CS181-S18

Assignment #4

Due: 11:59pm March 30, 2018

Collaborators:

Homework 4: Clustering and EM

This homework assignment focuses on different unsupervised learning methods from a theoretical and practical standpoint. In Problem 1, you will explore Hierarchical Clustering and experiment with how the choice of distance metrics can alter the behavior of the algorithm. In Problem 2, you will derive from scratch the full expectation-maximization algorithm for fitting a Gaussian mixture model. In Problem 3, you will implement K-Means clustering on a dataset of handwritten images and analyze the latent structure learned by this algorithm.

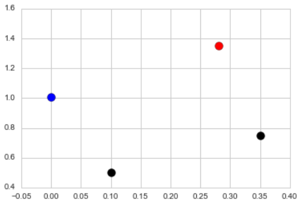
There is a mathematical component and a programming component to this homework. Please submit your PDF and Python files to Canvas, and push all of your work to your GitHub repository. If a question requires you to make any plots, please include those in the writeup.

Hierarchical Clustering [7 pts]

At each step of hierarchical clustering, the two most similar clusters are merged together. This step is repeated until there is one single group. We saw in class that hierarchical clustering will return a different result based on the pointwise-distance and cluster-distance that is is used. In this problem you will examine different choices of pointwise distance (specified through choice of norm) and cluster distance, and explore how these choices change how the HAC algorithm runs on a toy data set.

Problem 1

Consider the following four data points in $\mathbb{R}^2$, belonging to three clusters: the black cluster consisting of $\mathbf{x}_1 = (0.1, 0.5)$ and $\mathbf{x}_2 = (0.35, 0.75))$, the red cluster consisting of $\mathbf{x}_3 = (0.28, 1.35)$, and the blue cluster consisting of $\mathbf{x}_4 = (0, 1.01)$.



Different pointwise distances $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_p$ can be used. Recall the definition of the ℓ_1, ℓ_2, and ℓ_∞ norm:

$$\|\mathbf{x}\|_1 = \sum_{j=1}^{m} |x_i| \qquad \|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{m} x_i^2} \qquad \|\mathbf{x}\|_\infty = \max_{j \in \{1,\ldots,m\}} |x_j|$$

Also recall the definition of min-distance, max-distance, centroid-distance, and average-distance between two clusters (where $\boldsymbol{\mu}_G$ is the center of a cluster G):

$$\begin{aligned} d_{\min}(G, G') &= \min_{\mathbf{x} \in G, \mathbf{x}' \in G'} d(\mathbf{x}, \mathbf{x}') \\ d_{\max}(G, G') &= \max_{\mathbf{x} \in G, \mathbf{x}' \in G'} d(\mathbf{x}, \mathbf{x}') \\ d_{\text{centroid}}(G, G') &= d(\boldsymbol{\mu}_G, \boldsymbol{\mu}_{G'}) \\ d_{\text{avg}}(G, G') &= \frac{1}{|G||G'|} \sum_{\mathbf{x} \in G} \sum_{\mathbf{x}' \in G'} d(\mathbf{x}, \mathbf{x}') \end{aligned}$$

1. Draw the 2D unit sphere for each norm, defined as $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| = 1\}$. Feel free to do it by hand, take a picture and include it in your pdf.
2. For each norm ($\ell_1, \ell_2, \ell_\infty$) and each clustering distance, specify which two clusters would be the first to merge.
3. Draw the complete dendrograms showing the order of agglomerations for the ℓ_2 norm and each of the clustering distances. We have provided some code to make this easier for you. You are not required to use it.

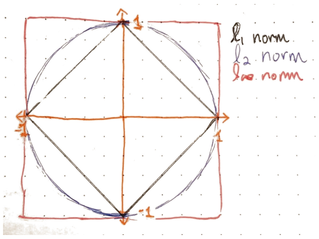


Figure 1: Problem 1, 2d unit ball for l1, l2, and l∞ norms

Solution: Hierarchical Clustering Clustering

1. Draw the 2D unit sphere for each norm, defined as $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| = 1\}$. Feel free to do it by hand, take a picture and include it in your pdf.

   See fig. 1

2. For each norm $(\ell_1, \ell_2, \ell_\infty)$ and each clustering distance, specify which two clusters would be the first to merge.

   See table 1.

