MATHDNN_HW2

September 19, 2021

Problem 1,2,3,7 were solved in this pdf.

Problem 4,5,6 were solved in another file.

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

[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'}
```

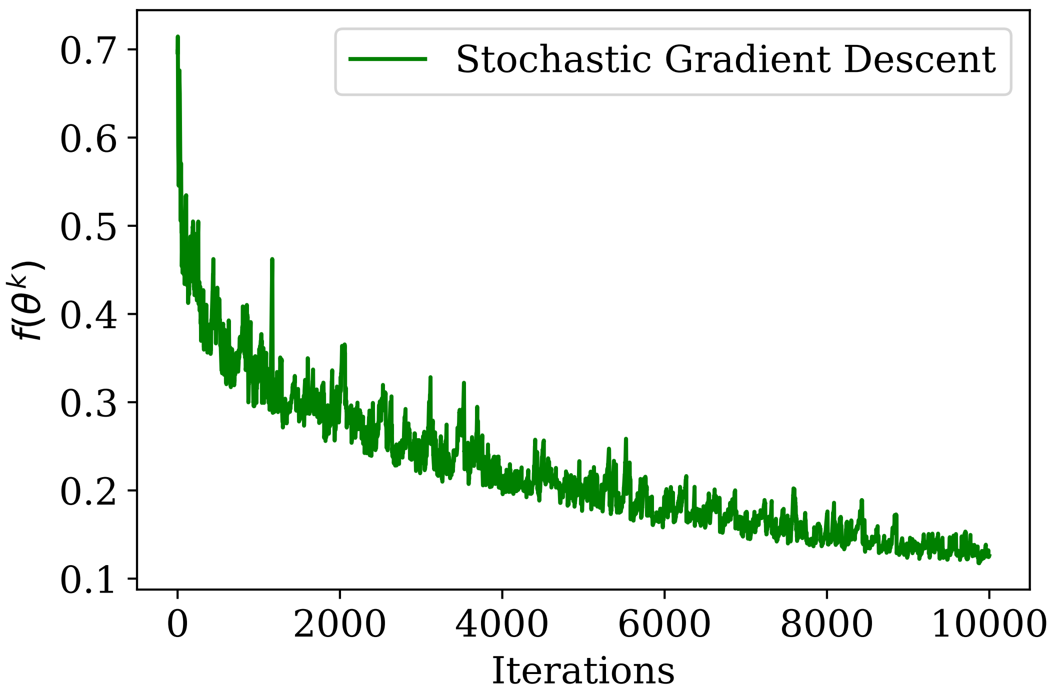
Problem 1

[4]:
```
N, p = 30, 20
np.random.seed(0)
X = np.random.randn(N,p) # generates N vectors of R^p; form of np.array
Y= 2*np.random.randint(2, size = N)-1 # 2*(0,1)-1= -1 or 1 ; generates N scalars
```

[5]:
```
theta = np.zeros(p)
alpha = 0.1  #0.1 is roughly the best value

K = 10000
f_val = []
for _ in range(K):
    ind = np.random.randint(N)
    theta -= alpha*(np.exp(-Y[ind]*X[ind,:]@theta)*(-Y[ind]))*X[ind,:]/(1+np.
 ↪exp(-Y[ind]*X[ind,:]@theta))
    f_val.append(N**(-1)*sum([np.log(1+np.exp(-Y[i]*X[i,:]@theta)) for i in␣
 ↪range(N)]))

print(f"Ans ) The minimizer value of theta is {theta}.")
```

```
plt.rc('text',usetex=False)
plt.rc('font',family='serif')
plt.rc('font', size = 14)
plt.plot(list(range(K)),f_val, color = "green", label = "Stochastic Gradient␣
 ↪Descent")
plt.xlabel('Iterations')
plt.ylabel(r'$f(\theta^k)$')
plt.legend()
plt.show()
```

