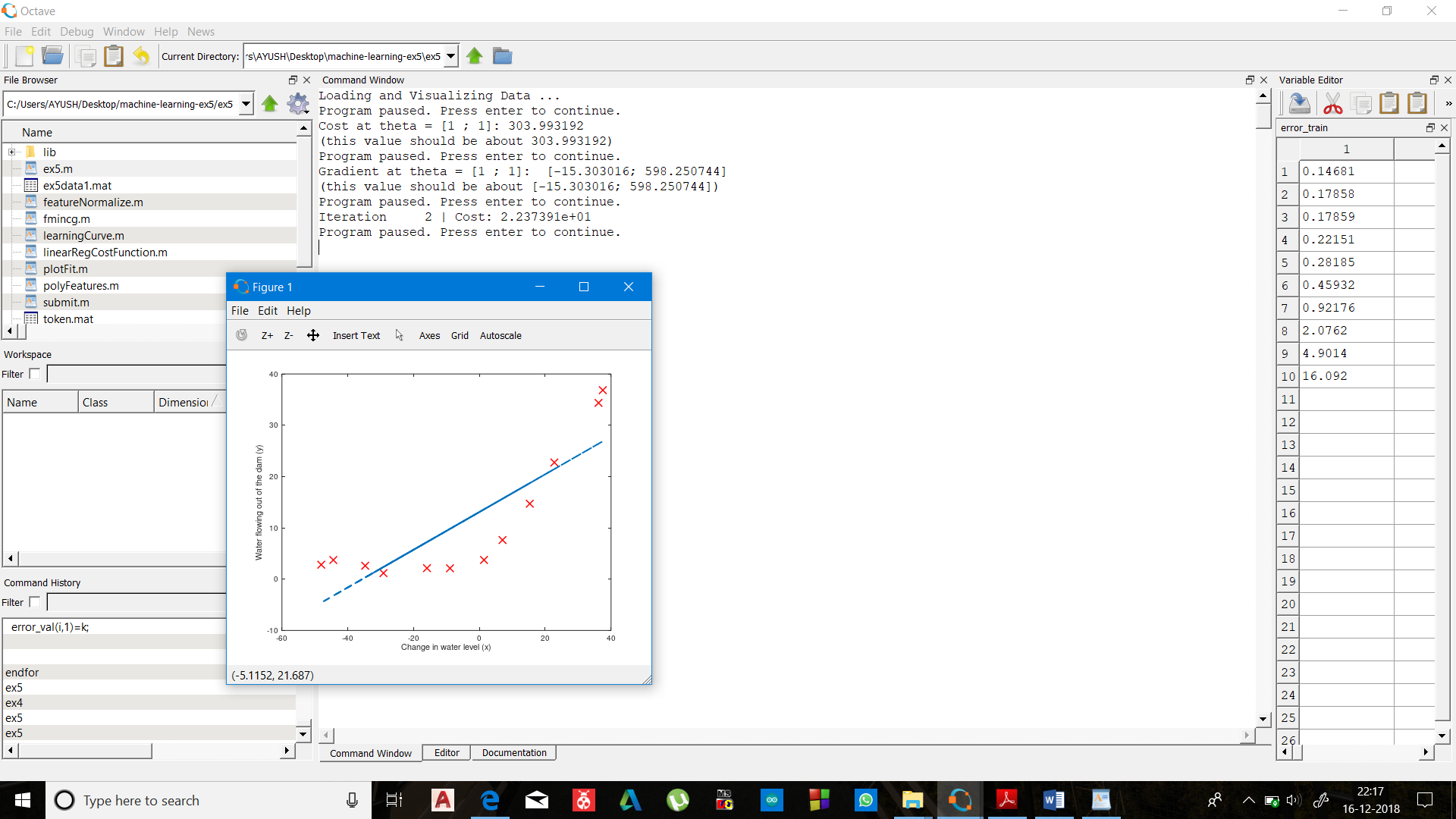
Topics covered this week :

