Assignment #4

Due: 11:59pm April 5, 2019

Homework 4: Clustering and EM

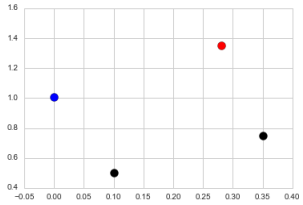
This homework assignment focuses on different unsupervised learning methods from a theoretical and practical standpoint. In Problem 1, you will explore Hierarchical Clustering and experiment with how the choice of distance metrics can alter the behavior of the algorithm. In Problem 2, you will derive from scratch the full expectation-maximization algorithm for fitting a Poisson mixture model. In Problem 3, you will implement PCA 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, tex, 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.

Problem 1 (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.

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:

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

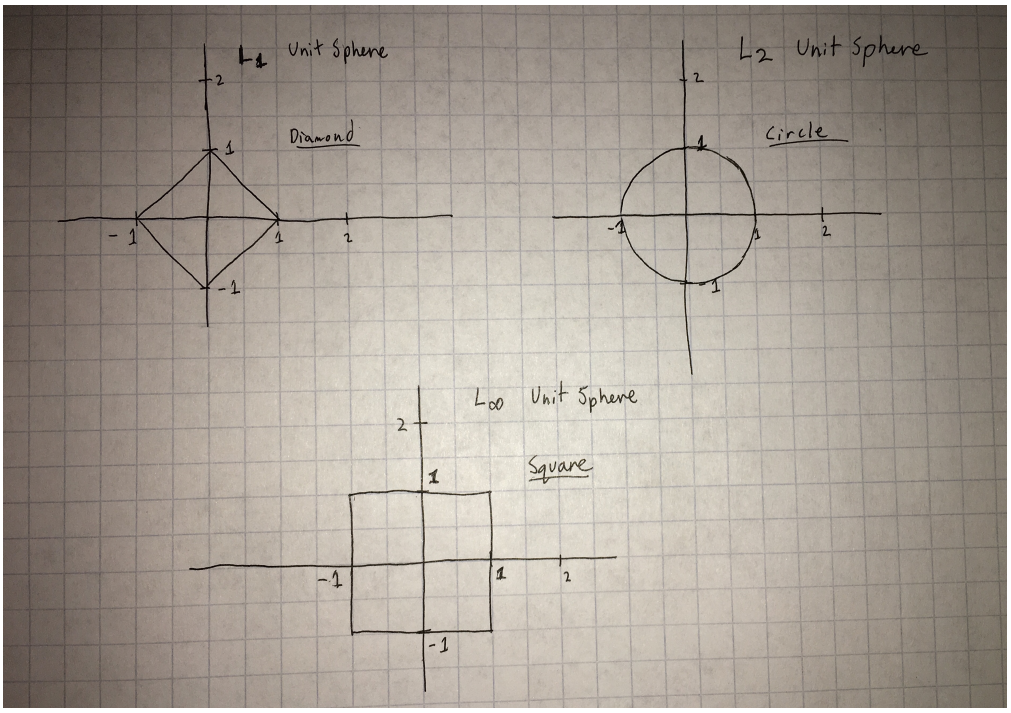
Also recall the definition of min-distance, max-distance, centroid-distance, and average-distance between two clusters (where $\boldsymbol{\mu}_G$ is the center of a cluster G):

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

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.

