Assignment #4

Due: 11:59pm April 5, 2019

Homework 4: Clustering and EM

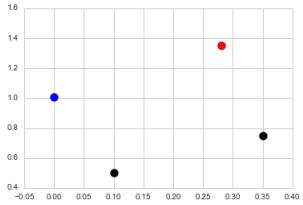
This homework assignment focuses on different unsupervised learning methods from a theoretical and practical standpoint. In Problem 1, you will explore Hierarchical Clustering and experiment with how the choice of distance metrics can alter the behavior of the algorithm. In Problem 2, you will derive 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):

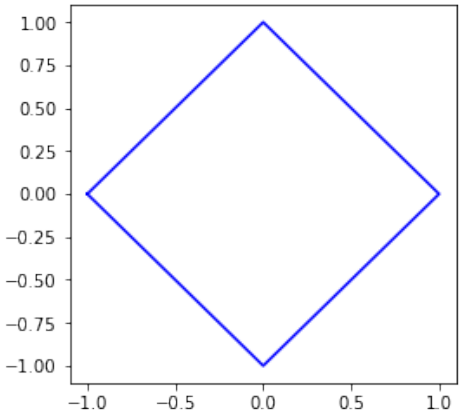
$$\begin{aligned} d_{\text{min}}(G, G') &= \min_{\mathbf{x} \in G, \mathbf{x}' \in G'} d(\mathbf{x}, \mathbf{x}') \\ d_{\text{max}}(G, G') &= \max_{\mathbf{x} \in G, \mathbf{x}' \in G'} d(\mathbf{x}, \mathbf{x}') \\ d_{\text{centroid}}(G, G') &= d(\boldsymbol{\mu}_G, \boldsymbol{\mu}_{G'}) \\ d_{\text{avg}}(G,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

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.

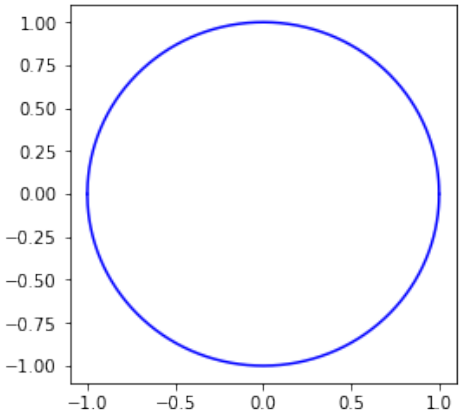
Solution

1.