```
Ans ) The minimizer value of theta is [-0.48651935  1.23818473  0.41953234
4.96438239 -1.58543155 -0.91604375
 -4.6043269  -2.67397853  1.16083962  2.06781531  4.87451857 -7.32035519
 -0.31449899 -2.10115613  4.67614609  5.37479007 -4.47990975  0.67006012
 -0.76432097 -7.14086801].
```

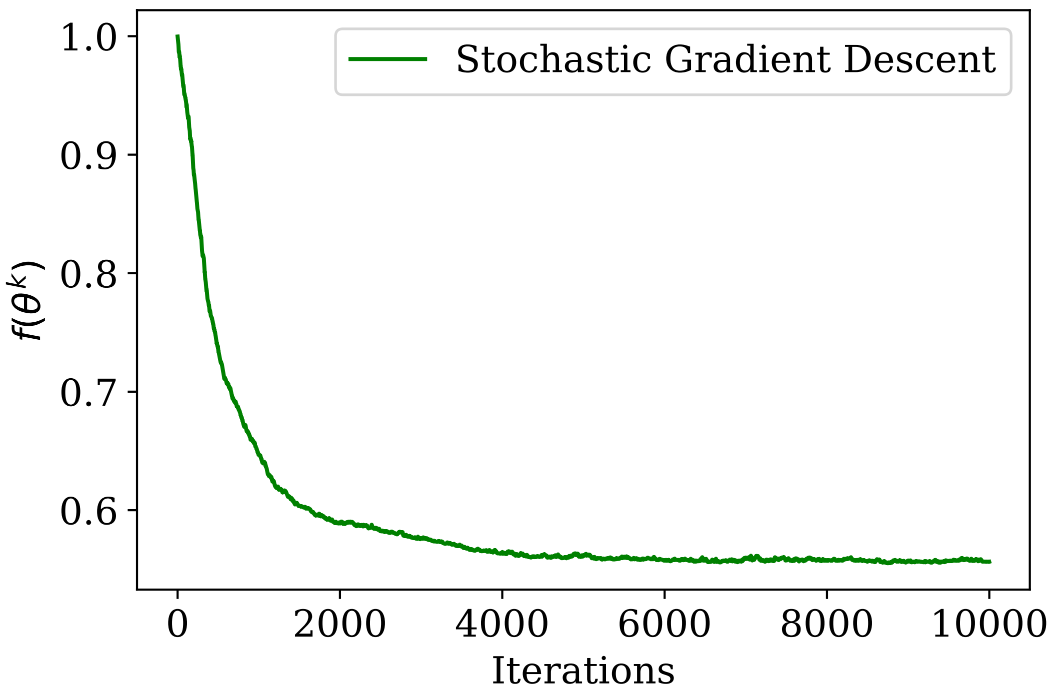


Problem 2

[6]:
```
# Using same data with previous problem

theta = np.zeros(p)
alpha = 0.001  #0.1 is roughly the best value
```

```
lb = 0.1

K = 10000
check = 0
f_val = []
for _ in range(K):
    ind = np.random.randint(N)
    if Y[ind]*X[ind,:]@theta>1:
      theta -= alpha*(lb*2*theta)
    elif Y[ind]*X[ind,:]@theta==1:
      check +=1
      theta -= alpha*(lb*2*theta - Y[ind]*X[ind,:])
    else:
      theta -= alpha*(lb*2*theta - Y[ind]*X[ind,:])
    f_val.append(lb*theta.T@theta+N**(-1)*sum([max(0,1-Y[i]*X[i,:]@theta) for i␣
 ↪in range(N)]))

if check>=1:
  print("Encountering non-differentiable point!")

print(theta)
plt.rc('text',usetex=False)
plt.rc('font',family='serif')
plt.rc('font', size = 14)
plt.plot(list(range(K)),f_val, color = "green", label = "Stochastic Gradient␣
 ↪Descent")
plt.xlabel('Iterations')
plt.ylabel(r'$f(\theta^k)$')
plt.legend()
plt.show()
```

```
[ 0.06872846  0.03260651 -0.32546856  0.17607469 -0.03492844 -0.15437353
 -0.30438171 -0.2029697   0.27944936 -0.14706277  0.08053559 -0.13851692
  0.12392779 -0.22784955  0.22965351  0.25477181 -0.35999374 -0.09447343
 -0.01532621 -0.47561883]
```



Ans) Encountering a point of non-differentiability didn't happen when I tried.

Problem 3

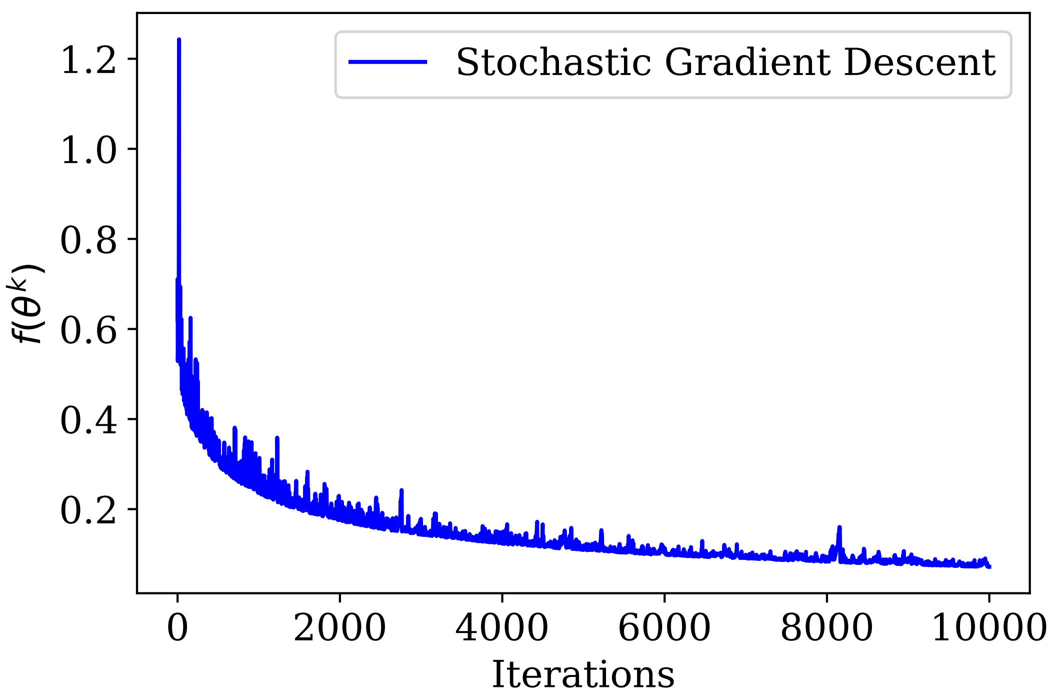
[7]:
```
N=30
np.random.seed(0)
X = np.random.randn(2,N)
y = np.sign(X[0,:]**2+X[1,:]**2-0.7)
theta = 0.5
c, s = np.cos(theta), np.sin(theta)
X = np.array([[c, -s], [s, c]])@X
X = X + np.array([[1],[1]]) # add 1,1 pointwise
```

[8]:
```
def phi(u,v):
  return np.array([1,u,u**2,v,v**2])
```

[9]:
```
import copy as c
datasets = [phi(X[0][i],X[1][i]) for i in range(N)]
X_=c.deepcopy(datasets)
Y_=c.deepcopy(y)
```

[10]:
```
# using the logistic regression
theta = np.zeros(5)
```

```
alpha = 0.1  #0.1 is roughly the best value

K = 10000
f_val = []
for _ in range(K):
    ind = np.random.randint(N)
    theta -= alpha*(np.exp(-Y_[ind]*X_[ind]@theta)*(-Y_[ind]))*X_[ind]/(1+np.
 ↪exp(-Y_[ind]*X_[ind]@theta))
    f_val.append(N**(-1)*sum([np.log(1+np.exp(-Y_[i]*X_[i]@theta)) for i in␣
 ↪range(N)]))

plt.rc('text',usetex=False)
plt.rc('font',family='serif')
plt.rc('font', size = 14)
plt.plot(list(range(K)),f_val, color = "blue", label = "Stochastic Gradient␣
 ↪Descent")
plt.xlabel('Iterations')
plt.ylabel(r'$f(\theta^k)$')
plt.legend()
plt.show()

print(theta)
```



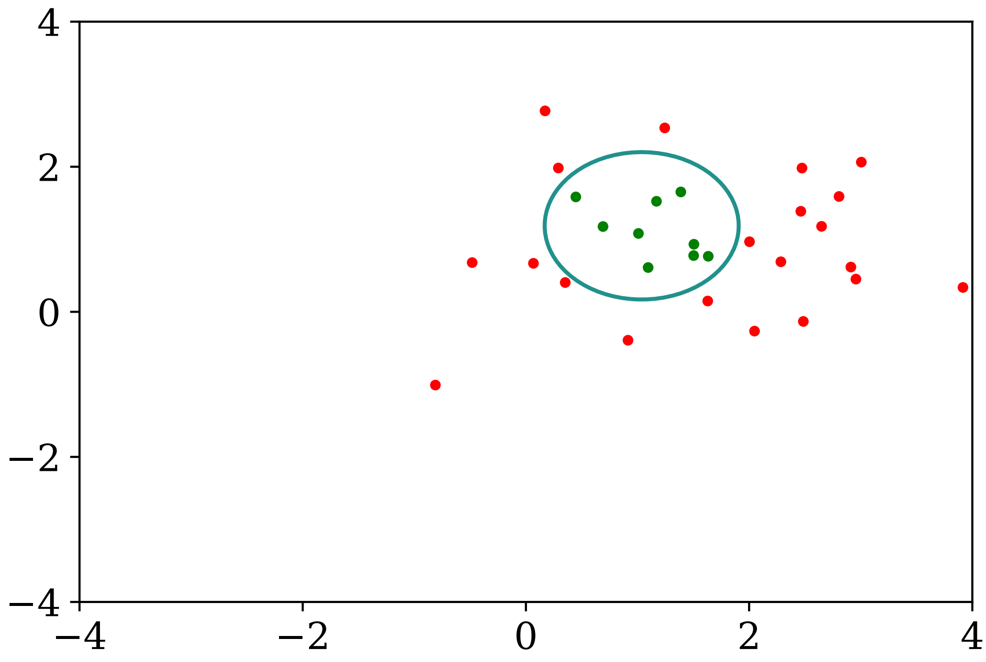

```
[  6.7173994  -10.32495424   4.9762752   -8.64245319   3.64384716]
```

[11]:

```python
xx = np.linspace(-4,4,1024)
yy = np.linspace(-4,4,1024)
xx,yy = np.meshgrid(xx,yy)
Z = theta@phi(xx,yy)
plt.contour(xx,yy,Z,0,color='blue')

pt_pos = [[X[0][i],X[1][i]] for i in range(N) if theta@phi(X[0][i],X[1][i])>0]
pt_neg = [[X[0][i],X[1][i]] for i in range(N) if theta@phi(X[0][i],X[1][i])<0]
pt_pos_x = [pt_pos[i][0] for i in range(len(pt_pos))]
pt_pos_y = [pt_pos[i][1] for i in range(len(pt_pos))]
pt_neg_x = [pt_neg[i][0] for i in range(len(pt_neg))]
pt_neg_y = [pt_neg[i][1] for i in range(len(pt_neg))]

plt.plot(pt_pos_x,pt_pos_y,'.',color="red")
plt.plot(pt_neg_x,pt_neg_y,'.',color="green")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning: The
following kwargs were not used by contour: 'color'
  """
```

[11]: [<matplotlib.lines.Line2D at 0x7f691e3e0b50>]



The purpose of using logistic regression is to classify the sign of each data.

If we minimise KL-divergence, then the dismatch of sign shall vanish.

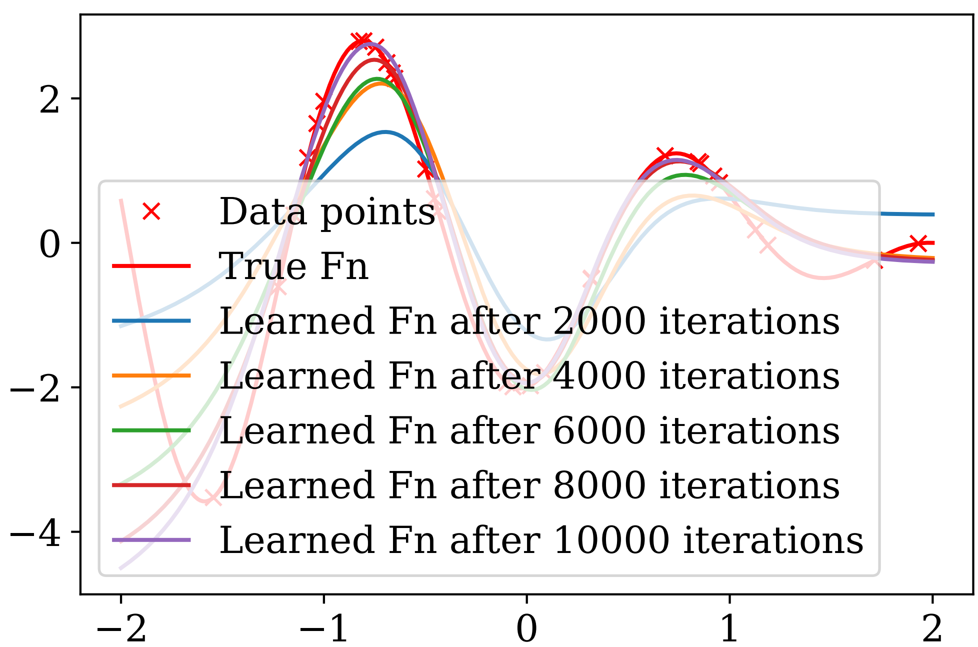
Problem 7

