Final Exam, CPSC 8420, Fall 2023

Last Name, First Name

Due 12/16/2023, Saturday, 5:59PM EST

Problem 1 [15 pts]

Consider the following problem:

$$\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda[\alpha\|\beta\|_2^2 + (1-\alpha)\|\beta\|_1]. \tag{1}$$

1. Show the objective can be reformulated into a lasso problem, with revised $\hat{\mathbf{X}}, \hat{\mathbf{y}}$.

   Assume we can find

$$\begin{aligned}
\|\hat{\mathbf{y}} - \hat{\mathbf{X}}\beta\|^2 &= \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\alpha\|\beta\|_2^2 \\
\implies \|\hat{\mathbf{y}}\|^2 - 2\hat{\mathbf{y}}^T\hat{\mathbf{X}}\beta + \|\hat{\mathbf{X}}\beta\|^2 &= \|\mathbf{y}\|^2 - 2vy^T\mathbf{X}\beta \;+\; \underbrace{\|\mathbf{X}\beta\|^2 + \lambda\alpha\|\beta\|_2^2}_{\text{combine these 2 together as } \hat{\mathbf{X}}} \\
\implies \hat{\mathbf{X}} &= \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda\alpha}\mathbf{I} \end{bmatrix} \\
\hat{\mathbf{y}} &= \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}
\end{aligned}$$

2. If $\alpha = 1/2, \lambda = 1$, please derive the closed-form solution by making use of alternating minimization that each time we fix the rest by optimizing one single element in β. You need randomly generate $\mathbf{X}, \mathbf{y}$ and initialize β_0, and show the objective decreases monotonically with updates.

$$\min_{\beta} \frac{1}{2}\|\hat{\mathbf{y}} - \hat{\mathbf{X}}\beta\|^2 + \lambda(1-\alpha)\|\beta\|_1$$

when we try to optimize β_i

$$\Longrightarrow \min_{\beta_i} \frac{1}{2}\|\hat{\mathbf{y}} - \sum_{j \neq i} \hat{\mathbf{x}}_j \beta_j - \hat{\mathbf{x}}_i \beta_i\|^2 + \lambda(1-\alpha)|\beta_i|$$

$$\text{Set } \Delta_i = \hat{\mathbf{y}} - \sum_{j \neq i} \hat{\mathbf{x}}_j \beta_j \Longrightarrow \min_{\beta_i} \frac{1}{2}\|\hat{\mathbf{x}}_i \beta_i - \Delta_i\|^2 + \lambda(1-\alpha)|\beta_i|$$

$$\Longrightarrow \beta_i = \begin{cases} \dfrac{\langle \hat{\mathbf{x}}_i, \Delta_i \rangle - \lambda(1-\alpha)}{\|\hat{\mathbf{x}}_i\|^2}, \text{ if } \langle \hat{\mathbf{x}}_i, \Delta_i \rangle > \lambda(1-\alpha) \\ \dfrac{\langle \hat{\mathbf{x}}_i, \Delta_i \rangle + \lambda(1-\alpha)}{\|\hat{\mathbf{x}}_i\|^2}, \text{ if } \langle \hat{\mathbf{x}}_i, \Delta_i \rangle < -\lambda(1-\alpha) \\ 0, \text{ otherwise} \end{cases}$$

See Figure 1

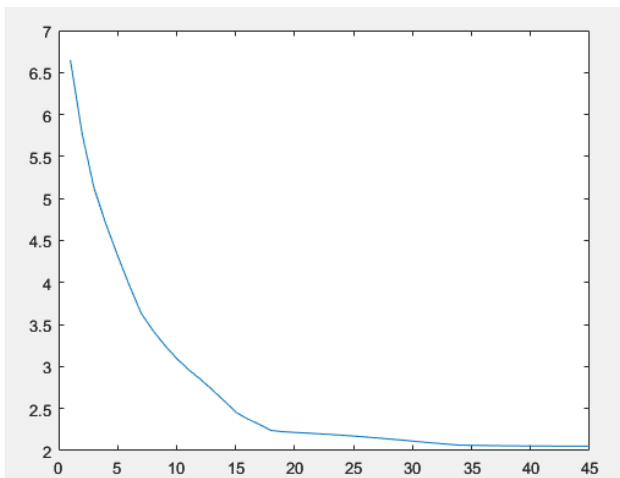

Figure 1: Q1.2

Problem 2 [10 pts]

