a). $A^TAA^T + \lambda A^T$

$A^T(AA^T+\lambda I) = (A^TA+\lambda I)A^T$

if $\lambda > 0$ then they are invertible

$A^T(AA^T+\lambda I)^{-1} = (A^TA+\lambda I)^{-1}A^T$

$A^T = A^T$

CS/ECE/ME532 Activity 17

Estimated Time: 25 min for P1, 15 min for P2, 25 min for P4

1. **Alternative regularization formulas.** This problem is about two alternative ways of solving the L_2-regularized least squares problem.

   a) Prove that for any $\lambda > 0$, the following matrix identity holds:

   $$(\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^T = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T + \lambda\boldsymbol{I})^{-1}$$

   Hint: Start by considering the expression $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{A}^T + \lambda\boldsymbol{A}^T$ and factor it in two different ways (from the right or from the left).

   b) The identity proved in part **a)** shows that there are actually two equivalent formulas for the solution to the L_2-regularized least squares problem. Suppose $\boldsymbol{A} \in \mathbb{R}^{8000\times100}$ and $\boldsymbol{y} \in \mathbb{R}^{8000}$, and use this identity to find $\boldsymbol{w}$ that minimizes $||\boldsymbol{A}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_2^2$ in two different ways. Which formula will compute more rapidly? Why? *Note:* The number of operations required for matrix inversion is proportional to the cube of the matrix dimension. we want a smaller matrix to inverse $\rightarrow A^TA = 100\times100$, $AA^T = 8000\times8000 \rightarrow$ use $(A^TA+\lambda I)^{-1}A^T$

   c) A breast cancer gene database has approximately 8000 genes from 100 subjects. The label y_i is the disease state of the ith subject (+1 if no cancer, -1 if breast cancer). Suppose we build a linear classifier that combines the 8000 genes, say $\boldsymbol{g}_i, i = 1, 2, \ldots, 100$ to predict whether a subject has cancer $\hat{y}_i = \text{sign}\{\boldsymbol{g}_i^T\boldsymbol{w}\}$. Note that here $\boldsymbol{g}_i$ and $\boldsymbol{w}$ are 8000-by-1 vectors.

      i. Write down the least squares problem for finding classifier weights $\boldsymbol{w}$ given 100 labels. Does this problem have a unique solution? $\min_w ||Aw-d||_2^2 \rightarrow w = (A^TA)^{-1}A^Td$

      ii. Write down a Tikhonov(ridge)-regression problem for finding the classifier weights given 100 labels. Does this problem have a unique solution? Which form of the identity in part **a)** leads to the most computationally efficient solution for the classifier weights? $\min_w ||Aw-d||_2^2 + \lambda||w||_2^2$

      $w = (A^TA+\lambda I)^{-1}A^Td$

      rank(A) < dim(w) so no unique solution

2. The key idea behind proximal gradient descent is to reformulate the general regularized least-squares problem into a set of simpler scalar optimization problems. Consider the regularized least-squares problem

   $$\min_{\boldsymbol{w}} ||\boldsymbol{z} - \boldsymbol{w}||_2^2 + \lambda r(\boldsymbol{w})$$

   An upper bound and completing the square was used to simplify the generalized least-squares problem into this form. Let the i^{th} elements of $\boldsymbol{z}$ and $\boldsymbol{w}$ be z_i and w_i, respectively.

$\min_w \|z-w\|_2^2 + \lambda\|w\|_2^2 \longrightarrow \min_w \sum_i (z_i-w_i)^2 + \lambda \sum_i w_i^2$

a) Assume $r(\boldsymbol{w}) = ||\boldsymbol{w}||_2^2$. Write the regularized least-squares problem as a series of separable problems involving only w_i and z_i.

b) Assume $r(\boldsymbol{w}) = ||\boldsymbol{w}||_1$. Write the regularized least-squares problem as a series of separable problems involving only w_i and z_i.

