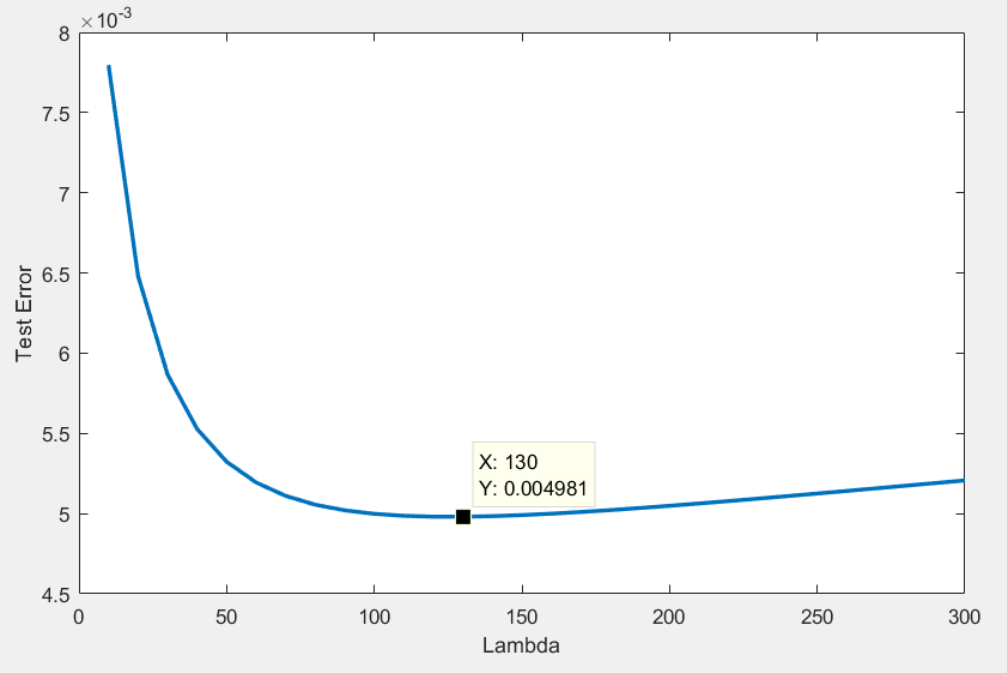
Alex Elkman
11/16/2016
ECS 171 FQ16

HW3

1. I decided to use ridge regression method for this problem. Ridge regression is a method of regularizing the least squares problem as a close form solution to the linear regression model. It incorporates a penalty function R which is the squared L2 norm, which arrives at a closed form solution: $w = (X^T X + \alpha I)^{-1} X^T Y$
   Thus, if we pick alpha (or lambda) = 0, then we have the standard OLS solution to the linear regression model, and changing values of alpha > 0 incorporates regularization into the model, ensuring all the eigenvalues are strictly greater than 0. It shrinks coefficients and in turn, reduces sensitivity to the data.

   Using a KFold estimator for the test error at k = 10, we can see how the error changes with varying Lambda:



   We can see the **optimal lambda parameter value is in the range of 120-130** for the dataset with random shuffling, a **test error of 0.004981**, and a **training error of 0.001898.** Since ridge regression was used, none of the weights were zeroed out completely so this section of the response is not answerable.

2. Using bootstrapping, the 95% quantile result is **[0.3585, 0.4428]**. This was calculated from running 100 iteration of the bootstrapping samples, recalculating the weights each time and putting the expected outputs into a matrix. Then, the matrix is run into the MATLAB 'quantile' function to predict the results from the outputs given the 95% CI inspection. See code for further information.

3. Using ridge regression with lambda = 130, and the training weights calculated from Problem 1, the **expected growth rate for the mean gene data is 0.3125.** This was achieved by simply using the MATLAB 'mean' function on the test_x matrix, and then passing that into the 1st order polynomial equation for prediction. See code for details.

4. To save time and not have to use the wrapper method, I picked an arbitrary threshold for the abs(weights) to be removed if less than 0.0009, this number came from observations of the values in the weights matrix. This reduced to 586 features.