```
[12]: def f_true(x) :
          return (x-2)*np.cos(x*4)

      def sigmoid(x) :
          return 1 / (1 + np.exp(-x))

      def sigmoid_prime(x) :
          return sigmoid(x) * (1 - sigmoid(x))
```

```
[13]: K = 10000
      alpha = 0.007
      N, p = 30, 50
      np.random.seed(0)
      a0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
      b0 = np.random.normal(loc = 0.0, scale = 4.0, size = p)
      u0 = np.random.normal(loc = 0, scale = 0.05, size = p)
      theta = np.concatenate((a0,b0,u0))
```

```
[14]: X = np.random.normal(loc = 0.0, scale = 1.0, size = N) # given data
      Y = f_true(X) # given data

      def f_th(theta, x):
        # np.reshape(x,(-1,1)) : if x : [1,2,3,4,5,6,7,8] and (-1,1) -> [[1],[2],...
       ↪,[8]] : 8/1,1
        # if (-1,2) -> [[1,2],...,[7,8]]
        # theta[2*p : 3*p] * sigmoid(theta[0 : p] * np.reshape(x,(-1,1)) + theta[p :␣
       ↪2*p])
        # A return value is an array. It's because x is formed as linspace, so it␣
       ↪approaches the elements one by one.
        # (axis=0 means approaching values by a_j,b_j,u_j)
        return np.sum(theta[2*p : 3*p] * sigmoid(theta[0 : p] * np.reshape(x,(-1,1))␣
       ↪+ theta[p : 2*p]), axis=1)

      # def diff_f_th(theta, x) :
      # pass
```

```python
def f_th_oneelem(theta,x):
  return np.sum(theta[2*p : 3*p] * sigmoid(theta[0 : p] * x + theta[p : 2*p]))

def grad_of_loss(theta,x,y):
  u_ = [sigmoid(theta[i]*x+theta[p+i]) for i in range(p)]
  b_ = [sigmoid_prime(theta[i]*x+theta[p+i])*theta[2*p+i] for i in range(p)]
  a_ = [sigmoid_prime(theta[i]*x+theta[p+i])*theta[2*p+i]*x for i in range(p)]
  return ((f_th_oneelem(theta,x))-y) * np.concatenate((a_,b_,u_),axis=None)

xx = np.linspace(-2,2,1024) # training data
plt.plot(X,f_true(X),'rx',label='Data points')
plt.plot(xx,f_true(xx),'r',label='True Fn')

for k in range(K):
  ind = np.random.randint(N)
  theta -= alpha * grad_of_loss(theta,X[ind],Y[ind])
  if (k+1)%2000 == 0 :
    plt.plot(xx,f_th(theta, xx),label=f'Learned Fn after {k+1} iterations')

