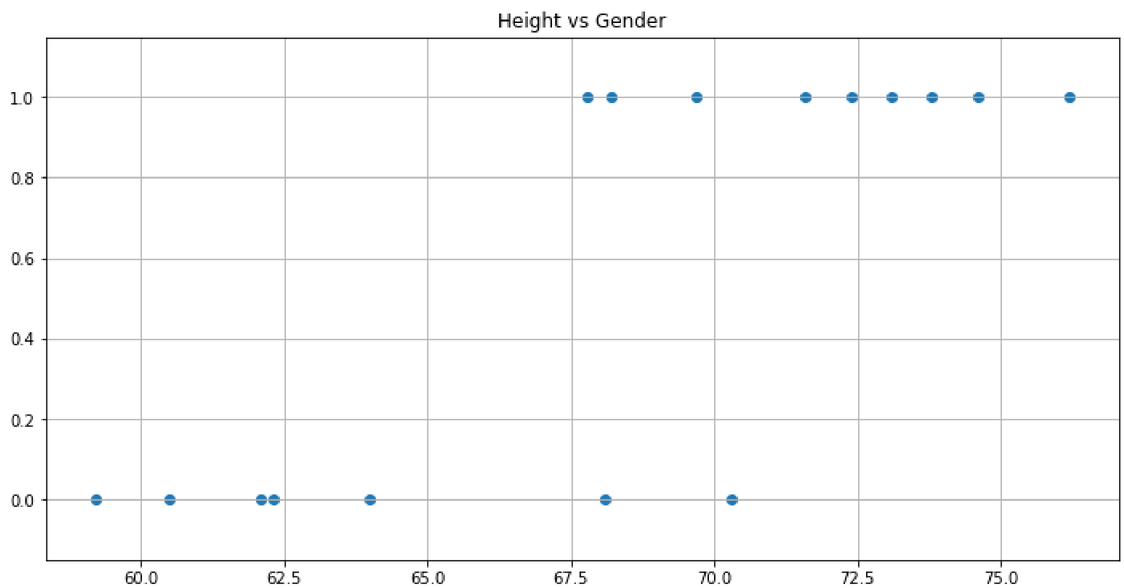
Logistic Regression

In [1]:
```
# Jupyter notebook specific
from IPython.display import Image
from IPython.core.display import HTML
from IPython.display import display_html
from IPython.display import display
from IPython.display import Math
from IPython.display import Latex
from IPython.display import HTML

# General useful imports
import numpy as np
from numpy import arange,linspace,mean, var, std,log
import matplotlib.pyplot as plt
from numpy.random import random, randint, uniform, choice, binomial, geometric, poisson
import math
from collections import Counter
import pandas as pd
%matplotlib inline




# Round to 4 decimal places
def round4(x):
    return round(float(x)+0.00000000001,4)
```

In [2]:
```
# Example 1 from slides

H = [59.2, 60.5, 62.1, 62.3, 73.8, 64.0, 71.6, 67.8,68.1,68.2,69.7,70.3, 72.4,73.1,74.6,76.2]

G = [0,0,0,0,1,0,1,1,0,1,1,0,1,1,1,1]

print()
print("Heights:", H)
print("Gender:", G)

plt.figure(figsize=(12,6))
plt.grid()
plt.title("Height vs Gender")
plt.ylim([-0.15,1.15])
plt.scatter(H,G)
plt.show()
```

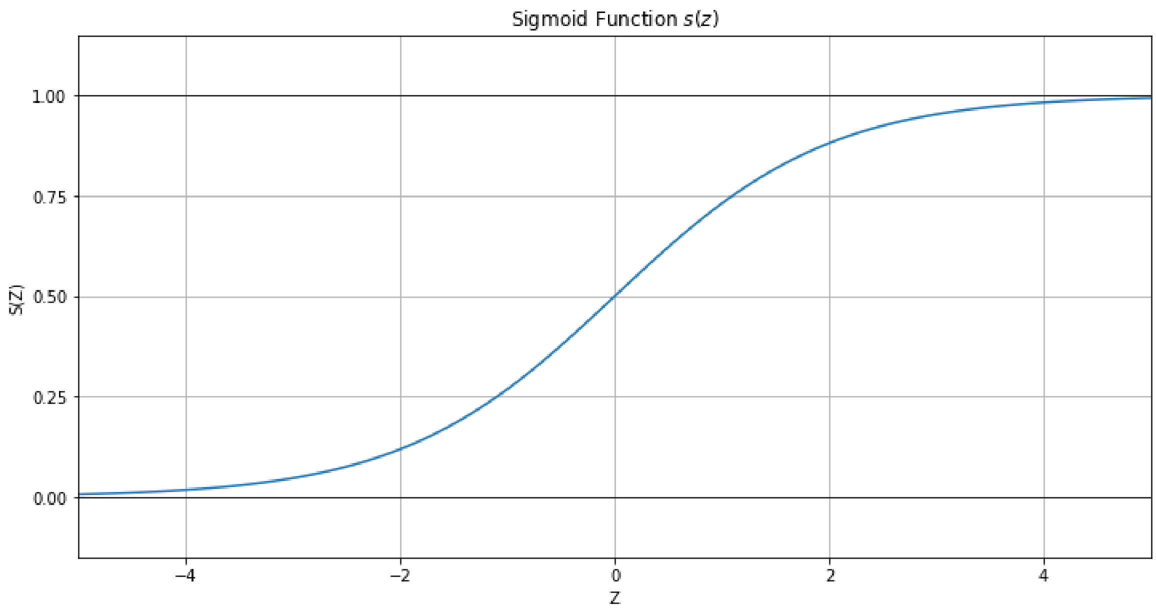
```
Heights: [59.2, 60.5, 62.1, 62.3, 73.8, 64.0, 71.6, 67.8, 68.1, 68.2, 69.7, 70.3, 72.4, 73.1,
74.6, 76.2]
Gender: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
```



In [3]:
```
mean(H)
```

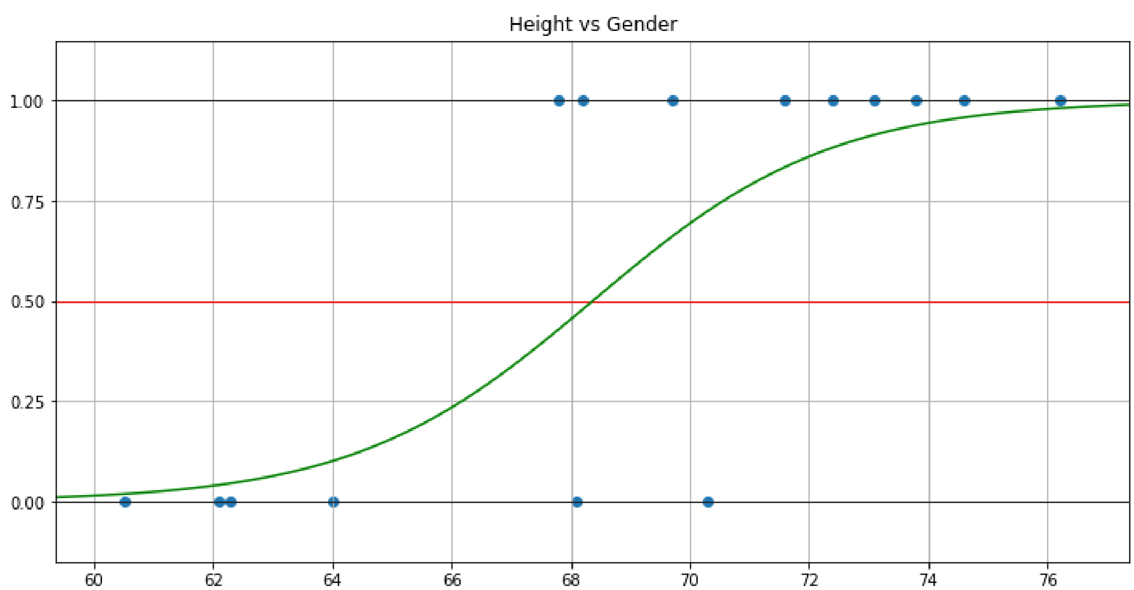
Out[3]:
```
68.36875
```

In [4]:
```
def s(z):
    return 1/(1+np.exp(-z))

X = np.linspace(-5,5,100)
Y = [s(x) for x in X]

plt.figure(figsize=(12,6))
plt.grid()
plt.title("Sigmoid Function $s(z)$")
plt.ylim([-0.15,1.15])
plt.xlim([-5,5])
plt.xlabel("Z")
plt.ylabel("S(Z)")
plt.yticks([0.0,0.25,0.5,0.75,1.0])
plt.plot([-5,5],[0,0],color='k',lw='0.75')
plt.plot([-5,5],[1,1],color='k',lw='0.75')
plt.plot(X,Y)
plt.show()
```

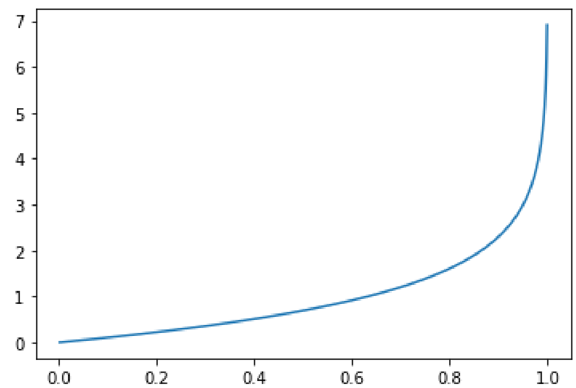


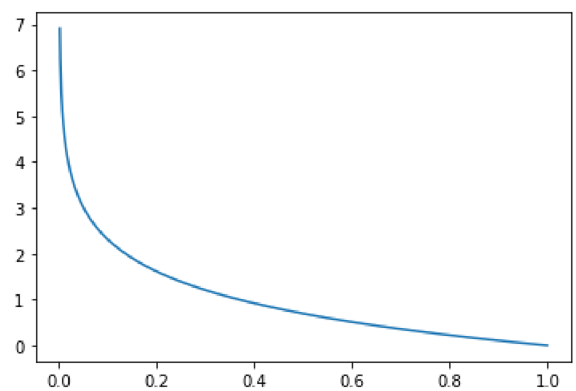
In [5]:
```
print()
print("Heights:", H)
print("Gender:", G)

mu = mean(H)
X = np.linspace(-9,9,100)
XG = [x+mu for x in X]
Y = [s(x/2) for x in X]

plt.figure(figsize=(12,6))
plt.grid()
plt.title("Height vs Gender")
plt.ylim([-0.15,1.15])
plt.xlim([min(XG),max(XG)])
plt.yticks([0.0,0.25,0.5,0.75,1.0])
plt.plot([min(XG),max(XG)],[0,0],color='k',lw='0.75')
plt.plot([min(XG),max(XG)],[0.5,0.5],color='r',lw='0.75',)
plt.plot([min(XG),max(XG)],[1,1],color='k',lw='0.75')
plt.plot(XG,Y,color='g')
plt.plot()
plt.scatter(H,G)
plt.show()
print()
```

