Assignment 1, COMP4702

Roy Portas

March 27, 2017

Question 1.2

The first column of the data is the date, the second column appears to be a unique ID for each entry.

The third column contains numbers between 25 and 30, this is possibly a temperature in degrees. The values also change gradually which seems correct for the given time intervals

The fourth column contains numbers between 26 and around 50000. If this is plotted against the ID field, it produces a line.

The fifth column contains numbers between 7.3 and 8.3, with a mean of 7.846 and a standard StdDev of 0.142. This suggests that the value doesn't change much.

It could possibly be weather data, containing temperature, humidity, etc.

Question 1.6

```
% in is the input array
% n is the group size
function out = q6(in, n)

    out = [];

    chunks = length(in)/n;

    for i = 1:chunks
        end_ind = length(in) - i * n + n;

        start_ind = end_ind - n + 1;
        start_ind = max([1, start_ind]);

        temp = in(start_ind:end_ind);

        out = [out, temp];
    end

end
```

Question 2.1

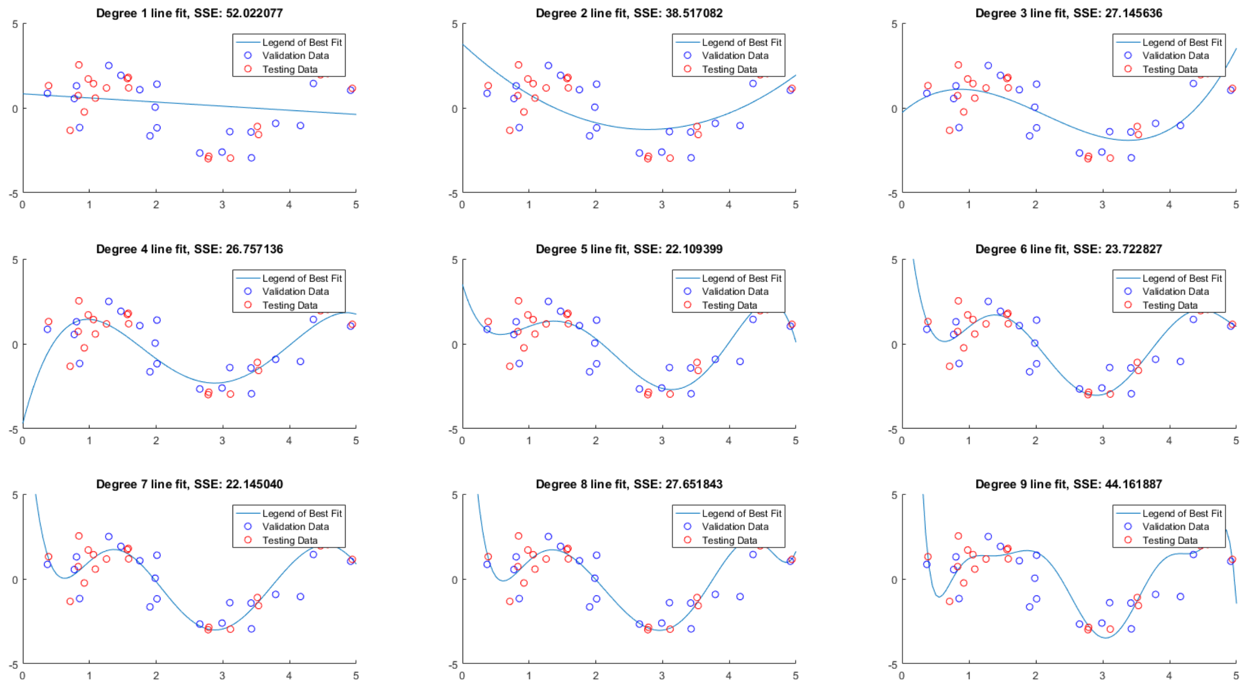

Figure 1: Lines of best fit

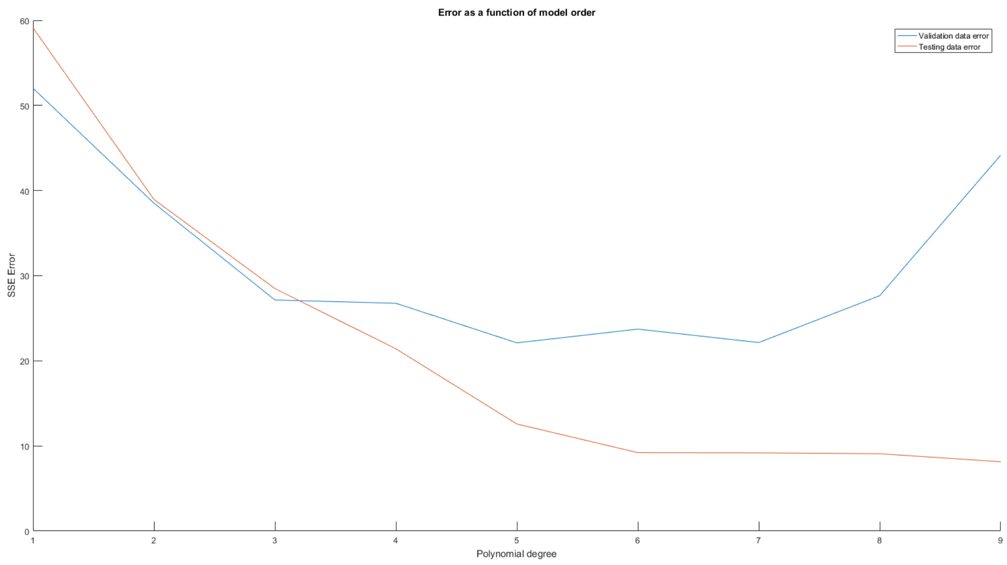

Figure 2: Error vs Polynomial Degree

The above figure shows that as more polynomial degrees are introduced the error on the testing data decreases, which is expected as this was the data the model was trained on.

However the error on the validation data increases after the 7th polynomial, thus we are overfitting the data.

Question 2.4

```
function q4(data, class)
    class_names = unique(class);
    x_values = 1:length(class);

