ECE595 HW4

Ruijie Song

Mar.18.2021

Ex1.

ECE 595 HW4

Ex 1

i) $h_\theta(x) = \frac{1}{1+\exp(-\theta^T x)} = \frac{1}{1+\exp(-(\underline{w}^T \underline{x} + w_0))}$

$h(x) = \begin{cases} 1 & g(x) > 0 \quad \cancel{x \to \infty} \\ 0 & g(x) < 0 \quad \cancel{x \to -\infty} \\ \text{either} & g(x) = 0 \end{cases}$

For $h(x) = 1$ or 0, $g(x)$ has to $\to \infty$ or $-\infty$

if $x = 0$, $w^T x + w_0 \to \infty \Rightarrow |w_0| \to \infty$

if $x \neq 0$, $w^T x \leq \frac{1}{2}(w^T w + x^T x)$ since: $(\underline{w} + \underline{x})^2 \geq 0$.

$w^T x \to \infty$ to satify $\theta^T x \to \infty$

$\therefore \|w\|_2 \to \infty$

ii) when $\|w\|_2 \to \infty$ & $|w_0| \to \infty$, $\Rightarrow (h_\theta(x_n) - y_n) \to 0$

Then $\|\theta^{(k+1)} - \theta^{(k)}\|_2 \Rightarrow 0$

But GD needs ∞ steps to ~~make~~ make $\|\theta^{(k+1)} - \theta^{(k)}\|_2 = 0$

iii) $\|w\|_2 \leq c_1$, $|w_0| < c_2$ for $c_1, c_2 > 0$

One way: $\|\theta^{(k+1)} - \theta^{(k)}\|_2 \leq$ Const.

Another way: Change α_k

iv) No. Linear regression.

Ex.2

Ex 2.

$$J(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \{ y_n \log h_\theta(x_n) + (1-y_n)\log(1-h_\theta(x_n)) \}$$

$$h_\theta(x) = \frac{1}{1+\exp\{-\theta^T x\}} \qquad J = \sum_{n=1}^{N} -\frac{1}{N}\left(y_n \log h_\theta(x_n) + (1-y_n)\log(1-h_\theta(x_n))\right)$$

$y_n \log h_\theta(x_n)$ ~~covex~~ convex

$$\nabla_\theta[-\log(1-h_\theta(x_n))] = \nabla_\theta \left[-\log\left(1-\frac{1}{1+\exp(-\theta^T x)}\right)\right)$$

$$= \nabla_\theta\left[-\log\left(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}\right)\right] = \nabla_\theta\left(-\log(e^{-\theta^T x}) + \log(1+e^{-\theta^T x})\right)$$

$$= \nabla_\theta\left(+\theta^T x + \log(1+e^{-\theta^T x})\right) = +x + \nabla_\theta \log(1+e^{-\theta^T x})$$

$$= +x + \left(\frac{-e^{-\theta^T x}}{1+e^{-\theta^T x}}\right)x = +h_\theta(x)x$$

Hessian: $\nabla_\theta^2[-(1-y_n)\log(1-h_\theta(x_n))] = \nabla_\theta[+h_\theta(x)x]\cdot(1-y_n)$

$$= +\nabla_\theta\left[\left(\frac{+1}{1+e^{-\theta^T x}}\right)x\right] = +\left(\frac{1}{(1+e^{-\theta^T x})^2}\right)(-e^{-\theta^T x})xx^T$$

$$= +\left(\frac{1}{1+e^{-\theta^T x}}\right)\left(1-\frac{1}{1+e^{-\theta^T x}}\right)xx^T = +h_\theta(x)[1-h_\theta(x)]xx^T \cdot (1-y_n)$$

For $v \in R^d$

$\cancel{v^T\nabla_\theta^2[-\log(1-h_\theta(x))]v = (h_\theta(x)[1-h_\theta(x)])\|v^Tx\|^2 \geq 0}$

$\because$ $\cancel{(1-y_n)\log(1-h}$ $\quad (1-y_n) \geq 0$

$$v^T\nabla_\theta^2[-\log(1-h_\theta(x_n))] = h_\theta(x)[1-h_\theta(x)])\|v^Tx\|^2 \geq 0$$

$\therefore$ Hessian is positive semi-definite.

$-(1-y_n)\log(1-h_\theta(x_n))$ is convex in θ.

Ex3.

a.