Solution

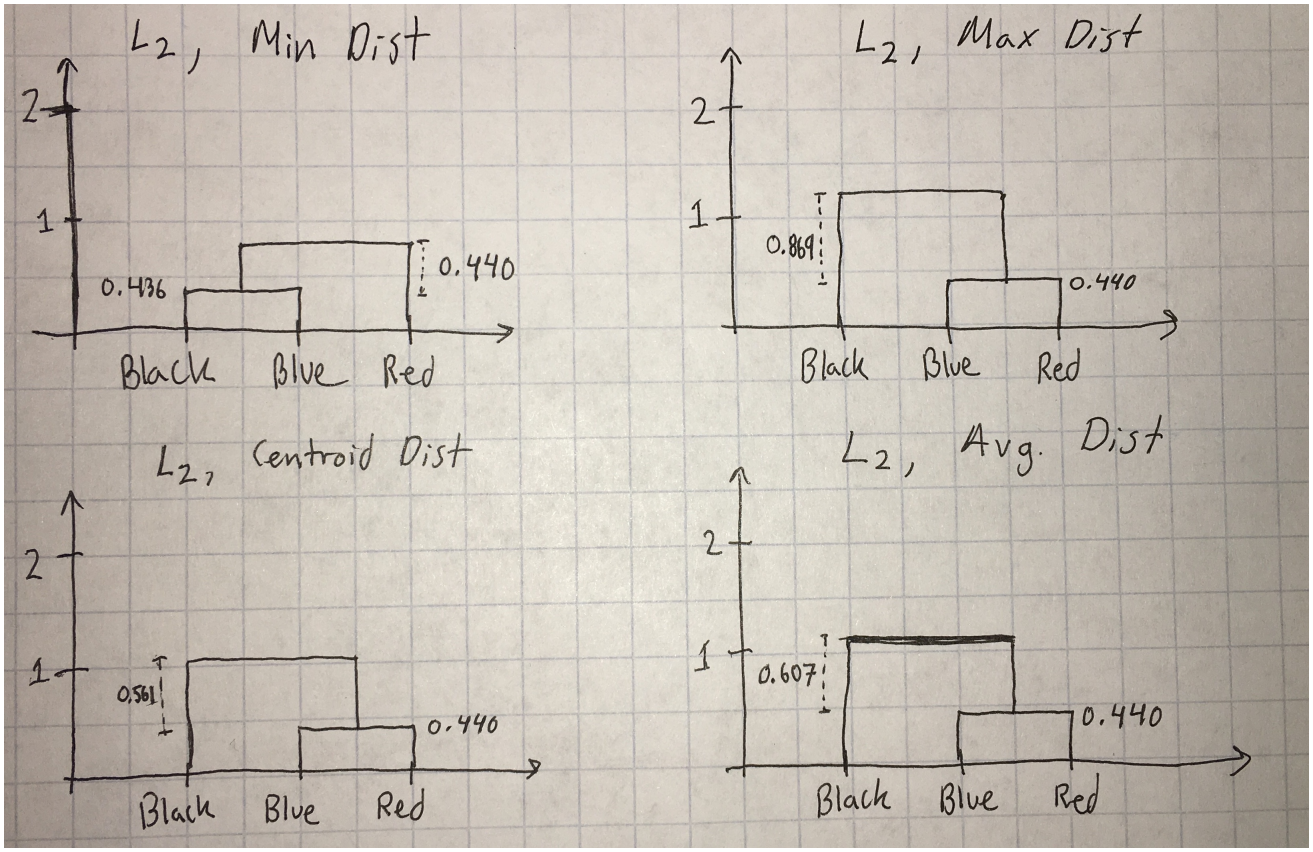
1. The unit spheres are drawn below. They are a diamond, circle, and square for l_1, l_2, and l_∞ respectively.



2. See the table below for my sesults of which two clusters would merge first under each combination of distance metric and cluster center.

Part 2: First Two Clusters to Merge			
	l_1 distance	l_2 distance	l_∞ distance
Min	Black and Blue	Black and Blue	Red and Blue
Max	Black and Blue	Red and Blue	Red and Blue
Centroid	Black and Blue	Red and Blue	Red and Blue
Average	Black and Blue	Red and Blue	Red and Blue

3. Below are the dendrograms, which I attempted to draw to scale. The distance which is hard to read in the "L2, Centroid Dist" dendrogram is 0.561.


