Project Assignment

System Identification

Fitting an unknown function

Cuc Diana Maria

Fildan Claudiu

Gog Ionela - Maria

Content

1. MOTIVATION

Through the process of linear regression, a future output of a function f based on the initial data is being estimated. The f function has unknown variables and coefficients which can make it very hard to know something about it. Having the known input and output data, the function can be considered as a black box ("a shape without background"). To understand better this principle, the mathematical prerequisites will be presented next.

2. MATHEMATICAL BACKGROUND

The main mathematical idea on which the whole algorithm is based upon is the principle of curve fitting and linear regression. The most notable formulas used to accomplish this are formulas (1) and (2).

$$\phi_l(x) = \exp\left[-\frac{(x_1 - c_{l,1})^2}{b_1^2} - \frac{(x_2 - c_{l,2})^2}{b_2^2}\right] \qquad (1)$$

Formula (1) presents a regressor which is represented by the exponential equation. The exponent is represented by the sum of the negative values of 2 ratios. These ratios are computed as the difference of an input coordinate value and all the center points over the square of the distance between 2 center points. The reason for having 2 ratios is the expansion of the algorithm with an input data on a 2-dimensional space.

$$Y = \phi \cdot \theta \Leftrightarrow \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_1(x_N) & \varphi_2(x_N) & \cdots & \varphi_n(x_N) \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_n \end{bmatrix} \qquad (2)$$

In formula (2) Y represents the outputs of an unknown function f after computations have been made. Y values are affected by noise so they are not exact. Phi is a matrix of regressors with the dimension of n x N, where n represents the size of the input vector coordinates and N is the number of the centers of each regressor. A regressor, also known as RBF (radial basis function), is the distance between an input point and another point which is taken as a reference and helps in creating a connection between all the above mentioned points. Theta represents the vector with n unknown parameters, which works like a coefficient adjusting the final value of the regressors, tuning them in order to obtain the outputs of function f.

3. PRACTICAL APPROACH

The theoretical background must be supported by the programming sequences.

In order to find the best approximation of the function f a linearly spaced vector of centers is taken, "Cone = linspace(Xone(1),Xone(end),Cdim)" having the dimension "Cdim = 7" such that the input data can be interpreted. By computing the distance between two centers b1=b2, "b1 = abs(Cone(2)-Cone(1))", the formula for finding the expression of each RBF can be applied. Firstly, all the combinations of all the centers on the

grid must be placed in a matrix form of 2 x N taking two indexes, k and j, forming each element of those two rows, "Cflat(1,(k-1)*Cdim+j) = Cone(k)" and "Cflat(2,(k-1)*Cdim+j) = Cone(j)". The same method is used to find all the combinations of the input data points, but placed in a vector called Xflat. Next, the matrix of regressors is computed taking again two indexes that take values up to n^2 and N^2 and the mathematical formula is written, "Phi(i,k) = exp(-(((Cflat(1,k)-Xflat(1,i))/b1)^2-(((Cflat(2,k)-Xflat(2,i))/b2)^2)", accessing in this way all the combinations of centers and input data. The output matrix is transformed in a row vector concatenating all the matrix rows, "Yflat((k-1)*Xdim+j) = Ymatrix(k,j)". After that the vector of unknown parameters is computed by dividing the matrix of regressors by the column vector of the inputs, "Theta = Phi\Yflat'". To see the correctness of the algorithm an approximated model having the outputs computed with the newly found parameters is found, "Yhat = Phi*Theta". Because all the previous work was done in vector form for simplicity , the vector of output data is turned into a matrix of dimensions n x n again, "YhatMatrix(i,j) = Yhat((i-1)*Xdim+j)". The newly approximated model is then plotted, " surf(Xone,Xtwo, YhatMatrix)"

All this process is repeated in the same manner for a different set of input data, the validation one. After finding the validation model and the approximated one, they should look mostly similar to the models found in identification.

To make sure the difference between them is not discernible, a mean square error is computed both for the identification model, "Mse = sum((Yflat-Yhat').^2)/length(Yflat)", and for the validation model, "MseVal = sum((YflatVal-YhatVal').^2)/length(YflatVal)", is computed. The optimal mean square error is computed for a series of numbers of centers and placed in vectors then plotting the results for identification, "plot(CdimFrom:1:CdimTo,MseV(CdimFrom:1:CdimTo))", and for validation, "plot(CdimFrom:1:CdimTo,MseValV(CdimFrom:1:CdimTo))".

4. INTERPRETATION OF THE RESULTS

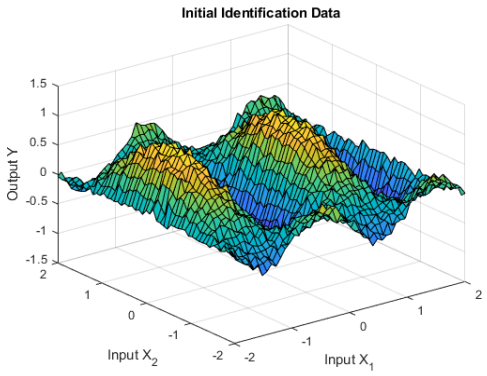

Figure 1.1

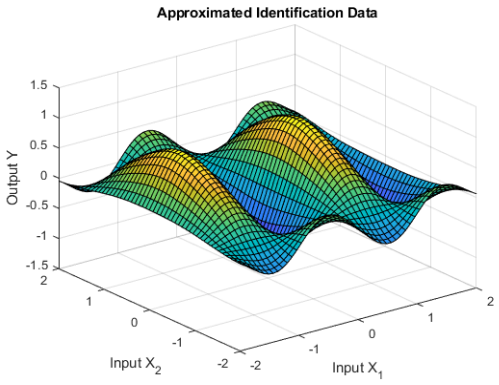

Figure 1.2

In Figures 1.1 and 1.2 the axes X and Y are represented by the identification coordinates. The identification data serves as a mean of computing the parameters which helps in realizing the reconstruction of the initial shape after the noise has been removed. The output value of the function which has the

coordinates as input parameters can be seen on the Z axis. The highest values of the function f are represented on the graphs in warm colors and the lowest ones in cold colors.

In Figure 2 a very close shape to the one in Figure 1 can be recognized. There are 2 areas with high values and 2 zones with low values. Also a sine wave shape is present on the X axis. By choosing the dimension of the centers grid as being 7, the interval is very well interspaced such that the overfitting phenomena due to a large number of center points doesn't appear. This matter can be observed from the lack of undefined shapes in the model.

From the computation of the mean squared error which is 0.0038, it can be proven that the approximation is lower than the upper threshold admitted for a good representation of the model.

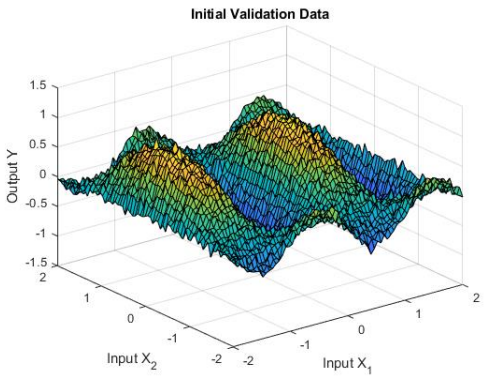

Figure 2.1

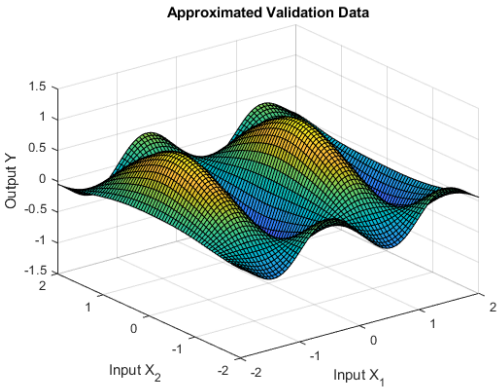

Figure 2.2

In Figures 2.1 and 2.2 we keep the same axes as in Figures 1.1 and 1.2. The validation process is used to check the coherence of the identification data model for a different set of input data, making the model as accurate as possible. This makes the algorithm suitable for any shapes resulting from other sets of data.

Having the Mean Squared Error for the validation set at the value of 0.0033, it results that the model manages to be precise enough such that the noise is not modeled. The low number of centers leads to the avoidance of overfitting. In the same time, the algorithm requires a smaller memory capacity and a lower computing power, making it more efficient than for a high number of centers.

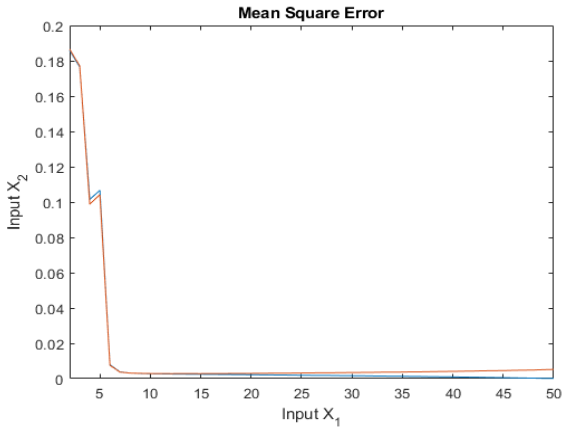

Figure 3

In Figure 3, the horizontal axis represents the range of the matrix created with the centers. The vertical axis shows the error between the data with noise and the noise-free data. There is a representation for identification and validation for different number of centers. It can be seen that between 2 and 7 there is a small variation between the error at identification and validation, but the value of the error is very high. This causes the problem of underfitting of the data. Between 7 and 25 both the errors and the difference between them are very low. For a value higher than 25 we can see that the difference between the validation model error and the real model error begins to increase, creating the effect of overfitting.

5. FINAL CONCLUSION

The correctness of modeling the input data highly depends on the choice of the grid for the centers of the RBFs. By creating a 7 x 7 centers grid the final values will be best distributed estimating the form the function f has. That is the value that best fits on the coordinates that are being supplied. This means the Mean Squared Error value should be very small, as in this project MSE being 0.0038. After the identification and validation have been applied the models which have been obtained in both stages of the process must be approximately similar.

If this theoretical matter doesn't also apply in practice, it means some things may have gone wrong or the programmer handling these cases hasn't paid too much attention to the specifications or to the evolution of the program (algorithm) as a whole.

In some cases, choosing a higher number of centers may help up to a point if the input point values are high, but predominantly modeling of noise will appear – overfitting ($N > 25$). Also choosing a small number of center points may lead to a similarly unwanted event happening, like underfitting ($N < 7$). Both cases lead to imprecision in the final result. The approximated and the real models should almost juxtapose one over the other in a meticulous and thorough approach of checking the consistency of the work done. If they don't, an extended revision should be taken into account in order to find the mistake that has gone unnoticed. This shows that the density of centers affects most the shape formed after the approximators and their corresponding parameters have been linked together. The recommended values to be chosen are $7 \leq N \leq 25$.

As a conclusion, the process of applying the method of linear regression on some input data in order to obtain the correct and final form of an unknown function f is a rather arduous and at times troublesome process. It is so because during the development of the algorithm the mathematical background plays the most important role. Having understood this side of the problem, the rest of it lays upon the programming skills one wields.

APPENDIX

Identification

```
% Closes and clears all the previously loaded data
clear all
close all

% Loads the identification and validation data
load('proj_fit_09.mat');

% Creates 2 vectors which hold the input coordinates, corresponding to the X
% and Y axes, the dimension of the input data and the output matrix of the
% non-linear function
Xone = id.X{1,1};
Xtwo = id.X{2,1};
Xdim = id.dims(1);
Ymatrix = id.Y;

% Plots the three-dimensional representation of the data
surf(Xone, Xtwo, Ymatrix);
title('Initial Identification Data'); xlabel('Input X_1');ylabel('Input X_2');zlabel('Output Y');

% We select the dimension of the matrix holding the center points
Cdim = 7;

% Creates a vector of zeros
% Selects the evenly spaced points forming the centers of the RBFs
Cone = zeros(1,Cdim);
Cone = linspace(Xone(1),Xone(end),Cdim);

% Computes the distance between 2 centers of the RBFs
b1 = abs(Cone(2)-Cone(1));
b2 = b1;

% Cflat creates a 2 by N^2 matrix having all the combinations of centers
% Xflat creates a 2 by n^2 matrix having all the combinations of input data
% Yflat creates a 1 by n^2 vector having all the combination of output data
Cflat = zeros(2,Cdim^2);
Xflat = zeros(2,Xdim^2);
Yflat = zeros(1,Xdim^2);

% Creates a matrix which on its first row has the first term of the centers by n times followed by the
next ones,
% also taken by n times. On the second row the terms are placed in their initial order, rewritten by
% n times. This way, we have access to all the combinations of the center points
for k = 1:1:Cdim
    for j = 1:1:Cdim
       Cflat(1,(k-1)*Cdim+j) = Cone(k);
       Cflat(2,(k-1)*Cdim+j) = Cone(j);
    end
end
```

```
% Creates a matrix which on its first row has the first term of the input coordinates by n times
followed by the next ones,
% also taken by n times. On the second row the terms are placed in their initial order, rewritten by
% n times. This way, we have access to all the combinations of the input data points
for k = 1:1:Xdim
    for j = 1:1:Xdim
        Xflat(1,(k-1)*Xdim+j) = Xone(k);
        Xflat(2,(k-1)*Xdim+j) = Xtwo(j);
    end
end

% Creates a vector which contains all the output data from the initial marix, concatenating all the rows
% of the initial matrix one after the other, making it easier to access the outputs
for k = 1:1:Xdim
    for j = 1:1:Xdim
       Yflat((k-1)*Xdim+j) = Ymatrix(k,j);
    end
end

% Creates a n^2 x N^2 matrix having all the RBFs corresponding to each center on the grid.
% Taking the matrix as having element which have the following notations Phi_k(x_i)
% Two counters are taken, the first one going from 1 to n^2, being the index of the rows representing
the RBFs(regressors)
% and a second one which is going from 1 to N^2 being the index of the
% colums representing the position of the elements from the data coordinates
Phi = zeros(Xdim^2,Cdim^2);
for i = 1:1:Xdim^2
    for k = 1:1:Cdim^2
        Phi(i,k) = exp(-((Cflat(1,k)-Xflat(1,i))/b1)^2-((Cflat(2,k)-Xflat(2,i))/b2)^2);
    end
end

% A parameter vector with an unknown value is computed by applying linear
% regression methods with the RBF matrix and the transposed matrix of the
% output. We also compute the approximated model, Yhat, of the output checking the
% correctness of the previously applied linear regression methods by
% multiplying the regressors with the parameters
Theta = Phi\Yflat';
Yhat = Phi*Theta;

% Because we transformed our matrices in vectors previously, our output
% approximation is also a vector, but in order to display the values it
% must be converted back to a matrix form
YhatMatrix = zeros(Xdim,Xdim);
for i = 1:1:Xdim
   for j = 1:1:Xdim
       YhatMatrix(i,j) = Yhat((i-1)*Xdim+j);
   end
end

% Plots the approximation of the identification model
figure
surf(Xone,Xtwo, YhatMatrix);
```

```
title('Approximated Identification Data'); xlabel('Input X_1');ylabel('Input X_2');zlabel('Output Y');

% Calculates the difference between the initial values of the function and
% our approximated values to see if the approximated model resembles the
% initial one, being aware of the difference caused by noise
Mse = sum((Yflat-Yhat').^2)/length(Yflat);
```

Validation

```
% We follow the same procedure as in the identification part in obtaining
% the regressor matrix
XoneVal = val.X{1,1};
XtwoVal = val.X{2,1};
XdimVal = val.dims(1);
YmatrixVal = val.Y;

figure
surf(XoneVal, XtwoVal, YmatrixVal);
title('Initial Validation Data'); xlabel('Input X_1');ylabel('Input X_2');zlabel('Output Y');


% We keep the same dimension for the matrix of centers
CdimVal = Cdim;

ConeVal = zeros(1,CdimVal);
ConeVal = linspace(XoneVal(1),XoneVal(end),CdimVal);

b1Val = abs(ConeVal(2)-ConeVal(1));
b2Val = b1Val;

CflatVal = zeros(2,CdimVal^2);
XflatVal = zeros(2,XdimVal^2);
YflatVal = zeros(1,XdimVal^2);

for k = 1:1:CdimVal
    for j = 1:1:CdimVal
       CflatVal(1,(k-1)*CdimVal+j) = ConeVal(k);
       CflatVal(2,(k-1)*CdimVal+j) = ConeVal(j);
    end
end

for k = 1:1:XdimVal
    for j = 1:1:XdimVal
        XflatVal(1,(k-1)*XdimVal+j) = XoneVal(k);
        XflatVal(2,(k-1)*XdimVal+j) = XtwoVal(j);
    end
end
```

```
for k = 1:1:XdimVal
    for j = 1:1:XdimVal
       YflatVal((k-1)*XdimVal+j) = YmatrixVal(k,j);
    end
end

PhiVal = zeros(XdimVal^2,CdimVal^2);
for i = 1:1:XdimVal^2
    for k = 1:1:CdimVal^2
        PhiVal(i,k) = exp(-((CflatVal(1,k)-XflatVal(1,i))/b1)^2-((CflatVal(2,k)-XflatVal(2,i))/b2)^2);
    end
end

% Using the Theta from identification part and the regressor for the new
% set of data, we calculate the approximation of the validation output data
YhatVal = PhiVal*Theta;

YhatMatrixVal = zeros(XdimVal,XdimVal);
for i = 1:1:XdimVal
   for j = 1:1:XdimVal
       YhatMatrixVal(i,j) = YhatVal((i-1)*XdimVal+j);
   end
end

figure
surf(XoneVal,XtwoVal, YhatMatrixVal);
title('Approximated Validation Data'); xlabel('Input X_1');ylabel('Input X_2');zlabel('Output Y');

% Calculates the difference between the initial validation data of the function and
% our approximated values using the computed values in identification for theta
MseVal = sum((YflatVal-YhatVal').^2)/length(YflatVal);
```

