# Lab Assignment 4

Steven Truong

## Task 1

Assume that at a given step $n$, we have $\boldsymbol{\mu}^n$ and $r^n$.

In the first phase of minimizing $E(\boldsymbol{\mu}^n, r^n)$, for each $\boldsymbol{x}_i$ in our data set, we assign

$$r_{ik}^{n+1} = \begin{cases} 1 & \text{if } k = \arg\min_{1 \leq j \leq k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\| \\ 0 & \text{otherwise.} \end{cases}$$

This clearly causes $E$ to decrease (or stay the same) because we replace the previous $\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|$ term with the one that gives the minimum norm.

In the second phase, we have $E(\boldsymbol{\mu}^n, r^{n+1})$, and minimize with respect to $\boldsymbol{\mu}^n$. Since $E$ is convex as a function of $\boldsymbol{\mu}^n$, then $\boldsymbol{\mu}^{n+1}$, which gives a 0 gradient, must be the global minimizer with our given $r^{n+1}$. Hence, in this phase, $E$ must decrease (or stay the same) also.

Thus, the $k$-cluster means algorithm guarantees descent at each step.

## Task 2

In one step of the algorithm, we have to calculate all the norms between $\boldsymbol{x}_n$ and $\boldsymbol{\mu}_j$ in order to update $r$.

We first have to calculate the centroids, which are given by

$$\boldsymbol{\mu}_j = \frac{\displaystyle\sum_{n=1}^{N} r_{nj} \boldsymbol{x}_n}{\displaystyle\sum_{n=1}^{N} r_{nj}}.$$

So, for the $k$ classes, we have to do $N$ products, $2N$ sums, and a division, which gives us an upper bound of $k(3N+1)$ steps for the calculation of the means.

We then have to calculate all the norms $\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|$, which involves $d$ subtractions, multiplications, and sums for $Nk$ pairs of $\boldsymbol{x}_n$ and $\boldsymbol{\mu}_j$, which gives us $3Ndk$ calculations to figure out how to update $r_{nj}$.

So, each update has $k(3N+1) + 3Ndk$ calculations total.

## Task 3

Because of symmetry, if we had $\boldsymbol{\mu}$ which gave us a minimum, we can switch around the order of the $\boldsymbol{\mu}_j$ and have the same minimum, but with class labels changed. Convex functions must only have one minimum, so if $k$ is at least 2, then $J(\boldsymbol{\mu})$ is not convex.

# Task 4

```matlab
load iris_data.mat

[N, d] = size(data);
global k;
k = 3;  % Number of classes

%%%% Helper Functions

% Given r and j, this computes |Cj|
function y = clusterSize(r, j)
  y = sum(r(:, j) == 1);
end

% E step (calculating means)
function newMu = updateMu(data, r)
  global k;
  [N, d] = size(data);

  newMu = zeros(1, d);
  for i = 1:k
    currentSum = zeros(1, d);
    for j = 1:N
      if (r(j, i) == 1)
        currentSum = currentSum + data(j, :);
      end
    end
    cSize = clusterSize(r, i);
    newMu(i, :) = currentSum / (cSize + (cSize == 0) * eps); % Avoid division by 0
  end
end

% M step (updating r)
function newR = updateR(data, mu, r)
  global k;
  [N, d] = size(data);

  newR = r;
  % min always takes the first occurance of the minimum,
  % so we don't have to worry about infinite loops
  for i = 1:N
    norms = zeros(k, 1);
    for j = 1:k
      norms(j) = norm(data(i, :) - mu(j, :));
    end
    [M, I] = min(norms);
    newR(i, :) = zeros(1, k);
    newR(i, I) = 1;
  end
end

% Assign via random partition method (RPM)
function [mu, r] = rpm(data)
  global k;
  [N, d] = size(data);

  r = zeros(N, k); % The row is the data point, and the columns give its current classification
  for i = 1:N
    col = ceil(rand * k);
    r(i, col) = 1;
  end
  mu = updateMu(data, r); % updateMu just calculates the means, essentially
end

% Assign via the Forgy method
function [mu, r] = forgy(data)
  global k;
  [N, d] = size(data);

  r = zeros(N, k); % The row is the data point, and the columns give its current classification
  p = randperm(N, k);
  for i = 1:k
    mu(i, :) = data(p(i), :);
  end
end

%%%% Main Script

% Initialization
[mu, r] = forgy(data);
```

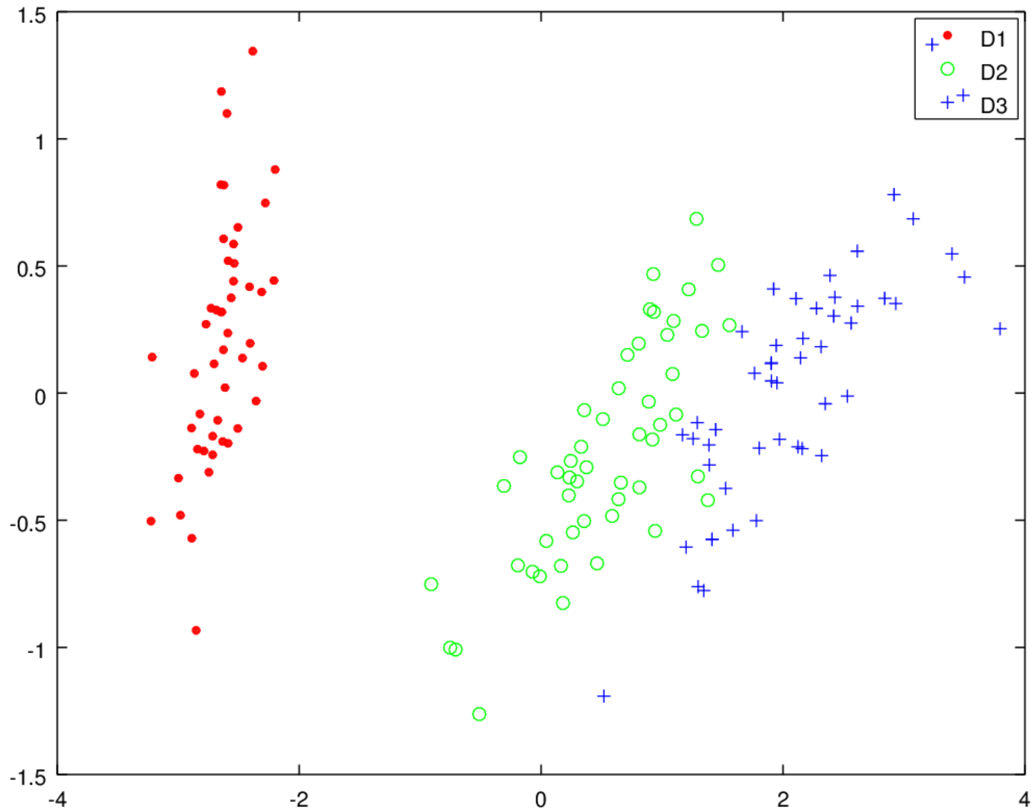
```
while true
  newR = updateR(data, mu, r); % E step
  mu = updateMu(data, newR);   % M step
  if (r == newR) % Check if points were moved to a new cluster
    break;
  else
    r = newR;
  end
end

classifications = zeros(N, 1);
for i = 1:N
  for j = 1:k
    if (r(i, j) == 1)
      classifications(i) = j;
    end
  end
end

mu
classifications
```

## Task 5

On average, the Forgy method performed slightly better. Each cluster's purity for the Forgy method was larger every purity for the random partition method.

| | | | |
|---|---|---|---|
| Forgy Method | Purity($C_1$) | Purity($C_2$) | Purity($C_1$) |
| | 0.88 | 0.88 | 0.87 |
| Random Parition Method | Purity($C_1$) | Purity($C_2$) | Purity($C_1$) |
| | 0.85 | 0.82 | 0.80 |

## Task 6




The performance of $k$-means clustering was fairly good because the classes roughly made up different clusters. However, class 2 and class 3 mixed quite a bit, so it was more difficult to separate those two classes when clustering the data points. So, 80–90% accuracy between the two methods on average is not surprising.