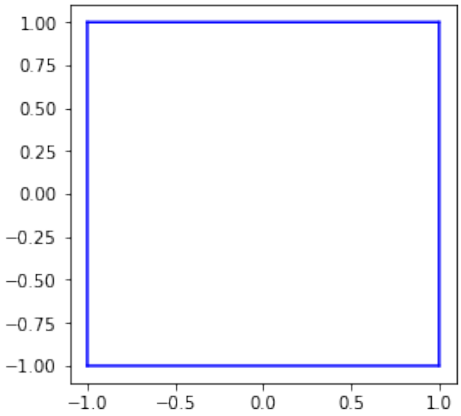
L1 Norm



L2 Norm



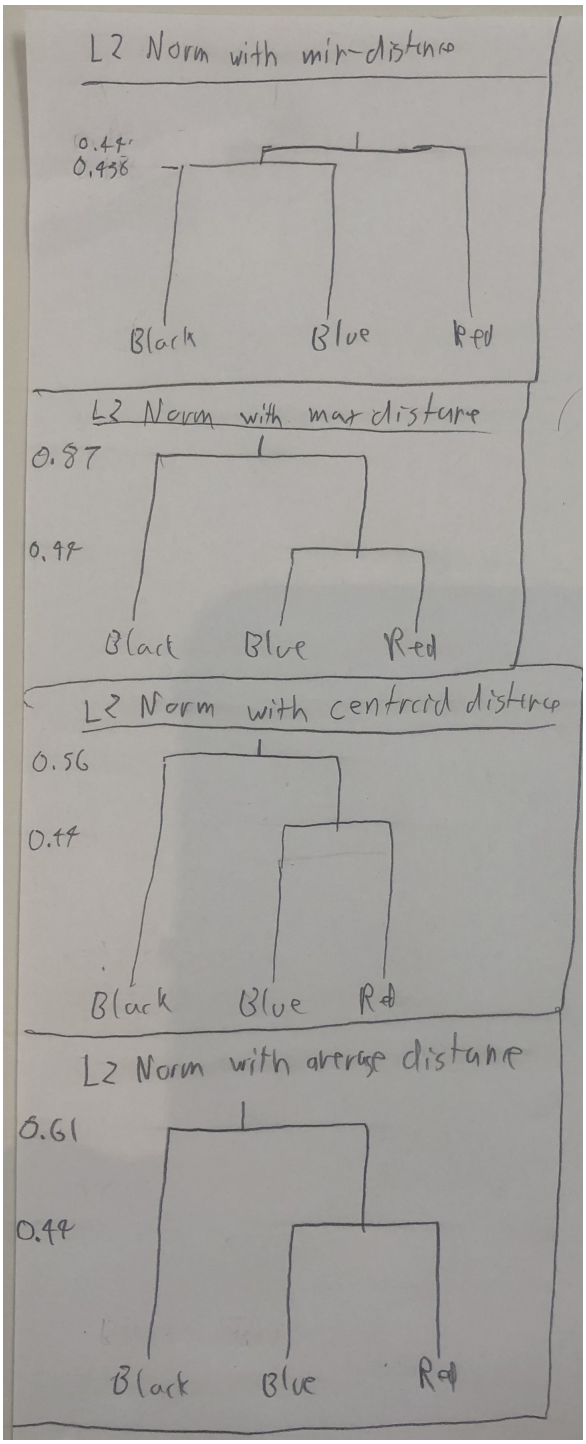
L∞ Norm



2.

norm\clustering	min-distance	max-distance	centroid-distance	average distance
L1	black and blue	black and blue	black and blue	black and blue
L2	black and blue	red and blue	red and blue	red and blue
L∞	red and blue	red and blue	red and blue	red and blue

3.


L2 Norm with min-distance
0.44
0.438
Black
Blue
Red
L2 Norm with max distance
0.87
0.44
Black
Blue
Red
L2 Norm with centroid distance
0.56
0.44
Black
Blue
Red
L2 Norm with average distance
0.61
0.44
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 $\mathbf{z}_n$.

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

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

$p(x_n|\lambda_k)$ is the Poisson p.d.f. for observation n with parameter λ_k

1.

$$\log p(\{\mathbf{x}_n\}_{n=1}^N|\{\lambda_k\}_{k=1}^K) = \sum_{n=1}^{N} \log[\sum_{k=1}^{K} p(x_n, z_{n,k}|\lambda_k)] = \sum_{n=1}^{n} \log[\sum_{k=1}^{K} p(x_n|\lambda_k)\theta_k]$$

Above, we see that our desired parameters that we want to maximize are inside the logarithm, meaning we cannot find a closed form solution for the MLE of these variables in this manner.

2.

$$-\ln p(D|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = -\sum_{n=1}^{N} \ln p(x_n, z_n|\boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K)) = -\sum_{n=1}^{N}(\ln p(x_n|z_n, \lambda_k) + \ln p(z_n|\boldsymbol{\theta}))$$
$$= -\sum_{n=1}^{N} \ln(\prod_{k=1}^{K} p(x_n|\lambda_k)^{z_{n,k}}) - \sum_{n=1}^{N} \ln(\prod_{k=1}^{K} \theta_k^{z_{n,k}}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k} \ln p(x_n|\lambda_k) - \sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k} \ln \theta_k$$

3.