Table 1: First clusters to merge

	L1 norm	L2 norm	L∞ norm
Min dist	black, blue	black, blue	red, blue
Max dist	black, blue	red, blue	red, blue
Avg dist	red, blue	red, blue	red, blue
Centroid dist	black, blue	black, blue	black, blue

3. Draw the complete dendrograms showing the order of agglomerations for the ℓ_2 norm and each of the clustering distances. We have provided some code to make this easier for you. You are not required to use it.

   See fig. 2

   For reference on distances used, see table 2

Expectation-Maximization for Gaussian Mixture Models [7pts]

In this problem we will explore expectation-maximization for the Gaussian Mixture model. Each observation $\mathbf{x}_i$ is a vector in $\mathbb{R}^D$. We posit that each observation comes from *one* mixture component. For this problem, we will assume there are c components. Each component $k \in \{1, \ldots, c\}$ will be associated with a mean vector $\mu_k \in R^D$ and a covariance Σ_k. Finally let the (unknown) overall mixing proportion of the components be $\boldsymbol{\theta} \in [0,1]^c$, where $\sum_{k=1}^{c} \theta_k = 1$.

Our generative model is that each of the n observations comes from a single component. We encode observation i's component-assignment as a one-hot vector $\mathbf{z}_i \in \{0,1\}^c$ over components. This one-hot vector is drawn from $\boldsymbol{\theta}$; then, $\mathbf{x}_i$ is drawn from $N(\mu_{z_i}, \Sigma_{z_i})$. Formally documents are generated in two steps:

$$\begin{aligned} \mathbf{z}_i &\sim \text{Categorical}(\boldsymbol{\theta}) \\ \mathbf{x}_i &\sim N(\mu_{z_i}, \Sigma_{z_i}) \end{aligned}$$

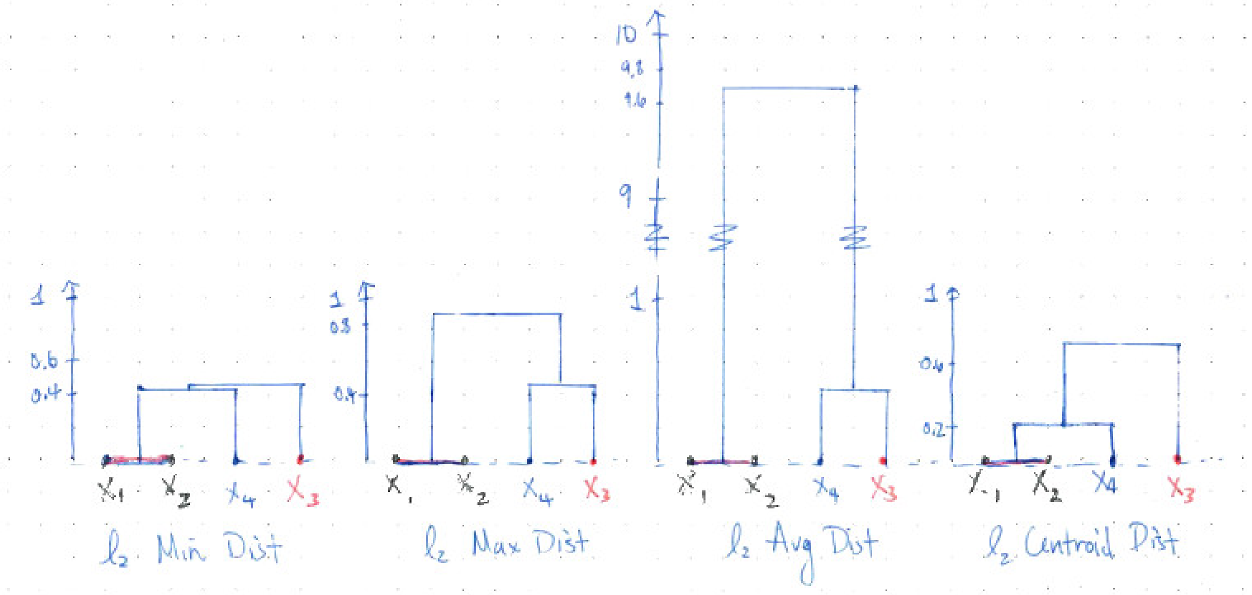


Figure 2: Problem 1, dendogram for clustering with l2 norm and each of the clustering distances.

