Coursework 1: Machine Learning & Neural Computation

Course Coordinator: Dr. Aldo Faisal

Report Prepared By: Jenna Luchak, MSc HBR
Email: jkl17@ic.ac.uk
CID: 01429938

Date Submitted: November 20, 2017

Question 1

The boundaries of Greater London are defined as -0.510˚ to 0.303˚ for Longitude and 51.30˚ to 51.72˚ for latitude. The boundaries were found by hovering a computer cursor over a longitude and latitude tracer map of Greater London and then recording the maximum and minimum values seen[1][2]. To produce an accurate plot of the rental price raw data of just Greater London, data points located outside of the defined latitude and longitude boundary ranges were removed from the assignments provided data set. To see the Matlab commands used in this filtering process, see the transcribed code in the Appendix. A few different views of the plotted data can be found in Figure 1 below.

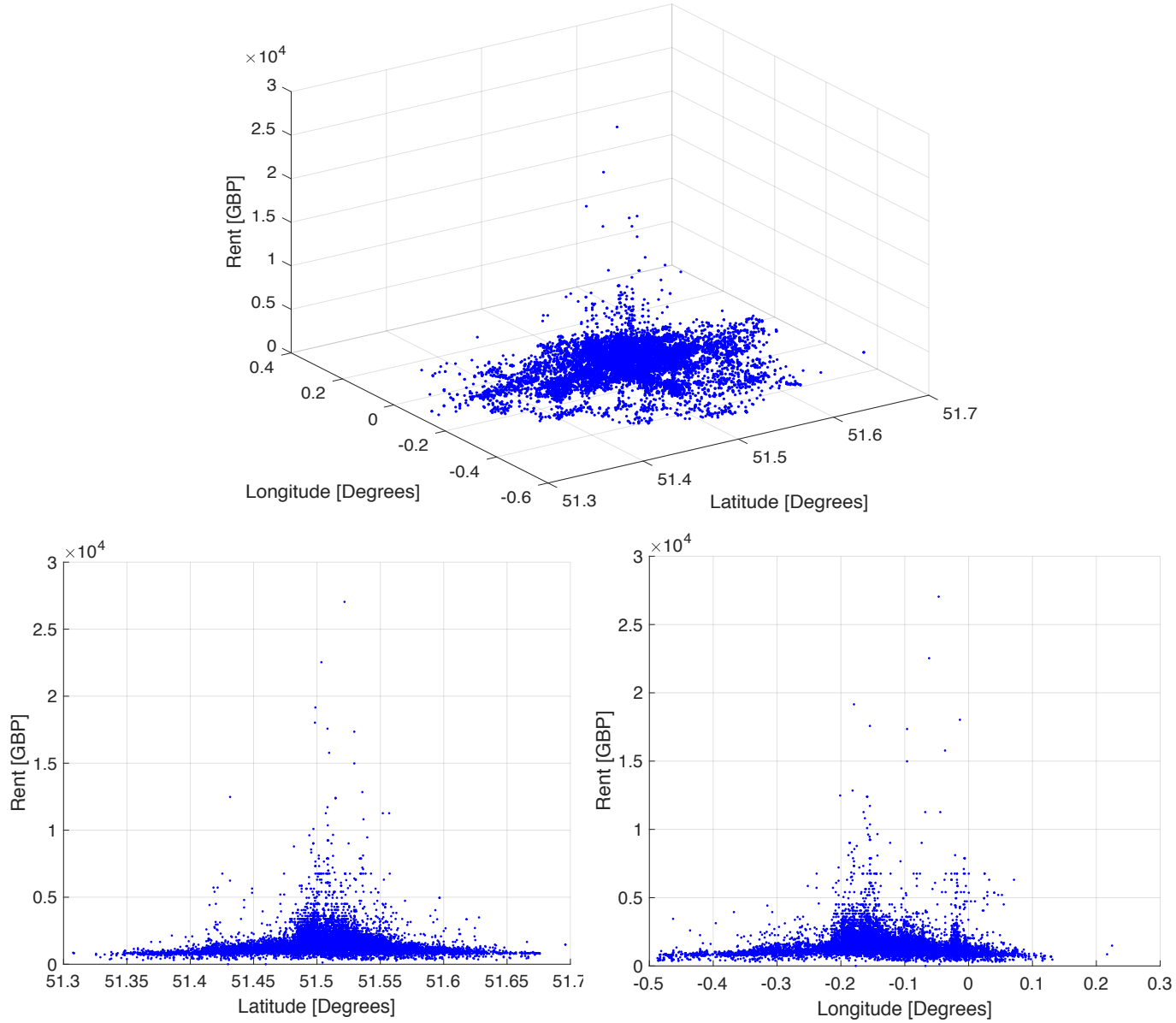

Figure 1: Three views of one data set, representing the predicted rental prices, in GBP, for a single bedroom property in Greater London with respect to its latitude and longitude coordinates in degrees. The data is plotted with longitude along the x-axis, latitude along the y-axis and rental prices along the z-axis. [Top] A three-dimensional view. [Bottom left] A two-dimensional view in the YZ plane. [Bottom Right] A two-dimensional view in the XZ plane.

Observably, the 3D plot in Figure 1 resembles a multivariate Gaussian bell curve. The data is normally distributed in both the XZ and YZ planes. The distribution in the XY plane is approximately radially symmetric around a central cluster of data points. The central cluster of data points with maximum rental prices occur when latitude is between 51.45˚ and 51.55˚ and longitude

is between -0.2˚ and 0˚. The average rental price across all data points is £1629.20 while the mode price is £1578.00.

Question 2/Question 3

The optimal computational model was selected under the criteria that the model should not be over fitted, and will predict a faux home price while maintaining a low root mean square error (RMSE) that minimizes computation time.

The first decision made in this model was to use a Gaussian basis function. The Gaussian was chosen over basis functions like binomial, and sigmoidal basis functions because it is a type of radial basis function, and since the data sets distribution, as seen in Figure 1 above, resembles a multivariate Gaussian bell curve; a Gaussian function is the best choice to fit the data.

The next two design choices were how many Gaussian equations to use, and how to split the data points to calculate function parameters? To an extent, as the number of Gaussian functions increase, RMSE decreases, but computing time increases; therefore, to find the optimal choices, a variety of data splitting methods were tested. Note that for these tests a default four-fold cross validation scheme was used to calculate the RMSE of each method.