L_2, Min Dist
2
1
0.436
0.440
Black
Blue
Red
L_2, Max Dist
2
1
0.869
0.440
Black
Blue
Red
L_2, Centroid Dist
2
1
0.561
0.440
Black
Blue
Red
L_2, Avg. Dist
2
1
0.607
0.440
Black
Blue
Red

Problem 2 (Expectation-Maximization for Poisson Mixture Models, 7pts)

In this problem we will explore expectation-maximization for the Poisson Mixture model. Each observation $\mathbf{x}_n$ is a vector in the non-negative integers $\mathbb{Z}^*$. We posit that each observation comes from *one* mixture component. For this problem, we will assume there are K components. Each component $k \in \{1, \ldots, K\}$ will be associated with a mean vector $\lambda_k \in R^+$. Finally let the (unknown) overall mixing proportion of the components be $\boldsymbol{\theta} \in [0,1]^K$, where $\sum_{k=1}^K \boldsymbol{\theta}_k = 1$.

Our generative model is that each of the N observations comes from a single component. We encode observation n's component-assignment as a one-hot vector $\mathbf{z}_n \in \{0,1\}^K$ over components. This one-hot vector is drawn from $\boldsymbol{\theta}$; then, $\mathbf{x}_n$ is drawn from Poisson(λ_{z_n}), which simply means that if $\mathbf{z}_{nj} = 1$ for some $j \in \{1, \ldots, K\}$ (i.e. the jth element of $\mathbf{z}_n$ equals 1), then $\mathbf{x}_n \sim \text{Poisson}(\lambda_j)$.

Formally, documents are generated in two steps:

$$\begin{aligned} \mathbf{z}_n &\sim \text{Categorical}(\boldsymbol{\theta}) \\ \mathbf{x}_n &\sim \text{Poisson}(\lambda_{z_n}) \end{aligned}$$

1. **Intractability of the Data Likelihood** We are generally interested in finding a set of parameters λ_k that maximize the data likelihood $\log p(\{\mathbf{x}_n\}_{n=1}^N | \{\lambda_k\}_{k=1}^K)$. Expand the data likelihood to include the necessary sums over observations $\mathbf{x}_n$ and latents $\mathbf{z}_n$. Why is optimizing this loss directly intractable?

2. **Complete-Data Log Likelihood** Define the complete data for this problem to be $D = \{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^N$. Write out the complete-data negative log likelihood. Note that optimizing this loss is now computationally tractable if we know $\mathbf{z}_n$.

$$\mathcal{L}(\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = -\log p(D \,|\, \boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K).$$

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

   - **Part 3.A** Write down and simplify the expression for $\mathbf{q}_n$.
   - **Part 3.B** Give an algorithm for calculating $\mathbf{q}_n$ for all n, given the observed data $\{\mathbf{x}_n\}_{n=1}^N$ and settings of the parameters $\boldsymbol{\theta}$ and $\{\lambda_k, \}_{k=1}^K$.

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

   - **Part 4.A** Derive an expression for the expected complete-data log likelihood in terms of $\mathbf{q}_n$.
   - **Part 4.B** 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 enforce the constraint $\sum \theta_k = 1$. Why does this optimized $\boldsymbol{\theta}$ make intuitive sense?
   - **Part 4.C** Apply a similar argument to find the values of $\{\lambda_k\}_{k=1}^K$ that maximizes the expected complete-data log likelihood.

5. Suppose that this had been a classification problem, that is, you were provided the "true" categories $\mathbf{z}_n$ of each document, and you were going to perform the classification by inverting the provided generative model. Could you reuse any of your inference derivations above?

