problem2

December 17, 2020

Problem 2

[1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import csv
import math
```

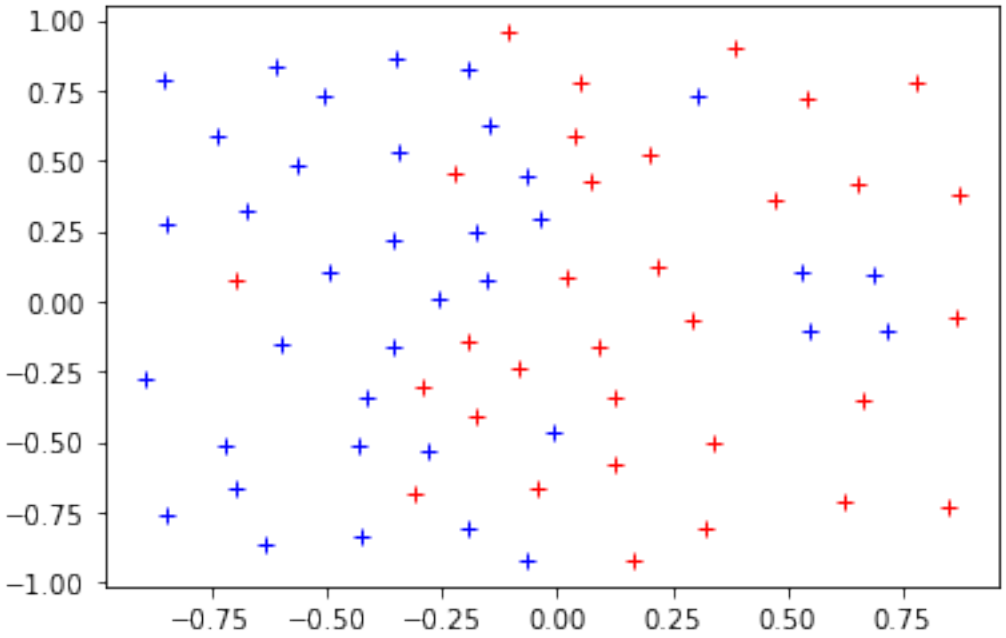
[2]:
```
with open('q2/data/x.dat') as csvfile:
    xreader = csv.reader(csvfile, delimiter = ' ')
    Xt = []
    for row in xreader:
        Xt.append([1] + [float(i) for i in row if i])

X = np.array(Xt)
print(X[:10], X.shape)
```

```
[[ 1.         -0.34792627  0.8625731 ]
 [ 1.         -0.14516129  0.62865497]
 [ 1.         -0.03456221  0.28947368]
 [ 1.         -0.14976959  0.07309942]
 [ 1.         -0.35253456 -0.16081871]
 [ 1.         -0.43087558 -0.51754386]
 [ 1.         -0.19124424 -0.80994152]
 [ 1.          0.16359447 -0.92105263]
 [ 1.         -0.0437788  -0.66374269]
 [ 1.         -0.17281106 -0.4122807 ]] (69, 3)
```

[3]:
```
with open('q2/data/y.dat') as csvfile:
    yreader = csv.reader(csvfile, delimiter = ' ')
    Yt = []
    for row in yreader:
        Yt.append([int(float(i)) for i in row if i])
Y = np.array(Yt)
```

[4]:
```
# Plot
x1_0 = [X[i, 1] for i in range(Y.size) if Y[i, 0] == 0]
x2_0 = [X[i, 2] for i in range(Y.size) if Y[i, 0] == 0]
```

```
plt.plot(x1_0, x2_0, 'b+')
x1_1 = [X[i, 1] for i in range(Y.size) if Y[i, 0] == 1]
x2_1 = [X[i, 2] for i in range(Y.size) if Y[i, 0] == 1]
plt.plot(x1_1, x2_1, 'r+');
```



[19]:
```
# Newton's method

# Weights
def weights(x, X, tau):
    return([math.exp(-sum((X[i] - x)**2) / (2 * tau**2)) for i in range(Y.
 ↪size)])

# Hypothesis function
def hyp(theta, x):
    try:
        h_t = 1 / (1 + math.exp(-np.dot(x, theta)))
    except OverflowError:
        h_t = 0
    return h_t