plt.legend()
plt.show()
plt.savefig('plot.png')
```




```
<Figure size 432x288 with 0 Axes>
```

Ans) As shown in the picture above, increasing the number of trials gradually leads to a real function, like the convergence of Taylor series.

과제 2 심층신경망의 수학적 기초

공과대학 산업공학과 산업공학전공
2020-10007 **김형윤**

2021.9.19

Problem 4 (KL-divergence의 nonnegativity) 집합 $C \subseteq \mathbb{R}^m$가 볼록집합인 것은

$$x_1, x_2 \in C \Rightarrow \theta x_1 + (1-\theta)x_2 \in C \quad \forall\, \theta \in (0,1)$$

라고 하자. 또, 함수 $\varphi : C \to \mathbb{R}$이 볼록함수인 것은

$$\varphi(\theta x_1 + (1-\theta)x_2 \leq \theta\varphi(x_1) + (1-\theta)\varphi(x_2) \quad \forall\, x_1, x_2 \in C,\ \theta \in (0,1)$$

이라고 하자. 한편, $X \in C$가 확률변수이고 φ가 볼록함수이면,

$$\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$$

이 성립하고 이를 옌센 부등식(Jensen's Inequality)라고 한다. 위의 사실을 이용하여,

$$D_{KL}(p\|q) \geq 0$$

이 모든 확률질량함수 $p, q \in \mathbb{R}^n$에 대해 성립함을 보여라.

Proof.
먼저 $-\log x$는 두 번 미분한 함수 $\frac{1}{x^2}$가 항상 양수이므로 볼록함수이다. 따라서 옌센 부등식을 적용할 수 있다. 한편 p, q가 확률질량함수이므로 $\sum_{i=1}^N p_i = 1, \sum_{i=1}^N q_i = 1$이 성립한다. 이제

$$\begin{aligned} D_{KL}(p\|q) &= \sum_{i=1}^N p_i \log \frac{p_i}{q_i} = \sum_{i=1}^N p_i(-\log \frac{q_i}{p_i}) \\ &\geq \sum_{i=1}^N (-\log(p_i \frac{q_i}{p_i}) \ (\because -\log : convex) \\ &= \sum_{i=1}^N (-\log q_i) = 0 \end{aligned}$$

이므로 KL-divergence는 항상 음이 아닌 실수 값을 가진다. 만약, q_i에 0인 값이 존재한다면, 대응하는 p_i가 0이 아니면 KL-divergence는 반드시 무한대의 값을 가지고, $p_i = 0$이면 $\{p_n\}, \{q_n\}$ 중 0이 아닌 값들에 대해서만 다시 위의 부등식을 생각하더라도 $\{p_i\}, \{q_i\}$의 합이 각각 1이므로 마찬가지로 음이 아닌 실수 값을 가진다. □

Problem 5 (KL-divergence의 positivity) 함수 $\varphi : C \to \mathbb{R}$이 strictly convex인 것은

$$\varphi(\theta x_1 + (1-\theta)x_2 < \theta\varphi(x_1) + (1-\theta)\varphi(x_2) \quad \forall\, x_1, x_2 \in C,\ \theta \in (0,1)$$

이라고 하자. 한편, $X \in C$가 상수가 아닌 확률변수(i.e. not a uniform r.v)이고 φ가 strictly convex이면,

$$\varphi(\mathbb{E}[X]) < \mathbb{E}[\varphi(X)]$$

이 성립하고 이를 Strict Jensen's Inequality라고 한다. 위의 사실을 이용하여,

$$D_{KL}(p\|q) > 0$$

이 $p \neq q$인 모든 확률질량함수 $p, q \in \mathbb{R}^n$에 대해 성립함을 보여라.

Proof.
먼저 $-\log x$는 두 번 미분한 함수 $\frac{1}{x^2}$가 항상 양수이므로 strictly convex이다. 따라서 Strict Jensen's Inequality를 적용할 수 있다.
한편 p, q가 확률질량함수이므로 $\sum_{i=1}^N p_i = 1, \sum_{i=1}^N q_i = 1$이 성립한다. 이제

$$\begin{aligned} D_{KL}(p\|q) &= \sum_{i=1}^N p_i \log\frac{p_i}{q_i} = \sum_{i=1}^N p_i(-\log\frac{q_i}{p_i}) \\ &> \sum_{i=1}^N (-\log(p_i\frac{q_i}{p_i})\ (\because -\log : strictly\, convex) \\ &= \sum_{i=1}^N (-\log q_i) = 0 \end{aligned}$$

