Dhanashree Kharate
3087817

Excersise 2.1

a. $g(w) = \frac{1}{2} q w^2 + r w + d$ Where q, r, d are constants

1st derivative

$$g'(w) = \frac{1}{2} \times 2 q w + r$$

2nd derivative

$$g''(w) = q$$

b. $g(w) = -\cos(2\pi w^2) + w^2$

$$g'(w) = \sin(2\pi w^2) \times 2\pi \times 2w + 2w$$
$$g'(w) = 4\pi w \sin(2\pi w^2) + 2w$$

$$g''(w) = 4\pi \left[\sin(2\pi w^2) + w\, 2\pi \times 2w \cos(2\pi w^2) \right] + 2$$
$$= 4\pi \sin(2\pi w^2) + 16\pi^2 w^2 \cos(2\pi w^2) + 2$$

c. $g(w) = \sum_{p=1}^{P} \log(1 + e^{-a_p w})$

$$g'(w) = \sum_{p=1}^{P} \frac{-e^{-a_p w}}{1+e^{-a_p w}} \cdot a_p$$

$$g'(w) = \sum_{p=1}^{P} \frac{-a_p e^{-a_p w}}{(1+e^{-a_p w})} \times \frac{e^{a_p w}}{e^{a_p w}}$$

$$g'(w) = -\sum_{p=1}^{P} \frac{a_p}{1+e^{a_p w}}$$

$$g''(w) = a_p \sum_{p=1}^{P} \frac{1}{(1+e^{a_p w})^2} e^{a_p w} a_p$$

$$= \sum_{p=1}^{P} \frac{e^{a_p w} a_p^2}{(1+e^{a_p w})^2}$$

Exercise 2.2

a. $g(w) = \frac{1}{2} w^T Q w + r^T w + d$

$$\nabla g(w) = \frac{1}{2} Q \, 2w + r^T$$

$$\nabla g(w) = Qw + r^T$$

$$\nabla^2 g(w) = Q$$

b. $g(w) = -\cos(2\pi w^T w) + w^T w$

$\nabla g(w) = \sin(2\pi w^T w)\, 2\pi\, 2w + 2w$

$\nabla g(w) = 4\pi w \sin(2\pi w^T w) + 2w$

$\nabla^2 g(w) = 4\pi\left[\sin(2\pi w^T w) + \cancel{\sin}\, 8\pi w \cdot \cos(2\pi w^T w)\, 2w\right] +$

$$\nabla^2 g(w) = 4\pi \sin(2\pi w^T w) + 8\cancel{4}\pi w^2 \cos(2\pi w^T w) + \overset{2}{2}$$

$\nabla^2 g(w) = (4\pi \sin(2\pi w^T w) + 2)\, I_{N \times N} +$

$\cos(2\pi w^T w)\,(4\pi)^2\, w \cdot w^T$

c. $g(w) = \sum_{p=1}^{P} \log(1 + e^{-a_p^T w})$

$$\nabla g(w) = \sum_{p=1}^{P} \frac{-a_p e^{-a_p^T w}}{1 + e^{-a_p^T w}}$$

$$= -a_p \sum_{p=1}^{P} \frac{1}{(1 + e^{-a_p^T w})} \frac{e^{-a_p^T w} \times e^{a_p^T w}}{e^{a_p^T w}}$$

$$= -a_p \sum_{p=1}^{P} \frac{1}{(e^{a_p^T w} + 1)}$$

$$\nabla^2 g(w) = \sum_{p=1}^{P} \frac{e^{a_p^T w}}{(1+e^{a_p^T w})^2} a_p a_p^T$$

Excersise 2.5

By 1st order taylor series approximation

$$h(w) = g(v) + \nabla^T g(v)\,(\bar{w}-\bar{v}) \qquad w = [w_1\ w_2 \cdots w_N]^T$$

$$h(w) = g(v) + \nabla^T g(v)\,\bar{w} - \nabla^T g(v)\,\bar{v}$$

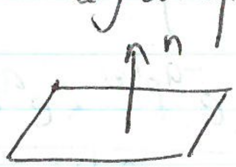
$$\nabla g(v) = \left[\frac{\partial}{\partial w_1} g(v)\ \ \frac{\partial}{\partial w_2} g(v) \cdots \frac{\partial}{\partial w_N} g(v)\right]^T$$

$$h(w) - \nabla^T g(v)\,\bar{w} \underbrace{- g(v) + \nabla^T g(v)\,\bar{v}}_{\gamma} = 0 \qquad (1)$$

$$n^T [h \quad \bar{w}] + \gamma = 0 \qquad (2)$$

tangent hyperplane equation

By comparing eq (1) and (2)