	L1 norm		**L2 norm**		**LâĹđ norm**	
min dist		(merge dist)		(merge dist)		(merge dist)
First merge	black, blue	0.61	black, blue	0.436	red, blue	0.34
Second merge	all	0.62	all	0.440	all	0.35
max dist						
First merge	black, blue	0.61	red, blue	0.440	red, blue	0.34
Second merge	all	1.03	all	0.869	all	0.34
avg dist						
First merge	red, blue	0.62	red, blue	0.440	red, blue	0.34
Second merge	all	11.68	all	9.715	all	9.24
centroid dist						
First merge	black, blue	0.25	black, blue	0.210	black, blue	0.205
Second merge	all	1.09	all	0.657	all	0.515

Table 2: Problem 1. Not asked for, but here are actual merge distances as output by python script for each combination of norms (distances) and clustering criteria

Problem 2

1. **Intractability of the Data Likelihood** Let ϕ_k represent all the parameters associated with a component (μ_k, Σ_k). We are generally interested in finding a set of parameters ϕ_k that maximize the data likelihood $\log p(\{x_i\}|\{\phi_k\})$. Expand the data likelihood to include the necessary sums over observations x_i and latents z_i. Why is optimizing this loss directly intractable?

2. **Complete-Data Log Likelihood**

   Define the complete data for this problem to be $D = \{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^n$. Write out the complete-data (negative) log likelihood.

   $$\mathcal{L}(\boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^c) = -\ln p(D \,|\, \boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^c).$$

3. **Expectation Step** Our next step is to introduce a mathematical expression for $\mathbf{q}_i$, the posterior over the hidden topic variables $\mathbf{z}_i$ conditioned on the observed data $\mathbf{x}_i$ with fixed parameters, i.e $p(\mathbf{z}_i|\mathbf{x}_i; \boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^c)$.

   - Write down and simplify the expression for $\mathbf{q}_i$.
   - Give an algorithm for calculating $\mathbf{q}_i$ for all i, given the observed data $\{\mathbf{x}_i\}_{i=1}^n$ and settings of the parameters $\boldsymbol{\theta}$ and $\{\mu_k, \Sigma_k\}_{k=1}^c$.

4. **Maximization Step** Using the $\mathbf{q}_i$ estimates from the Expectation Step, derive an update for maximizing the expected complete data log likelihood in terms of $\boldsymbol{\theta}$ and $\{\mu_k, \Sigma_k\}_{k=1}^c$.

   - Derive an expression for the expected complete-data log likelihood in terms of $\mathbf{q}_i$.
   - Find an expression for $\boldsymbol{\theta}$ that maximizes this expected complete-data log likelihood. You may find it helpful to use Lagrange multipliers in order to force the constraint $\sum \theta_k = 1$. Why does this optimized $\boldsymbol{\theta}$ make intuitive sense?
   - Apply a similar argument to find the value of the (μ_k, Σ_k)'s that maximizes the expected complete-data log likelihood.

5. Finally, compare this EM approach to the generative model for classification in Homework 2. How are the computations similar? Different?

Solution EM for Gaussian Mixture Models

1. **Intractability of the Data Likelihood**

   Let ϕ_k represent all the parameters associated with a component (μ_k, Σ_k). We are generally interested in finding a set of parameters ϕ_k that maximize the data likelihood $\log p(x_i|\phi_k)$. Expand the data likelihood to include the necessary sums over observations x_i and latents z_i.

   Suppose we observe some data and we would like to calculate the likelihood of observing such data.

   Let ϕ_k be our model parameters. If we assume a mixture of Gaussians model, ϕ_k consists of the μ_k and Σ_k for each component (aka class).

   Note: The "complete-data" case is if we knew the true class labels, in which case we cold directly calculate $p(\mathbf{x}, \mathbf{z}|\phi)$. However, as $\mathbf{z}$ is a latent (unobservable) variable, we may only calculate $p(\mathbf{x}|\phi)$.

(a) Let θ_k indicate the **class** distribution for our latent variable $\mathbf{z}$'s.

$$p(\mathbf{z} = C_k) = \theta_k, \text{for } k \in \{1, ..., c\} \tag{1}$$

For this problem we assume z_i to be identically distributed, so that θ_k is the same for each.