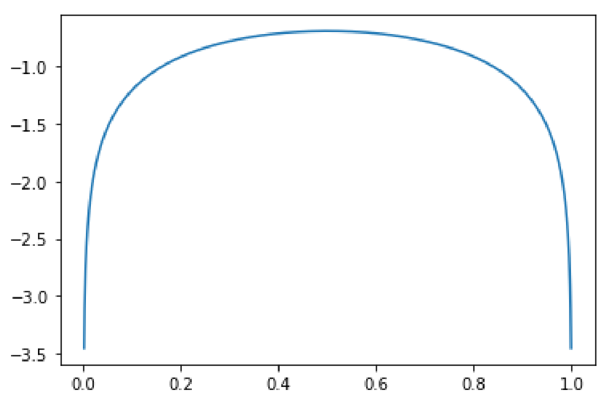
```
Heights: [59.2, 60.5, 62.1, 62.3, 73.8, 64.0, 71.6, 67.8, 68.1, 68.2, 69.7, 70.3, 72.4, 73.1,
74.6, 76.2]
Gender: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
```



In [6]:
```
# y can be either 0 or 1
def CE(yHat, y):
    if y == 1:
        return -log(yHat)
    else:
        return -log(1 - yHat)

X = np.linspace(0.001,0.999,1000)
Y = [CE(x,0) for x in X]
plt.plot(X,Y)
plt.show()

X = np.linspace(0.001,0.999,1000)
Y = [CE(x,1) for x in X]
plt.plot(X,Y)
plt.show()

def Cost(yHat,y) :
    return -CE(yHat,0)*y - CE(yHat,1)*(1-y)

X = np.linspace(0.001,0.999,1000)
Y = [Cost(x,0.5) for x in X]
plt.plot(X,Y)
plt.show()
```







In [7]:
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np

def s(z):
    return 1/(1+np.exp(-z))

def Cost(yHat, y):
    if y == 1:
        return -log(yHat)
    else:
        return -log(1 - yHat)

def h(b,m,xi):
    return b + m*xi

def J(b,m,X,Y):
    N = len(X)
    res = 0
    for k in range(len(X)):
        res += Cost(s(h(b,m,X[k])),Y[k])

    return res/N

def MSE(thetaHat0, thetaHat1,X,Y):
    Yhat = [thetaHat1 * x + thetaHat0 for x in X ]
    res = 0
    for k in range(len(X)):
        res += (Y[k] - Yhat[k])**2
    return res/len(X)
theta0  = - 19.0844
theta1 = 0.5836

def f(a,b):
    F = [45.2, 47.1, 47.5, 49.6, 49.8, 52.0, 54.3, 58.6, 63.2, 64.1]
    C = [7.8752, 8.117, 9.2009, 9.3167, 8.4564, 11.4075, 13.9236, 15.0762, 17.4678, 18.4362]
    return MSE(a,b,F,C)

def showIt(e,a):
    # limit in display

    xlo = -10
    xhi = 10
    ylo = 0.001
    yhi = 0.1

    b =  -0.836975399806
    m =  0.0046263029185

    zlo = 0
    zhi = 10
    # make the X,Y grid

    X = np.linspace(xlo, xhi, 100)
    Y = np.linspace(ylo, yhi, 100)
    X, Y = np.meshgrid(X, Y)

    def f(a,b):
        F = [45.2, 47.1, 47.5, 49.6, 49.8, 52.0, 54.3, 58.6, 63.2, 64.1]
        C = [7.8752, 8.117, 9.2009, 9.3167, 8.4564, 11.4075, 13.9236, 15.0762, 17.4678, 18.4362
]
        return MSE(a,b,F,C)

    # Yhat = [0.5836 * x - 19.0844 for x in X ]
    # calculate z = f(x,y)

    Z = np.zeros(shape=(100,100))

    # or, explicitly, with loops:
```

```
    for r in range(100):
        for c in range(100):
          #  print(X[r][c],Y[r][c])
          #  print(f(X[r][c],Y[r][c]))
            Z[r][c] = J(X[r][c],Y[r][c],X100,Y100)


    #print(Z)
    # Draw the figure

    fig = plt.figure(figsize=(14,12))
    ax = fig.gca(projection='3d')
    ax.view_init(elev=0, azim=180)         # <==  set viewing angle here
    ax.set_xlim(xlo,xhi)
    ax.set_ylim(ylo,yhi)
#   ax.set_zlim(zlo,zhi)
    ax.set_xlabel("Theta0")
    ax.set_ylabel("Theta1")
    ax.set_zlabel("MSE")

    '''
    # plot reference grid
    # planes
    ax.plot_surface(X, Y, 0, alpha=0.1)
    ax.plot_surface(X, 0, Y, alpha=0.1)
    ax.plot_surface(0, X, Y, alpha=0.1)
    # lines
    ax.plot([0,0],[0,0],[zlo,zhi],c='k', alpha=0.5)
    ax.plot([0,0],[ylo,yhi],[0,0],c='k', alpha=0.5)
    ax.plot([xlo,xhi],[0,0],[0,0],c='k', alpha=0.5)
    # origin point
    ax.scatter([0],[0],[0])
    '''

    # plot the surface
    ax.plot_surface(X, Y, Z, alpha=0.4)

    plt.show()

    fig = plt.figure(figsize=(14,12))
    ax = fig.gca(projection='3d')
    ax.view_init(elev=0, azim=110)         # <==  set viewing angle here
    ax.set_xlim(xlo,xhi)
    ax.set_ylim(ylo,yhi)
#   ax.set_zlim(zlo,zhi)
    ax.set_xlabel("Theta0")
    ax.set_ylabel("Theta1")
    ax.set_zlabel("MSE")

    '''
    # plot reference grid
    # planes
    ax.plot_surface(X, Y, 0, alpha=0.1)
    ax.plot_surface(X, 0, Y, alpha=0.1)
    ax.plot_surface(0, X, Y, alpha=0.1)
    # lines
    ax.plot([0,0],[0,0],[zlo,zhi],c='k', alpha=0.5)
    ax.plot([0,0],[ylo,yhi],[0,0],c='k', alpha=0.5)
    ax.plot([xlo,xhi],[0,0],[0,0],c='k', alpha=0.5)
    # origin point
    ax.scatter([0],[0],[0])
    '''

    # plot the surface
    ax.plot_surface(X, Y, Z, alpha=0.4)

    plt.show()

    fig = plt.figure(figsize=(14,12))
    ax = fig.gca(projection='3d')
    ax.view_init(elev=0, azim=80)         # <==  set viewing angle here
    ax.set_xlim(xlo,xhi)
```

```
    ax.set_ylim(ylo,yhi)
#   ax.set_zlim(zlo,zhi)
    ax.set_xlabel("Theta0")
    ax.set_ylabel("Theta1")
    ax.set_zlabel("MSE")

    '''
    # plot reference grid
    # planes
    ax.plot_surface(X, Y, 0, alpha=0.1)
    ax.plot_surface(X, 0, Y, alpha=0.1)
    ax.plot_surface(0, X, Y, alpha=0.1)
    # lines
    ax.plot([0,0],[0,0],[zlo,zhi],c='k', alpha=0.5)
    ax.plot([0,0],[ylo,yhi],[0,0],c='k', alpha=0.5)
    ax.plot([xlo,xhi],[0,0],[0,0],c='k', alpha=0.5)
    # origin point
    ax.scatter([0],[0],[0])
    '''

    # plot the surface
    ax.plot_surface(X, Y, Z, alpha=0.4)

    plt.show()




'''
showIt(5,160)
showIt(5,150)
showIt(5,147)
showIt(5,145)
showIt(5,140)
showIt(5,130)
showIt(5,120)
showIt(5,100)
showIt(5,80)
showIt(5,60)
showIt(5,40)
showIt(5,20)
showIt(5,10)
showIt(5,0)
showIt(5,340)
showIt(5,320)
showIt(5,300)
'''
```

Out[7]:

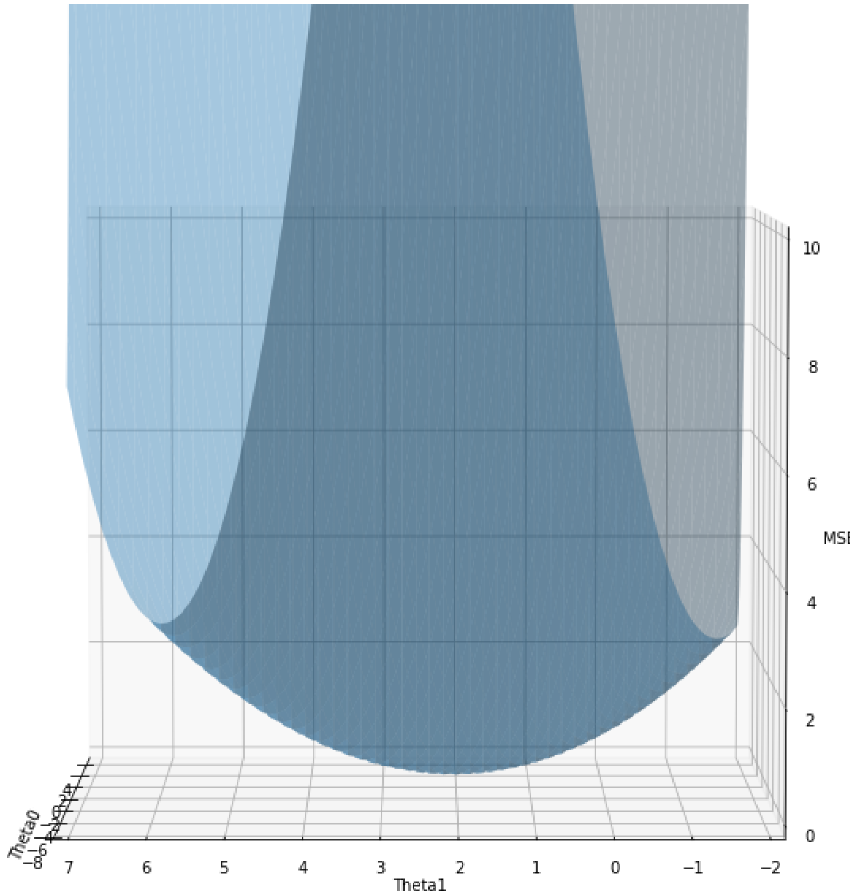
```
'\nshowIt(5,160)\nshowIt(5,150)\nshowIt(5,147)\nshowIt(5,145)\nshowIt(5,140)\nshowIt(5,130)\n
showIt(5,120)\nshowIt(5,100)\nshowIt(5,80)\nshowIt(5,60)\nshowIt(5,40)\nshowIt(5,20)\nshowIt
(5,10)\nshowIt(5,0)\nshowIt(5,340)\nshowIt(5,320)\nshowIt(5,300)\n'
```

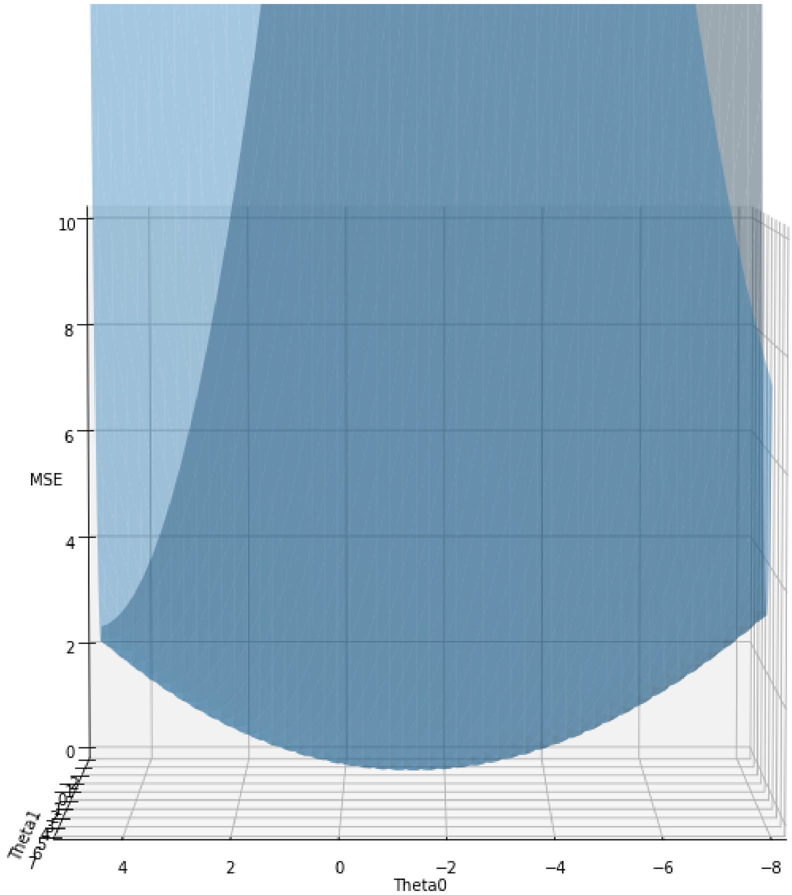
In [8]:
```
def MSE(thetaHat0, thetaHat1,X,Y):
    Yhat = [thetaHat1 * x + thetaHat0 for x in X ]
    res = 0
    for k in range(len(X)):
        res += (Y[k] - Yhat[k])**2
    return res/len(X)

theta0  = - 0.6364
theta1 = 1.5455


def f(a,b):
    X1 = [1,1.5,2,2]
    Y1 = [1,1.5,2,3]
    return MSE(a,b,X1,Y1)

f(theta0,theta1)
#f(theta0+0.01,theta1)
#f(theta0,theta1+0.01)
f(theta0+0.01,theta1+0.01)
```

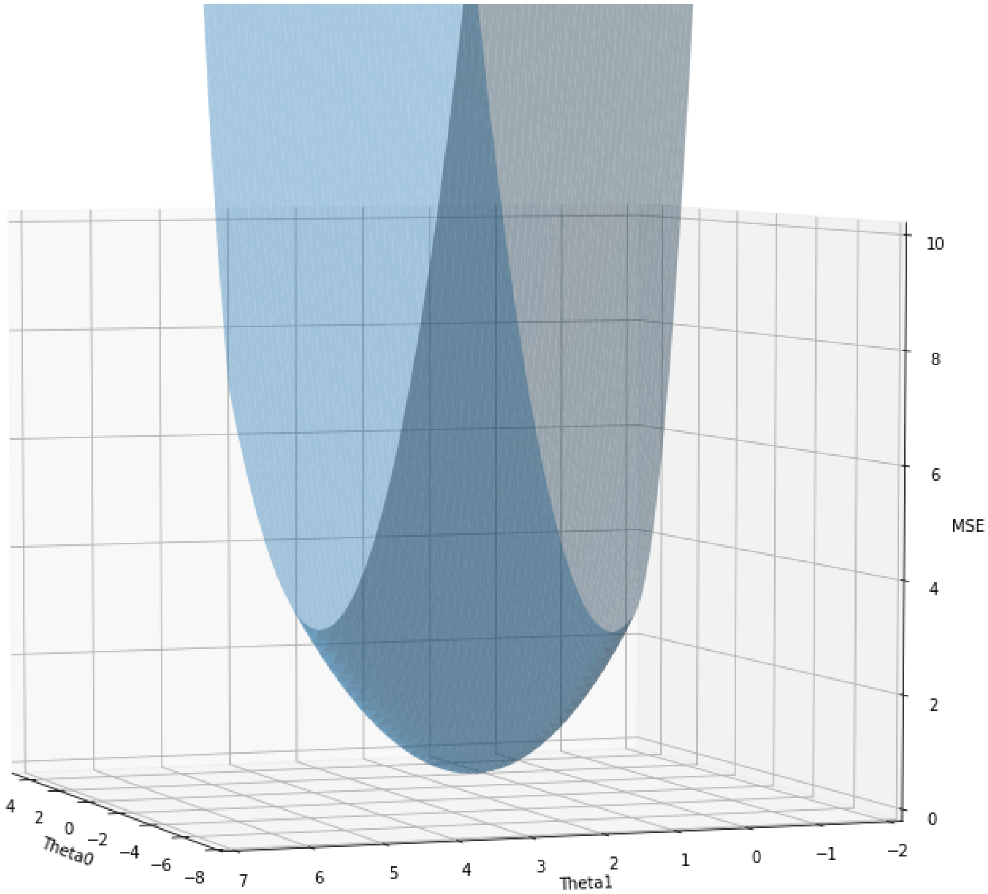
Out[8]: 0.137072013125

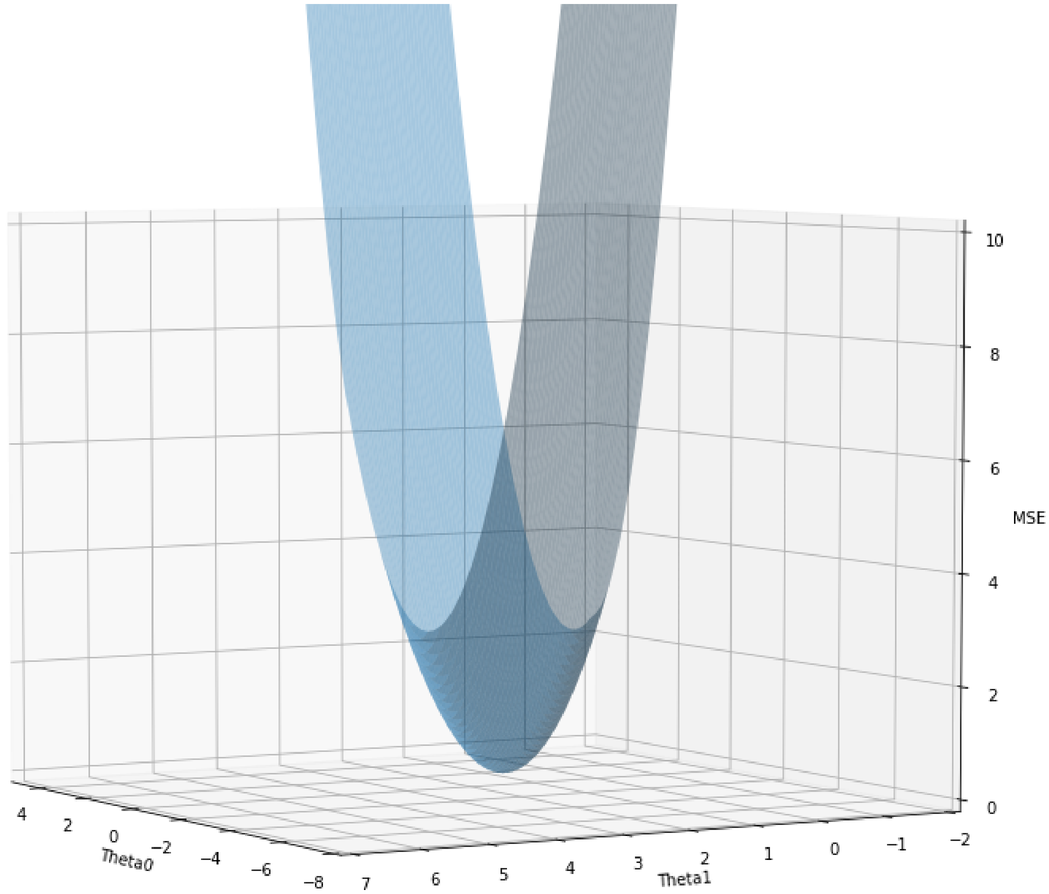
In [9]:
```
def MSE(thetaHat0, thetaHat1,X,Y):
    Yhat = [thetaHat1 * x + thetaHat0 for x in X ]
    res = 0
    for k in range(len(X)):
        res += (Y[k] - Yhat[k])**2
    return res/len(X)
theta0  = - 0.6364
theta1 = 1.5455


