ANLY 561 HW

Name:Yuqi Wang
NetID:yw545

Problem 1

In [*]:

```
import tensorflow as tf
import numpy as np
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer() # Loads the Wisconsin Breast Cancer dataset (569 examples in 30 dimensions)

# Parameters for the data
dim_data = 30
num_labels = 2
num_examples = 569

# Parameters for training
num_train = 400

X = data['data'] # Data in rows
targets = data.target # 0-1 labels
labels = np.zeros((num_examples, num_labels))
for i in range(num_examples):
    labels[i,targets[i]]=1 # Conversion to one-hot representations


# Backtracking parameters
alpha = 0.1
beta = 0.5

# Let's use TensorFlow to train logisitic regression

x = tf.placeholder(tf.float32, shape=[None, dim_data])
y_ = tf.placeholder(tf.float32, shape=[None, num_labels])

#########################
b = tf.Variable(tf.zeros(num_labels))
w = tf.Variable(tf.zeros([dim_data, num_labels]))

b_bt = tf.Variable(tf.zeros(num_labels))
w_bt = tf.Variable(tf.zeros([dim_data, num_labels]))
#########################

y_prime = tf.matmul(x, w) + b
y_prime_bt = tf.matmul(x, w_bt) + b_bt
```

```python
y = tf.nn.softmax(y_prime)
#y_bt = tf.nn.softmax(y_prime_bt)

f = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_prime))
f_bt = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_prime_bt))

#################
# Start

sess = tf.Session()
sess.run(tf.global_variables_initializer())

correct_prediction = tf.equal(tf.argmax(y , 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

train_accuracy = sess.run(accuracy, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
train_cross_entropy = sess.run(f, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
print("Initial training accuracy %g, cross entropy %g" % (train_accuracy, train_cross_entropy))


for i in range(200):

    f0 = sess.run(f,feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
    f0= float(f0)

    dbf = sess.run(tf.gradients(f,b), feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})[0]
    dbf = np.array(dbf)
    dwf = sess.run(tf.gradients(f,w), feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})[0]
    dwf = np.array(dwf)

    delta = -(np.sum(dbf*dbf)+np.sum(dwf*dwf))*alpha
    #print(delta)
    t = 1
    ##################################
    sess.run(tf.assign(w_bt, sess.run(w) - t*dwf))
    sess.run(tf.assign(b_bt, sess.run(b) - t*dbf))

    fval = sess.run(f_bt, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
    fval = float(fval)

    #print(f_orig + delta*t)
```

```
    while (not np.isfinite(fval)) or f0 + delta*t < fval:
    #for j in range (2):
        t = beta * t
        sess.run(tf.assign(w_bt, sess.run(w) - t*dwf))
        sess.run(tf.assign(b_bt, sess.run(b) - t*dbf))

        fval = sess.run(f_bt, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
        fval = float(fval)

    sess.run(tf.assign(w, w_bt))
    sess.run(tf.assign(b, b_bt))

    train_accuracy = sess.run(accuracy, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})
    train_cross_entropy = sess.run(f, feed_dict={x: X[:num_train, :], y_: labels[:num_train, :]})

    print("i=%g, accuracy=%g, cross entropy=%g" % (i, train_accuracy, train_cross_entropy))

logistic_bt_accuracy = sess.run(accuracy, feed_dict={x: X[num_train:, :], y_: labels[num_train:, :]})
print("Final accuracy: %g" % logistic_bt_accuracy)

sess.close()
```

```
Initial training accuracy 0.4325, cross entropy 0.693147
i=0, accuracy=0.4325, cross entropy=0.668975
i=1, accuracy=0.4325, cross entropy=0.657811
i=2, accuracy=0.4325, cross entropy=0.655563
i=3, accuracy=0.4325, cross entropy=0.650564
i=4, accuracy=0.4325, cross entropy=0.64719
i=5, accuracy=0.4325, cross entropy=0.64578
i=6, accuracy=0.4325, cross entropy=0.640541
i=7, accuracy=0.4325, cross entropy=0.638319
i=8, accuracy=0.4325, cross entropy=0.634278
i=9, accuracy=0.4325, cross entropy=0.631467
i=10, accuracy=0.4325, cross entropy=0.6284
i=11, accuracy=0.49, cross entropy=0.625912
i=12, accuracy=0.4325, cross entropy=0.6201
i=13, accuracy=0.465, cross entropy=0.618234
i=14, accuracy=0.4325, cross entropy=0.614554
i=15, accuracy=0.4575, cross entropy=0.611923
i=16, accuracy=0.4325, cross entropy=0.610526
```

```
i=17, accuracy=0.455, cross entropy=0.606422
i=18, accuracy=0.4325, cross entropy=0.603528
i=19, accuracy=0.495, cross entropy=0.598911
i=20, accuracy=0.4375, cross entropy=0.596304
i=21, accuracy=0.55, cross entropy=0.59181
i=22, accuracy=0.4425, cross entropy=0.588991
i=23, accuracy=0.62, cross entropy=0.585093
i=24, accuracy=0.4725, cross entropy=0.581876
i=25, accuracy=0.6875, cross entropy=0.578734
i=26, accuracy=0.465, cross entropy=0.575524
i=27, accuracy=0.76, cross entropy=0.568087
i=28, accuracy=0.5875, cross entropy=0.565483
i=29, accuracy=0.7725, cross entropy=0.562524
i=30, accuracy=0.5925, cross entropy=0.557942
i=31, accuracy=0.82, cross entropy=0.553191
i=32, accuracy=0.64, cross entropy=0.549513
i=33, accuracy=0.8425, cross entropy=0.544543
i=34, accuracy=0.5975, cross entropy=0.542258
i=35, accuracy=0.875, cross entropy=0.529595
i=36, accuracy=0.7025, cross entropy=0.509504
i=37, accuracy=0.8575, cross entropy=0.499305
i=38, accuracy=0.91, cross entropy=0.482354
i=39, accuracy=0.895, cross entropy=0.480432
i=40, accuracy=0.8875, cross entropy=0.479322
i=41, accuracy=0.8975, cross entropy=0.477928
i=42, accuracy=0.9075, cross entropy=0.476823
i=43, accuracy=0.9025, cross entropy=0.475555
i=44, accuracy=0.9125, cross entropy=0.474476
i=45, accuracy=0.905, cross entropy=0.473317
i=46, accuracy=0.9075, cross entropy=0.472277
i=47, accuracy=0.905, cross entropy=0.471205
i=48, accuracy=0.91, cross entropy=0.470208
i=49, accuracy=0.9075, cross entropy=0.469205
i=50, accuracy=0.91, cross entropy=0.46825
i=51, accuracy=0.9075, cross entropy=0.467301
i=52, accuracy=0.9125, cross entropy=0.466813
i=53, accuracy=0.9, cross entropy=0.465501
i=54, accuracy=0.9075, cross entropy=0.464422
i=55, accuracy=0.9075, cross entropy=0.463332
i=56, accuracy=0.91, cross entropy=0.462356
i=57, accuracy=0.905, cross entropy=0.461394
i=58, accuracy=0.91, cross entropy=0.460491
```

