

Homework 3

Matt Dickenson
CS 527
Fall 2014

Problem 1

(a) kmeans.m

```
function [M, R] = kmeans(Z, M)
    % dim(Z) = d x I
    % dim(M) = d x K
    % dim(R) = K x I

    % initialize f(S) values for iteration
    f_p = intmax
    f_c = f_p - 1

    while f_p > f_c + sqrt(eps)
        R = closest_cluster(Z, M)
        M = (Z * transpose(R)) ./ (transpose(sum(R, 2) * ones(1, 2)))
        f_p = f_c
        f_c = sum(sum(dists(Z, M) .* R, 1), 2)
    end
end

function [assignments] = closest_cluster(Z, mu)
    % partition the points in Z by their closest mean
    K = size(mu, 2)
    all_dists = dists(Z, mu)
    min_dists = ones(K, 1) * min(all_dists, [], 1)
    assignments = all_dists == min_dists
end

function [all_dists] = dists(Z, mu)
    % compute distance of each point in Z from each centroid
    K = size(mu, 2)
    I = size(Z, 2)
    all_dists = zeros(K, I)
    for k=1:K
        means = mu(:, k) * ones(1, I)
        dists = sqrt(sum((Z - means).^2, 1))
        all_dists(k, :) = dists
    end
end
```

end

Output:

M =

0.6816	0.8271	-1.0890
-0.6389	1.1812	-0.6784

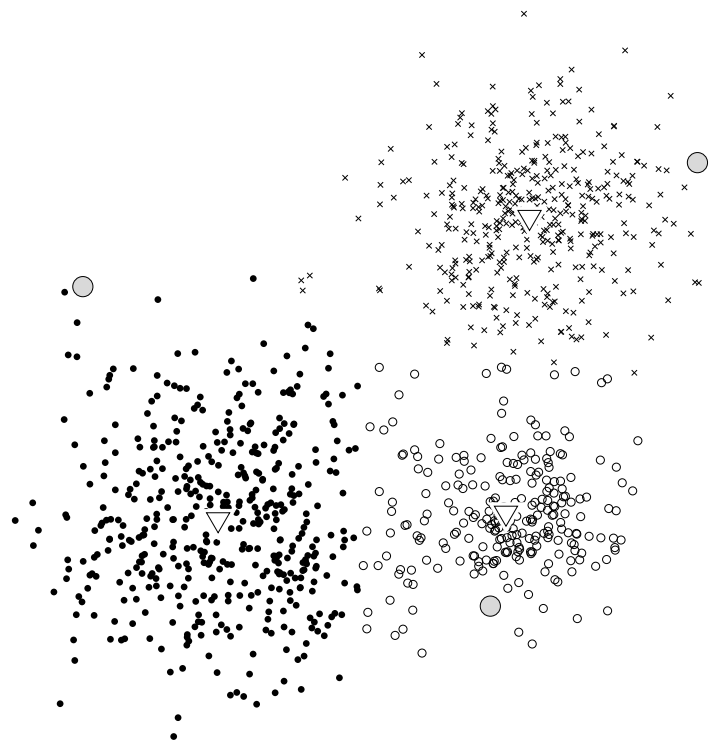


Figure 1: Result of K-Means with blobs data for 1(a)

(b) Output:

M =

-1.0371	1.0207
-0.0176	0.0173

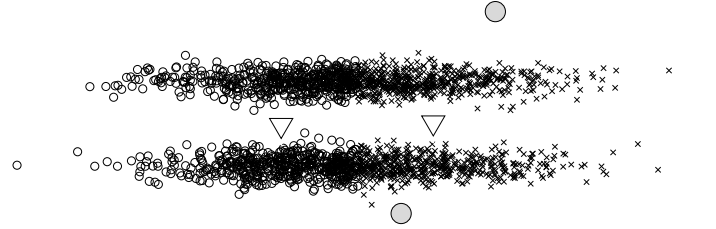


Figure 2: Result of K-Means with cigars data for 1(b)

(c) Output:

