Homework 10

Due Monday, April 11 before 5:00pm

Use [Live Editor > Save > Export to PDF] to prepare your submission for Gradescope.

Cell/Matrix Interactions

Multivariable linear regression can be used to analyze large-scale datasets and to develop models that describe complex biological phenomena. The dataset that you will analyze in this problem is from an experiment designed to study how cancer cells interact with a set of proteins termed extracellular matrix (ECM) proteins.

ECM proteins (collagen is a very common example) provide structure to normal tissues as well as diseased tissues such as tumors. In addition to a structural role, ECM proteins can also provide important signals to cancer cells, which can help the cancer cells survive, proliferate, or migrate to other sites in the body. Although it is clear that ECM proteins are important in cancer, the overall matrix of a tumor consists of a complex combination of many different ECM proteins. What is not currently known is, (i) what are the most important ECM proteins in different types of cancer, and (ii) what happens when cancer cells interact with combinations of ECM proteins at the same time?

For this problem, you will analyze actual data associated with the following experiment.

- A lung cancer cell line was cultured on 55 different combinations of 10 ECM proteins (10 proteins listed in next bullet). These unique combinations included the 10 ECM proteins individually, and the 45 2-factor combinations of the 10 proteins. For example, one of the conditions was collagen 1 (C1) by itself, and then 9 other conditions were C1 paired with each of the other ECM proteins.
- ECM proteins (and abbreviations): C1= collagen 1; C3= collagen 3; C4= collagen 4; G3= galectin-3; G8= galectin-8; OP= osteopontin; TC= tenascin-C; TR= tenascin-R; FN= fibronectin; LN= laminin
- After culture, the number of adherent cancer cells was quantified. If an ECM combination enhanced the number of adherent cells, then this ECM likely promotes cancer cell adhesion, survival, and possibly proliferation.
- For each ECM combination, there were approximately 15-20 replicate measurements of adherent cell number.

From this dataset, the goal is to quantitatively determine how individual ECM proteins, and specific combinations of ECM proteins, control lung cancer cell adhesion (adherent cell number).

Fit a linear model to predict the number of adherent cells for each ECM and their interactions. The data are stored in the variable `ecm`.

Notes (see `help fitlm` for more details):

- This experiment examined 10 distinct ECMs, not 10 variations on one “standard” ECM. As such, there should be no intercept in your model. You can disable the intercept with the ‘intercept’ argument.
- You can easily include interactions for all variables using the ‘interactions’ argument to fitlm.
- You can specify the response variable with the ‘ResponseVar’ argument.

```
load HW10_data.mat
```

```
rng(210); % set the random number generator

% place your code here
proteins = table2array(ecm(:,1:10));
cell_count = table2array(ecm(:,end));
model = fitlm(proteins, cell_count, 'interactions', 'Intercept', false)
```

```
model =
Linear regression model:
    y ~ x1*x2 + x1*x3 + x1*x4 + x1*x5 + x1*x6 + x1*x7 + x1*x8 + x1*x9 + x1*x10 + x2*x3 + x2*x4 + x2*x5 + x2*x6 + x2
```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
x1	14.259	1.7029	8.373	2.2371e-16
x2	13.747	1.7029	8.0725	2.2886e-15
x3	7.4353	1.7029	4.3661	1.4171e-05
x4	10.443	1.7148	6.0899	1.6954e-09
x5	1.3867	1.8129	0.76488	0.44455
x6	2.3882	1.7029	1.4024	0.16115
x7	0.725	1.7554	0.41302	0.67969
x8	1.1404	1.6659	0.68454	0.49382
x9	6.8051	1.6659	4.0849	4.8184e-05
x10	5.9941	1.7029	3.5199	0.00045424
x1:x2	-16.294	2.9496	-5.5242	4.3756e-08
x1:x3	-6.5118	2.9496	-2.2077	0.027527
x1:x4	-11.155	2.9564	-3.773	0.00017224
x1:x5	-4.7808	3.0144	-1.586	0.11311
x1:x6	-1.8412	2.9496	-0.62422	0.53265
x1:x7	-6.2368	2.9802	-2.0928	0.03666
x1:x8	4.7831	2.9284	1.6334	0.10275
x1:x9	-7.2757	2.9284	-2.4846	0.013159
x1:x10	-4.2647	2.9496	-1.4459	0.14858
x2:x3	-6.5882	2.9496	-2.2336	0.025763
x2:x4	-16.237	2.9564	-5.4921	5.2188e-08
x2:x5	-4.1161	3.0144	-1.3655	0.17246
x2:x6	-14.747	2.9496	-4.9997	6.9472e-07
x2:x7	-9.6368	2.9802	-3.2336	0.0012684
x2:x8	-9.0639	2.9284	-3.0952	0.0020298
x2:x9	-8.7639	2.9284	-2.9928	0.002843
x2:x10	-6.0706	2.9496	-2.0581	0.039877
x3:x4	-7.6312	2.9564	-2.5812	0.010009
x3:x5	9.8369	3.0144	3.2633	0.0011444
x3:x6	4.1294	2.9496	1.4	0.16187
x3:x7	8.8691	2.9802	2.9761	0.0030008
x3:x8	5.8537	2.9284	1.999	0.045923
x3:x9	-4.0051	2.9284	-1.3677	0.17176
x3:x10	-2.5647	2.9496	-0.86951	0.38481
x4:x5	-4.4414	3.0211	-1.4701	0.1419
x4:x6	-9.3076	2.9564	-3.1483	0.0016986
x4:x7	-8.2091	2.9869	-2.7483	0.0061142
x4:x8	-10.089	2.8275	-3.5683	0.00037904
x4:x9	-15.197	2.8009	-5.4259	7.4839e-08
x4:x10	-13.924	3.0211	-4.6088	4.6579e-06
x5:x6	5.7663	3.0144	1.9129	0.05609
x5:x7	4.1354	3.0443	1.3584	0.17469
x5:x8	6.1141	2.9937	2.0423	0.041419
x5:x9	0.19646	2.9937	0.065626	0.94769
x5:x10	-3.8573	3.0144	-1.2796	0.20103
x6:x7	-1.2662	2.9802	-0.42487	0.67104
x6:x8	0.22431	2.9284	0.076598	0.93896
x6:x9	-2.2169	2.9284	-0.75703	0.44924

```
x6:x10    -4.4324     2.9802     -1.4873        0.1373
x7:x8     0.31317     3.0624     0.10226       0.91857
x7:x9     -2.7772     2.9592     -0.9385       0.34825
x7:x10    -2.8074     2.9802    -0.94202       0.34645
x8:x9     -1.7153     2.7528    -0.62312       0.53337
x8:x10    -3.2228     2.9284     -1.1005       0.27141
x9:x10    -6.0933     2.9284     -2.0808      0.037745


Number of observations: 923, Error degrees of freedom: 868
Root Mean Squared Error: 7.02
```