i=59, accuracy=0.9025, cross entropy=0.460099
i=60, accuracy=0.9075, cross entropy=0.458899
i=61, accuracy=0.905, cross entropy=0.457712
i=62, accuracy=0.91, cross entropy=0.456716
i=63, accuracy=0.905, cross entropy=0.455757
i=64, accuracy=0.905, cross entropy=0.454878
i=65, accuracy=0.9025, cross entropy=0.454503
i=66, accuracy=0.91, cross entropy=0.453295
i=67, accuracy=0.9025, cross entropy=0.452156
i=68, accuracy=0.9075, cross entropy=0.451209
i=69, accuracy=0.905, cross entropy=0.450316
i=70, accuracy=0.905, cross entropy=0.449495
i=71, accuracy=0.8975, cross entropy=0.448766
i=72, accuracy=0.9075, cross entropy=0.447713
i=73, accuracy=0.9075, cross entropy=0.446753
i=74, accuracy=0.905, cross entropy=0.445916
i=75, accuracy=0.9025, cross entropy=0.445472
i=76, accuracy=0.9075, cross entropy=0.444309
i=77, accuracy=0.905, cross entropy=0.443288
i=78, accuracy=0.905, cross entropy=0.442441
i=79, accuracy=0.9025, cross entropy=0.44215
i=80, accuracy=0.905, cross entropy=0.440924
i=81, accuracy=0.905, cross entropy=0.439887
i=82, accuracy=0.905, cross entropy=0.439052
i=83, accuracy=0.9025, cross entropy=0.438734
i=84, accuracy=0.9075, cross entropy=0.437532
i=85, accuracy=0.905, cross entropy=0.436542
i=86, accuracy=0.91, cross entropy=0.435746
i=87, accuracy=0.9, cross entropy=0.435254
i=88, accuracy=0.9075, cross entropy=0.434163
i=89, accuracy=0.905, cross entropy=0.433271
i=90, accuracy=0.9075, cross entropy=0.432532
i=91, accuracy=0.9, cross entropy=0.431815
i=92, accuracy=0.91, cross entropy=0.43088
i=93, accuracy=0.905, cross entropy=0.430102
i=94, accuracy=0.9075, cross entropy=0.429521
i=95, accuracy=0.9025, cross entropy=0.428485
i=96, accuracy=0.9075, cross entropy=0.427718
i=97, accuracy=0.9, cross entropy=0.427198
i=98, accuracy=0.91, cross entropy=0.426178
i=99, accuracy=0.905, cross entropy=0.425383
i=100, accuracy=0.9075, cross entropy=0.424893