1. For PCA, the loading vectors can be directly computed from the q columns of $\mathbf{U}$ where $[\mathbf{U}, \mathbf{S}, \mathbf{U}] = svd(\mathbf{X}^T\mathbf{X})$, please show that any $[\pm\mathbf{u}_1, \pm\mathbf{u}_2, \ldots, \pm\mathbf{u}_q]$ will be equivalent to $[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q]$ in terms of the same variance while satisfying the orthonormality constraint. This demonstrates that if the function is nonconvex, it may have various optimal solutions, which is different from (non-trivial) convex function.

   (a) For orthonormality

   $$\text{When } i \neq j, \langle \pm\mathbf{u}_i, \pm\mathbf{u}_j \rangle = \pm\langle \mathbf{u}_i, vu_j \rangle = 0$$

   (b) For variance

   $$\begin{aligned}
   \|\mathbf{X}\mathbf{u}_i\|^2 &= \mathbf{u}_i^T\mathbf{X}^T\mathbf{X}\mathbf{u}_i, \text{ s.t. } \|\mathbf{u}_i\|^2 = 1 \\
   &= trace(\mathbf{u}_i^T\mathbf{X}^T\mathbf{X}\mathbf{u}_i) = trace((-\mathbf{u}_i)^T\mathbf{X}^T\mathbf{X}(-\mathbf{u}_i)) \\
   &= \|\mathbf{X}(-\mathbf{u}_i)\|^2
   \end{aligned}$$

2. Use the fact that $vec(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})vec(\mathbf{X})$ to find the best solution to $\min_{\mathbf{X}} \|\mathbf{AXB} - \mathbf{Y}\|_F^2$, where $\mathbf{A} \in \mathbb{R}^{m\times p}, \mathbf{X} \in \mathbb{R}^{p\times q}, \mathbf{B} \in \mathbb{R}^{q\times n}, \mathbf{Y} \in \mathbb{R}^{m\times n}$.

   $$\begin{aligned}
   \min_{\mathbf{X}} \|\mathbf{AXB} - \mathbf{Y}\|_F^2 &= \min_{\mathbf{X}} \|vec(\mathbf{AXB}) - vec(\mathbf{Y})\|_2^2 \\
   &= \min_{\mathbf{X}} \|(\mathbf{B}^T \otimes \mathbf{A})vec(\mathbf{X}) - vec(\mathbf{Y})\|_2^2
   \end{aligned}$$

   $$\text{According to the regression model: } \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2, \mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

   $$\implies vec(\mathbf{X}^*) = ((\mathbf{B}^T \otimes \mathbf{A})^T(\mathbf{B}^T \otimes \mathbf{A}))^{-1}(\mathbf{B}^T \otimes \mathbf{A})^T vec(\mathbf{Y}),$$