The first method tested split the data into clusters that followed a grid system. The latitude and longitude plane was split into a grid of equi-sized rectangles, dependent on the number of Gaussian functions, and the data was split such that a new mean and covariance term was calculated for the data points that fell inside it's corresponding area. The plane was split into an even number of sections from 2 to 14, all resulting in a constant RMSE of approximately 960.

The second method tested split the data into clusters according to its rental price value. All of the rental price data points were first sorted by ascending value. Next, depending on the number of Gaussians used, the data set was split into even number clusters according to its price value order. For example, if the number of Gaussians was two, the top half of data points with the largest rental prices would be one cluster, while the bottom half was another cluster. This method was over fitted at 15 Gaussians. The RMSE for 1 to 14 Gaussians tested remained constant at approximately 1000.

The final method grouped the data using rings of varying radii, all concentric around the same point. The center point was determined by calculating the average latitude and longitude of the top 1000 rental price data points. The number of rings was dependent on the number of Gaussian functions defined. This method became over fitted at 25 Gaussians; the lowest RMSE occurred when using 10 to 25 functions, as the RMSE error was approximately constant at 850.

The third method produced the lowest RMSE and was chosen as the optimal method to cluster the data for parameter calculation.

The last step in the model design was to determine the optimal number of folds for splitting the data into testing and training data sets. To offset the lack of enough data points, a cross validation method was used. The data points were shuffled randomly and then split into a certain number of groups (folds). Fold values from 4 to 900 were tested. As the number of folds chosen increased, the RMSE continually decreased. For example, the RMSE at 10 folds was on average 848, but at 900 folds, RMSE was on average 663. To get the model with the lowest error, one might select the highest fold number; however, as fold number increases, computation time greatly increases; therefore having a fold number too high would be unacceptable. Upon inspection of the predicted rental prices computation, it was found that if the fold number was at least 20, the predicted rental price of my faux home location never changed by more than a few pence, therefore, even though RMSE could be further decreased, the predicted prices accuracy would not improve. To keep computation time low and accuracy high, a fold number of 20 were selected.

In summary, the optimal design choices selected for this model include a fitting function of 10 Gaussian basis functions, where the mean and covariance parameters are calculated using a concentric clustering method, and the model is trained and assessed using a 20 fold cross validation scheme. See the Appendix for an annotated explanation of the models Matlab code.

