starter

March 28, 2023

[ ]:

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

[ ]:

```
## DO NOT Change
def graddescent(X,y,tau,w_init,it):
    """
    compute 10 iterations of gradient descent starting at w1
    w_{k+1}= w_k - tau*X'*(X*w_k - y)
    """
    W = np.zeros((w_init.shape[0],it))
    W[:,[0]] = w_init
    for k in range(it-1):
        W[:,[k+1]] = W[:,[k]] - tau * X.T @ (X @ W[:,[k]] - y)
    return W
```

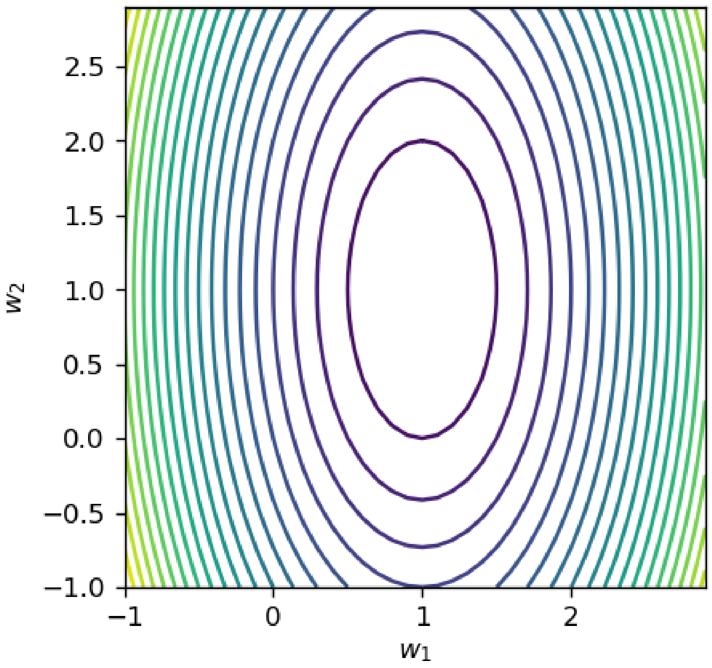
1 Question 1b)

[ ]:

```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [0.5], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w2)):
    for j in range(len(w1)):
        w = np.array([ [w1[j]], [w2[i]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

```
### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square')
```

```
[ ]: (-1.0, 2.8999999999999999, -1.0, 2.8999999999999999)
```

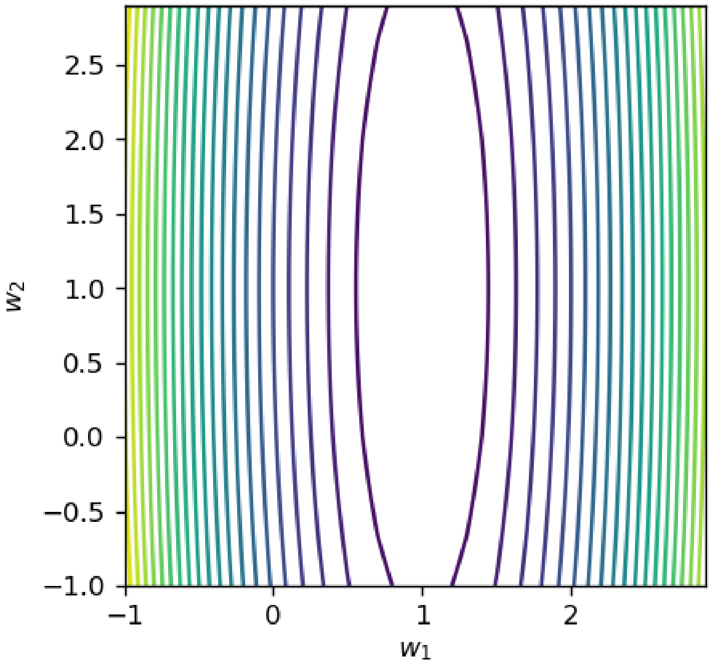


2 Question 1c)

```
[ ]: U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
     S = np.array([[1, 0], [0, 1/5]])
     Sinv = np.linalg.inv(S)
     V = np.eye(2)
     X = U @ S @ V.T
     y = np.array([[1], [1/5], [1], [0]])

     ### Find Least Squares Solution
     w_ls = V @ Sinv @ U.T @ y
     c = y.T @ y - y.T @ X @ w_ls
```

```
### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w2)):
    for j in range(len(w1)):
        w = np.array([ [w1[j]], [w2[i]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c

### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square')
```

```
[ ]: (-1.0, 2.8999999999999999, -1.0, 2.8999999999999999)
```



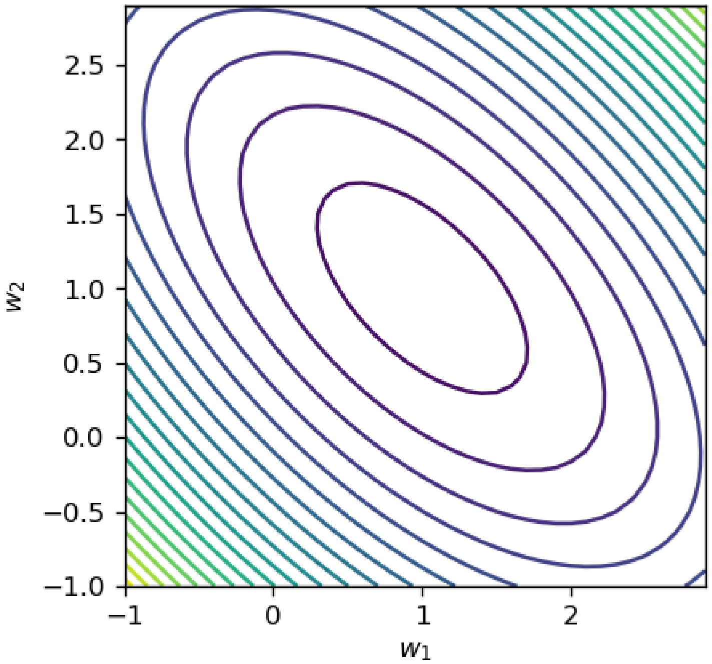
3 Question 1d)

[ ]:

```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w2)):
    for j in range(len(w1)):
        w = np.array([ [w1[j]], [w2[i]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c

### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square')
```

[ ]:

```
(-1.0, 2.8999999999999999, -1.0, 2.8999999999999999)
```



4 Question 2b)

[ ]:

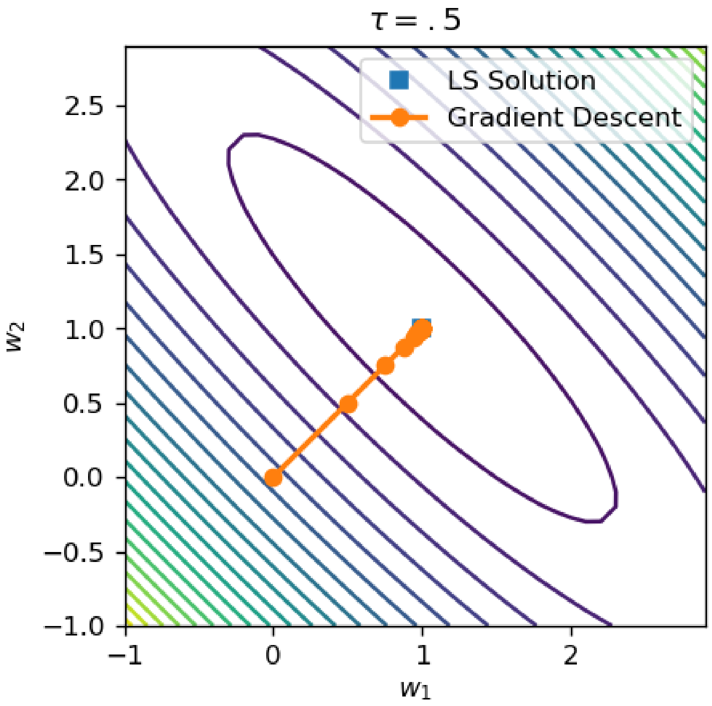
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w1)):
    for j in range(len(w2)):
        w = np.array([ [w1[i]], [w2[j]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

[ ]:

```
w_init = np.array([[0], [0]])  # complete this line with a 2x1 numpy array for␣
 ↪the values specified in the activity
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it)