$\min_w \|z-w\|_2^2 + \lambda\|w\|_1 \longrightarrow \min_w \sum_i (z_i-w_i)^2 + \lambda \sum_i |w_i|$

3. A script is available to compute a specified number of iterations of the proximal gradient descent algorithm for solving a Tikhonov-regularized least squares problem

$$\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 + \lambda||\boldsymbol{w}||_2^2$$

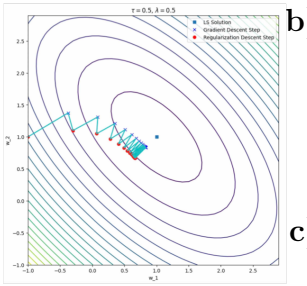
The provided script will get you started displaying the path taken by the weights in the proximal gradient descent iteration superimposed on a contour plot of the squared error surface. Assume $\boldsymbol{y} = \begin{bmatrix} \sqrt{2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$, the 4-by-2 $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ has singular value decomposition $\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$, and $\boldsymbol{V} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Complete 20 iterations of gradient descent in each case specified below.

Include the plots you generate below with your submission.

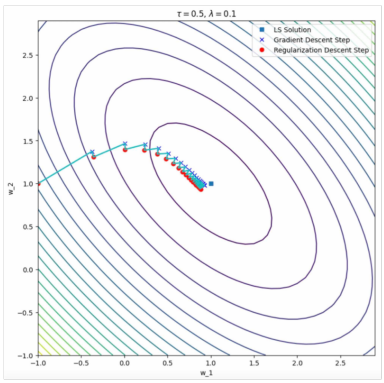
a) What is the maximum value for the step size τ that will guarantee convergence?

τ has to be between 0 and $\frac{1}{\|X\|_{op}^2} = \frac{1}{1^2} = 1$

b) Start proximal gradient descent from the point $\boldsymbol{w} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ using a step size of $\tau = 0.5$ and tuning parameter $\lambda = 0.5$. How do you explain the trajectory the weights take toward the optimum, e.g., why is it shaped this way? What direction does each iteration move in the regularization step? gradient descent tries to go in the direction of maximum descent, but regularization tries to get $\|w\|_2^2$ closer to (0,0) to minimize it. So GD travels to the center of the ellipse which is not (0,0), so regularization drags it back.



c) Repeat the previous case with $\lambda = 0.1$ What happens? How does λ affect each iteration and why?



with a smaller λ, the regularization step doesn't have as much power, so the solution ends up much closer to the center of the ellipse.

Activity 17

Setup

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


def prxgraddescent_l2(X,y,tau,lam,w_init,it):

## compute it iterations of L2 proximal gradient descent starting at w1
## w_{k+1}= (w_k - tau*X'*(X*w_k - y)/(1+lam*tau)
## step size tau
    W = np.zeros((w_init.shape[0], it+1))
    Z = np.zeros((w_init.shape[0], it+1))
    W[:,[0]] = w_init
    for k in range(it):
        Z[:,[k+1]] = W[:,[k]] - tau * X.T @ (X @ W[:,[k]] - y);
        W[:,[k+1]] = Z[:,[k+1]]/(1+lam*tau)

    return W,Z


## Proximal gradient descent trajectories
## Least Squares Problem
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]])
y = np.array([[np.sqrt(2)], [0], [1], [0]])