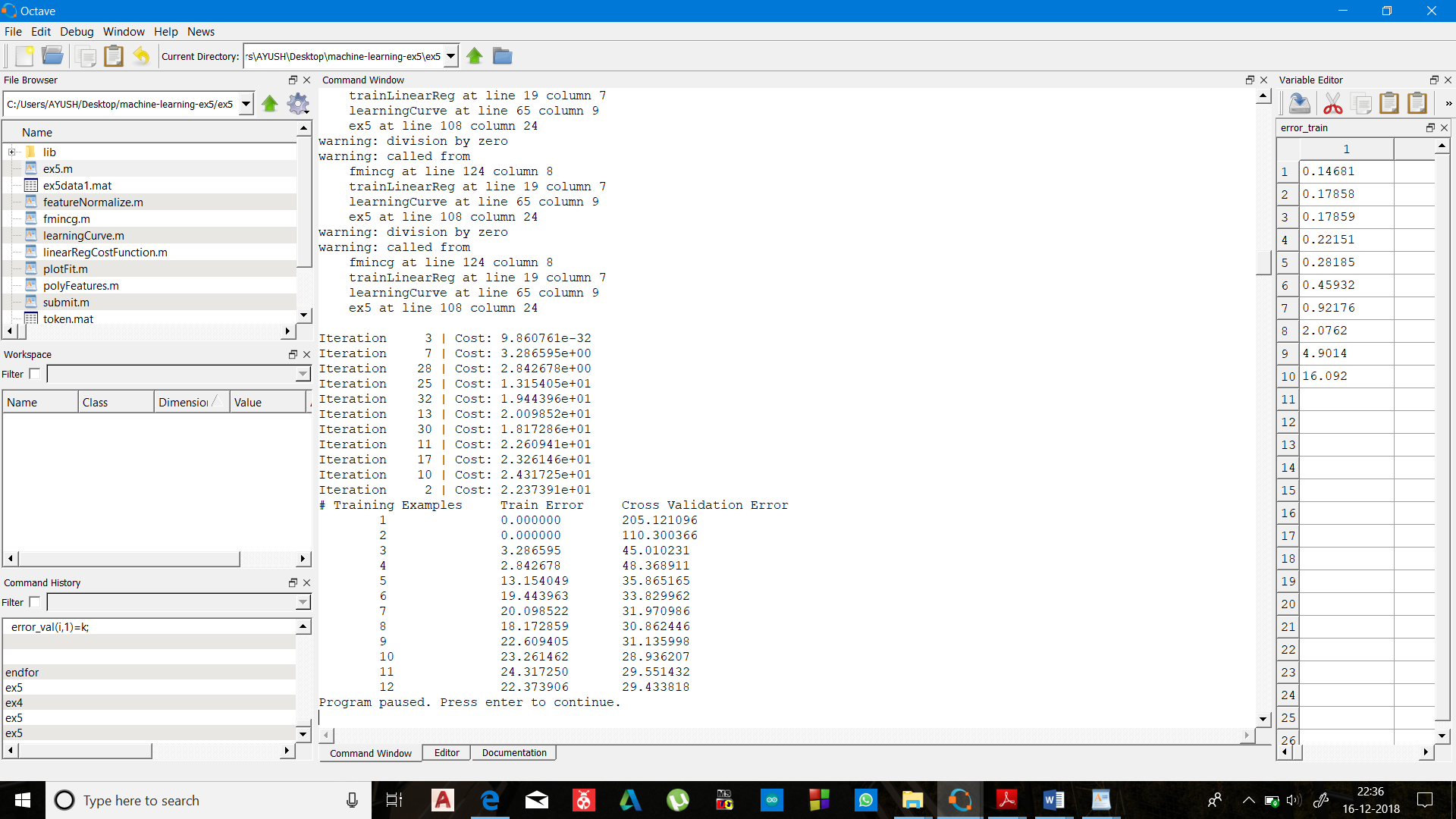
1. Linear regression with polynomial features.
2. Then regularization of the cost function to prevent high variance (overfitting the data)
3. Logistic regression including multiple features (for non linear hypothesis) and ones vs all to have multiple classifiers.
4. Neural networks (image detection ) //(still implementing)
5. Learning curves analyse our hypothesis .
6. SVM (theory).//(still implementing)
7. Linear regression with multiple features .

Since it is not always possible to have a straight line to be our hypothesis

to predict the ouput because it may not well fit the data set. Therefore one way to solve this problem is have multiple features to form a non- linear hypothesis .



1. Here my dataset was non linear but when I applied my linear regression model with hypothesis: Ɵ0 x0 + Ɵ1x1  it dosent fit the model well which can also be interpreted by costfunction



B)clearly cost is 22.37 which is unacceptable.

(training set has 12 total examples thus 12th row is cost calculated over all 12 examples)

Thus to fit our data set I created more no. of features from existing features .

If u have features suppose x1 and x2 the new features can be x1^2 , x2^2 , x1\*x2 etc

(this table is cost calculated overno. Of training

Examples in dataset)

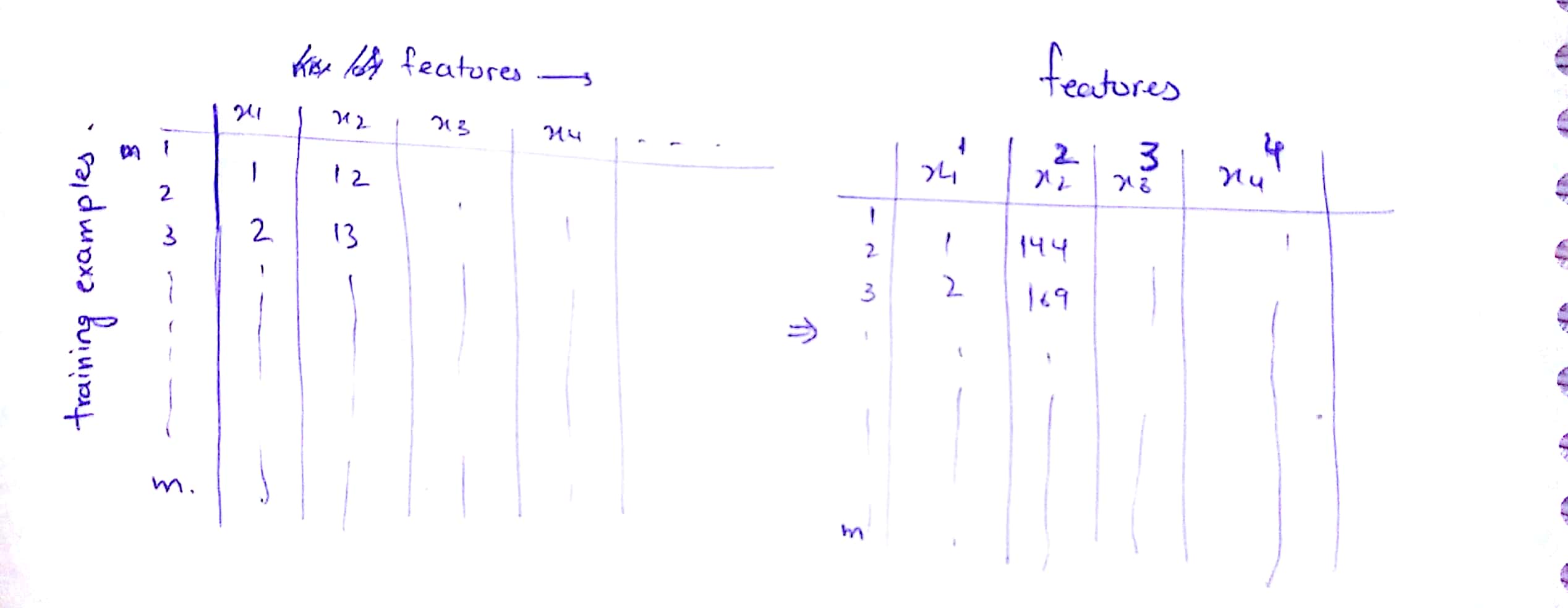
function [X\_poly] = polyFeatures(X, p)

