# Week 3

## Machine Learning in Mechanics

Divyansh Mittal , 2020CS10342

cs1200342@iitd.ac.in

## 1. 1 Nonlinear Regression

| | Date | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainTomorrow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01-12-2008 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | WNW | 20.0 | 24.0 | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.0 | NaN | 16.9 | 21.8 | No |
| 1 | 02-12-2008 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | WSW | 4.0 | 22.0 | 44.0 | 25.0 | 1010.6 | 1007.8 | NaN | NaN | 17.2 | 24.3 | No |
| 2 | 03-12-2008 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | WSW | 19.0 | 26.0 | 38.0 | 30.0 | 1007.6 | 1008.7 | NaN | 2.0 | 21.0 | 23.2 | No |
| 3 | 04-12-2008 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | E | 11.0 | 9.0 | 45.0 | 16.0 | 1017.6 | 1012.8 | NaN | NaN | 18.1 | 26.5 | No |
| 4 | 05-12-2008 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | NW | 7.0 | 20.0 | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.0 | 8.0 | 17.8 | 29.7 | No |

| | Date | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainTomorrow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01-12-2008 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | WNW | 20.0 | 24.0 | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.0 | NaN | 16.9 | 21.8 | 0 |
| 1 | 02-12-2008 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | WSW | 4.0 | 22.0 | 44.0 | 25.0 | 1010.6 | 1007.8 | NaN | NaN | 17.2 | 24.3 | 0 |
| 2 | 03-12-2008 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | WSW | 19.0 | 26.0 | 38.0 | 30.0 | 1007.6 | 1008.7 | NaN | 2.0 | 21.0 | 23.2 | 0 |
| 3 | 04-12-2008 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | E | 11.0 | 9.0 | 45.0 | 16.0 | 1017.6 | 1012.8 | NaN | NaN | 18.1 | 26.5 | 0 |
| 4 | 05-12-2008 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | NW | 7.0 | 20.0 | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.0 | 8.0 | 17.8 | 29.7 | 0 |

| | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustSpeed | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainTomorrow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 13.4 | 22.9 | 0.6 | NaN | NaN | 44.0 | 20.0 | 24.0 | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.0 | NaN | 16.9 | 21.8 | 0 |
| 1 | 7.4 | 25.1 | 0.0 | NaN | NaN | 44.0 | 4.0 | 22.0 | 44.0 | 25.0 | 1010.6 | 1007.8 | NaN | NaN | 17.2 | 24.3 | 0 |
| 2 | 12.9 | 25.7 | 0.0 | NaN | NaN | 46.0 | 19.0 | 26.0 | 38.0 | 30.0 | 1007.6 | 1008.7 | NaN | 2.0 | 21.0 | 23.2 | 0 |
| 3 | 9.2 | 28.0 | 0.0 | NaN | NaN | 24.0 | 11.0 | 9.0 | 45.0 | 16.0 | 1017.6 | 1012.8 | NaN | NaN | 18.1 | 26.5 | 0 |
| 4 | 17.5 | 32.3 | 1.0 | NaN | NaN | 41.0 | 7.0 | 20.0 | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.0 | 8.0 | 17.8 | 29.7 | 0 |

| | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustSpeed | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainTomorrow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 13.4 | 22.9 | 0.6 | 5.52876 | 7.568706 | 44.0 | 20.0 | 24.0 | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.00000 | 4.487216 | 16.9 | 21.8 | 0 |
| 1 | 7.4 | 25.1 | 0.0 | 5.52876 | 7.568706 | 44.0 | 4.0 | 22.0 | 44.0 | 25.0 | 1010.6 | 1007.8 | 4.43216 | 4.487216 | 17.2 | 24.3 | 0 |
| 2 | 12.9 | 25.7 | 0.0 | 5.52876 | 7.568706 | 46.0 | 19.0 | 26.0 | 38.0 | 30.0 | 1007.6 | 1008.7 | 4.43216 | 2.000000 | 21.0 | 23.2 | 0 |
| 3 | 9.2 | 28.0 | 0.0 | 5.52876 | 7.568706 | 24.0 | 11.0 | 9.0 | 45.0 | 16.0 | 1017.6 | 1012.8 | 4.43216 | 4.487216 | 18.1 | 26.5 | 0 |
| 4 | 17.5 | 32.3 | 1.0 | 5.52876 | 7.568706 | 41.0 | 7.0 | 20.0 | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.00000 | 8.000000 | 17.8 | 29.7 | 0 |

| | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustSpeed | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.569921 | 0.454139 | 0.001617 | 0.038129 | 0.536788 | 0.289062 | 0.153846 | 0.289157 | 0.701031 | 0.212121 | 0.452579 | 0.477080 | 0.888889 | 0.560902 | 0.490196 |
| 1 | 0.411609 | 0.503356 | 0.000000 | 0.038129 | 0.536788 | 0.289062 | 0.030769 | 0.265060 | 0.422680 | 0.242424 | 0.500832 | 0.488964 | 0.492462 | 0.560902 | 0.497549 |
| 2 | 0.556728 | 0.516779 | 0.000000 | 0.038129 | 0.536788 | 0.304688 | 0.146154 | 0.313253 | 0.360825 | 0.292929 | 0.450915 | 0.504244 | 0.492462 | 0.250000 | 0.590686 |
| 3 | 0.459103 | 0.568233 | 0.000000 | 0.038129 | 0.536788 | 0.132812 | 0.084615 | 0.108434 | 0.432990 | 0.151515 | 0.617304 | 0.573854 | 0.492462 | 0.560902 | 0.519608 |
| 4 | 0.678100 | 0.664430 | 0.002695 | 0.038129 | 0.536788 | 0.265625 | 0.053846 | 0.240964 | 0.814433 | 0.323232 | 0.504160 | 0.458404 | 0.777778 | 1.000000 | 0.512255 |