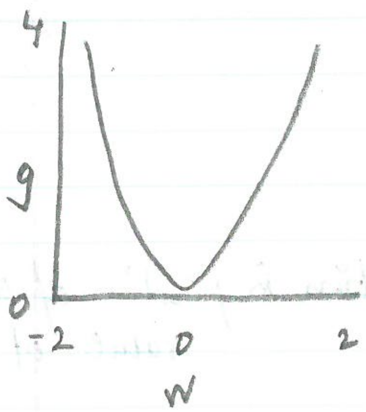
$$\bar{n} = \begin{bmatrix} 1 \\ -\nabla g(v) \end{bmatrix}$$

Exercise 2.7

By 2nd order definition of convexity

$g''(v) \geq 0$ g is convex at v

$g''(v) \leq 0$ g is concave at v

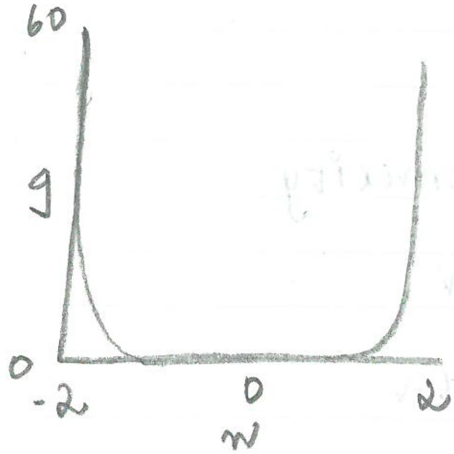


$g(w) = w^2$

$g''(w) = 2$

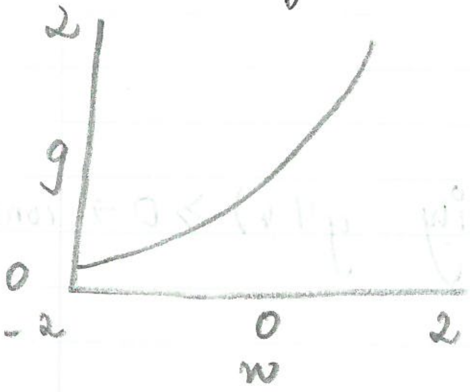
By condition of 2nd order convexity $g''(v) \geq 0 \rightarrow$ convex

g is convex



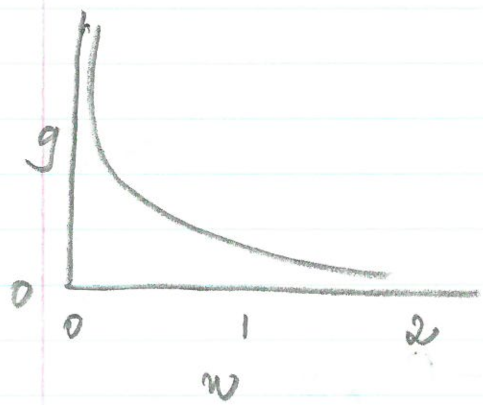
$$g(w) = e^{w^2}$$
$$g'(w) = 2w e^{w^2}$$
$$g''(w) = 2\left[e^{w^2} + w\, e^{w^2} 2w \right]$$
$$g''(w) = 2 e^{w^2} + 4 w^2 e^{w^2}$$ → function is position of any value of w

Thus g is convex.



$$g(w) = \log(1+e^w)$$
$$g'(w) = \frac{1}{1+e^w} e^w$$
$$g''(w) = \frac{(1+e^w)e^w + e^w \cdot e^w}{(1+e^w)^2} = \frac{e^w + 2 e^{2w}}{(1+e^w)^2}$$
$$= e^w (2 + e^w)$$
$$= \frac{e^w(1+2e^w)}{(1+e^w)^2}$$