X\_poly = zeros(numel(X), p);

for i=1:p;

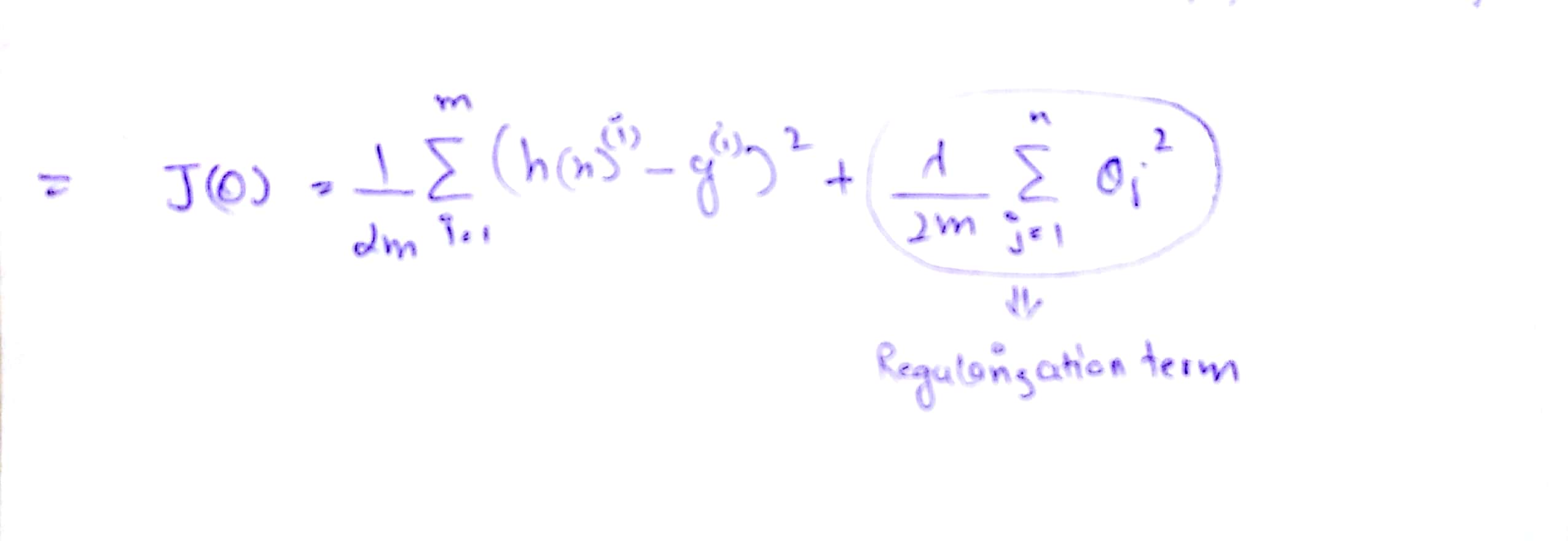
X\_poly (:,i)=X(:,1).^i;

Endfor



1. Overfitting and underfitting : overfitting also known as high variance is reflected in the hypothesis when our hypothesis overfits the training set and cost in that case can be very low or even equal to zero and thus fail to generalize our model , overfitting can be prevented by introducing regularization term in the cost function.

Whereas when our hypothesis badly fits the data it is said to be have high bias .

Regularized cost function for linear regression(to prevent overfitting) =

Particular note on lambda :

when lambda is very large then it heavily penalize the cost function therefore theta tend to be close to zero (hypothesis is tending towards linearty) when cost function is minimized which causes the hypothesis to reflect high bias problem

On the other hand when lambda is close to zero, after minimizing the cost function ,hypothesis tend to reflect problem of high variance.

Thus lambda must be chosen carefully.