Optimal Mean Square error

```
% CdimFrom is the minimal dimension of the centers matrix from which we compute the optimal MSE
% CdimTo is the value of the maximal dimension of the centers matrix
% MseV is the vector in which we place all the values of the identification error
% MseValV is the vector in which we place all the values of the validation error

CdimFrom=2;
CdimTo=50;
MseV=zeros(1,(CdimTo-CdimFrom));
MseValV=zeros(1,(CdimTo-CdimFrom));

% We use the previously presented procedure for identification and
% validation, checking the optimal MSE for a series of different number of
% centers
for CdimOp=CdimFrom:1:CdimTo
```

```
Xone = id.X{1,1};
Xtwo = id.X{2,1};
Xdim = id.dims(1);
Ymatrix = id.Y;

Cdim = CdimOp;
Cone = zeros(1,Cdim);
Cone = linspace(Xone(1),Xone(end),Cdim);

b1 = abs(Cone(2)-Cone(1));
b2 = b1;

Cflat = zeros(2,Cdim^2);
Xflat = zeros(2,Xdim^2);
Yflat = zeros(1,Xdim^2);

for k = 1:1:Cdim
    for j = 1:1:Cdim
        Cflat(1,(k-1)*Cdim+j) = Cone(k);
        Cflat(2,(k-1)*Cdim+j) = Cone(j);
    end
end

for k = 1:1:Xdim
    for j = 1:1:Xdim
        Xflat(1,(k-1)*Xdim+j) = Xone(k);
        Xflat(2,(k-1)*Xdim+j) = Xtwo(j);
    end
end

for k = 1:1:Xdim
    for j = 1:1:Xdim
        Yflat((k-1)*Xdim+j) = Ymatrix(k,j);
    end
end

Phi = zeros(Xdim^2,Cdim^2);

for i = 1:1:Xdim^2
    for k = 1:1:Cdim^2
        Phi(i,k) = exp(-((Cflat(1,k)-Xflat(1,i))/b1)^2-((Cflat(2,k)-Xflat(2,i))/b2)^2);
    end
end

Theta = Phi\Yflat';
Yhat = Phi*Theta;

YhatMatrix = zeros(Xdim,Xdim);
for i = 1:1:Xdim
    for j = 1:1:Xdim
        YhatMatrix(i,j) = Yhat((i-1)*Xdim+j);
    end
```

```
        end

        Mse = sum((Yflat-Yhat').^2)/length(Yflat);
        MseV(Cdim) = Mse;



        XoneVal = val.X{1,1};
        XtwoVal = val.X{2,1};
        XdimVal = val.dims(1);
        YmatrixVal = val.Y;

        CdimVal = CdimOp;

        ConeVal = zeros(1,CdimVal);
        ConeVal = linspace(XoneVal(1),XoneVal(end),CdimVal);

        b1Val = abs(ConeVal(2)-ConeVal(1));
        b2Val = b1Val;

        CflatVal = zeros(2,CdimVal^2);
        XflatVal = zeros(2,XdimVal^2);
        YflatVal = zeros(1,XdimVal^2);

        for k = 1:1:CdimVal
            for j = 1:1:CdimVal
                CflatVal(1,(k-1)*CdimVal+j) = ConeVal(k);
                CflatVal(2,(k-1)*CdimVal+j) = ConeVal(j);
            end
        end

        for k = 1:1:XdimVal
            for j = 1:1:XdimVal
                XflatVal(1,(k-1)*XdimVal+j) = XoneVal(k);
                XflatVal(2,(k-1)*XdimVal+j) = XtwoVal(j);
            end
        end

        for k = 1:1:XdimVal
            for j = 1:1:XdimVal
                YflatVal((k-1)*XdimVal+j) = YmatrixVal(k,j);
            end
        end

        PhiVal = zeros(XdimVal^2,CdimVal^2);

        for i = 1:1:XdimVal^2
            for k = 1:1:CdimVal^2
                PhiVal(i,k) = exp(-((CflatVal(1,k)-XflatVal(1,i))/b1)^2-((CflatVal(2,k)-
XflatVal(2,i))/b2)^2);
            end
        end
```

```
        YhatVal = PhiVal*Theta;

        YhatMatrixVal = zeros(XdimVal,XdimVal);

        for i = 1:1:XdimVal
            for j = 1:1:XdimVal
                 YhatMatrixVal(i,j) = YhatVal((i-1)*XdimVal+j);
            end
        end

        MseVal = sum((YflatVal-YhatVal').^2)/length(YflatVal);
        MseValV(CdimOp) = MseVal;

end

% Plots the MSE for the series of centers chosen at the previous step
% for both identification and validation
figure
plot(CdimFrom:1:CdimTo,MseV(CdimFrom:1:CdimTo));
hold on
plot(CdimFrom:1:CdimTo,MseValV(CdimFrom:1:CdimTo));
title('Mean Square Error'); xlabel('Dimension of the Center Matrix'); ylabel('Value of the Error')
axis([CdimFrom CdimTo 0 0.2]);
```