**REPORT ON**

**SUPPORT VECTOR REGRESSION**

# ABSTRACT

Data mining is widely used in diverse areas like Financial Data Analysis, artificial intelligence, machine learning, statistics, and database systems .One of the important application of data mining is predicting/imputing missing values as vast volume of data is collected nowadays. There are number of techniques available for imputing missing values like K-Nearest Neighbor, K-means Clustering Imputation, Fuzzy K-means Clustering (FKMI), SVM (Support vector Machine). In our project we will be focusing on Support vector Regression which belongs to class Support vector machine.

# TOOLS USED:

* Matlab 2009a
* Visual studio 2008 express compiler.
* LIBSVM (Library for Support vector machine).

# INTRODUCTION

Nowadays vast amount of data is collected due to increase in technology and ease of availability. There can be some missing values in this collected data due to various reasons like faulty equipment, incorrect measurements, and nonresponse in survey, misunderstanding, and anonymous data. These missing values are important as it leads to loss of efficiency and complications in analyzing data by DM techniques. Support vector machine belongs to machine learning methods to handle missing data.

# SUPPORT VECTOR REGRESSION

Support Vector machine (SVM), which is based on the structural risk minimization principle in statistical learning theory, is a powerful tool for general purpose machine learning problem In practice, two kinds of SVMs are provided for different purpose: Support Vector machine for classification (SVC) and Support Vector machine for regression (SVR). SVR is carried out with two steps: first, the SVR maps the samples from the input space with a low dimension into a much higher (sometimes infinite) dimensional space with a kernel function, and then searches for the global optimal solution to the corresponding problem using the quadratic programming. The so called support vectors (Figure 9) are these samples with non-zero Lagrange multiplier. Given a set of observed training data (circles and triangles), which are sampled from the hidden original function f(x) (solid line) and maybe polluted by noise during this procedure, SVR constructs the fitted regression function φ(x)(dashed line) by solving the corresponding optimal problem with constraints.

Figure 1:Illustration of the support vector machine method used for regression

# WORKING

In order to implement Support vector regression we have used **LIBSVM** library .LIBSVM is integrated software for support vector regression .Two main methods available in libsvm are svm\_train and predict.

1. **Svmtrain:** returns a model which can be used for future prediction.  It is a structure and is organized as [Parameters, nr\_class, totalSV, rho, Label, ProbA, ProbB, nSV, sv\_coef, SVs].

Parameters passed to this function include but not limited to:

-s svm\_type: set type of SVM (default 0)

0 -- C-SVC

1 -- nu-SVC

2 -- one-class SVM

**3 -- epsilon-SVR**

4 -- nu-SVR

-t kernel type: set type of kernel function (default 2)

0 -- linear: u'\*v

1 -- Polynomial: (gamma\*u'\*v + coef0)^degree

**2 -- Radial basis function: exp(-gamma\*|u-v|^2)**

3 -- Sigmoid: tanh(gamma\*u'\*v + coef0)

-g gamma: set gamma in kernel function (default 1/num\_features)

-e epsilon: set tolerance of termination criterion (default 0.001)

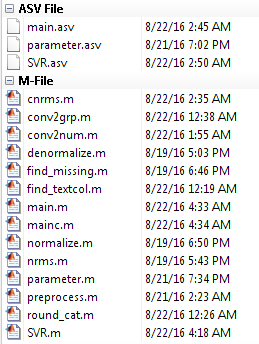
-b probability estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)

-d degree: set degree in kernel function (default 3).

1. **Svmpredict** : This function has 3 outputs

The first one, predictd\_label, is a vector of predicted labels. The second output, accuracy, is a vector including squared correlation coefficient (for regression). The third is a matrix containing decision values or probability estimates (if ‘-b 1’ is specified).

## MATLAB FILES



Main.m – File that take i/p (in our case missing file) and invoke all other functions like preprocess, normalize, svr, denormalize, nrms and at the end displays nrms value.

Preprocess.m – File that take as input incomplete sheet and based on Nan values it separate missing rows from no missing rows and returns training and test set.

Normalize.m: This file takes i/p training and test set and normalizes data Column wise and returns normalized training and test set.

SVR.m – This file take i/p training and test set and pass it to svmtrain and svmpredict method to impute missing value using model trained by training set.

Denormalize: This takes i/p imputed values from SVR.m and denormalize data column wise in order to get effective results.

NRMS.m – This file take i/p imputed and original matrix to calculate NRMSE error.

Cnrms.m – This file is used when we deal with categorical and numerical data hence we need both nrms and aes.

Conv2num – This file is used to convert categorical data to numerical.

Conv2grp – This file is used to convert numerical data back to categorical.

Find\_textcol – This file is used to find text/categorical column in dataset.

Parameter.m - This file is used to provide best c and g values which are supplied to SVR function.

Round\_cat.m – This file is used to round the predicted values for categorical column as they are in type double.

# ALGORITHM

Given training vectorx_i \in \mathbb{R}^p, i=1,..., n, and a vector y \in \mathbb{R}^n \varepsilon-SVR solves the following primal problem:

\min_ {w, b, \zeta, \zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)



\textrm {subject to } & y_i - w^T \phi (x_i) - b \leq \varepsilon + \zeta_i,\\
                      & w^T \phi (x_i) + b - y_i \leq \varepsilon + \zeta_i^*,\\
                      & \zeta_i, \zeta_i^* \geq 0, i=1, ..., n

Its dual is

\min_{\alpha, \alpha^*} \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*)


\textrm {subject to } & e^T (\alpha - \alpha^*) = 0\\
& 0 \leq \alpha_i, \alpha_i^* \leq C, i=1, ..., n

Where e is the vector of all ones, C > 0 is the upper bound, Q is an n by n positive semi-definite matrix, Q_{ij} \equiv K(x_i, x_j) = \phi (x_i)^T \phi (x_j) is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function \phi.

The decision function is:

\sum_{i=1}^n (\alpha_i - \alpha_i^*) K(x_i, x) + \rho

These parameters can be accessed through the members dual\_coef\_ which holds the difference \alpha_i - \alpha_i^*,support\_vectors\_ which holds the support vectors, and intercept\_ which holds the independent term \rho

# DEVELOPMENT PROCEDURE

Step 1**:** In order to impute missing values we need to first preprocess our data so that it can predict more accurate results.First we need to define our Training and Test set from given dataset.

Let us consider below dataset:

|  |  |  |  |
| --- | --- | --- | --- |
| 3.4 | 4.6 | 1.4 | 0.3 |
| 3.4 | 5 | 1.5 | 0.2 |
| 2.9 | 4.4 | 1.4 | 0.2 |
| 3.1 | 4.9 | 1.5 | 0.1 |
| 3.7 | 5.4 | 1.5 | 0.2 |
| 3.4 | 4.8 | 1.6 | 0.2 |
| 3 | 4.8 | 1.4 | 0.1 |
| 3 | 4.3 | 1.1 | 0.1 |
| 4 | 5.8 | 1.2 | 0.2 |
|  | 5.7 | 1.5 |  |
| 3.9 | 5.4 | 1.3 | 0.4 |
| 3.5 | 5.1 | 1.4 | 0.3 |
| 3.8 | 5.7 | 1.7 | 0.3 |

Table 1Incomplete dataset

## Step 2:

## IMPUTING MISSING VALUE (10,1)

Before generating training and test set we need to find missing rows and column as SVR takes only 1 column as target at a time we cannot have missing values in multiple column. Since in our given dataset we have missing values in 2 and 4 column we can replace value in 4 column with its mean (particular column) as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.7 | 1.5 | 0.2 |

**Replacing missing value at column 4 with it mean value**

From this new dataset we need to generate our Training and Test set. When this dataset is given as i/p to preprocess it generates below Training and test set as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| 3.4 | 4.6 | 1.4 | 0.3 |
| 3.4 | 5 | 1.5 | 0.2 |
| 2.9 | 4.4 | 1.4 | 0.2 |
| 3.1 | 4.9 | 1.5 | 0.1 |
| 3.7 | 5.4 | 1.5 | 0.2 |
| 3.4 | 4.8 | 1.6 | 0.2 |
| 3 | 4.8 | 1.4 | 0.1 |
| 3 | 4.3 | 1.1 | 0.1 |
| 4 | 5.8 | 1.2 | 0.2 |
| 3.9 | 5.4 | 1.3 | 0.4 |
| 3.5 | 5.1 | 1.4 | 0.3 |
| 3.8 | 5.7 | 1.7 | 0.3 |

Table 2:Training set

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.7 | 1.5 | 0.2 |

Table 3:Test set

## Step 3:

After generating training and test set we need we need to normalize this data before passing it to SVR function.

Normalize.m function is used to normalize training and test data column wise as follows:

function [outtr ,outte] = normalize (tr, te, tempm)

[row,col] = size(tempm);

ma = max(tempm,[],1);

mi = min(tempm,[],1);

outtr=[];

outte=[];

for i = 1:col

maxt = ma(i);

mint = mi(i);

difft = maxt - mint;

trainn = (tr(:,i)- mint)/difft;

testn = (te(:,i) - mint)/difft;

outtr =[outtr trainn];

outte =[outte testn];

end

a = size(outte)

nr = a(1);

nc = a(2);

for i = 1: nr

for j = 1:nc

if (outte(i,j) < 0)

outte(i,j)= 0;

end

end

end

xlswrite('comp.xlsx', outtr);

xlswrite('miss.xlsx', outte);

end

After performing these steps the dataset is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| 3.4 | 4.6 | 1.4 | 0.3 |
| 3.4 | 5 | 1.5 | 0.2 |
| 2.9 | 4.4 | 1.4 | 0.2 |
| 3.1 | 4.9 | 1.5 | 0.1 |
| 3.7 | 5.4 | 1.5 | 0.2 |
| 3.4 | 4.8 | 1.6 | 0.2 |
| 3 | 4.8 | 1.4 | 0.1 |
| 3 | 4.3 | 1.1 | 0.1 |
| 4 | 5.8 | 1.2 | 0.2 |
| 3.9 | 5.4 | 1.3 | 0.4 |
| 3.5 | 5.1 | 1.4 | 0.3 |
| 3.8 | 5.7 | 1.7 | 0.3 |

Table 4: Normalized Training set

**Normalized Training set**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5.7 | 1.5 | 0.2 |

Table 5: Normalized Test set

## Step 4:

Once data is normalized, both training and test set is passed to svr.m file which include 2 function svmtrain and svmpredict. Svmtrain functions returns model which is then applied to test set to predict missing values. The output of svr.m is imputed value for location 10, 1 as follows:

Value (10, 1): **0.7957**

**SVR.m**

function [predict] = SVR(trainn,testn,mc)

% Irisc = xlsread('comp.xlsx');

umc = unique(mc);

if(umc == 1)

labels = trainn(:,1); % labels from the 1st column

features = trainn(:, 2:end);

l = testn(:,1); % labels from the 1st column

f = testn(:, 2:end);

elseif(umc == 2)

labels = trainn(:,2); % labels from the 1st column

features = trainn(:,[1 3:end]);

l = testn(:,2); % labels from the 1st column

f = testn(:,[1 3:end]);

elseif(umc == 3)

labels = trainn(:,3); % labels from the 1st column

features = trainn(:,[1 2 end]);

l = testn(:,3); % labels from the 1st column

f = testn(:,[1 2 end]);

elseif(umc == 4)

labels = trainn(:,4); % labels from the 1st column

features = trainn(:,[1 2 3]);

l = testn(:,4); % labels from the 1st column

f = testn(:,[1 2 3]);

end

model = svmtrain(labels,features,'-s 3 -g 1 -t 2 -b 1');

[predict] = svmpredict(l,f,model,'-b 1');

end

## Step 5:

Once the data is imputed it is passed to denormalize function which denormalizes imputed value column wise as follows:

function [dimpute] = denormalize(impute,tempm,mc)

umc = unique(mc);

[r c] = size(mc);

dimpute = [];

[ri ci] = size(impute);

ma = max(tempm,[],1);

mi = min(tempm,[],1);

for i = 1:ri

maxt = ma(umc);

mint = mi(umc);

difft = maxt - mint;

dimpute(i) = (impute(i)\* difft)+ mint;

end

disp(dimpute)

end

## Step 6:

After demoralizing data, nrms is calculated as follows:

function [nrms] = nrms(imputed ,original)

t = sqrt(sum(sum(imputed-original).^2,2));

t1 = sqrt(sum(sum(original.^2),2));

nrms = t/t1;

end

**Value (10, 1): 3.7753**

**RESULT**

|  |  |  |  |
| --- | --- | --- | --- |
| 3.4 | 4.6 | 1.4 | 0.3 |
| 3.4 | 5 | 1.5 | 0.2 |
| 2.9 | 4.4 | 1.4 | 0.2 |
| 3.1 | 4.9 | 1.5 | 0.1 |
| 3.7 | 5.4 | 1.5 | 0.2 |
| 3.4 | 4.8 | 1.6 | 0.2 |
| 3 | 4.8 | 1.4 | 0.1 |
| 3 | 4.3 | 1.1 | 0.1 |
| 4 | 5.8 | 1.2 | 0.2 |
| 3.77 | 5.7 | 1.5 |  |
| 3.9 | 5.4 | 1.3 | 0.4 |
| 3.5 | 5.1 | 1.4 | 0.3 |
| 3.8 | 5.7 | 1.7 | 0.3 |

