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

QUESTION 1

Student Name: Piyush Bagad
Roll Number: 150487
Date: November 17, 2018

Let X be the $N \times D$ data matrix. The covariance matrix is given by (assuming centered data) $S = \frac{1}{N}X^TX$. Let $T = \frac{1}{N}XX^T$. Suppose λ is an eigenvalue and $\mathbf{v}$ is an eigenvector of T, then my claim is that λ is also an eigenvalue of S with coressponding eigenvector being $X^T\mathbf{v}$. The proof follows:

$$T\mathbf{v} = \lambda\mathbf{v}$$

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

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

$$\therefore S(X^T\mathbf{v}) = \lambda(X^T\mathbf{v})$$

Let $\mathbf{u} := X^T\mathbf{v}$. We have $S\mathbf{u} = \lambda\mathbf{u}$. Therefore, $\mathbf{u}$ turns out to be the eigenvector for S coressponding to the eigenvalue λ. Thus, if we know eigenvectors of matrix T, we can find the eigenvectors of matrix S by simple matrix multiplication which is $O(ND)$. The advantage of using this approach to obtain the eigenvectors is that we will need to diagonalize the $N \times N$ matrix T instead of the $D \times D$ matrix S for getting eigenvectors. Note that we have been given $N < D$. Thus, it is computationally cheaper to obtain eigenvectors in this manner when $D > N$. Also, note that we can kernelize the matrix T and then conduct eigendecomposition of the kernel matrix enabling us to do non-linear PCA.

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

Student Name: Piyush Bagad
Roll Number: 150487
Date: November 17, 2018

QUESTION 2

Given $h(x) = x\sigma(\beta x)$ where σ is the usual sigmoid activation function.

$$h(x) = \frac{x}{1 + \exp(-\beta x)}$$

Case 1: Linear approximation

Take $\beta = 0$, we will have

$$h(x) = \frac{x}{2} \quad \text{[Linear]}$$

Case 2: Approximating ReLU

Take $\beta \to \infty$, we will have $\exp(-\beta x) \to \infty$ for all $x < 0$ and $\exp(-\beta x) \to 0$ for $x \geq 0$. Thus, we get

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

Hence, we can approximate ReLU using given activation function $h(x)$.

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

Student Name: Piyush Bagad
Roll Number: 150487
Date: November 17, 2018

QUESTION 3

We have been given data $\{(\mathbf{x}_n, y_n)\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^2, y_n \in \{0,1\}$. We are given

$$z_n \sim \text{multinoulli}(\pi_1, \pi_2, .., \pi_K)$$

$$y_n \sim \text{Bernoulli}(\sigma(w_{z_n}^T \mathbf{x}_n))$$

We want to estimate $p(y_n = 1|\mathbf{x}_n)$:

$$p(y_n = 1|\mathbf{x}_n) = \sum_{k=1}^{K} p(y_n = 1, z_n = k|\mathbf{x}_n) = \sum_{k=1}^{K} p(y_n = 1|z_n = k, \mathbf{x}_n)p(z_n = k) = \sum_{k=1}^{K} \sigma(w_k^T \mathbf{x}_n)\pi_k$$

$$\boxed{p(y_n = 1|\mathbf{x}_n) = \sum_{k=1}^{K} \sigma(w_k^T \mathbf{x}_n)\pi_k}$$

We can think of this as a neural network in the following way:

- **Input layer**: $(x_1, x_2, .., x_D), \mathbf{x} \in \mathbb{R}^D$ is the input example.
- **Hidden layer**: We consider a single hidden layer of size K with each hidden node k having output $\sigma(w_k^T \mathbf{x})$. The weight matrix will be $\mathbf{W} = [w_{ij}], i \in \{1, 2, .., D\},\ j \in \{1, 2., , K\}$ i.e. $\mathbf{W} = [w_1^T\ w_2^T\ ...\ w_K^T]$.
- **Final layer**: The output layer will consist of only a single node whose output will be $p(y = 1|\mathbf{x})$. The weight vector $\mathbf{U} = [\pi_1\ \pi_2\ ...\ \pi_K]^T$.