A.

$$p(z_n|x_n, \boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) \propto p(x_n|z_n, \{\lambda_k\}_{k=1}^K)p(z_n|\boldsymbol{\theta}) \propto (\prod_{k=1}^{K} p(x_n\ \lambda_k)^{z_{n,k}}) \cdot (\prod_{k=1}^{K} \theta_k^{z_{n,k}})$$

After making use of our one-hot encoded vector and normalizing,

$$q_{n,k} = p(z_n = C_k|x_n, \boldsymbol{\theta}, \{\lambda_k\}_{k=1}^K) = \frac{p(x_n|\lambda_k) \cdot \theta_k}{\sum_{k'=1}^{K} p(x_n|\lambda_{k'}) \cdot \theta_{k'}}$$

B.

$$\log q_{n,k} = \log p(x_n|\lambda_k) + \log \theta_k - \sum_{k'=1}^{K} p(x_n|\lambda_{k'}) \cdot \theta_{k'} \propto \log p(x_n|\lambda_k) + \log \theta_k - constant$$

Algo:
1) for all n, k compute $\log q_{n,k}$
2) compute $\exp(\log q_{n,k}) = q_{n,k}$
3) Normalize each $q_{n,k}$ by dividing by $\sum_{k'=1}^{K} q_{n,k'}$

4.

A.

plug in our estimate of $q_{n,k}$ for $z_{n,K}$ in the above expression for complete-data log likelihood
$-\sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \ln p(x_n|\lambda_k) - \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \ln \theta_k$

B.

Lagragian: $\lambda \cdot (\sum_{k=1}^{K} \theta_k - 1) = 0$

Take partial w.r.t. $\theta_k \rightarrow -\sum_{n=1}^{N} \frac{q_{n,k}}{\theta_k} + \lambda = 0$

$\theta_k = \frac{1}{\lambda} \sum_{n=1}^{N} q_{n,k}$ where we see that $\lambda = \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k}$

Thus, $\theta_k = \frac{1}{N} \sum_{n=1}^{N} q_{n,k}$

Intuitively, this makes sense as θ_k is the probability of an observation coming from class k; thus, we would estimate it with the expected number of times that we observe class k.

C.

Similarly, take partial of the complete data log likelihood for λ_k

$\sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \frac{\partial(\ln p(x_n|\lambda_k))}{\partial \lambda_k} = \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \frac{\partial(\ln(\lambda_k^{x_n} \exp^{-\lambda_k} \cdot (x_n!)^{-1})}{\partial \lambda_k} \propto \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \frac{\partial(x_n \ln \lambda_k - \lambda_k)}{\partial \lambda_k}$

$= \sum_{n=1}^{N} q_{n,k} (\frac{x_n}{\lambda_k} - 1)$

set to 0

$\sum_{n=1}^{N} q_{n,k} (\frac{x_n}{\lambda_k} - 1) = 0$

$\sum_{n=1}^{N} q_{n,k} \frac{x_n}{\lambda_k} = \sum_{n=1}^{N} q_{n,k}$

$$\lambda_k = \frac{\sum_{n=1}^{N} q_{n,k} x_n}{\sum_{n'=1}^{N} q_{n',k}}$$

D.

Yes, we could reuse some of these derivations. The generative model would correspond to assigning observation n to the class k which maximizes $q_{n,k}$.

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`.

