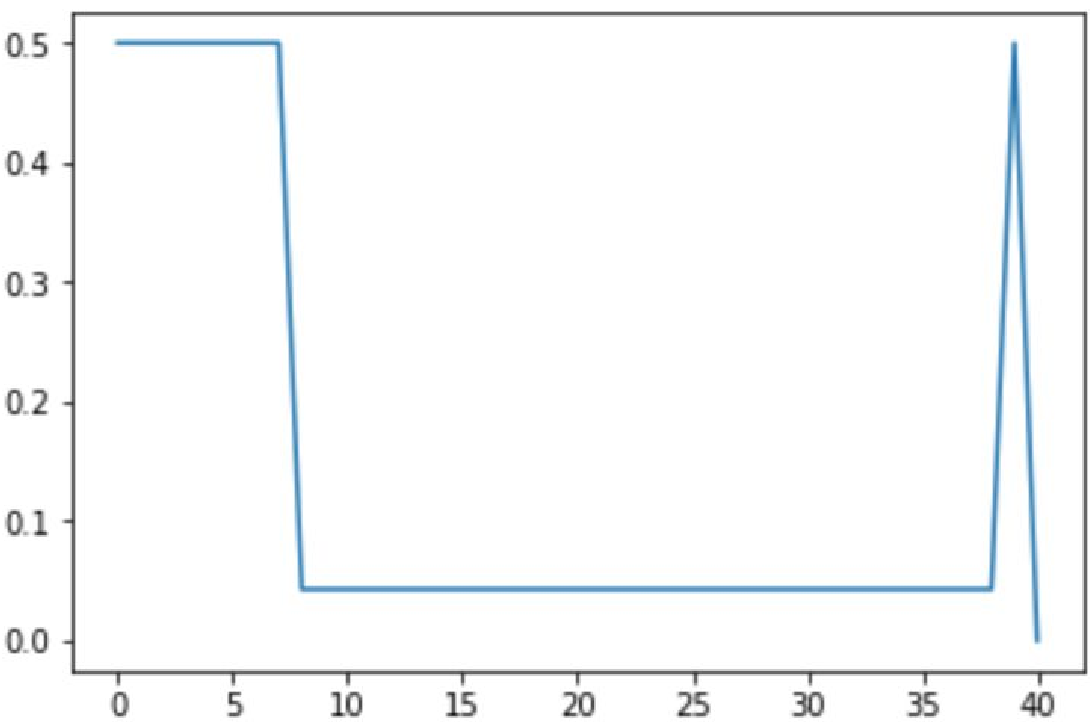
1.1

```
import numpy as np
import matplotlib.pyplot as plt
import random
get_ipython().magic(u'matplotlib inline')

data = np.loadtxt('Q1_data.txt',  delimiter=",",converters={ 4: lambda s: {b'Iris-setosa': 1,
b'Iris-versicolor': -1}[s]})
X = data[:,0:4]
Y = data[:,4]
train_index = range(0,35)+range(50, 85)
test_index = range(35, 50)+range(85, 100)
trainx = X[train_index,:]
trainy = Y[train_index]
testx = X[test_index,:]
testy = Y[test_index]

w = np.array([0 for i in range(len(trainx[0]))])
b = 0
lamb = 1
final_error = 1
index = 0;
errors = []

def helper(trainx, trainy):
    predict = []
    cal = np.dot(trainx, w)+b;
    for i in range(0, len(cal)):
        if cal[i] >= 0:
            predict.append(1)
        else:
            predict.append(-1)
    accuracy = predict == trainy
    return 1-sum(accuracy)/float(len(cal))

while index < 100 and final_error != 0:
    index = index+1
    final_error = helper(trainx, trainy)
    errors.append(final_error)
    rand = random.randrange(0, len(trainx))
    tmp = np.dot(w.T, trainx[rand])+b;
    if(tmp>= 0):
        predict = 1
```

```
    else:
        predict = -1

    if(predict == trainy[rand]):
        continue
    else:
        w = w + lamb*(trainy[rand] - predict)*trainx[rand]
        b = b + lamb*(trainy[rand] - predict)

plt.plot([i for i in range(0, len(errors))], errors)
plt.savefig('Q1.png')
```



1.2

w = array([2.8,   8.6, -13.2, -5.4])
b = 2.0
Decision boundary: <[2.8,   8.6, -13.2, -5.4], x>+2.0

1.3

```
cal = np.dot(testx, w)+b
predict = []
for i in range(0, len(cal)):
    if(cal[i] >= 0):
```

```
        predict.append(1)
    else:
        predict.append(-1)
accuracy = sum(predict == testy)/float(len(testx))
predict = np.array(predict)
true_pos=0;
test_pos = (predict == 1).sum()
condition_pos = (testy == 1).sum()
for i in range(0,len(testy)):
    if(testy[i] == 1 and predict[i] == 1):
        true_pos = true_pos+1
precision = float(true_pos)/test_pos
recall = float(true_pos)/condition_pos
F_value = 2*precision*recall/(precision+recall)
```

