Homework 6

Nate Mankovich
Math Modeling

May 11, 2018

Part A: Theory and Foundations

Problem 1

The RBF linear system requires the solution of the overdetermined least squares problem

$$y = \Phi w$$

a) Show that by applying the operator Φ^T to both sides of the equation that

$$w = (\Phi^T\Phi)^{-1}\Phi^T y$$

Note that this derivation is different from the one provided in class, don't reproduce that here.

$$\begin{aligned} \Phi w &= y \\ \Phi^T\Phi w &= \Phi^T y \\ &\text{Let us assume } \Phi^T\Phi \text{ is invertible.} \\ (\Phi^T\Phi)^{-1}(\Phi^T\Phi w) &= (\Phi^T\Phi)^{-1}\Phi^T y \\ w &= (\Phi^T\Phi)^{-1}\Phi^T y \end{aligned}$$

b) Verify that the actual output of the model is $\bar{y} = \Phi w$ where

$$\bar{y} = \Phi(\Phi^T\Phi)^{-1}\Phi^T y$$

$$\begin{aligned} \bar{y} &= \Phi(\Phi^T\Phi)^{-1}\Phi^T y \\ &= \Phi((\Phi^T\Phi)^{-1}\Phi^T y) \\ &= \Phi w \end{aligned}$$

c) Let $\mathbb{P} = \Phi(\Phi^T\Phi)^{-1}\Phi^T$. Show that $\mathbb{P}$ is an orthogonal projection, i.e.,

i) $\mathbb{P} = \mathbb{P}^2$ (projection criterion)

$$\begin{aligned} \mathbb{P}^2 &= (\Phi(\Phi^T\Phi)^{-1}\Phi^T)(\Phi(\Phi^T\Phi)^{-1}\Phi^T) \\ &= \Phi(\Phi^T\Phi)^{-1}(\Phi^T\Phi)(\Phi^T\Phi)^{-1}\Phi^T) \\ &= \Phi(\Phi^T\Phi)^{-1}\Phi^T \\ &= \mathbb{P} \end{aligned}$$

ii) $\mathbb{P} = \mathbb{P}^T$ (criterion for projection to be orthogonal)

$$\begin{aligned}
\mathbb{P}^T &= (\Phi(\Phi^T\Phi)^{-1}\Phi^T)^T \\
&= ((\Phi^T\Phi)^{-1}\Phi^T)^T\Phi^T \\
&= \Phi^{T^T}(\Phi^T\Phi)^{-1^T}\Phi^T \\
&= \Phi(\Phi^T\Phi)^{-1^T}\Phi^T \\
&= \Phi(\Phi^T\Phi)^{-1}\Phi^T \text{ because } \Phi^T\Phi \text{ is symmetric.} \\
&= \mathbb{P}
\end{aligned}$$

Problem 2

Show how the problem

$$y = \Phi w$$

can be solved using

a) the SVD

Do the SVD on Φ. This yields:

$$\Phi = U\Sigma V^T$$

Then we have

$$\begin{aligned}
y &= U\Sigma V^T w \\
U^T y &= \Sigma V^T w \\
\Sigma^\dagger U^T y &= V^T w \\
V\Sigma^\dagger U^T y &= w
\end{aligned}$$

b) the QR decomposition

Do the QR decomposition on Φ. This yields:

$$\Phi = QR$$

Then we have

$$\begin{aligned}
y &= QRw \\
Q^T y &= Rw \\
R^\dagger Q^T y &= w
\end{aligned}$$

Problem 3

Consider the set of input-output pairs $\{(0,1),(2,4),(3,8),(4,12),(5,17)\}$. Let the model for this data be

$$f(x) = a\sqrt{x} + bx^{3/2}$$

Find a and b.

$$a\sqrt{x} + bx^{3/2} = f(x)$$

$$[\sqrt{x} \mid x^{3/2}] \begin{bmatrix} a \\ b \end{bmatrix} = f(x)$$

$$\begin{bmatrix} 0^{1/2} 0^{1/2} \\ 2^{1/2} 2^{3/2} \\ 3^{1/2} 3^{3/2} \\ 4^{1/2} 4^{3/2} \\ 5^{1/2} 5^{3/2} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 8 \\ 12 \\ 17 \end{bmatrix}$$

$$\begin{bmatrix} 0^{1/2} 0^{1/2} \\ 2^{1/2} 2^{3/2} \\ 3^{1/2} 3^{3/2} \\ 4^{1/2} 4^{3/2} \\ 5^{1/2} 5^{3/2} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 8 \\ 12 \\ 17 \end{bmatrix}$$

$$Ax = b$$

Now we try to minimize$||Ax - b||^2$

This amounts to solving $A^T A\hat{x} = A^T b$ which is the same as

$$A^T(A\hat{x} - b) = 0$$

$$\hat{x} = \begin{bmatrix} -0.1878 \\ 1.5584 \end{bmatrix}$$

Therefore

$$a = -0.1878 \text{ and } b = 1.5584$$

Part B: Computing

Problem 1

a) Load and plot the time series data from canvas. This temperature data was collected by the CSU Weather Station over September 2017 at 5 minute intervals.

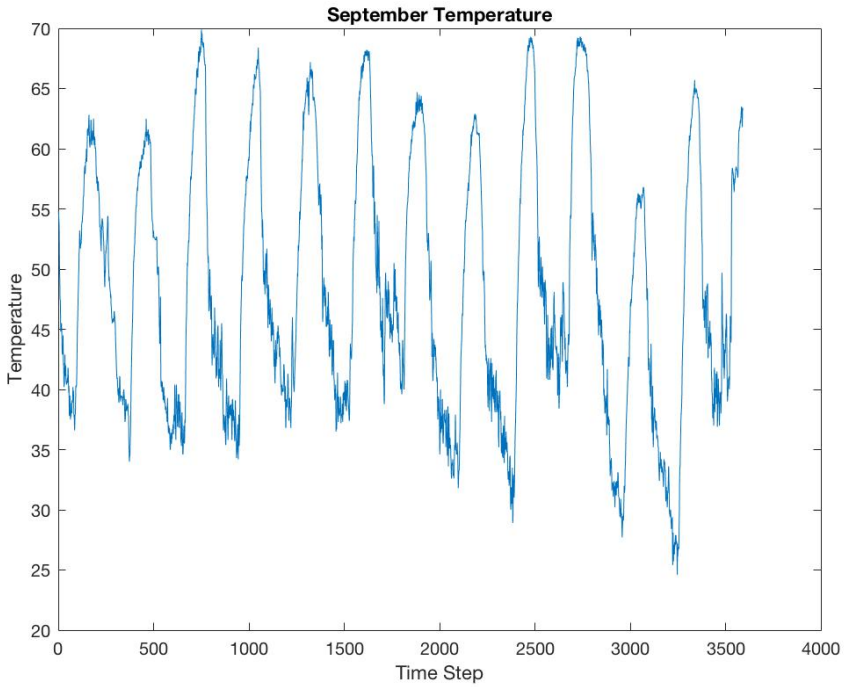

Figure 1: The raw temperature data.

b) Write a subroutine to create time lagged vectors of the data using time-delay embed- ding, i.e., vectors of the form

$$z_n = (x_n, x_{n-T}, x_{n-2T})$$

and plot the resulting data in three dimensions. Experiment with different values of T.