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$')
plt.axis('square')
```

[ ]:

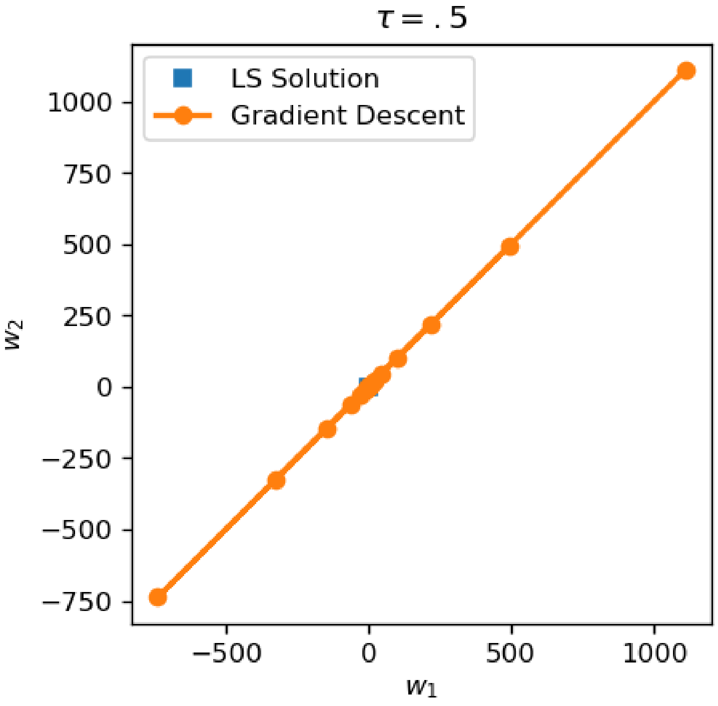
```
(-1.0, 2.8999999999999999, -1.0, 2.8999999999999999)
```



5 Question 2c)

```
[ ]: w_init = np.array([[1.5], [-.5]])  # complete this line with a 2x1 numpy array␣
      ↪for the values specified in the activity
     it = 20
     tau = 2.5
     W = graddescent(X,y,tau,w_init,it)


     ### Create plot
     plt.figure(num=None, figsize=(4, 4), dpi=120)
     plt.contour(w1,w2,fw,20)
     plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
     plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
     plt.legend()
     plt.xlim([-1,3])
     plt.xlabel('$w_1$')
     plt.ylim([-1,3])
     plt.ylabel('$w_2$')
     plt.title(r'$\tau = .5$')
     plt.axis('square')
```

```
[ ]: (-830.3141824977304,
      1201.7871525657204,
      -830.3141825420869,
      1201.7871525213639)
```



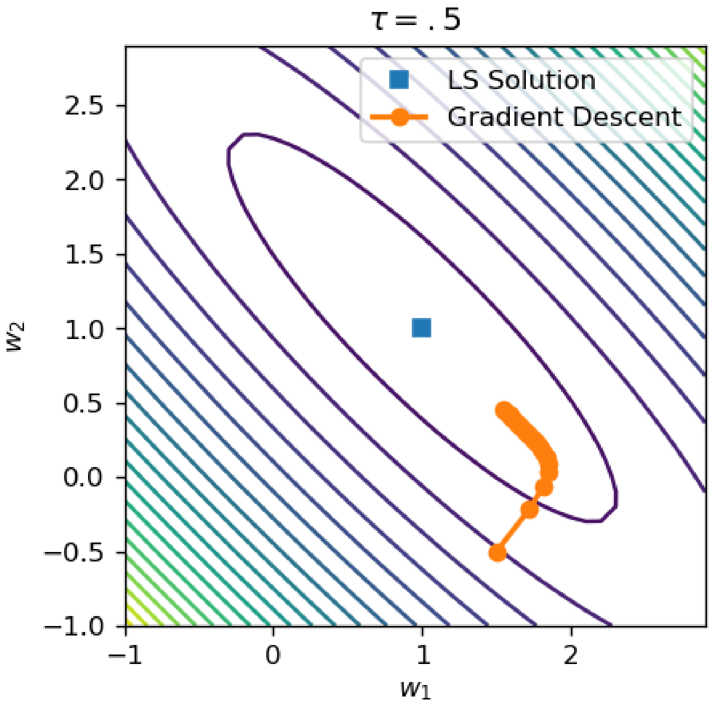
6 Question 2d)

[ ]:

```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 1/4]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w1)):
    for j in range(len(w2)):
        w = np.array([ [w1[i]], [w2[j]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
w_init = np.array([[1.5], [-.5]])  # complete this line with a 2x1 numpy array␣
 ↪for the values specified in the activity
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it)


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$')
plt.axis('square')
```

[ ]:

```
(-1.0, 2.8999999999999999, -1.0, 2.8999999999999999)
```


$\tau = .5$
LS Solution
Gradient Descent
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
w_2
−1
0
1
2
w_1

[ ]: