Inclass lab. Please hand in table at the end of the lab.

**STEP 1:**
First you need to fire up matlab and make a directory to work in. Then download the files from www.rpi.edu/~bennek/class/mds/Aqua-all.txt If you are using one of the class computers, do the following.

Login in locally using these credentials:

```plaintext
    userID: guest
    password: bio.info
```

Have the students open a local terminal window by right clicking on the desktop background and selecting "Open Terminal". Make a directory in your RCS account

```plaintext
    mkdir mds
    cd mds
```

In the terminal window at the prompt enter

```plaintext
    ssh -X -l <rcs_userID> rcs-sun1.rpi.edu
```

The minus options are an uppercase X and a lowercase L. Once logged into rcs-sun1.rpi.edu, at a Unix prompt enter

```plaintext
    /campus/mathworks/matlab/7sp3/bin/matlab
```

Matlab should display on the local machine with the students working on the RCS Sun system from inside their RCS accounts.

Copy

```plaintext
    /afs/rpi.edu/home/68/bennek/public_html/class/mds/Aqua-all.txt
```

**STEP 2:**
Load in the data

```plaintext
    Data=load('Aqua-all.txt');
```

The matrix Data now contains the Aqua Sol Data. The first column is an id number. The second is the response. The remaining are the 525 descriptors.

First let’s break the data up into training and testing sets. The training data will be the first 100 points. Xtrain contains the descriptors, Ytrain is the response.

```plaintext
    Xtrain=Data(1:100,3:527);
    Ytrain=Data(1:100,2);
```

The remaining points are the testing data:

```plaintext
    Xtest=Data(101:197, 3:527);
    Ytest=Data(101:197,2);
```

**Step 3:**
Let’s duplicate the models we tried in class
First is regular ridge regression with \( \lambda = 10; \)
\[
\min_w L_\lambda(w, S) = \lambda \|w\|^2 + \|y - Xw\|^2
\]
Write down the optimality condition for \( w \) for this problem?

Now compute the optimal \( w \) using Matlab. Enter the following in Matlab
\( L=10; \)
\[
w1 = \text{inv}(Xtrain'*Xtrain + L* eye(525))*(Xtrain'*Ytrain)
\]
Let’s see how well it works. Calculate the predicted values for the test set.
\[
Ypred1=Xtest*w1;
Ypred1_trn=Xtrain*w1
\]
Compute the mean square error of the train and test sets.
\[
\text{MSE1}=\text{mean}((Ypred1-Ytest).^2)
\]
\[
\text{MSE1_trn}=\text{mean}((Ypred1_trn-Ytrain).^2)
\]
Enter the error in the appropriate entry in the table on the last page.

**Step 4:**
As discussed in class the model can be improved by taking into account a bias turn so \( f(x)=xw+b. \)
For Ridge regression this can be accomplished by centering \( X \) and \( Y. \) Calculate the mean of \( Xtrain \) and \( Ytrain. \)
\[
\text{mu}=\text{mean}(Ytrain);
\text{Xbar}=\text{mean}(Xtrain);
Xtrain2=Xtrain-ones(100,1)*Xbar;
Ytrain2=Ytrain-mu;
Xtest2=Xtest-ones(97,1)*Xbar;
\]
Create the new model
\[
w2 = \text{inv}(Xtrain2'*Xtrain2 + L* eye(525))*(Xtrain2'*Ytrain2)
\]
To predict the new one, we need to know the bias \( b. \)
\[b=\text{mu};\]
The new prediction and the error are:
\[
Ypred2=Xtest2*w2+b;
Ypred2_trn=Xtrain2*w2 +b;
\text{MSE2}=\text{mean}((Ypred2-Ytest).^2)
\text{MSE2_trn}=\text{mean}((Ypred2_trn-Ytrain).^2)
\]
Can you improve it by playing around with \( L? \) Enter your results in the table on the last page.

**Step 5:**
According the theory, calculating the function in the dual space should work exactly the same.
Let’s give it a try. The optimal coefficients in the dual space are
\[ \alpha = (G + \lambda I)^{-1} y \] where \( G = XX' \) and
\[ w = X'\alpha \]

In Matlab
\[ G=Xtrain2*Xtrain2'; \]
\[ a=\text{inv}(G+L*\text{eye}(100))*Ytrain2; \]
\[ w3=Xtrain2'*a; \]

Compare \( w3 \) to \( w2 \)? Are they identical? Why or why not?

Computationally is there any reason to prefer one method over the other?

**Step 6:**
The advantage of the dual approach is that one can replace the Gram matrix \( G \) with any appropriate kernel matrix \( K \). This corresponds to mapping the data to feature space and finding the best predictive model in that space.

The ridge regression algorithm in Steps 4, consisted of the following Matlab commands.

```
%Center the Y data
mu=mean(Ytrain);
Ytrain2=Ytrain-mu;

%Center the X data
Xbar=mean(Xtrain);
Xtrain2=Xtrain-ones(100,1)*Xbar;
Xtest2=Xtest-ones(97,1)*Xbar;

% Compute W,b
w2 = inv(Xtrain2'*Xtrain2+ L* eye(525))*(Xtrain2'*Ytrain2)
b=mu;

%Compute the predictions
Ypred2=Xtest2*w2+b;
Ypred2_trn=Xtrain2*w2 +b;
MSE2=mean((Ypred2-Ytest).^2)
MSE2_trn=mean((Ypred2_trn-Ytrain).^2)
```

Now we would like to make this algorithm work in feature space by using the kernel trick.

Let’s try this process for a degree \( d \) polynomials. Try a quadratic polynomial first.

Set the degree
\( d=2 \)
Compute the polynomial
\[ K=(Xtrain2*Xtrain2' + 1).^d; \]
Now we need to replace each of the steps with the kernel space.

Centering the Y data is not changed.
But now we cannot center the X data directly in feature space.
We cannot center X data directly in feature space.  We must instead figure out the equivalent kernel operations.

Note $\bar{x} = \frac{1}{\ell} \sum_{i=1}^{\ell} x_i = \frac{1}{\ell} e'X$

$X_c = X - e\bar{x} = X - e(\frac{1}{\ell} e'X) = (I - \frac{1}{\ell} ee')X$

$K_c = (I - \frac{1}{\ell} ee')X * X'(I - \frac{1}{\ell} ee')$

$= (I - \frac{1}{\ell} ee')K(I - \frac{1}{\ell} ee')$

$K_c$ is now the centered kernel

Create a matlab expression that “centers” the kernel.
$K_c =$ ?

The next is to compute a.  The only change is now we use the dual version.
Now that we have the centered kernel we can go ahead and fit a degree d polynomial.
$a = \text{inv}(K_c + L*\text{eye}(100)) * Y\text{train2};$

We can’t compute w, but we can predict the test points.  Note we have to be careful to center the testing kernel before we predict. Note that since we centered the training data before computing the kernel, we center the testing data in the same way.  The uncentered test kernel is

$K_{test} = (X_{test} * X_{train2'} + 1)^d;$

Before we used the matlab command $X_{test2} = X_{test} - \text{ones}(97,1) * X_{bar}$, to center the test data.  Now we need to compute the kernel equivalent of $K_{test} = X_{test2} * X_{train2'};$

Note that for prediction, for the case of training with 100 points and testing with 97, the test kernel centering becomes:

$K_{testc} = (X_{mt} - e_{97}\bar{x})*(X_{nn} - e_{100}\bar{x}) \text{ where } \bar{x} = \frac{1}{\ell} e_{100}'X_{nn}$

$= (X_{mt} - \frac{1}{\ell} e_{97}e_{100}'X_{nn})*X_{nn} ' (I - \frac{1}{\ell} e_{100}e_{100})$

$= X_{mt}X_{nn} ' (I - \frac{1}{\ell} e_{100}e_{100}) - \frac{1}{\ell} e_{97}e_{100}'X_{nn}X_{nn} ' (I - \frac{1}{\ell} e_{100}e_{100}) '$

$= (K_{mt} - \frac{1}{100} e_{97}'e_{100} 'K)) (I - 1/100)$

Multiplying out and putting it in Matlab:
$K_{testc} = (K_{test} - 1/100*\text{ones}(97,1)*K_{bar}')*(\text{eye}(100)-1/100);$

The prediction is then
Ypred6 = Ktest * a + b;
Ypred6_trn = Kc * a + b;
MSE6 = mean((Ypred6 - Ytest).^2)
MSE6_trn = mean((Ypred6_trn - Ytrain).^2)

Repeat this experiment with \( L = 50 \). Repeat this experiment for \( d = 3 \).

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>MSE train</th>
<th>MSE test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridge Regression</td>
<td>( L = 10 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ridge Regression</td>
<td>( L = )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ridge Regression with bias</td>
<td>( L = 10 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ridge Regression with bias</td>
<td>( L = )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual Ridge Regression with bias</td>
<td>( L = 10 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel Ridge Regression</td>
<td>( L = 10 ), ( d = 2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel Ridge Regression</td>
<td>( L = 50 ), ( d = 2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel Ridge Regression</td>
<td>( L = 50 ), ( d = 3 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HINT: Here is one way to calculate the centered training kernel
\[
K_{bar} = K \cdot \text{ones}(100,1); \quad K_c = (K - 1/100 \cdot \text{ones}(100,1) * K_{bar}) * (\text{eye}(100) - 1/100);\]