Question 4

A heat map of the models predicted rental price results can be seen in Figure 2 below.

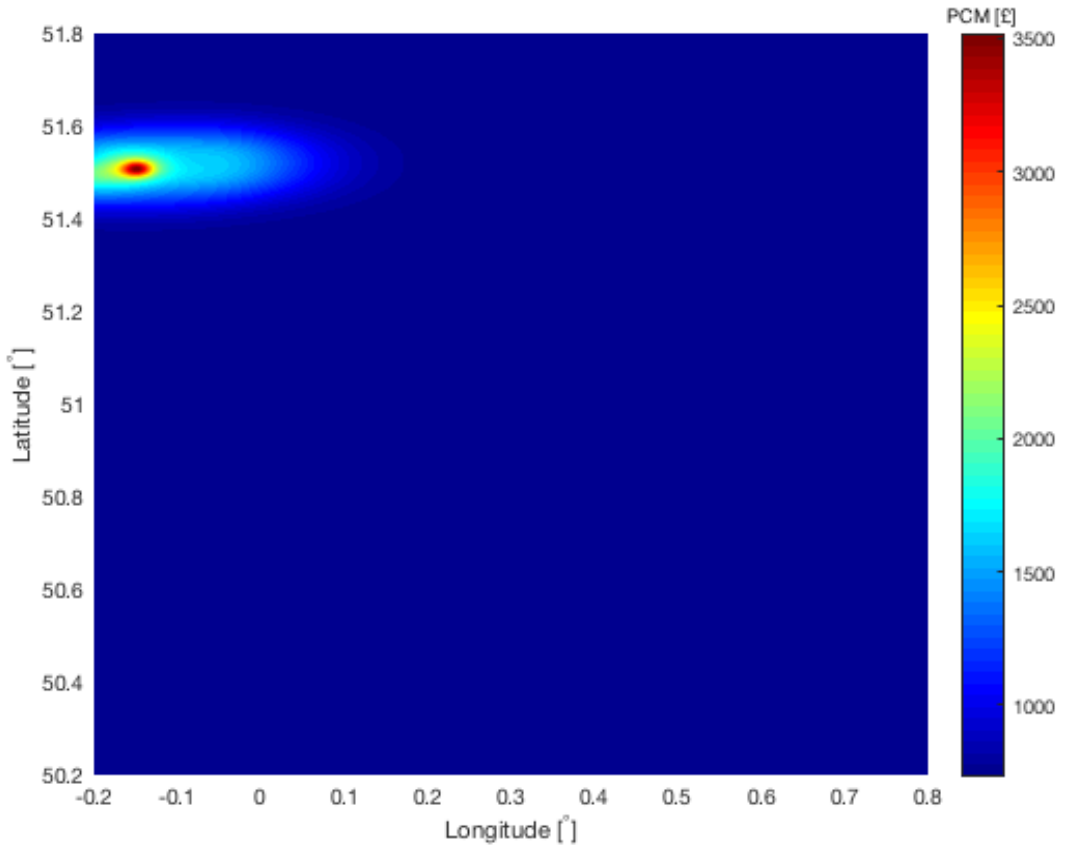

Figure 2: A two-dimensional Heat map that expresses the range of predicted rental prices with respect to the latitude-longitude plane. The color red indicates the areas with the highest predicted rental price in GBP; where as dark blue indicates the lowest.

A majority of the space captured in the heat map, as seen in Figure 2, accounts for area outside of Greater London. The complete range of latitudes that define the Greater London area (51.39˚ to 51.72˚) is captured in the heat map, but the area from longitude -0.51˚ to -0.20˚ is not included in the map. This results in 38.1% of the Greater London area and 16.46% of the data set missing from the map. Therefore, the heat map captures the data set only to a moderate amount.

Additionally, artifacts of the models choices can be seen in Figure 2. For example, the heat map contains only one circular zone of maximum pricing and all predicted values around it gradually decrease with distance from the center, similar to a Gaussian bell curve distribution. Additionally, the dark red zone (maximum pricing) on the heat map is in the same location as the calculated center point in the model (51.5˚ latitude and -0.15˚ longitude).