Figure 1: Pandas head after various modifications

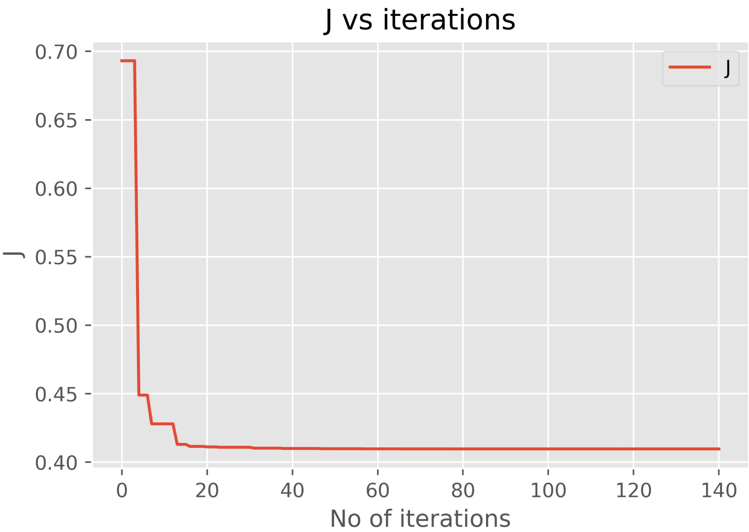


Figure 2: J vs iterations

## 2. Question 2

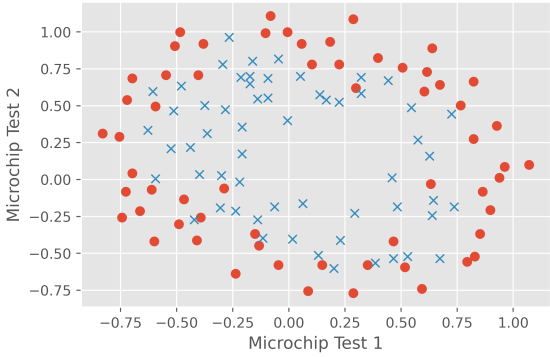


Figure 3: Plotted data

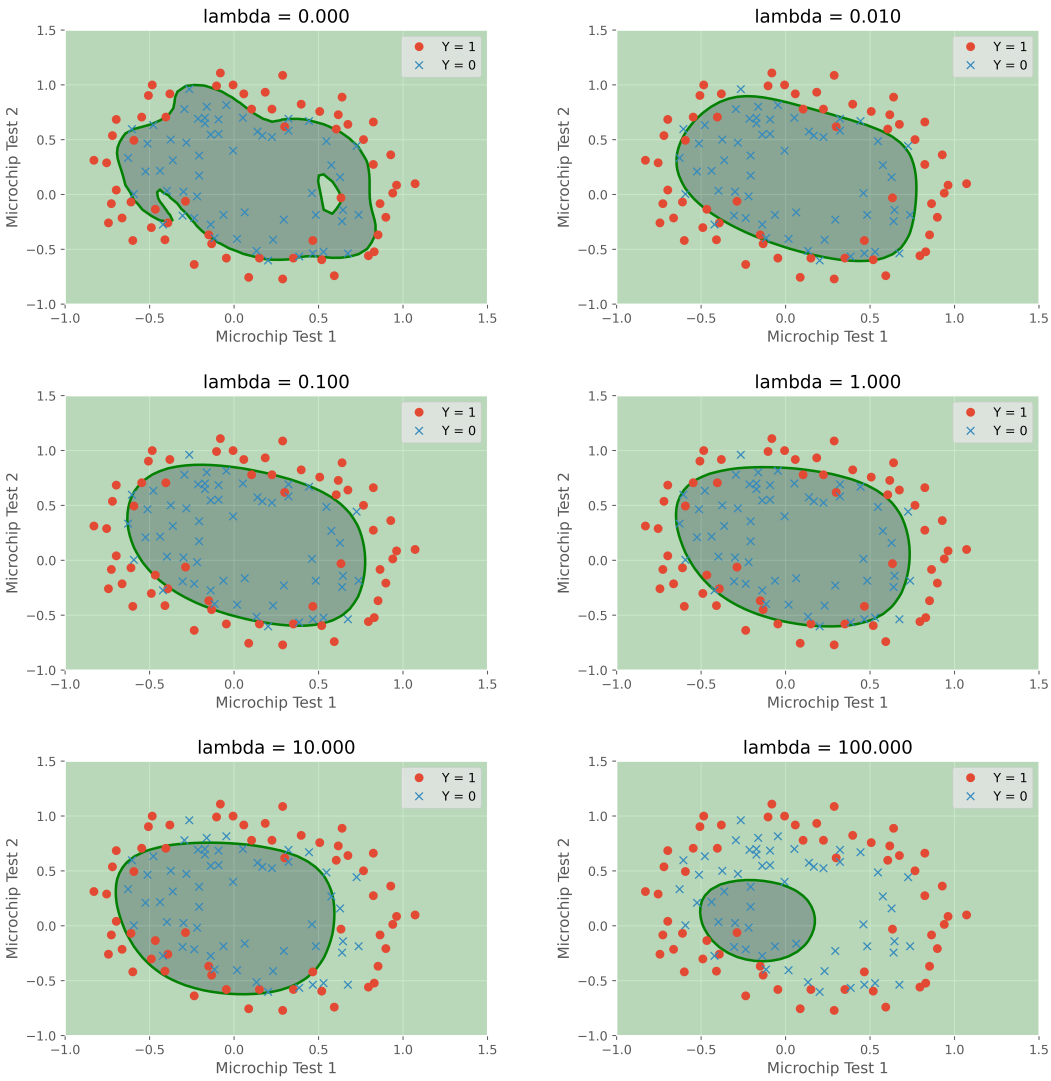


Figure 4: Contour for various values of lambda

With a higher value of lambda, we get a stronger regularisation effect. The weight of coeeficients tend to get smaller as lambda increases. If we take a too small lambda, we overfit the data. But if we take a lambda that is too large, we might underfit. Thus it becomes important to take a correct lambda for regression.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import optimize

plt.style.use('ggplot')

dat = np.loadtxt("nonLinearClass.txt" ,delimiter=',')

data_zero = dat[dat[:,2] == 0]
print(data_zero)

data_one = dat[dat[:,2] == 1]
print(data_one)

plt.plot(data_zero[:,0],data_zero[:,1],'o')
plt.plot(data_one[:,0],data_one[:,1],'x')


plt.xlabel('Microchip Test 1')
plt.ylabel('Microchip Test 2')

# Specified in plot order
plt.legend(['y = 1', 'y = 0'], loc='upper right')

def costFunctionReg(w,X,y,lambda_):
        y = np.round(np.array(y))
        m,col = X.shape
        x = X.T

        h = sigmoid(np.dot(w,x))

        J = (-1*np.dot(y,np.log(h)) - np.dot(1-y,np.log(1-h)))/m + lambda_*np.sum(np.square(w))/(2*m) # Cost 'J' should be a scalar


        grad =  (h - y)
        grad_J = np.dot(x,grad.T)

        grad =  np.divide(grad_J,m)    +  lambda_*w/(m)   # Gradient 'grad' should be a vector
        print(J)
        return J, grad
def sigmoid(z):

        return 1/(1 + np.exp(-z))

def mapFeature(x1,x2):
    a = len(x1)
    X = np.ones(a)
    for j in range(1,7):
        for i in range(j+1):
            t = (x1**(j-i)) *(x2**i)
            X= np.np.c_[X,t]
    return X

def minCostFun( w_ini, X_train, y_train, iters,lambda_):
        row,col = X_train.shape
        m = col
        options = {'maxiter':iters}
```

```

        res = optimize.minimize(costFunctionReg,w_ini,(X_train,y_train,lambda_),jac
    = True,method='TNC',options = options)

        cost = res.fun

        w_opt =  res.x     # Optimized weights rounded off to 3 decimal places
        return w_opt

X1 = np.array(dat[:,0])
X2 = np.array(dat[:,1])

Y = dat[:,2]
m = X1.size

lambda_ = 0.001

X_set = mapFeature(X1,X2,6).T
w = minCostFun(np.zeros(28),X_set.T,Y,1000,lambda_)


u = np.linspace(-1, 1.5, 50) # 1D array from -1 to 1.5 based on the limits of data
v = np.linspace(-1, 1.5, 50)

z = np.zeros((u.size, v.size))
# Evaluate z = w*x over the grid
for i, ui in enumerate(u):
    for j, vj in enumerate(v):
        z[i, j] = np.dot(mapFeature(np.array([ui]),np.array([vj])), w)

z = z.T

plt.contour(u, v, z, levels=[0], linewidths=2, colors='g') # Plots contour lines
plt.contourf(u, v, z, levels=[np.min(z), 0, np.max(z)], cmap='Greens', alpha=0.4) #
    Plots filled contours

plt.plot(data_zero[:,0],data_zero[:,1],'o', label = 'Y = 1')
plt.plot(data_one[:,0],data_one[:,1],'x', label = 'Y = 0')


# Specified in plot order

plt.xlabel('Microchip Test 1')
plt.ylabel('Microchip Test 2')
plt.legend()
plt.savefig("lambda001",dpi=500)
plt.title('lambda = %0.2f' % lambda_)
```

Listing 1: Python code