    class1 = zeros(1, length(class));
    class2 = zeros(1, length(class));

    for i = x_values
        if strcmp(class{i}, class_names{1})
            class1(i) = 1;
        else
            class2(i) = 1;
        end
    end

    % Verify the classes are correct
    % figure;
    % hold on;
    % scatter(1:length(class1), class1);
    % scatter(1:length(class2), class2);
    % hold off;

    estimate_range = 1:0.1:8;

    class1_data = data;
    class1_data(class2 == 1) = NaN;

    class1_mle = mle(class1_data);
    class1_pdf = normpdf(estimate_range, class1_mle(1), class1_mle
       (2));

    class2_data = data;
    class2_data(class1 == 1) = NaN;

    class2_mle = mle(class2_data);
    class2_pdf = normpdf(estimate_range, class2_mle(1), class2_mle
       (2));


    % figure;

    % scatter(x_values, data);

    % Verify the classes are divided
    figure;

    hold on;

    yyaxis left;
    scatter(class1_data, x_values, 'r');
    scatter(class2_data, x_values, 'b');

    yyaxis right;
    plot(estimate_range, class1_pdf, 'r');
    plot(estimate_range, class2_pdf, 'b');
    legend('Iris Setosa', 'Iris Versicolor');
    hold off;

    % Plot the likelihood
    figure;
    hold on;
    plot(estimate_range, class1_pdf);
    plot(estimate_range, class2_pdf);
    xlim([1, 10]);

    title('Likelihoods');
    xlabel('x');
    ylabel('P(x|C_i)');

    p_class1 = class1_pdf ./ (class1_pdf + class2_pdf);
    p_class2 = class2_pdf ./ (class1_pdf + class2_pdf);

    figure;
    hold on

    plot(estimate_range, p_class1);
    plot(estimate_range, p_class2);

    title('Posteriors');
    xlabel('x');
    ylabel('P(x|C_i)');

    hold off;