Question 5

The corresponding faux home tube stop for CID 1429938 is Holland Park; the coordinates of Holland Park are 51.5072˚ latitude and -0.20553˚ longitude. The computational model predicted the rental price for this location to be £1969.39 with a RMSE of approximately 829. This predicted rental place indicates that the Holland Park area is an above average rental price location, as the average rental price of Greater London is £1629.20.

Comparing the faux home tube stop rental price with the expected model results from the heat map, as seen in Figure 2 above. The faux home tube stop location is not visible on the heat map, as the longitudinal axis ends at -0.2˚. However, by extrapolating the data it can be inferred that the location would land in the green/light blue range, which indicates a rental price between £1500 and £2000. Therefore, the faux home tube stop predicted rental price was an expected result, with respect to the computational model. However, when you compare the faux home tube stop predicted price with reality, a different conclusion can be formed. The average actual price of 22 single bedroom properties within a ¼ mile radius of Holland Park station is £2734.09[3]. This results in a percent error of 38.8% between the actual and predicted price. In conclusion, the model is only moderately accurate at generalizing real life results.

References

[1] "Degrees Minutes Seconds to Decimal Degrees Coordinates", *Latlong.net*, 2017. [Online]. Available: https://www.latlong.net/degrees-minutes-seconds-to-decimal-degrees. [Accessed: 11- Nov- 2017].

[2] "Map of Greater London districts and boroughs - Maproom", Maproom, 2017. [Online]. Available: https://maproom.net/shop/map-greater-london-districts/. [Accessed: 11- Nov- 2017].

[3] "Rightmove.co.uk", Rightmove.co.uk, 2017. [Online]. Available: http://www.rightmove.co.uk/property-to-rent/map.html?locationIdentifier=STATION%5E4658&maxBedrooms=1&minBedrooms=1&numberOfPropertiesPerPage=499&radius=0.25&includeLetAgreed=false&viewType=MAP. [Accessed: 20- Nov- 2017].

Appendix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prepared By: Jenna Luchak
% CID: 01429938
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Clear the workspace and command window before begining the script
clc;
clear;

%% Define the useable data for this assignment
% i.e. Load and then filter the data set such that it only represents the
% Greater London Area.

% Load the dataset
    load('london.mat');

% Rename subsets of the structure array Prices
% i.e. Define variables to represent the three columns:latitude, longitude
% and rent prices
    lat = Prices.location(:,1); % Latitude data points in Degrees
    long = Prices.location(:,2); % Longitude data points in Degrees
    rent = Prices.rent; % Rent prices data points in GBP

% Filter the Prices subsets with respect to latitudes
% i.e. Only keep data points with latitudes less than 51.72 degrees and greater
% than 51.30 degrees.
    lat_2 = lat(lat < 51.72 & lat > 51.30); % In Degrees
    long_2 = long(lat < 51.72 & lat > 51.30); % In Degrees
    rent_2 = rent(lat < 51.72 & lat > 51.30); % In GBP

% Filter the Prices subsets AGAIN, this time with respect to longitudes
% i.e. Only keep data points with longitudes less than 0.303 degrees and greater
% than -0.510 degrees.
    lat_3 = lat_2(long_2 < 0.303 & long_2 > -0.510); % In Degrees
    long_3 = long_2(long_2 < 0.303 & long_2 > -0.510); % In Degrees
    rent_3 = rent_2(long_2 < 0.303 & long_2 > -0.510); % In GBP

%% Question #1
% Print a 3-D plot of the filtered dataset and the Tube Station locations
% i.e. Latitude on the x axis, longitude on the y axis and rental prices on
% the z axis.
    figure(1)
    plot3(lat_3,long_3,rent_3,'.b'); % Display rent locations as blue dots
    view(-34,31)
    grid on % Display the gridlines on the plot
    xlabel('Latitude [Degrees]','fontsize',15); % x axis title
    ylabel('Longitude [Degrees]','fontsize',15); % y axis title
    zlabel('Rent [GBP]','fontsize',15); % z axis title
    set(gca,'fontsize',12)

% Display the 3D plot in the xz plane
    figure(2)
    plot3(lat_3,long_3,rent_3,'.b'); % Display rent locations as blue dots
    view([0,0])
    grid on
    xlabel('Latitude [Degrees]','fontsize',15); % x axis title
    zlabel('Rent [GBP]','fontsize',15); % z axis title
    set(gca,'fontsize',15)

% Display the 3D plot in the yz plane
    figure(3)
    plot3(lat_3,long_3,rent_3,'.b'); % Display rent locations as blue dots
```

```
    view([90,0])
    grid on
    ylabel('Longitude [Degrees]','fontsize',15); % y axis title
    zlabel('Rent [GBP]','fontsize',15); % z axis title
    set(gca,'fontsize',15)

%% Split Data into Training and Testing Data
% i.e. Randomly shuffle the data points and then divide the shuffled data
% into a number of sections, such that one of those sections is defined
% as testing data and the other remaining sections as training data.