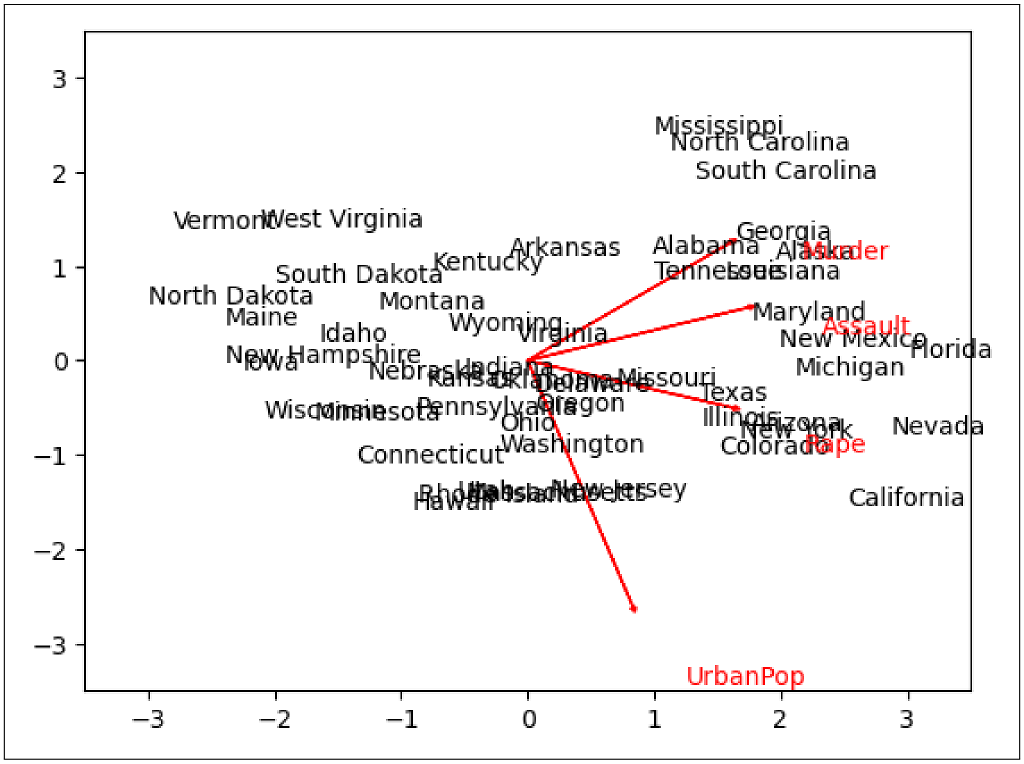

Figure 2: USArrtests

Problem 3 [30 pts]

Please find *USArrests* dataset online and

1. Implement your own program to reproduce the image on page 16/26 of Dimensionality Reduction slides on Canvas.

   - Fig see figure 2
   - Code See Code 1

   ```
   import pandas as pd
   import numpy as np

   from sklearn.preprocessing import StandardScaler
   from numpy import linalg as LA
   import matplotlib.pyplot as plt

   dataset = pd.read_csv("USArrests.csv")
   dataset.head(5)

   states = dataset. iloc[:,0]

   scaler = StandardScaler()
   data = dataset[['Murder', "Assault", "UrbanPop","Rape"]]
   ```

```
scaled_data = scaler.fit_transform(data)

center = np.mean(scaled_data,axis=0)

n_samples = scaled_data.shape[0]
scaled_data = scaled_data - center
Vari = np.dot(scaled_data.T,scaled_data)/n_samples

eigenvalues, eigenvectors = LA.eig(Vari)

PC1 = eigenvectors[:,0]
PC2 = eigenvectors[:,1]

x_list = np.dot(scaled_data,PC1.T)
y_list = np.dot(scaled_data,PC2.T)

murder = [PC1[0],PC2[0]]
assault = np.array(PC1[1],PC2[1])
urbanpop = np.array(PC1[2],PC2[2])
rape = np.array(PC1[3],PC2[3])

features = ["Murder","Assault", "UrbanPop","Rape"]

plt.xlim(-3.5,3.5)
plt.ylim(-3.5,3.5)

for s in range(50):
    plt.text(x_list[s],y_list[s], states[s])

for i in range(4):
# (starting_x, starting_y, dx, dy, ...)
    plt.arrow(0,0, PC1[i]*3,PC2[i]*3, head_width=0.05, head_length=0.05, color='red')
    plt.annotate(features[i],
             xy=(PC1[i]*3.5, PC2[i]*3.5),
             xytext=(20, -20),
             textcoords='offset pixels', color='red')


plt.show()
```

2. For each state, out of 4 features, please randomly mask one and assume it is missing (therefore you have your own Ω and X), please write a program following what we discussed in class (you may refer to ProximalGradientDescent.pdf on Canvas) to optimize