$g(w)$ is convex for any value of w



$$g(w) = -\log(w)$$

$$g'(w) = \frac{-1}{w}$$

$$g''(w) = \frac{1}{w^2}$$

$\therefore$ $g(w)$ is not always non negative
g is non-convex

Excersie 2.8

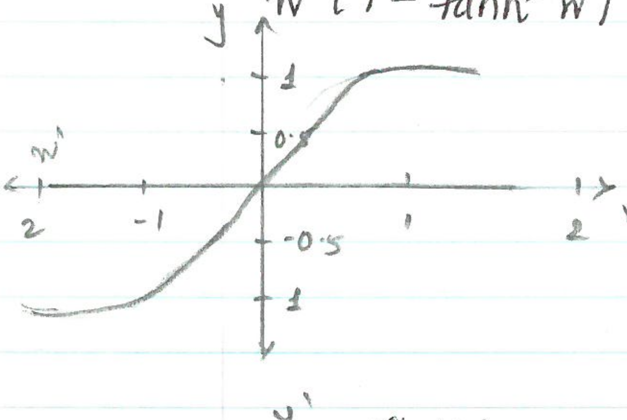
a. $g(w) = w \tanh(w)$

$$\nabla g(v) = 0$$

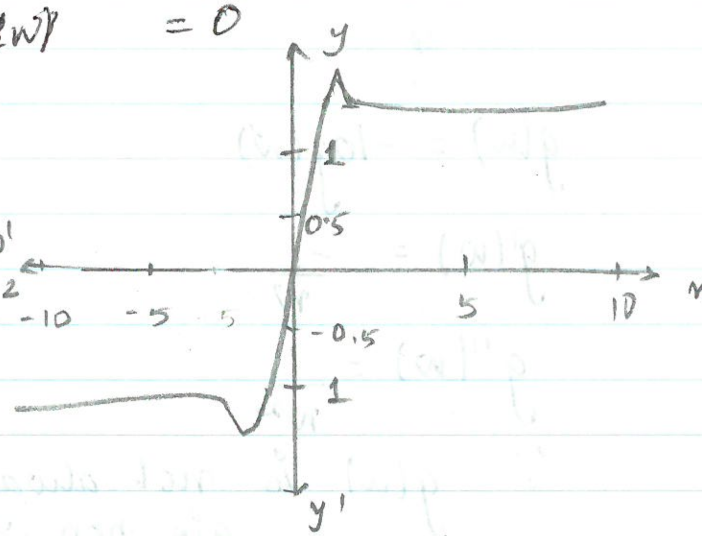
$$g'(w) = 0$$

$$\frac{\partial g(w)}{\partial g w} = 0$$

$$w(1 - \tanh^2 w) + \tanh(w) = 0$$



W 2 to -2



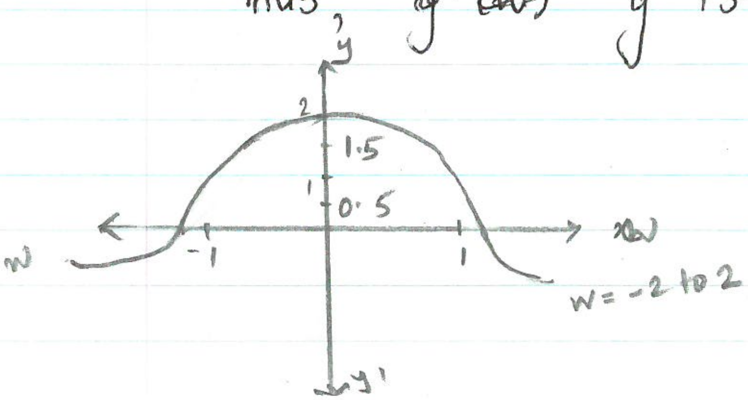
$$\cancel{g''(w)} = \cancel{0}$$

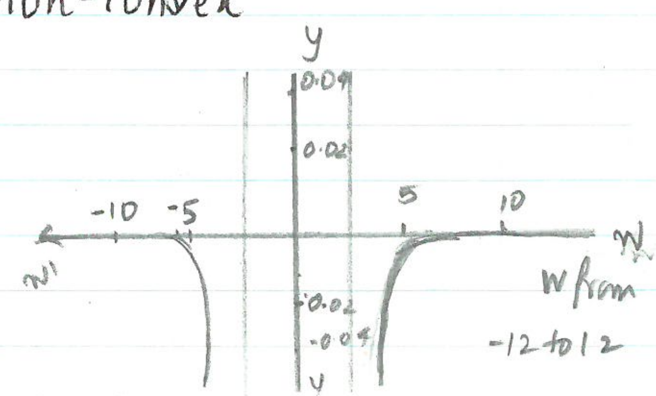
$$g''(w) = (1 - \tanh^2 w) + w - 2 \tanh w (1 - \tan^2 h w) + (1 - \tanh^2 w) = 0$$