(b) Let our class conditional distribution be

$$p(\mathbf{x}|\mathbf{z} = C_k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2}$$

As per the s17 lecture 14 slides, the log likelihood of observed data $\mathbf{x}$ may thus be written

$$\ln p(\mathbf{x}_i|\phi) = \ln \sum_{k=1}^{c} \theta_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3}$$

That is, the probability of our $\mathbf{x}$'s given our model parameters is equal to the sum, across all possible classes, of the probability θ_k of $\mathbf{x}_k$ being from each class times the normal distribution from where x would be drawn if it was of class k.

Question: Why is optimizing this loss directly intractable?

We cannot find a closed solution for this, as the log of the sum prevents us from being able to decompose it by ϕ. In general, it is hard to simplify the log of a sum of expressions (unliked decomposing a log of a product of expressions).

2. **Complete-Data Log Likelihood**

Define the complete data for this problem to be $D = \{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^{n}$. Write out the complete-data (negative) log likelihood.

$$\mathcal{L}(\boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^{c}) = -\ln p(D \,|\, \boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^{c}).$$

As before:

(a) Let D be the data $\mathbf{x}, \mathbf{z}$.

(b) Let ϕ be the parameters θ (class distribution) as well as $\{\mu_k, \Sigma_k\}_{k=1}^{c}$ (parameters for class conditional distribution.)

Per the definition of a joint probability distribution, the probability of both x_i and z_i happening is equal to the probability of z_i happening, times the probability of x_i happening given that z_i happened.

$$p(D) = p(\mathbf{x}_i, \mathbf{z}_i) = p(\mathbf{z}_i)p(\mathbf{x}_i|\mathbf{z}_i) \tag{4}$$

Let z_{ik} be the indicator variable, which is 1 only when x_i is of true class k, and zero otherwise.

We can now write out the negative log of the complete-data log likelihood:

$$\begin{aligned} -\ln p(D|\phi) &= -\ln p(\mathbf{x}|\phi) - \ln p(z|\phi) \tag{5} \\ p(\mathbf{z}_i) &= \prod_{k=1}^{c} \theta_z^{z_{ik}} \tag{6} \\ p(\mathbf{x}_i|\mathbf{z}_i) &= \prod_{k=1}^{c} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{ik}} \tag{7} \\ p(\mathbf{z}_i)p(\mathbf{x}_i|\mathbf{z}_i) &= \prod_{k=1}^{c} \theta_k^{z_{ik}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{ik}} \tag{8} \end{aligned}$$

Note by thinking carefully that in the product series case, we use the indicator variable as an exponent (think through what raising something to the power of 1 or 0 does). In a product series, multiplying by one "does nothing", while in a summation series, summing a zero "does nothing".

Taking the log of eq. (8) we get that

$$\ln p(\mathbf{x}_i, \mathbf{z}_i|\phi) = -\sum_{k=1}^{c} z_{ik} \ln \theta_k - \sum_{k=1}^{c} z_{ik} \ln \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{ik}} \tag{9}$$

3. **Expectation Step** Our next step is to introduce a mathematical expression for $\mathbf{q}_i$, the posterior over the hidden topic variables $\mathbf{z}_i$ conditioned on the observed data $\mathbf{x}_i$ with fixed parameters, i.e $p(\mathbf{z}_i|\mathbf{x}_i; \boldsymbol{\theta}, \{\mu_k, \Sigma_k\}_{k=1}^{c})$.

   - Write down and simplify the expression for $\mathbf{q}_i$.

     As per section 8 notes, problem 3.4 :

     $$\begin{aligned} \mathbf{q}_i &= p(\mathbf{z}_i = C_k|\mathbf{x}_i, \phi) \tag{10} \\ &\propto p(\mathbf{z}_i, \phi)p(\mathbf{x}_i|\mathbf{z}_i, \phi) \tag{11} \\ &\propto \theta_k^{z_{ik}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{ik}} \tag{12} \end{aligned}$$

   - Give an algorithm for calculating $\mathbf{q}_i$ for all i, given the observed data $\{\mathbf{x}_i\}_{i=1}^{n}$ and settings of the parameters $\boldsymbol{\theta}$ and $\{\mu_k, \Sigma_k\}_{k=1}^{c}$.