$$\min_{Z} \frac{1}{2}\|P_{\Omega}(X - Z)\|_F^2 + \|Z\|_* \tag{2}$$

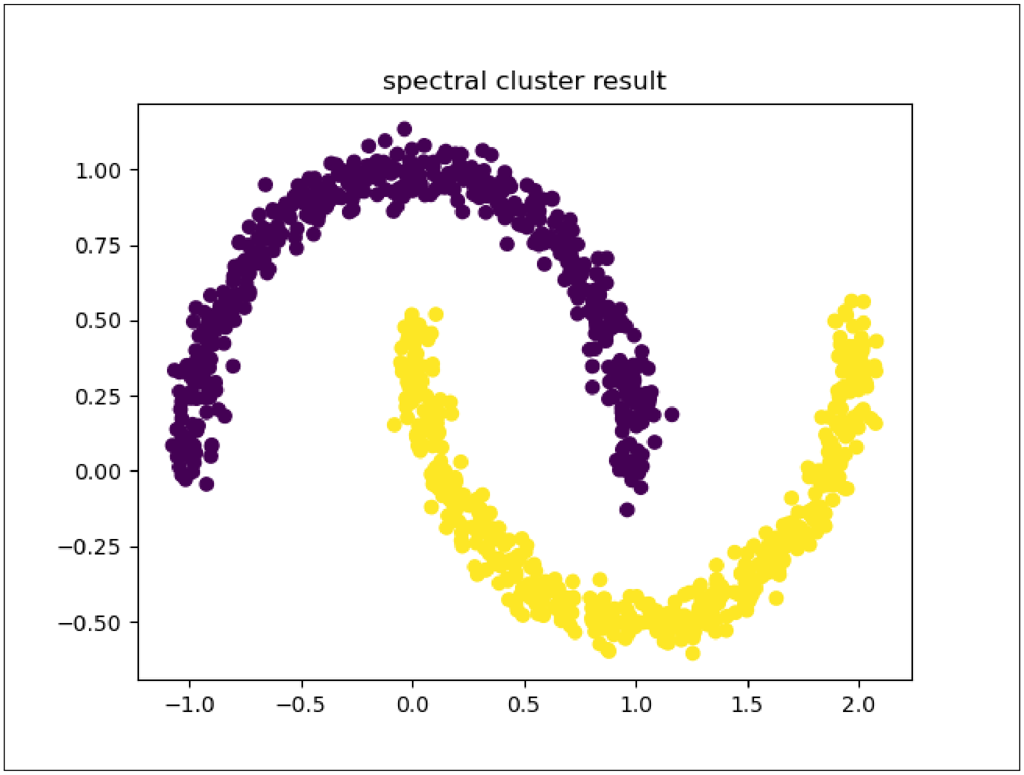

Figure 3: Q4 Sepctral Clustering

Problem 4 [15 pts]

Please refer to here (for Python) or here (for Matlab) to create a *two (half) moon* dataset. Write your own *spectral clustering* codes to separate the data into two groups with different colors. You are not a allowed to call the built-in function for Python or Matlab.