i=101, accuracy=0.9025, cross entropy=0.423833
i=102, accuracy=0.9075, cross entropy=0.423085
i=103, accuracy=0.9, cross entropy=0.422537
i=104, accuracy=0.91, cross entropy=0.421573
i=105, accuracy=0.905, cross entropy=0.420836
i=106, accuracy=0.91, cross entropy=0.420221
i=107, accuracy=0.905, cross entropy=0.419303
i=108, accuracy=0.9075, cross entropy=0.41863
i=109, accuracy=0.9025, cross entropy=0.41791
i=110, accuracy=0.9075, cross entropy=0.417113
i=111, accuracy=0.9025, cross entropy=0.416732
i=112, accuracy=0.91, cross entropy=0.415694
i=113, accuracy=0.905, cross entropy=0.414963
i=114, accuracy=0.9125, cross entropy=0.414375
i=115, accuracy=0.905, cross entropy=0.413496
i=116, accuracy=0.9075, cross entropy=0.413195
i=117, accuracy=0.9025, cross entropy=0.412084
i=118, accuracy=0.9075, cross entropy=0.411396
i=119, accuracy=0.9025, cross entropy=0.410785
i=120, accuracy=0.9075, cross entropy=0.409979
i=121, accuracy=0.9, cross entropy=0.409611
i=122, accuracy=0.9075, cross entropy=0.408617
i=123, accuracy=0.9075, cross entropy=0.407951
i=124, accuracy=0.91, cross entropy=0.407294
i=125, accuracy=0.905, cross entropy=0.406554
i=126, accuracy=0.91, cross entropy=0.406017
i=127, accuracy=0.905, cross entropy=0.405182
i=128, accuracy=0.91, cross entropy=0.404781
i=129, accuracy=0.905, cross entropy=0.403835
i=130, accuracy=0.9125, cross entropy=0.403576
i=131, accuracy=0.9025, cross entropy=0.402508
i=132, accuracy=0.905, cross entropy=0.401894
i=133, accuracy=0.9025, cross entropy=0.401229
i=134, accuracy=0.905, cross entropy=0.400573
i=135, accuracy=0.9025, cross entropy=0.399975
i=136, accuracy=0.905, cross entropy=0.399272
i=137, accuracy=0.9025, cross entropy=0.398744
i=138, accuracy=0.905, cross entropy=0.397989
i=139, accuracy=0.9025, cross entropy=0.397529
i=140, accuracy=0.905, cross entropy=0.396724
i=141, accuracy=0.9025, cross entropy=0.396326
i=142, accuracy=0.905, cross entropy=0.395474

```
i=143, accuracy=0.9025, cross entropy=0.395127
i=144, accuracy=0.905, cross entropy=0.394237
i=145, accuracy=0.9025, cross entropy=0.393928
i=146, accuracy=0.9075, cross entropy=0.393011
i=147, accuracy=0.9025, cross entropy=0.392724
i=148, accuracy=0.9075, cross entropy=0.391796
i=149, accuracy=0.9025, cross entropy=0.391512
i=150, accuracy=0.9075, cross entropy=0.390592
i=151, accuracy=0.9025, cross entropy=0.390294
i=152, accuracy=0.9075, cross entropy=0.389397
i=153, accuracy=0.9025, cross entropy=0.38907
i=154, accuracy=0.9075, cross entropy=0.388214
i=155, accuracy=0.9025, cross entropy=0.387847
i=156, accuracy=0.905, cross entropy=0.387043
i=157, accuracy=0.9025, cross entropy=0.386628
i=158, accuracy=0.905, cross entropy=0.385885
i=159, accuracy=0.9025, cross entropy=0.38542
i=160, accuracy=0.905, cross entropy=0.384742
```

```
After about $50$ steps, the accuracy will be around $0.9$ and $0.91$.
```