def f(a,b):
    X1 = [1,1.5,2,2]
    Y1 = [1,1.5,2,3]
    return MSE(a,b,X1,Y1)

def showIt(e,a):
    # limit in display
    theta0  = - 0.6364
    theta1 = 1.5455

    xlo = -8      # theta0
    xhi = 5
    ylo = -2      # theta1
    yhi = 7

    zlo = 0
    zhi = 10
    # make the X,Y grid

    X = np.linspace(xlo, xhi, 100)
    Y = np.linspace(ylo, yhi, 100)
    X, Y = np.meshgrid(X, Y)


    # Yhat = [0.5836 * x - 19.0844 for x in X ]
    # calculate z = f(x,y)

    Z = np.zeros(shape=(100,100))

    # or, explicitly, with loops:
    for r in range(100):
        for c in range(100):
          #  print(X[r][c],Y[r][c])
          #  print(f(X[r][c],Y[r][c]))
            Z[r][c] = f(X[r][c],Y[r][c])


    # Draw the figure

    fig = plt.figure(figsize=(14,12))
    ax = fig.gca(projection='3d')
    ax.view_init(elev=e, azim=a)          # <==  set viewing angle here
    ax.set_xlim(xlo,xhi)
    ax.set_ylim(ylo,yhi)
    ax.set_zlim(zlo,zhi)
    ax.set_xlabel("Theta0")
    ax.set_ylabel("Theta1")
    ax.set_zlabel("MSE")

    '''
    # plot reference grid
    # planes
    ax.plot_surface(X, Y, 0, alpha=0.1)
    ax.plot_surface(X, 0, Y, alpha=0.1)
    ax.plot_surface(0, X, Y, alpha=0.1)
    # lines
    ax.plot([0,0],[0,0],[zlo,zhi],c='k', alpha=0.5)
    ax.plot([0,0],[ylo,yhi],[0,0],c='k', alpha=0.5)
    ax.plot([xlo,xhi],[0,0],[0,0],c='k', alpha=0.5)
    # origin point
    ax.scatter([0],[0],[0])
```

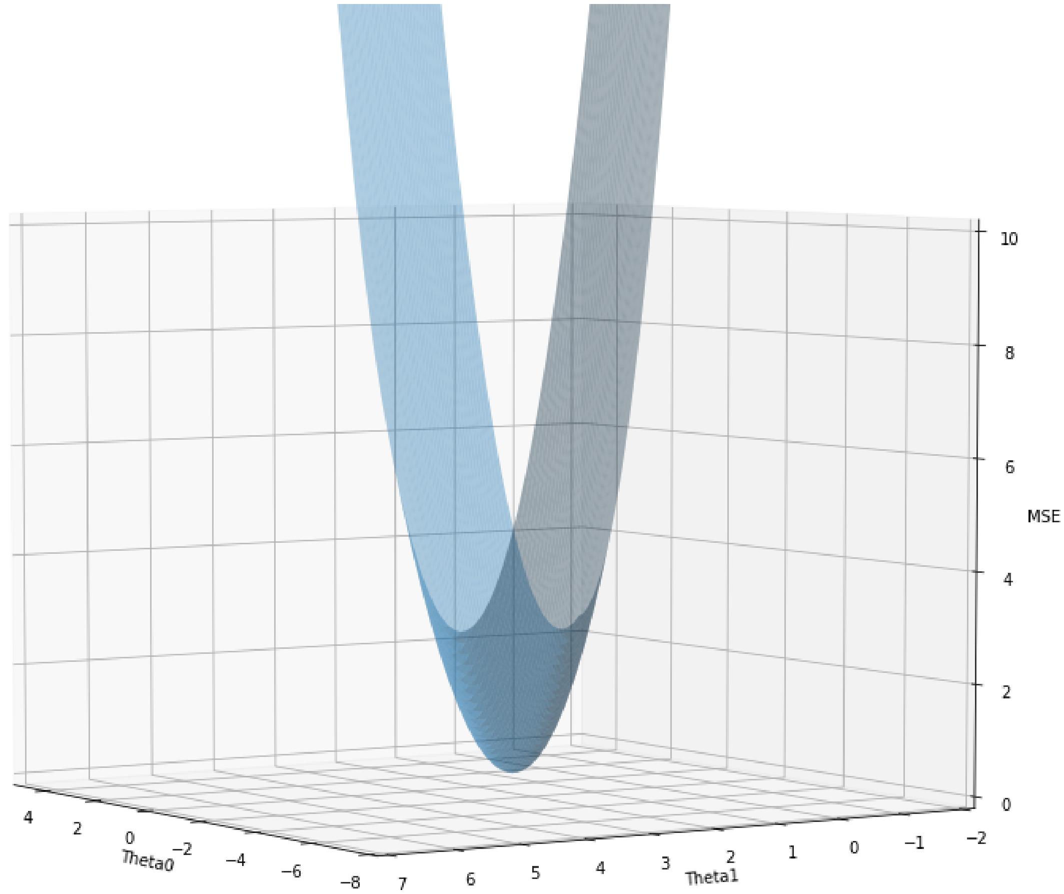
```
    '''

    # plot the surface
    ax.plot_surface(X, Y, Z, alpha=0.4)

    plt.show()