X = U @ S @ V.T

### 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
```

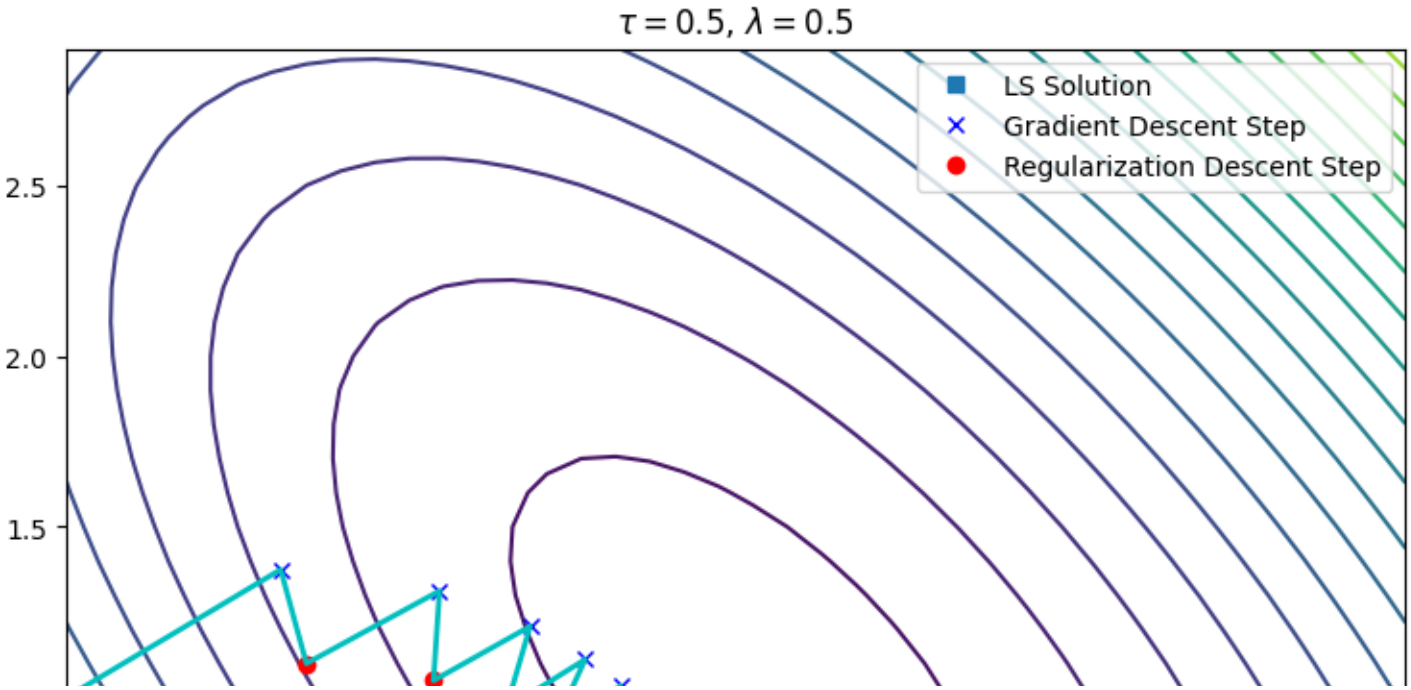
Question 3b)

```
## Find and display weights generated by gradient descent

w_init = np.array([[-1],[1]])
lam = 0.5;
it = 20
tau = 0.5
W,Z = prxgraddescent_l2(X,y,tau,lam,w_init,it)

# Concatenate gradient and regularization steps to display trajectory
G = np.zeros((2,0))
for i in range(it):
    G = np.hstack((G,np.hstack((W[:,[i]],Z[:,[i+1]]))))