Problem 2

Part (a)

We have

$$\mathcal{A} = \left(\left(\begin{pmatrix} a_{1,1,1,1} & a_{1,1,1,2} \\ a_{1,1,2,1} & a_{1,1,2,2} \end{pmatrix}, \begin{pmatrix} a_{1,2,1,1} & a_{1,2,1,2} \\ a_{1,2,2,1} & a_{1,2,2,2} \end{pmatrix} \right), \left(\begin{pmatrix} a_{2,1,1,1} & a_{2,1,1,2} \\ a_{2,1,2,1} & a_{2,1,2,2} \end{pmatrix}, \begin{pmatrix} a_{2,2,1,1} & a_{2,2,1,2} \\ a_{2,2,2,1} & a_{2,2,2,2} \end{pmatrix} \right), \left(\begin{pmatrix} a_{3,1,1,1} & a_{3,1,1,2} \\ a_{3,1,2,1} & a_{3,1,2,2} \end{pmatrix}, \begin{pmatrix} a_{3,2,1,1} & a_{3,2} \\ a_{3,2,2,1} & a_{3,2} \end{pmatrix}\right.\right.$$

$$= \left(\left(\begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -2 & 2 \end{pmatrix} \right), \left(\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \right), \left(\begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \right) \right)$$

and

$$\mathcal{B} = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -2 & 2 \end{pmatrix}$$

Since $\mathcal{A}$ has the shape of 3 by 2 by 2 by 2 and $\mathcal{B}$ has the shape of 2 by 2, $\mathcal{F}$ (the contraction of $\mathcal{A}$ and $\mathcal{B}$ along $i = 2, 3$ and $j = 1, 2$) must have the shape of 3 by 2.Therefore, the order of $\mathcal{F}$ is 2.

Order: 2

Shape: 3 by 2

Part (b)

$$\mathcal{F}_{1,1} = a_{1,1,1,1}b_{1,1} + a_{1,1,2,1}b_{1,2} + a_{1,2,1,1}b_{2,1} + a_{1,2,2,1}b_{2,2} = (1)(1) + (-2)(-1) + (1)(-2) + (-2)(2) = 1 + 2 - 2 - 4 = -3$$

$$\mathcal{F}_{1,2} = a_{1,1,1,2}b_{1,1} + a_{1,1,2,2}b_{1,2} + a_{1,2,1,2}b_{2,1} + a_{1,2,2,2}b_{2,2} = (-1)(1) + (1)(-1) + (1)(-2) + (2)(2) = -1 - 1 - 2 + 4 = 0$$

$$\mathcal{F}_{2,1} = a_{2,1,1,1}b_{1,1} + a_{2,1,2,1}b_{1,2} + a_{2,2,1,1}b_{2,1} + a_{2,2,2,1}b_{2,2} = (2)(1) + (-1)(-1) + (2)(-2) + (1)(2) = 2 + 1 - 4 + 2 = 1$$

$$\mathcal{F}_{2,2} = a_{2,1,1,2}b_{1,1} + a_{2,1,2,2}b_{1,2} + a_{2,2,1,2}b_{2,1} + a_{2,2,2,2}b_{2,2} = (-1)(1) + (1)(-1) + (1)(-2) + (2)(2) = -1 - 1 - 2 + 4 = 0$$

$$\mathcal{F}_{3,1} = a_{3,1,1,1}b_{1,1} + a_{3,1,2,1}b_{1,2} + a_{3,2,1,1}b_{2,1} + a_{3,2,2,1}b_{2,2} = (1)(1) + (-2)(-1) + (1)(-2) + (-1)(2) = 1 + 2 - 2 - 2 = -1$$

$$\mathcal{F}_{3,2} = a_{3,1,1,2}b_{1,1} + a_{3,1,2,2}b_{1,2} + a_{3,2,1,2}b_{2,1} + a_{3,2,2,2}b_{2,2} = (-2)(1) + (1)(-1) + (1)(-2) + (1)(2) = -2 - 1 - 2 + 2 = -3$$