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

QUESTION 4

Student Name: Piyush Bagad
Roll Number: 150487
Date: November 17, 2018

Consider an $N \times M$ rating matrix X, where the rows represents the N users and the columns represent the M items. We are also given some side information: for each user n, a feature vector $\mathbf{a}_n \in \mathbb{R}^{D_U}$, and for each item m, a feature vector $\mathbf{b}_m \in \mathbb{R}^{D_I}$. Let $\mathbf{u}_n$, $\mathbf{v}_m \in \mathbb{R}^K$ represent the latent factors for user n and item m respectively. Let θ_n be user bias and ϕ_m be the popularity of item. We have

$$p(X_{nm}|\mathbf{u}_n, \theta_n, \mathbf{v}_m, \phi_m) = \mathcal{N}(X_{nm}|\theta_n + \phi_m + \mathbf{u}_n^T\mathbf{v}_m, \lambda^{-1})$$
$$p(\mathbf{u}_n) = \mathcal{N}(\mathbf{u}_n|\mathbf{W}_u\mathbf{a}_n, \lambda_u^{-1}\mathbf{I}_K)$$
$$p(\mathbf{v}_m) = \mathcal{N}(\mathbf{v}_m|\mathbf{W}_v\mathbf{b}_m, \lambda_v^{-1}\mathbf{I}_K)$$

Assume $\Omega = \{(n, m)\}$ to denote the set of indices of the observed entries of X, Ω_{r_n} to be the set of items rated by user n, and Ω_{c_m} to be the set of users who rated item m. Let $\Theta := \{\{(\mathbf{u}_n, \theta_n)\}, \{\mathbf{v}_m, \phi_m\}, \mathbf{W}_u, \mathbf{W}_v\}$.

The MAP objective can be written as follows:

$$\hat{\Theta}_{MAP} = \arg\max_{\Theta}\{\log(p(X|\Theta)) + \log(p(\Theta))\}$$

$$\log(p(X|\Theta)) = \sum_{(n,m)\in\Omega} \log(p(X_{nm}|\Theta)) = -\frac{\lambda}{2}\sum_{(n,m)\in\Omega}(X_{nm} - (\theta_n + \phi_m + \mathbf{u}_n^T\mathbf{v}_m))^2$$

$$\log(p(\Theta)) = \sum_{n=1}^{N}\log(p(\mathbf{u}_n)) + \sum_{m=1}^{M}\log(p(\mathbf{v}_m)) = -\frac{\lambda_u}{2}\sum_{n=1}^{N}\|\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n\|^2 - \frac{\lambda_v}{2}\sum_{m=1}^{M}\|\mathbf{v}_m - \mathbf{W}_v\mathbf{b}_m\|^2$$

Thus, the consolidated loss can be written as

$$\mathcal{L}(\Theta) = \frac{\lambda}{2}\sum_{(n,m)\in\Omega}(X_{nm} - (\theta_n + \phi_m + \mathbf{u}_n^T\mathbf{v}_m))^2 + \frac{\lambda_u}{2}\sum_{n=1}^{N}\|\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n\|^2 + \frac{\lambda_v}{2}\sum_{m=1}^{M}\|\mathbf{v}_m - \mathbf{W}_v\mathbf{b}_m\|^2$$

Optimizing using ALT-OPT

- **Estimating latent variables and parameters for users**: Let us keep $\{\{\hat{\mathbf{v}}_m, \hat{\phi}_m\}, \hat{\mathbf{W}}_v\}$ fixed and estimate the remaining variables. We will only consider relevant terms in the loss function.

$$\mathcal{L}(\Theta) = \frac{\lambda}{2}\sum_{(n,m)\in\Omega}(X_{nm} - (\theta_n + \phi_m + \mathbf{u}_n^T\mathbf{v}_m))^2 + \frac{\lambda_u}{2}\sum_{n=1}^{N}\|\mathbf{u}_n - \mathbf{W}_u\mathbf{a}_n\|^2$$

1. Estimating $\mathbf{u}_n$ keeping $\theta_n, \mathbf{W}_u$ fixed: Note that this will turn out to be simple a least-squares problem with the prior on $\mathbf{u}_n$ being non-zero mean. Thus, we can write the solution in closed form as follows:

$$\mathbf{u}_n = \left(\sum_{m\in\Omega_{r_n}}\mathbf{v}_m^T\mathbf{v}_m + \lambda_u\mathbf{I}_K\right)^{-1}\left(\lambda_u\mathbf{W}_u\mathbf{a}_n + \lambda\sum_{m\in\Omega_{r_n}}(X_{nm} - \theta_n - \phi_m)\mathbf{v}_m\right)$$

2. Estimating θ_n keeping $\mathbf{u}_n, \mathbf{W}_u$ fixed: Now, we can simply set the derivative to 0 and obtain θ_n as follows:

$$\theta_n = \frac{\sum_{m \in \Omega_{r_n}} (X_{nm} - \phi_m - \mathbf{u}_n^T \mathbf{v}_m)}{\sum_{m \in \Omega_{r_n}} 1}$$

3. Estimating $\mathbf{W}_u$ keeping $\theta_n, \mathbf{u}_n, \forall n$ fixed: This time we get the loss function which resembles the loss of a multi-output linear regression problem. Thus, we can write the solution as follows -

$$\mathbf{W}_u = \left(\sum_{n=1}^{N} \mathbf{u}_n \mathbf{a}_n^T \right) \left(\sum_{n=1}^{N} \mathbf{a}_n \mathbf{a}_n^T \right)^{-1}$$

- **Estimating latent variables and parameters for items**: Similar procedure (as for estiminating user variables and paramters) can be followed for items.

