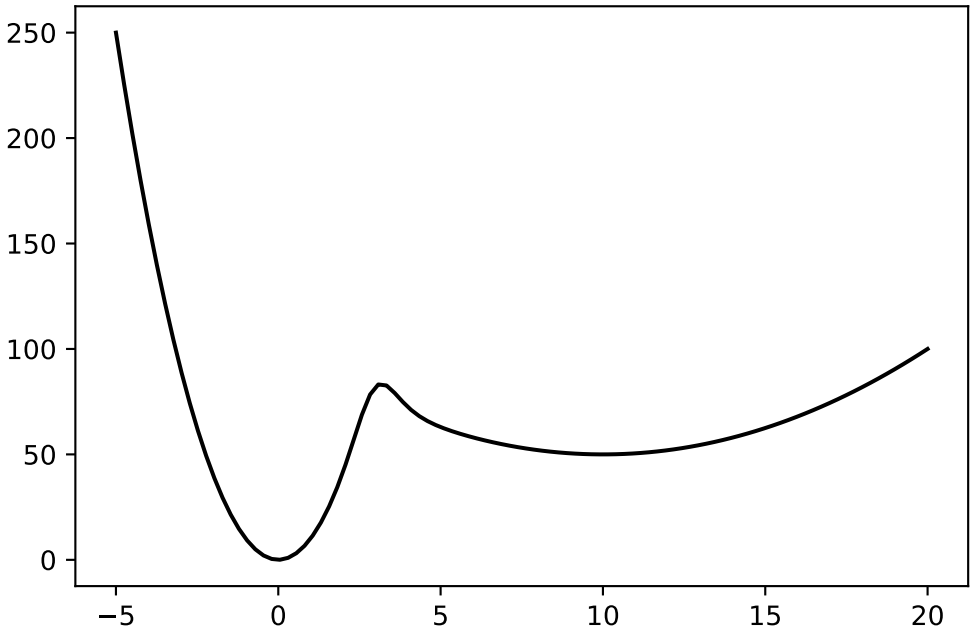
[4]:
```
def f(x):
    return 1/(1 + np.exp(3*(x-3))) * 10 * x**2  + 1 / (1 + np.exp(-3*(x-3))) *␣
↪(0.5*(x-10)**2 + 50)

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)
```

[5]:
```
def GD_plot(alpha,run_time,num_of_sample):
  rt=[i for i in range(1,run_time+1)]
  for _ in range(num_of_sample):
    # randomly generating samples
    x=[r.uniform(-5,20)]
    for j in range(run_time-1):
      x.append(x[-1]-alpha*fprime(x[-1]))
    plt.plot(rt,x)
  plt.show()
```

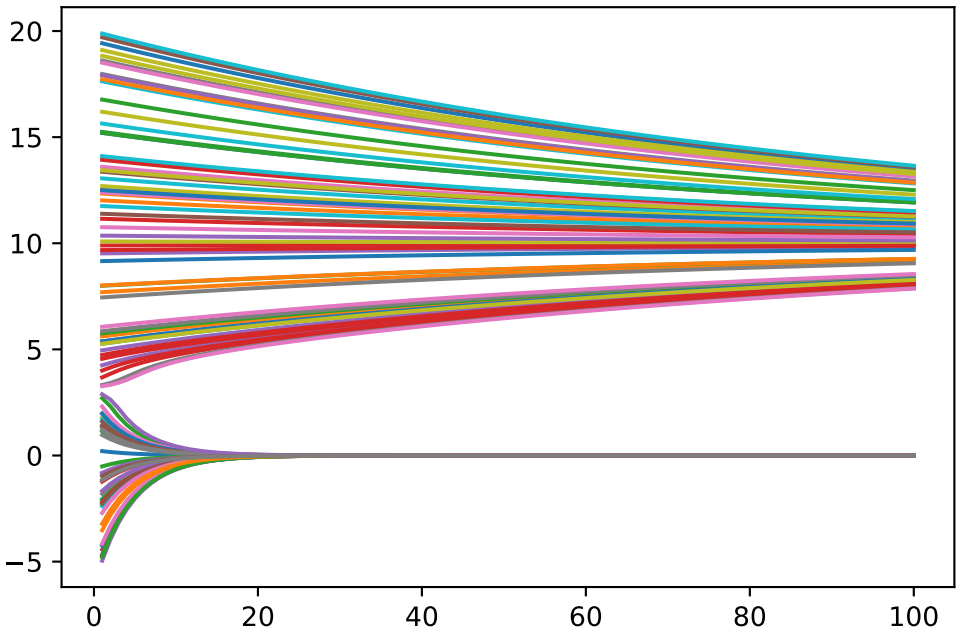
[6]:
```
def GD_freq(alpha,run_time,num_of_sample):

  num_0,num_10=0,0
  rt=[i for i in range(1,run_time+1)]
  for _ in range(num_of_sample):
    x=[r.uniform(-5,20)]
    for j in range(run_time-1):
      x.append(x[-1]-alpha*fprime(x[-1]))
    if 0.99*f(0)<=f(x[-1]) and f(x[-1])<=1.01*f(0):
```

```
      num_0+=1
    elif 0.99*f(10)<=f(x[-1]) and f(x[-1])<=1.01*f(10):
      num_10+=1
  return num_0/num_of_sample,num_10/num_of_sample
```