$$= 1 - \tanh^2 w - 2w \tanh w (1 - \tanh^2 w) + (1 - \tanh^2 w) = 0$$

Since it is not always non-negative

Thus, ~~$g''(w)$~~ g is non-convex





Excercise 2.17

a. $g(w) = \log(1 + e^{w^Tw})$ $\quad N=2 \quad w = [w_1\ w_2]^T$

$$g'(w) = \frac{e^{w^Tw}}{1+e^{w^Tw}} \times 2w$$

$$g'(w) = 0$$

$w = 0$ at stationary point

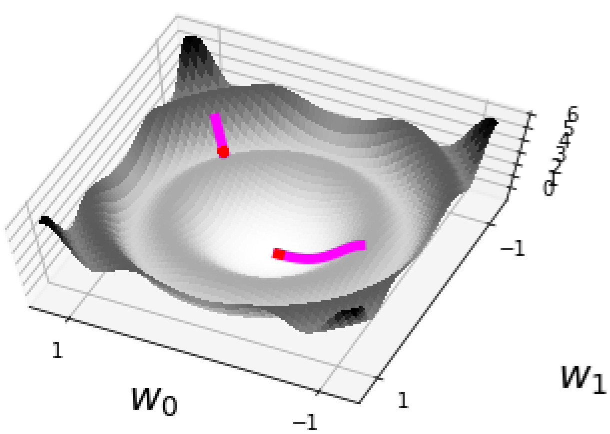
Exercise 2.13

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
from autograd import grad

###### ML Algorithm functions ######
def gradient_descent(w0,alpha):
    w = w0
    g_path = []
    w_path = []
    w_path.append(w)
    g_path.append(-cos(2*pi*dot(w.T,w)) + 2*dot(w.T,w))
    #grad=lambda w,w.T:-cos(2*pi*dot(w.T,w)) + 2*dot(w.T,w)
    # start gradient descent loop
    grad = 1
    iter = 1
    max_its = 50
    while linalg.norm(grad) > 10**(-5) and iter <= max_its:
        # take gradient step
        grad = 4 * w* pi*sin(2*pi *dot(w.T,w)) + 2*w # added this line
        w = w - alpha*grad

        # update path containers
        w_path.append(w)
        g_path.append(-cos(2*pi*dot(w.T,w)) + 2*dot(w.T,w))
        iter+= 1
    g_path = asarray(g_path)
    g_path.shape = (iter,1)
    w_path = asarray(w_path)
    w_path.shape = (iter,2)

    # show final average gradient norm for sanity check
    s = dot(grad.T,grad)/2
    s = 'The final average norm of the gradient = ' + str(float(s))
    print(s)


    # # for use in testing if algorithm minimizing/converging properly
    # plot(asarray(obj_path))
    # show()

    return (w_path,g_path)

###### plotting functions #######
```

```
def make_function():
    global fig,ax1

    # prepare the function for plotting
    r = linspace(-1.15,1.15,300)
    s,t = meshgrid(r,r)
    s = reshape(s,(size(s),1))
    t = reshape(t,(size(t),1))
    h = concatenate((s,t),1)
    h = dot(h*h,ones((2,1)))
    b = -cos(2*pi*h) + 2*h
    s = reshape(s,(int(sqrt(size(s))),int(sqrt(size(s)))))
    t = reshape(t,(int(sqrt(size(t))),int(sqrt(size(t)))))
    b = reshape(b,(int(sqrt(size(b))),int(sqrt(size(b)))))

    # plot the function
    fig = plt.figure(facecolor = 'white')
    ax1 = fig.add_subplot(111, projection='3d')
    ax1.plot_surface(s,t,b,cmap = 'Greys',antialiased=False) # optinal surface-smoothing args rstride=1,
cstride=1,linewidth=0
    ax1.azim = 115
    ax1.elev = 70