//code of function of cost function implemented in octave.

function [J, grad] = linearRegCostFunction(X, y, theta, lambda)

m = length(y); % number of training examples

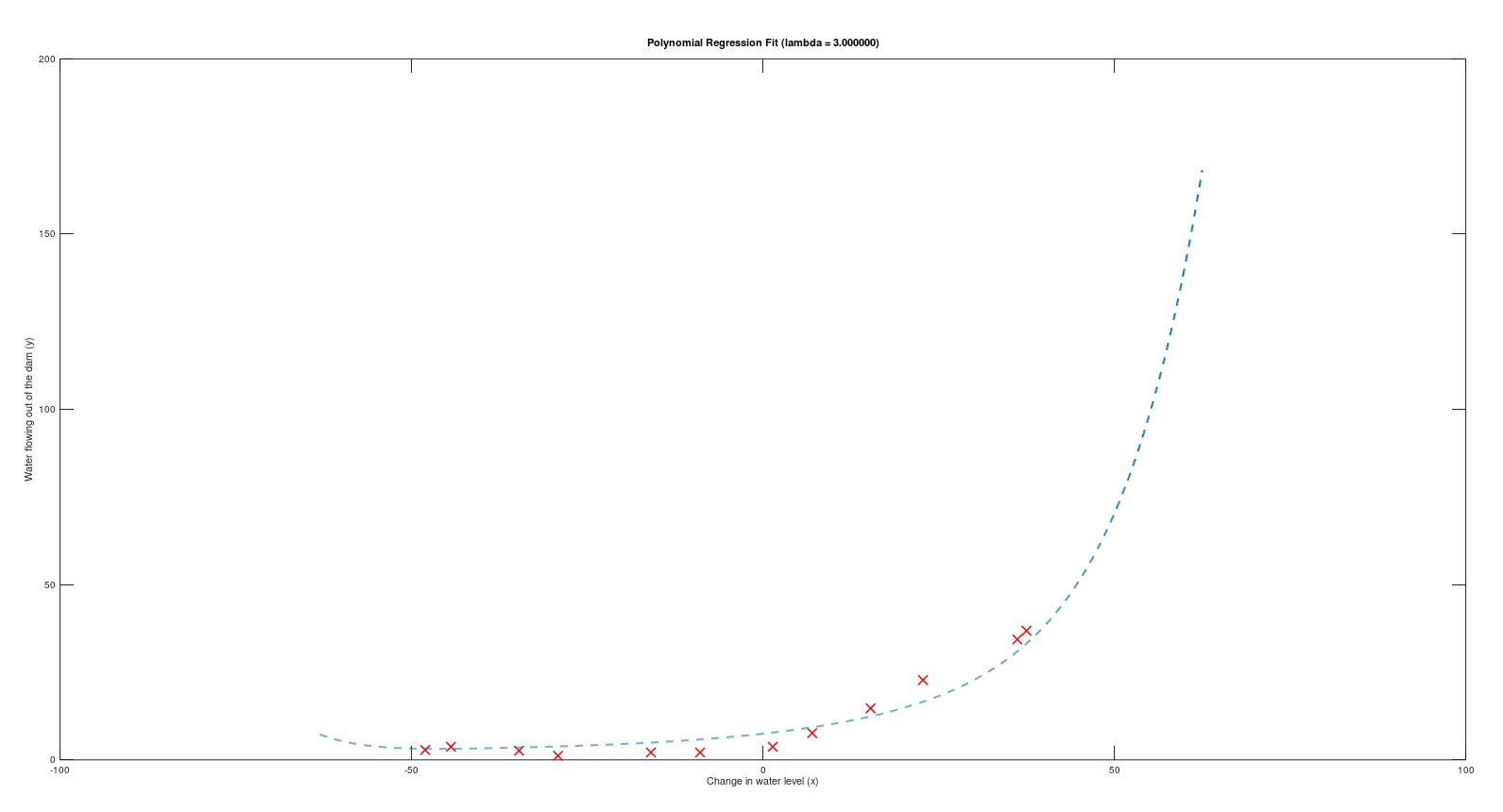
J = 0;

grad = zeros(size(theta));

J=(1/(2\*m))\*sum(((X\*theta-y).^2))+(sum((theta.^2))-theta(1,1).^2)\*(lambda/(2\*m)); // cost function

grad=(1/m)\*(((X\*theta-y)')\*X)'+((lambda/m).\*theta);

grad(1,1)=grad(1,1)-((lambda/m).\*theta(1,1));



Dataset and final hypothesis plotted together.

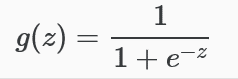
1. Logistic regression

Logistic regression helps us in classification problem i.e its ouput gives the probability of the class the given input belongs.

Hypothesis of the input looks like : 0≤hθ(x)≤1

H(x)=sigmoid(Ɵ0 + Ɵ1x1  + Ɵ2x2 + Ɵ3x3……..)

sigmoid function :



//code implemented in octave

function g = sigmoid(z)

r=size(z,1); //no. of rows in z vector

c=size(z,2); // no. of columns in vector z

for i=1:r;