1. Estimating $\mathbf{v}_m$ keeping $\phi_m, \mathbf{W}_v$ fixed:

$$\mathbf{v}_m = \left(\sum_{n \in \Omega_{c_m}} \mathbf{u}_n^T \mathbf{u}_n + \lambda_v \mathbf{I}_K \right)^{-1} \left(\lambda_v \mathbf{W}_v \mathbf{b}_m + \lambda \sum_{n \in \Omega_{c_m}} (X_{nm} - \theta_n - \phi_m) \mathbf{u}_n \right)$$

2. Estimating ϕ_m keeping $\mathbf{v}_m, \mathbf{W}_v$ fixed:

$$\phi_m = \frac{\sum_{n \in \Omega_{c_m}} (X_{nm} - \theta_n - \mathbf{u}_n^T \mathbf{v}_m)}{\sum_{n \in \Omega_{c_m}} 1}$$

3. Estimating $\mathbf{W}_v$ keeping $\phi_m, \mathbf{v}_m, \forall m$ fixed:

$$\mathbf{W}_v = \left(\sum_{m=1}^{M} \mathbf{v}_m \mathbf{b}_m^T \right) \left(\sum_{m=1}^{M} \mathbf{b}_m \mathbf{b}_m^T \right)^{-1}$$

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

Student Name: Piyush Bagad
Roll Number: 150487
Date: November 17, 2018

QUESTION 5

Part 1: PPCA using ALT-OPT

The follwoing section shows visual results for the PPCA task for each of the values of $K \in \{10, 20, 30, 40, 50, 100\}$. I observed that the image quality certainly improves on increasing K. As evident from the results for $K = 100$, it almost exactly reproduces the original images. Also, I observed that on further increasing K, the quality dropped down possibly due to overfitting. Appropriate regularization helped in avoiding overfitting. For each K, I have also shown the first 10 columns of matrix W. It resembles the templates of images which the model will be using to reconstruct original images as a linear combination. For K =10, it seems that W has a very rough structure which does not focus on the minor details of the faces but only focusses on the high-level characteristics. Whereas for higher K, the details included in the template images seem to increase. It seems that for higher K, each of the columns of W is a better template to reconstruct many original images. In other words, each column of W is formed from knowledge about multiple original images.

Figure 1: Reconstructed images of randomly chosen samples for K = 10.

Figure 2: Basis images for K = 10.

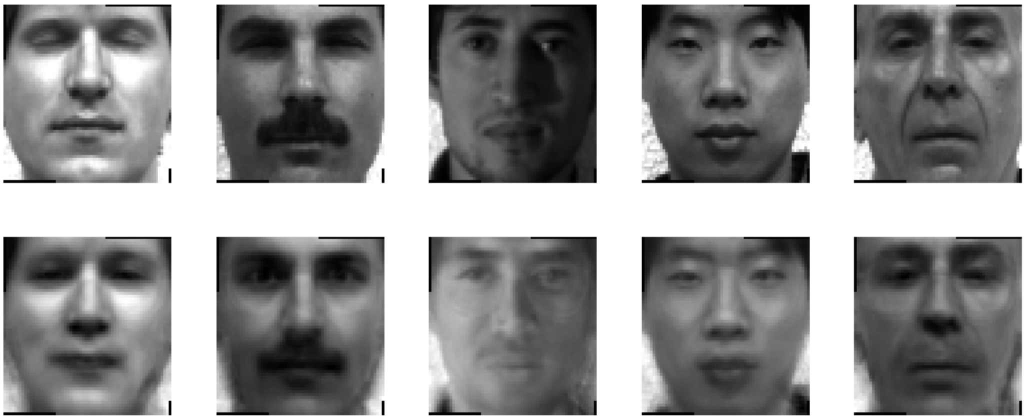


Figure 3: Reconstructed images of randomly chosen samples for K = 20.

Figure 4: Basis images for K = 20.

Figure 5: Reconstructed images of randomly chosen samples for K = 30.