    # pretty the figure up
    ax1.xaxis.set_rotate_label(False)
    ax1.yaxis.set_rotate_label(False)
    ax1.zaxis.set_rotate_label(False)
    ax1.get_xaxis().set_ticks([-1,1])
    ax1.get_yaxis().set_ticks([-1,1])
    ax1.set_xlabel('$w_0$   ',fontsize=20,rotation = 0,linespacing = 10)
    ax1.set_ylabel('$w_1$',fontsize=20,rotation = 0,labelpad = 50)
    ax1.set_zlabel('   $g(\mathbf{w})$',fontsize=20,rotation = 0,labelpad = 20)

def plot_steps(w_path,g_path):
    # colors for points
    ax1.plot(w_path[:,0],w_path[:,1],g_path[:,0],color = [1,0,1],linewidth = 5)   # add a little to output path
so its visible on top of the surface plot
    ax1.plot(w_path[-8:-1,0],w_path[-8:-1,1],g_path[-8:-1,0],color = [1,0,0],linewidth = 5)   # add a little to
output path so its visible on top of the surface plot


def main():
    make_function()                           # plot objective function
```

```
    # plot first run on surface
    alpha = 10**-2
    w0 = array([-0.7,0])
    w0.shape = (2,1)
    w_path,g_path = gradient_descent(w0,alpha)    # perform gradient descent
    plot_steps(w_path,g_path)

    # plot second run on surface
    w0 = array([0.8,-0.8])
    w0.shape = (2,1)
    w_path,g_path = gradient_descent(w0,alpha)    # perform gradient descent
    plot_steps(w_path,g_path)
    show()
main()
```

```
The final average norm of the gradient = 0.011667426577188758
The final average norm of the gradient = 4.415498975795742e-11
```

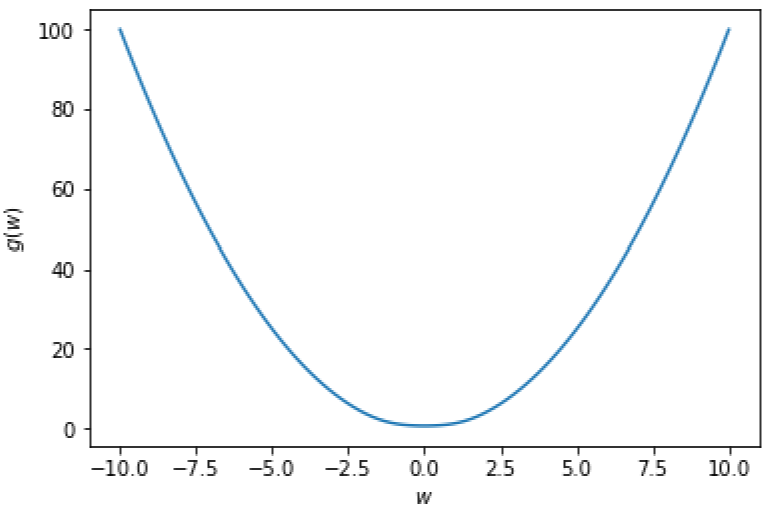


Exercise 2.17 b.

Plot g(w) = log(1+e^w^T.w)

```
import numpy as np
import matplotlib.pyplot as plt
import math as math
def f(t):
    return math.log1p(math.exp(np.dot(t,np.transpose(t))))

t = np.linspace(-10, 10, 100)
```

```
y = np.zeros(len(t))
for i in range(len(t)):
   y[i] = f(t[i])