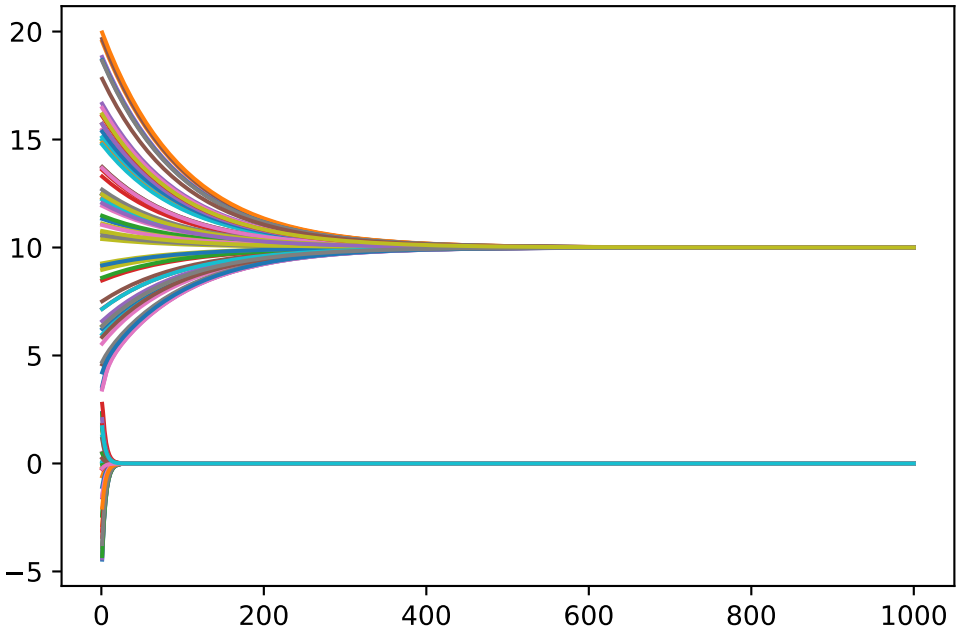
[7]:
```
x = np.linspace(-5,20,100)
plt.plot(x,f(x), 'k')
plt.show()
```



