Assignment 2, COMP4702

Roy Portas, 43560846

April 23, 2017

Question 4.2

```
% Q2

a = randn(200, 2);
b = a + 4;
c = a;

c(:, 1) = 3 * c(:, 1);
c = c - 4;

d = [a; b];
e = [a; b; c];

% hold on;
% plot(a(:, 1), a(:, 2), '+');
% plot(b(:, 1), b(:, 2), 'o');
% plot(c(:, 1), c(:, 2), '*');

% Use the first dataset
data = e;


figure;
subplot(3, 2, 1);
hold on;
% ksdensity(data, 'PlotFcn', 'contour');
plot(a(:, 1), a(:, 2), '+');
plot(b(:, 1), b(:, 2), 'o');
plot(c(:, 1), c(:, 2), '*');

title('Contours Overlay');

% Calculate the grid
[f, xi] = ksdensity(data); x = linspace(min(xi(:, 1)),max(xi(:, 1)));
y = linspace(min(xi(:, 2)),max(xi(:, 2)));
[xq,yq] = meshgrid(x,y);
z = griddata(xi(:, 1),xi(:, 2),f,xq,yq);

% We now have x, y, z that can be used to get the gradient at any point

contour(x, y, z);
```

```
xlim([-10, 10]);
ylim([-10, 10]);
hold off;

copy = data;
new_points = copy;

for i = 1:5
    new_points = step(new_points, x, y, z);
    subplot(3, 2, i + 1);
    hold on;
    % ksdensity(data, 'PlotFcn', 'contour');
    scatter(new_points(:, 1), new_points(:, 2));
    title(sprintf('Step %d', i));
    xlim([-10, 10]);
    ylim([-10, 10]);
    hold off;

end


function dist = euclid_distance(point1, point2)
    dist = sqrt((point1(1) - point2(1))^2 + (point1(2) - point2(2)
       )^2);
end

function new_points = step(points, x, y, z)
    % Calibration factor (lambda)
    max_distance = 1.8;

    new_points = zeros(length(points), 2);
    for i = 1:length(points)
        point = points(i, :);

        within_range = zeros(length(points), 1);

        numerator = 0;
        denominator = 0;

        for j = 1:length(points)
            other_point = points(j, :);
            if euclid_distance(point, other_point) < max_distance
                within_range(j) = 1;
                % weight = interp2(x, y, z, other_point(1),
                   other_point(2));
```

```matlab

                % Calculate part of sum
                distance = euclid_distance(point, other_point);
                numerator = numerator + (distance * other_point);
                denominator = denominator + distance;
            end
        end

        mx = numerator / denominator;
        new_points(i, :) = mx(1, :);
    end
end
```

Question 4.3

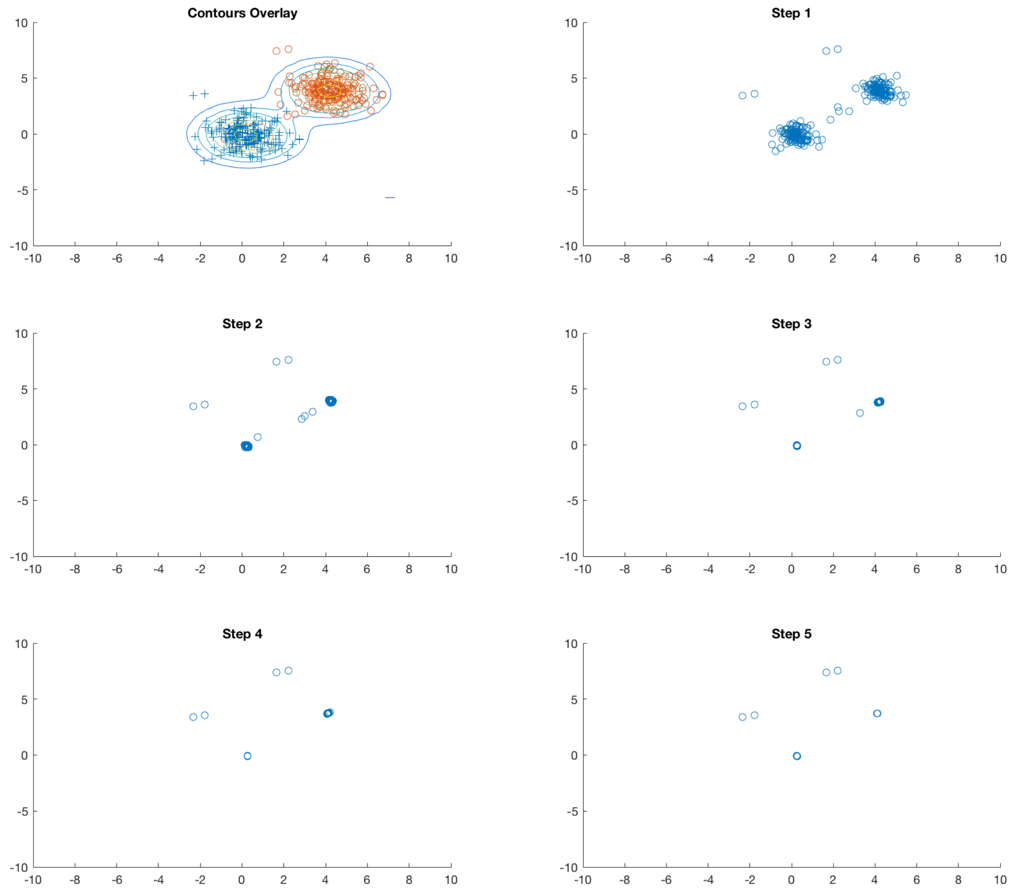

Figure 1: 2 Classes, Lambda = 1.8

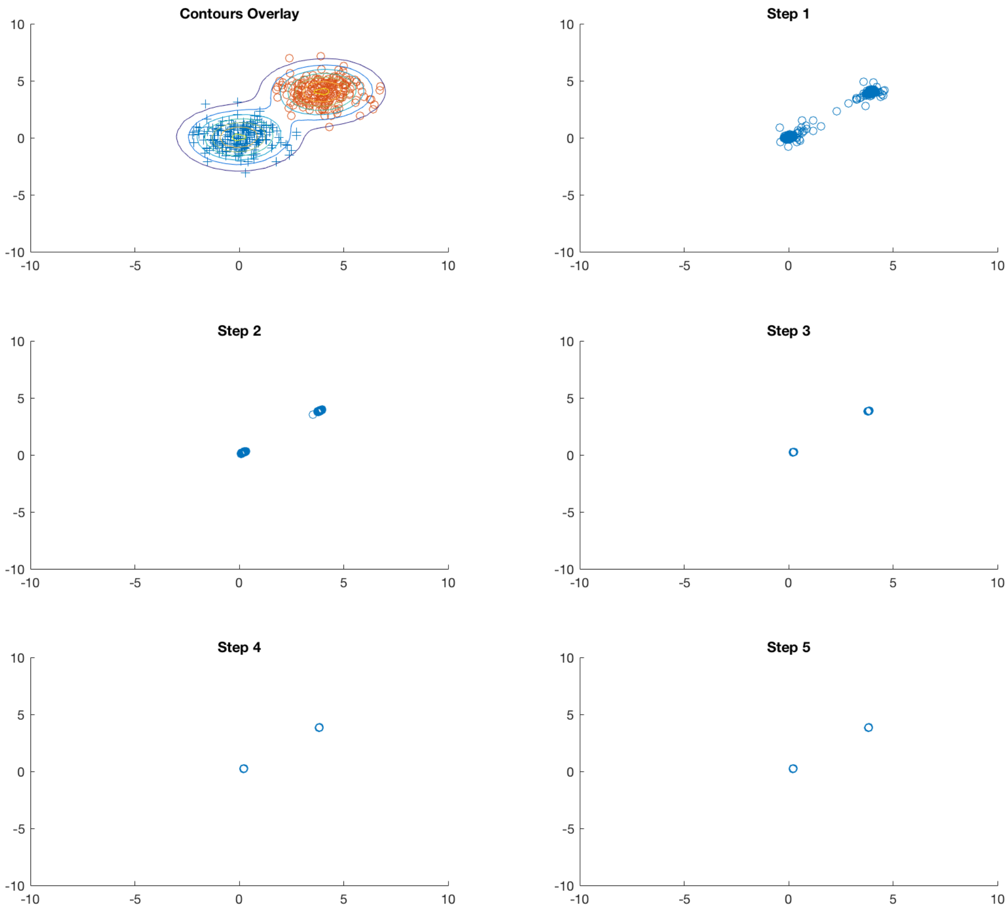

Figure 2: 2 Classes, Lambda = 3

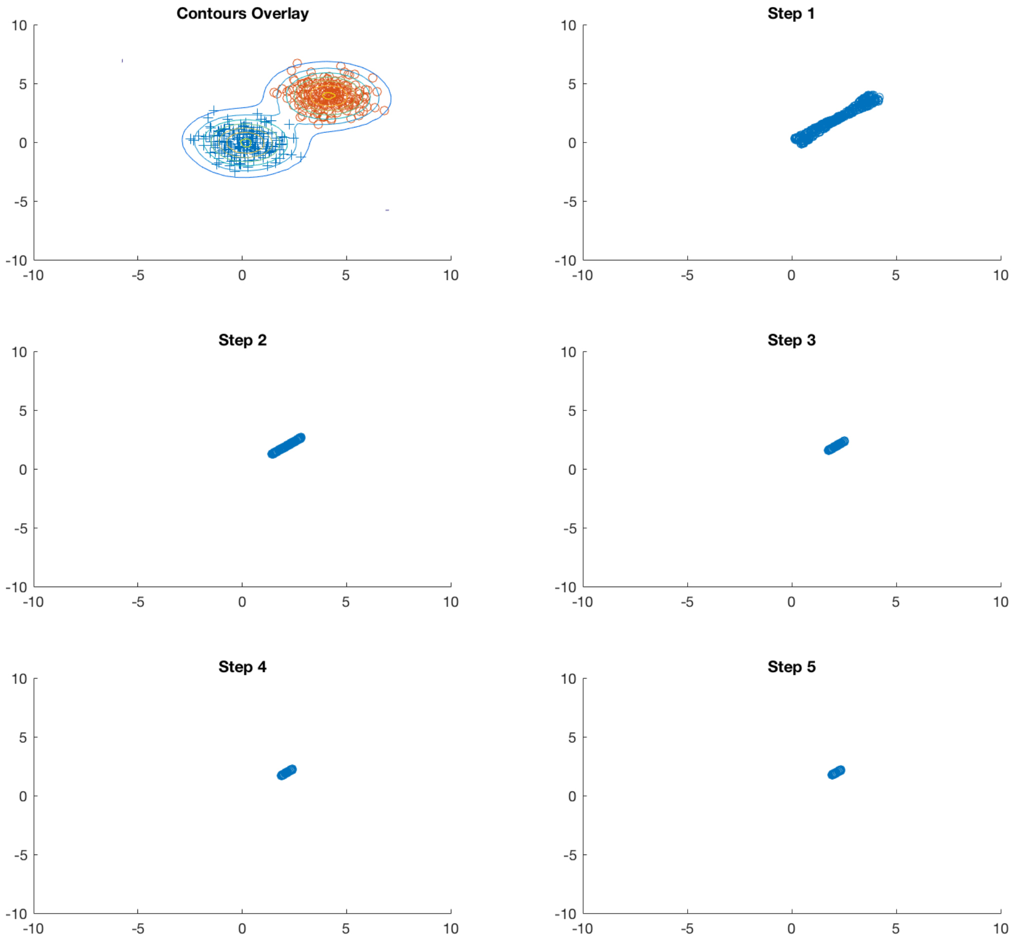

Figure 3: 2 Classes, Lambda = 5

For the 2 class problem, a lambda value set to 3 provided the best result. A lambda of 1.8 cause a few outliers not to shift towards the mean value. Whereas a lambda value of 5 caused all the points to shift towards one of the means, which is not desired. A lambda value of 3 shifted all the points to the two mean values without leaving outliers, thus is the best lambda value out of the three.

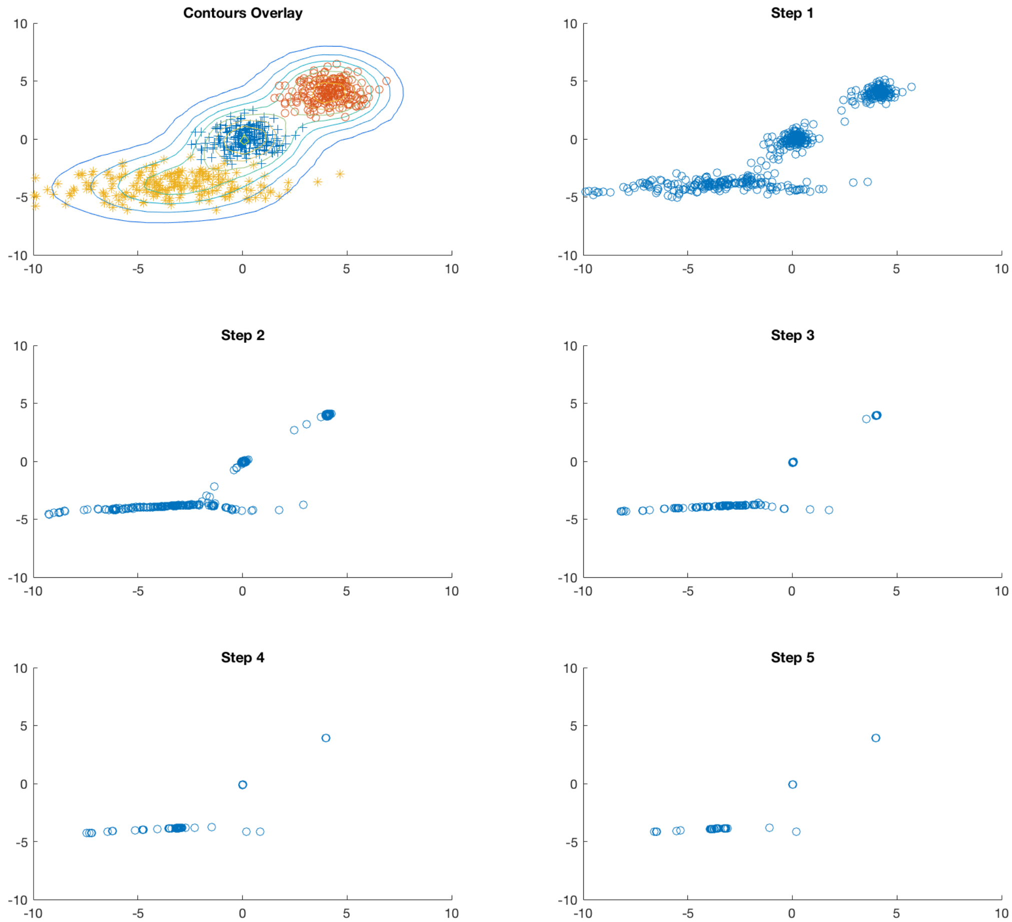

Figure 4: 3 Classes, Lambda = 1.8

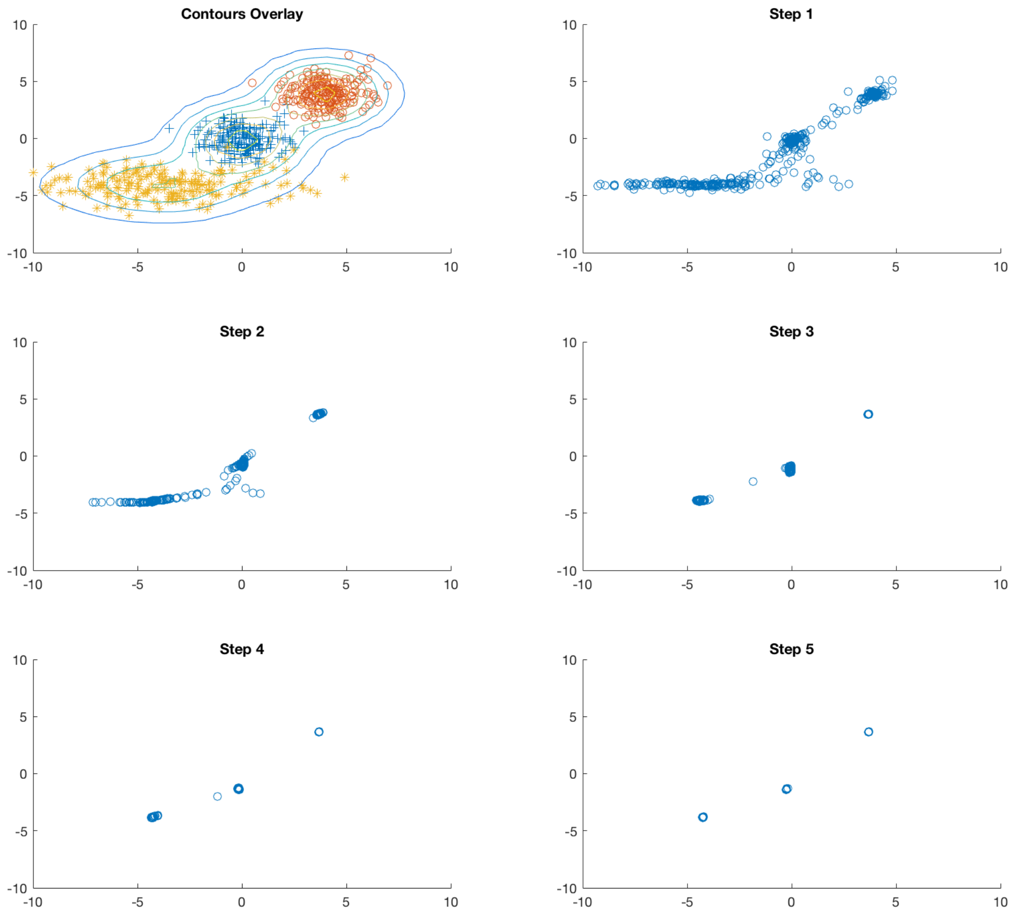

Figure 5: 3 Classes, Lambda = 3

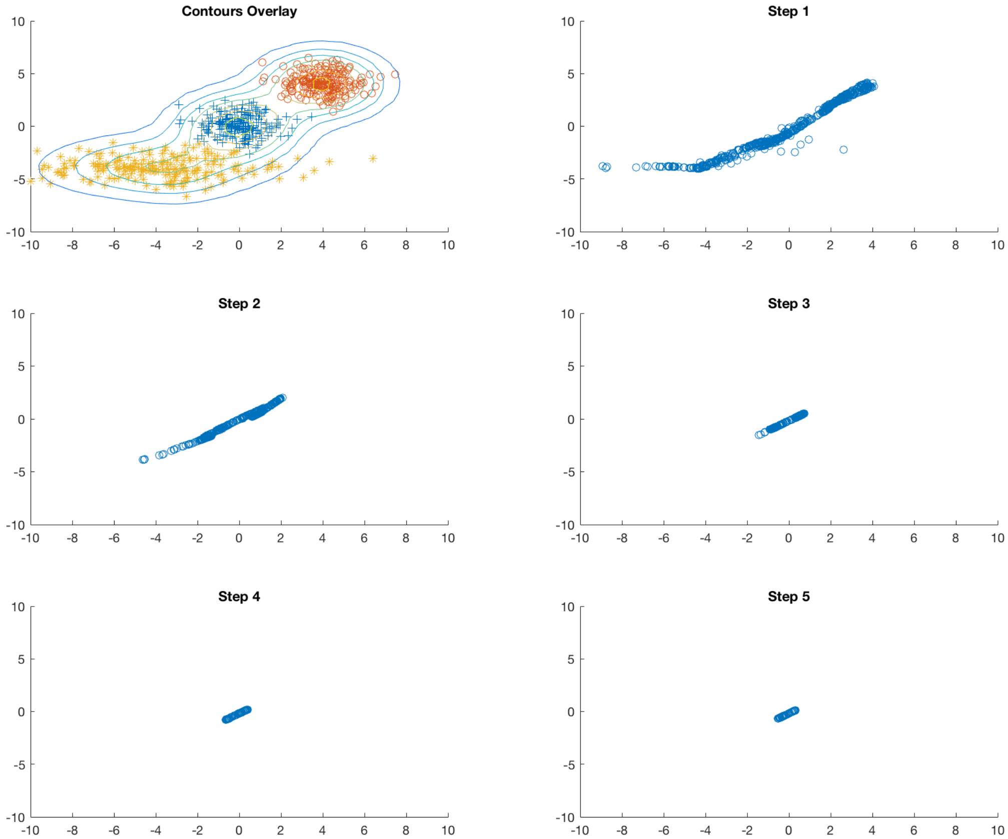

Figure 6: 3 Classes, Lambda = 5

The 3 class problem showed similar results with the same lambda values 1.8, 3 and 5. Again the 1.8 lambda value failed to shift some outliers towards the mean and the 5 lambda value shifted all of them towards a single point. The lambda value 3 correctly shifted all the points to their respective means, thus the lambda value of 3 was the best choice out of the three numbers.

Question 5.1

```
%
% Principle Component Analysis
%

function result = pca(data)
    m = mean(data);
```

```
    S = cov(data - m);
    [evec, eval] = eigs(S);

    % Sort the eigenvalues
    [y, i] = sort(diag(eval), 'descend');
    % Sort the eigenvectors columns by the eigenvalue indexes
    evec = evec(:, i);

    % PCA only works if you subtract the mean
    result = evec' * (data - m)';
end
```

Question 5.2

a

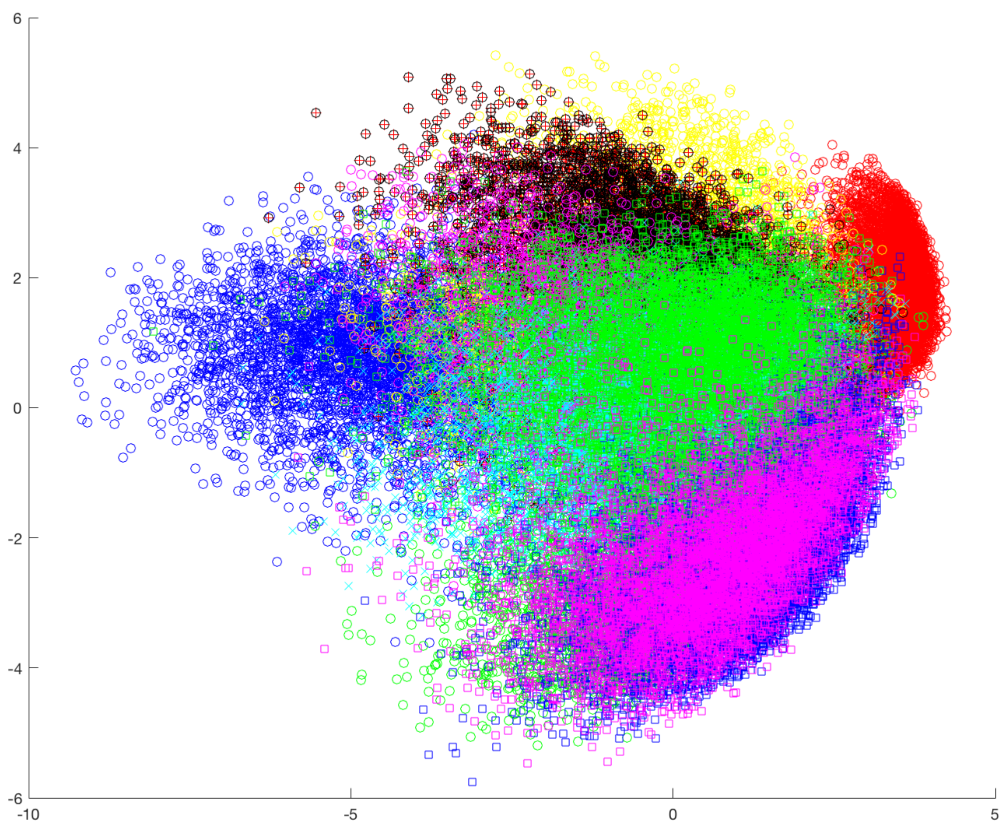

Figure 7: PCA on MNIST dataset

b

The first principle component accounts for 5.116% of the data, whereas the second principle component accounts for 3.7414% of the data.

c

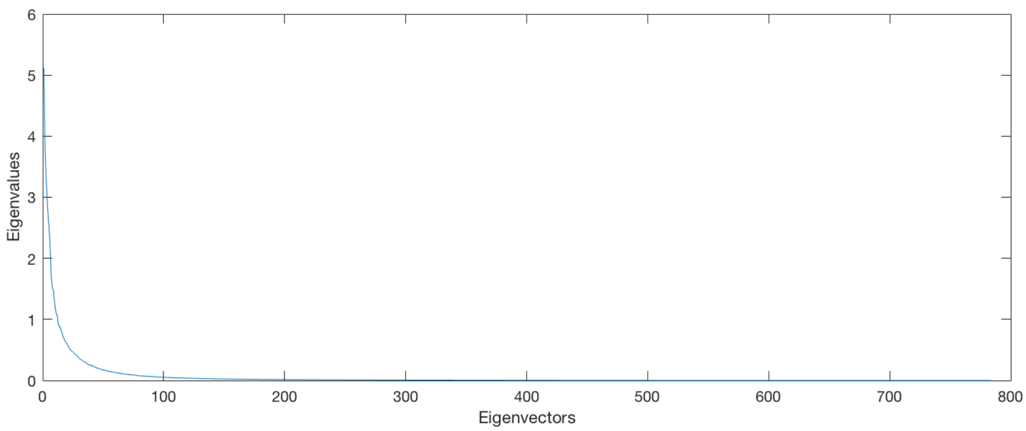

Figure 8: Scree Graph

Question 5.6

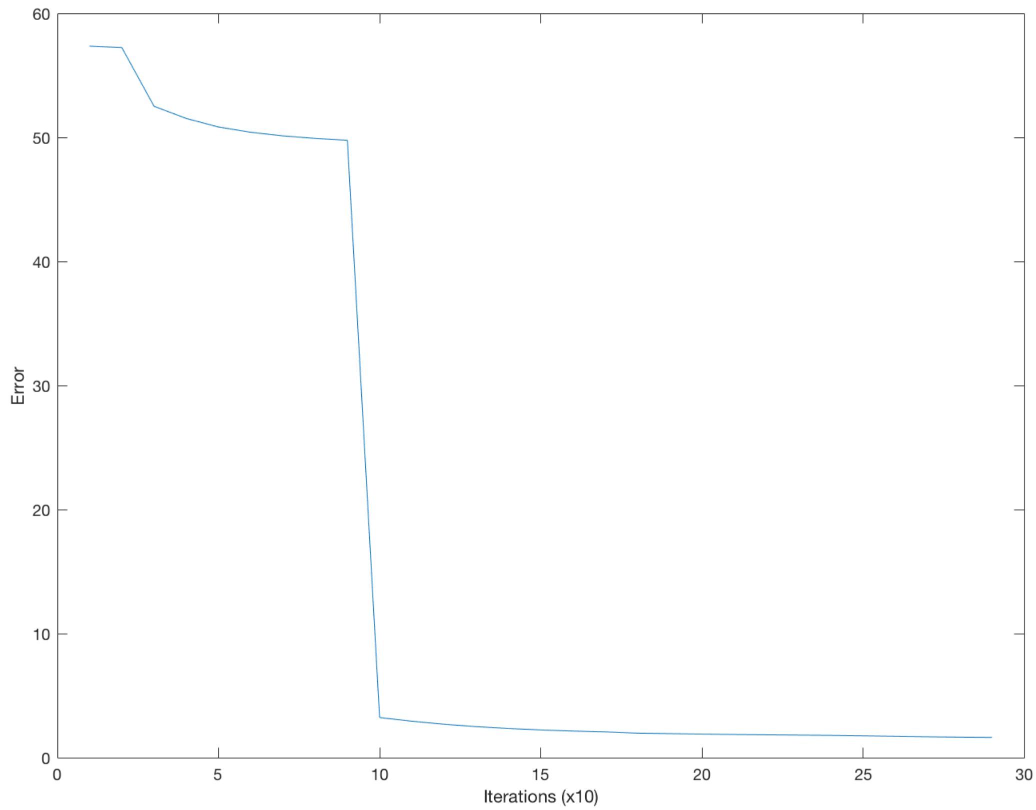

Figure 9: Error vs Iteration

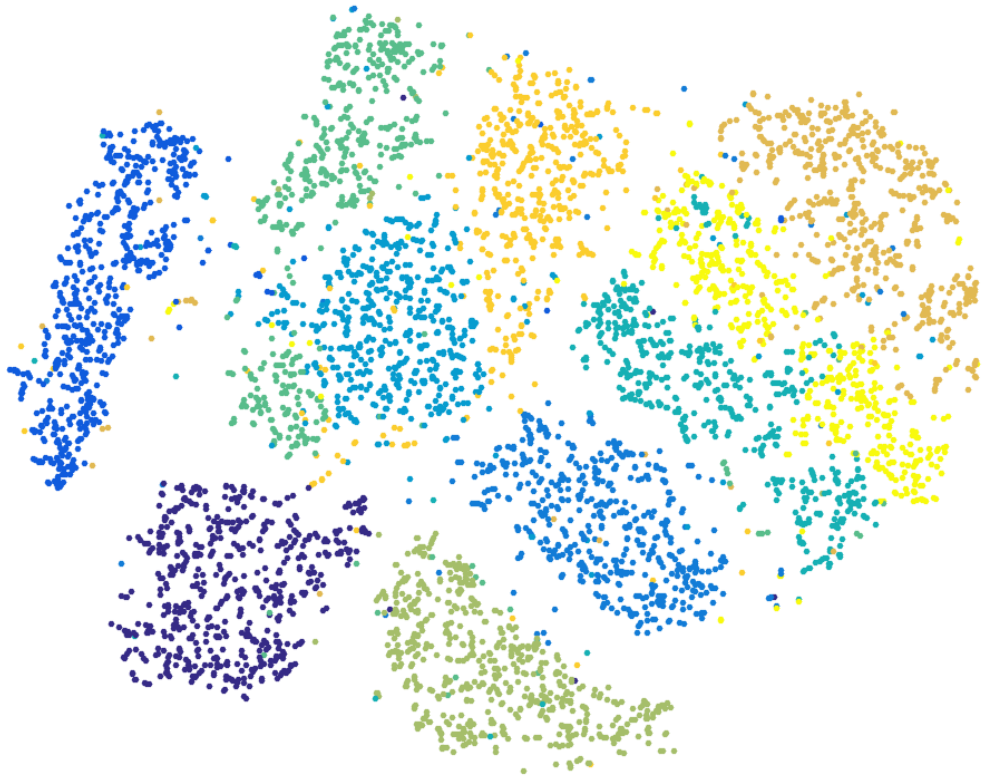

Figure 10: Iteration 300

Question 5.8

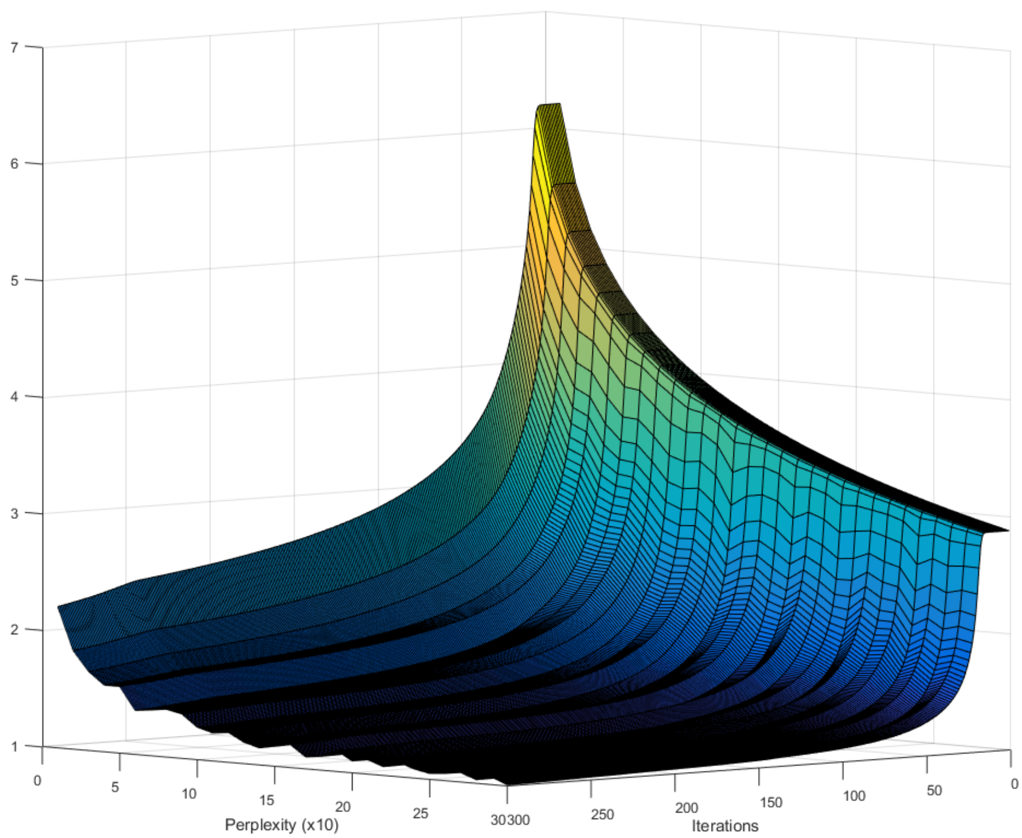

Figure 11: Perplexity and Iterations on Cost

The perplexity determines how to balance the attention between local and global aspects of the dataset. The graph shows the error change on iterations, thus we should choose a perplexity the reduces the error as quickly as possible, such that the algorithm requires less iterations to find a good solution. A simple heuristic to choose a good perplexity would be to look for the greatest rate of change in the error, by inspecting the chart we can see that the lower perplexities have far greater rate of change. Further inspection showed that a perplexity value of 2 provided the best rate of change.

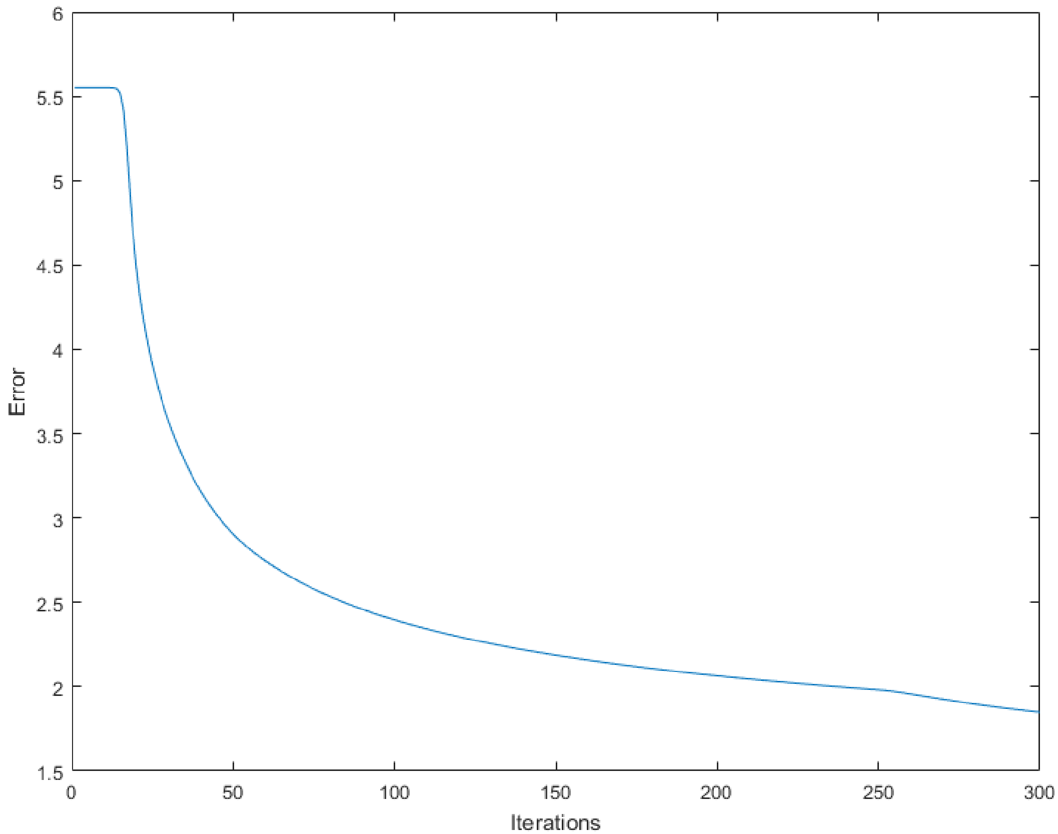

Figure 12: Error on Perplexity