# Correction for each step
def step(X, Y, x, tau, theta):
    wt = weights(x, X, tau)
    h = [hyp(theta, X[i]) for i in range(Y.size)]
    z = np.dot(np.diag(wt), Y[:, 0] - h)
```

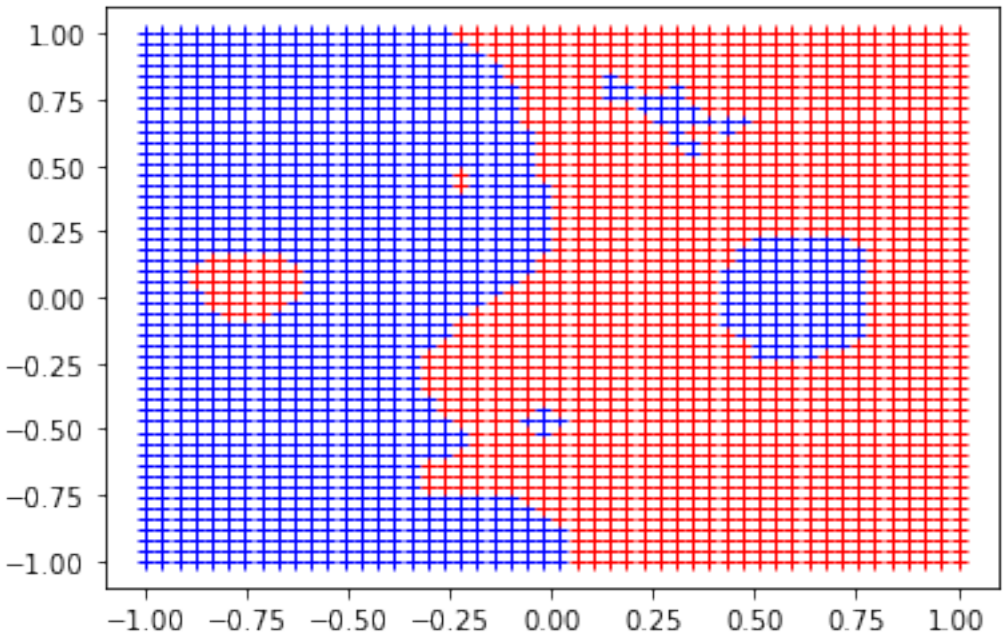
```
    D = np.diag([-wt[i] * h[i] * (1 - h[i]) for i in range(Y.size)])
    lambda_val = 0.0001