plt.plot(t,y)
plt.xlabel('$w$')
plt.ylabel('$g(w)$')
plt.show()
```

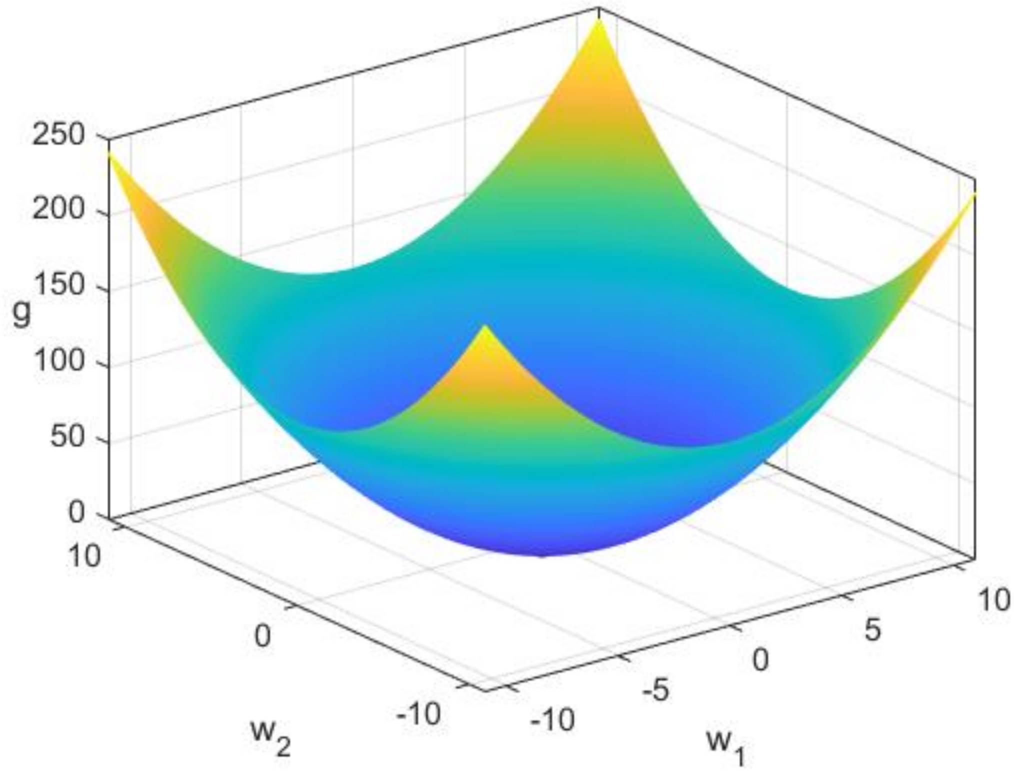


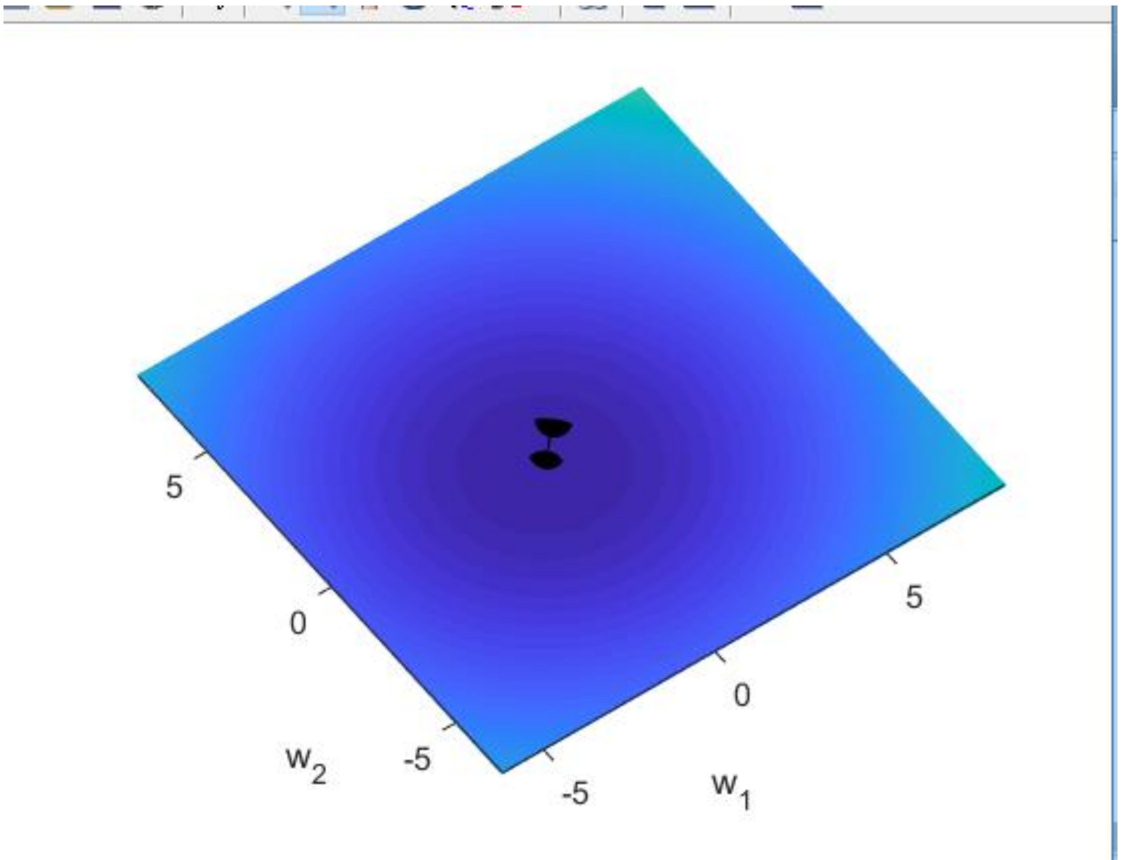
```python
"""
Created on Mon Apr  9 18:13:17 2018