     We have settings for $\boldsymbol{\theta}$ and $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ (specifically: if it's the first iteration, these are chosen randomly, otherwise these are output from the previous iteration); and that we have observed data $\mathbf{x}$. We see that our posterior belief is our belief about our the distribution of our class variables z given our observed data x (In k–means, the posterior would be our belief about the location of our centroids given our unlabeled data).

     Above, we wrote down how to calculate q_i for one datapoint (implicitly, 11 may be denoted more accurately as $(q_i)_k$. That is to say, we will have i number of vectors q_i, where each vector is of length k:

```
def gaussian(x, mu, sigma):
    g = 1/(sigma * sqrt(2*pi) * exp(-.05*((x-mu)/sigma)**2
    return g

for each datapoint x_i from i=(1 to n)
    q_i_per_i = []
    For each class k from k=(1 to c)
        q_i[k] = ( thetas[k] *
            gaussian(x_i, current_mean_estimates[k],
                    current_sigma_estimates[k]) )
    q_i_per_i[x_i] = q_i[k]
```

4. **Maximization Step** Using the $\mathbf{q}_i$ estimates from the Expectation Step, derive an update for maximizing the expected complete data log likelihood in terms of $\boldsymbol{\theta}$ and $\{\mu_k, \Sigma_k\}_{k=1}^c$.

- Derive an expression for the expected complete-data log likelihood in terms of $\mathbf{q}_i$.

  Now we have the true class labels (well, our estimate of them) from the expectation step. We can now calculate, as per slide 19, sp17 lec14, the expected (positive) complete-data log likelihood (which we will next try to maximize):

$$\mathbb{E}_Z[\mathcal{L}(\phi)] = \mathbb{E}_Z\left[\sum_{i=1}^{n} \ln(p(\mathbf{x}_i, \mathbf{z}_i|\phi)\right] \tag{13}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \ln \theta_k + \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \ln \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{14}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \left(\ln \theta_k + \ln \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \tag{15}$$

(25 revisited)

- Find an expression for $\boldsymbol{\theta}$ that maximizes this expected complete-data log likelihood. You may find it helpful to use Lagrange multipliers in order to force the constraint $\sum \theta_k = 1$. Why does this optimized $\boldsymbol{\theta}$ make intuitive sense?

  To maximize given a constraint, we will use the method of Lagrangian multipliers. As we are maximizing with respect to θ_k, we may also drop the terms (on the right) not including θ. Previously, in homework 2, we had z_{ik} as our indicator variable. However, we now treat z as a latent variable, and instead we have a "soft estimate" q variable.

$$\mathcal{L}(\theta_{\mathbf{k}}, \lambda) = \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik} \ln \theta_k + \lambda(\sum_{k=1}^{c} \theta_k - 1) \tag{16}$$

  Take the partial derivative of $\mathcal{L}$ with respect to θ_k and set it equal to zero, and noting that

$$\sum_{i} q_{ik} = n_k \tag{17}$$

(see homework 2 problem 2.2 for more details)

$$0 = \sum_{i=1}^{n} \frac{q_{ik}}{\theta_k} - \lambda \tag{18}$$

$$\theta_k = \frac{n_k}{\lambda} \tag{19}$$

Take the partial derivative with respect to λ and set it equal to zero to get

$$0 = \sum_{k=1}^{c} \theta_k - 1 \tag{20}$$

$$\tag{21}$$

Now let's solve from lambda by combining eq. (19) and eq. (21)

$$0 = \sum_{k=1}^{c} \frac{n_k}{\lambda} - 1 \tag{22}$$

$$\lambda = n_k \tag{23}$$

Returning to eq. (19) we now see that

$$\theta_k = \frac{n_k}{n} \tag{24}$$

(Note that since we are actually estimating θ_k, it would be clearer to write $\hat{\theta}_k$)

This solution for $\hat{\theta}_k$makes sense: the optimal (prior) probability of a given x_i belong to a class k is equal to the proportion of observations (we've estimated in this iteration) that come from class k.

- Apply a similar argument to find the value of the (μ_k, Σ_k)'s that maximizes the expected complete-data log likelihood. For $\boldsymbol{\mu}_k$ case.

