Asmt 6: Regression

Turn in through Canvas by 2:45pm:
Wednesday, March 25
100 points

1 Linear Regression & Cross-Validation (100 points)

We will find coefficients `alpha` to estimate `X*alpha` $\approx$ `y`, using the provided datasets `X` and `y`. We will compare two approaches *least squares* and *ridge regression*. (e.g., in python as)

Least Squares: Set `alpha = LA.inv(X.T @ X) @ X.T @ y.T`

Ridge Regression: Set `alphas = LA.inv(X.T @ X + s*np.identity(50)) @ X.T @ y.T`

A (30 points): Solve for the coefficients `alpha` (or `alphas`) using Least Squares and Ridge Regression with $s \in \{0.2, 0.4, 0.8, 1.0, 1.2, 1.4, 1.6\}$ (i.e. s will take on one of those 7 values each time you try, say obtaining `alpha04` for $s = 0.4$). For each set of coefficients, report the error in the estimate $\hat{y}$ of y as `norm(y - X*alpha,2)`.

Least Squares error: 3.47
Ridge Regression errors:
s = 0.2 : 3.87
s = 0.4 : 4.03
s = 0.8 : 4.30
s = 1.0 : 4.42
s = 1.2 : 4.54
s = 1.4 : 4.64
s = 1.6 : 4.74

```
X = np.loadtxt ('X.csv', delimiter=",")
y = np.loadtxt ('y.csv', delimiter=",")
svalues = [0.2,0.4,0.8,1.0,1.2,1.4,1.6]
alpha = LA.inv(X.T @ X) @ X.T @ y.T
print ('Least Squares error %.2f'% LA.norm(y -X @ alpha,2))

print ('Ridge Regression errors:')
svalues = [0.2,0.4,0.8,1.0,1.2,1.4,1.6]
for s in svalues:
  alphas = LA.inv(X.T @ X + s*np.identity(50)) @ X.T @ y.T
  print('s = ', s, ': %.2f'% LA.norm(y -X @ alphas,2))
```

B (30 points): Create three row-subsets of `X` and `Y`

- `X1 = X[:66,:]` and `Y1 = Y[:66]`
- `X2 = X[33:,:]` and `Y2 = Y[33:]`
- `X3 = np.vstack((X[:33,:], X[66:,:]))` and `Y3 = np.vstack((Y[:33], Y[66:]))]`

Repeat the above procedure on these subsets and *cross-validate* the solution on the remainder of `X` and `Y`. Specifically, learn the coefficients `alpha` using, say, `X1 and Y1` and then measure `np.norm(Y[66:] - X[66:,:] @ alpha,2)`.

Least Square				
	Subset 1	Subset 2	Subset 3	Average Error
	6.35	4.85	4.48	5.23
Ridge Regression				
s/Subset	Subset 1	Subset 2	Subset 3	Average Error
0.20	3.54	3.04	2.69	3.09
0.40	3.42	3.13	2.52	3.03
0.80	3.41	3.34	2.48	3.07
1.00	3.43	3.43	2.50	3.12
1.20	3.46	3.51	2.53	3.16
1.40	3.48	3.58	2.57	3.21
1.60	3.51	3.64	2.62	3.26

```
print('Least Squares error:')
alpha_1 = LA.inv(X1.T @ X1) @ X1.T @ y1.T
resid_1 = LA.norm(Y[66:] - X[66:,:] @ alpha_1,2)
print ('Subset 1 %.2f'% resid_1)

alpha_2 = LA.inv(X2.T @ X2) @ X2.T @ y2.T
resid_2 = LA.norm(Y[:33] - X[:33,:] @ alpha_2,2)
print ('Subset 2 %.2f'% resid_2)

alpha_3 = LA.inv(X3.T @ X3) @ X3.T @ y3.T
resid_3 = LA.norm(Y[33:66] - X[33:66,:] @ alpha_3,2)
print ('Subset 3 %.2f'% resid_3)

print('average LS error %.2f' %  ((resid_1+resid_2+resid_3)/3))
```

```
print("Subset 1")
err_1 = []
for s in svalues:
  alphas = LA.inv(X1.T @ X1 + s*np.identity(50)) @ X1.T @ y1.T
  resid = LA.norm(Y[66:] - X[66:,:] @ alphas,2)
  err_1.append(resid)
  print('s = ', s, ':%.2f' % resid)

print("Subset 2")
err_2 = []
for s in svalues:
  alphas = LA.inv(X2.T @ X2 + s*np.identity(50)) @ X2.T @ y2.T
  resid = LA.norm(Y[:33] - X[:33,:] @ alphas,2)
  err_2.append(resid)
  print('s = ', s, ':%.2f' % resid)

print("Subset 3")
err_3 = []
for s in svalues:
  alphas = LA.inv(X3.T @ X3 + s*np.identity(50)) @ X3.T @ y3.T
  resid = LA.norm(Y[33:66] - X[33:66,:] @ alphas,2)
  err_3.append(resid)
  print('s = ', s, ':%.2f' %resid)

print('Average error')
print(np.average(np.stack((err_1, err_2, err_3)), axis = 0))
```

C (15 points): Which approach works best (averaging the results from the three subsets): Least Squares, or for which value of s using Ridge Regression?

average Least Squares error 5.23
Average Ridge Regression errors for $s = [0.2, 0.4, 0.8, 1.0, 1.2, 1.4, 1.6]$:
[3.09, 3.03, 3.07, 3.12, 3.16, 3.21, 3.26]

Ridge Ression has lower error on new data (test set). $s = 0.4$ works best

D (15 points): Use the same 3 test / train splits, taking their average errors, to estimate the average squared error on each predicted data point. What is problematic about the above estimate, especially for the best performing parameter value s?

We use the average error of the 3 subset, call it e. To get average squared error on each predicted data point, we raise it to power of 2 and divide by the length of the overall dataset, in this case, 100: $\frac{e^2}{100}$

Least Square		
	Average Error (e)	Individual squared error (eˆ2/100)
	5.23	0.27
Ridge Regression		
s	Average Error (e)	Individual squared error (eˆ2/100)
0.20	3.09	0.10
0.40	3.03	0.09
0.80	3.07	0.09
1.00	3.12	0.10
1.20	3.16	0.10
1.40	3.21	0.10
1.60	3.26	0.11

It is problematic if the dataset is not uniform and subset could have different distributions.

E (10 points): Even circumventing the issue raised in part **D**, what *assumptions* about how the data set $(\mathtt{X}, \mathtt{y})$ is generated are needed in an assessment based on cross-validation?
Assumptions:
Training data cannot be different than test data. If training data has different distribution than test data then model will not perform well.
No information leakage meaning that information from validation set cannot leak to training set. If it does then one could learn the test data and produce low error but not necccessarily evaluate how well the model is doing when new data is introduced, this is cheating.

2 Bonus: Matching Pursuit (5 points)

Consider a linear equation `W = M*S` where `M` is a measurement matrix filled with random values $\{-1, 0, +1\}$ (although now that they are there, they are no longer random), and `W` is the output of the sparse signal `S` when measured by `M`.

Use Matching Pursuit (as described in the book as **Algorithm 5.5.1**) to recover the non-zero entries from `S`. Record the order in which you find each entry and the residual vector after each step.