```

```
end
```

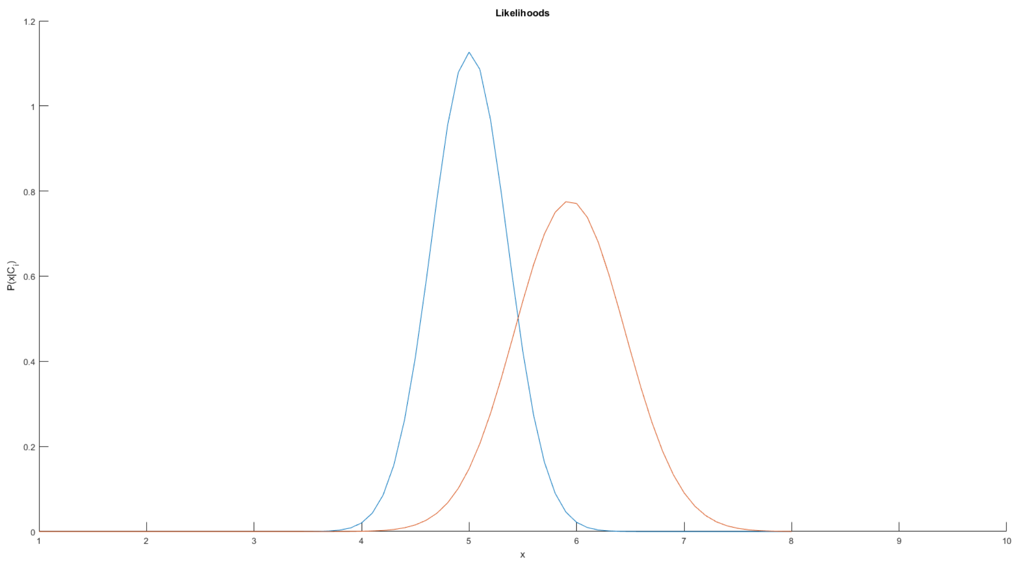

Figure 3: Likelihoods

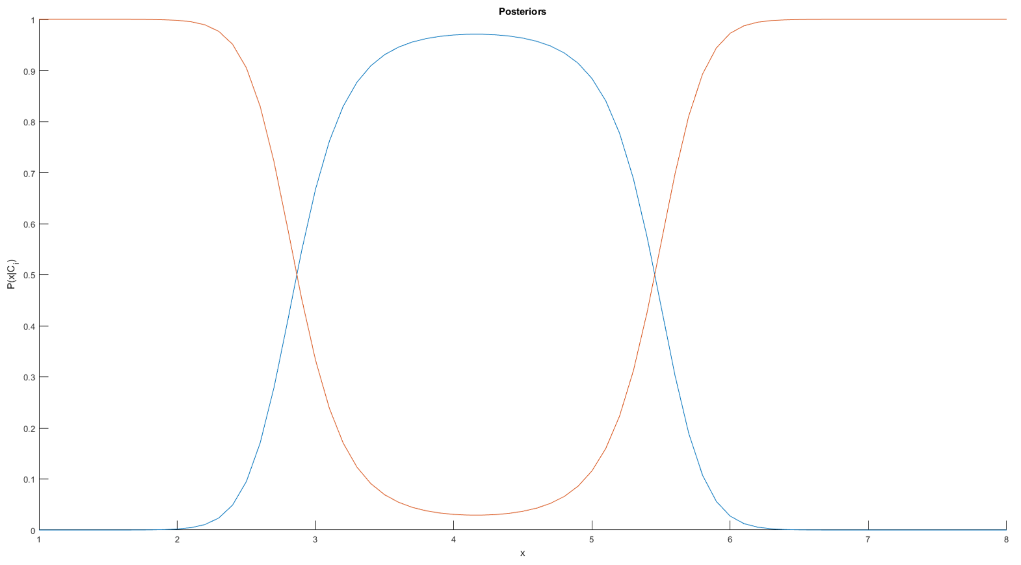

Figure 4: Posteriors

Question 3.1

With the given dataset, we want to find a way to classify the two classes, this can be given by the posterior $P(C_i|x)$. To do so we first need to estimate the prior $P(C_i)$ and $p(x|C)$.