showIt(5,180)
showIt(5,90)
showIt(5,160)
showIt(5,150)
showIt(5,147)
showIt(5,145)
showIt(5,140)
showIt(5,130)
showIt(5,120)
showIt(5,100)
showIt(5,80)
showIt(5,60)
showIt(5,40)
showIt(5,20)
showIt(5,10)
showIt(5,0)
showIt(5,340)
showIt(5,320)
showIt(5,300)
```


MSE
Theta1
Theta0
10
8
6
4
2
0
7
6
5
4
3
2
1
0
−1
−2

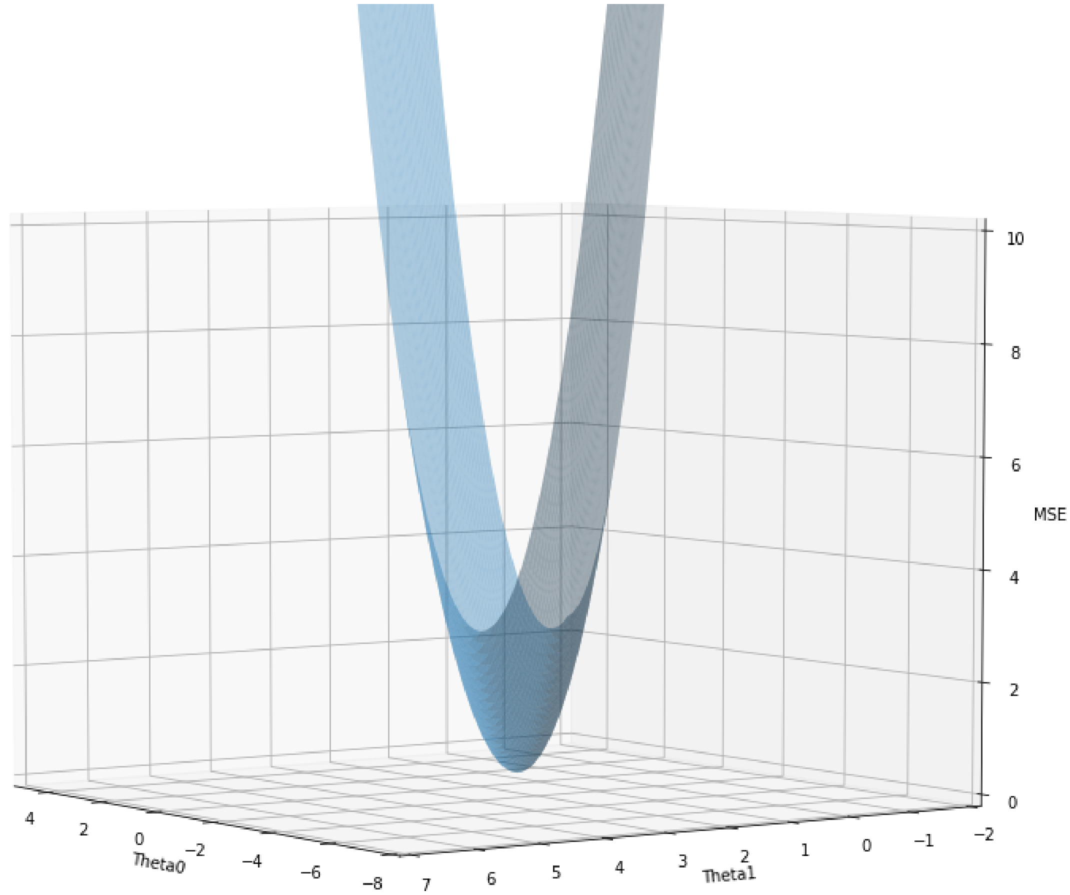

MSE
10
8
6
4
2
0
Theta1
4
2
0
−2
−4
−6
−8
Theta0


MSE
Theta0
Theta1
10
8
6
4
2
0
4 2 0 −2 −4 −6 −8
7 6 5 4 3 2 1 0 −1 −2

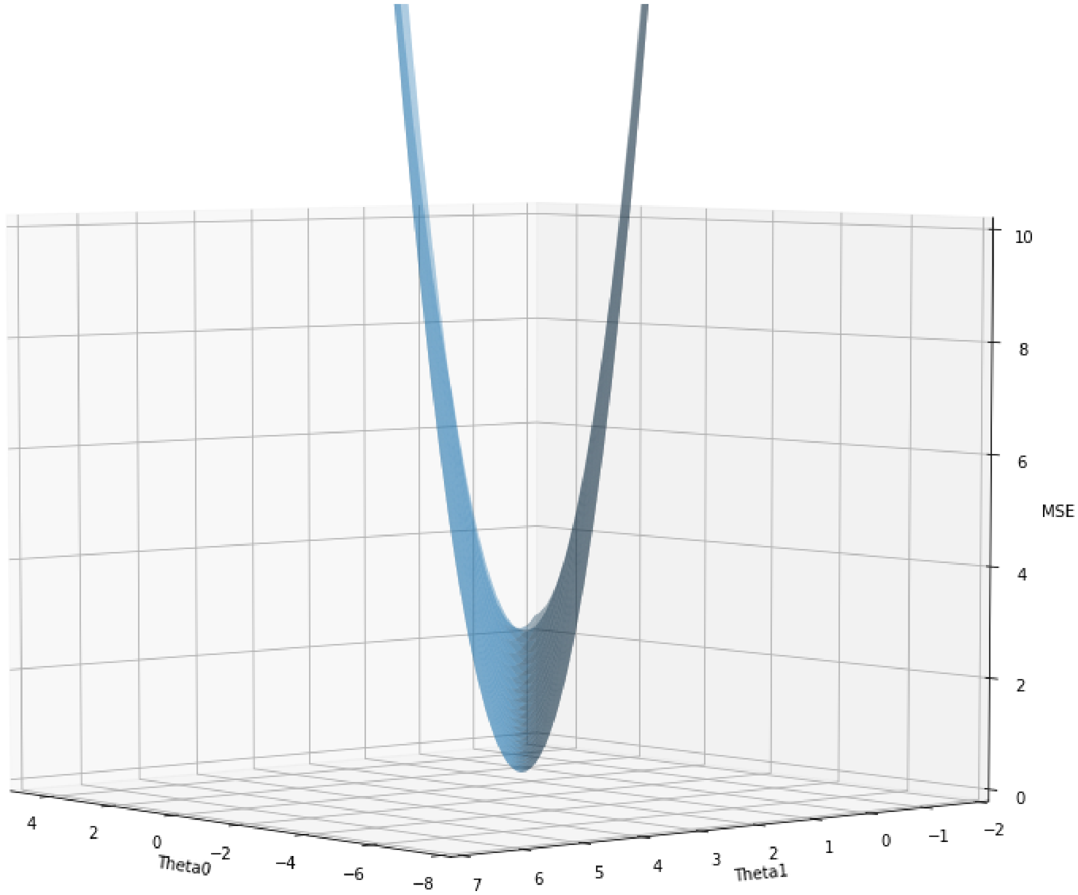