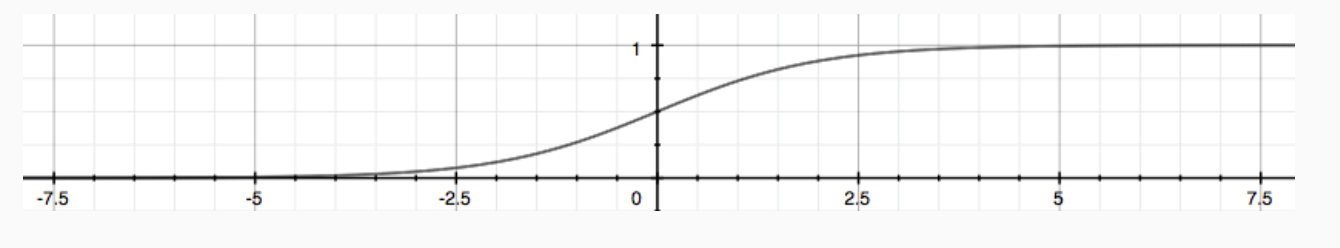
for k=1:c;

g(i,k)=1/(1+e^(-z(i,k)));

end;

end;

Graphically :



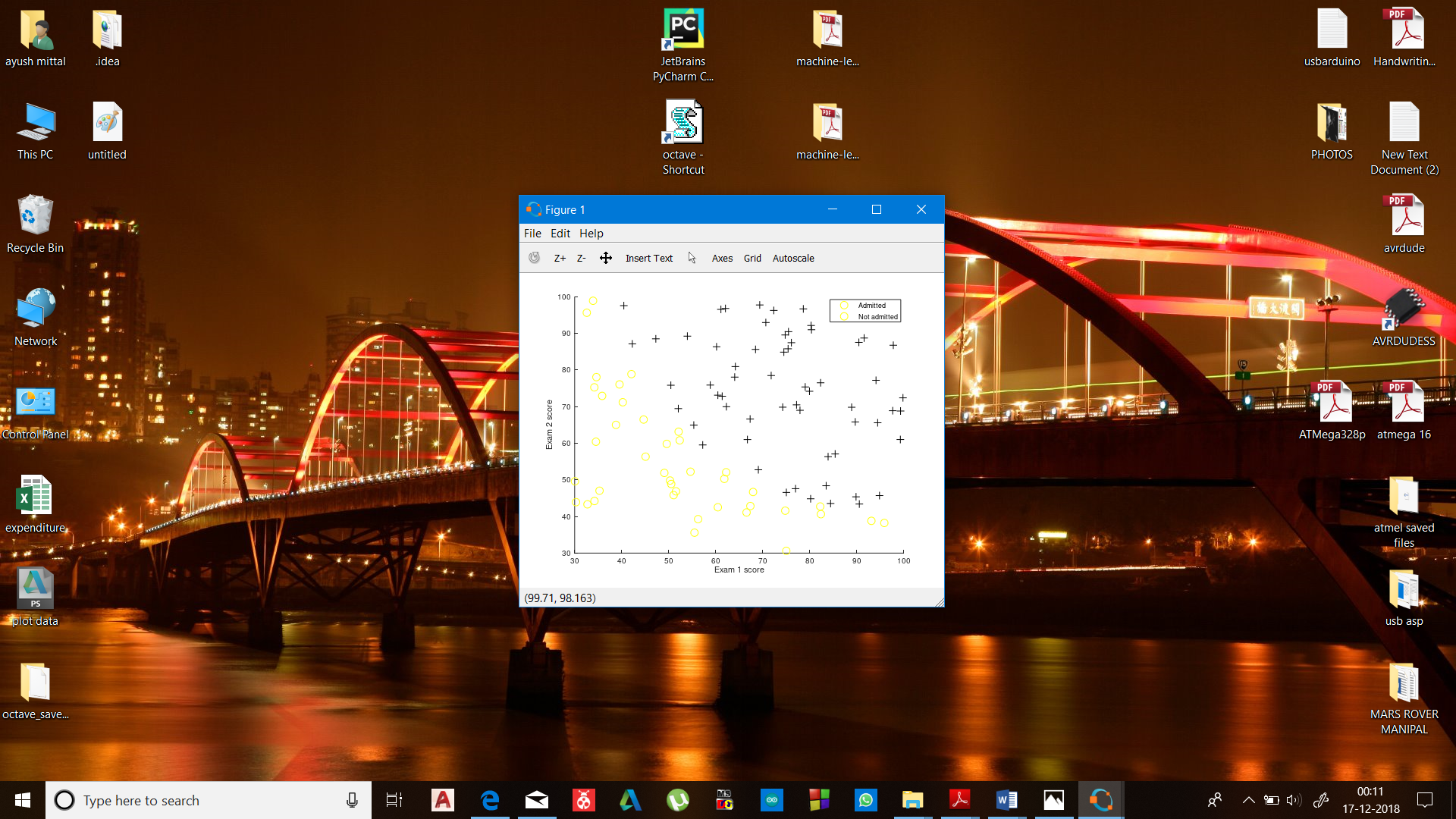
i.e

h(x)>=0.5 when Ɵ0 + Ɵ1x1  + Ɵ2x2 + Ɵ3x3…….. >=zero.

h(x)<0.5 when Ɵ0 + Ɵ1x1  + Ɵ2x2 + Ɵ3x3…….. < zero .

I implemented logistic regression with binary classification along with multiple class classification .

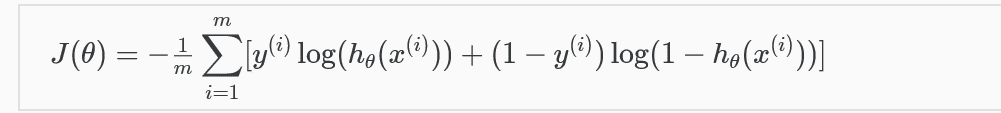
Binary classification :



Problem statement was that we want to predict what are the chances that a student will be admitted based on the result of two test :

And the data set look like this

Cost function of logistic regression :



//without regularization.