  As in homework 2:

  $$= \sum_{i=1}^{n} \sum_{k=1}^{c} q_{ik} \left(\ln \theta_k + \ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \tag{25}$$

  To solve for optimal $\boldsymbol{\mu}_k$, taking the derivative of eq. (25) with respect to $\boldsymbol{\mu}_k$ and set to zero. Note that the left hand terms drop out. Furthermore, we remove terms in the log gaussian without $\boldsymbol{\mu}_k$

  $$\begin{aligned} &= \sum_{i=1}^{n} \sum_{k=1}^{c} q_{ik} \left(\ln \theta_k + \ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{c} q_{ik} \ln \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned} \tag{25 revisited}$$

The natural log of the Gaussian is equal to

$$\begin{aligned}
&= -\frac{1}{2}\Big(D\ln 2\pi + \ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\Big) \qquad (26)\\
&= -\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) \qquad (27)
\end{aligned}$$

Thus we get that we are solving for $\boldsymbol{\mu}_k$ s.t.

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\mu}_k} &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) + const \qquad (28)\\
&= 0 \qquad (29)
\end{aligned}$$

Carrying the derivative out:

$$\begin{aligned}
0 &= -\frac{1}{2}(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})\sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k) \qquad (30)\\
0 &= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k) \qquad (31)\\
0 &= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}\mathbf{x}_i - q_{ik}\boldsymbol{\mu}_k \qquad (32)\\
\sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}\boldsymbol{\mu}_k &= \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}\mathbf{x}_i \qquad (33)\\
& \qquad (34)
\end{aligned}$$

As the μ_k is the same for all points in the class, $\sum_{k=1}^{c} q_{ik}\boldsymbol{\mu}_k$ is simply $n_k\boldsymbol{\mu}_k$. Thus we get that

$$\boldsymbol{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n} q_{ik}\mathbf{x}_i \qquad (35)$$

- Apply a similar argument to find the value of the (μ_k, Σ_k)'s that maximizes the expected complete-data log likelihood. $\boldsymbol{\Sigma}_k$ case.

  As before for $\boldsymbol{\mu}_k$

$$\hat{\boldsymbol{\Sigma}_k} = \frac{1}{n_k}\sum_{i=1}^{n} q_{ik}(x_i - \hat{\boldsymbol{\mu}}_k)(x_i - \hat{\boldsymbol{\mu}}_k)^T \qquad (36)$$

Dropping terms without $\boldsymbol{\Sigma}$

$$\mathcal{L} = \sum_{i=1}^{n}\sum_{k=1}^{c} q_{ik}\left[-\frac{1}{2}(\ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k))\right] \qquad (37)$$

Taking the partial derivative and collapsing sums, we get

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \frac{n_k}{2}\boldsymbol{\Sigma}_k - \frac{1}{2}\sum_{i=1}^{n} q_{ik}(x_i - \hat{\boldsymbol{\mu}}_k)(x_i - \hat{\boldsymbol{\mu}}_k)^T \tag{38}$$

$$\tag{39}$$

For our final answer

$$\boldsymbol{\Sigma}_k = \frac{1}{n_k}\sum_{i=1}^{n} q_{ik}(x_i - \boldsymbol{\mu}_k)(x_i - \boldsymbol{\mu}_k)^T \tag{40}$$

5. Finally, compare this EM approach to the generative model for classification in Homework 2. How are the computations similar? Different?

In homework 2, classification with generative models, we view the data x that we observe as being generated by a random variable y. We estimate the joint probability model $p(x, y)$ to create predications about y given future observations of x.

[y] $\rightarrow$ [x]

In the EM approach, we similarly consider the data to come from a generative *process*, and apply Baye's rule to model the posterior distribution aka our estimate of the class label given a particular datapoint x and parameters for the distribution from which z is drawn. But we consider that this process involves an unknown and unobservable random variable z. Thus, we do not know which sample came from which class, and we must use EM as a way to maximize the likelihoods we are solving for.

[z] $\rightarrow$ [x]

In short, homework 2 was supervised learning, and the EM approach is unsupervised learning. We are looking at the discrete model in both homework 2 and this homework.

K-Means [15 pts]

For this problem you will implement K-Means clustering from scratch. Using `numpy` is fine, but don't use a third-party machine learning implementation like `scikit-learn`. You will then apply this approach to clustering of image data.

We have provided you with the MNIST dataset, a collection of handwritten digits used as a benchmark of image recogntion (you can learn more about the data set at `http://yann.lecun.com/exdb/mnist/`). The MNIST task is widely used in supervised learning, and modern algorithms with neural networks do very well on this task.

