Load the Observations (x_obs, y_obs)

Select 'N_train' samples for training

```

N_train = 50;
x_train = x_obs(1:N_train);
y_train = y_obs(1:N_train);
```

Select remaining point as test

```
N_test = 10;
x_test = x_obs(N_train+1:N_train+N_test);
y_test = y_obs(N_train+1:N_train+N_test);
```

Possible values of x are in this range

range(1) is (-5) and range(2) is (5)

linspace(X1, X2, N) generates N points between X1 and X2

Specify possible values for the lambda parameter to test

```
range = [-5 5];

num_basis_functions = 11;
centres = linspace(range(1),range(2),num_basis_functions);
```

Basis function widths of interest

```

lambdavals = 0.01:0.04:1;

for model=1:length(lambdavals)
```

All widths are equal

```
    lambdas=lambdavals(model)*ones(1,size(centres,2));
```

% number of basis vectors

```
 B=size(centres,2);
    sumPhi = zeros(B);
    sumyPhi = zeros(B,1);

    for n=1:N_train
        [dum, phi] = rbf(x_train(n),centres,lambdas,ones(B,1)); % The RBF
        sumPhi = sumPhi+ phi* phi';
        sumyPhi = sumyPhi+y_train(n)*phi;
    end
%
```

Each value of lambda defines a distinct ’model’.

Here you need to optimize the hyperparameters (alpha, beta) of

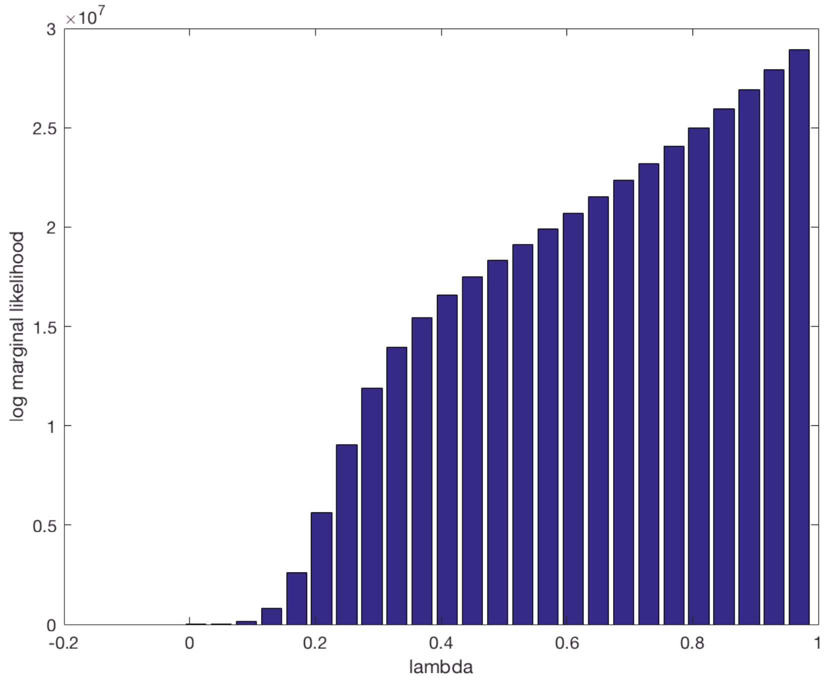
the model using the training data and compute the corresponding

marginal likelihood.

```
opts.maxits = 50;
    opts.tol=0.001;
    opts.plotprogress=0;
    [m, S, alpha, beta, marglik(model)] = BayesLinReg(y_train,sumPhi,sumyPhi,opts);
    end
```

The ptimal value of lambda, that maximizes the marginal likelihood?

```
[val, ind] = max(marglik);
lambda_opt = lambdavals(ind);
bar(lambdavals,marglik);
ylabel('log marginal likelihood');
xlabel('lambda')
```



Fit the model one more time using

The optimal lambda and the training data

```
lambdas = lambda_opt*ones(1,size(centres,2));
B = size(centres,2);
sumPhi = zeros(B);
sumyPhi = zeros(B,1);
for n=1:N_train
    [dum, phi] = rbf(x_train(n),centres,lambdas,ones(B,1));
    sumPhi = sumPhi+phi*phi';
    sumyPhi = sumyPhi+y_train(n)*phi;
end
opts.maxits = 50;
opts.tol=0.001;
opts.plotprogress=0;
[m, S, alpha, beta, marglik(model)] = BayesLinReg(y_train,sumPhi,sumyPhi,opts);
```

Compute the final regression function

```
x_coord = linspace(range(1),range(2),100);
y_mean = zeros(1,length(x_coord));
for i = 1:length(x_coord)
    [y_mean(i) phi]=rbf(x_coord(i),centres,lambdas,m);
```

predictive variance

```
varty(i)=phi'*S*phi+1/beta;
end
```

Plot the final learned regression function, together with the samples

```
figure;
plot(x_coord, y_mean);
hold on;
plot(x_train, y_train, 'kx');
plot(x_test, y_test, 'rx');
```

Make predictions for inputs in the test data, so that you get

predictions 'y_pred' for inputs in x_test.

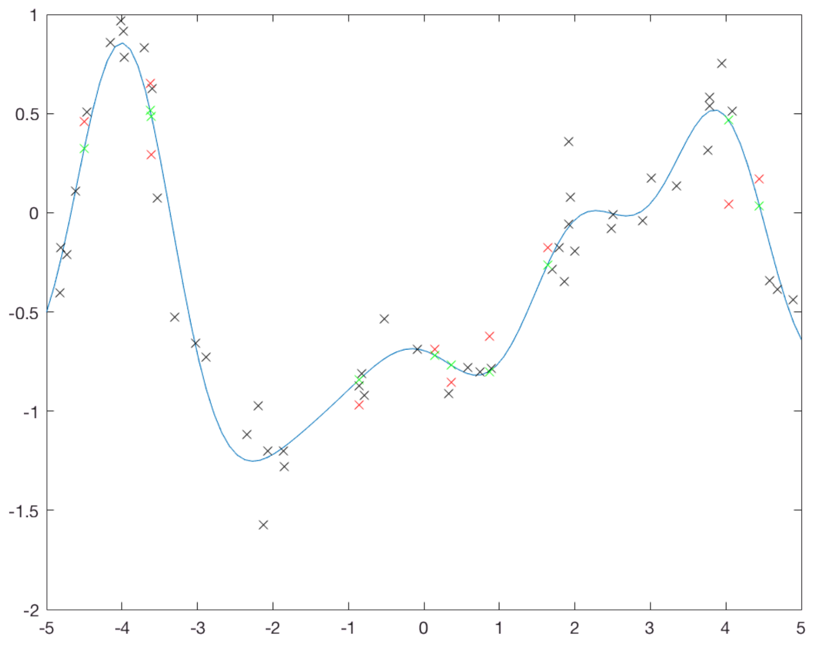
```
for ct=1:length(x_test)
```

Compute y_pred(ct)

```
    [y_pred(ct) phi] = rbf(x_test(ct),centres,lambdas,m);
end
```

Plot the predictions

```
plot(x_test, y_pred, 'gx');
```



Compute the mean squared prediction error for the test data.

```
mse_test = 1/N_test*sum((y_pred'-y_test).^2);
```