1. Result
   See Figure 3
2. Code
   See Code 2

```
import shadow.utils
shadow.utils.set_seed(0, cudnn_deterministic=True)  # set seeds for reproducibility
#%matplotlib inline
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
import random
import math as m

n_samples = 1000  # number of samples to generate
noise = 0.05  # noise to add to sample locations
X, y = datasets.make_moons(n_samples=n_samples, noise=noise)
```

```
class my_kmeans:
    def __init__(self, clusers=2):
        self.k = clusers

    def cal_dis(self, data, centeroids):
        dis = []
        for i in range(len(data)):
            dis.append([])
            for j in range(self.k):
                dis[i].append(m.sqrt((data[i, 0] - centeroids[j, 0])**2 + (data[i, 1]-centeroids[j
        return np.asarray(dis)

    def divide(self, data, dis):
        clusterRes = [0] * len(data)
        for i in range(len(data)):
            seq = np.argsort(dis[i])
            clusterRes[i] = seq[0]

        return np.asarray(clusterRes)

    def centeroids(self, data, clusterRes):
        centeroids_new = []
        for i in range(self.k):
            idx = np.where(clusterRes == i)
            sum = data[idx].sum(axis=0)
            avg_sum = sum/len(data[idx])
            centeroids_new.append(avg_sum)
        centeroids_new = np.asarray(centeroids_new)
        return centeroids_new[:, 0: 2]

    def cluster(self, data, centeroids):
        clulist = self.cal_dis(data, centeroids)
        clusterRes = self.divide(data, clulist)
        centeroids_new = self.centeroids(data, clusterRes)
        err = centeroids_new - centeroids
        return err, centeroids_new, clusterRes

    def fit(self,data):
        clu = random.sample(data[:, 0:2].tolist(), 2)
        clu = np.asarray(clu)
        err, clunew,  clusterRes = self.cluster(data, clu)
        while np.any(abs(err) > 0):
            #print(clunew)
            err, clunew,  clusterRes = self.cluster(data, clunew)

        clulist = self.cal_dis(data, clunew)
        clusterResult = self.divide(data, clulist)

        return clusterResult
```

```
def myKNN(S, k, sigma=2.0):
    N = len(S)
    A = np.zeros((N,N))

    for i in range(N):
        dist_with_index = zip(S[i], range(N))
        dist_with_index = sorted(dist_with_index, key=lambda x:x[0])
        neighbours_id = [dist_with_index[m][1] for m in range(k+1)] # xi's k nearest neighbours

        for j in neighbours_id: # xj is xi's neighbour
            A[i][j] = np.exp(-S[i][j]/2/sigma/sigma)
            A[j][i] = A[i][j] # mutually

    return A

def calLaplacianMatrix(adjacentMatrix):

    # compute the Degree Matrix: D=sum(A)
    degreeMatrix = np.sum(adjacentMatrix, axis=1)

    # compute the Laplacian Matrix: L=D-A
    laplacianMatrix = np.diag(degreeMatrix) - adjacentMatrix

    # normailze
    # D^(-1/2) L D^(-1/2)
    sqrtDegreeMatrix = np.diag(1.0 / (degreeMatrix ** (0.5)))
    return np.dot(np.dot(sqrtDegreeMatrix, laplacianMatrix), sqrtDegreeMatrix)

def euclidDistance(x1, x2, sqrt_flag=False):
    res = np.sum((x1-x2)**2)
    if sqrt_flag:
        res = np.sqrt(res)
    return res

def Distance(X):
    X = np.array(X)
    S = np.zeros((len(X), len(X)))
    for i in range(len(X)):
        for j in range(i+1, len(X)):
            S[i][j] = 1.0 * euclidDistance(X[i], X[j])
            S[j][i] = S[i][j]
    return S

clusters = 2

Similarity = Distance(X)
Adjacent = myKNN(Similarity, k=5)
Laplacian = calLaplacianMatrix(Adjacent)
x, V = np.linalg.eig(Laplacian)
```

```
x = zip(x, range(len(x)))
x = sorted(x, key=lambda x:x[0])
H = np.vstack([V[:,i] for (v, i) in x[:clusters]]).T
result = my_kmeans(2).fit(H)
plt.title('spectral cluster result')
plt.scatter(X[:,0], X[:,1],marker='o',c=result)
plt.show()
```

Problem 5 [35 pts]

For Logistic Regression, if the label is ± 1, the objective is:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} log(1 + exp(-y_i \mathbf{w}^T \mathbf{x}_i)) \tag{3}$$

while if the label is $\{1, 0\}$ the objective is:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} log(1 + exp(\mathbf{w}^T \mathbf{x}_i)) - y_i \mathbf{w}^T \mathbf{x}_i \tag{4}$$

1. Write a program to show that the optimal solutions to the two cases are the same by making use of gradient descent method where $m = 100$ (please carefully choose the stepsize as we discussed in class). You can generate two class samples, one class's label is 1 and the other is -1 or 0 corresponding to the two formulations respectively. You can initialize $\mathbf{w}$ as $\mathbf{0}$.

   (a) Convert to Matrix form
   Seperatly convert equations 4 and 3 to Matrix Format
   For Lable $\{1, 0\}$

   $$\min_{\mathbf{w}} \mathbf{L} = \min_{\mathbf{w}} -(\mathbf{Y}^T \mathbf{X}\mathbf{w} - \mathbf{I}^T log(\mathbf{I} + exp(\mathbf{X}\mathbf{w}))) \tag{5}$$

   For Lable $\{1, -1\}$

   $$\min_{\mathbf{w}} \mathbf{L} = \min_{\mathbf{w}} \mathbf{I}^T log(\mathbf{I} + exp(-\mathbf{Y}^T \mathbf{X}\mathbf{w})) \tag{6}$$

   where, $\mathbf{I} \in \mathbb{R}^{m \times 1}$ and all elements are 1s

   (b) Gradient
   Firstly, we express the sigmoid function as

   $$h_{\mathbf{w}}(\mathbf{Z}) = \frac{1}{1 + exp(-\mathbf{Z}\mathbf{w})} \tag{7}$$