@author: dhana
"""



import numpy as np
import matplotlib.pyplot as plt
import math as math


def f(t):
    return math.log1p(math.exp(np.dot(t,np.transpose(t))))

t = np.linspace(-10, 10, 100)
y = np.zeros(len(t))
for i in range(len(t)):
    y[i] = f(t[i])

plt.plot(t,y)
plt.xlabel('$w$')
plt.ylabel('$g(w)$')
plt.show()

```

Exercise 2.17 c

```matlab
w= [1,1]'
range = 1.1;
x = [-range*10:0.01:range*10];
y = [-range*10:0.01:range*10];
 z= zeros(length(x),length(y));
for i=1: length(x)
     for j=1: length(y)
         z(i,j)= log(1+exp(x(i)^2+y(j)^2));
     end
end
mesh(x,y,z)

 box on
 xlabel('w_1','Fontsize',18,'FontName','cmmi9')
 ylabel('w_2','Fontsize',18,'FontName','cmmi9')
 zlabel('g','Fontsize',18,'FontName','cmmi9')
```

```
set(get(gca,'ZLabel'),'Rotation',0)
set(gca,'FontSize',12);
set(gcf,'color','w');
grad_stop = 10^-3;
max_its = 10;
iter = 1;
grad_eval = 1;
in = [w];
out = [b];
 while iter <= max_its
       % take gradient step

    grad_eval= (2*exp(w'*w)*w)/(exp(w' * w) + 1)

    z11 = ((2*exp(w'*w)/(exp(w'*w) + 1))-
(((4*exp(2*(w'*w))*w(1)^2/(exp(w'*w) +
1)^2))))+((((4*exp((w'*w))*w(1)^2/(exp(w'*w) + 1)))));
    z21 = -(((4*exp(2*(w'*w))*w(1)*w(2)/(exp(w'*w) +
1)^2)))+((((4*exp((w'*w))*w(1)*w(2)/(exp(w'*w) + 1)))));
    z12 = z21;
    z22 = ((2*exp(w'*w)/(exp(w'*w) + 1))-
(((4*exp(2*(w'*w))*w(2)^2/(exp(w'*w) +
1)^2))))+((((4*exp((w'*w))*w(2)^2/(exp(w'*w) + 1)))));
    z = [z11 z12 ; z21 z22];
    u= pinv(z);
    w1(iter)= w(1,:);
    w2(iter)= w(2,:);
    gp(iter) = log (1+exp(w'* w));
    grad1(iter)= grad_eval(1,:);
    grad2(iter)=grad_eval(2,:);
    w = w - u * grad_eval;

       % update containers
       in = [in w];
       out = [out b];

