Problem 2

Use this notebook to write your code for problem 2. You may reuse your SGD code from last week.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import math
import random
%matplotlib inline
```

The following function may be useful for loading the necessary data.

In [ ]:

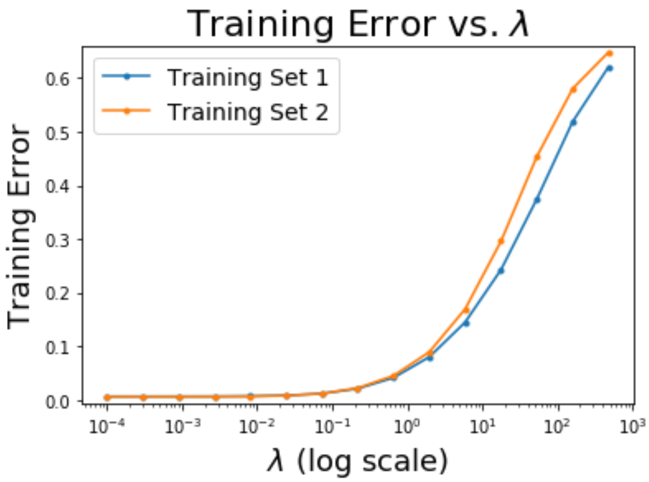
```
def load_data(filename):
    return np.loadtxt(filename, skiprows=1, delimiter=',')

def normalX(x):
    xN = x
    means = []
    stds = []
    for i in range(1, len(xN[0])):
        means.append(np.mean(xN[:,[i]]))
        stds.append(np.std(xN[:,[i]]))
    for i in inputsNormalized:
        for j in range(1, len(i)):
            i[j] = (i[j] - means[j-1]) / stds[j-1]
    return xN

def normalY(x, y):
    xN = x
    yN = y
    means = []
    stds = []
    for i in range(1, len(xN[0])):
        means.append(np.mean(xN[:,[i]]))
        stds.append(np.std(xN[:,[i]]))
    for i in yN:
        for j in range(1, len(i)):
            i[j] = (i[j] - means[j-1]) / stds[j-1]
    return yN

def getXYnorm(data):
    x = []
    y = []
    arr = np.asarray([1.0])
    for i in data:
        x.append(np.concatenate((arr, i[1:]), axis = 0))
        y.append(i[0])
    return normalX(np.asarray(x)), np.asarray(y)

def getXY(data):
    x = []
    y = []
    arr = np.asarray([1.0])
    for i in data:
        x.append(np.concatenate((arr, i[1:]), axis = 0))
        y.append(i[0])
    return np.asarray(x), np.asarray(y)

def loss(weights, y, x):
    totalLoss = 0
    for i in range(len(x)):
        if (y[i] == -1):
            add = np.log(1 / (1 + math.exp(np.inner(weights, x[i]))))
        else:
            add = np.log(1 / (1 + math.exp(-np.inner(weights, x[i]))))
        totalLoss += add
    return totalLoss / len(x) * -1
```

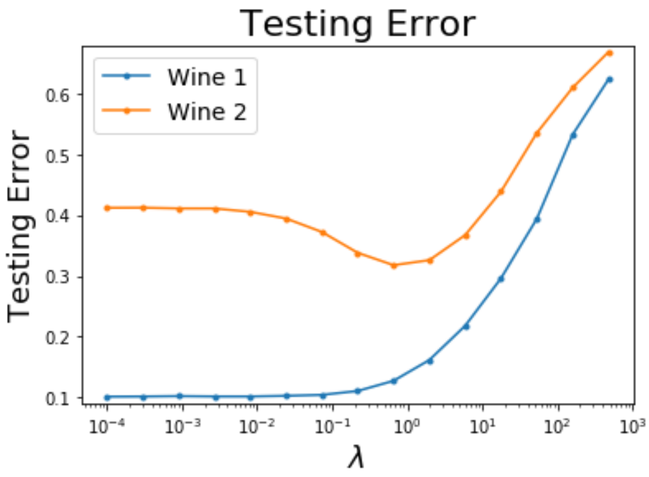
In [ ]:

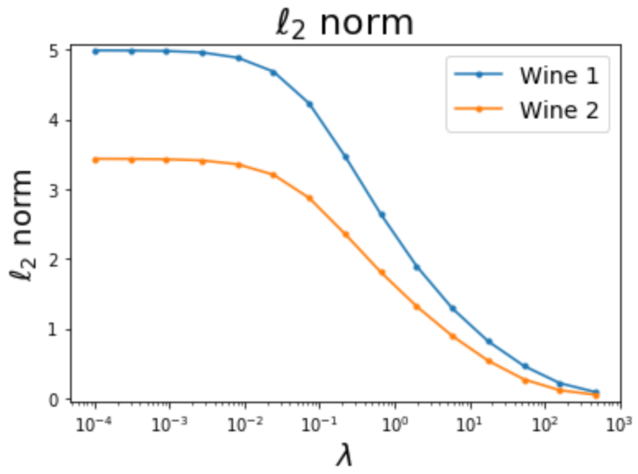
```
def calcGrad(x, y, w, l, size):
    ret =  2* l * w / size - x * y / (math.exp(np.inner(w, x) * y) + 1)
    return ret