//costfunction code implemented in octave;

function [J, grad] = costFunctionReg(theta, X, y, lambda)

m = length(y); % number of training examples

J = 0;

grad = zeros(size(theta));

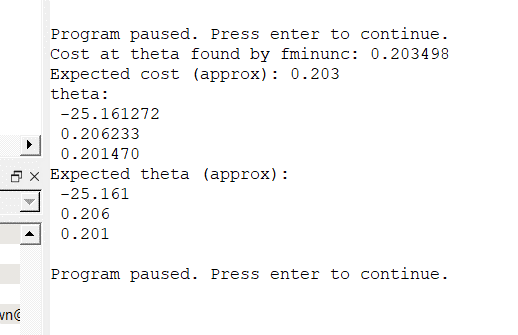
v=sigmoid(X\*theta);

J=sum(1\*[(y'\*log(v)+(1y')\*log(1v))]).\*(1/m)+sum(theta(2:end,1).^2)\*(lambda/(2\*m));(regularization term )

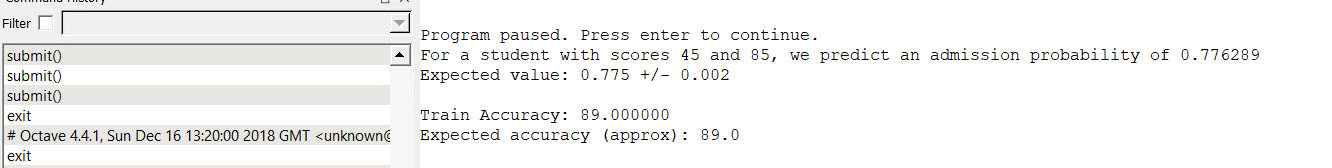
grad=(1/m)\*(X'\*(v-y))+theta\*(lambda/m);

grad(1,1)=(1/m)\*(sum(v-y));

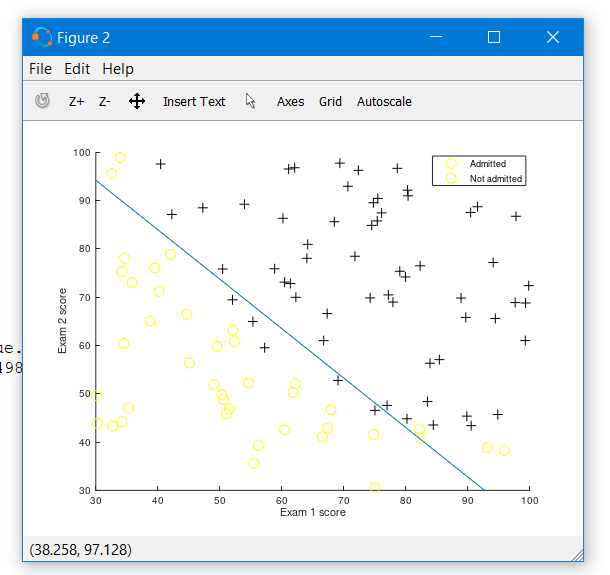
end;



// after the minimization of the cost function theta predicted.



// predicted output of my hypothesis



//Plot of data set and hypothesis

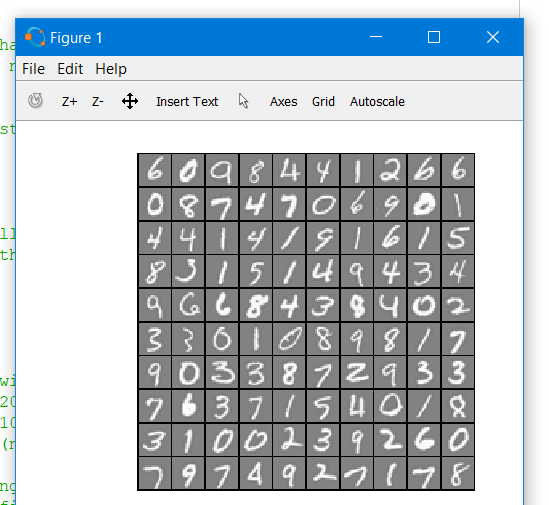
// one vs all classifier

Problem statement : digit recognition with the help of logistic regression with the help of principel one vs all .

In one vs all classifier we modify our ouput label for each class as :

for eg: we have 10 different classes (1,2…k) then for kth  we will keep the output label to be same but for all other class we will make it 0.