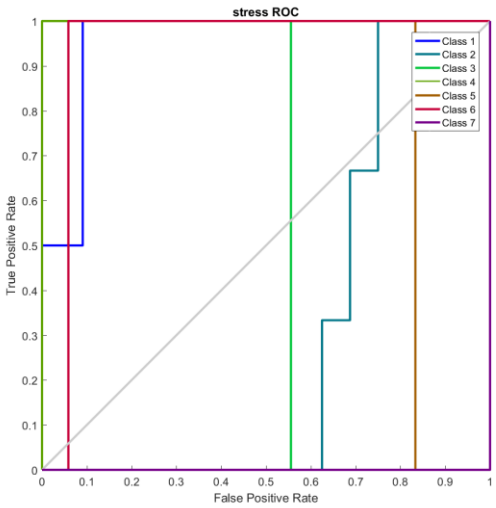
   Upon using the plotroc function built into MATLAB (not the one found on the libsvm site), and using the trainsvm and predictsvm functions from the libsvm library, my results from the graphs seem wrong. This is because there needs to be a probabilistic distribution for the outputs created from the predict_y matrix, which in my case is simply binary values. I looked online on how to achieve this, but could not find a simple solution. The graphs are displayed show a basic idea of the curves, however do not show as well what ROC should be.

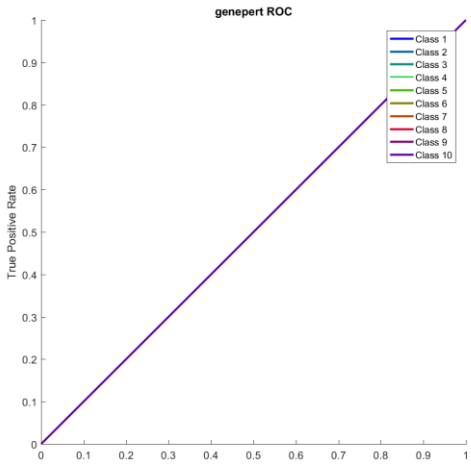
   [K = 10] Classification Performance:

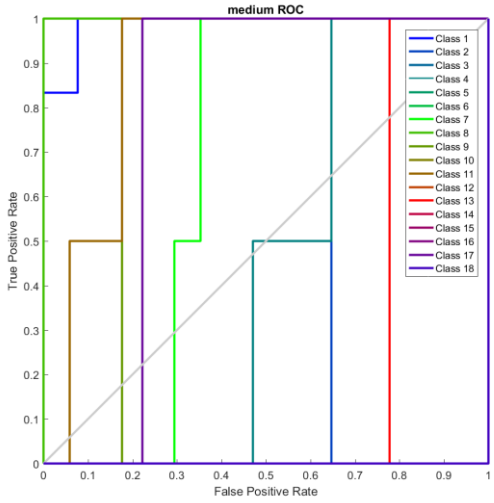
Strain	Medium	Stress	Gene Perturbation
82.6316%	73.1579%	74.2105%	81.5789%

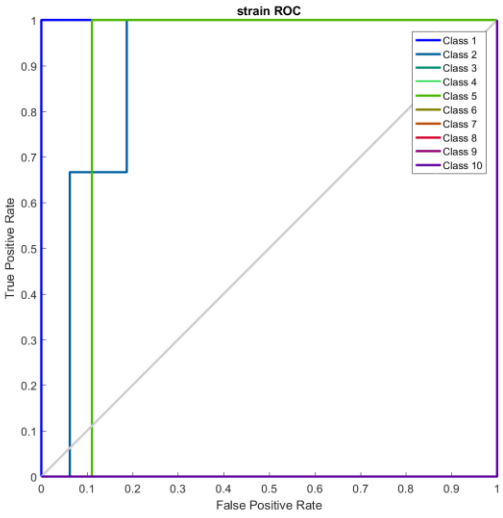
   AUC is calculated using trapz function for the different classes. Results were as follows:

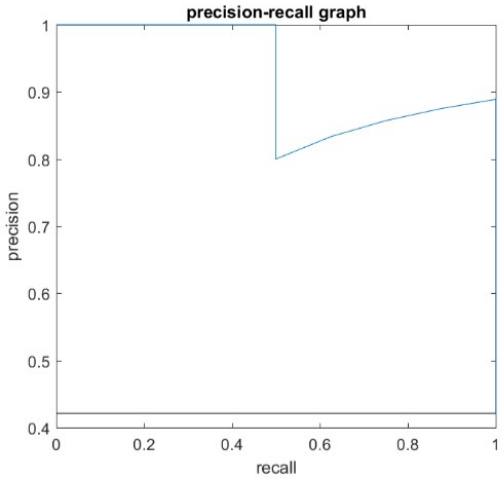
Strain	Medium	Stress	Gene Perturbation
0.5448	0.5963	0.7004	0.9869
0.8607	0.8352	0.6703	0.6402
0.9987	0.6935	0.6584	0.5870
0.5540	0.5351	0.6821	0.9610
0.7284	0.8405	0.8079	0.6301
0.5665	0.9281	0.7987	0.6875
0.8519	0.5485	0.8426	0.8224
0.6531	0.6959		0.6582
0.5322	0.6880		0.5962
0.5869	0.7578		0.6166
	0.9471		
	0.8209		
	0.6016		
	0.5344		
	0.9001		
	0.7115		
	0.9462		
	0.9960		