3.

a) $$J(\theta) = \frac{-1}{N}\sum_{n=1}^{N}\{y_n \log h_{\underline{\theta}}(\underline{x}_n) + (1-y_n)\log(1-h_{\underline{\theta}}(\underline{x}_n))\}$$

$$= \frac{-1}{N}\sum_{n=1}^{N}\{y_n \log h_{\underline{\theta}}(\underline{x}_n) + \log(1-h_{\underline{\theta}}(\underline{x}_n)) - y_n \log(1-h_{\underline{\theta}}(\underline{x}_n))\}$$

$$= \frac{-1}{N}\sum^{N}\{y_n \log\left(\frac{h_{\underline{\theta}}(\underline{x}_n)}{1-h_{\underline{\theta}}(\underline{x}_n)}\right) + \log(1-h_{\underline{\theta}}(\underline{x}_n))\}$$

$$= \frac{-1}{N}\sum\{y_n \underline{\theta}^T \underline{x}_n + \log(1-h_{\underline{\theta}}(\underline{x}_n))\}$$

$$= \frac{-1}{N}\sum\{y_n \underline{\theta}^T \underline{x}_n + \log\left(1-\frac{1}{1+e^{-\underline{\theta}^T \underline{x}_n}}\right)\}$$

$$= \frac{-1}{N}\sum\{y_n \underline{\theta}^T \underline{x}_n + \log\left(\frac{e^{-\underline{\theta}^T \underline{x}_n}}{1+e^{-\underline{\theta}^T \underline{x}_n}}\right)\}$$

$$= \frac{-1}{N}\sum\{y_n \underline{\theta}^T \underline{x}_n + \log\left(\frac{1}{1+e^{\underline{\theta}^T \underline{x}_n}}\right)\}$$

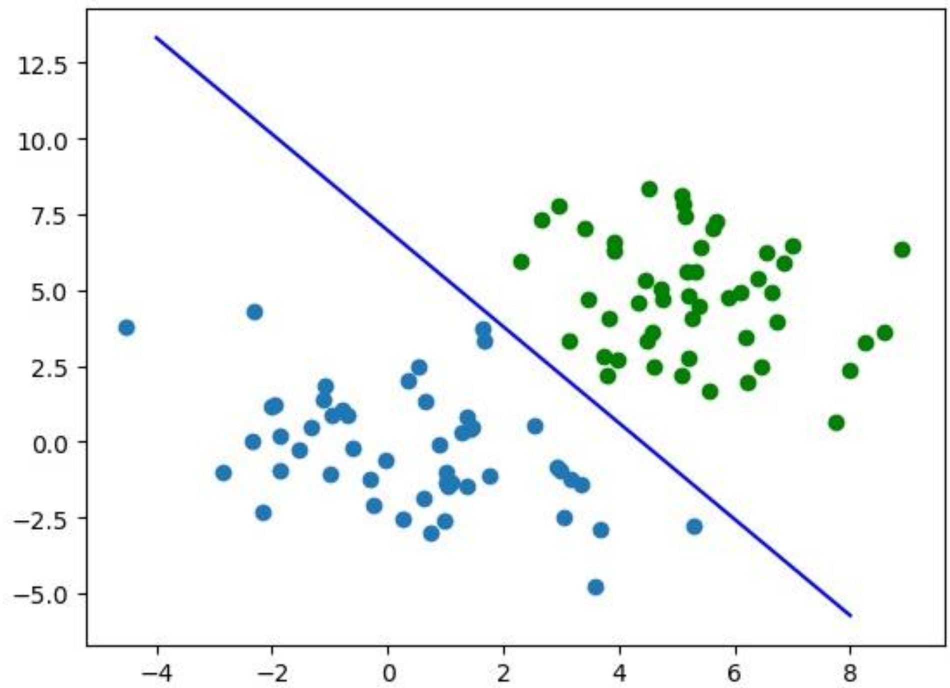
$$= \frac{-1}{N}\sum\{y_n \underline{\theta}^T \underline{x}_n - \log(1+e^{\underline{\theta}^T \underline{x}_n})\}$$

$$= \frac{-1}{N}\cdot\{(\sum y_n \underline{x}_n)^T \underline{\theta} - \sum \log(1+e^{\underline{\theta}^T \underline{x}_n})\}$$