Solution

1. Using the law of total probability, we have that for a single data point, x_n, the following holds

$$p(x_n|\{\lambda_k\}_{k=1}^K) = \sum_{j=1}^K p(x_n|z_n = j, \{\lambda_k\}_{k=1}^K)p(z_n = j)$$

Then, since $x_n|z_n$ is distributed Pois(λ_{z_n}) and $p(z_n = j) = \theta_j$, we have that

$$p(x_n|\{\lambda_k\}_{k=1}^K) = \sum_{j=1}^K \frac{\lambda_j^{x_n} e^{-\lambda_j}}{x_n!}\theta_j$$

and thus we have that for all n data points,

$$p(\{\mathbf{x}_n\}_{n=1}^N|\{\lambda_k\}_{k=1}^K) = \prod_{i=1}^N \sum_{j=1}^K \frac{\lambda_j^{x_i} e^{-\lambda_j}}{x_i!}\theta_j$$

meaning

$$\boxed{\log p(\{\mathbf{x}_n\}_{n=1}^N|\{\lambda_k\}_{k=1}^K) = \sum_{i=1}^N \log(\sum_{j=1}^K \frac{\lambda_j^{x_i} e^{-\lambda_j}}{x_i!}\theta_j)}$$

Optimizing this loss directly is intractable since we have the logarithm of a sum. We would maximize over λ and θ which would give us $2k$ first order equations with $2k$ unknowns, but these will be highly nonlinear and complicated equations so there will not be a tractable solution.

2. We have that

$$p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = p(\{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^N|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = \prod_{i=1}^N p(x_i, z_i|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K)$$

By the rules of conditional and joint probability, this becomes

$$= \prod_{i=1}^N p(x_i|z_i, \boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K)p(z_i|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = \prod_{i=1}^N \frac{\lambda_{z_i}^{x_i} e^{-\lambda_{z_i}}}{x_i!}\boldsymbol{\theta}_{z_i}$$

Therefore, we have that

$$-\log p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = -\sum_{i=1}^N \log(\frac{\lambda_{z_i}^{x_i} e^{-\lambda_{z_i}}}{x_i!}\boldsymbol{\theta}_{z_i}) = -\sum_{i=1}^N \sum_{j=1}^K z_{ij} \log(\frac{\lambda_j^{x_i} e^{-\lambda_j}}{x_i!}\boldsymbol{\theta}_j)$$

$$\implies \boxed{-\log p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = -\sum_{i=1}^N \sum_{j=1}^K z_{ij}\big(x_i \log \lambda_j - \lambda_j - \log(x_i!) + \log \boldsymbol{\theta}_j\big)}$$

3. A) By the rules of conditional and joint probability, we have that

$$\mathbf{q}_n = p(\mathbf{z}_n|\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K) = \frac{p(\mathbf{z}_n,\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K)}{p(\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K)}$$

Using the complete data probability found in the previous part (but now only for one data point), and applying law of total probability in the denominator, this is

$$\mathbf{q}_n = p(\mathbf{z}_n|\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K) = \frac{\frac{\lambda_{z_n}^{x_n}e^{-\lambda_{z_n}}}{x_n!}\boldsymbol{\theta}_{z_n}}{\sum_{k=1}^K \frac{\lambda_k^{x_n}e^{-\lambda_k}}{x_n!}\boldsymbol{\theta}_k}$$

$$\implies \boxed{\mathbf{q}_n = \frac{\lambda_{z_n}^{x_n}e^{-\lambda_{z_n}}\boldsymbol{\theta}_{z_n}}{\sum_{k=1}^K \lambda_k^{x_n}e^{-\lambda_k}\boldsymbol{\theta}_k}}$$

B) $\mathbf{q}_n$ is a K dimensional vector where the jth element is $p(\mathbf{z}_n = j|\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K)$. So, to calculate $\mathbf{q}_n$ for data point n, calculate

$$p(\mathbf{z}_n = j|\mathbf{x}_n;\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K) = \frac{\lambda_j^{x_n}e^{-\lambda_j}\boldsymbol{\theta}_j}{\sum_{k=1}^K \lambda_k^{x_n}e^{-\lambda_k}\boldsymbol{\theta}_k}$$

for $j = 1 \dots K$ and store these values in a K dimensional vector.

This can be done efficiently by calculating $\lambda_j^{x_n}e^{-\lambda_j}\boldsymbol{\theta}_j$ for $j = 1 \dots K$, which can be calculated since we have the observed data $\{\mathbf{x}_n\}_{n=1}^N$ and we are setting the parameters $\boldsymbol{\theta}$ and $\{\lambda_k,\}_{k=1}^K$. Then, store these K values in a vector of length K, called $\mathbf{q}_n$. Then rewrite this vector with $\mathbf{q}_n := \frac{\mathbf{q}_n}{\text{sum}(\mathbf{q}_n)}$. We have calculated $\mathbf{q}_n$. Now, repeat this for all n data points to get a $Q_{N\times K}$ matrix where row i is $\mathbf{q}_i$.

4. A) Using the result from (2), we have

$$\log p(D|\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K) = \sum_{i=1}^N\sum_{j=1}^K z_{ij}\big(x_i\log\lambda_j - \lambda_j - \log(x_i!)\big) + z_{ij}\log\boldsymbol{\theta}_j$$

$$\implies E\big(\log p(D|\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K)\big) = E\big(\sum_{i=1}^N\sum_{j=1}^K z_{ij}\big(x_i\log\lambda_j - \lambda_j - \log(x_i!)\big) + z_{ij}\log\boldsymbol{\theta}_j\big)$$

By linearity of expectation, this is

$$= \sum_{i=1}^N\sum_{j=1}^K E\big(z_{ij}\big(x_i\log\lambda_j - \lambda_j - \log(x_i!)\big) + z_{ij}\log\boldsymbol{\theta}_j\big)$$

and then since the expectation of an indicator r.v. is its probability of equaling 1, we have

$$\boxed{E\big(\log p(D|\boldsymbol{\theta},\{\lambda_k\}_{k=1}^K)\big) = \sum_{i=1}^N\sum_{j=1}^K \mathbf{q}_{ij}\big(x_i\log\lambda_j - \lambda_j - \log(x_i!)\big) + \mathbf{q}_{ij}\log\boldsymbol{\theta}_j}$$

B) Using Lagrange multipliers as the hint suggests, we have

$$\arg\max_{\boldsymbol{\theta}} E\big(\log p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K)\big) + \gamma\big(\sum_{k=1}^K \boldsymbol{\theta}_k - 1\big)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^N \sum_{j=1}^K [\mathbf{q}_{ij}\big(x_i \log \lambda_j - \lambda_j - \log(x_i!)\big) + \mathbf{q}_{ij} \log \boldsymbol{\theta}_j] + \gamma\big(\sum_{k=1}^K \boldsymbol{\theta}_k - 1\big)$$

Taking the derivative w.r.t. $\boldsymbol{\theta}_\ell$ for $\ell = 1 \dots K$ and w.r.t. γ yields the $K+1$ first order conditions:

$$\text{for } \ell = 1 \dots K\text{: } \sum_{i=1}^N \frac{\mathbf{q}_{i\ell}}{\boldsymbol{\theta}_\ell} - \gamma = 0 \text{ and } \sum_{k=1}^K \boldsymbol{\theta}_k - 1 = 0$$

$$\implies \sum_{i=1}^N \frac{\mathbf{q}_{i\ell}}{\boldsymbol{\theta}_\ell} = \gamma, 1 = \sum_{k=1}^K \boldsymbol{\theta}_k \implies \frac{1}{\boldsymbol{\theta}_\ell} \sum_{i=1}^N \mathbf{q}_{i\ell} = \gamma, 1 = \sum_{k=1}^K \boldsymbol{\theta}_k \implies \frac{1}{\gamma} \sum_{i=1}^N \mathbf{q}_{i\ell} = \boldsymbol{\theta}_\ell, 1 = \sum_{k=1}^K \boldsymbol{\theta}_k$$

$$\implies 1 = \sum_{k=1}^K \frac{1}{\gamma} \sum_{i=1}^N q_{ik} \implies 1 = \frac{1}{\gamma} \sum_{k=1}^K \sum_{i=1}^N q_{ik} \implies \gamma = N$$

Thus, we have that

$$\boxed{\hat{\boldsymbol{\theta}}_\ell = \frac{\sum_{i=1}^N \mathbf{q}_{i\ell}}{N}}$$

for $\ell = 1 \dots K$ defines the ℓth component of the $\boldsymbol{\theta}$ which maximizes the expected complete-data log likelihood. This makes intuitive sense, since our choice of the prior probability of being in category ℓ would just become the proportion of data points that were in this category if we knew all the true assignments, i.e. if each q_i was just a one-hot vector encoding the actual category of data point i. Since we do not have these actual category assignments, we instead sum up the probability that each point is in category k which seems like a logical solution given the information we have (probabilities close to 1 will treat the data point almost as if it were in that category and probabilities close to 0 will do the exact opposite).

C) Now, we must optimize with respect to each λ_ℓ. We do not need to meet a constraint, so we simply have

$$\arg\max_{\boldsymbol{\lambda}} E\big(\log p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K)\big)$$

$$= \arg\max_{\boldsymbol{\lambda}} \sum_{i=1}^N \sum_{j=1}^K \mathbf{q}_{ij}\big(x_i \log \lambda_j - \lambda_j - \log(x_i!)\big) + \mathbf{q}_{ij} \log \boldsymbol{\theta}_j$$

which produces the K first order equations, all of the form

$$\sum_{i=1}^{N}(\frac{\mathbf{q}_{i\ell}x_i}{\lambda_\ell} - \mathbf{q}_{i\ell}) = 0 \implies \sum_{i=1}^{N}\mathbf{q}_{i\ell}x_i = \lambda_\ell \sum_{i=1}^{N}\mathbf{q}_{i\ell}$$

Hence, we have that

$$\boxed{\lambda_\ell = \frac{\sum_{i=1}^{N}\mathbf{q}_{i\ell}x_i}{\sum_{i=1}^{N}\mathbf{q}_{i\ell}}}$$

for $\ell = 1 \dots K$ defines the ℓth component of the λ which maximizes the expected complete-data log likelihood.

5. Now that we have the true $\mathbf{z}_n$ for each document, the likelihood from (1) is still not useful, but we can use our complete-data log likelihood from (2) since optimizing this loss is tractable if we know the $\mathbf{z}_n$'s, but now we do not need to simply pretend that we know them! So, in (3) we have $\mathbf{q}_n = \mathbf{z}_n$, the one-hot encoded vector that will have a 1 in the k'th category if $\mathbf{x}_n$ is in category k and 0 otherwise. Then, in (4), our answers for the $\boldsymbol{\theta}$ and λ which maximize the expected complete-data log likelihood, which is now simply the complete-data log likelihood since the $\mathbf{z}_n$ are known, will be the same. However, since $\mathbf{q}_n = \mathbf{z}_n$ which is one-hot encoded, our answers will simplify. Specifically, we have that

$$\hat{\boldsymbol{\theta}}_\ell = \frac{\sum_{i=1}^{N}\mathbf{q}_{i\ell}}{N} = \frac{\sum_{i=1}^{N}\mathbf{z}_{i\ell}}{N} = \frac{N_\ell}{N}$$

and

$$\lambda_\ell = \frac{\sum_{i=1}^{N}\mathbf{q}_{i\ell}x_i}{\sum_{i=1}^{N}\mathbf{q}_{i\ell}} = \frac{\sum_{i=1}^{N}\mathbf{z}_{i\ell}x_i}{\sum_{i=1}^{N}\mathbf{z}_{i\ell}} = \frac{\sum_{i:z_i=\ell}^{N}x_i}{N_\ell}$$

where N_ℓ is the number of points in category ℓ and $i : z_i = \ell$ is the indices of all data points such that $\mathbf{z}_i = \ell$, so this is summing over all data points in category ℓ. We see that the prior probability of being in a category is simply the proportion of data points that were in that category, and the mean of our distribution for each category is simply the sample mean for each category. These equations resemble the formulas for MLE's we found in other classification problems (when we were working with Gaussians)!