Table 6: Final output

## 

## Final output

In order to impute value (10,4) we need to repeat step1 to step 6 which is done using loop in main.m file and we get following values:

Imputed value: 0.54

Denormalized value: 0.263

## NRMSE - 0.0281

## UPDATED FINAL OUTPUT

To get the final output we will repeat from step 1 to step 5 using for loop in main.m file.

|  |  |  |  |
| --- | --- | --- | --- |
| 3.4 | 4.6 | 1.4 | 0.3 |
| 3.4 | 5 | 1.5 | 0.2 |
| 2.9 | 4.4 | 1.4 | 0.2 |
| 3.1 | 4.9 | 1.5 | 0.1 |
| 3.7 | 5.4 | 1.5 | 0.2 |
| 3.4 | 4.8 | 1.6 | 0.2 |
| 3 | 4.8 | 1.4 | 0.1 |
| 3 | 4.3 | 1.1 | 0.1 |
| 4 | 5.8 | 1.2 | 0.2 |
| 3.77 | 5.7 | 1.5 | 0.26 |
| 3.9 | 5.4 | 1.3 | 0.4 |
| 3.5 | 5.1 | 1.4 | 0.3 |
| 3.8 | 5.7 | 1.7 | 0.3 |

Table 7: Updated Final output

## IMPUTING CATEGORICAL DATA

To impute categorical data we perform following steps:

**Step 1:** The very first step in imputing categorical data is to find columns that contain categorical data. This is done using function created called “find\_textcol” which make use of cell function ischar to find character in dataset.

**Step 2:** Since we know the columns with categorical data we need to convert this column in numerical values as SVR only impute numerical values. For this we use function grp2idx.

**Step 3:** After converting into numerical values we will find missing values using function find\_missing.m.

**Step 4:** Then we will call SVR function by passing incomplete matrix and target column.

**Step 5:** SVR function will first call preprocess function in order to find training and test set

**Step 6:** SVR then calls normalize function by passing training and test set received from preprocess function

**Step 7:** After normalizing data it then calls SVR train and SVR predict method to impute missing values

**Step 8**: The imputed values are then rounded to nearest number using round function as they are of type double.

**Step 9**: Imputed numerical values are converted back to categorical using grp2idx fuction.

**Step 10**: The output matrix is then used to calculate NRMSE error for numerical column and AES error for categorical column using cnrms function created as follows:

**AE**

To impute AE of the categorical data we use the following function cnrms:

function [nrms,ae] = cnrms( out,original,tcol)

outputn=[];

for i=1:length(tcol)

if(tcol(i))

out\_text(:,i) = out(:,i);

original\_text(:,i) = original(:,i);

else

outputn(:,i) = out(:,i);

original\_num(:,i) = original(:,i);

end

end

if(length(outputn) > 0)

m = sqrt(sum(sum(outputn-original\_num).^2,2));

m1 = sqrt(sum(sum(original\_num.^2),2));

nrms = m/m1;

else

nrms = 0;

end

if(length(out\_text) > 0)

sum\_ae = 0;

[row col] = size(out\_text);

for i=1:row

for j=1:col

if(out\_text(i,j) == original\_text(i,j))

sum\_ae = sum\_ae + 1;

end

end

end

ae = sum\_ae/(row \* col);

else

ae = 0;

end

end

**Data Comparison**

Figure 2: Illustration of data comparison using scatter plot.

In figure 2, we took the first 10 elements of Blend1 of the dataset. On the right pane you can see 10 elements of the graph. The first 5 are from the original dataset and last 5 are from the imputed dataset. We plot it on x-y axis to obtain the trend-line. Trend-line shows the data deviation from its original path. We can see that there is minimal or no data deviations for the data elements presented in the graph as most of the data lines are overlapping each other.

In figure 3 the same data is represented using bar graph to illustrate results more clearly.

Figure 3: Illustration of data comparison using bar chart.

# CONCLUSION

In our project support vector regression is implemented using Matlab Programming language and LIBSVM library. SVR impute missing value using kernel function which is responsible for converting data into high dimension so that it can be separated easily or the margin width can be increased.

# REFERENCES

[1] “Missing value estimation for DNA microarray gene expression data by Support Vector Regression imputation and orthogonal coding scheme”, Xian Wang, Ao Li, Zhaohui Jiang and Huanqing Feng.

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[3] <http://www.alivelearn.net/?p=1083>.

[4] A tutorial on Support Vector Regression http://link.springer.com/article/10.1023/B:STCO.0000035301.49549.88