This will allow us to estimate the proportion of data points for a given class i.

$p(x|c)$ can easily be found by using the matlab function `mvnpdf`.

Finding the posteriors for each class yields the below histogram.

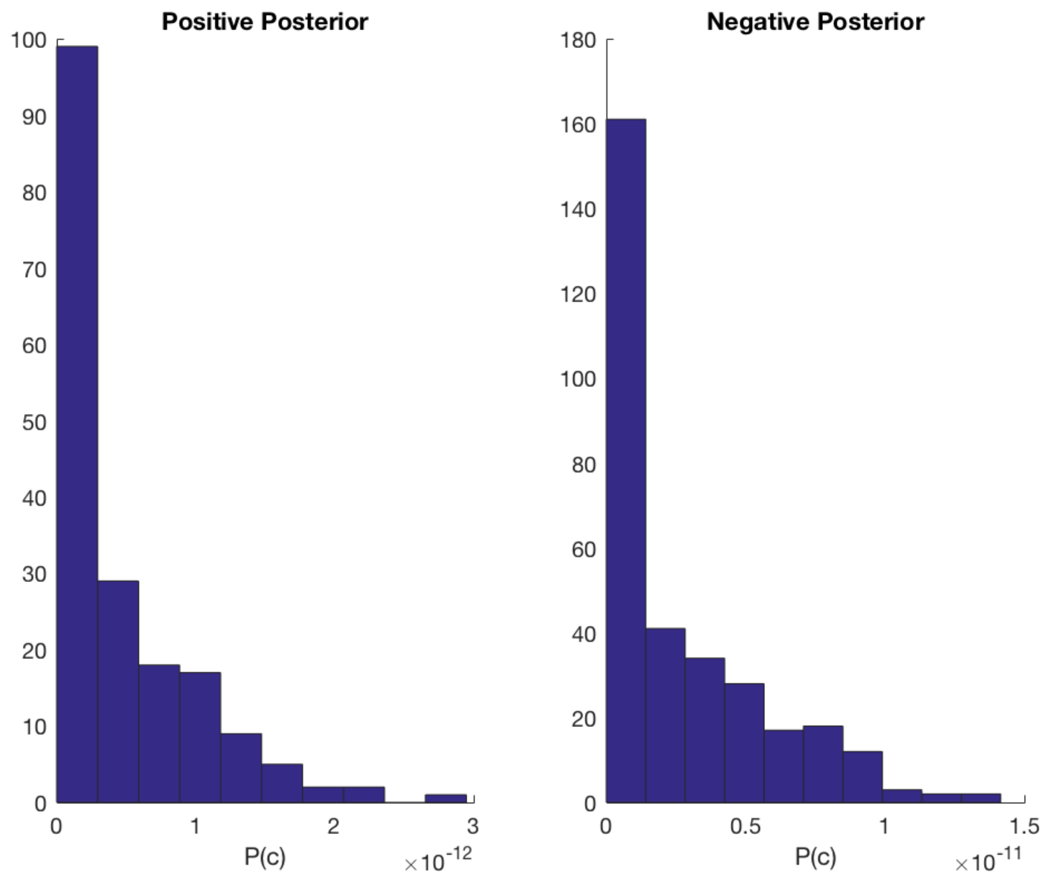

Figure 5: Posteriors

Question 3.2

Question 3.5

The computed KL values are listed in the table below

M and H1	2.6898
M and K1	0.2741
M and K2	0.28186

There was an issue encountered while finding these variables, it was caused by the 0 values in some of the bins of the histogram estimator, this caused the division to produce Infinity values. This was fixed by replacing the infinity values with 0.