% Redefine the filtered data subset into one matrix
    data = [lat_3,long_3,rent_3];
    rows = length(data); % The number of rows in data matrix

% Randomly shuffle each row of data
    shuffled_data = data(randperm(size(data,1)),:);

% Define the number of folds to divide the data into
    k=20;

% Calculate the integer of data points in each fold
    num = round(rows/k,0);

% Create a vector to identify which row of data starts each fold
    intervals = zeros(1,k+1);  % Empty vector
    intervals(1:k) = 1:num:((k)*num); % First to the second last element of
vector
    intervals(k+1) = rows; % The last element of vector

% Divide the data into a k x 2 structure array to prepare for
% cross-validation,such that each element of the structure array defines
% the testing and traing data as a new group of data points
    for i=2:1:k+1
    % Define data
        fold(i-1,2).a = shuffled_data(intervals(i-1):intervals(i),:);

    % Define training data
        temporary = shuffled_data; % Create a temporary matrix containing all
data points
        temporary(intervals(i-1):intervals(i),:)=[]; % Delete testing data from
temporary matrix
        fold(i-1,1).a = temporary;
    end

%% Set up an outer "for" loop to compute the data needed for Questions 2 and 3
% Through each pass of the outer for loop, a new training and testing data
% set is called, for the trainRegressor and testRegressor functions
 for i=2:1:k+1
%% Question 2

% Define the training data set
    training = fold(i-1,1).a;

% Define the input variables for the trainRegressor function
    trainIn = training(:,1:2);
    trainOut = training(:,3);

% Call the trainRegressor function
    param = trainRegressor(trainIn, trainOut);

% Store the weights parameter into a matrix
    w(i-1,:) = param.w;
```

```
%% Question 3

% Define the testing data set
    testing = fold(i-1,2).a;

% Define the input variable for the testRegressor function
    testIn = testing(:,1:2);

% Call the testRegressor function
    val = testRegressor(testIn, param);

%% Calculate the Root Mean Square Error (RMSE) of the Rental price Predictions

% Reset the value of sum to zero
    sum=0;

% Calculate the summation of error between each predicted rental price and
% and the actual testing data point
    for j=1:1:length(testing)
        sum = sum + (val(j)-testing(j,3))^2;
    end
% Calculate RMSE
    RMSE(i-1) =sqrt(sum/length(testing));

 end
%% Check that the average RMSE of the computational model is acceptable

 % Calculate the average root mean square error of all folds of test data
    avg_RMSE = round(mean(RMSE),0);

 % Calculate the average real rental price for all data points
    avg_rental = mean(data(:,3));

 % Print a response to the proposed question
     if avg_RMSE<avg_rental
         fprintf('Root Mean Square Error of %.1f is an Acceptable
Solution\n',avg_RMSE);
     else
         fprintf('Root Mean Square Error of %.1f is NOT an Acceptable
Solution\n',avg_RMSE);
     end

%% Sanity Check
% Check to make sure both trainRegressor and testRegressor functions pass
% the sanity check
    sanityCheck ( @trainRegressor ,@testRegressor )

%% Question #4

% Average the parameter matrix "W"
    average_W = mean(w)';

% Substitute the average "w" parameter back into param structure array
    param.w = average_W(:);

% Call the heatmapRent function
    figure(4)
    heatmapRent(@testRegressor, param);

%% Question #5
```

```
% Define the input for personalisedTube function
    CID = 1429938;

% Call the personalisedTube function to determine the latitude and
% longitude coordinates of my faux home tube stop
    [TubeStop, TubeCoor] = personalisedTube(CID);

% Define the input for testRegressor function
    In = TubeCoor;

% Call the testRegressor function to determine the predicted rent at my
% faux home tube stop
    fauxhome = testRegressor(In, param);
    fprintf('The predicted price is %.2f GBP\n',fauxhome);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```