Problem 3 (PCA, 15 pts)

For this problem you will implement PCA from scratch. Using `numpy` to call SVDs is fine, but don't use a third-party machine learning implementation like `scikit-learn`.

We return to the MNIST data set from T3. You have been given representations of 6000 MNIST images, each of which are 28×28 greyscale handwritten digits. Your job is to apply PCA on MNIST, and discuss what kinds of structure is found.

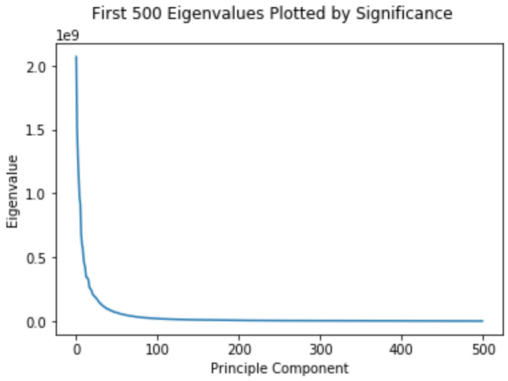
As before, the given code loads the images into your environment as a 6000x28x28 array.

- Compute the PCA. Plot the eigenvalues corresponding to the most significant 500 components in order from most significant to least. Make another plot that describes the cumulative proportion of variance explained by the first k most significant components for values of k, 1 through 500. How much variance is explained by the first 500 components? Describe how the cumulative proportion of variance explained changes with k.
- Plot the mean image as well as the images corresponding the first 10 principle components. How does images compare to the cluster centers from K-means? Discuss any similarities and differences.
- Compute the reconstruction error on the data set using the mean image as well as the first 10 principle components. How does this error compare to running K-means and using the cluster centers as the reconstructions for each image? Discuss any similarities and differences.

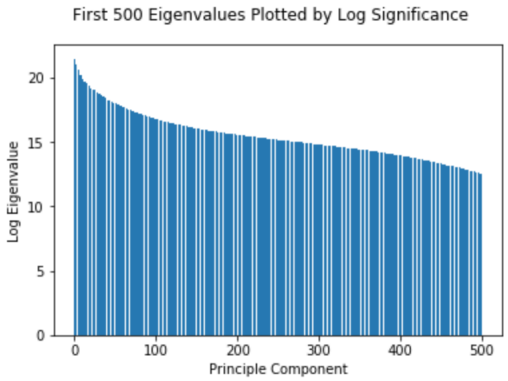
As in past problem sets, please include your plots in this document. (There may be several plots for this problem, so feel free to take up multiple pages.)

Solution

- After performing PCA, I see that the eigenvalues decrease rapidly over the first 500 components, with the sharpest decline happening in the first 50 components or so[1]:



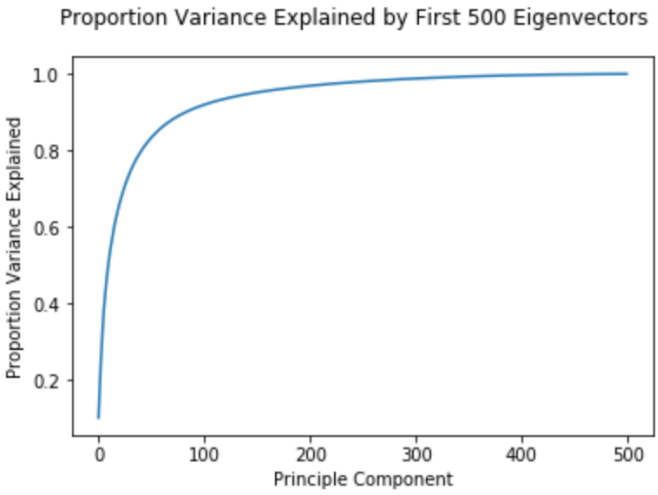
Eigenvalues for the first 500 PC's



Log Eigenvalues for the first 500 PC's

Furthermore, graphing the cumulative proportion of variance explained by the first k most significant components shows how this proportion grows rapidly and approaches 1 asymptotically.