    grad_ll = np.dot(np.transpose(X), z) - [lambda_val * i for i in theta]
    H = np.dot(np.dot(np.transpose(X), D), X) - (np.identity(X.shape[1]) *␣
↪lambda_val)
    H_inv = np.linalg.inv(H)
    return(np.dot(H_inv, grad_ll))
```

[20]:
```
def estimate_theta(X, Y, x, tau):
    theta = [1, 1, 1] # initialize
    while True:
        newTheta = theta - step(X, Y, x, tau, theta)
        epsilon = np.ones(X.shape[1]) * 0.000001
        if all(np.absolute(newTheta - np.absolute(theta) < epsilon)):
            return(newTheta)
        theta = newTheta
        #print(theta)
```

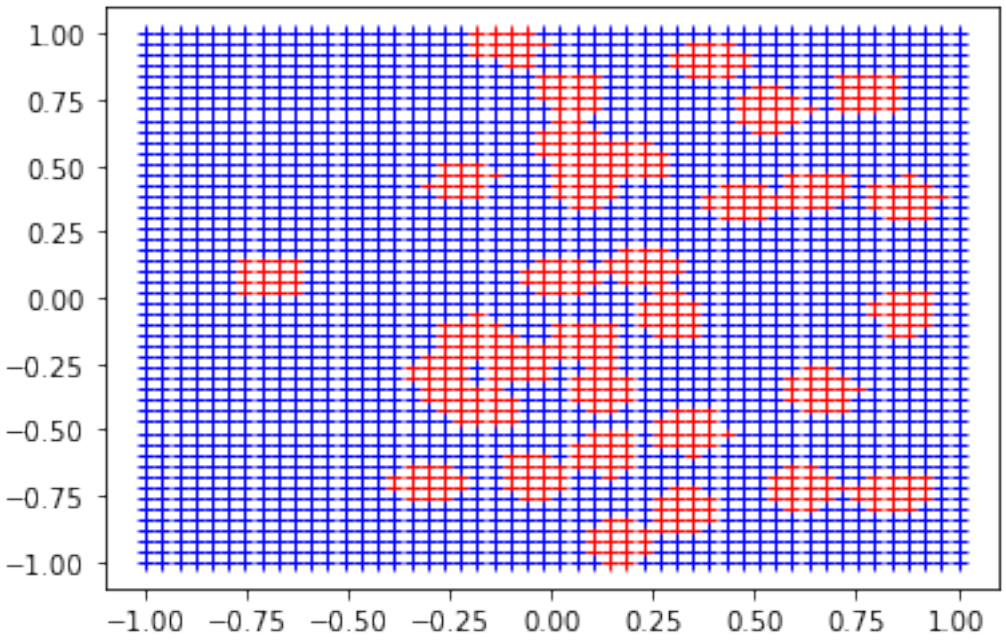
[21]:
```
def lwlr(X, Y, x, tau):
    theta = estimate_theta(X, Y, x, tau)
    try:
        h_t = 1 / (1 + math.exp(-np.dot(x, theta)))
    except OverflowError:
        h_t = 0
    return(1 if h_t > 0.5 else 0)
```

[29]:
```
# Plot decision boundary
def lwlr_plot(X, Y, tau):
    #np.arange(-1, 1, 0.1)
    ticks = np.linspace(-1, 1, 50)
    for i in ticks:
        x1_0 = []
        x2_0 = []
        x1_1 = []
        x2_1 = []
        for j in ticks:
            x = [1, i, j]
            pred = lwlr(X, Y, x, tau)
            if pred == 0:
                x1_0 = x1_0 + [i]
                x2_0 = x2_0 + [j]
            else:
                x1_1 = x1_1 + [i]
                x2_1 = x2_1 + [j]
        plt.plot(x1_0, x2_0, 'b+');
        plt.plot(x1_1, x2_1, 'r+');
```

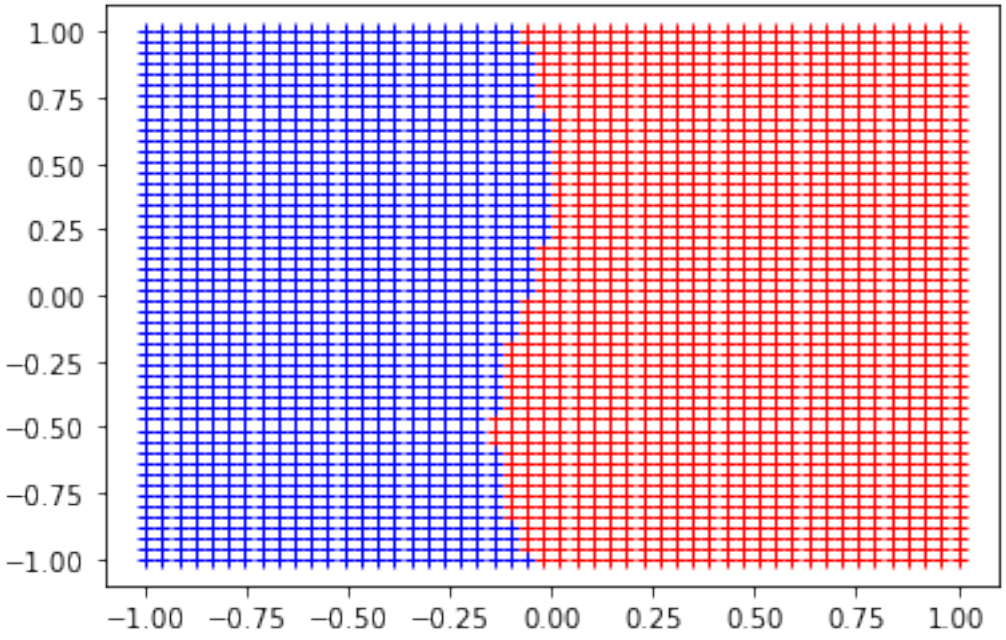
[30]:
```
tau = 0.1
lwlr_plot(X, Y, tau)
```



[28]:
```
tau = 0.01
lwlr_plot(X, Y, tau)
```



[27]:
```
tau = 0.5
lwlr_plot(X, Y, tau)
```



[ ]:
```

```