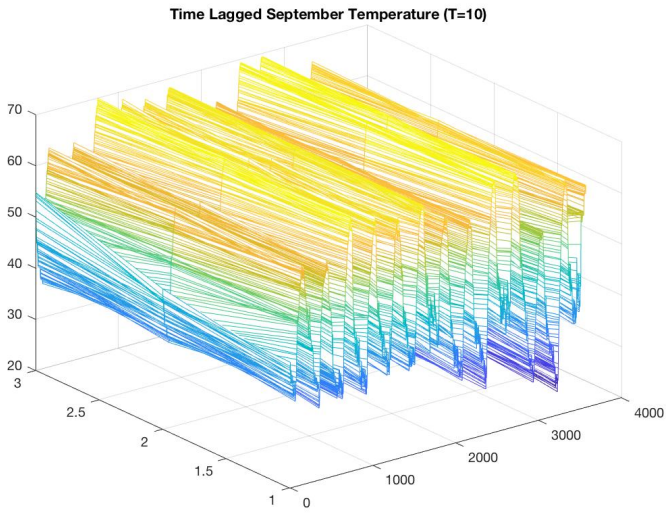

Figure 2: Time lagged data surface plot with $T = 10$.

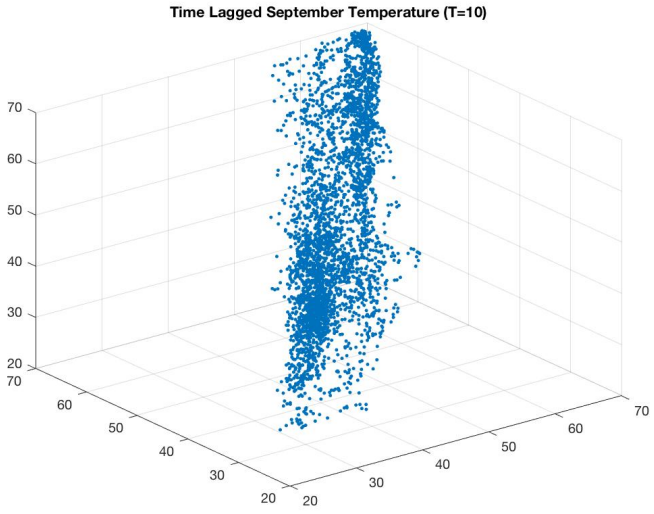

Figure 3: Time lagged data scatter plot with $T = 10$.

Problem 2

Model this time series using RBFs. The spirit of this problem is to explore various options, e.g., which RBF should you choose? How many centers? How large should the training, and validation sets be? Use the following approaches:

- Select your centers randomly from the data.
- Select your centers using LBG clustering.

Evaluate the quality of your model as a function of the number of centers in each case using the error

$$E = \frac{\sum_{n=1}^{P}(x_{n+L} - f(x_n, x_{n-L}, x_{n-2L}))^2}{\sum_{n=1}^{P}(x_{n+L} - \bar{x})^2}$$

Include plots of the error on the validation and training sets as a function of the number of centers. Save the last 500 points of your data for testing the model with this error. After you have determined the number of centers in your RBF using the training procedure above, compute the error on your test set.

We will use $L = 10$ for the data for running RBF and Multilayer perceptron. This value of T removes some noise from the data, without over-simplifying the situation. Effectively, it means we are taking one data point to be a set of temperatures over three hours of the day.

The following are the plots of the error of RBF using randomly selected centers.

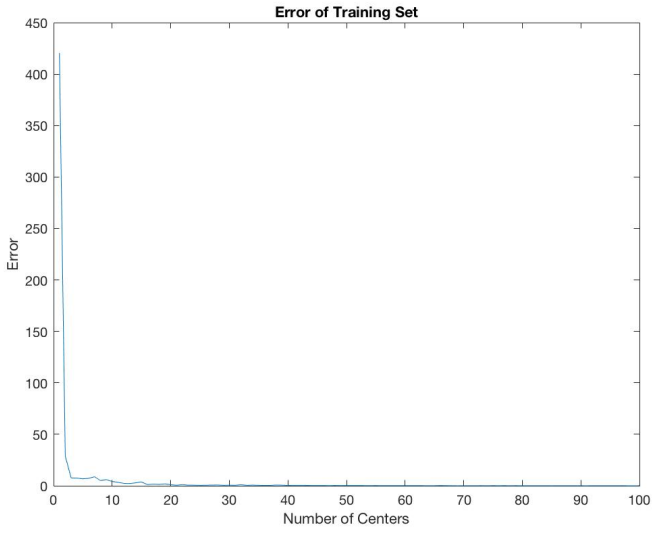

Figure 4: Training error of RBF with 1:1 training to validation ratio.

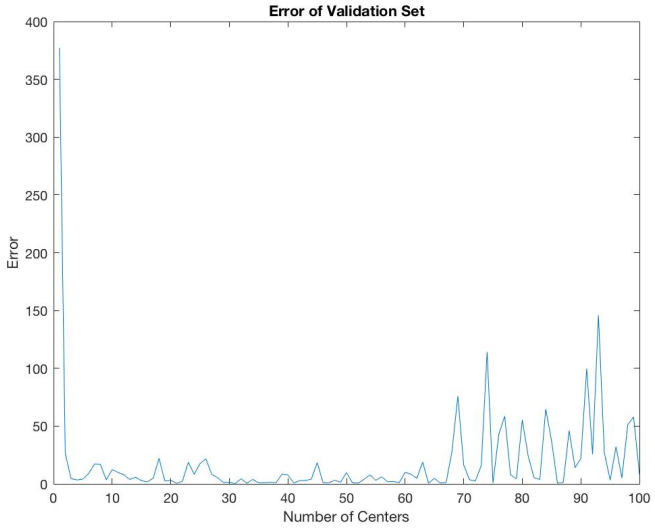

Figure 5: Validation error of RBF with 1:1 training to validation ratio.

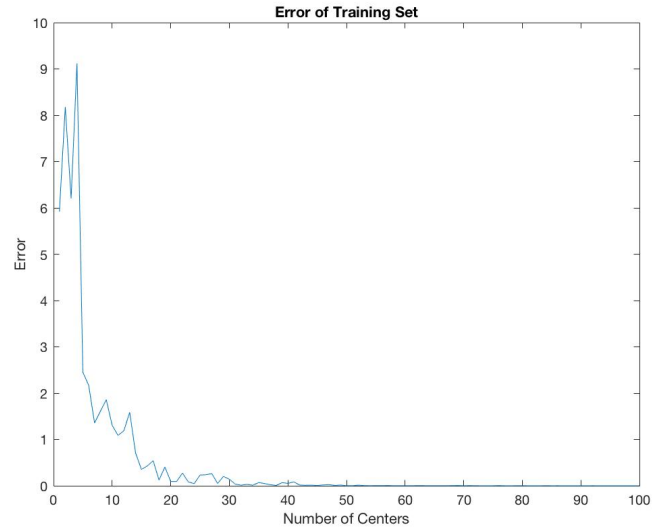

Figure 6: Training error of RBF with 1:4 training to validation ratio.

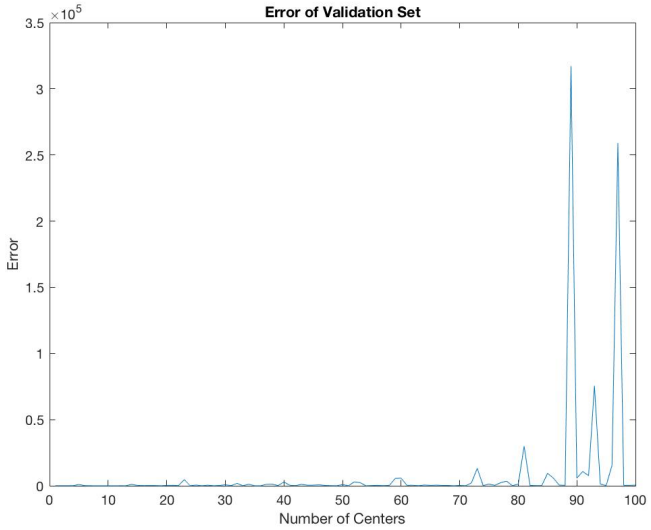

Figure 7: Validation error of RBF with 1:4 training to validation ratio.

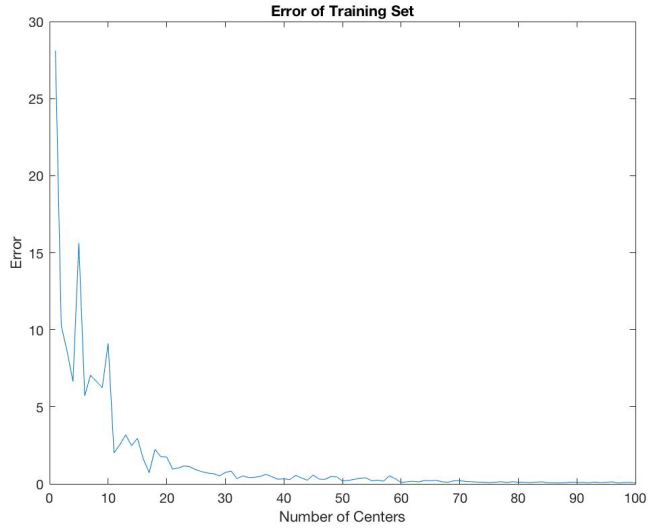

Figure 8: Training error of RBF with 4:1 training to validation ratio.

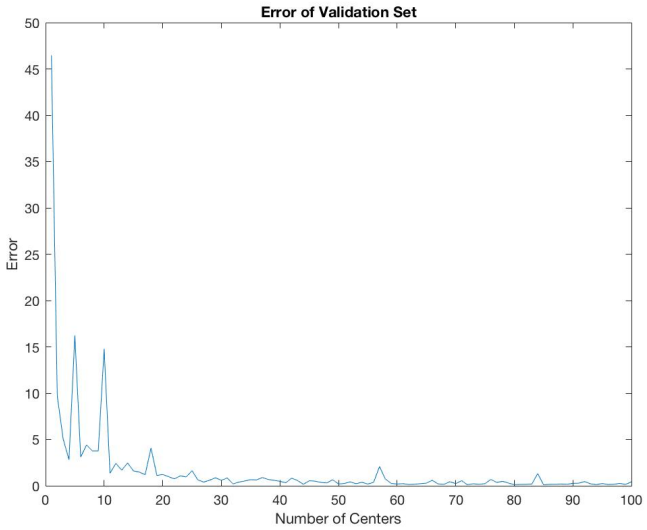

Figure 9: Validation error of RBF with 4:1 training to validation ratio.

The following are the plots of the error of RBF using LGB clustering selected centers. Due to a bug in the LGB code, we were not able to run the code with one center. Since we can notice that the more centers, the lower the validation error, we can justify leaving out running training and validation for fewer than ten centers.

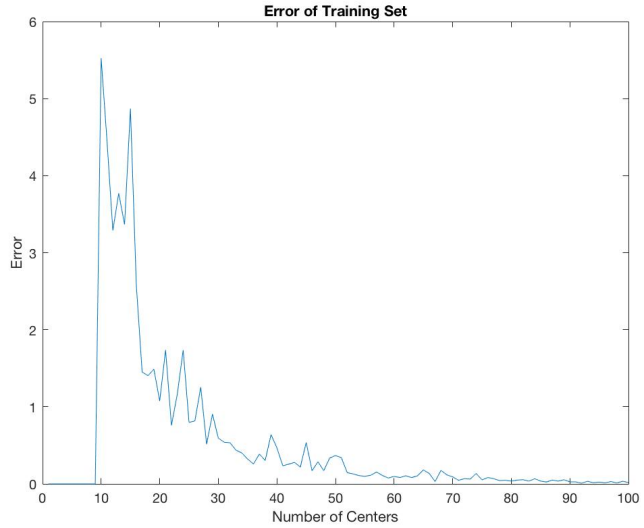

Figure 10: Training error of RBF with 1:1 training to validation ratio.

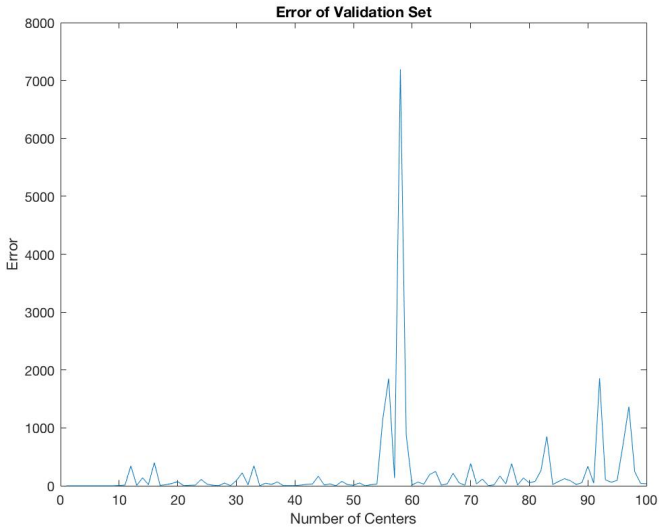

Figure 11: Validation error of RBF with 1:1 training to validation ratio.

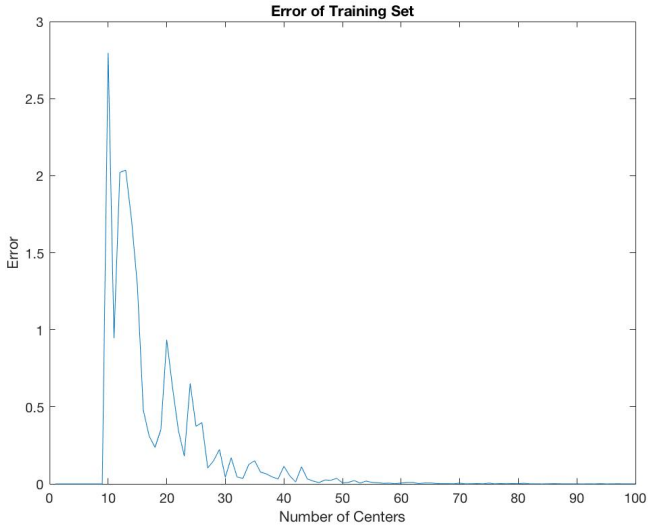

Figure 12: Training error of RBF with 1:4 training to validation ratio.

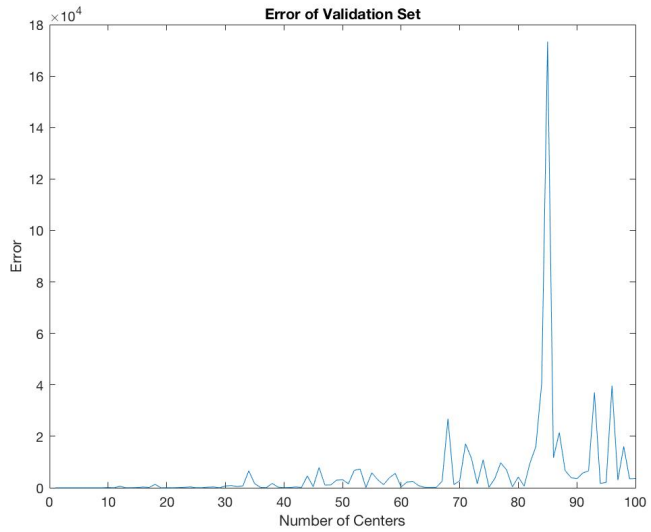

Figure 13: Validation error of RBF with 1:4 training to validation ratio.

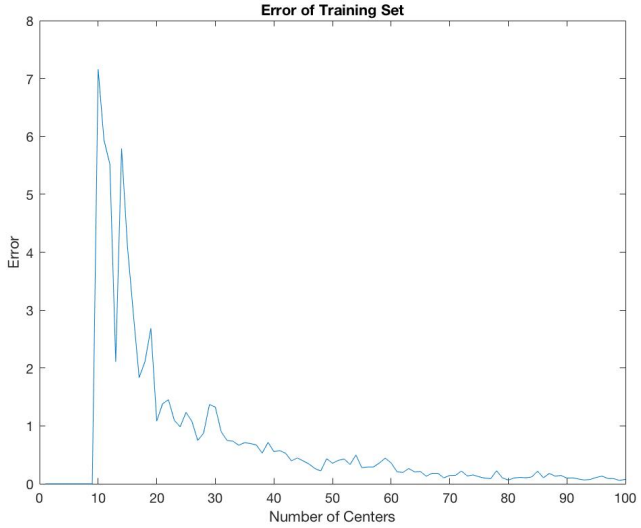

Figure 14: Training error of RBF with 4:1 training to validation ratio.

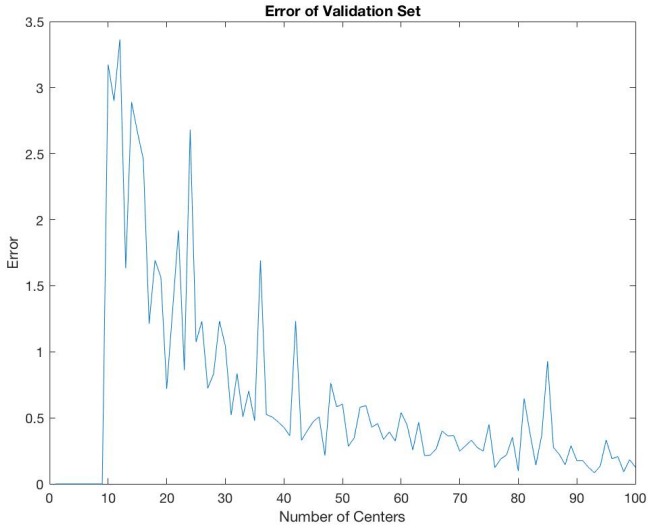

Figure 15: Validation error of RBF with 4:1 training to validation ratio.

It is clear that RBG with LGB clustering to find centers with a 4:1 training to validation set ratio is the best approach because the error in the validation set is lowest as a whole. The error stays below 3.5 for more than 10 and less than 100 centers. We attempted to run the 4:1 training and validation set ratio with 10 to 500 centers to find a minimum error in terms of the number of centers. We ran into error during this test using the SVD in LGB clustering. We were able to run this test for up to 238 centers. The minimum error occurred at 179 centers. Below is a plot of the validation error:

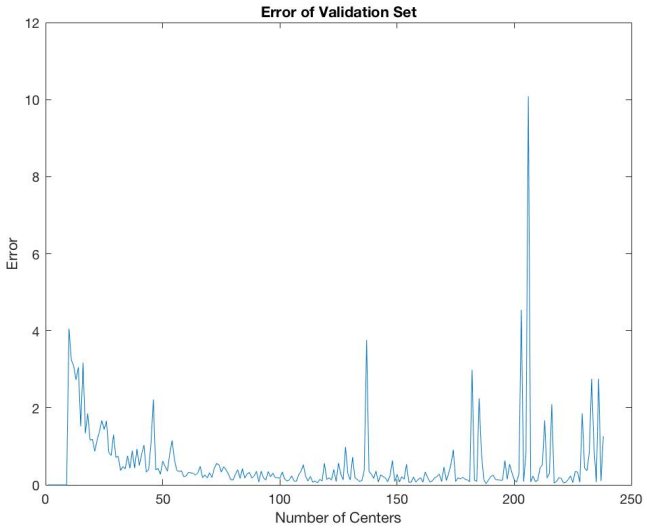

Figure 16: Validation error of RBF using LGB clustering with 4:1 training to validation ratio.

We proceeded to use 4:1 training to validation with LGB clustering and 179 centers on the test data. The error on the test data was

11.802

Problem 3

Repeat problem 2 using the multilayer perceptron. Use both the hyperbolic tangent and the relu function as the nonlinear transfer function and compare. You will need to determine the number of nodes in the hidden layer by considering the error on the validation set. Include a plot of the training error and validation error as a function of the number of nodes in the hidden layer of the network. Report your error on the test set after you have determined the number of nodes using your validation set.

We began this process with running the multilayer perceptron on a smaller data set, specifically 10 points sampled form $y = x^2$. The relu function drastically outperformed the hyperbolic tangent. We also decided to use the training validation ratio of 4:1 due to its success in the RBF case. Therefore, when determining the number of nodes, we implemented the relu function with a 4:1 training to validation ratio. Note: this choice will also benefit us when we compare the two methods in problem 4.

Since we were not advised to compute batch or online error for our back propagation step, we chose to compute online error. We used a learning rate of 10^{-6} and 100 iterations per data point.

We decided to omit testing 1,2,...,100 nodes in the hidden layer due to the long run time of the multilayer perceptron algorithm with online error. Instead we only tested with 10,20,30,40,50 nodes in the hidden layer. Below are plots of the training and validation errors for this test: Notice the decrease in training error as

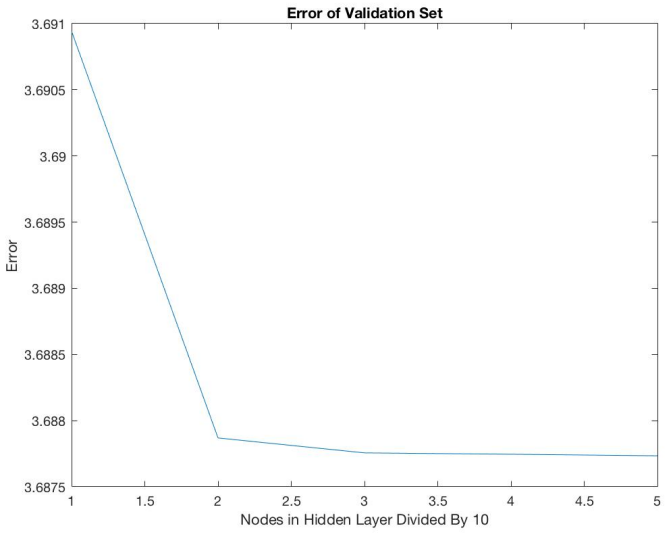

Figure 17: Training error of MLP with 4:1 training to validation ratio.

the number of nodes is increased. Also, the minimum in the validation set error with 40 nodes. Therefore we will run the test set using 40 nodes in the hidden layer. This results in the error:

3.5979334

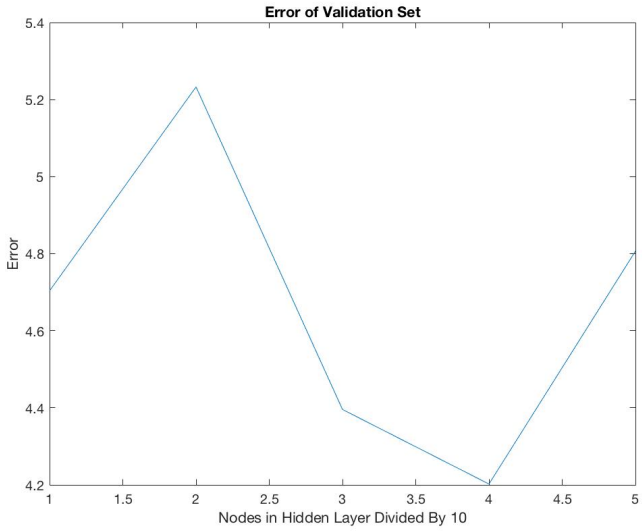

Figure 18: Validation error of MLP with 4:1 training to validation ratio.

Problem 4

Based on your results in Problems 2 and 3 which modeling approach would you recommend and why?

For this data I would suggest using MLP because it produced a lower error. However, the computation time of MLP is much slower. So, for extremely large data sets and low computation power I would suggest RBF with randomized centers because RBF is faster and we would not need to fix the error we found in problem 2. For large data sets and not much computation power I would suggest RBF with LGB clustering because RBF is faster than MLP. For anything else I'd suggest MLP, because it's accuracy certainly outweighs its slow speed.

Problem 5

a) Using the Indian Pines Hyperspectral Data Set obtain 25 points on the Grassmannian $Gr(k, n)$. These points should not share any samples.

b) For each class compute the class mean of your subspaces.

c) Build a distance matrix using the chordal metric applied to the data and the class means.

d) Use your MDS code to visualize the Grassmannian in two dimensions. Clearly label each class and the class means.

Repeat the above taking $k = 1, 5, 10$.

- $k = 1$
  Unless a) and b) were done incorrectly, we get a distance matrix of zeroes. This is because there is only one singular value from the SVD which is used to find the chordal distance. So there is really no point in running MDS. Here are plots of the means of the subspaces in the Grassmannian.

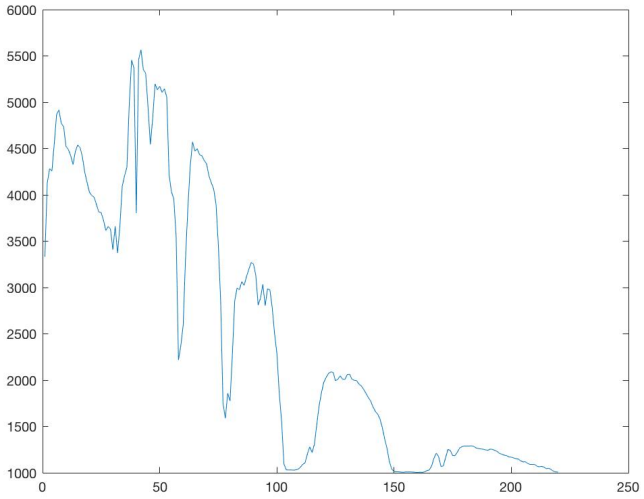

Figure 19: $\bar{X}$

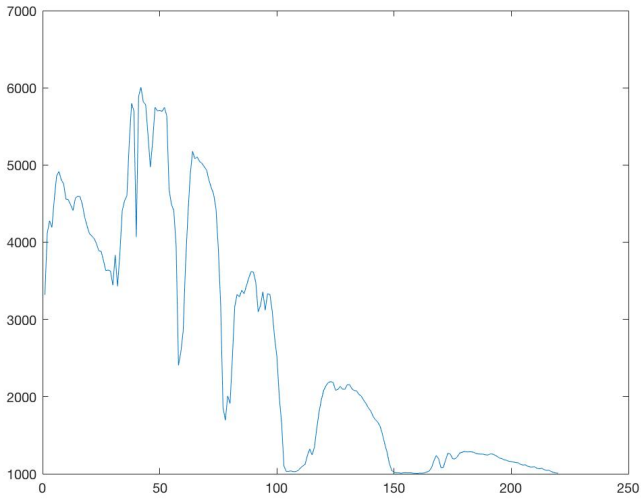

Figure 20: $\bar{Y}$

- $k = 5$

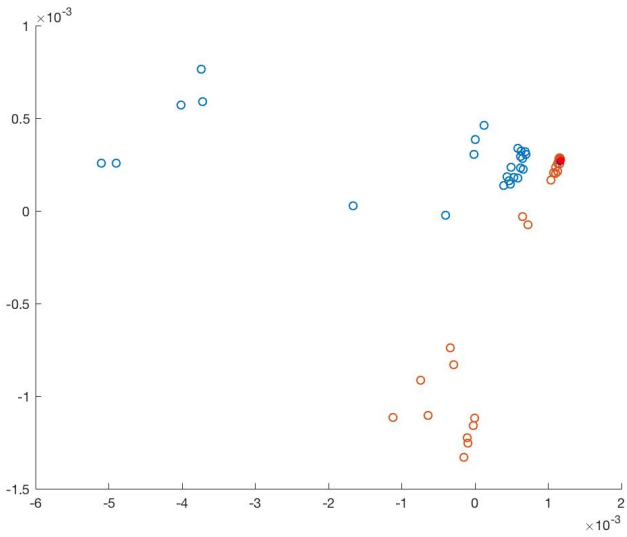

Figure 21: A plot of the MDS output with $k = 5$. One class is red, the other is blue, the class means are the filled-in circles with the respective colors.

- $k = 10$

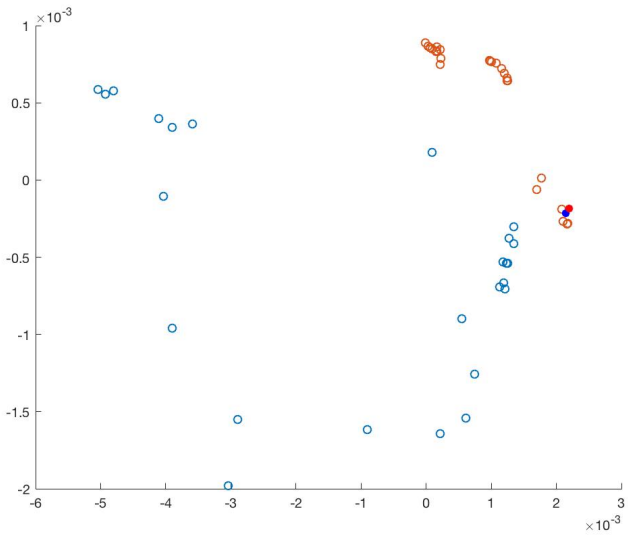

Figure 22: A plot of the MDS output with $k = 10$. One class is red, the other is blue, the class means are the filled-in circles with the respective colors.

Problem 6

Extra Credit Implement the OLS algorithm for RBFs and compare the resulting model with what you found in Problem 2.

Code

Below is the code for each of the algorithms:

Problem 1

```
 %define T
T = 10;


%import data
data = load('FoCOwxSept.mat');
X = data.Temp;
n = size(X,1);

%plot data X
plot(1:n,X)
title('September Temperature')
xlabel('Time Step') % x-axis label
ylabel('Temperature') % y-axis label

%create time lagged data using t
for i=1+2*T:n
    Z(:,i-2*T) = [X(i); X(i-T); X(i-2*T)];
end

%plot TLD data
%scatter3(Z(1,:),Z(2,:),Z(3,:),'.')
%surf(Z)
mesh(Z)
title('Time Lagged September Temperature (T=10)')
```