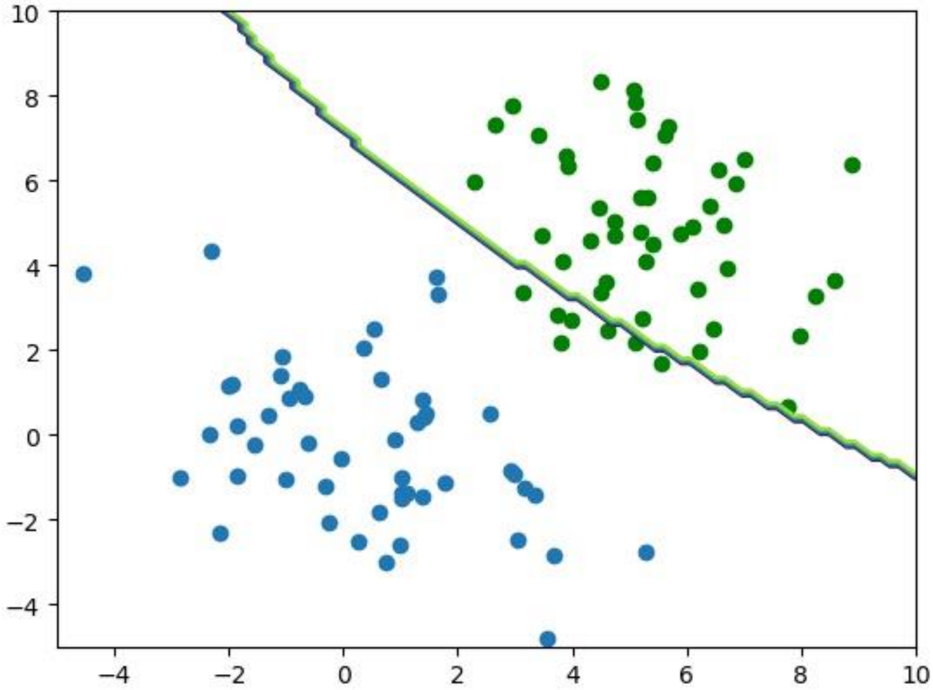
b.

theta = [2.38, 1.50, -10.44]

c.



d.



Ex4.

a.

[[1.00000000e+00 5.78667187e-13 6.19586231e-20 7.94511151e-25
 4.31787909e-17]
 [5.78667187e-13 1.00000000e+00 2.66618165e-12 2.63683326e-31
 1.07334027e-11]
 [6.19586231e-20 2.66618165e-12 1.00000000e+00 3.36906580e-42
 1.49290933e-18]
 [7.94511151e-25 2.63683326e-31 3.36906580e-42 1.00000000e+00
 1.20563730e-05]
 [4.31787909e-17 1.07334027e-11 1.49290933e-18 1.20563730e-05
 1.00000000e+00]]

b.

4 b) $\theta = \sum_{n=1}^{N} \alpha_n x_n$, $\theta^T x = \sum \alpha_n \langle x_n, x \rangle$

$$J(\theta) = \frac{-1}{N} \left\{ \left(\sum y_n x_n\right)^T \left(\sum \alpha_m x_m\right) - \sum \log\left(1 + e^{x_n^T (\sum \alpha_m x_m)}\right) \right\} + \lambda \|\sum \alpha_n x_n\|^2$$

$$= \frac{-1}{N} \left\{ \sum_{n=1}^{N} y_n \Big(\sum \alpha_m \underbrace{x_n^T x_m}_{\Downarrow\ K(x_n, x_m)}\Big) - \sum \log\Big(1 + e^{\sum \alpha_m \underbrace{(x_n^T x_m)}_{\Downarrow\ K(x_n, x_m)}}\Big) \right\} + \lambda \|\sum \alpha_n x_n\|^2$$

$\therefore \sum \alpha_m K(x_n, x_m)$

$= K_n^T \alpha$

$$= \frac{-1}{N} \left\{ \sum y_n K_n^T \alpha - \sum \log\left(1 + e^{K_n^T \alpha}\right) \right\} + \lambda \alpha^T x^T x \alpha$$

$$= \frac{-1}{N} \left\{ y^T K \alpha - 1^T \left(\log\left(1 + e^{K\alpha}\right)\right) \right\} + \lambda \alpha^T K \alpha$$

$$= \frac{-1}{N} \left\{ y^T K \alpha - 1^T \log\left(e^0 + e^{K\alpha}\right) \right\} + \lambda \alpha^T K \alpha$$

c.