In other words in one vs all each classifier will be trying to classify the object between its own class and all other classes treated to be single unit (labelled 0 after modification )

// input images

Size of the images 20 \* 20 pixels

Thus total no. of features =400 +1(theta0)=401

(for such a large no. of features regularization becomes important )

// cost function : (same as above)

function [J, grad] = costFunctionReg(theta, X, y, lambda)

m = length(y); % number of training examples

J = 0;

grad = zeros(size(theta));

v=sigmoid(X\*theta);

J=sum(1\*[(y'\*log(v)+(1y')\*log(1v))]).\*(1/m)+sum(theta(2:end,1).^2)\*(lambda/(2\*m));(regularization term )

grad=(1/m)\*(X'\*(v-y))+theta\*(lambda/m);

grad(1,1)=(1/m)\*(sum(v-y));

end;

// one vs all classifier (here training theta vector(401 features) for k no. of classes (here 10 classes) )

function [all\_theta] = oneVsAll(X, y, num\_labels, lambda)

m = size(X, 1);

n = size(X, 2);

all\_theta = zeros(num\_labels, n + 1);//10\*401 dimension matrix

X = [ones(m, 1) ,X]; //5000\*401 dimension matrix

options = optimset('GradObj', 'on', 'MaxIter', 50);

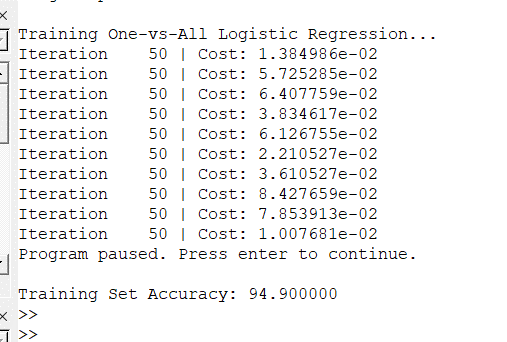
theta=ones(401,1);

for c=1:num\_labels; // no. of labels=10

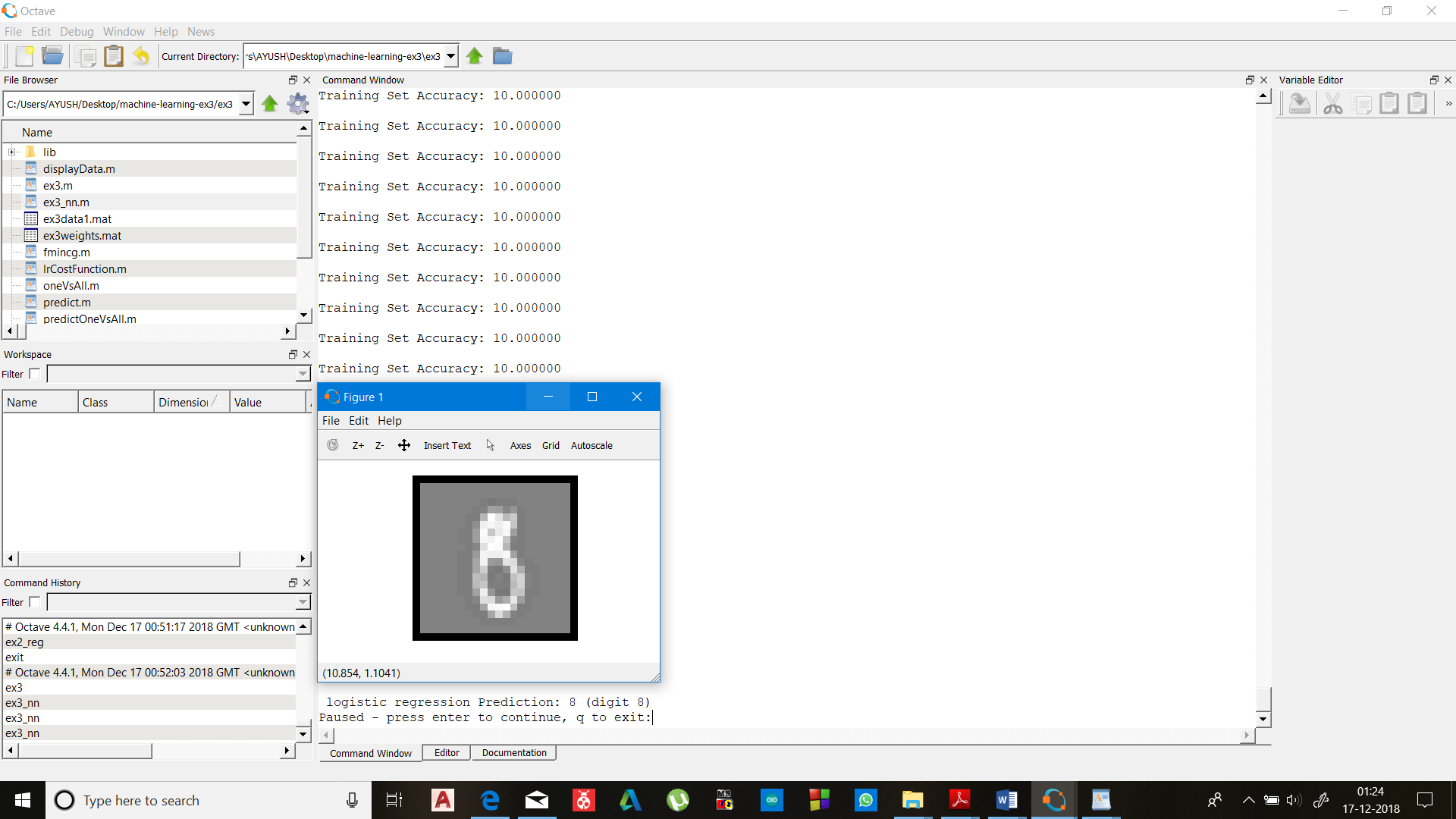
[theta]=fmincg (@(t)(lrCostFunction(t, X, (y == c), lambda)),zeros(n + 1, 1), options);