PR Curve for Stress Class 1

AUPRC:

Strain	Medium	Stress	Gene Pert
0.6385	0.5924	0.8983	0.6615
0.7049	0.9364	0.8206	0.8022
0.8599	0.9891	0.5724	0.6179
0.8784	0.9678	0.9964	0.7910
0.5796	0.5487	0.8885	0.9291
0.8254	0.6219	0.9677	0.7585
0.7055	0.9856	0.5778	0.5146
0.8606	0.7571	0.8535	0.7568
0.8654	0.5771		0.7440
0.6153	0.7593		0.8713
	0.8966		0.7401
	0.5746		0.5303
	0.7517		
	0.8447		
	0.9616		
	0.7916		
	0.9937		
	0.7535		

5. I found the accuracy to be a mean of 67% when combining the two classifiers of medium and stress. I combined them by creating a map of all 126 combinations of the two classifiers, and then creating a new output vector corresponding to these unique mapping values. This was then used to create the train_y and test_y vectors for the SVM. We need to keep in mind that this accuracy is for predicting two classifiers both correctly at the same time, and doesn't account for situations where one of them may be correctly guessed, but the other is wrong. To fully check the accuracy of both classifiers, we need to split up the predicted outcomes into two vectors corresponding to their unique outputs, and then recalculate the error for them individually based on this. However, we know that the accuracy must in fact be better than this mean since there are only cases of missed accuracy by examining the combined output (example, classifier A is correct, classifier B is incorrect, so the SVM registers the output as wrong, while in fact one of them was correct).

AUC Data:

```
Columns 1 through 13

   0.3019    0.3920    0.6914    0.6486    0.2602    0.6385    0.3602    0.4451    0.3109    0.3470    0.5986    0.2892    0.4302

Columns 14 through 23

   0.5155    0.3384    0.5636    0.5168    0.4589    0.3585    0.4357    0.5967    0.2659    0.5234
```

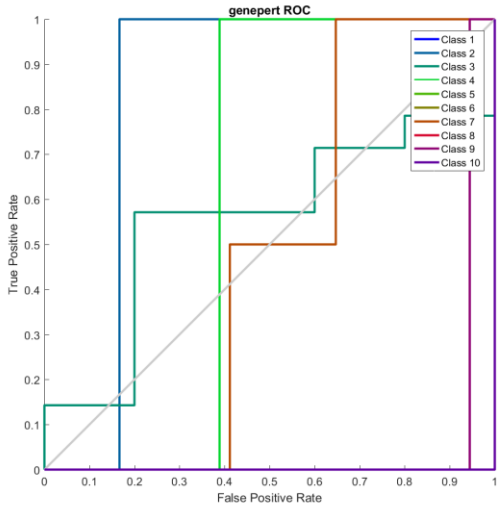
AUPRC Data:

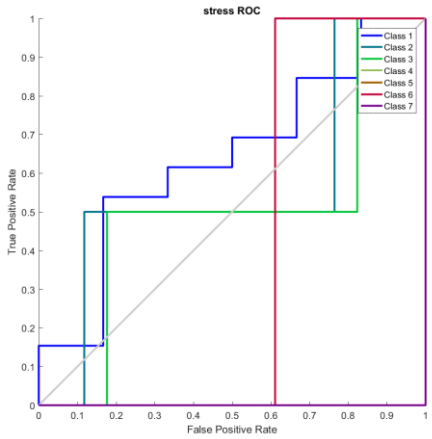
```
Columns 1 through 13

   0.2358    0.3869    0.6435    0.5792    0.2236    0.5274    0.2517    0.4184    0.5909    0.2075    0.5685    0.5527    0.2872

Columns 14 through 23

   0.6108    0.5077    0.5998    0.5208    0.4182    0.5749    0.2408    0.2115    0.3511    0.4191
```