       % update stopers
       iter = iter + 1;
 end
   hold on
   plot3(w1,w2,gp,'.-k','MarkerSize',50)
```

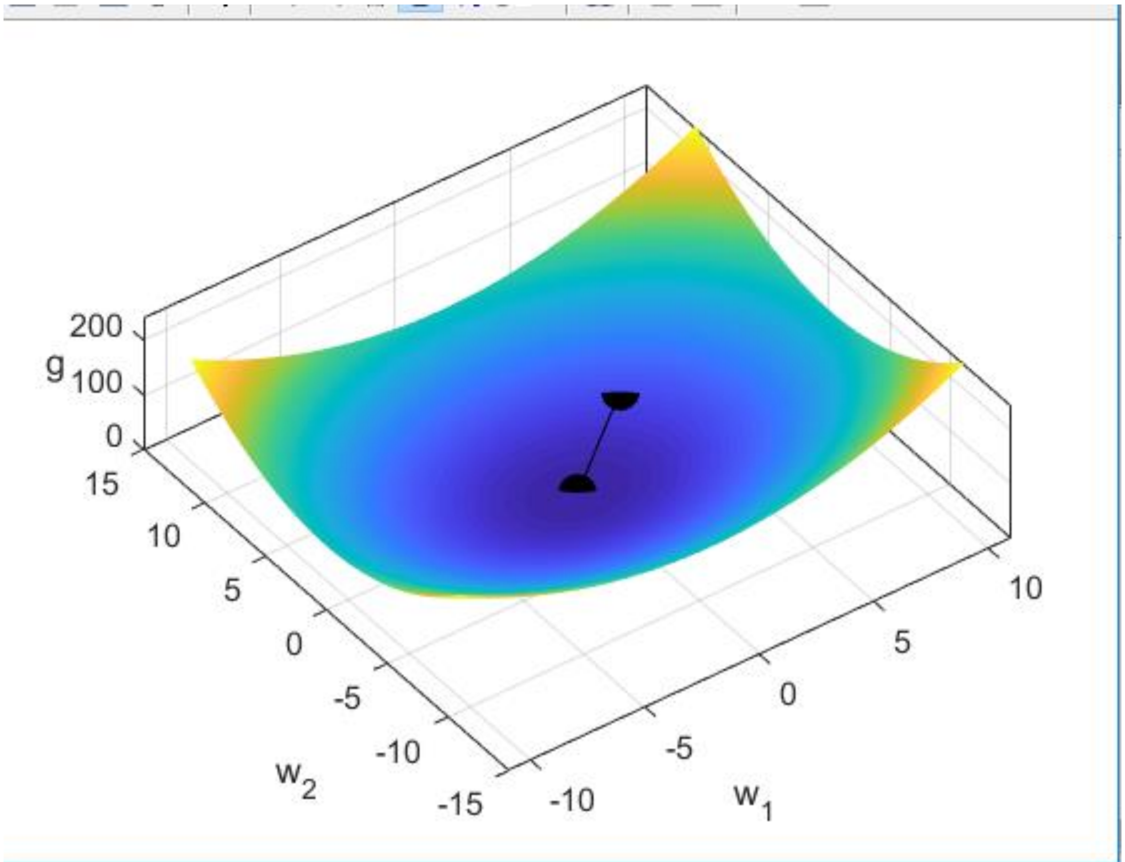

250
200
150
100
50
0
g
10
0
-10
w2
-10
-5
0
5
10
w1



Exercise 2.17 d

```
w= 4*[1,1]'
range = 1.1;
x = [-range*10:0.01:range*10];
y = [-range*10:0.01:range*10];
 z= zeros(length(x),length(y));
for i=1: length(x)
     for j=1: length(y)
         z(i,j)= log(1+exp(x(i)^2+y(j)^2));
     end
end
mesh(x,y,z)

 box on
 xlabel('w_1','Fontsize',18,'FontName','cmmi9')
 ylabel('w_2','Fontsize',18,'FontName','cmmi9')
 zlabel('g','Fontsize',18,'FontName','cmmi9')
 set(get(gca,'ZLabel'),'Rotation',0)
 set(gca,'FontSize',12);
 set(gcf,'color','w');
 grad_stop = 10^-3;
 max_its = 10;
 iter = 1;
 grad_eval = 1;
 in = [w];
 out = [b];
```

```
  while iter <= max_its
        % take gradient step

     grad_eval= (2*exp(w'*w)*w)/(exp(w' * w) + 1)

     z11 = ((2*exp(w'*w)/(exp(w'*w) + 1))-
(((4*exp(2*(w'*w))*w(1)^2/(exp(w'*w) +
1)^2))))+((((4*exp((w'*w))*w(1)^2/(exp(w'*w) + 1)))));
     z21 = -(((4*exp(2*(w'*w))*w(1)*w(2)/(exp(w'*w) +
1)^2)))+((((4*exp((w'*w))*w(1)*w(2)/(exp(w'*w) + 1)))));
     z12 = z21;
     z22 = ((2*exp(w'*w)/(exp(w'*w) + 1))-
(((4*exp(2*(w'*w))*w(2)^2/(exp(w'*w) +
1)^2))))+((((4*exp((w'*w))*w(2)^2/(exp(w'*w) + 1)))));
     z = [z11 z12 ; z21 z22];
     u= pinv(z);
     w1(iter)= w(1,:);
     w2(iter)= w(2,:);
     gp(iter) = log (1+exp(w'* w));
     grad1(iter)= grad_eval(1,:);
     grad2(iter)=grad_eval(2,:);
     w = w - u * grad_eval;

        % update containers
        in = [in w];
        out = [out b];

        % update stopers
        iter = iter + 1;
  end
    hold on
    plot3(w1,w2,gp,'.-k','MarkerSize',50)
```



Exercise 2.17

d. Reason behind why minimum of second order taylor series approximation of g(w) centered at w=4*[1,1] gives minimum of g(w) is because log(1+e^t) approximately equal to t i.e. t here is w^T.w . Also, the second order Taylor series approximation is more closely resembles the underlying function around w, given that second derivative contains curvature information i.e the quadratic function itself.