Thus,

$$\mathcal{F} = \begin{pmatrix} -3 & 0 \\ 1 & 0 \\ -1 & -3 \end{pmatrix}$$

In [2]:
```python
import tensorflow as tf

A = tf.Variable([[[[1, -1], [-2, 1]],[[1, 1], [-2, 2]]],[[[2, -1], [-1, 1]],[[2, 1], [1, 2]]],[[[1, -2], [-2, 1]],[[1, 1], [-1, 1]]]],
B = tf.Variable([[1, -1], [-2, 2]], name='B') # This is a 2 by 2 by 2 tensor
f = tf.tensordot(A, B, [[1,2], [0,1]]) # Contraction along two indices

with tf.Session() as sess:
    tf.global_variables_initializer().run()
    result = f.eval()

print('Result should be a 3 by 2 matrix:')
print(result)
```

```
Result should be a 3 by 2 matrix:
[[-3  0]
 [ 1  0]
 [-1 -3]]
```

We can see that two results are the same!

Problem 3

Part (a)

Our group have found 4 datasets. In this homework, I will only use one raw dataset called 'train.csv'.

In [10]:
```
import matplotlib.pyplot as plt
import pandas as pd

myDF = pd.read_csv('C:/Users/45336/Desktop/2017 Fall/Anlytics 561 Optimization/Project/train.csv')
print(myDF[:5],'\n')
print(myDF.describe())
```

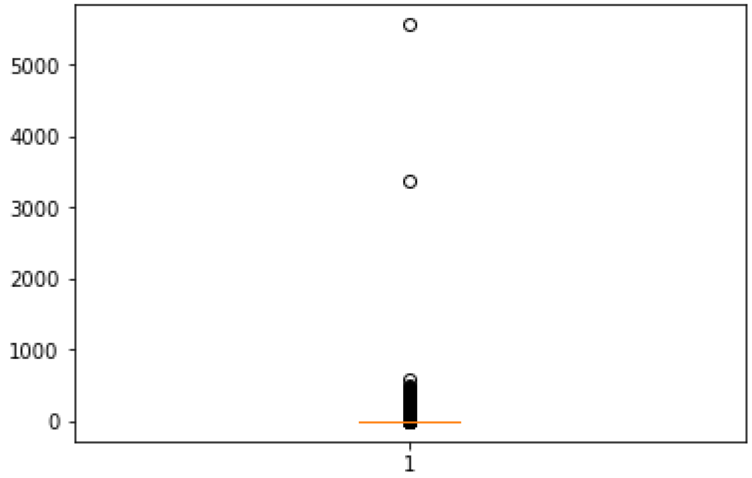
```
         date  store_nbr  item_nbr  units
0  2012-01-01          1         1      0
1  2012-01-01          1         2      0
2  2012-01-01          1         3      0
3  2012-01-01          1         4      0
4  2012-01-01          1         5      0

          store_nbr      item_nbr         units
count  4.617600e+06  4.617600e+06  4.617600e+06
mean   2.309108e+01  5.600000e+01  9.868756e-01
std    1.295281e+01  3.204164e+01  9.875798e+00
min    1.000000e+00  1.000000e+00  0.000000e+00
25%    1.200000e+01  2.800000e+01  0.000000e+00
50%    2.300000e+01  5.600000e+01  0.000000e+00
75%    3.400000e+01  8.400000e+01  0.000000e+00
max    4.500000e+01  1.110000e+02  5.568000e+03
```

We have 4.6176×10^6 records (data points) in this dataset, and 4 fields for each record.
In this dataset, there is 1 column indicating date, which is 'date'. And column 'units' will change with 'date'.

Part (b)

In [7]:
```
plt.boxplot(myDF['units'])
plt.show()
```



Most of data in the column 'units' are between 0 and 1000.

Part (c)

In [8]:
```
# There are too many zero units.
# So in the future analysis, we'd better eliminate all rows with 0 units.
myDF_without_zero_unit = myDF[myDF['units']!=0]
myDF_without_zero_unit = myDF_without_zero_unit.reset_index(drop = True)
print(myDF_without_zero_unit[:10])
```

```
         date  store_nbr  item_nbr  units
0  2012-01-01          1         9     29
1  2012-01-01          1        28      2
2  2012-01-01          1        51      1
3  2012-01-01          2         5    191
4  2012-01-01          2        44    215
5  2012-01-01          3         5    214
6  2012-01-01          3        45    112
7  2012-01-01          4         9     61
8  2012-01-01          4        27     21
9  2012-01-01          5        16     24
```

In [38]:

```
from pandas.plotting import scatter_matrix
scatter_matrix(myDF_without_zero_unit, figsize=(6, 6), alpha=0.5, diagonal='kde')
plt.show()
```