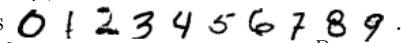


The objective of this lab is for you to program several representative clustering algorithms in Matlab, apply them to some datasets and observe their behavior. The TA will first demonstrate the results of the algorithms on a toy dataset and the MNIST dataset, and then you will program them, replicate those results, and further explore the datasets with the algorithms. You can use the textbook, lecture notes and your own notes.

I Datasets

Construct your own toy datasets in 2D, such as Gaussian clusters with more or less overlap, or clusters with curved shapes as in the 2moons dataset. You will also use the MNIST dataset of handwritten digits . Since clustering algorithms are unsupervised, they do not use the class labels $y_n \in \{0, \dots, 9\}$, only the instances $\mathbf{x} \in \mathbb{R}^D$ (where $D = 784$). You can use the labels to see if they agree with the resulting clusters found by an algorithm.

II Implementing and using clustering algorithms

Implementation and plots Implement the following algorithms: k -means, EM for Gaussian mixtures with full covariances, mean-shift and connected-components. The figure shows pseudocode for the algorithms. For algorithms that take an infinite number of iterations to converge, stop them after `maxit` iterations (e.g. 100) or once the error function changes by less than a small value `tol` (e.g. 10^{-3}). With toy datasets in 2D, plot the following figures:

- For every algorithm: the dataset in 2D, with points colored according to the cluster they belong to.
- For k -means: the value of the error function after each iteration; it should decrease monotonically and stop in a finite number of iterations.
- For EM with Gaussian mixtures: the value of the log-likelihood function after each iteration; it should increase monotonically. To get a hard clustering, assign point \mathbf{x}_n to cluster k if $p(k|\mathbf{x}_n) > p(j|\mathbf{x}_n) \forall j \neq k$. To get a soft clustering (which is more informative), plot $p(k|\mathbf{x})$ itself for each cluster as a function of $\mathbf{x} \in \mathbb{R}^2$ (as a color plot, or as a contour plot for each cluster).
- For mean-shift: the contours of the kernel density estimate and its modes; and, for any given point \mathbf{x}_n , the value of the density $p(\mathbf{x})$ after each mean-shift iteration (initialized from \mathbf{x}_n), which should increase monotonically.
- For connected-components: the dataset in 2D, with points connected by edges in the ϵ -ball graph.

With the MNIST dataset, try EM with Gaussian mixtures (also k -means) with different K values and plot:

1. The mean $\boldsymbol{\mu}_k$ of each cluster $k = 1, \dots, K$, as a grayscale image, with its mixing proportion $\pi_k = p(k)$.
2. The posterior probabilities $p(k|\mathbf{x}_n)$ for $k = 1, \dots, K$ for a given digit image \mathbf{x}_n , plotted as a bar chart.

Exploration Explore each algorithm in different settings. First, using the same dataset:

- Try different values of the user parameter (number of clusters K for k -means and Gaussian mixtures with EM, bandwidth $\sigma > 0$ for mean-shift, scale $\epsilon > 0$ for connected-components).
- For algorithms that depend on the initialization (k -means and EM), try different random initializations.

Then, explore your algorithms and plots with different datasets, number of clusters, clusters with more or less overlap, with different shapes, etc. See the end of file [lab04.m](#) for suggestions of things to explore.

Advice

- Keep your code simple and vectorized. It should look as close as possible to the pseudocode in the figure.
- Some of these algorithms may be slow (in particular, EM and mean-shift). Use small datasets to get results fast, and try to vectorize your Matlab code.
- For EM with Gaussian mixtures, add a small number to the diagonal of each covariance matrix $\boldsymbol{\Sigma}_k$ (e.g. $10^{-10} \text{tr}(\boldsymbol{\Sigma}_k)/D$) to make $\boldsymbol{\Sigma}_k$ be full rank. Do this right after updating $\boldsymbol{\Sigma}_k$ in the M step as in eq. (7.13).
- Useful Matlab functions (among others): `mean cov find randn linspace scatter contour gplot bar`.

III What you have to submit

Follow these instructions strictly. Email the TA the following packed into a **single** file (lab04.tar.gz or lab04.zip) and with email subject [CSE176] lab04:

- Matlab code for the functions `k_means.m`, `gm_EM.m`, `mean_shift.m` and `cc_eball.m`, using the templates provided. Read carefully the templates to understand what each function should do; the functions should work when called from `lab04.m`. For EM, look into `GMpdf.m` (from the [Gaussian mixture tools](#)) to understand the structure we use to store a Gaussian mixture.
Note: for each function, you should write your own code based on the pseudocode of the figure. You are not allowed to use any functions from the Matlab Toolboxes (in particular, the Statistics and Machine Learning Toolbox, or the Neural Network Toolbox). You can only use basic Matlab functions.
- A brief report (2 pages) in PDF format describing your experience with 3 datasets: 2moons, MNIST and one other dataset of your choice.
The more extensive and insightful your exploration, the higher the grade. Be concise. Don't include code or figures, we can recreate them by running your functions. Indicate the part that each member of the group did.

k-MEANS ALGORITHM

```

{μk}k=1K ← K random points from {xn}n=1N
repeat
  for n ∈ {1, ..., N}
    k* = arg mink=1, ..., K ||xn - μk||           closest mean
                                                    to xn
    znk* = 1 and znk = 0 ∀k ≠ k*
  for k ∈ {1, ..., K}
    μk ← ∑n=1N znkxn / ∑n=1N znk           mean of points
                                                    in cluster k
until stop
return {μk}k=1K, Z

```

GAUSSIAN MIXTURE ESTIMATED WITH EM ALGORITHM

```

Initialize {πk, μk, Σk}k=1K from k-means
repeat
  for n ∈ {1, ..., N}
    znk ← p(k|xn) = eq. (7.14)                       E step
  for k ∈ {1, ..., K}
    πk ← eq. (7.11), μk, Σk ← eq. (7.13)           M step
until stop
return {πk, μk, Σk}k=1K

```

GAUSSIAN MEAN-SHIFT ALGORITHM

```

for n ∈ {1, ..., N}
  x ← xn
  repeat
    ∀n: p(n|x) ←  $\frac{\exp(-\frac{1}{2}\|x-x_n\|/\sigma)^2}{\sum_{n'=1}^N \exp(-\frac{1}{2}\|x-x_{n'}\|/\sigma)^2}$ 
    x ← ∑n=1N p(n|x)xn
  until stop
  zn ← x                                           mode found from xn
end
return connected-components({zn}n=1N, ε)         aggregate
                                                    modes found

```

CONNECTED-COMPONENTS ALGORITHM (ε-BALL GRAPH)

```

Define an ε-ball graph:
  • vertices x1, ..., xN
  • edges (xn, xm) ⇔ d(xn, xm) < ε,
    ∀n, m = 1, ..., N.
Apply DFS to this graph.
return its connected components

```

Figure 1: Pseudocode for k -means, EM for Gaussian mixtures, mean-shift for the Gaussian kernel and connected-components for an ϵ -ball graph (using a distance function $d(\cdot, \cdot)$). In all cases, the input is a dataset $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$ and a user parameter: number of clusters K for k -means and Gaussian mixtures with EM, bandwidth $\sigma > 0$ for mean-shift, and scale $\epsilon > 0$ for connected-components. Equation numbers refer to the textbook.