M =

-1.0402	1.0175
-0.0199	0.0195

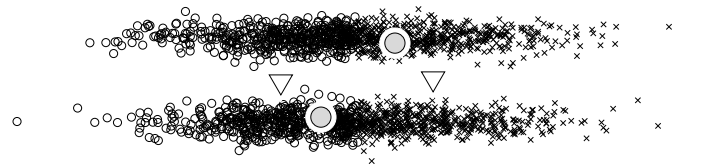


Figure 3: Result of K-Means with cigars data for 1(c)

(d) The shortcomings are not that the K-means algorithm computes a local minimum. The problem is that both dimensions are treated equally for the purposes of computing the error rate despite the fact that they may have very different variances.

The problem is that the data points vary much more along the x-dimension than the y dimension. Although they appear to be well-described by two clusters with the same x range, one above the other, the high variance in the x dimension would force two such centroids to still have a high error measure $f(S)$.

(e) I constructed four clusters whose points form approximately the outline of a unit square. The four initial centroids m all lie at the origin. The points and centroids are displayed in Figure 4 below. Note that the final centroids and initial centroids are exactly equal.



Figure 4: Result of K-Means with pathological input

The true centroids are $\mu_1^* = (0, 1)$, $\mu_2^* = (1, 0)$, $\mu_3^* = (0, -1)$, $\mu_4^* = (-1, 0)$. The initial (and final) centroids are $m_1 = m_2 = m_3 = m_4 = (0, 0)$.

The points in each cluster are

$$\begin{aligned}
S_1 &= \{(\epsilon, 1), (1 + \epsilon, 0), (-\epsilon, -1), (-1 - \epsilon, 0)\} \\
S_2 &= \{(-\epsilon, 1), (1 - \epsilon, 0), (\epsilon, -1), (-1 + \epsilon, 0)\} \\
S_3 &= \{(0, 1 + \epsilon), (1, \epsilon), (0, -1 - \epsilon), (1, -\epsilon)\} \\
S_4 &= \{(0, 1 - \epsilon), (1, -\epsilon), (0, -1 + \epsilon), (-1, \epsilon)\},
\end{aligned}$$

although any random, symmetric perturbation about the true centroids would work in expectation.

The algorithm makes no progress because the initial centroids m are each symmetrically located between all four clusters and all clusters are equally sized. Moving any centroid nearer to one cluster would make it closer to a few points but farther away from all other points, cancelling out the advantage of relocating the centroid.

(f) The previous answer shows that the success K -means algorithm depends upon the initial centroids. K -means also depends on knowing the true number of clusters. It can also fail when clusters are symmetric and of equal size.

Problem 2

(a) EM.m

```

function [lambda, M, Sigma, R] = EM(Z, M, sigma2)
[d, I] = size(Z)
K = size(M, 2)

% Initialize uninformative prior
lambda = repmat(1/K, K, 1)
Sigma = zeros(d, d, K)
for k=1:K
    Sigma(:, :, k) = sigma2(k)*eye(d)
end
R = zeros(K, I)

% initialize f(theta) values for iteration
f_p = intmax;
f_c = f_p - 1;

while f_p > f_c + sqrt(eps)
    % E-step
    for k=1:K
        R(k, :) = lambda(k) * mvnpdf(Z', M(:, k)', Sigma(:, :, k))
    end
    f_c = f_p;
    [f_p, M] = M_step(Z, R, Sigma);
end

```

```

end
R = R ./ (ones(K, 1) * sum(R));

% M-step
for k=1:K
    lambda(k) = mean(R(k, :));
    r_d = ones(d, 1) * R(k, :);
    M(:, k) = sum( r_d .* Z, 2 ) / sum(R(k, :));
    diff = Z - (M(:, k) * ones(1, I));
    Sigma(:, :, k) = ((r_d .* diff) * diff') / sum(R(k, :));
end

f_p = f_c;
f_c = -sum(log(sum(R, 2)));
end
end

```

Output:

```

lambda =
    0.3031
    0.3638
    0.3331

M =
    0.2733    0.8352   -1.1607
   -0.7696    1.1658   -0.5729

Sigma(:, :, 1) =
    0.5661    0.1155
    0.1155    0.1176

Sigma(:, :, 2) =
    0.1578    0.0080
    0.0080    0.1610

Sigma(:, :, 3) =
    0.2156    0.0474
    0.0474    0.2841

```

(b) For the cigars data, the E-M result is:

```

lambda =
    0.5000

```

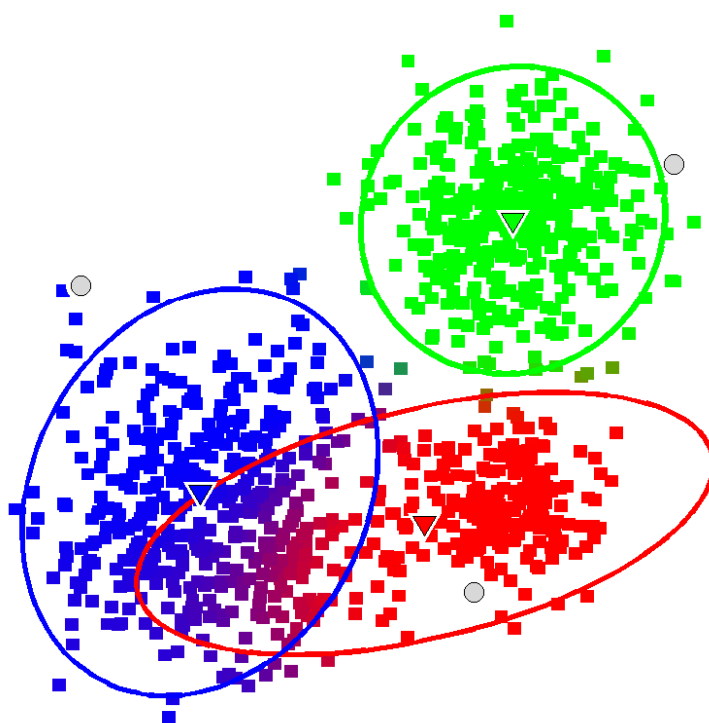


Figure 5: Result of EM with blobs data for 2(a)

```

0.5000

M =

-0.0333    0.0333
-0.5701    0.5701

Sigma(:,:,1) =
1.6468    -0.0001
-0.0001    0.0175

Sigma(:,:,2) =
1.6652    0.0007
0.0007    0.0162

```

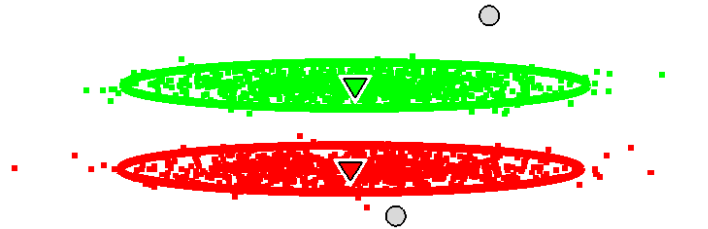


Figure 6: Result of EM with cigars data for 2(b)

(c) Yes, the result of E-M using the cigars data is much more satisfactory than using K -means. Unlike with K -means, the centroids are now each centered on one of the two clusters in the data. There is also little uncertainty about which cluster each point belongs to.

(d) For E-M on the blobs data with $k=2$ the result is:

With $k = 4$ the results are:

(e) For the bananas data, the E-M result is:

```

M =

-1.0771    1.0769
-0.6489    0.6488

```

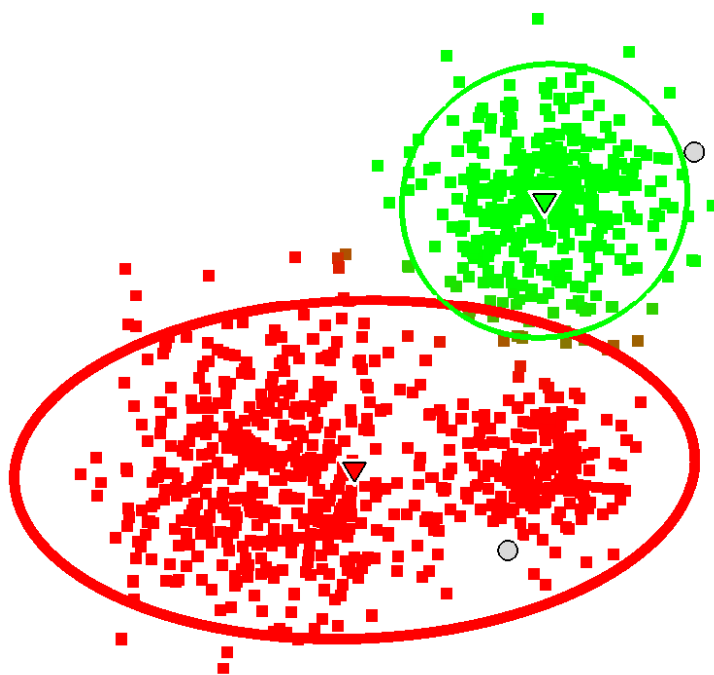



Figure 7: Result of EM with blobs data and $k=2$

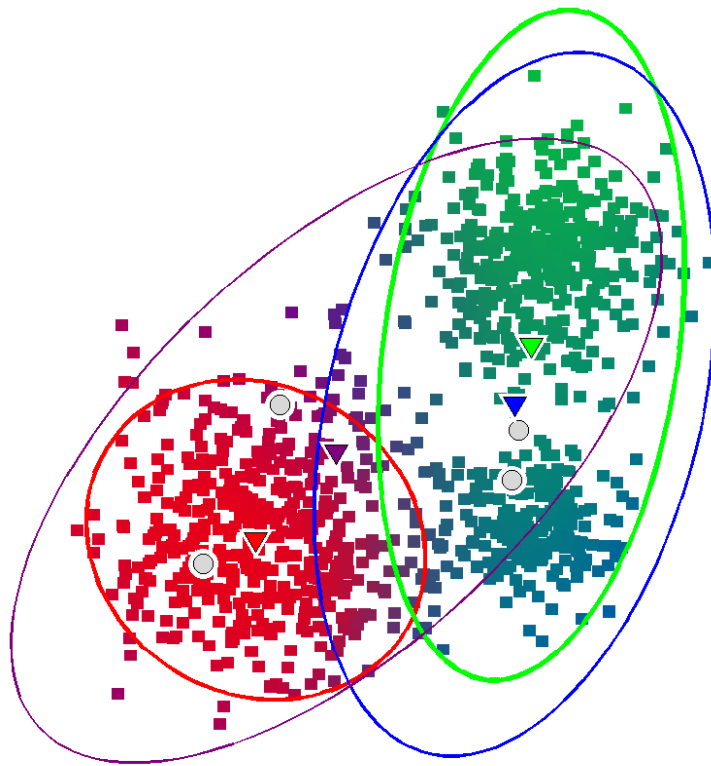


Figure 8: Result of EM with blobs data and $k=4$

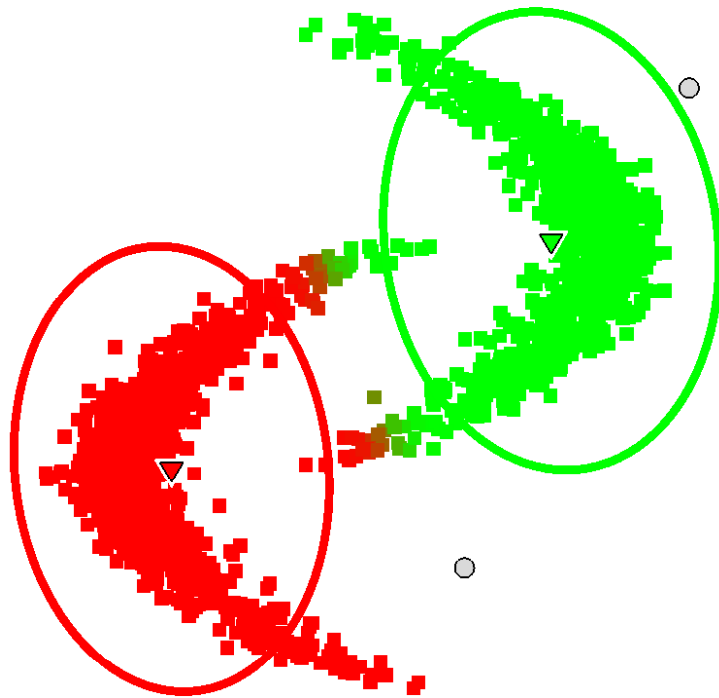


Figure 9: Result of EM with bananas data for 2(d)

(f) The result in 2(d) is unsatisfactory for two reasons. First, points at the top tip of the left banana and the bottom tip of the right banana are each misclassified as belonging to the other banana. This is a result of using a mixture of normal distributions, since the clusters are constrained to be ellipse-shaped.

Second, the ellipses defined by the mean and variance of each cluster exclude a number of points belonging to the cluster. This is a result of the E-M algorithm: this data could likely be well-described if $k = 6$, but it's difficult to know this a priori and k is fixed in this E-M implementation.

Problem 3

(a) `meanShift.m`:

```
function [z, zh] = meanShift(zstart, Z, h)
    [d, I] = size(Z);
    z_prime = zstart;
    zh = z_prime;
    terminate = false;
    while ~terminate
        z = z_prime;
        number = zeros(d, I);
        for i=1:I
            number(:, i) = Z(:, i) * kernel(z-Z(:, i), h);
        end
        z_prime = sum(number, 2) ./ sum(number ./ Z, 2);
        zh = [zh'; z_prime']';
        terminate = mynorm(z-z_prime) <= h/1000;
    end
end

function [d] = mynorm(x)
    d = sqrt(sum(x.^2));
end

function [k] = kernel(x, h)
    k = exp(-(mynorm(x)/h)^2);
end
```

Running this code as instructed produced the following plot:

The first run with $h = 0.2$ took 37 iterations and the second run with $h = 2$ took 14 iterations.

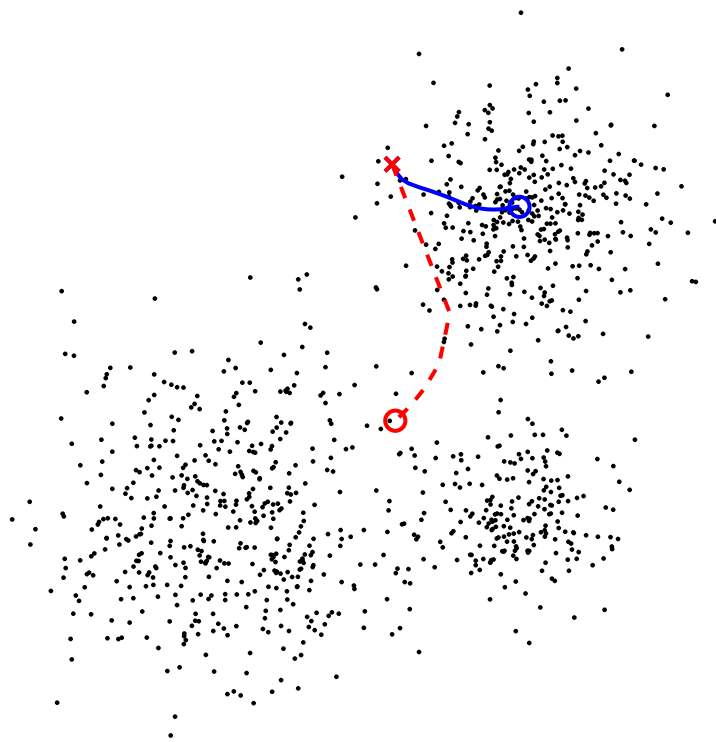


Figure 10: Result of Mean-Shift with blobs data

(b) Given the range of the data, by the time the threshold $h = 2$ our “local” neighborhood is global in the sense that it includes the entire sample. Increasing h beyond this level while maintaining the same starting point always leads us to approximately the same point—the global mean.