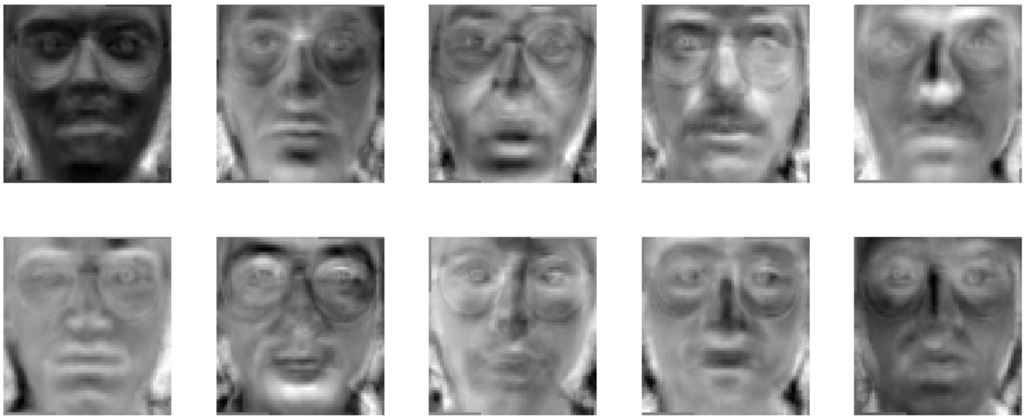


Figure 6: Basis images for K = 30.

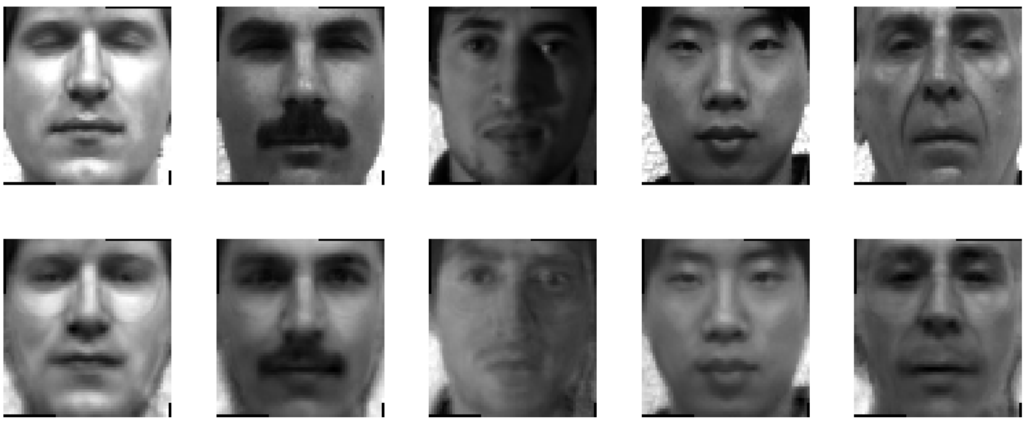


Figure 7: Reconstructed images of randomly chosen samples for K = 40.

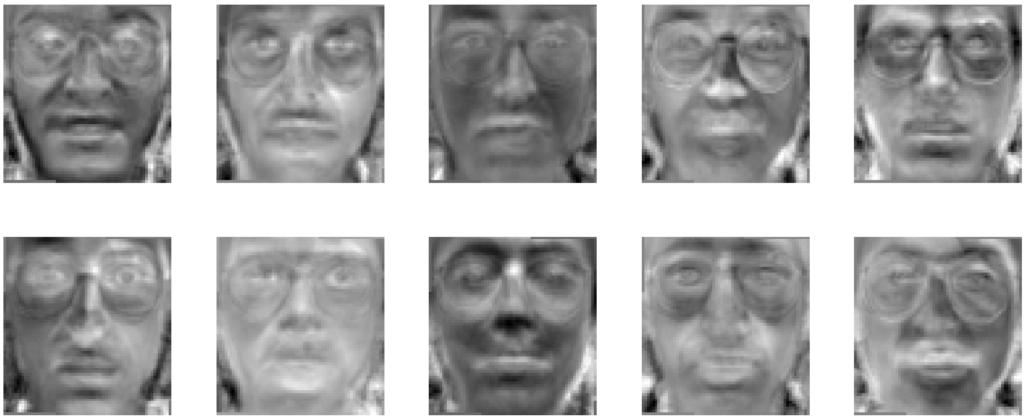


Figure 8: Basis images for K = 40.

Figure 9: Reconstructed images of randomly chosen samples for K = 50.

Figure 10: Basis images for K = 50.

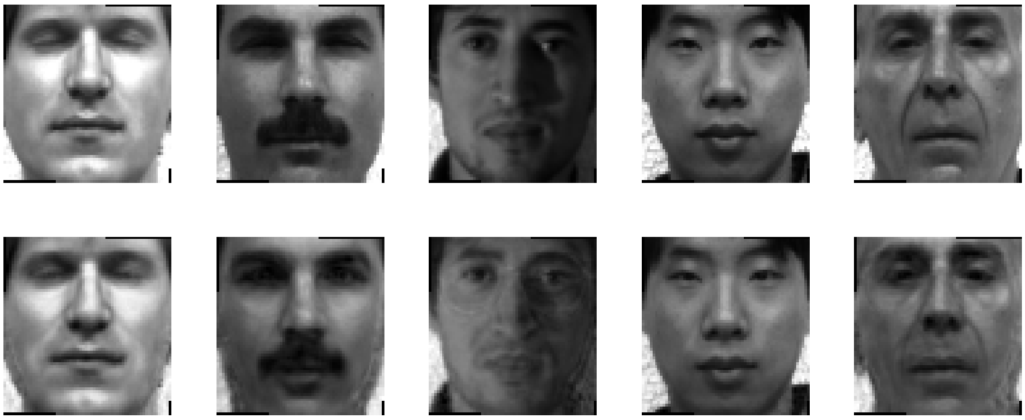


Figure 11: Reconstructed images of randomly chosen samples for K = 100.

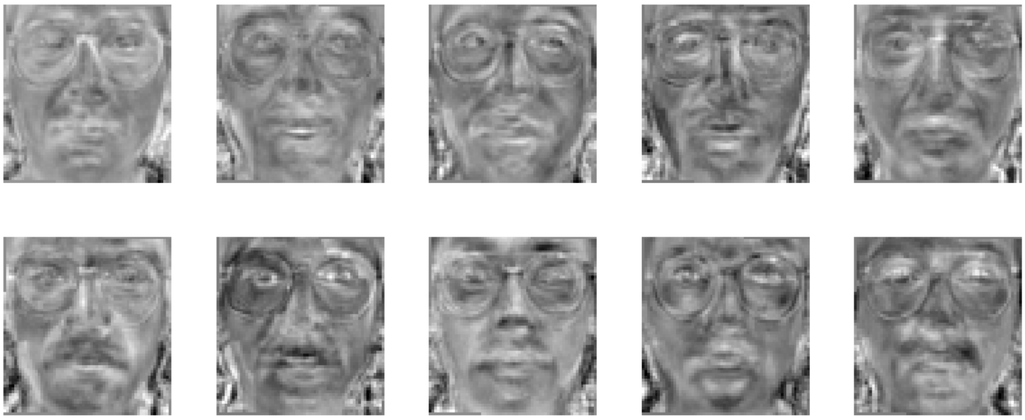


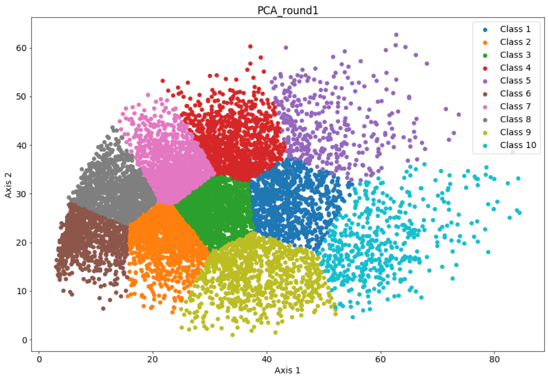
Figure 12: Basis images for K = 100.

Part 2: kMeans Clustering

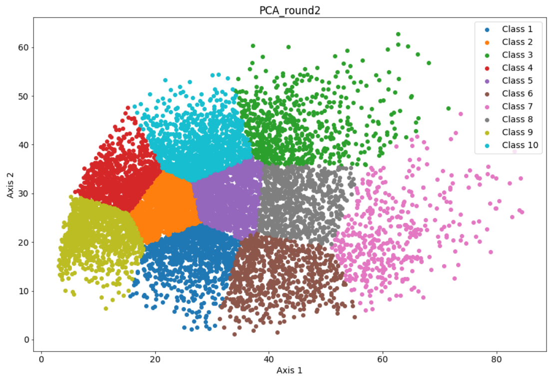
This section shows clustering results on the 2D embeddings produced by either PCA or TSNE. The clustering seem to be better for tSNE primarily because:

- It focusses more on pair wise distances and tries to preserve local structure. Here, say for instance, all images coressponding to 0 will look similar as against those for 1 or tohers. Thus, even in the embedding space they must be closer. This is evident from the plots.
- kMeans assumes approximately similar cluster sizes and shapes. However, PCA since it does not focus on local neighborhoods given sort of a homogenous embedded dataset with hardly any distinctions across classes.

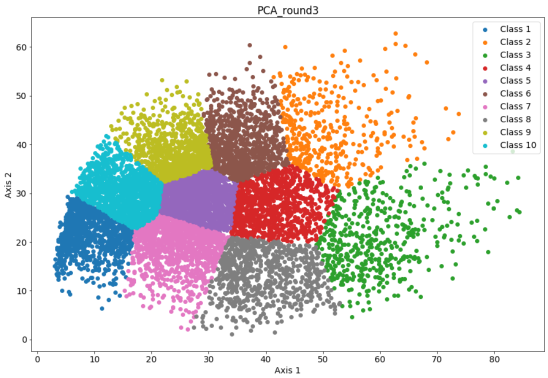
A] With PCA: 10 random initializations



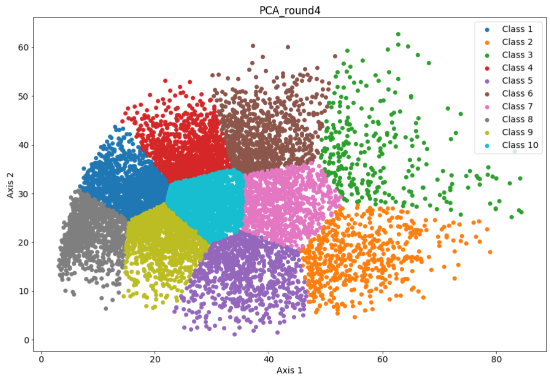
(a) Clustering with PCA: Round 1



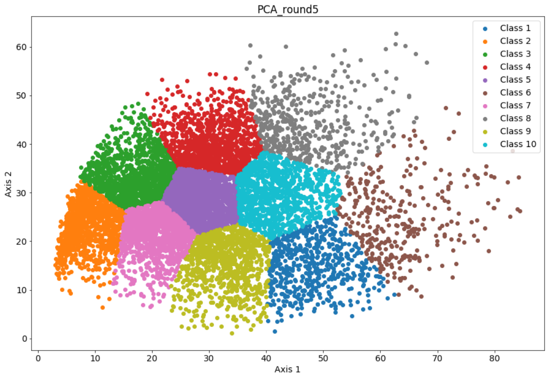
(b) Clustering with PCA: Round 2



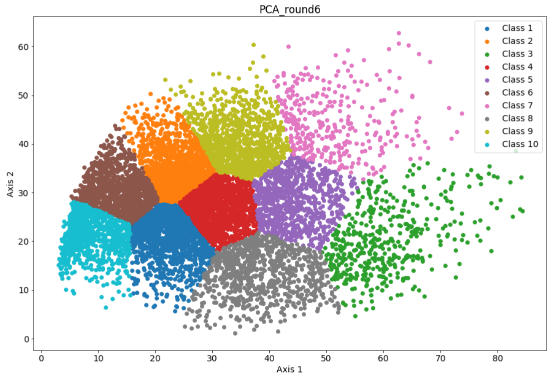
(a) Clustering with PCA: Round 3



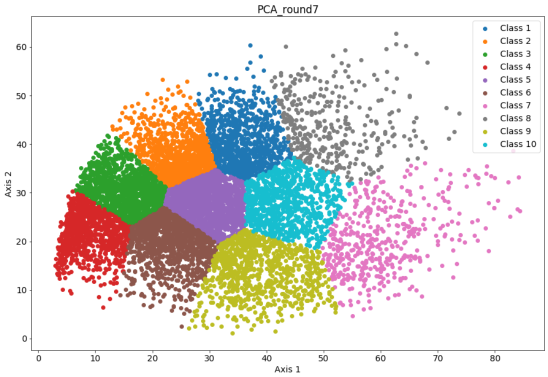
(b) Clustering with PCA: Round 4



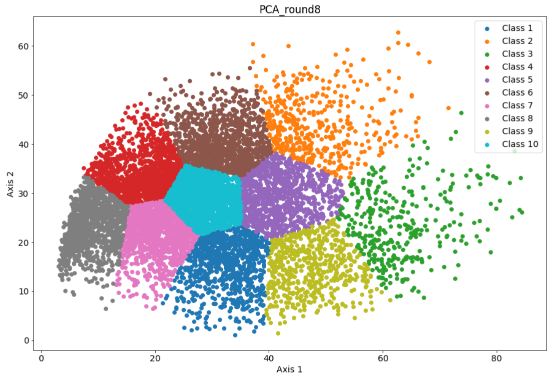
(a) Clustering with PCA: Round 5



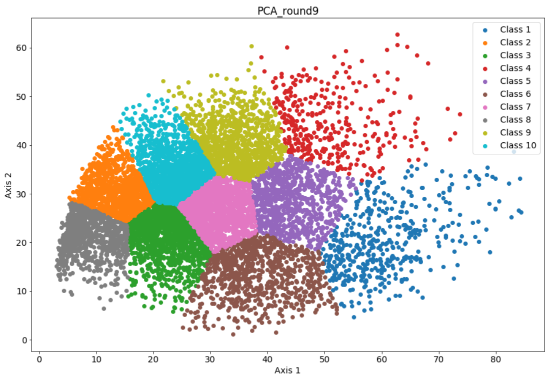
(b) Clustering with PCA: Round 6



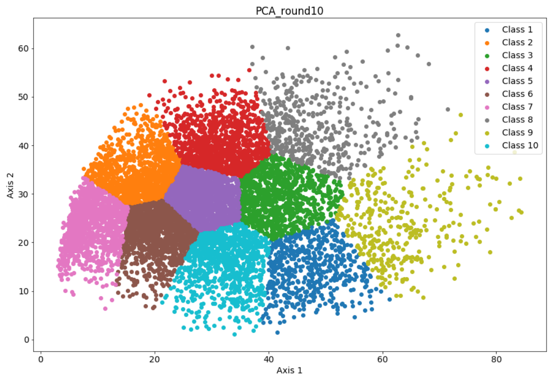
(a) Clustering with PCA: Round 7



(b) Clustering with PCA: Round 8

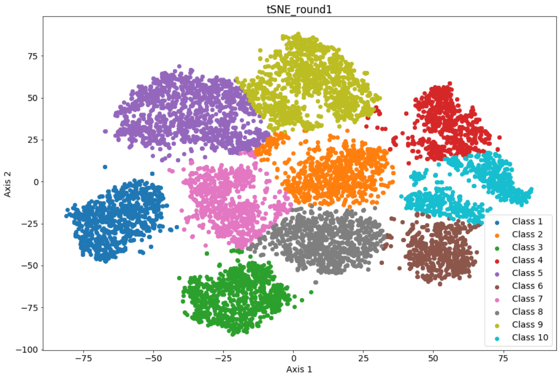


(a) Clustering with PCA: Round 9

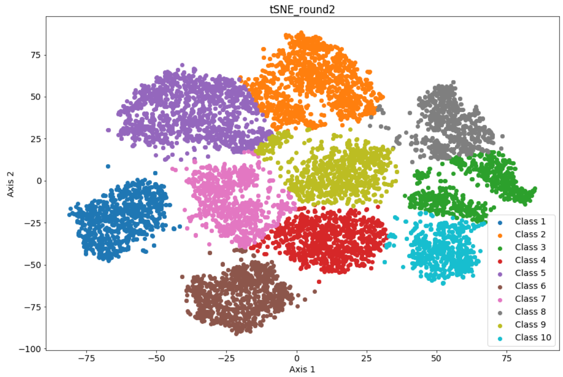


(b) Clustering with PCA: Round 10

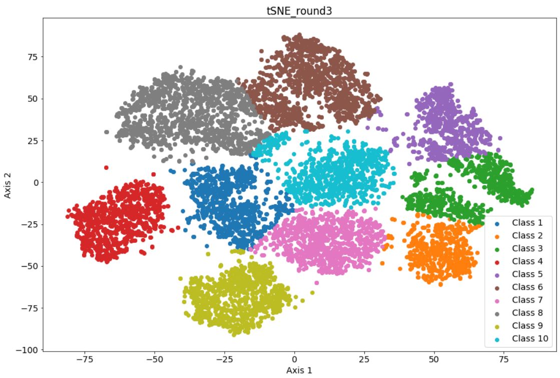
B] With t-SNE: 10 random initializations



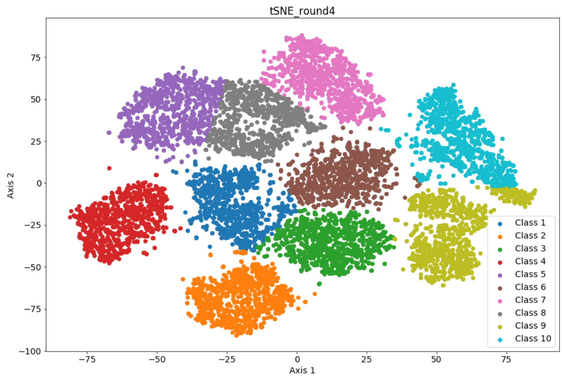
(a) Clustering with tSNE: Round 1



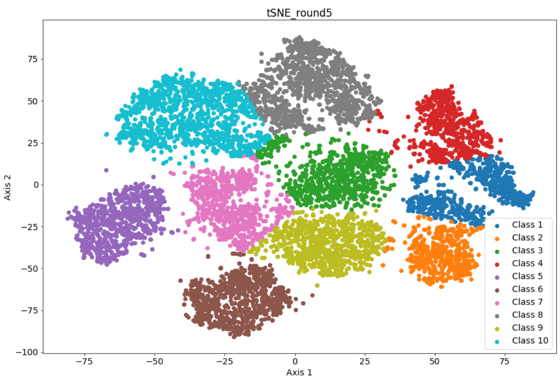
(b) Clustering with tSNE: Round 2



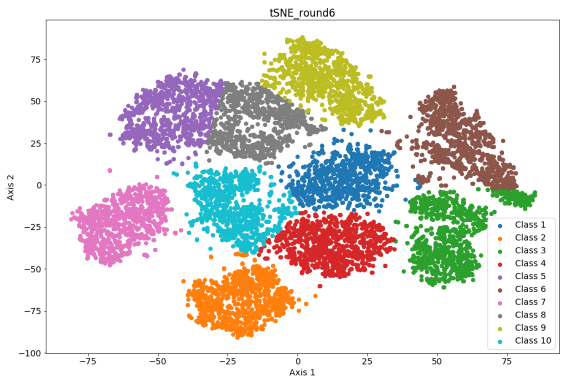
(a) Clustering with tSNE: Round 3



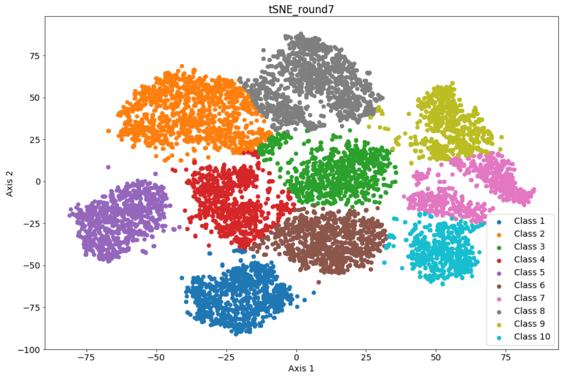
(b) Clustering with tSNE: Round 4



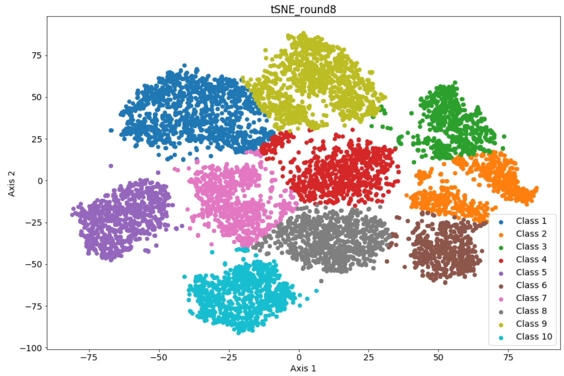
(a) Clustering with tSNE: Round 5



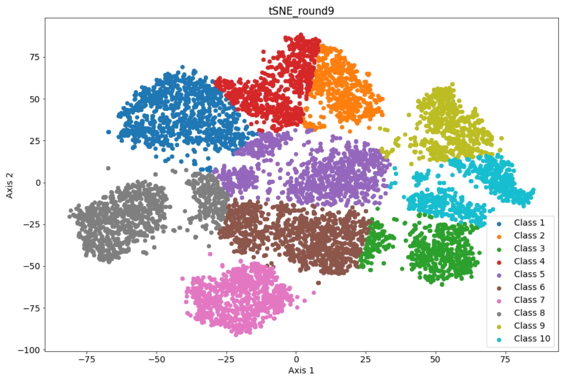
(b) Clustering with tSNE: Round 6



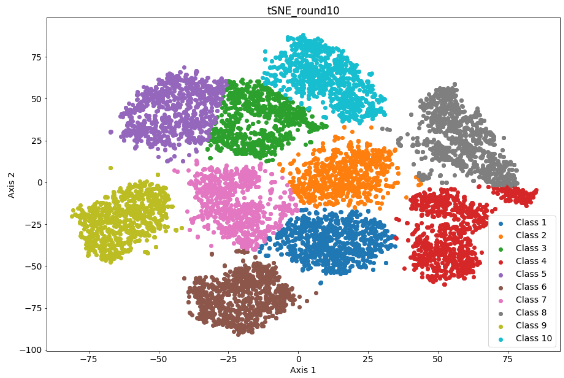
(a) Clustering with tSNE: Round 7



(b) Clustering with tSNE: Round 8



(a) Clustering with tSNE: Round 9



(b) Clustering with tSNE: Round 10