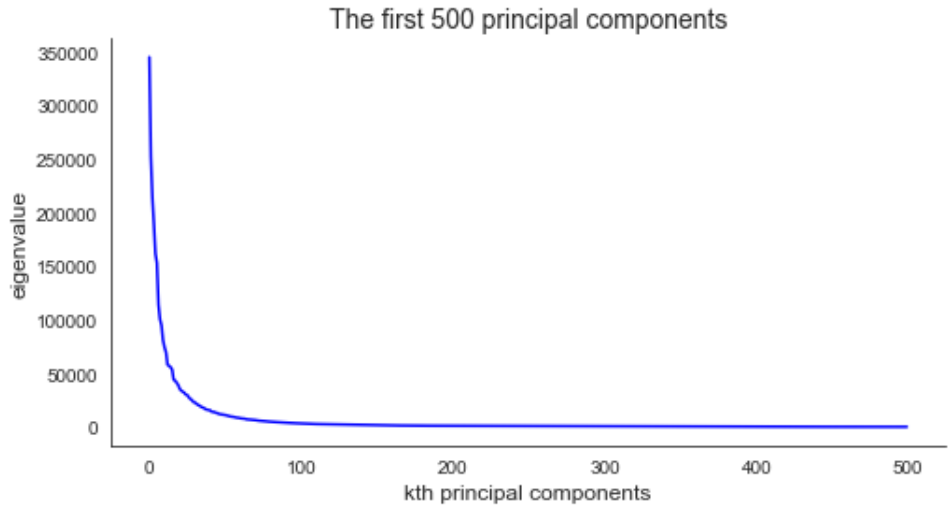
We return to the MNIST data set from T3. You have been given representations of 6000 MNIST images, each of which are 28×28 greyscale handwritten digits. Your job is to apply PCA on MNIST, and discuss what kinds of structure is found.

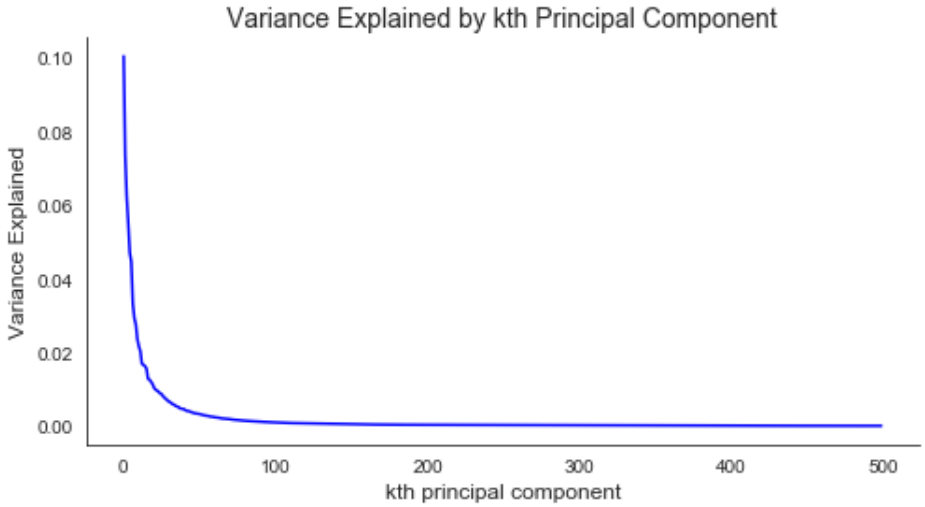
As before, the given code loads the images into your environment as a 6000x28x28 array.

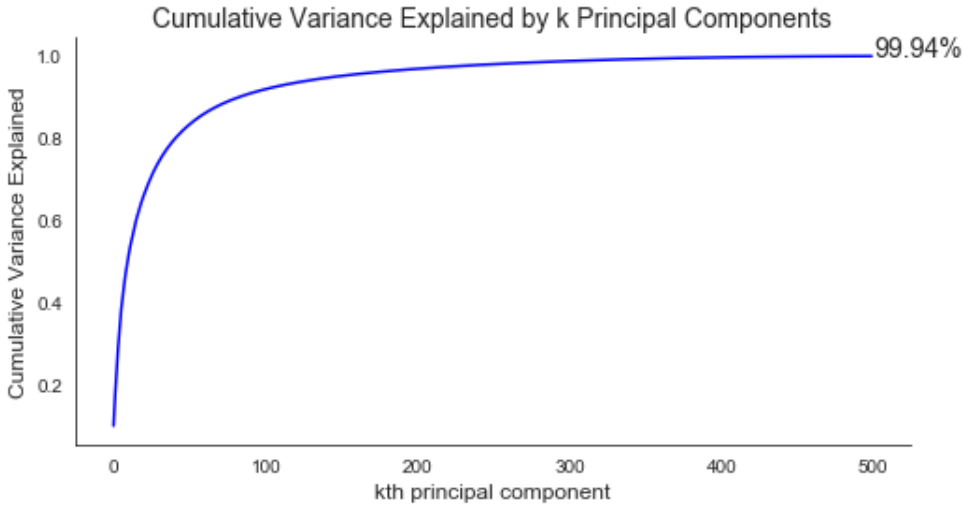
- Compute the PCA. Plot the eigenvalues corresponding to the most significant 500 components in order from most significant to least. Make another plot that describes the cumulative proportion of variance explained by the first k most significant components for values of k, 1 through 500. How much variance 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.)

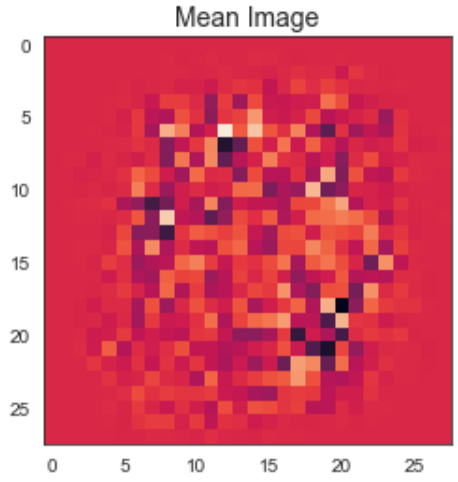
Solution

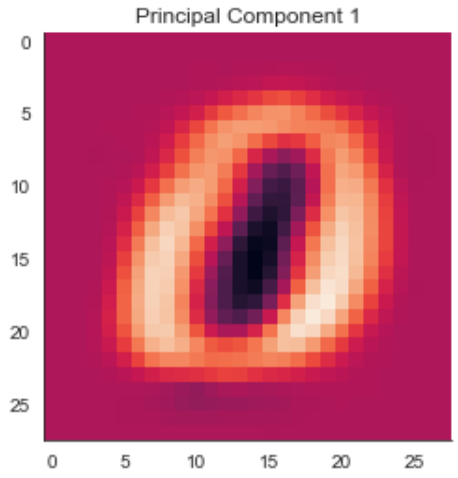

The first 500 principal components
eigenvalue
350000
300000
250000
200000
150000
100000
50000
0
0
100
200
300
400
500
kth principal components

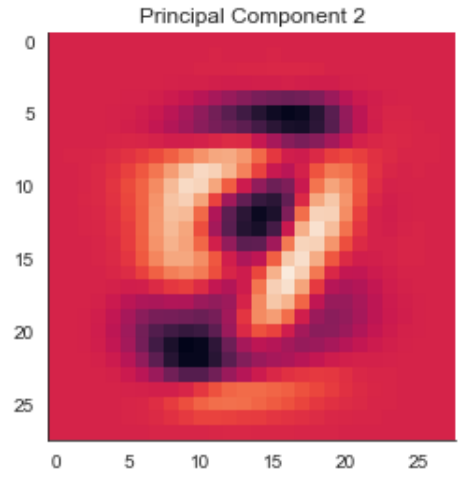

Variance Explained by kth Principal Component
Variance Explained
0.10
0.08
0.06
0.04
0.02
0.00
0
100
200
300
400
500
kth principal component

