Boston Housing Exercise 10 - Kernel Ridge Regression

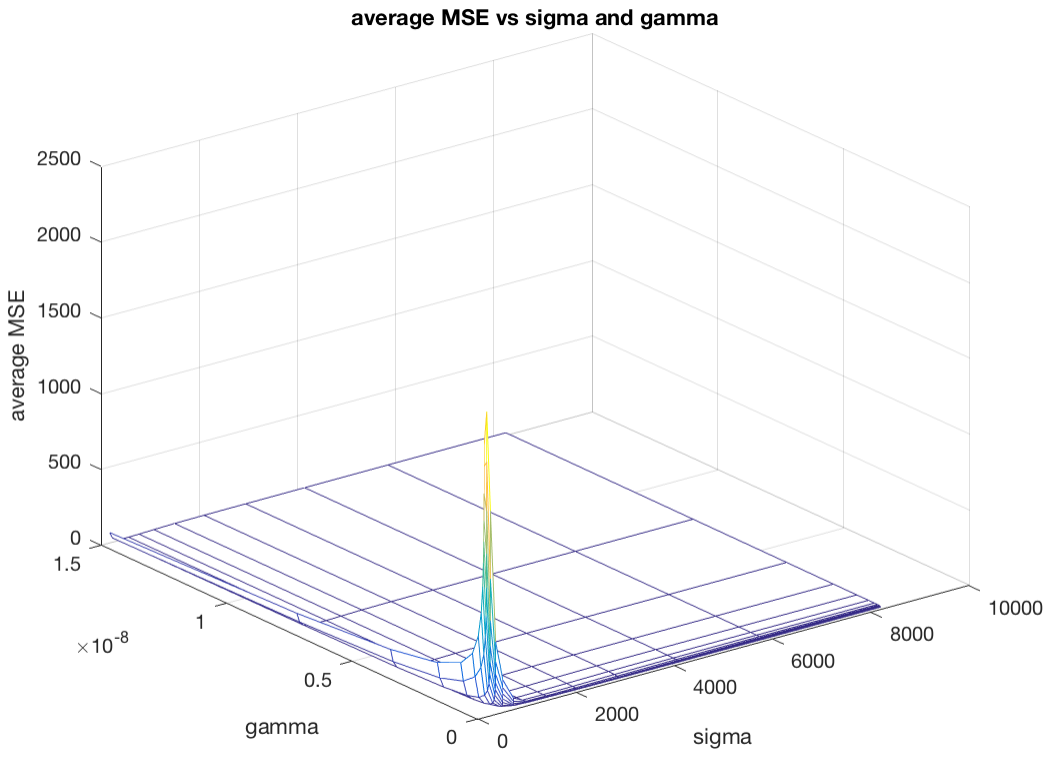
```
clear all; close all;

% For Part A and B see functions kridgereg and dualcost respectively

% Part C and D - further exploring Kernel Ridge Regression (KRR)

% setup:
p = -40:1:-26; % code scalar ridge parameter
lengamma = size(p,2);
gamma = (2*ones(1,lengamma)).^p;

pp = 7:0.5:13; % code variance parameter of gaussian kernel
lensigma= size(pp,2);
sigma = (2*ones(1,lensigma)).^pp;
data = importdata('boston.mat');
[dataS dataN] = size(data);

rng(8); % fix time for random shuffle (same value produced when re-
run)
count = 20; % for loop in Part D
fold = 5; % no of folds for cross validation

trainSize = round(dataS*2/3); % data split (1/3 test and 2/3 train)
testSize = dataS-trainSize;

MSEtrain = zeros(lengamma,lensigma,fold);
MSEtest= zeros(lengamma,lensigma,fold);

minMSEtest = zeros(1,count);
minMSEtrain = zeros(1,count);
MSEtrain4 = zeros(1,count);

for d = 1:count

    % Shuffle all data and split into separate test and training sets

    dataShuffle = data(randperm(dataS), :);
    Xdata =  dataShuffle(:,1:13);
    Ydata =  dataShuffle(:,14);

    XdataTrain = Xdata(1:trainSize,:);
    XdataTest = Xdata(trainSize + 1:dataS,:);

    YdataTrain = Ydata(1:trainSize,:);
    YdataTest = Ydata(trainSize + 1:dataS,:);

   for k =  1:fold % loop to implement cross validation
```

```
        if k == fold
        % training set does not divide into 5 folds exactly, hence make
 the
        % last fold larger than the remaining ones
            foldSize = trainSize - ((k-1)* round(trainSize/fold));
            % find index of fold within full data array:
            indexS = 1 + trainSize - foldSize;  % starting point
            indexE = trainSize;                 % end point

           % split training data into reduced training and a validation
 sets
           % on the current fold (k)
           XfoldVald = XdataTrain(indexS:indexE, :);
           XfoldTrain = XdataTrain(1:(indexS-1),:);

           YfoldVald = YdataTrain(indexS:indexE, :);
           YfoldTrain = YdataTrain(1:(indexS-1),:);

        else
            foldSize = round(trainSize/fold);
            % find index of fold within full data array:
            indexS = 1 + foldSize*(k - 1);  % starting point
            indexE = foldSize*k;            % end point

           % split training data into reduced training and a validation
 sets
           % on the current fold (k)
           if k ~= 1
           XfoldVald = XdataTrain(indexS:indexE, :);
           XfoldTrain = XdataTrain([1:(indexS-1), (indexE
+1):size(XdataTrain,1)],:);

           YfoldVald = YdataTrain(indexS:indexE, :);
           YfoldTrain = YdataTrain([1:(indexS-1), (indexE
+1):size(YdataTrain,1)],:);

           else
           XfoldVald = XdataTrain(indexS:indexE, :);
           XfoldTrain = XdataTrain((indexE+1):size(XdataTrain,1),:);

           YfoldVald = YdataTrain(indexS:indexE, :);
           YfoldTrain = YdataTrain((indexE+1):size(YdataTrain,1),:);
           end
        end
      %loop through each gamma and sigma configuration :