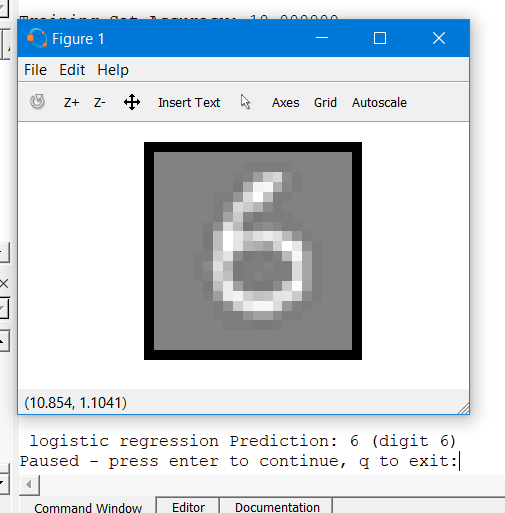
all\_theta(c,:)=theta';

end;



// result of training 10 classifiers





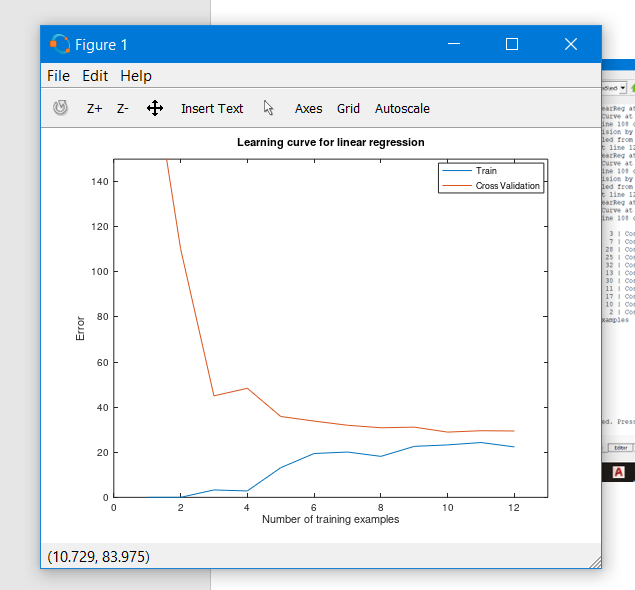
1. Learning curves :

For error analysis of our hypothesis we should divide our dataset into training set , cross validation set and test set (typically in ratio 60:20:20)

According to which we should train our hypothesis on training set . and test its performance on cross validation set .

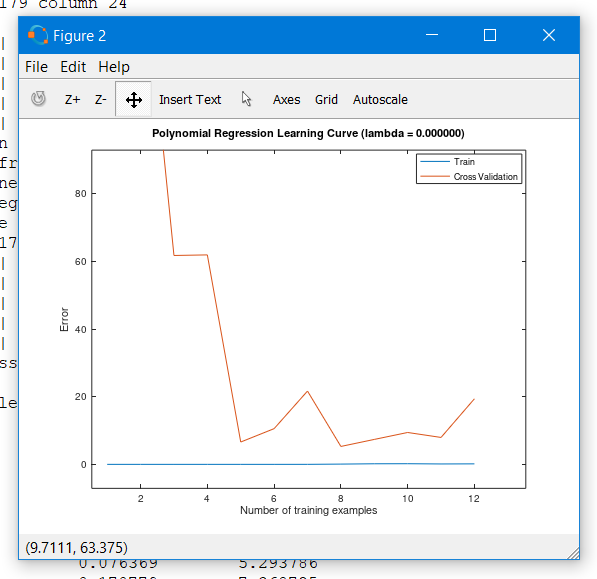
the curve between the no. of training examples to cost function (error) helps us in debugging our hypothesis.

On the other hand a very low training set error and high cross validation set error reflects high variance problem .



a high training set error and a high cross validation set error reflects the problem of high bias (underfit).

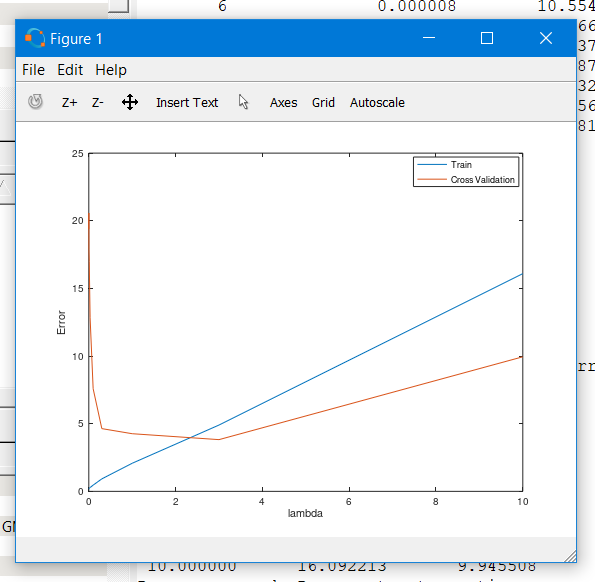
Can be overcome by increasing the no. of features , or decreasing the regularization parameter(lambda). Wasting time in collecting more dataset is unlikely to help.



// On the other hand a very low training set error and high cross validation set error reflects high variance problem .

Can be overcome by :

Increasing regularization term , increasing the dataset , decreasing the no. of features.



//plotting different values of regularization term and analysing error helps us for choosing optimum value of lambda.