[-0.52, -0.60]

d.

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 19 01:21:35 2021

@author: 11327
"""

import numpy as np
import matplotlib.pyplot as plt
import cvxpy as cvx
import scipy

# Ex 3
# b
# read data from txt file
xclass0 = np.matrix(np.loadtxt('./data/homework4_class0.txt'))
xclass1 = np.matrix(np.loadtxt('./data/homework4_class1.txt'))

# create x
x = np.concatenate((xclass0,xclass1),axis=0)
[rowx,colx] = np.shape(x)
x = np.concatenate((x,np.ones((rowx,1))),axis=1)

# create y
[rowx0,colx0] = np.shape(xclass0)
[rowx1,colx1] = np.shape(xclass1)
y0 = np.zeros((rowx0,1))
y1 = np.ones((rowx1,1))
y = np.concatenate((y0,y1),axis=0)

# CVX
lambd = 0.0001
N = rowx0 + rowx1
theta       = cvx.Variable((3,1))
loss        = - cvx.sum(cvx.multiply(y, x @ theta)) \
              + cvx.sum(cvx.log_sum_exp( cvx.hstack([np.zeros((N,1)), x @ theta]), axis=1 ) )
reg         = cvx.sum_squares(theta)
prob        = cvx.Problem(cvx.Minimize(loss/N + lambd*reg))
prob.solve()
w = theta.value

# c
# calculate the boundary
xb = np.linspace(-4,8,100)
yb = (-w[0]*xb-w[2])/w[1]

# do the plot
'''
plt.figure()
plt.scatter(xclass0[:,0].tolist(),xclass0[:,1].tolist())
plt.scatter(xclass1[:,0].tolist(),xclass1[:,1].tolist(), c='g')

plt.plot(xb,yb,c='b')
plt.show()
'''
# d
# create testing sites
n = 100
testing = np.linspace(-5,10,n)
# y = np.linspace(-5,10,n)
xv,yv = np.meshgrid(testing,testing)
```

```
boundary = np.zeros((n,n))

# find parameters
miu0 = np.zeros(colx0)
for i in range(colx0):
    miu0[i] = np.mean(xclass0[i])

miu1 = np.zeros(colx1)
for i in range(colx1):
    miu1[i] = np.mean(xclass1[i])

Sigma0 = np.cov(xclass0.T)
Sigma1 = np.cov(xclass1.T)

d = rowx0
abs_Sigma1 = np.linalg.det(Sigma1)
abs_Sigma0 = np.linalg.det(Sigma0)
inv_Sigma1 = np.linalg.inv(Sigma1)
inv_Sigma0 = np.linalg.inv(Sigma0)
const = np.power((2*np.pi),d)

# do Bayesian Decision
for i in range(100):
    for j in range(100):
        block = np.matrix([testing[i],testing[j]]).T
        # block = np.matrix((x[i,0],x[i,1])).T
        c1 = 1/(np.sqrt(const*abs_Sigma1)) * np.exp(-0.5*np.dot(np.dot((block-miu1).T,inv_Sigma1
        c0 = 1/(np.sqrt(const*abs_Sigma0)) * np.exp(-0.5*np.dot(np.dot((block-miu0).T,inv_Sigma0
        if c1[0,0] > c0[0,0]:
            boundary[i,j] = 1
        elif c1[0,0] < c0[0,0]:
            boundary[i,j] = 0
'''
plt.contour(testing,testing,boundary)
plt.show()
'''
# Ex 4
# a
m,n = 100,100
K = np.zeros((m,n))

h = 1
x = x[:,0:2]
for i in range(m):
    for j in range(n):
        K[i,j] = np.exp(-np.power(np.linalg.norm(x[i,:]-x[j,:],ord=1),2)/h)

# print(K[47:52,47:52])

# c
lambd = 0.001
alpha     = cvx.Variable((N,1))
loss        = - cvx.sum(cvx.multiply(y, K @ alpha)) \
              + cvx.sum(cvx.log_sum_exp( cvx.hstack([np.zeros((N,1)), K @ alpha]), axis=1 ) )
reg         = cvx.sum(cvx.quad_form(alpha, K))
prob        = cvx.Problem(cvx.Minimize(loss/N + lambd*reg))
prob.solve()
ALPHA = alpha.value
```