Here we will use MNIST unsupervised learning. You have been given representations of 6000 MNIST images, each of which are 28×28 greyscale handwritten digits. Your job is to implement K-means clustering on MNIST, and to test whether this relatively simple algorithm can cluster similar-looking images together.

Problem 3

The given code loads the images into your environment as a 6000x28x28 array.

- Implement K-means clustering from different random initializations and for several values of K using the ℓ_2 norm as your distance metric. (You should feel free to explore other metrics than the ℓ_2 norm, but this is strictly optional.) Compare the K-means objective for different values of K and across random initializations.
- For three different values of K, and a couple of random restarts for each, show the mean images for each cluster (i.e., for the cluster prototypes), as well as the images for a few representative images for each cluster. You should explain how you selected these representative images. To render an image, use the pyplot `imshow` function.
- Are the results wildly different for different restarts and/or different values of K? For one of your runs, plot the K-means objective function as a function of iteration and verify that it never increases.

As in past problem sets, please include your plots in this document. (There may be several plots for this problem, so feel free to take up multiple pages.)

Solution: K-Means

- Implement K-means clustering from different random initializations and for several values of K using the ℓ_2 norm as your distance metric. (You should feel free to explore other metrics than the ℓ_2 norm, but this is strictly optional.) Compare the K-means objective for different values of K and across random initializations.

  From the following table 3 we may observe that the objective decreases as K increases. Additionally, for each run of K, the final objective (cost) is fairly consistent, without large variations, compared to the changes in final cost between different K values.

Table 3: K-Means l2 objective for different values of K. Normalized = divided by 6000*28*28, for readability.

K (# clusters)	Objective (normalized)	Objective
2	2.2632	10646092.8
2	2.2631	10645622.4
2	2.2631	10645622.4
5	2.1117	9933436.8
5	2.1191	9968246.4
5	2.1183	9964483.2
10	1.99	9360960
10	1.9888	9355315.2
10	1.9868	9345907.2
15	1.9222	9042028.8
15	1.9212	9037324.8
15	1.9187	9025564.8

- For three different values of K, and a couple of random restarts for each, show the mean images for each cluster (i.e., for the cluster prototypes), as well as the images for a few representative images for each cluster. You should explain how you selected these representative images. To render an image, use the pyplot `imshow` function.

  In the representative images, each row indicates images from one cluster. For each row, the three left images are chosen as the three closest (as defined by the objective function) to the centroid, and the three right images are the three worst scoring images.

  TODO REMEMBER to comment out images below for final submission!!

- Are the results wildly different for different restarts and/or different values of K? For one of your runs, plot the K-means objective function as a function of iteration and verify that it never increases.

  Compared to the k-means objective (as answered in part 1), The representative images / cluster results do seem to vary noticeably between runs, aside from the K=2 case, which exhibits little variation. The results do vary even more for different values of K; for K=10, the mean images will sometimes miss entire numbers, while for k=15 we will get repeats of a few of the values even if we get most of the digits.

  For the following, we graph a run of the KMeans algorithm for K=10 clusters. The run converged after 24 iterations (where by converge I set it to mean that the previous and current objective value did not change). Although a bit hard to see, puts 'Hellow World'

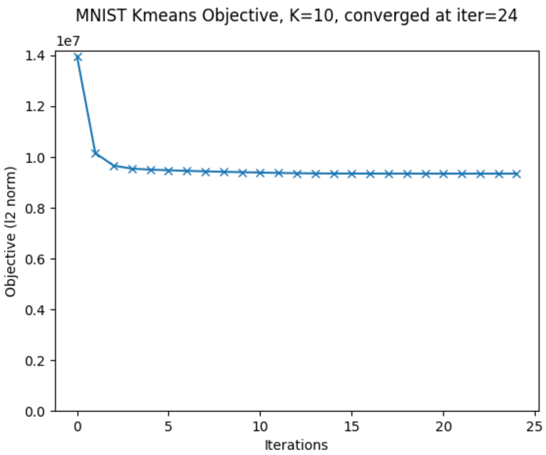


Figure 3: Problem 3: Verification that K-means objective function decreases as a function of iteration

Problem 4 (Calibration, 1pt)

Approximately how long did this homework take you to complete?

30-40 hours

Name, Email, and Collaborators

Name: Nao Ouyang

Email: nouyang@g.harvard.edu

Collaborators: Eric Buse Philip David