def L2Norm(w):
    return math.sqrt((np.inner(w, w)))

def runSGD(data, iW, stepSize, l):
    numEpochs = 20000
    totalLoss = []
    w = iW
    x, y = getXYnorm(data)
    currLoss = loss(w, y, x)
    totalLoss.append(currLoss)

    for _ in range(numEpochs):
        np.random.shuffle(data)
        x, y = getXYnorm(data)

        for i in range(len(x)):
            grad = calcGrad(x[i], y[i], w, l, len(x))
            w -= stepSize * grad

        currLoss = loss(w, y, x)
        totalLoss.append(currLoss)

    return w, totalLoss

# wine 1 -------------
data1 = load_data("data/wine_training1.txt")
lambda0 = 0.00001
lambdas = []
w = []
loss = []
step = math.exp(-4)

for i in range(15):
    start = [0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001,
0.001, 0.001, 0.001, 0.001]
    finWeights, totalLoss = runSGD(data1, start, step, lambda0)
    w.append(finWeights)
    loss.append(totalLoss[-1])
    lambdas.append(lambda0)
    lambda0 *= 3

# wine 2 -------------
lambdas2 = []
w2 = []
loss2 = []
lambda0 = 0.00001
step = math.exp(-4)
data2 = load_data("data/wine_training2.txt")
```

In [29]:

```
for i in range(15):
    start = [0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001,
0.001, 0.001, 0.001, 0.001]
    finWeights2, totalLoss2 = runSGD(data2, start, step, lambda0)
    w2.append(finWeights2)
    loss2.append(totalLoss2[-1])
    lambdas2.append(lambda0)
    lambda0 *= 3

fig = plt.figure()
plt.title(r'Training Error vs. $\lambda$', fontsize = 22)
plt.plot(lambdas, loss, lambdas1, loss1, marker = '.')
plt.legend(('Training Set 1', 'Training Set 2'), loc = 'best', fontsize = 14)
plt.xscale('log')
plt.xlabel('$\lambda$ (log scale)', fontsize = 18)
plt.ylabel('Training Error', fontsize = 18)
plt.margins(y=0.02)

# test -------------
trainx1, trainy1 = getXY(allData)
trainx2, trainy2 = getXY(allData1)
testData = load_data("data/wine_testing.txt")
testx1, testy1 = getXY(testData)
testx2, testy2 = getXY(testData)

testerr1 = []
testerr2 = []

testxnorm1 = normalY(trainx1, testx1)
testxnorm2 = normalY(trainx2, testx2)

for i in w:
    testerr1.append(loss(i, trainy1, testxnorm1))
for j in w2:
    testerr2.append(loss(j, trainy1, testxnorm2))

fig = plt.figure()
plt.title(r'Testing Error')
plt.plot(lambdas, testerr1, lambdas1, testerr2, marker = '.')
plt.legend(('Wine 1', 'Wine 2'))
plt.xscale('log')
plt.xlabel('$\lambda$')
plt.ylabel('Testing Error')
plt.margins(y=0.02)

# lambda -------------
norm1 = []
norm2 = []

for i in w:
    norm1.append(L2Norm(i))
for j in w2:
    norm2.append(L2Norm(j))

fig = plt.figure()
plt.title(r'$\ell_2$ norm')
plt.plot(lambdas, norm1, lambdas1, norm2, marker = '.')
plt.legend(('Wine 1', 'Wine 2'))
plt.xscale('log')
plt.xlabel('$\lambda$')
plt.ylabel('$\ell_2$ norm')
plt.margins(y=0.02)
```


Training Error vs. λ
Training Set 1
Training Set 2
Training Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
λ (log scale)


Testing Error
Wine 1
Wine 2
Testing Error
0.1
0.2
0.3
0.4
0.5
0.6
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
λ


ℓ_2 norm
Wine 1
Wine 2
ℓ_2 norm
0
1
2
3
4
5
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
λ

In [ ]:

In [ ]: