Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

QUESTION 1

Student Name: Aman Tiwari
Roll Number: 160094
Date: April 18, 2019

Solution 1:

Let $\mathbf{v}$ be an eigenvector of the covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$ then we have the following equation:

$\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{v}$

Multiplying the equation on the left by $\mathbf{X}^T$ we get the following:

$(\frac{1}{N}\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{v}) = \lambda(\mathbf{X}^T\mathbf{v})$

This implies that $\mathbf{u} = \mathbf{X}^T\mathbf{v}$ is an eigenvector of $\mathbf{X}^T\mathbf{X}$. So for every eigenvector $\mathbf{v}$ of $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ we have $\mathbf{u} = \mathbf{X}^T\mathbf{v}$ which is an eigenvector of $\frac{1}{N}\mathbf{X}^T\mathbf{X}$.

This means that we need to do eigendecomposition of $N \times N$ matrix instead of $D \times D$ matrix. Since $D > N$ this method is more computationally efficient.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Aman Tiwari
Roll Number: 160094
Date: April 18, 2019

QUESTION
2

Solution 2:

The function $h(x)$ is given as:
$h(x) = \frac{x}{1+exp(-\beta x)}$
If we set $\beta = 0$ then the function reduces to:
$h(x) = \frac{x}{2}$
which is a linear activation function.
If the value of $\beta \to \infty$ then $exp(-\beta x) \to \infty$ for $x < 0$ and $exp(-\beta x) \to 0$ for $x > 0$. So $h(x)$ can approximately be written as:

$$h(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

This is the definition of ReLU function. So $h(x)$ approximates ReLU function when $\beta \to \infty$.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

QUESTION 3

Student Name: Aman Tiwari
Roll Number: 160094
Date: April 18, 2019

Solution 3:

We have the following equations from the question:
$p(z_n = k) = \pi_k$
$p(y_n = 1|\mathbf{x}_n, z_n = k) = \sigma(\mathbf{w}_k^T \mathbf{x}_n) = \frac{1}{1+exp(-\mathbf{w}_k^T \mathbf{x}_n)}$
We can compute the marginal probability $p(y_n = 1|\mathbf{x}_n)$ as follows:
$p(y_n = 1|\mathbf{x}_n) = \sum_{n=1}^{N} p(y_n = 1|\mathbf{x}_n, z_n = k)p(z_n = k)$
$p(y_n = 1|\mathbf{x}_n) = \sum_{n=1}^{N} \frac{\pi_k}{1+exp(-\mathbf{w}_k^T \mathbf{x}_n)}$
We can have the following architecture of neural network which will give the same output:
First Layer: This will be the input layer consisting of D nodes where node d will store the input value x_{nd} , $d = 1, 2, ...D$.
Second layer: This is the hidden layer consisting of K nodes. The values of nodes is given as : $h_{nk} = g(\mathbf{w}_k^T \mathbf{x}_n)$. So the weight matrix between Input layer and hidden layer is a $D \times K$ matrix $\mathbf{W}_1$ where $W_1[d][k] = w_{kd}$. The function g is given as : $g(z) = \sigma(z) = \frac{1}{1+exp(-z)}$.
Last Layer : This consist of a single node . The value s_n of node is given as : $s_n = \sum_{k=1}^{K} \pi_k h_{nk}$. So the weight matrix $\mathbf{W}_2$ between last layer and hidden layer is a $K \times 1$ matrix where $W_2[k][1] = \pi_k$.
The output y_n is given as $y_n = s_n$.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

QUESTION

4

Student Name: Aman Tiwari
Roll Number: 160094
Date: April 18, 2019

Solution 4:

The map objective is given as:
$\hat{\Theta}_{MAP} = \arg\max_{\Theta}(\log p(\mathbf{X}|\Theta) + \log p(\Theta))$
$\hat{\Theta}_{MAP} = \arg\max_{\Theta} \sum_{(n,m)\in\Omega} \log \mathcal{N}(X_{nm}|\Theta) + \sum_{n=1}^{N} \log \mathcal{N}(\mathbf{u}_n|\mathbf{W}_u\mathbf{a}_n, \lambda_u^{-1}\mathbf{I}_K) + \sum_{m=1}^{M} \log \mathcal{N}(\mathbf{v}_m|\mathbf{W}_v\mathbf{a}_m, \lambda_v^{-1}\mathbf{I}_K)$
Therefore the loss function is given by:
$\mathcal{L} = \sum_{(n,m)\in\Omega}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \theta_n + \phi_m))^2\lambda_x + \sum_{n=1}^{N}(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)^T(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)\lambda_u + \sum_{m=1}^{M}(\mathbf{v}_m - \mathbf{W}_v\mathbf{b}_m)^T(\mathbf{v}_m - \mathbf{W}_v\mathbf{b}_m)\lambda_v$
We can now use this loss function to estimate the parameters one by one keeping all others constant.Let us first estimate $\mathbf{u}_n$.We can write the loss function as:
$\mathcal{L} = \sum_{m\in\Omega_{r_n}}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \phi_m + \theta_n))^2\lambda_x + (\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)^T(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)\lambda_u$
Taking derivative and setting it to zero we get:
$\lambda_u(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n) = \sum_{m\in\Omega_{r_n}} \lambda_x(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \phi_m + \theta_n))\mathbf{v}_m$
This gives us :
$\hat{\mathbf{u}}_n = (\sum_{m\in\Omega_{r_n}} \lambda_x\mathbf{v}_m\mathbf{v}_m^T + \lambda_u\mathbf{I}_K)^{-1}(\sum_{m\in\Omega_{r_n}} \lambda_x(X_{nm} - (\phi_m + \theta_n))\mathbf{v}_m + \lambda_u\mathbf{W}_u\mathbf{a}_n)$
Similarly we can solve for $\mathbf{v}_m$. The solution is:
$\hat{\mathbf{v}}_m = (\sum_{n\in\Omega_{c_m}} \lambda_x\mathbf{u}_n\mathbf{u}_n^T + \lambda_v\mathbf{I}_K)^{-1}(\sum_{n\in\Omega_{c_m}} \lambda_x(X_{nm} - (\phi_m + \theta_n))\mathbf{u}_n + \lambda_v\mathbf{W}_v\mathbf{b}_m)$
Now let us estimate θ_n. The loss function can be written as:
$\mathcal{L} = \sum_{m\in\Omega_{r_n}}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \theta_n + \phi_m))^2\lambda_x$
Taking the derivative and setting it to zero we get:
$\sum_{m\in\Omega_{r_n}}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \theta_n + \phi_m))\lambda_x = 0$
This gives us:
$\hat{\theta}_n = \frac{1}{|\Omega_{r_n}|}\sum_{m\in\Omega_{r_n}}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \phi_m))$
Similarly we can slove for ϕ_m to get:
$\hat{\phi}_m = \frac{1}{|\Omega_{c_m}|}\sum_{n\in\Omega_{c_m}}(X_{nm} - (\mathbf{u}_n^T\mathbf{v}_m + \theta_n))$
Now let us estimate $\mathbf{W}_u$. The loss function can be written as:
$\mathcal{L} = \sum_{n=1}^{N}(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)^T(\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n)\lambda_u$
$\mathcal{L} = \sum_{n=1}^{N}(\mathbf{u}_n\mathbf{u}_n^T - \mathbf{a}_n^T\mathbf{W}_u^T\mathbf{u}_n - \mathbf{u}_n^T\mathbf{W}_u\mathbf{a}_n + \mathbf{a}_n^T\mathbf{W}_u^T\mathbf{W}_u\mathbf{a}_n)\lambda_u$
Using results from matrix cookbook we can take the derivative to get:
$\sum_{n=1}^{N} \mathbf{W}_u\mathbf{a}_n\mathbf{a}_n^T = \sum_{n=1}^{N} \mathbf{u}_n\mathbf{a}_n^T$
This gives us:
$\hat{\mathbf{W}}_u = (\sum_{n=1}^{N} \mathbf{u}_n\mathbf{a}_n^T)(\sum_{n=1}^{N} \mathbf{a}_n\mathbf{a}_n^T)^{-1}$
Similarly we can get the value of $\mathbf{W}_v$. The solution is:
$\hat{\mathbf{W}}_v = (\sum_{m=1}^{M} \mathbf{v}_m\mathbf{b}_m^T)(\sum_{m=1}^{m} \mathbf{b}_m\mathbf{b}_m^T)^{-1}$

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 4

Student Name: Aman Tiwari
Roll Number: 160094
Date: April 18, 2019

QUESTION

5

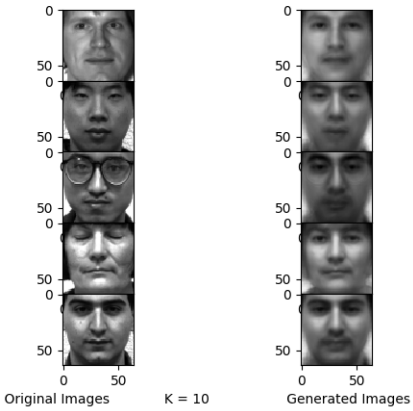
Solution 5:

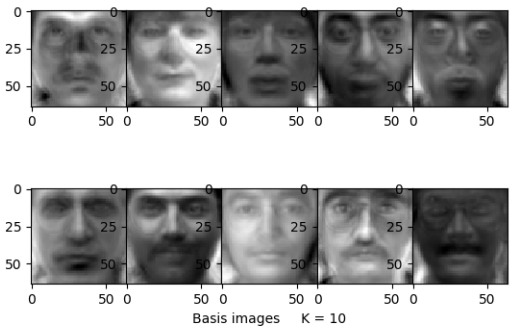
Problem 1:

As the value of K increases the reconstruction gets better. For small values of K reconstructed images only show a rough structure of the original image but lack the details of the image. This is because embeddings of smaller dimensions will learn less features of the input so as embedding dimension increases reconstruction gets better.

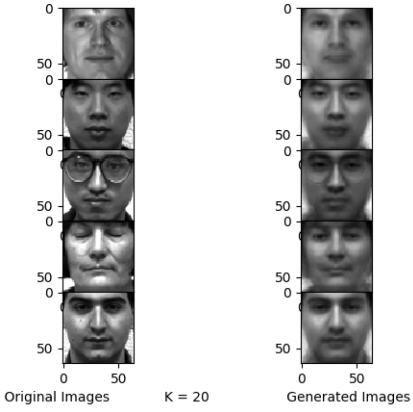
The images obtained for various values of K are as follows:

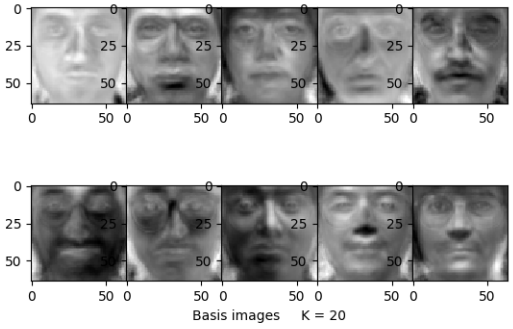
For K = 10:



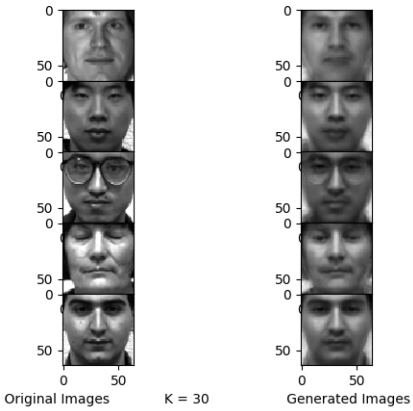


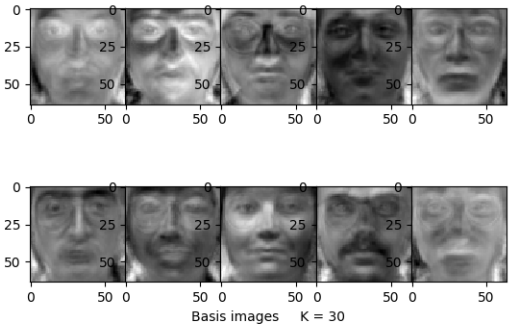
For K = 20:



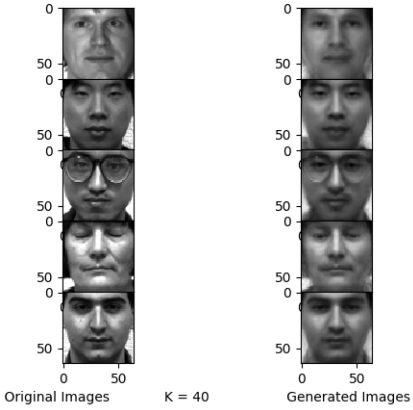


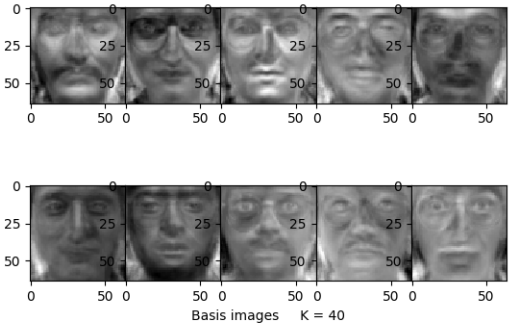
For K = 30:



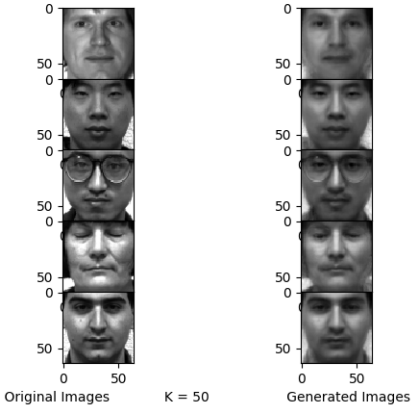


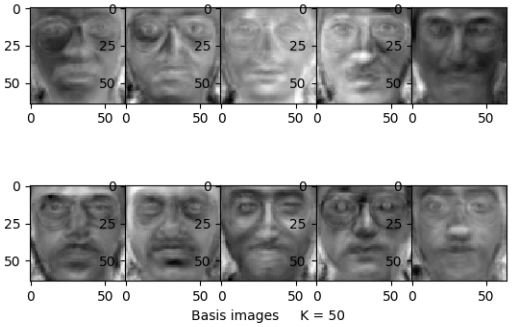
For K = 40:



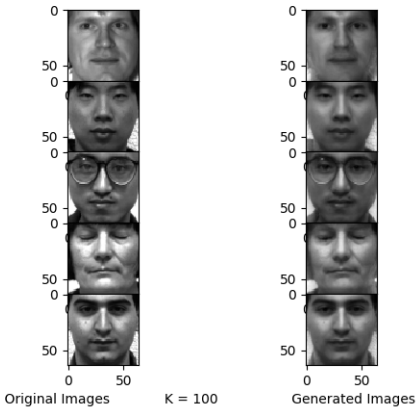


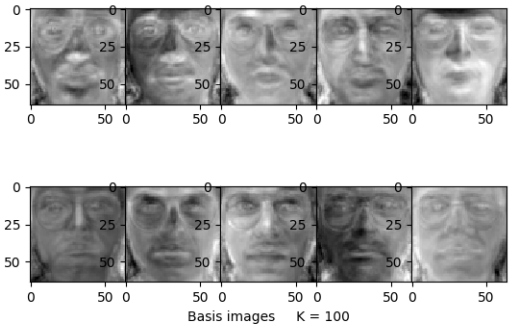
For K = 50:





For K = 100:

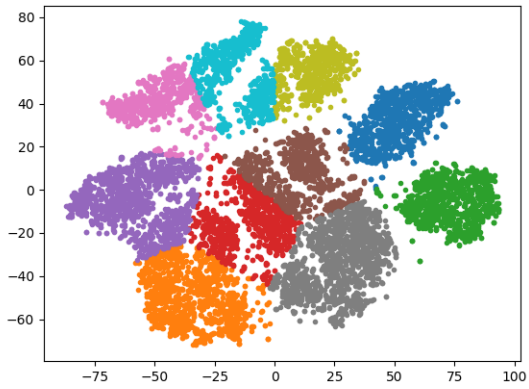


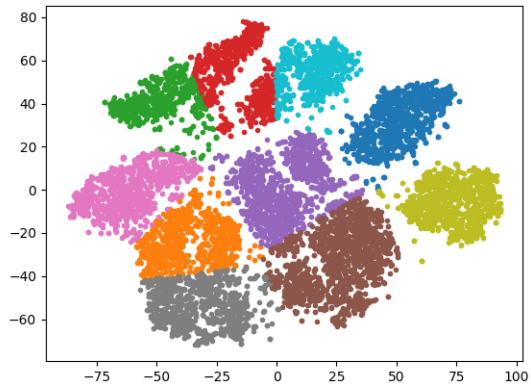


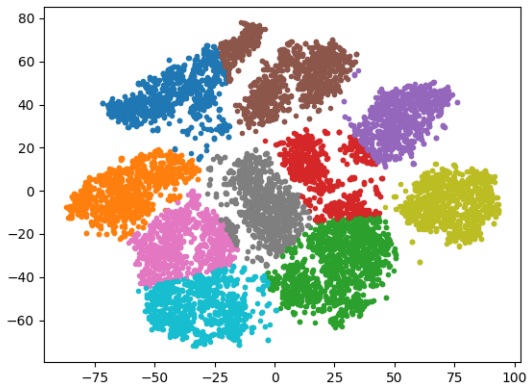
Problem 2:

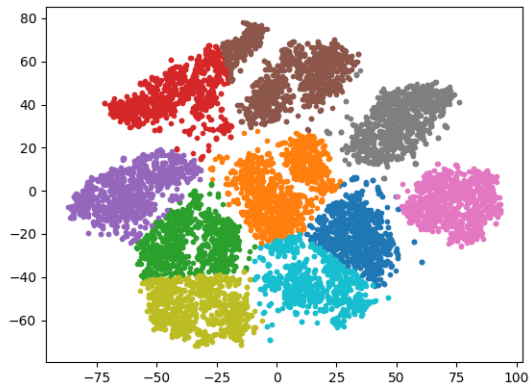
In this case tSNE works better as compared to PCA for clustering.The clusters are separated in case of tSNE whereas in case of PCA they are very close to eac other.

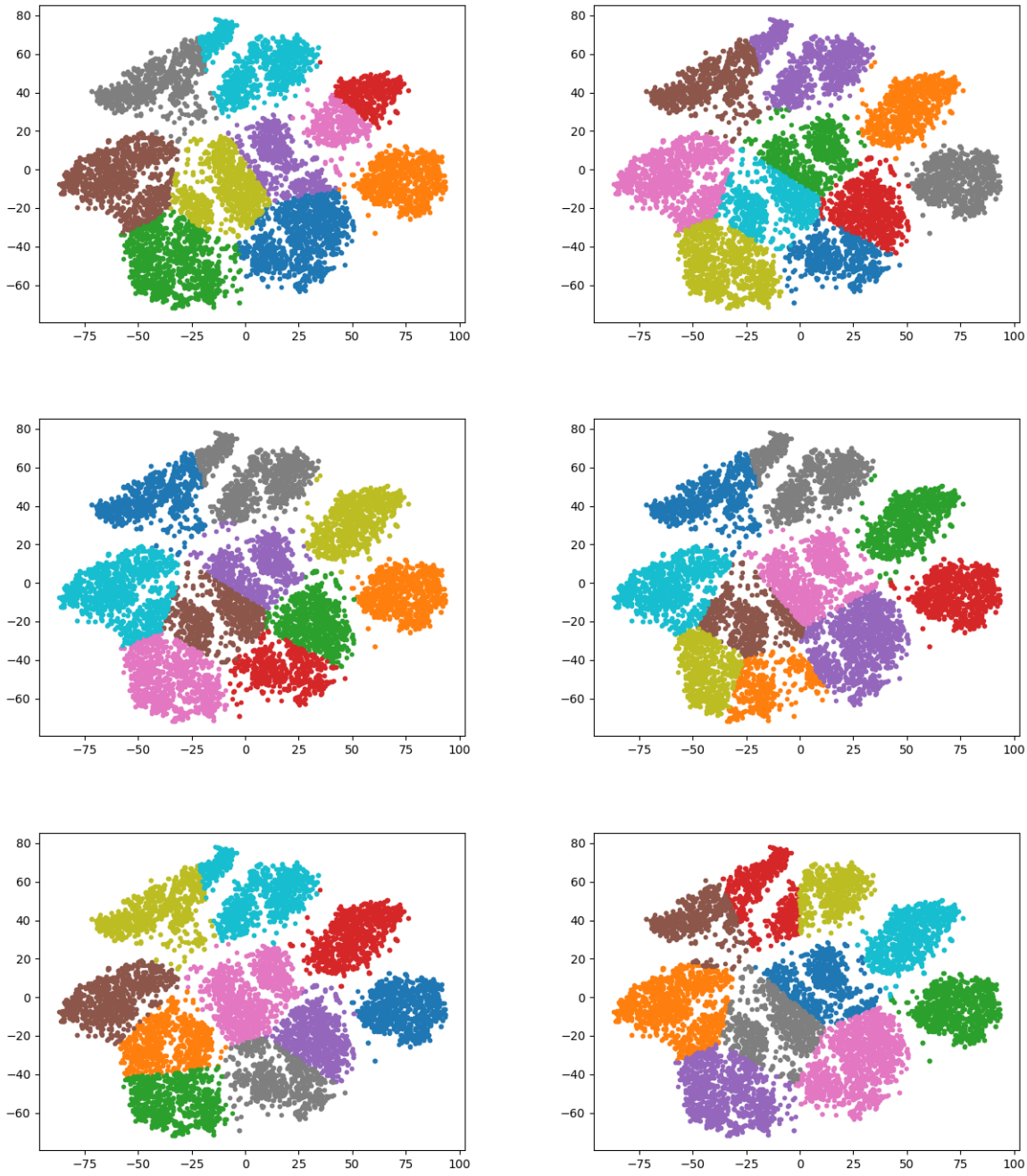
The plots obtained for tSNE for 10 random initialization of KMeans are as follows:



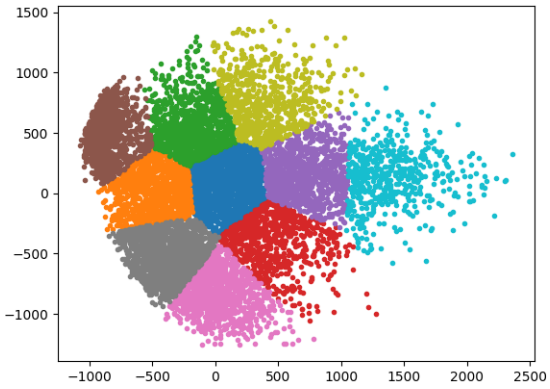
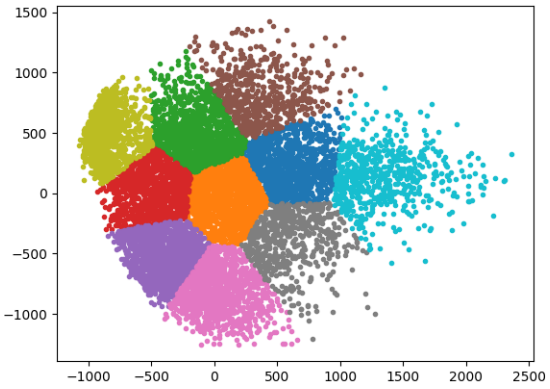
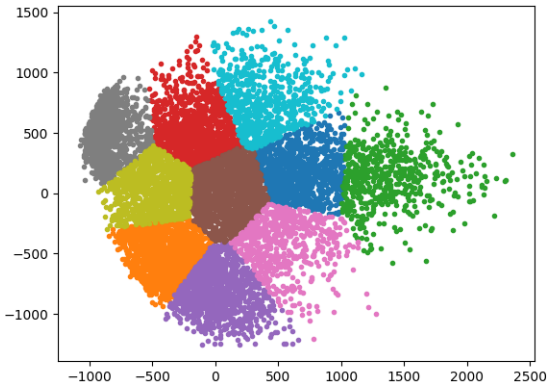
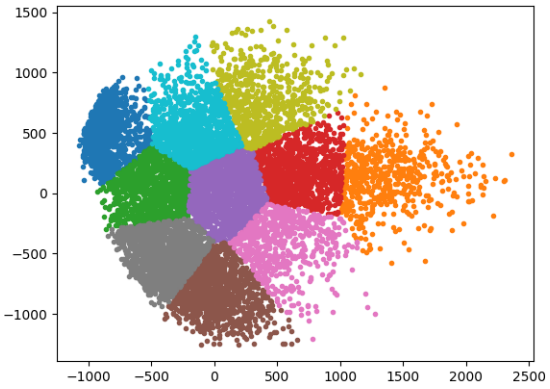
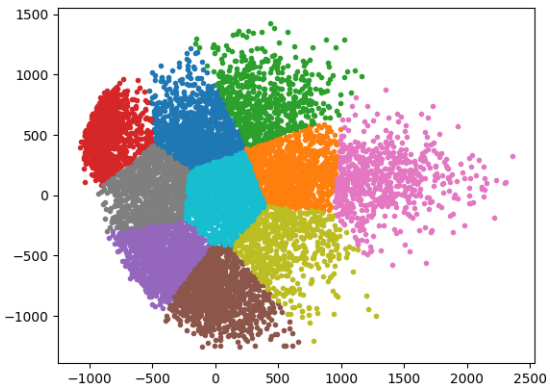
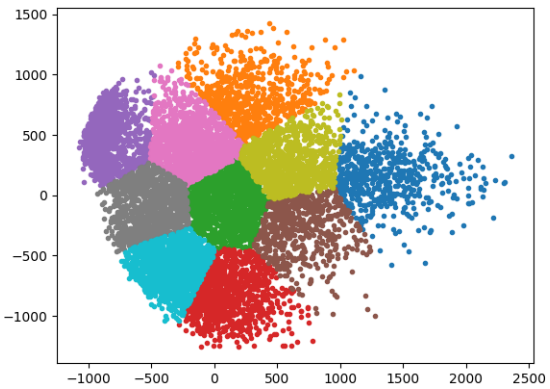


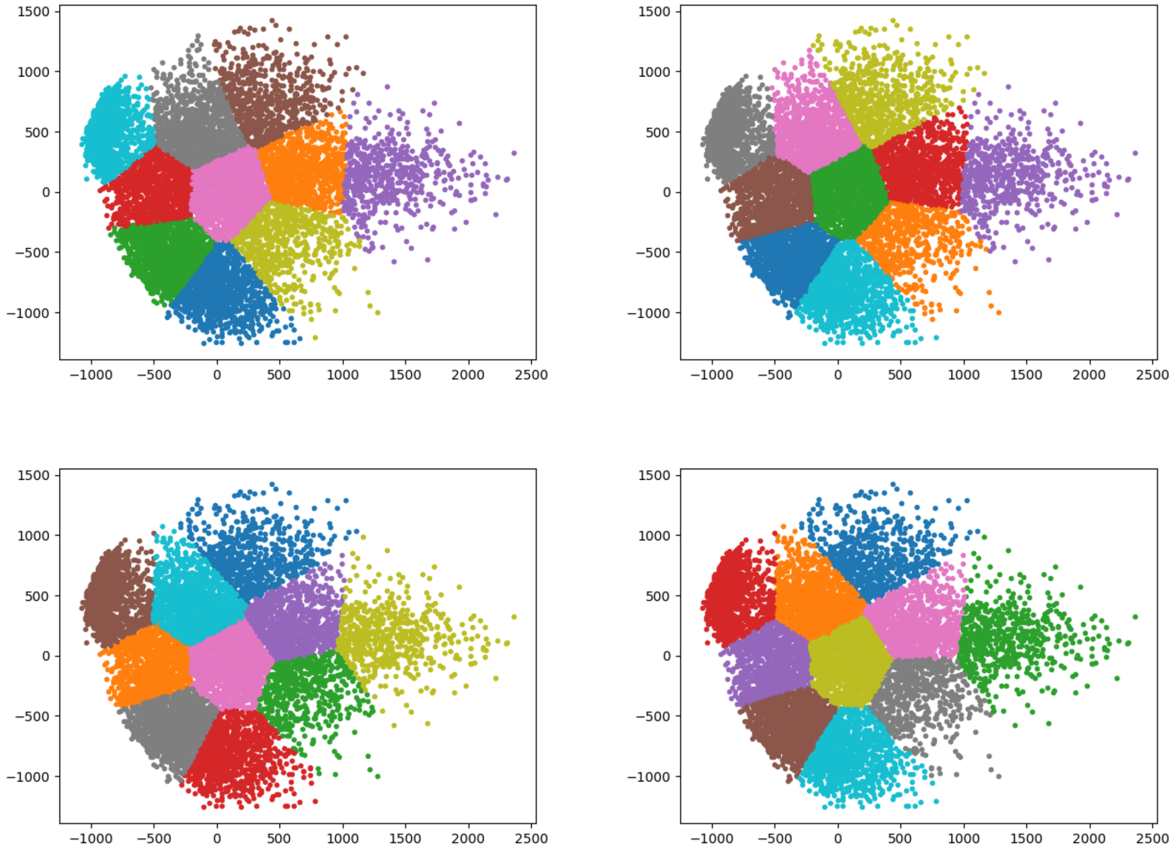






The plots obtained for PCA for 10 random initialization of KMeans are as follows:


1500
1000
500
0
−500
−1000
−1000 −500 0 500 1000 1500 2000 2500