10
8
6
MSE
4
2
0
4
2
0
−2
Theta0
−4
−6
−8
7
6
5
4
3
2
Theta1
1
0
−1
−2



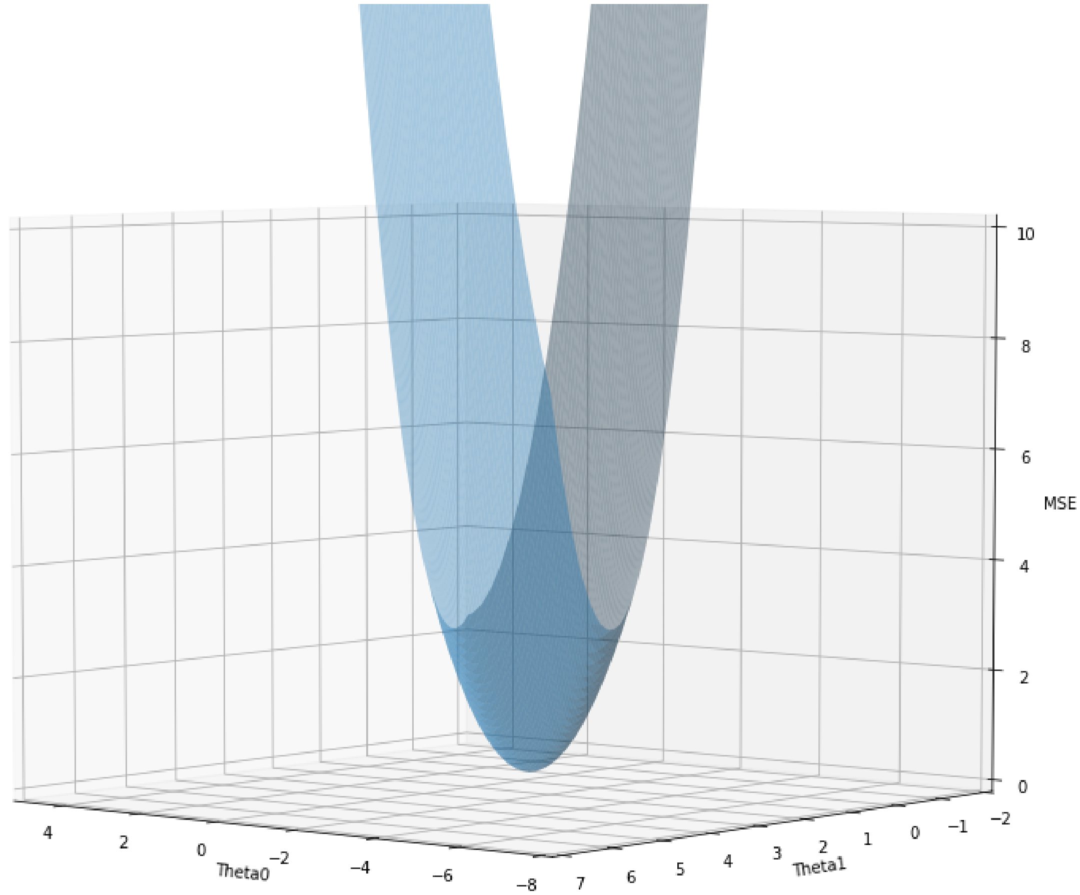
MSE
10
8
6
4
2
0
4
2
0
−2
−4
−6
−8
Theta0
7
6
5
4
3
2
1
0
−1
−2
Theta1



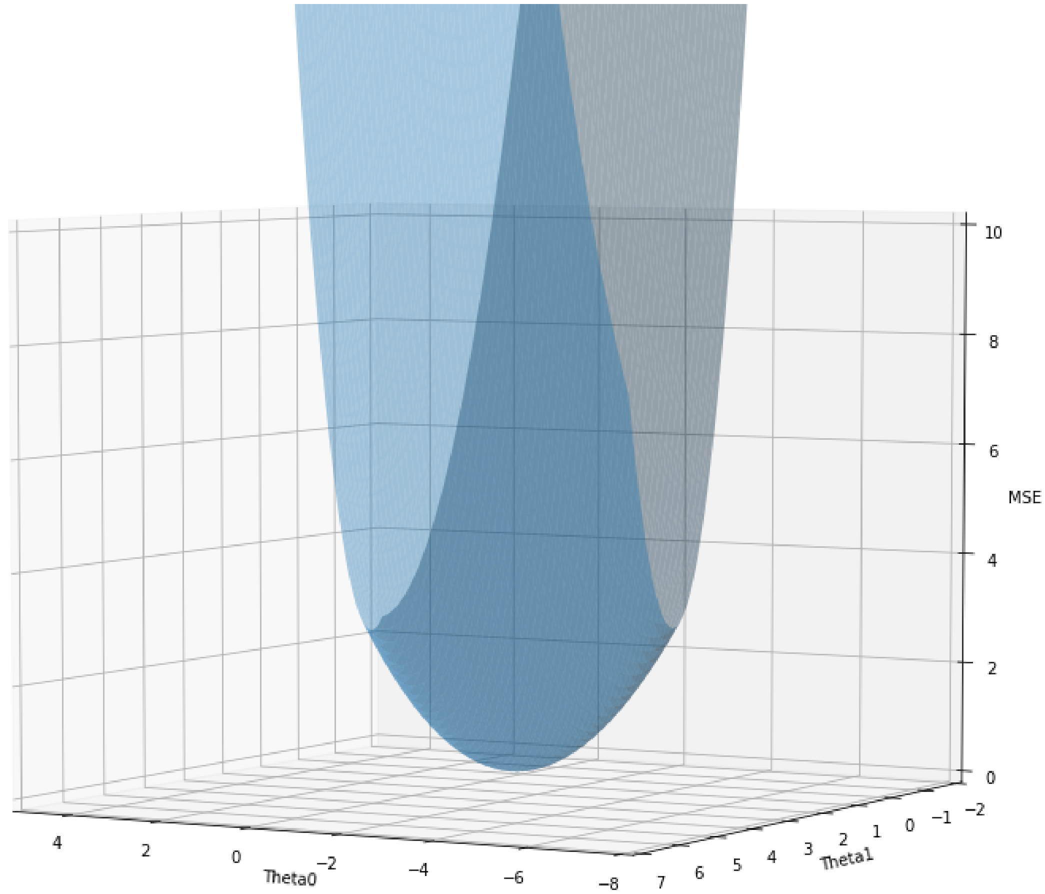
MSE
Theta0
Theta1
10
8
6
4
2
0
4 2 0 −2 −4 −6 −8
7 6 5 4 3 2 1 0 −1 −2



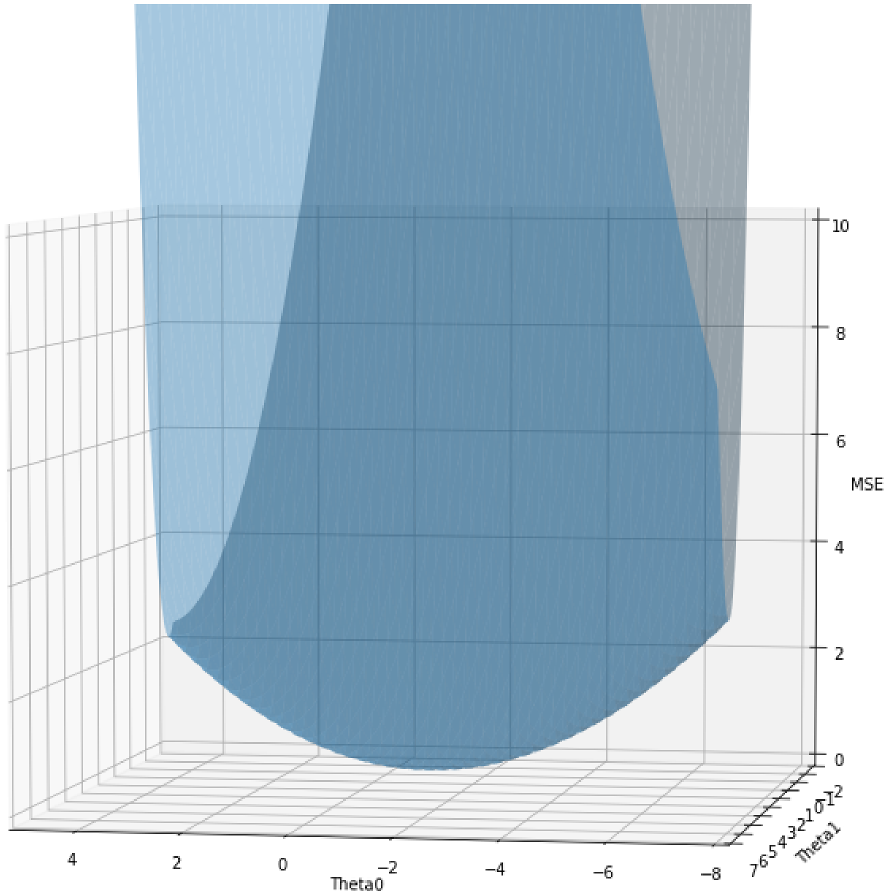
MSE
10
8
6
4
2
0
Theta0
4
2
0
−2
−4
−6
−8
Theta1
7
6
5
4
3
2
1
0
−1
−2

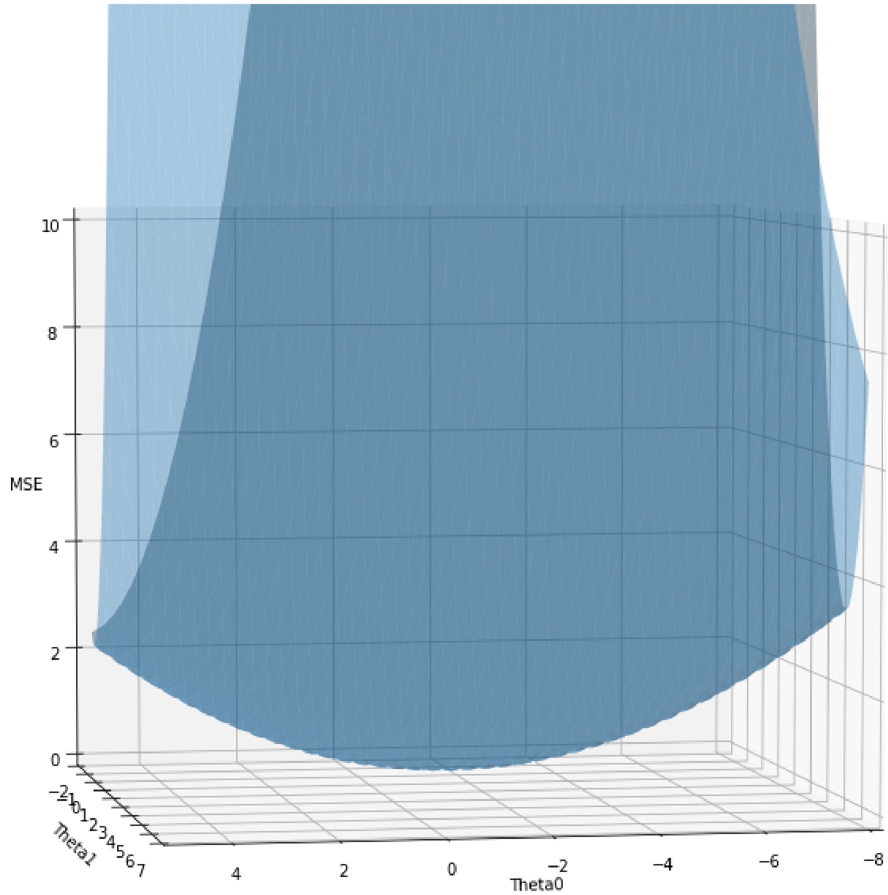


MSE
Theta0
Theta1
10
8
6
4
2
0
4
2
0
−2
−4
−6
−8
7
6
5
4
3
2
1
0
−1
−2

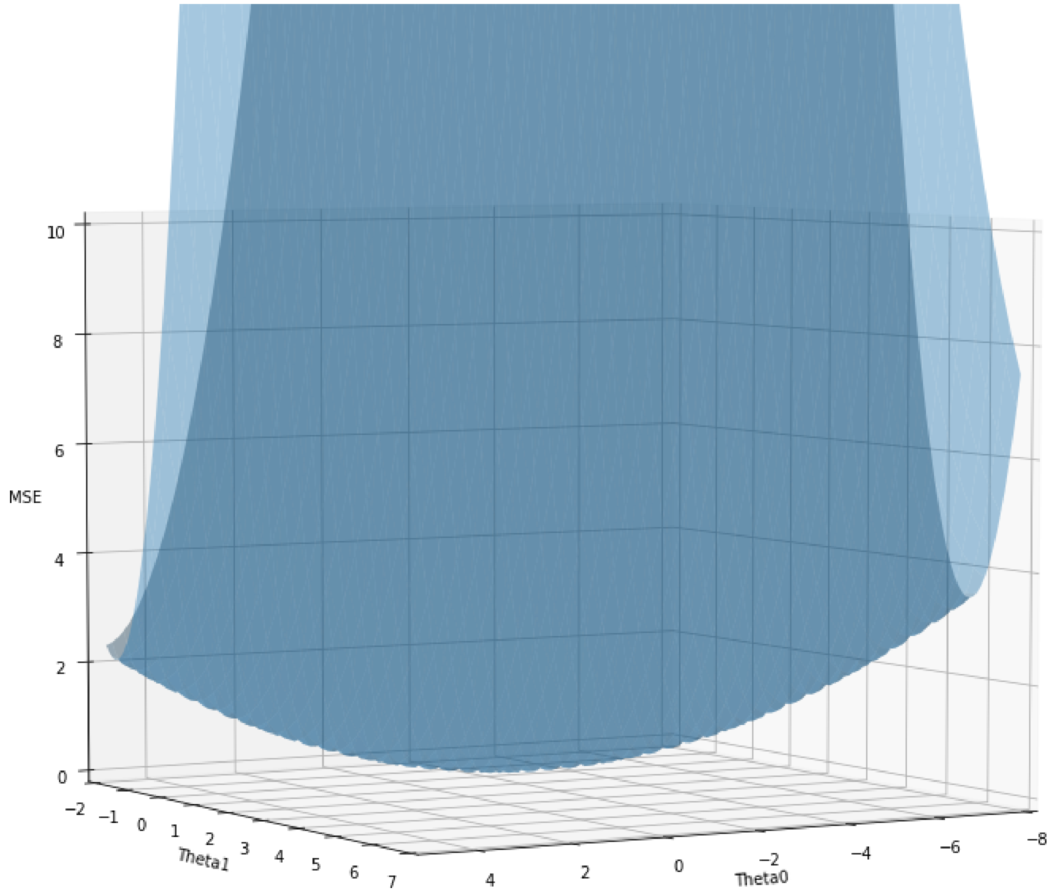


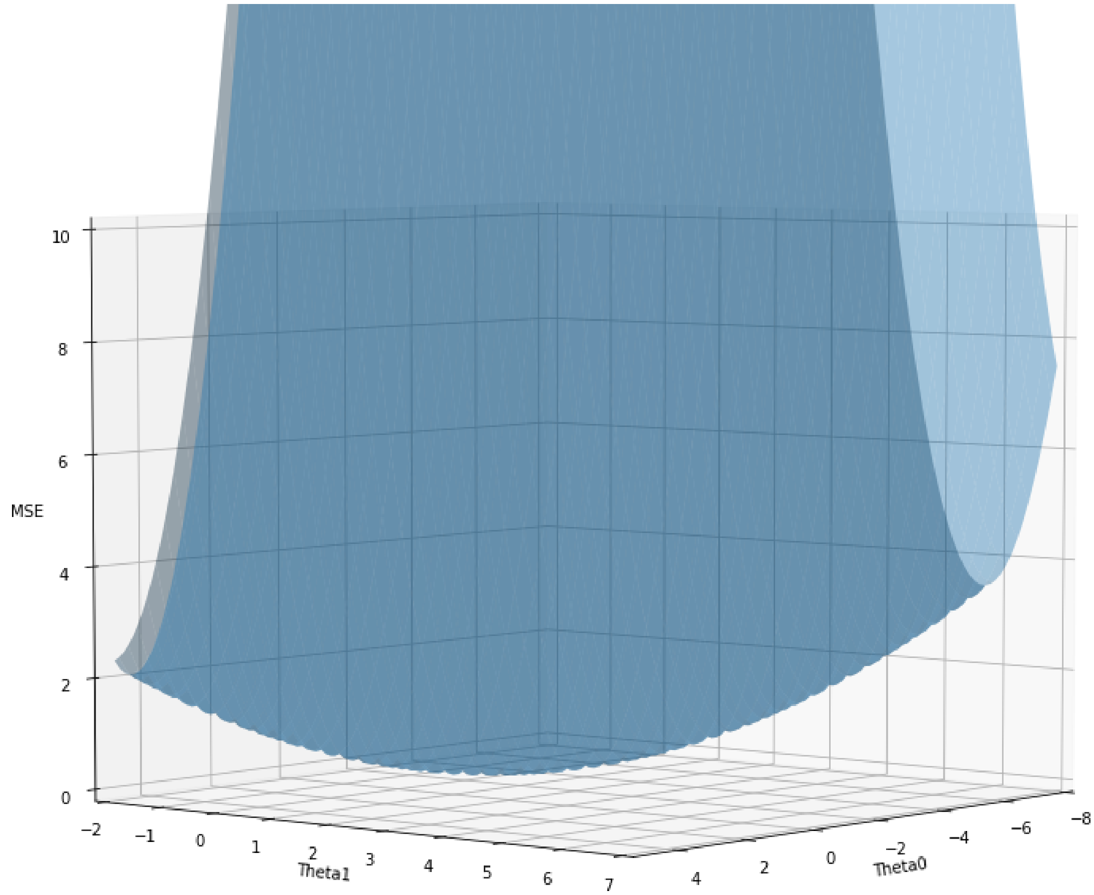
MSE
10
8
6
4
2
0
Theta0
4
2
0
−2
−4
−6
−8
Theta1
7
6
5
4
3
2
1
0
−1
−2

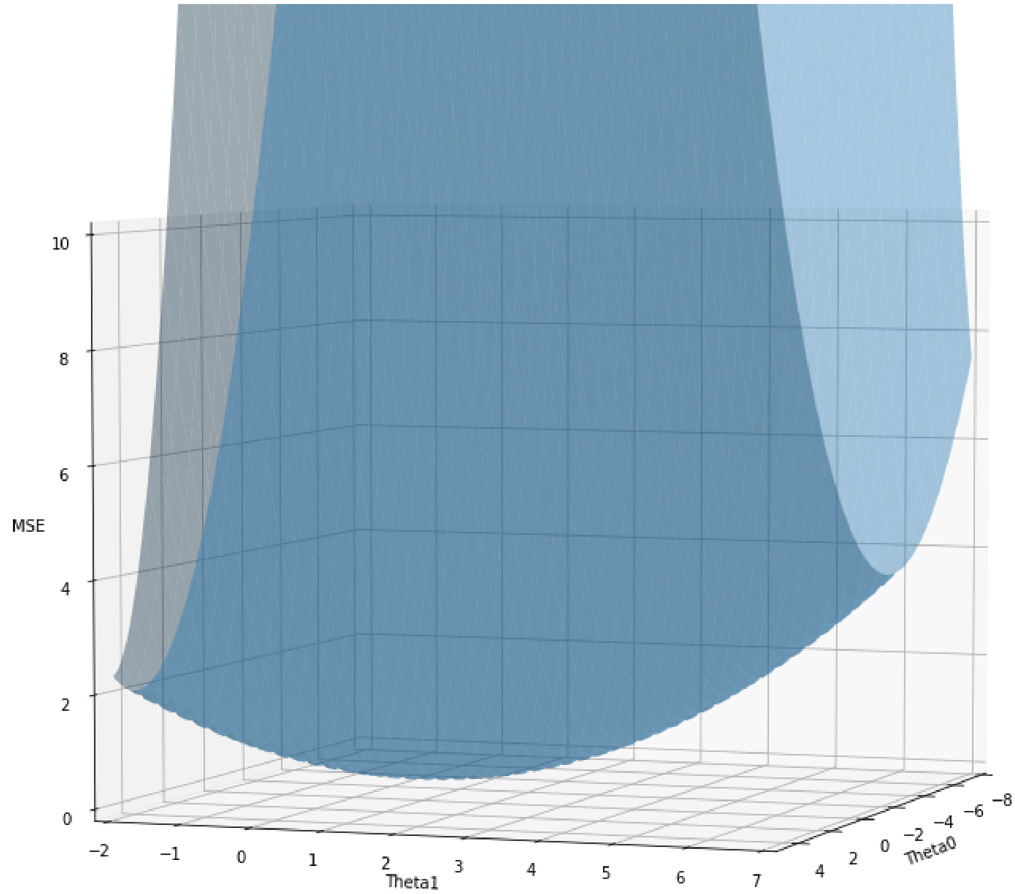

10
8
6
4
2
0
MSE
4
2
0
−2
−4
−6
−8
Theta0
7 6 5 4 3 2 1 0 −1 −2
Theta1

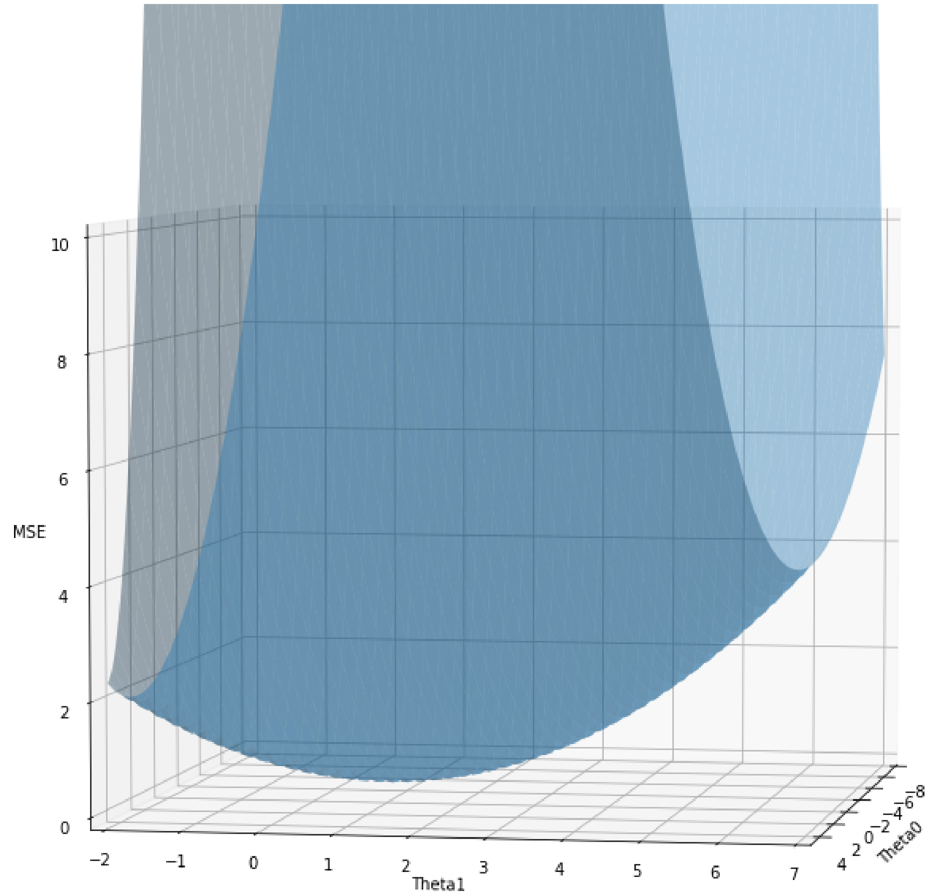


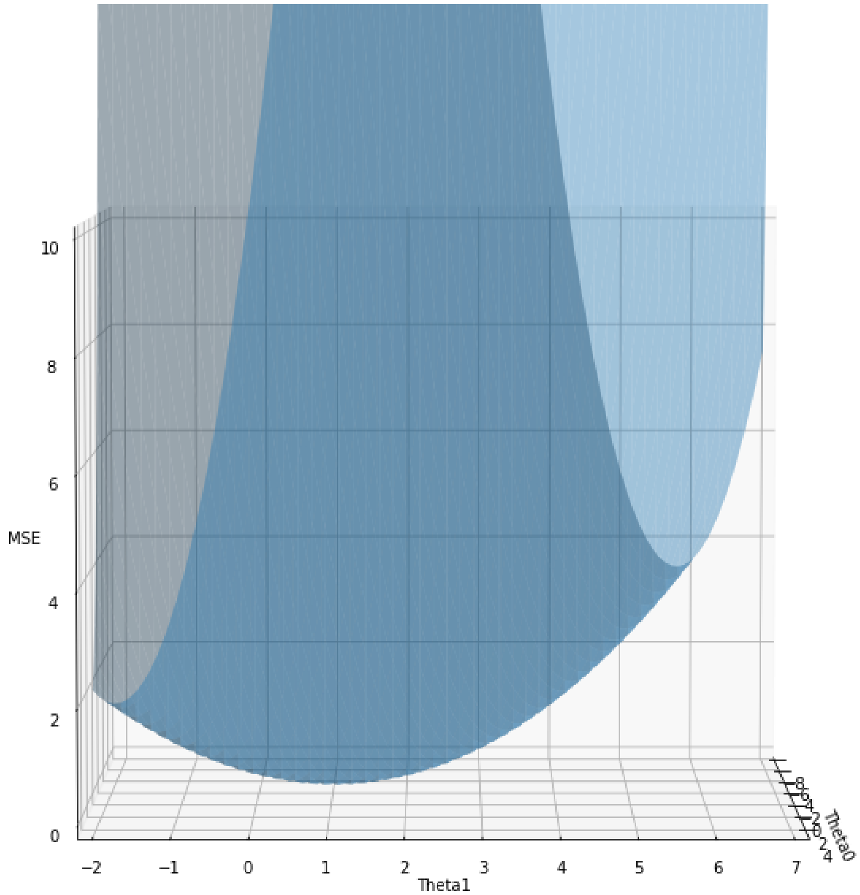
MSE
Theta1
Theta0


MSE
10
8
6
4
2
0
−2 −1 0 1 2 3 4 5 6 7
Theta1
4 2 0 −2 −4 −6 −8
Theta0

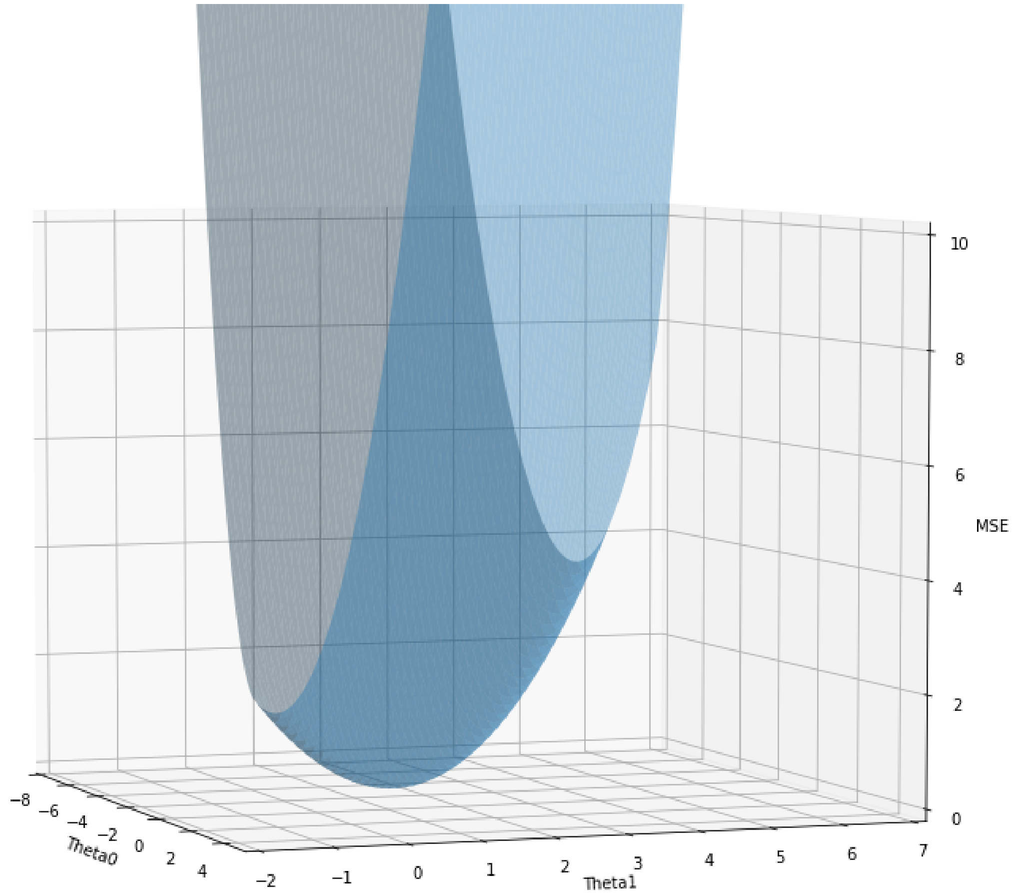

MSE
10
8
6
4
2
0
−2
−1
0
1
2
3
4
5
6
7
Theta1
4
2
0
−2
−4
−6
−8
Theta0

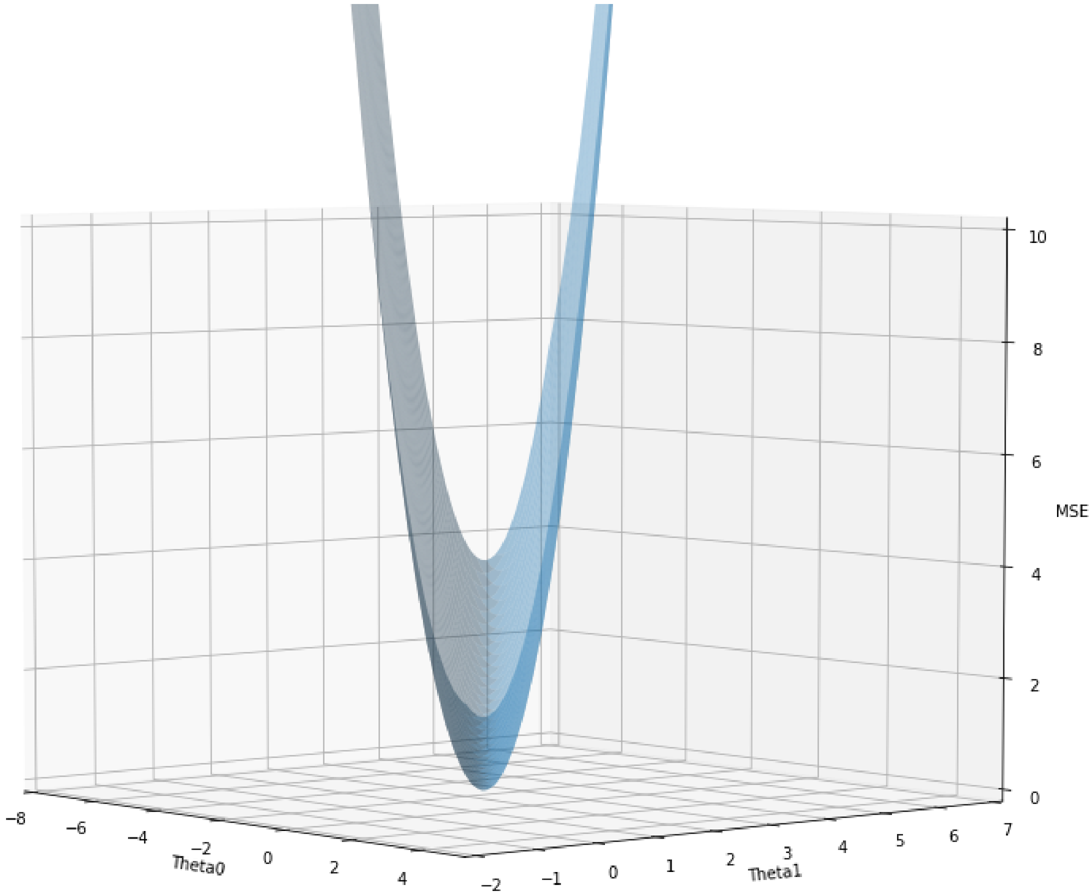

10
8
6
MSE
4
2
0
−2
−1
0
1
2
3
4
5
6
7
Theta1
4
2
0
−2
−4
−6
−8
Theta0


MSE
Theta1
Theta0
10
8
6
4
2
0
−2
−1
0
1
2
3
4
5
6
7
−8
−6
−4
−2
0
2
4

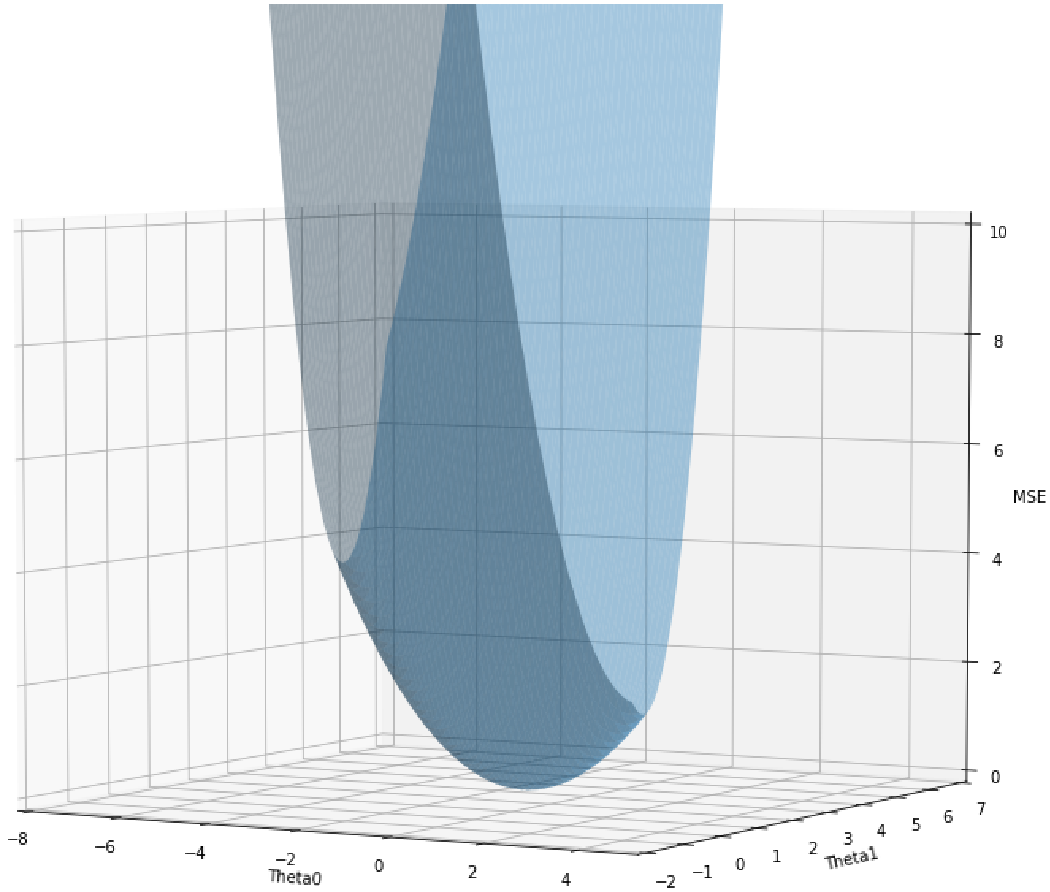


MSE
Theta1
Theta0
0
2
4
6
8
10
−2
−1
0
1
2
3
4
5
6
7


MSE
Theta0
Theta1



MSE
10
8
6
4
2
0
−8
−6
−4
−2
0
2
4
Theta0
−2
−1
0
1
2
3
4
5
6
7
Theta1



10
8
6
MSE
4
2
0
−8
−6
−4
−2
Theta0
0
2
4
−2
−1
0
1
2
3
4
5
6
7
Theta1