Leren homework # 6

Date: December 14, 2014

Name (Student Number):
M. Pfundstein (10452397)
T.M. Meijers (10647023)

Question 1

Data: $[1, 2, 3, 3, 4, 5, 5, 7, 10, 11, 13, 14, 15, 17, 20, 21]$

Initial centroids: $[1, 3, 8]$

$\mu = 9.4375, \quad \sigma = 2$

K-means
Clusters:
$S_1 : \{1, 2\}$
$S_2 : \{3, 3, 4, 5, 5\}$
$S_3 : \{7, 10, 11, 13, 14, 15, 17, 20, 21\}$

Initial cost

$$\begin{aligned} J(S) = &((1-1)^2 + (2-1)^2) + ((3-3)^2 + (3-3)^2 + (4-3)^2 + (5-3)^2 + (5-3)^2) + \\ &((7-8)^2 + (10-8)^2 + (11-8)^2 + (13-8)^2 + (14-8)^2 + (15-8)^2 + (17-8)^2 + \\ &(20-8)^2 + (21-8)^2) \\ J(S) = &\ 528 \end{aligned}$$

One iteration step

New centroids:
$C_1 = mean(S_1) = 1.5$
$C_2 = mean(S_2) = 4$
$C_3 = mean(S_1) \approx 14.22$

New clusters:
$S_1 : \{1, 2\}$
$S_2 : \{3, 3, 4, 5, 5, 7\}$
$S_3 : \{10, 11, 13, 14, 15, 17, 20, 21\}$

New cost

$$\begin{aligned} J(S) = &((1-1.5)^2 + (2-1.5)^2) + ((3-4)^2 + (3-4)^2 + (4-4)^2 + (5-4)^2 + (5-4)^2 + \\ &(7-4)^2) + ((10-14.22)^2 + (11-14.22)^2 + (13-14.22)^2 + (14-14.22)^2 + \\ &(15-14.22)^2 + (17-14.22)^2 + (20-14.22)^2 + (21-14.22)^2) \\ J(S) = &\ 130.9272 \end{aligned}$$

Expectation Maximization
Remark: We used matlab for this, see **appendix A** for the code.
Initial μ's: $[1, 3, 8]$
σ's: $[2, 2, 2]$
Clusters:
$S_1 : \{1, 2\}$
$S_2 : \{3, 3, 4, 5, 5\}$
$S_3 : \{7, 10, 11, 13, 14, 15, 17, 20, 21\}$

Initial Likelihood:

$$\begin{aligned} E(X|\mu_1) = [&0.3959, 0.2812, 0.1836, 0.1836, 0.1034, 0.0435, 0.0435, 0.0026, \\ &0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000] \\ E(X|\mu_2) = [&0.6002, 0.7029, 0.7566, 0.7566, 0.7026, 0.4872, 0.4872, 0.0783, \\ &0.0020, 0.0006, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000] \\ E(X|\mu_3) = [&0.0039, 0.0159, 0.0598, 0.0598, 0.1940, 0.4694, 0.4694, 0.9191, \\ &0.9980, 0.9994, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000] \end{aligned}$$

One iteration step
New μ's: $[2.3653, 3.2362, 13.0803]$
New σ's: $[1.2260, 1.3904, 5.1502]$

New likelihood:

$$\begin{aligned} E(X|\mu_1) = [&0.4441, 0.3755, 0.2729, 0.2729, 0.1623, 0.0711, 0.0711, 0.0013, \\ &0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000] \\ E(X|\mu_2) = [&0.4994, 0.5829, 0.6779, 0.6779, 0.7483, 0.7052, 0.7052, 0.0956, \\ &0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000] \\ E(X|\mu_3) = [&0.0565, 0.0416, 0.0492, 0.0492, 0.0894, 0.2238, 0.2238, 0.9030, \\ &1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000] \end{aligned}$$

New μ's: $[2.5142, 3.5051, 13.5269]$
New σ's: $[1.2255, 1.4081, 4.9459]$

Question 2

(a) A 'leertaak' for which our current methods are not really suited for is time series prediction.

(b) This is because our current algorithms don't have a *memory*. They can just predict for a snapshot of the data. As soon as we would try to take into account changes as a function of time, we would have a tremendous increase in dimensionality and it would become intractible. It is also not directly possible to use time as a feature vector as we don't have a good concept yet to represent time as a feature. Time isn't discriminative.

(c) A possible approximation would be to use a recurrent neural network. This can be trained by gradient descent and has a limited working memory. Another approach would be to not take into account the whole time series but only the last k-steps. We could then predict for each time step independently and use the output of that as input for another predictor. Or just plot the predictions and see if we can see a trend. For instance, if the predicition of a house price goes up during time-steps k-2, k-1, k, then we can eventually predict that it will also go up at k+1.

(d) This is difficult to answer. Maybe the Neural Turing Machine (`http://arxiv.org/pdf/1410.5401v2.pdf`) developed by Deepmind would be able to handle such a task. They can store data and retrieve it. Hence it has a working memory available that theoretically can be arbitrarily big. Other possible solutions would be statistical and mathematical methods like 'moving average', 'curve fitting' etc.. (See full list here: `http://www.ipredict.it/ForecastingMethods.aspx`)

Appendix A

```
% Examples used from:
% https://chrisjmccormick.wordpress.com/2014/08/04/
  gaussian-mixture-models-tutorial-and-matlab-code/

xs = [1, 2, 3, 3, 4, 5, 5, 7, 10, 11, 13, 14, 15, 17, 20, 21];
mus = [1, 3, 8];
sigmas = [2 2 2];
c1 = [1, 2];
c2 = [3,3,4,5,5];
c3 = [7,10,11,13,14,15,17,20,21];
cprob = [length(c1) / length(xs), length(c2) / length(xs), ...
        length(c3) / length(xs)];
E = zeros(length(mus), length(xs));
g = zeros(1, length(mus));

for k = 1:2
    % Calculate expectations
    for i = 1:length(xs)
        for j = 1:length(mus)
            % Gaussian prob
            g(j) = (1 / (sigmas(j) * sqrt(2 * pi))) * ...
                exp((-(xs(i) - mus(j))^2) / (2 * sigmas(j)^2));
            E(j, i) = g(j) * cprob(j);
        end
        sum_prob = sum(E(:,i));
        % Weighted expectation
        for j = 1:length(mus)
            E(j, i) = E(j, i) / sum_prob;
        end
    end
    % Calculate maximizations
    phis = zeros(1, length(mus));
    for j = 1:length(mus)
        % Get mean of expectations for each cluster
        phis(j) = mean(E(j,:));
        % Calc new mean
        mus(j) = E(j,:) * xs';
        mus(j) = mus(j) ./ sum(E(j,:));
        % Get variance
        var = E(j,:) * (xs - mus(j)).^2';
        var = var ./ sum(E(j,:));
        % Calculate new sigmas
        sigmas(j) = sqrt(var);
    end
    display(E);
    display(mus);
    display(sigmas);
end
```