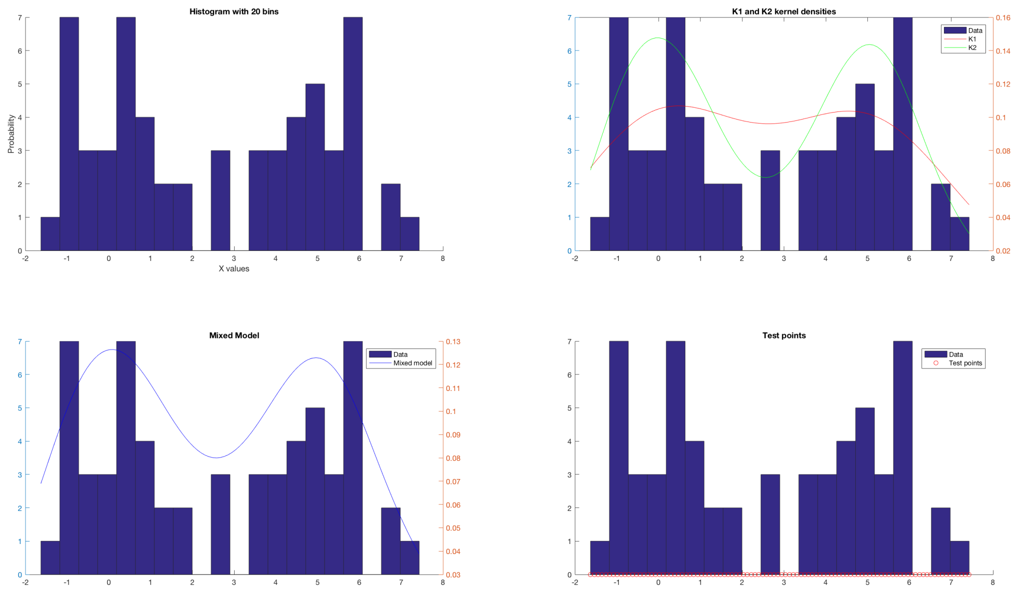

Figure 6: Q5

Below is the code used to calculate the KL function, see the function called `KL` at the bottom of the script.

```
% Q5

x = [randn(30, 1); 5 + randn(30, 1)];

subplot(2, 2, 1);
hold on;
hist(x, 20);
title('Histogram with 20 bins');
xlabel('X values');
ylabel('Probability');
hold off;

% Demo the results
subplot(2, 2, 2);
yyaxis left;
hist(x, 20);
yyaxis right;

% Generate 100 random datapoints between that covers the range of
    data in x
x_points = min(x):((max(x)-min(x))/99):max(x);

```

```
% Generate the histogram bin counts
[dist, edges] = histcounts(x, 20);
N = sum(dist);
x0 = edges(1);
h = edges(2) - edges(1);

data = zeros(1, 100);

% Histogram estimator
for i = 1:100
    bin = ceil((x_points(i) - x0) / h);
    x_in_bin = dist(bin);

    data(i) = x_in_bin / (N * h);
end


% Create the kernel density estimators

% Default kernel width
[K1, k1x, bw1] = ksdensity(x, x_points);

% Half of the default kernel width
[K2, k2x, bw2] = ksdensity(x, x_points, 'width', bw1/2);

hold on;
plot(k1x, K1, 'r');
plot(k2x, K2, 'g');

title('K1 and K2 kernel densities');
legend('Data', 'K1', 'K2');
hold off;

% Calculate the Guassian mixed model
mixed = K1/2 + K2/2;

subplot(2, 2, 3);
hold on;
yyaxis left;
hist(x, 20);
yyaxis right;
plot (k1x, mixed, 'b');

title('Mixed Model');
legend('Data', 'Mixed model');
```

```
hold off;

subplot(2, 2, 4);
hold on;
hist(x, 20);
scatter(x_points, zeros(1, 100), 'r');

title('Test points');
legend('Data', 'Test points');
hold off;

% Do the calculations
KL(mixed, data)
KL(mixed, K1)
KL(mixed, K2)

%
% Calculate the KL Divergence
%
function result = KL(p, q)
    result = zeros(size(p));

    valid = p > 0 & q > 0;

    result1 = sum(p(valid) .* log(p(valid) ./ q(valid)));
    result2 = sum(q(valid) .* log(q(valid) ./ p(valid)));

    result(valid) = result1 + result2;

    result = mode(result);
end
```