Accuracy = 1.0
Precision = 1.0
Recall = 1.0
F_value = 1.0

2

1. $P(y=0 \mid x) = 1 - \dfrac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} = \dfrac{1}{1+e^{\alpha+\beta x}}$

2. $P(y=1 \mid x) = \dfrac{1}{1+e^{-(\alpha+\beta x)}}$ $\quad P(y=0 \mid x) = \dfrac{1}{1+e^{\alpha+\beta x}}$

   when $y=1$ $\quad [P(y=1 \mid x)]' = \dfrac{1}{1+e^{-(2\times 1-1)(\alpha+\beta x)}} = \dfrac{1}{1+e^{-(\alpha+\beta x)}}$

   when $y=0$ $\quad [P(y=0 \mid x)]' = \dfrac{1}{1+e^{-(2\times 0-1)(\alpha+\beta x)}} = \dfrac{1}{1+e^{\alpha+\beta x}}$

   Therefore $[P(y=1 \mid x)]^{y} \times [P(y=0 \mid x)]^{1-y} = \dfrac{1}{1+e^{-(2y-1)(\alpha+\beta x)}}$

3. decision boundary

   $\alpha+\beta x = 0$

3.1

$$\boxed{L(w)} = -\sum_i \left(y^i \ln p^i + (1-y^i)\ln(1-p^i)\right)$$

$$= -\sum_i \left(-y_i \ln\left(1+e^{-(w^T x^i+b)}\right) + (1-y^i)\left(-(w^T x^i+b) - \ln\left(1+e^{-(w^T x^i+b)}\right)\right)\right)$$

$$= \sum_i \left((1-y^i)(w^T x^i+b) + \ln\left(1+e^{-(w^T x^i+b)}\right)\right)$$

$$\boxed{\frac{dL(w)}{dw}} = (1-y^i)x^i + \frac{1}{1+e^{-(w^T x^i+b)}} * e^{-(w^T x^i+b)} * (-x^i)$$

$$= \sum_i \left(\frac{1}{1+e^{-(w^T x^i+b)}} - y^i\right)x^i$$

$$\boxed{\frac{dL(b)}{db}} = \sum_i \left((1-y^i) + \frac{1}{1+e^{-(w^T x^i+b)}} * e^{-(w^T x^i+b)} \times (-1)\right)$$

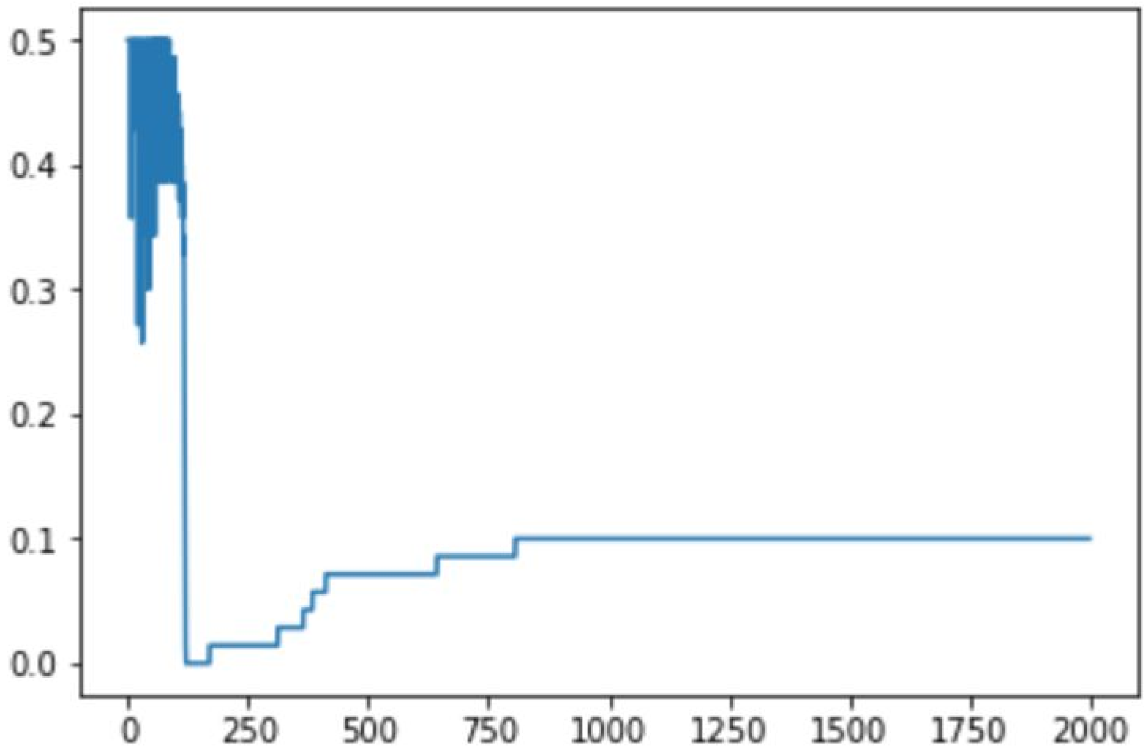
$$W_{t+1} = W_t - \alpha \cdot \frac{dL(W_t)}{dW_t} = W_t - \alpha \cdot \sum_i \left(\frac{1}{1+e^{-(w^T x^i+b)}} - y^i\right)x^i$$