        for i = 1:lengamma
             for j = 1:lensigma
                 % construct global gaussian Kernel matrix:
                 K = MygaussKernel(XfoldTrain, XfoldVald, sigma(1,j));

                 % as per Eq. 14 select the adequate portions of the
                 % 'big' Kernel matrix:
                 trainK = K(1:size(XfoldTrain,1),1:size(XfoldTrain,1));
```

```
                testK =
 K(size(XfoldTrain,1)+1:size(K,1),1:size(XfoldTrain,1));

                % find alpha as per Eq. 12:
                alpha = kridgereg(trainK,YfoldTrain,gamma(1,i));

                % find MSE as per Eq. 15:
                MSEtest(i,j,k) = dualcost(testK,YfoldVald,alpha);

            end
        end
    end

% initailise variables to average over all folds:
sumMSEtest = zeros(size(MSEtest,1),size(MSEtest,2));

% sum over all folds:
for kk=1:fold
   sumMSEtest= sumMSEtest+ MSEtest(:,:,kk);
end
% average over all folds:
aveMSEtest=sumMSEtest/fold;

% Part C(1) - plot cross validation error - interpreted as the average
 MSE
% on test set over all folds, as a function of sigma and gamma

if d==count %only plot for one iterration
    figure;
    mesh(sigma,gamma,aveMSEtest);
    xlabel('sigma');
    ylabel('gamma');
    zlabel('average MSE');
    title('average MSE vs sigma and gamma');
end

% Part C(2)- calculate train and test MSE for the best sigma and gamma
% pair. This is, for the sigma and gamma pair which yield the lowest
 aveMSE.

% based on test set MSE:
[row col] = find(aveMSEtest == min(min(aveMSEtest)));
bestGammaTest = gamma(1,row);
bestSigmaTest = sigma(1,col);
minMSEtest(1,d) = min(min(aveMSEtest));% store for each iteration of
 Part D

% using best paremetres need to need to recaluclate for train set:

 KK = MygaussKernel(XdataTrain,[], bestSigmaTest);
 alpha4 = kridgereg(KK,YdataTrain,bestGammaTest);
 MSEtrain4(1,d) = dualcost(KK,YdataTrain,alpha4);
```

```
if d==count %only output for one iteration - answer to Part C(2)
fprintf('After fixing the best sigma and gamma parameters for
 the test and training sets,\nimplementing KRR, the MSE on
 the test set was found to be %.3f and on the training\nset
 %.3f.',minMSEtest(1,d),MSEtrain4(1,d));
end

end

% Part D cnd
% finding mean and standard deviation of MSE on test and training sets
 after
% 20 iterrations :
krrMEANminMSEtest = mean(minMSEtest);
krrSDminMSEtest = std(minMSEtest);

krrMEANminMSEtrain = mean(MSEtrain4);
krrSDminMSEtrain = std(MSEtrain4);

fprintf('\nAfter iterrating 20 times,\nthe average MSE on
 the test set was found to be %.3f and on the training\nset
 %.3f.',krrMEANminMSEtest,krrMEANminMSEtrain);

% required summary table will be provided in report
save('ex10VARS','krrMEANminMSEtest','krrSDminMSEtest','krrMEANminMSEtrain','krrSDm
```

```
After fixing the best sigma and gamma parameters for the test and
 training sets,
implementing KRR, the MSE on the test set was found to be 15.578 and
 on the training
set 10.612.
After iterrating 20 times,
the average MSE on the test set was found to be 13.043 and on the
 training
set 7.751.
```



Published with MATLAB® R2016b