Every term in the linear model is nonzero (although only some a statistically significant). It's unlikely that all the combinations have a nonzero effect. The problem is our linear model includes a large number of features (1 intercept + 10 ECM proteins + 45 interactions) but lacks any regularization to enforce sparsity. Let's use the LASSO to fit a more parsimonious model.

Preparing the Data

Matlab's `lasso` function does not use the formula-style input of `fitlm`. We need to construct our own model matrix with for all the proteins and their two-way interactions. The following code builds the model matrix. Stop here and try it for yourself first if you want a challenge!

First, let's pull the response vector out of the dataframe. It's in the column `ecm.cells`.

```
cells = ecm.cells;
ecm.cells = [];
```

The second line deletes the column from the data table. Now let's compute the size of the model matrix.

```
[n,k] = size(ecm);
p = 1 + k + nchoosek(k,2);
```

There are `n` rows (one per observation) and `k` proteins. The number of interactions is the number of ways we can choose two of the `k` proteins. The extra column of ones is for the intercept. Now we can create a model matrix and fill the corresponding columns.

```
X = ones(n,p);
X(:,2:k+1) = table2array(ecm); % Fill the first-order terms

% create a cell array of names for each term in the model
treatment = cell(p,1);
treatment{1} = '(Intercept)';
treatment(2:k+1) = ecm.Properties.VariableNames;

twi = 1+k+1; % current column
for i = 1:k
    for j = i+1:k
        X(:,twi) = X(:,1+i) .* X(:,1+j); % create the interaction
        treatment{twi} = [treatment{1+i} ':' treatment{1+j}]; % add the interaction name
        twi = twi + 1;
    end
end
```

Regularized Linear Regression with the LASSO

Now we're ready to use the `lasso` command to fit a linear model that penalizes nonzero coefficients.

```
[B,fitinfo] = lasso(X,cells,'NumLambda',10,'CV',5);
```

As we said in class, there isn't a good way to find the value of λ, the regularization constant. Instead we ask `lasso` to try a range of 10 different values using the `'NumLambda'` parameter. The `fitinfo` structure provides information about the model fitting.

```
fitinfo
```

```
fitinfo = struct with fields:
         Intercept: [10.0981 8.0219 6.5435 6.1946 5.6582 5.5577 6.0901 6.1985 6.7636 8.3640]
            Lambda: [2.6340e-04 7.3293e-04 0.0020 0.0057 0.0158 0.0439 0.1223 0.3402 0.9466 2.6340]
             Alpha: 1
                DF: [55 54 53 54 52 48 36 25 6 0]
               MSE: [54.3102 54.3094 54.2809 54.2044 54.0292 53.8536 54.2212 55.7172 61.1556 72.5458]
    PredictorNames: {}
     UseCovariance: 1
                SE: [3.1073 3.1029 3.0949 3.0722 3.0304 2.8916 2.7921 3.0205 4.0866 5.5066]
      LambdaMinMSE: 0.0439
         Lambda1SE: 0.3402
       IndexMinMSE: 6
          Index1SE: 8
```

The `Lambda` field gives the 10 regularization constants tried by `lasso`. The `DF` field shows the number of degrees of freedom in the regularized models--this is the number of nonzero coefficients. Recall that our linear model used all 55 coefficients; the same is true for the model with the smallest regularization constant. As the regularization constant increases, the number of nonzero coefficients decreases.