   For Lable $\{1, 0\}$, the derivitive can be easily get from slides

   $$\frac{\partial \mathbf{L}}{\partial \mathbf{w}} = \mathbf{X}^T (h_{\mathbf{w}}(\mathbf{X}) - \mathbf{Y}) \tag{8}$$

For Lable $\{1,-1\}$

$$
\begin{aligned}
\mathrm{d}\mathbf{L}(\mathbf{w}) &= \mathrm{d}\mathbf{I}^T log(\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})) + \mathbf{I}^T\,\mathrm{d}log(\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})) \\
&= \mathbf{I}^T\,\mathrm{d}log(\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})) \\
&= \mathbf{I}^T\left(\frac{1}{\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})}\odot \mathrm{d}\left(\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})\right)\right) \\
&= \left(\mathbf{I}\odot\frac{1}{\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})}\right)^T \mathrm{d}(\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})) \\
&= \left(\mathbf{I}\odot\frac{1}{\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})}\right)^T \left(exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})\odot \mathrm{d}(-\mathbf{Y}^T\mathbf{X}\mathbf{w})\right) \\
&= \left(\mathbf{I}\odot\frac{1}{\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})}\odot exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})\right)^T \mathrm{d}(-\mathbf{Y}^T\mathbf{X}\mathbf{w}) \\
&= -\left(\mathbf{I}\odot\frac{1}{\mathbf{I}+exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})}\odot exp(-\mathbf{Y}^T\mathbf{X}\mathbf{w})\right)^T \mathbf{Y}^T\mathbf{X}\,\mathrm{d}\mathbf{w} \\
&= -h_{\mathbf{w}}\left(-\mathbf{Y}^T\mathbf{X}\right)\mathbf{Y}^T\mathbf{X}\,\mathrm{d}\mathbf{w}
\end{aligned}
$$

For above we get

$$
\frac{\partial \mathbf{L}}{\partial \mathbf{w}} = -h_{\mathbf{w}}\left(-\mathbf{Y}^T\mathbf{X}\right)\mathbf{Y}^T\mathbf{X} \tag{9}
$$

(c) StepSize

(d)

2. Consider the case where class label is $\{1,0\}$ and $P(y=1|\mathbf{x},\mathbf{w}) = \frac{1}{1+exp(-\mathbf{w}^T\mathbf{x})}$, the maximum likelihood function is $p^y(1-p)^{1-y}$, which is equivalent to $\min -ylog(p)-(1-y)log(1-p)$, exactly the binary cross entropy. Please find optimal p.

3. If we use Mean Square Error instead of cross entropy: $\min\ (y-p)^2$, and assume $y=1$ and our initial weight $\mathbf{w}$ result in p very close to 0, if we optimize $\mathbf{w}$ by making use of gradient descent method, what will happen? Convince yourself that it will stuck at initial point and explain briefly why.

4. For the second objective where the label is $\{1,0\}$, implement Newton method (with backtracking line search if necessary) where $m=100$. Compare with gradient descent method and plot objective versus time consumption in one figure to observe which is faster.

5. From now on, let's focus on the first objective where the label is ± 1. Please write a program to find the optimal $\mathbf{w}$ by using gradient descent method where $m=10K$, the stepsize in this case we set it as $\frac{1}{\|\mathbf{X}\|_F^2}$ where each column of $\mathbf{X}$ is $\mathbf{x}_i$.

6. Please write a stochastic gradient descent version for $m=10K$ (you may set the stepsize as $2/(t+1)$ where $t=1,\ldots,T$ and $T=100K$) with the final output being $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\frac{2t}{T+1}\mathbf{w}_t$.

7. Please compare those two methods (gradient descent vs. stochastic gradient descent) for $m=10K$ and $m=100$ by plotting objective changes versus time consumption respectively.

Problem 6 [15 pts]

We consider multiclass SVM based on binary SVM. There are two options we can consider: one versus one and one versus all. Assume we have 4 classes data where each class has 2 samples: class 1 $\{\{1,0\},\{2,0\}\}$, class 2 $\{\{0,-1\},\{0,-2\}\}$, class 3 $\{\{-1,0\},\{-2,0\}\}$ and class 4 $\{\{0,1\},\{0,2\}\}$. Now use the two options (one versus one and one versus all) respectively to determine the predicted class of new data $\{0.25,1.5\}$. You should explicitly find and write each hyperplane to get full credits.