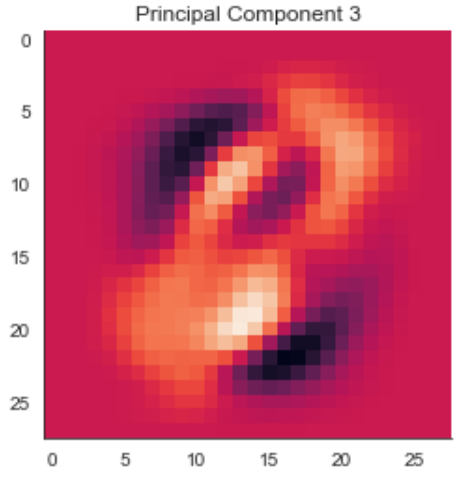

Cumulative Variance Explained by k Principal Components
99.94%
Cumulative Variance Explained
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500
kth principal component

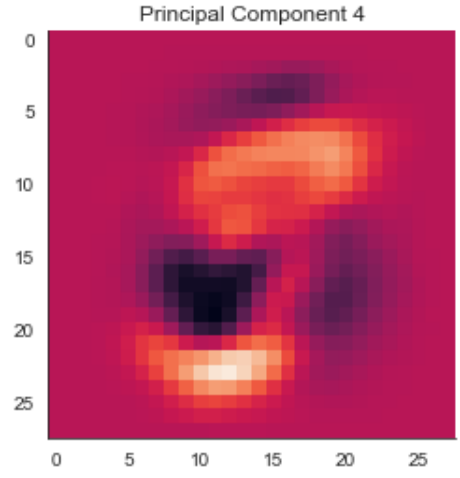
With the first 500 principal components, we are able to capture roughly 99.94% of the variance in the original data. This cumulative variance as we increase the number of components k continually increases but at a slower rate as the ordered components successively explain less of the variance.