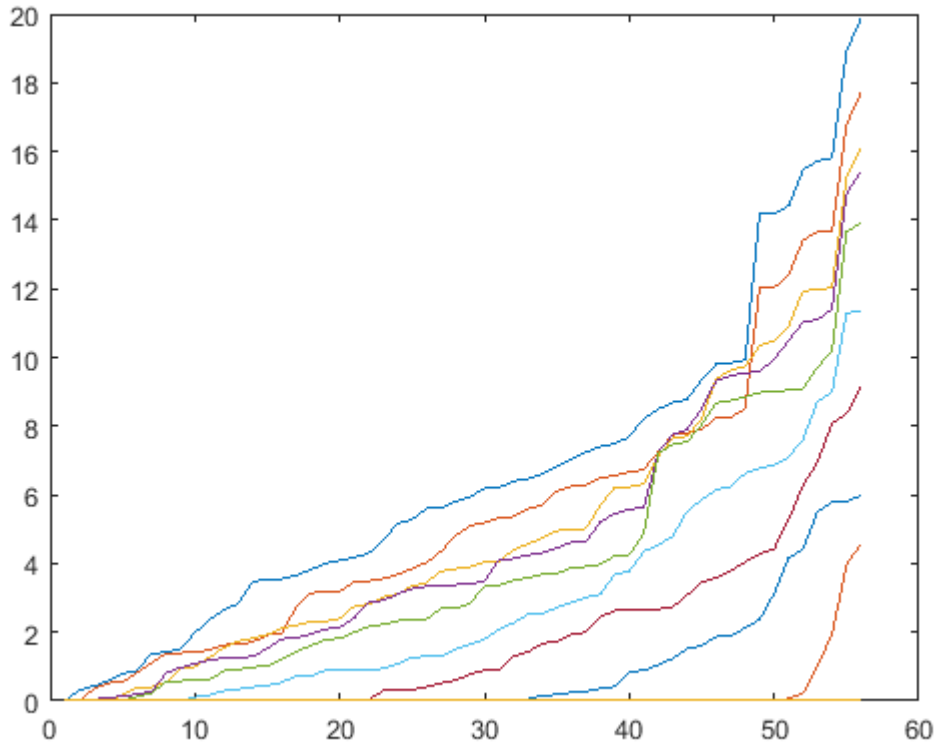
Which regularization constant is best? We can turn to the `SE` field, which gives the standard error of the cross validation (5-fold cross validation, as we specified in our call to `lasso`). The smallest standard error occured with the 6th value in Lambda (given by `IndexMinMSE`), which is 0.0439 (`LambdaMinMSE` or `Lambda(6)`). This model would have 48 nonzero coefficents (`DF(6)`). However, our estimate of the model's accuracy contains some uncertainty. The `lasso` output also gives the standard error of the cross validation and tells us the largest value of `Lambda` that is within one standard error of the minimum (`Lambda1SE`). In many cases we would favor this value of `Lambda` (0.3402) over the true minimum (0.0439). Why?

The goal of this exercise is to reduce the variance of and improve the generalization of the model. While maintaining a low lambda reduces the error of our model, it can also lead to overfitting which prevents the model from producing accurate predictions on future data. While lambda = 0.0439 does offer the lowest MSE, it may not adequately penalize the larger coefficients or reduce the number of predictive variables to produce a generalizable model. By choosing the largest lambda within one SE of our minimum, we select a model with relatively low error and greater regularization.

The matrix `B` returns the actual coefficients for each value of `Lambda`. Let's sort each model's coefficients by size and plot them to see the effects of regularization.

```
plot(sort(abs(B)))
```

```
Warning: MATLAB has disabled some advanced graphics rendering features by switching to software OpenGL. For
more information, click here.
```



Explain what you're seeing in this plot.

Each line in the plot corresponds to a specific lambda and is plotted for the 56 individual fitted coefficients along the x-axis and the absolute value of those fitted coefficients along the y-axis (order is not necessarily conserved so we are not always comparing the same coefficient at 'x' as we move between lines). The sorting function is used before plotting because we want to best visualize the variation of all coefficient values as lambda changes. We can see that as lambda increases, regularization becomes more apparent as more coefficients are reduced to zero and the remaining coefficients are diminished to values closer to zero.

Now we can identify the nonzero interactions from our regularized model. Let's put them in a table for display.

```
nonzeros = abs(B(:,8)) > 1e-5;
effect_size = B(nonzeros,8);
effect_name = treatment(nonzeros);
table(effect_name, effect_size)
```

ans = 25×2 table

	effect_name	effect_size
1	'C1'	5.7913
2	'C3'	1.8711
3	'C4'	4.4134

	effect_name	effect_size
4	'G8'	0.0524
5	'OP'	-0.0340
6	'TC'	-1.8743
7	'TR'	-0.2754
8	'C1:OP'	0.3665
9	'C1:TR'	5.9844
10	'C1:LN'	1.5149
11	'C3:G8'	0.4121
12	'C3:OP'	-4.1638
13	'C3:FN'	1.2351
14	'C3:LN'	3.1175

⋮