3.2

```
import numpy as np
import matplotlib.pyplot as plt
import random
get_ipython().magic(u'matplotlib inline')

data = np.loadtxt('Q3_data.txt',  delimiter=",",converters={ 4: lambda s: (labels.index(s))})
X = data[:,0:4]
Y = data[:,4]

train_range = range(15,50)+range(65,100)
test_range = range(0,15)+range(50,65)
trainx = X[train_range,:]
```

```
trainy = Y[train_range]
testx = X[test_range,:]
testy = Y[test_range]
(trainx[1,:]).T
def func_P(x, w, b):
    return 1.0/(1+np.exp(-(np.dot(w.T,x)+b)))

def func_L(x, y, w, b):
    L = 0
    for i in range(0,len(Y)):
        p = func_P(x, w, b)
        L = L-Y[i]*np.log(p)+(1-Y[i])*np.log(1-p)
    return L

def derivative_by_w(x, y, w, b):
    L = 0
    for i in range(0, len(Y)):
#         L = L+(func_P(X[i,:], w, b)-Y[i])*X[i,:]
        L=L+(1-Y[i])*X[i,:]+func_P(X[i,:],w,b)*np.exp(-(np.dot(w.T, X[i,:]) + b))*(-X[i,:])
    return L

def derivative_by_b(x, y, w, b):
    b=0
    for i in range(0, len(y)):
        b=b+1-Y[i]+func_P(X[i,:],w,b)*np.exp(-(np.dot(w.T, X[i,:]) + b))*(-1)
    return b

def confidence(trainx, trainy, w, b):
    predict = []
    for i in range(0, len(trainy)):
        val = func_P(trainx[i,:], w, b)
        if(val >= 0.5):
            predict.append(1)
        else:
            predict.append(0)
    accuracy = sum(predict == trainy)/float(len(trainx))
    return 1-accuracy

w = np.array([random.random() for i in range(len(trainx[0]))])
b = random.random()
alpha = 0.01
index = 0
errors = []

while index < 2000:
```

```
    index = index+1
    error = confidence(trainx, trainy, w, b)
    errors.append(error)
    w2 = w-alpha*derivative_by_w(trainx, trainy, w, b)
    b = b-alpha*derivative_by_b(trainx, trainy, w, b)
    w = w2
    plt.plot([x for x in range(0,len(errors))], errors)
    plt.savefig('./HW3_Q31.png')
```



3.3

w = array([47.41417797, 26.24358662, 105.72353662, 63.66874736])

b = -992.91575589646197

Decision boundary = <[47.41417797, 26.24358662, 105.72353662, 63.66874736], x>-992.91

3.4

```
predict= []
val = np.dot(testx,w)+b
for i in range(0,len(testy)):
    val = func_P(testx[i,:], w, b)
    if(val >= 0.5):
        predict.append(1)
    else:
```

```
        predict.append(0)
predict = np.array(predict)
accuracy = sum(predict == testy)/float(len(testx))
error = 1-accuracy

true_pos=0;
test_pos = (predict == 1).sum()
condition_pos = (testy == 1).sum()
for i in range(0,len(testy)):
    if testy[i]==1 and predict[i]==1:
        true_pos=true_pos+1
precision = float(true_pos)/test_pos
recall = float(true_pos)/condition_pos
F_value = 2*precision*recall/(precision+recall)
```

Precision = 0.9285714285714286
Recall = 0.8666666666666667
F_value= 0.896551724137931
Accuracy = 0.9000000000000002

4

4.

1. $P(y=+1|x) = \frac{1}{1+e^{-(b+w^Tx)}}$

$P(y=-1|x) = \frac{1}{1+e^{(b+w^Tx)}}$

Therefore the sign before $(b+w^Tx)$ is opposite to the sign of y.

Therefore $P(y|x) = \frac{1}{1+e^{-y(b+w^Tx)}}$

2. $\frac{1}{1+e^{-(b+w^Tx)}} \geq 0.5$

$1+e^{-(b+w^Tx)} \leq 2$

$\ln(e^{-(b+w^Tx)}) \leq \ln(1)$

$-(b+w^Tx) \leq 0$

$w^Tx+b \geq 0$

$$y = \begin{cases} 1 & w^Tx+b \geq 0 \\ -1 & \text{else.} \end{cases}$$

decision boundary : $w^Tx+b=0$