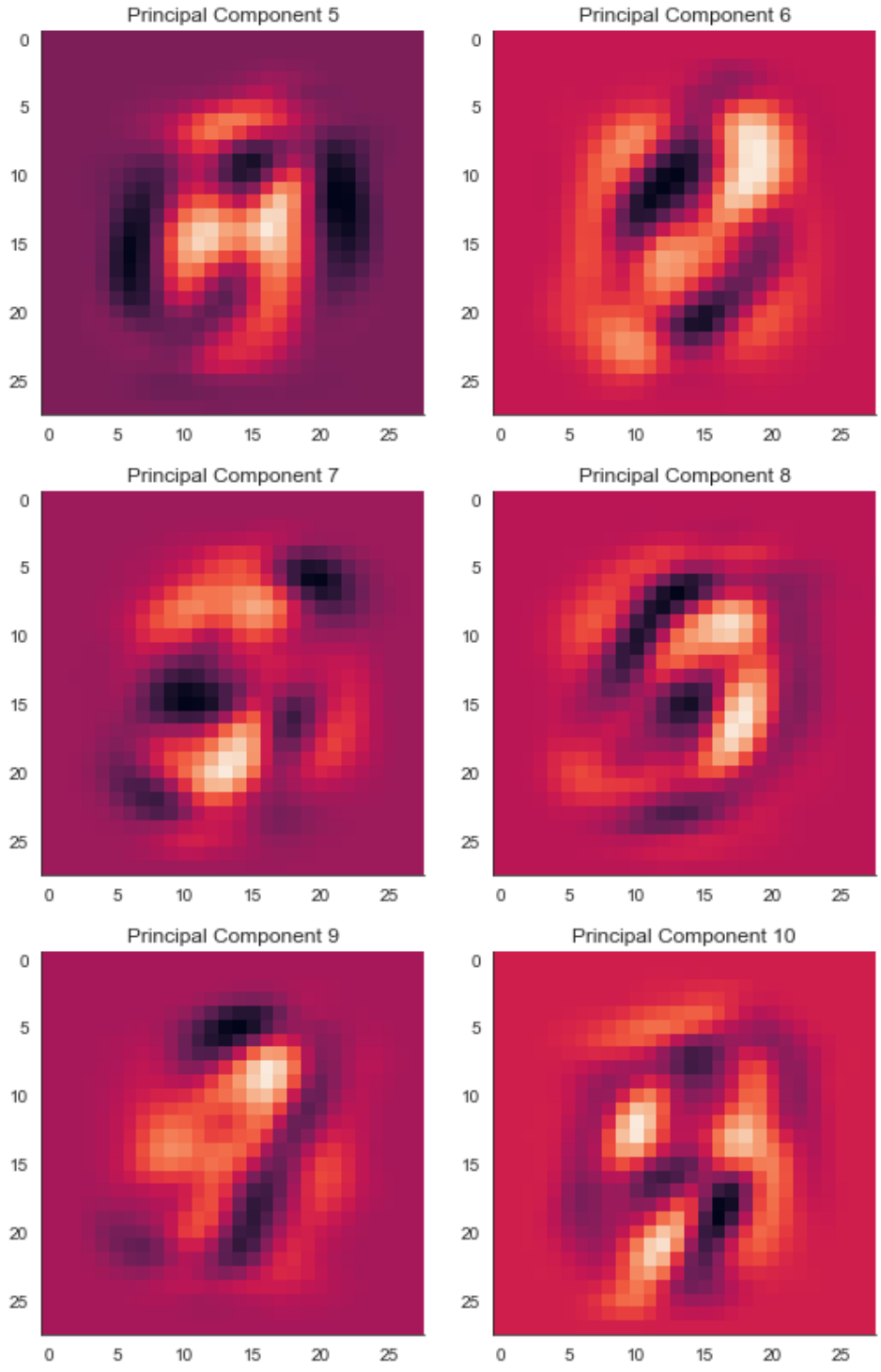












Comparing KMeans and PCA:
Although these two unsupervised methods appear to be different, they share an intuitive underlying similarity. PCA looks to represent the data as a linear combination of k principal components (eigenvectors) and does this by minimizing the reconstruction error. KMeans looks to characterize or represent the data with

k cluster centers; this latter statement can be extrapolated as KMeans looking to represent the data as a linear combination of the cluster centers. KMeans, too, can be shown to work by minimizing a reconstruction loss; the orinigal KMeans loss can be transformed can be equivalently written in other fashion akin to the eigenvecotor search of PCA. This is where the similarities lie, but that does not mean KMeans and PCA are the same. They share an underlying "idea" but are different. The weights on the linear combination of centers in the rephrased objective for KMeans must all be zero except for the particular cluster.

Reconstruction Error for PCA:

```
Reconstruction error
------------------------------
1 principal components: 3942.7106
2 principal components: 3621.2043
3 principal components: 3348.6219
4 principal components: 3107.0959
5 principal components: 2900.9137
6 principal components: 2705.7394
7 principal components: 2560.1568
8 principal components: 2432.0557
9 principal components: 2311.4192
10 principal components: 2208.3104
11 principal components: 2113.7524
12 principal components: 2025.0626
13 principal components: 1951.3542
14 principal components: 1878.5891
15 principal components: 1807.5961
16 principal components: 1739.3382
17 principal components: 1683.3764
18 principal components: 1628.3123
19 principal components: 1575.9508
20 principal components: 1526.1194
```

Reconstruction Error for KMean:
KMean Reconstruction Error: 3201.4207

Comparing the reconstruction errors/losses, we see that the KMeans reconstruction error lies between that error achieved when using 3 and 4 principal components, which capture 23.59% and 29.11% of the variance in the data. Since KMeans can be thought of as a sparse representation of PCA, this makes sense as the KMeans cannot capture much of the variability in the data with just center clusteroids weighted at one component.

- Name: Meriton Ibrahimi
- Email: meritonibrahimi@college.harvard.edu
- Collaborators: N/A
- Approximately how long did this homework take you to complete (in hours): maybe 15