이므로 KL-divergence는 항상 양의 실수 값을 가진다. 위의 부등식에서 등호가 성립하지 않는 이유는 등호가 성립하려면 모든 $\frac{q_i}{p_i}$의 값이 같아야 하고, 한편 $\sum_{i=1}^N p_i = 1, \sum_{i=1}^N q_i = 1$이 성립함에서 결과적으로 등호가 성립할 조건은 $p = q$이기 때문이다. 만약, q_i에 0인 값이 존재한다면, 대응하는 p_i가 0이 아니면 KL-divergence는 반드시 무한대의 값을 가지고, $p_i = 0$이면 $\{p_n\}, \{q_n\}$ 중 0이 아닌 값들에 대해서만 다시 위의 부등식을 생각하더라도 $\{p_i\}, \{q_i\}$의 합이 각각 1이므로 마찬가지로 양의 실수 값을 가진다. □

Problem 6 (2층 신경망의 미분) 미분가능한 함수 $\sigma : \mathbb{R} \to \mathbb{R}$와 벡터 $a, b, u \in \mathbb{R}^p$, 그리고 이들 벡터로 조합된 $\theta = (a_1, ..., a_p, b_1, ..., b_p, u_1, ..., u_p) \in \mathbb{R}^{3p}$가 주어졌을 때, 다음과 같은 2층 신경망

$$f_\theta(x) = u^T \sigma(ax + b) = \sum_{j=1}^{p} u_j \sigma(a_j x + b_j)$$

을 생각하자. 또, $\sigma(ax+b)$와 $\sigma'(ax+b)$는 성분별로 σ와 σ'을 취한 벡터라고 하고, $\odot$는 두 벡터를 성분별로 곱하여 새로운 벡터를 생성하는 이항연산이라고 하자. 이 때 아래의 관계가 성립함을 보여라.

$$\begin{aligned} \nabla_u f_\theta(x) &= \sigma(ax+b) \\ \nabla_b f_\theta(x) &= \sigma'(ax+b) \odot u = \mathrm{diag}(\sigma'(ax+b))u \\ \nabla_a f_\theta(x) &= (\sigma'(ax+b) \odot u)x = \mathrm{diag}(\sigma'(ax+b))ux \end{aligned}$$

Proof.
$\frac{\partial f_\theta(x)}{\partial u_i} = \frac{\partial}{\partial u_i} \sum_{j=1}^{p} u_j \sigma(a_j x + b_j) = \sum_{j=1}^{p} \frac{\partial}{\partial u_i} u_j \sigma(a_j x + b_j) = \sigma(a_i x + b_i)$
따라서, $\nabla_u f_\theta(x) = (\frac{\partial f_\theta(x)}{\partial u_1}, ..., \frac{\partial f_\theta(x)}{\partial u_p}) = (\sigma(a_1 x + b_1), ..., \sigma(a_p x + b_p)) = \sigma(ax+b)$

$\frac{\partial f_\theta(x)}{\partial b_i} = \frac{\partial}{\partial b_i} \sum_{j=1}^{p} u_j \sigma(a_j x + b_j) = \sum_{j=1}^{p} u_j \frac{\partial}{\partial b_i} \sigma(a_j x + b_j) = u_i \sigma'(a_i x + b_i)$
따라서, $\nabla_b f_\theta(x) = (\frac{\partial f_\theta(x)}{\partial b_1}, ..., \frac{\partial f_\theta(x)}{\partial b_p}) = (u_1 \sigma'(a_1 x + b_1), ..., u_p \sigma'(a_p x + b_p)) = (\sigma'(a_1 x + b_1), ..., \sigma'(a_p x + b_p)) \odot (u_1, ..., u_p) = \mathrm{diag}(\sigma'(ax+b))u$

$\frac{\partial f_\theta(x)}{\partial a_i} = \frac{\partial}{\partial a_i} \sum_{j=1}^{p} u_j \sigma(a_j x + b_j) = \sum_{j=1}^{p} u_j \frac{\partial}{\partial a_i} \sigma(a_j x + b_j) = u_i x \sigma'(a_i x + b_i)$
따라서, $\nabla_a f_\theta(x) = (\frac{\partial f_\theta(x)}{\partial a_1}, ..., \frac{\partial f_\theta(x)}{\partial a_p}) = (a_1 x \sigma'(a_1 x + b_1), ..., u_p x \sigma'(a_p x + b_p)) = ((\sigma'(a_1 x + b_1), ..., \sigma'(a_p x + b_p)) \odot (u_1, ..., u_p))x = \mathrm{diag}(\sigma'(ax+b))ux$ □