plt.figure(figsize=(9,9))
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(Z[0,1::],Z[1,1:],'bx',linewidth=2, label="Gradient Descent Step")
plt.plot(W[0,:],W[1,:],'ro',linewidth=2, label="Regularization Descent Step")
plt.plot(G[0,:],G[1,:],'-c',linewidth=2)
plt.legend()
plt.xlabel('w_1')
plt.ylabel('w_2')
plt.title('$\\tau = $'+str(tau)+', $\lambda = $'+str(lam));
```



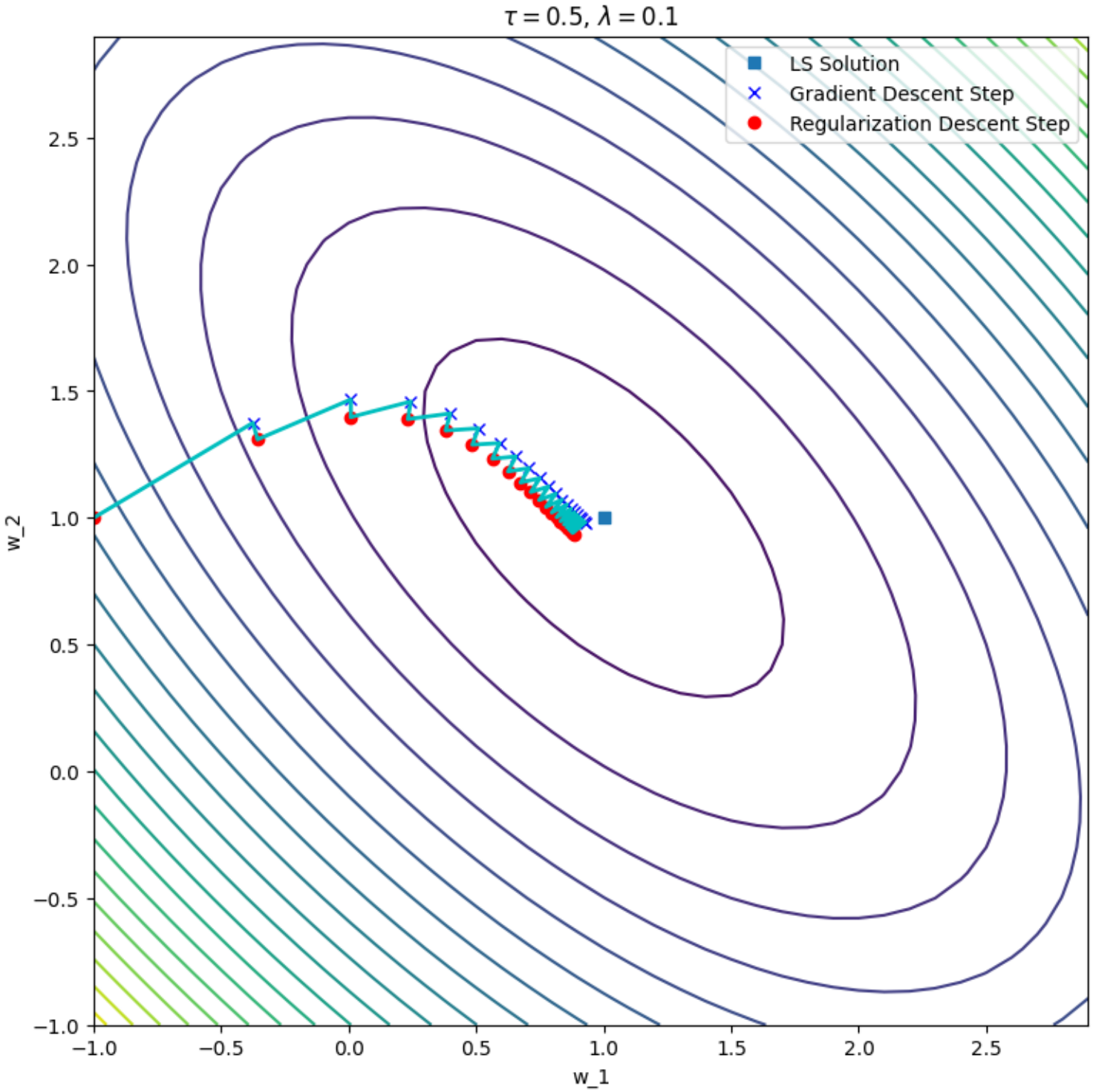
Question 3c)

```
## Find and display weights generated by gradient descent

w_init = np.array([[-1],[1]])
lam = 0.1;
it = 20
tau = 0.5
W,Z = prxgraddescent_l2(X,y,tau,lam,w_init,it)

# Concatenate gradient and regularization steps to display trajectory
G = np.zeros((2,0))
for i in range(it):
    G = np.hstack((G,np.hstack((W[:,[i]],Z[:,[i+1]]))))

plt.figure(figsize=(9,9))
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(Z[0,1::],Z[1,1:],'bx',linewidth=2, label="Gradient Descent Step")
plt.plot(W[0,:],W[1,:],'ro',linewidth=2, label="Regularization Descent Step")
plt.plot(G[0,:],G[1,:],'-c',linewidth=2)
plt.legend()
plt.xlabel('w_1')
plt.ylabel('w_2')
plt.title('$\\tau = $'+str(tau)+', $\lambda = $'+str(lam));
```


$\tau = 0.5, \lambda = 0.1$
LS Solution
Gradient Descent Step
Regularization Descent Step
w_1
w_2
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5