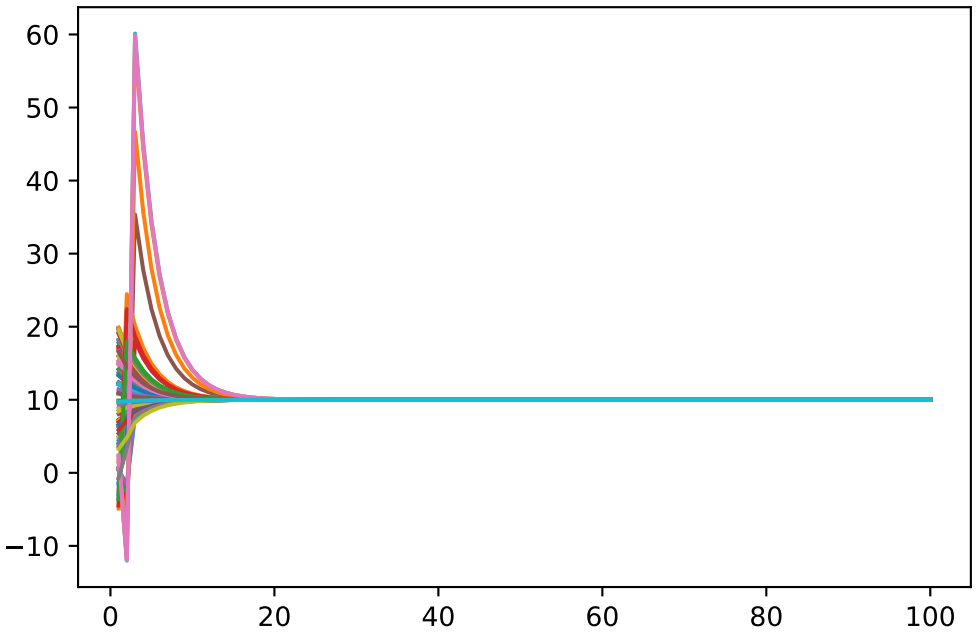
[8]:
```
GD_plot(0.01,100,100)
```



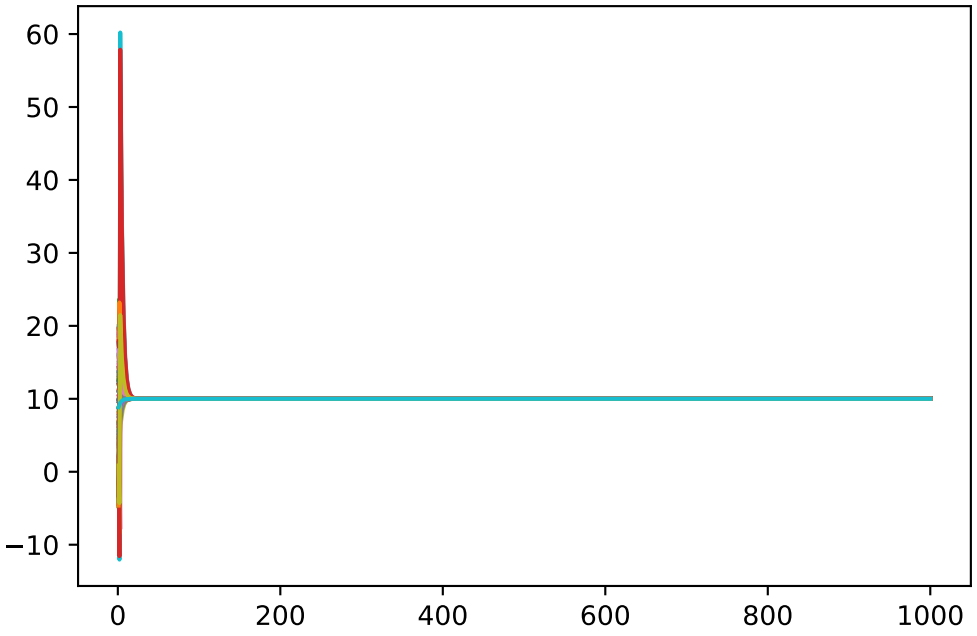
[9]:
```
GD_plot(0.01,1000,100)
```



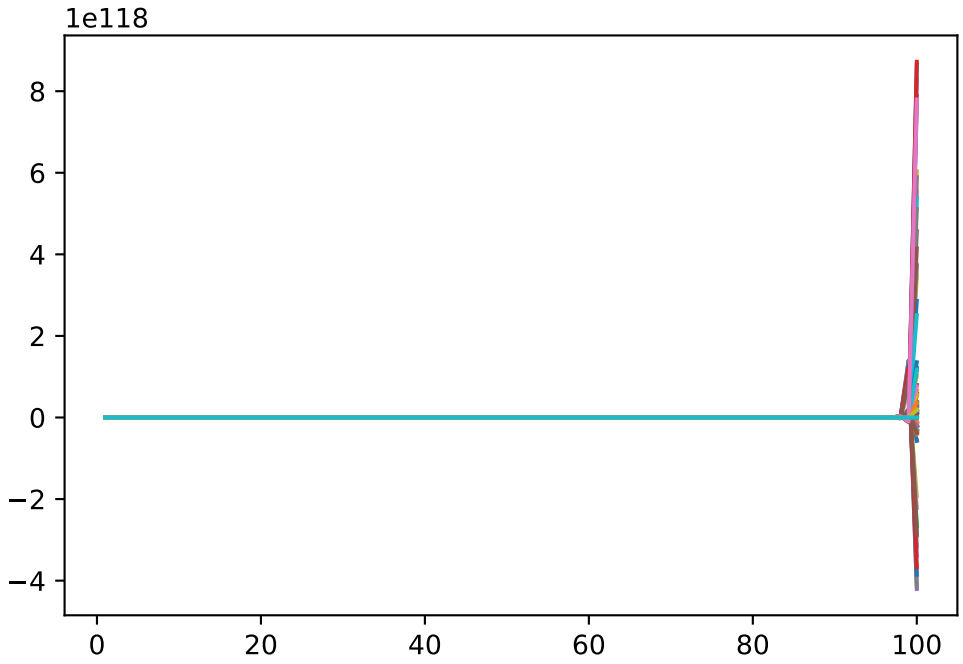
[10]:
```
GD_plot(0.3,100,100)
```



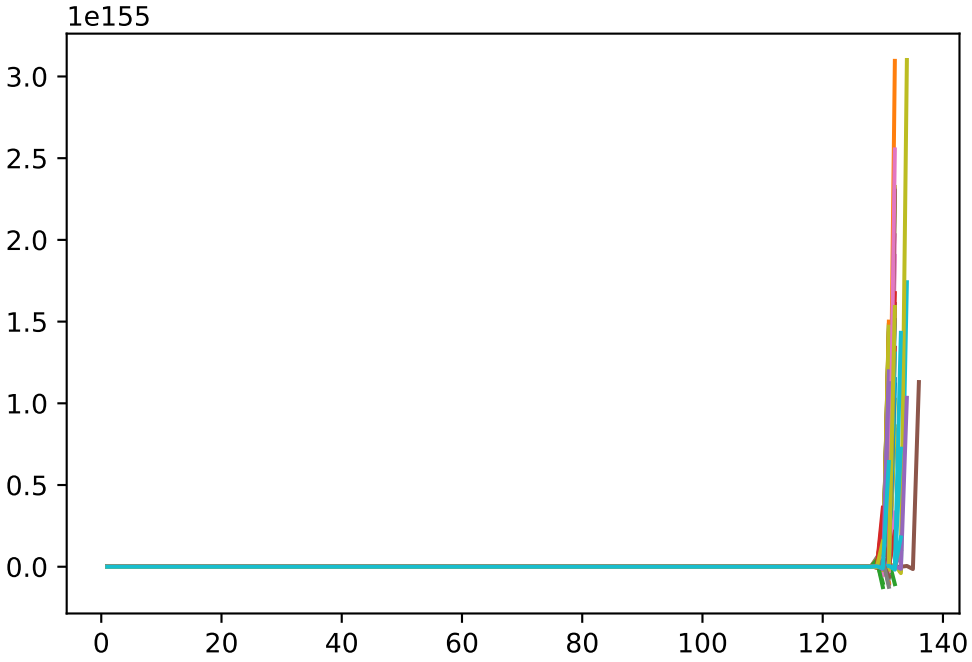
[11]:
```
GD_plot(0.3,1000,100)
```



[12]:
```
GD_plot(4,100,100)
```



[13]:
```
GD_plot(4,1000,100)
```



[15]:
```
#alpha=0.01
rt_1_to_0,rt_1_to_10=GD_freq(0.01,1000,300)
#alpha=0.3
rt_2_to_0,rt_2_to_10=GD_freq(0.3,1000,300)
#alpha=4
rt_3_div=1-sum(GD_freq(4,1000,300))
print(f"If rate is 0.01, approximately {round(100*rt_1_to_0,2)} percent of␣
 ↪samples converge to 0 and {round(100*rt_1_to_10,2)} percent of samples␣
 ↪converge to 10.")
print(f"If rate is 0.3, approximately {round(100*rt_2_to_0,2)} percent of␣
 ↪samples converge to 0 and {round(100*rt_2_to_10,2)} percent of samples␣
 ↪converge to 10.")
print(f"If rate is 4, approximately {round(100*rt_3_div,2)} percent of samples␣
 ↪diverge.")
```

```
If rate is 0.01, approximately 30.33 percent of samples converge to 0 and 69.67
percent of samples converge to 10.
If rate is 0.3, approximately 0.0 percent of samples converge to 0 and 100.0
percent of samples converge to 10.
If rate is 4, approximately 100.0 percent of samples diverge.
```

[16]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('MATHDNN_1_4.ipynb')
```

```
--2021-09-09 13:13:22--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-09-09 13:13:22 (39.3 MB/s) - colab_pdf.py saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

MATHDNN_1_5

September 9, 2021

[1]:
```
import numpy as np
```

[2]:
```
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
```

[3]:
```
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[k]*vector[k+i] for k in range(r)])␣
↪for i in range(n-r+1)])  # IMPLEMENT THIS
```

[4]:
```
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

    # Goal : Implementing matrix multiplication

  def __matmul__(self, vector) :
    r = self.__r
    n = vector.size + r - 1

    return np.asarray([sum([self.__filt[k]*vector[i-k] for k in range(0,i+1) if␣
↪k<=r-1 and i-k>=0 and i-k<=n-r]) for i in range(n)])  # IMPLEMENT THIS
```

[5]:
```
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)
```

[6]:
```
r, n, lam = 3, 20, 0.1

np.random.seed(0)
k = np.random.randn(r) # randn(r) : normalized seeds of r elements
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))))[2:,:]


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
```

[7]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('MATHDNN_1_5.ipynb')
```

```
--2021-09-09 13:16:07--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-09-09 13:16:07 (23.0 MB/s) - colab_pdf.py saved [1864/1864]
```

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```