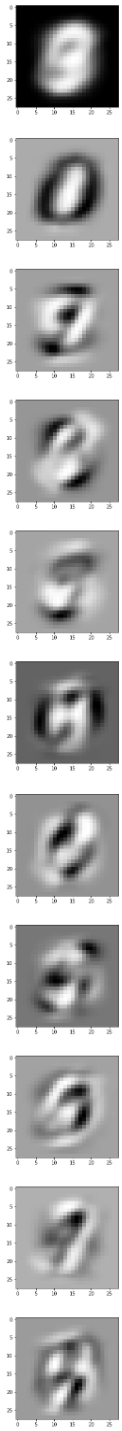
[1] Just a rough estimation!



As can be seen, the proportion of variance explained grows rapidly over the first 100 principle components or so, and reaches a value of about 0.9 by this point. Then, the proportion grows slower and asymptotically approaches 1. By 300 PC's, the proportion of variance explained appears to be nearly 1. Indeed, by the 500'th PC, I found that the proportion of variance explained is $\boxed{0.9994}$, accounting for total variance of $\boxed{20603735864.27}$.

In general, we can conclude that the proportion of variance explained grows with k approaching 1, and in this case it grows very fast at the beginning and then experiences slower asymptotic growth.

- The mean of the images and the first 10 most significant components are shown below:



Unlike the cluster centers for Kmeans, the principle components appear less similar to handwritten digits for the majority of the images. Some of the images still resemble digits, such as 0 and 9 (somewhat). Overall, I would describe these components as resembling digits (or their negated images) or mixtures of digits (or negated mixtures of digits). It makes sense that some of the PC's seem to be negated because the PC's would still be valid if we multiplied by an overall constant of -1. (They would still be orthonormal). It seems feasible that these images could be overlayed (e.g. a linear

combination) to approximate different digits in the data set! And of course, the mean of the data simply resembles a very blurry mixture of all 10 digits.

- The reconstruction error of the data set using the mean image as the approximation of each data point is $\boxed{11024181.94 \approx 1.1 \times 10^7}$ and the reconstruction error using the first 10 principle components is $\boxed{7777920.42 \approx 7.8 \times 10^6}$. For comparison, in the previous theory homework, the Kmeans reconstruction error for $K = 10$ was around 0.94×10^7 after convergence. So, we see that simply approximating the data with a single mean performs worse (in agreement with what we found last week since reconstruction error decreases with k) than approximating the date with $K = 10$ means. However, reconstructing the data with 10 principle components performs considerably better than using $K = 10$ means, but they are on the same order of magnitude.

  We can gain a little intuition behind why this is the case, although the following is not rigorous. Reconstructing the data with Kmeans only requires 10 28×28 data points and 6000 category assignments where as reconstructing the data with 10 PCs requires 10 28×28 data points (the PCs) and a matrix of size 6000×10 (the projection of the data onto these PCs). So, we have compressed the data less when we are using 10 PCs and hence it makes intuitive sense that our approximation of the original data is more accurate with this method.

  Or, consider that with Kmeans, we only have 10 distinct approximations as to what each data point could be, whereas in PCA we create 6000 distinct approximations (one for each data point). This is another intuitive way to shed light on why PCA may be doing better (although PCA would still do this with only one PC and presumably we would not out perform the Kmeans approximation in this case).

- Name: Zachary Dietz
- Email: zachdietz1@gmail.com
- Collaborators: Theo Walker, Kaishu Mason
- Approximately how long did this homework take you to complete (in hours): 12