Problem 2

Randomized Centers

```
 T = 10; %define time lag T
iteration = 10;
for iteration=1:100
Nc = iteration; %define number of centers Nc
%alpha = 2; %defin alpha for RBF must be bigger than 0

%import data
data = load('FoCOwxSept.mat');
X = data.Temp;
n = size(X);

%create time lagged data using T
for i=1+2*T:n
    Z(:,i-2*T) = [X(i); X(i-T); X(i-2*T)];
end

%number of time lagged data points is p
p = length(Z);
```

```
%training data Ptr, validation data Pv, test data sizes Pts
% Ptr= floor((p-500)/2); %as half
% Pv=p-500-Ptr; %as half
Ptr= 614*4; %as half
Pv=p-500-Ptr; %as half
Pts=500; %prescribed in problem
Z = Z'; %make the data tall

%choose centers
%randomly
C = datasample(Z,Nc,'Replace',false);
%lgb Clustering
%Centers = LGB(Z(1:Ptr,:),Nc,Ptr);
[E(iteration,1),w] = trRBF(X,Z,Ptr,Nc,C,T);
E(iteration,2) = valRBF(X,Z,Ptr,Pv,Nc,C,w,T);

%clean up
clearvars -except E iteration T
end

plot(E(:,1))
title('Error of Training Set')
xlabel('Number of Centers') % x-axis label
ylabel('Error') % y-axis label

plot(E(:,2))
title('Error of Validation Set')
xlabel('Number of Centers') % x-axis label
ylabel('Error') % y-axis label

%training
function [E,w,Pv] = trRBF(X,Z,Ptr,Nc,C,T)
%TRAINING
%Construct Phi
for i=1:Ptr
    for j=1:Nc+1
        if j==1
           Phi(i,j) = 1;
        else
            Phi(i,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end

%do svd on Phi
[U,Sigma,V] = svd(Phi);

%find W using trainig set
for i=1:Ptr
    y(i,1) = X(i+T);
end

%find w
w = V*pinv(Sigma)*U'*y;
```

```
%compute training error
for i=1:Ptr
    for j=1:Nc+1
        if j==1
           PhiV(i,j) = 1;
        else
            PhiV(i,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end

%compute ybar
ybarV = PhiV*w;

%find validation Y
yV = X(1+T:Ptr+T);

%compute error
E = (norm(yV - ybarV)^2)/(norm(yV'-mean(Z(1:Ptr))*ones(1,Ptr)))
end

%validation
function E = valRBF(X,Z,Ptr,Pv,Nc,C,w,T)
%VALIDATION
%Construct Phi
for i=Ptr+1:Ptr+Pv
    for j=1:Nc+1
        if j==1
           PhiV(i-Ptr,j) = 1;
        else
            PhiV(i-Ptr,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end

%compute ybar
ybarV = PhiV*w;

%find validation Y
yV = X(Ptr+1+T:Ptr+Pv+T);

%compute error
E = (norm(yV - ybarV)^2)/(norm(yV'-mean(Z(Ptr+1:Ptr+Pv))*ones(1,Pv)))
end

%define the RBF
function y = phi(r)
y=r; %identity
%y = exp(-r^2/alpha^2) %exponential
%y = r^2 * ln(r) %natural log
%y = r^3 %cubic
end
```

LGB Clustering

```
  T = 10; %define time lag T
iteration = 10;
for iteration=10:100
m = iteration; %define number of centers Nc
%alpha = 2; %defin alpha for RBF must be bigger than 0

%import data
data = load('FoCOwxSept.mat');
X = data.Temp;
n = size(X);

%create time lagged data using T
for i=1+2*T:n
    Z(:,i-2*T) = [X(i); X(i-T); X(i-2*T)];
end

%number of time lagged data points is p
p = length(Z);
%training data Ptr, validation data Pv, test data sizes Pts
% Ptr= floor((p-500)/2); %as half
% Pv=p-500-Ptr; %as half
Ptr= 614*4; %as half=
Pv=p-500-Ptr; %as half
Pts=500; %prescribed in problem
Z = Z'; %make the data tall

%choose centers
%lgb Clustering
C = LGB(Z,m,p);
%Centers = LGB(Z(1:Ptr,:),Nc,Ptr);
[E(iteration,1),w] = trRBF(X,Z,Ptr,m,C,T);
E(iteration,2) = valRBF(X,Z,Ptr,Pv,m,C,w,T);

%clean up
clearvars -except E iteration T
end

plot(E(:,1))
title('Error of Training Set')
xlabel('Number of Centers') % x-axis label
ylabel('Error') % y-axis label

plot(E(:,2))
title('Error of Validation Set')
xlabel('Number of Centers') % x-axis label
ylabel('Error') % y-axis label

%training
function [E,w,Pv] = trRBF(X,Z,Ptr,Nc,C,T)
%TRAINING
%Construct Phi
for i=1:Ptr
```

```
    for j=1:Nc+1
        if j==1
           Phi(i,j) = 1;
        else
            Phi(i,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end

%do svd on Phi
[U,Sigma,V] = svd(Phi);

%find W using trainig set
for i=1:Ptr
    y(i,1) = X(i+T);
end

%find w
w = V*pinv(Sigma)*U'*y;

%compute training error
for i=1:Ptr
    for j=1:Nc+1
        if j==1
           PhiV(i,j) = 1;
        else
            PhiV(i,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end

%compute ybar
ybarV = PhiV*w;

%find validation Y
yV = X(1+T:Ptr+T);

%compute error
E = (norm(yV - ybarV)^2)/(norm(yV'-mean(Z(1:Ptr))*ones(1,Ptr)))
end

%validation
function E = valRBF(X,Z,Ptr,Pv,Nc,C,w,T)
%VALIDATION
%Construct Phi
for i=Ptr+1:Ptr+Pv
    for j=1:Nc+1
        if j==1
           PhiV(i-Ptr,j) = 1;
        else
            PhiV(i-Ptr,j) = phi(norm(Z(i,:) - C(j-1,:)));
        end
    end
end
```

```
%compute ybar
ybarV = PhiV*w;

%find validation Y
yV = X(Ptr+1+T:Ptr+Pv+T);

%compute error
E = (norm(yV - ybarV)^2)/(norm(yV'-mean(Z(Ptr+1:Ptr+Pv))*ones(1,Pv)))
end

%define the RBF
function y = phi(r)
y=r; %identity
%y = exp(-r^2/alpha^2) %exponential
%y = r^2 * ln(r) %natural log
%y = r^3 %cubic
end


%LGB clustering
%X is data
%m is number of centers
%p is number of data pts
function C = LGB(X,m,p)
%initialize centers
C = datasample(X,m,'Replace',false);

C = C';
X = X';

Er = 1000;

iteration=1;

%TEST
%hold

while Er >=630
%for iteration=1:4

    %compute S_i

    %find a distance matrix
    for i=1:m
        for n=1:p
            dist(i,n) = norm(minus(X(:,n),C(:,i)));
        end
    end

    %calculate the minimum distance
    [M,index] = min(dist);

    %find the winning indices
```

```
    for i=1:m
        for j=1:p
            comp(i,j) = isequal(i,index(j));
        end
    end

    %Si is X(:,find(comp(i,:)));


    %update centers and calculate i for each S
    %ci = (1/abs(si))sum(x in Si of x)
    for i=1:m
        S = X(:,find(comp(i,:)));
        C(:,i) = (1/length(S(1,:)))*(sum(S,2));
    end

    Er = (1/p)*sum(M.*M);
    Error(iteration) = Er;
    iteration = iteration+1;
    %repeat until E(x,I) is less than desired tolerance

    %TEST
    %visualize the result
%     for i=1:m
%         k= find(comp(i,:));
%         for j=1:N
%             VisRaw(j,k) = C(j,i);
%         end
%     end
%     Vis = uint8(VisRaw);
%     scatter3(Vis(1,:),Vis(2,:),Vis(3,:))

C = C';
end
end
```