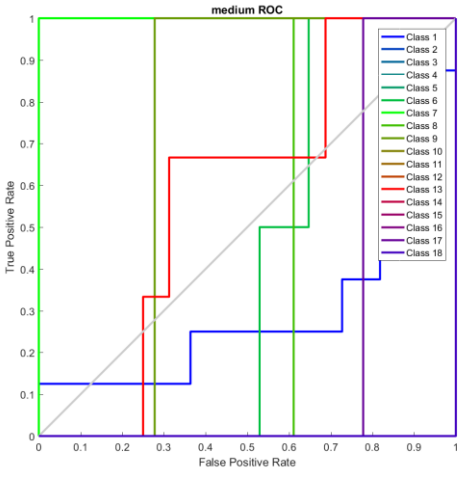
6.

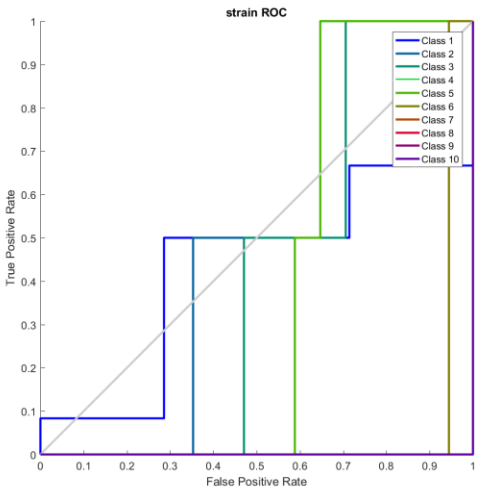
AUC is calculated using trapz function for the different classes. Results were as follows:

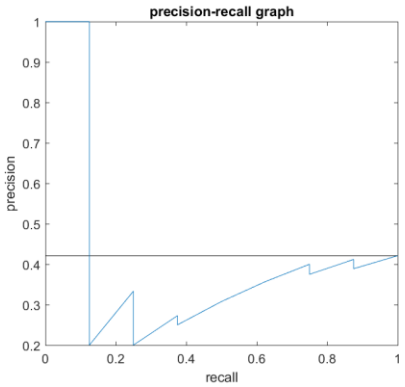
Strain	Medium	Stress	Gene Perturbation
0.6786	0.7973	0.3853	0.3152
0.5809	0.4069	0.3184	0.4312
0.4886	0.8695	0.7852	0.7265
0.6849	0.4198	0.3283	0.6294
0.6372	0.4306	0.6997	0.8218
0.6982	0.3051	0.5736	0.7433
0.3749	0.4804	0.7818	0.3619
0.7044	0.4647	0.5151	0.8829
0.7082	0.6519		0.7432
0.8581	0.5379		0.3083
	0.4118		0.7075
	0.5357		0.5913
	0.3928		
	0.7994		
	0.3899		
	0.7077		
	0.7501		
	0.3089		

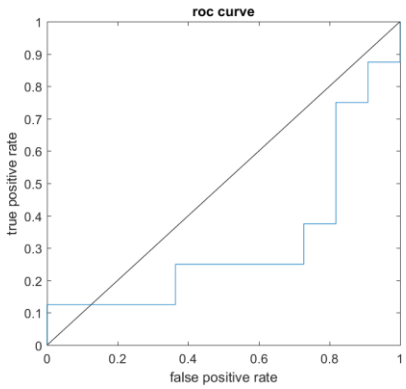












Graph of Medium Class 1 for PR/ROC

AUPRC:

Strain	Medium	Stress	Gene Pert
0.6877	0.4098	0.5107	0.5767
0.6046	0.5591	0.3400	0.6960
0.3090	0.4765	0.3397	0.6992
0.6508	0.7297	0.3362	0.7316
0.3079	0.4300	0.7502	0.6990
0.6419	0.7219	0.7847	0.3032
0.7390	0.5914	0.4957	0.7600
0.5159	0.6597	0.4568	0.3090
0.6157	0.4718		0.3147
0.5930	0.3046		0.6557
	0.3962		0.5692
	0.5537		0.5623
	0.3121		
	0.5750		
	0.4381		
	0.6481		
	0.7407		
	0.3123		
	0.4706		

The accuracy of the tests with the PR method were fairly accurate, I saw averages of around 70% with the results from predictsvm. This is very impressive considering how many dimensions reduced, from over 4000 down to just 3.

Note: Only one graph for PR is shown for this problem and the prior, this is because the graph only shows one class, and to show all the classes would take 48 graphs since there are that many classifiers in all of the different identifiers.