Problem 3

```
%define T
t = 10;
%number of nodes in hidden layer
%Nh = 30;
for iteration=1:5
    Nh = iteration*10;
%learning rate
epsilon = .000001;
%number of training data points
p=3585
Ptr= 614*4; %as 4
Pv=p-500-Ptr; %as 1
Pts=500; %prescribed in problem;

%import data
data0 = load('FoCOwxSept.mat');
```

```
data = data0.Temp;

[X,Y] = TLD(data,t);

[Etr(iteration),W1,W2] = TrainMLP(X,Y,epsilon,Nh,Ptr);
Eval(iteration) = TestMLP(X,Y,epsilon, Nh,Ptr,Pv,W1,W2)

end




function [X,Y] = TLD(data,T)
n = size(data);

%create time lagged data using t
for i=1+2*T:n-T
    X(i-2*T,:) = [data(i); data(i-T); data(i-2*T)];
    Y(i-2*T,1) = data(i+T);
end
end


function [E,W1,W2] = TrainMLP(X,Y,epsilon, Nh,p)


n = size(X,2);

X = X(1:p,:);
Y = Y(1:p,:);

%%
%TRAINING

%randomize W1
for i=1:Nh-1
    for j=1:n+1
        W1(i,j) = rand(1);
    end
end
%randomize W2
for i=1:Nh
    W2(1,i) = rand(1);
end

%online updating
k=1;
mu=1
for mu=1:p
    for k=1:100
        %layer 1 state
        s1(1,1) = 1;
        s1(2:n+1,1) = X(mu,:);
        %layer 1 to layer 2
```

```
p2(1) = 1;
p2(2:Nh,1) = W1*s1;
%apply sigma (relu) to layer 2
for i=1:Nh
    if p2(i) <=0
        s2(i,1) = 0;
    else
        s2(i,1) = p2(i);
    end
end
%layer 2 to layer 3
p3 = W2*s2;
%apply sigma (relu) to layer 3
if p3 <=0
    s3 = 0;
else
    s3 = p3;
end

%backpropogation
%define Deltas
if p3 <= 0
    Delta2 = 0;
else
    Delta2 = -(Y(mu)-s3);
end
if p2 <= 0
    Delta1 = zeros(Nh,1);
else
    Delta1 = W2'*Delta2;
end

%Define new weights
for i=1:Nh-1
    for j=1:n+1
        W1(i,j) = W1(i,j) - epsilon*Delta1(i)*s1(j);
    end
end
for j=1:Nh
    W2(1,j) = W2(j) - epsilon*Delta2*s2(j);
end

%layer 1
s1(1,1) = 1;
s1(2:n+1,1) = X(mu,:);
%layer 1 to layer 2
p2(1) = 1;
p2(2:Nh,1) = W1*s1;
%apply sigma (relu) to layer 2
for i=1:Nh
    if p2(i) <=0
        s2(i,1) = 0;
    else
        s2(i,1) = p2(i);
```

```
            end
        end
        %layer 2 to layer 3
        p3 = W2*s2;
        %apply sigma (relu) to layer 3
        if p3 <=0
            s3 = 0;
        else
            s3 = p3;
        end


        %calculate error
        E(mu,k) = 1/2*abs(Y(mu)-s3);
    end
    Yhat(mu) = s3;
end

E = (norm(Y - Yhat)^2)/(norm(Y-mean(X,2)*ones(1,p))^2);
end


function E = TestMLP(X,Y,epsilon, Nh,ptr,pv,W1,W2)
n = size(X,2);

X = X(ptr+1:ptr+pv,:);
Y = Y(ptr+1:ptr+pv,:);

s1(1,1) = 1;
s1(1,1) = 1;
for mu = 1:pv
    s1(2:n+1,1) = X(mu,:);
    %layer 1 to layer 2
    p2(1) = 1;
    p2(2:Nh,1) = W1*s1;
    %apply sigma (relu) to layer 2
    for i=1:Nh
        if p2(i) <=0
            s2(i,1) = 0;
        else
            s2(i,1) = p2(i);
        end
    end
    %layer 2 to layer 3
    p3 = W2*s2;
    %apply sigma (relu) to layer 3
    if p3 <=0
        s3 = 0;
    else
        s3 = p3;
    end
    Yhat(mu) = s3;
end
E = (norm(Y - Yhat)^2)/(norm(Y-mean(X,2)*ones(1,pv))^2);
```

```
end
```

Problem 4

```
%load data
data = load('IP2classes.mat');

%set n
n=25;
%set k
k=1;

%build grassmanian
for i=1:n
    for j=1:k
        X(:,j,i) = data.X(:,i+j-1);
        Y(:,j,i) = data.Y(:,i+j-1);
    end
end

%compute mean of subspaces
for i=1:k
mX(:,i) = mean(X(:,i,:),3);
    mY(:,i) = mean(Y(:,i,:),3);
end

%build chordal distance matrix
A = X;
A(:,:,n+1:2*n) = Y;
A(:,:,2*n+1) = mX;
A(:,:,2*n+2) = mY;
for i=1:2*n+2
    for j=1:2*n+2
        [U,S,V] = svd(A(:,:,i)'*A(:,:,j));
        S1 = S/max(diag(S));
        D(i,j) = trace(eye(k) - S1.*S1);
    end
end

%do mds on chordal distance matrix
vis = MDS(D);

%visualize the data
hold
scatter(vis(1:25,1),vis(1:25,2))
scatter(vis(26:50,1),vis(26:50,2))
scatter(vis(51,1),vis(51,2),'b','filled')
scatter(vis(52,1),vis(52,2),'r','filled')



function X = MDS(D)
%make p the rank of D
```

```
p = rank(D);

%make A
A = (-1/2)*D.*D;

%make H
H = eye(p)-(1/p)*ones(p);

%compute B
B = H*A*H;

%eigenvectors and eigenvalues
[V,L] = eig(B);
dL = diag(L);

%throw away negative eigenvalues
i=1;
for k=1:p
    if dL(k) >=0
        fixedL(i) = L(k,k);
        fixedV(:,i) = V(:,k);
        i = i+1
    end
end

%vsquiggle
SfixedL = sqrt(fixedL);
for i=1:size(SfixedL,2)
    vSquiggle(:,i) = SfixedL(i)*fixedV(:,i);
end

%make a matrix with rows as first two coords of matrix
%scatter(vSquiggle(:,1),vSquiggle(:,2))
size(vSquiggle)
%rotate and reflect vSquiggle to the proper orientaion
X = [vSquiggle(:,1),vSquiggle(:,2)];
end
```