HW3- CSCE 636

YUTAE LEE

Date: November 8th, 2022.

Non-coding part:

1. Question 1: Singular Value Decomposition

Our goal is to get the SVD of the matrix A where

$$A = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \\ u_{13} & u_{23} & u_{33} \end{bmatrix}$$

Currently we can clearly see that A is decomposed by its eigenvector.

Where $A = U\Lambda U^T$

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix},$$

$$\Lambda = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

Since the singular value decomposition has to be in the format $U\Sigma V^T$ where Σ is diagonal with nonnegative entries (also entries decreasing).

Which mean we can modify the above eigenvalue decomposition to transform into Singular Value Decomposition.

This step can be easily done by setting

$$\Sigma = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, U = \begin{bmatrix} u_{11} & -u_{12} & u_{13} \\ u_{21} & -u_{22} & u_{23} \\ u_{31} & -u_{32} & u_{33} \end{bmatrix}, \mathrm{V}^T = \begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \\ u_{13} & u_{23} & u_{33} \end{bmatrix}$$

Thus,

$$U\Sigma V^T =$$

$$\begin{bmatrix} u_{11} & -u_{12} & u_{13} \\ u_{21} & -u_{22} & u_{23} \\ u_{31} & -u_{32} & u_{33} \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \\ u_{13} & u_{23} & u_{33} \end{bmatrix}$$

2. Question 2: Ky Fan Theorem Proof

Proof. Ky Fan

Suppose $H = U\Lambda U^T$ by Eigenvalue decomposition

Our goal is to prove that :

$$\lambda_1 + \cdots + \lambda_k = \max_{A \in \mathbb{R}^{n\times k}: A^T A = I_k} tr(A^T HA)$$

Transforming $A^T HA$ using $H = U\Lambda U^t$

$$\max_{A \in \mathbb{R}^{n\times k}: A^T A = I_k} tr(A^T HA) = tr(A^T H = A^T U\Lambda U^T A)$$

$$= \max_{A \in \mathbb{R}^{n\times k}: A^T A = I_k} tr((U^T A)^T \Lambda (U^T A))$$

Let $P = U^T A$

Then we can say that $P^T P = A^T U \times U^T A = A^T \times I \times A = I$

This is caused because we know that U is orthogonal because they are eigenvectors of Symmetric Matrix H.

Transforming the equation in terms of P:

$$\max_{A \in \mathbb{R}^{n\times k}: A^T A = I_k} tr((U^T A)^T \Lambda (U^T A)) = tr(P^T \Lambda P)$$

By the associative property of trace, we can reorder $P^T \Lambda P$ to $\Lambda P P^T$

$$\max_{A \in \mathbb{R}^{n\times k}: A^T A = I_k} tr(P^T \Lambda P) = tr(\Lambda P P^T)$$

$$= \max(\sum_{i=1}^{n} \lambda_i p_i^T p_i)$$

Taking a look into $p_i^T p_i$:

$p_i^T p_i = a_i^T u_i u_i^T a_i$ and since u_i is orthogonal as H is symmetric $u_i u_i^T = 1$

Then our problem is now:

$$\max(\sum_{i=1}^{n} \lambda_i a_i^T a_i)$$

It has been given that $A^T A = I_k$

Which means that $a_i^T a_i = 1$ when its column index is less than k,

$a_i^T a_i = 0$ when the index is greater than k.

Which makes

$$\max(\sum_{i=1}^{n} \lambda_i a_i^T a_i) = \max(\sum_{i=1}^{k} \lambda_i a_i^T a_i + \sum_{i=k+1}^{n} \lambda_i a_i^T a_i)$$

Since, $\sum_{i=k+1}^{n} \lambda_i a_i^T a_i = 0$ and $\sum_{i=1}^{k} \lambda_i a_i^T a_i = \sum_{i=1}^{k} \lambda_i$

$$= \sum_{i=1}^{k} \lambda_i$$

□

3. Question 5: Bonus Question

Let $K_1 \in \mathbb{R}^{n \times n}$ represent training kernel and $K_2 \in \mathbb{R}^{n \times m}$ represent training kernel

Suppose we have $A \in \mathbb{R}^{n \times k}$ represent training data and $B \in \mathbb{R}^{m \times k}$ represent training data.

$K_1 = AA^T$

We can decompose K_1 by Singular Value Decomposition as $U\Sigma U^T$.

In this case we can see that

$$U\Sigma U^T = U\Sigma^{1/2} \times (U\Sigma^{1/2})^T$$

From this we can easily obtain $A = U\Sigma^{1/2}$

Here we can get compute training data using U and Σ,

which we can easily get by doing Singular Value Decomposition of A.

We are going to use A along with the labels to training using the primal solver.

Next with the testing data part,

$K_2 = AB^T$

From the last part we know that $A = U\Sigma^{1/2}$

Which means $K_2 = U\Sigma^{1/2}B^T$

Since we know that U is unitary matrix

$U^T K_2 = \Sigma^{1/2}B^T$

$B = ((\Sigma^{1/2})^{-1}U^T K_2)^T$

This also we can get compute testing data using U and Σ,

which we can easily get by doing Singular Value Decomposition of A.

Using this we can later get the prediction.

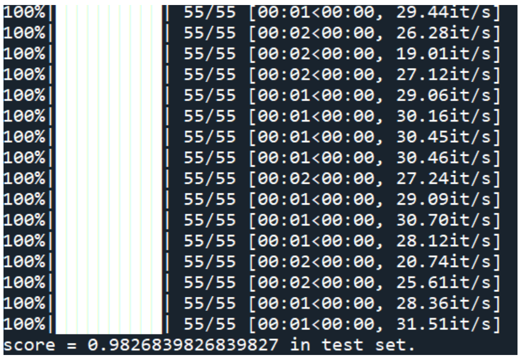
Section 2: Coding Part

3) Code implemented in the zip file.

Results:

a)

Kernel Logistic Regression Model:



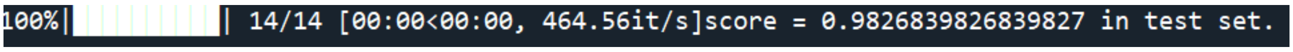
```
100%|          | 55/55 [00:01<00:00, 29.44it/s]
100%|          | 55/55 [00:02<00:00, 26.28it/s]
100%|          | 55/55 [00:02<00:00, 19.01it/s]
100%|          | 55/55 [00:02<00:00, 27.12it/s]
100%|          | 55/55 [00:01<00:00, 29.06it/s]
100%|          | 55/55 [00:01<00:00, 30.16it/s]
100%|          | 55/55 [00:01<00:00, 30.45it/s]
100%|          | 55/55 [00:01<00:00, 30.46it/s]
100%|          | 55/55 [00:02<00:00, 27.24it/s]
100%|          | 55/55 [00:01<00:00, 29.09it/s]
100%|          | 55/55 [00:01<00:00, 30.70it/s]
100%|          | 55/55 [00:01<00:00, 28.12it/s]
100%|          | 55/55 [00:02<00:00, 20.74it/s]
100%|          | 55/55 [00:02<00:00, 25.61it/s]
100%|          | 55/55 [00:01<00:00, 28.36it/s]
100%|          | 55/55 [00:01<00:00, 31.51it/s]
score = 0.9826839826839827 in test set.
```

The best parameters for the model were:

learning_rate = 0.001, max_epoch = 50, batch_size = 32, sigma = 5

b)

Radial Basis Function:



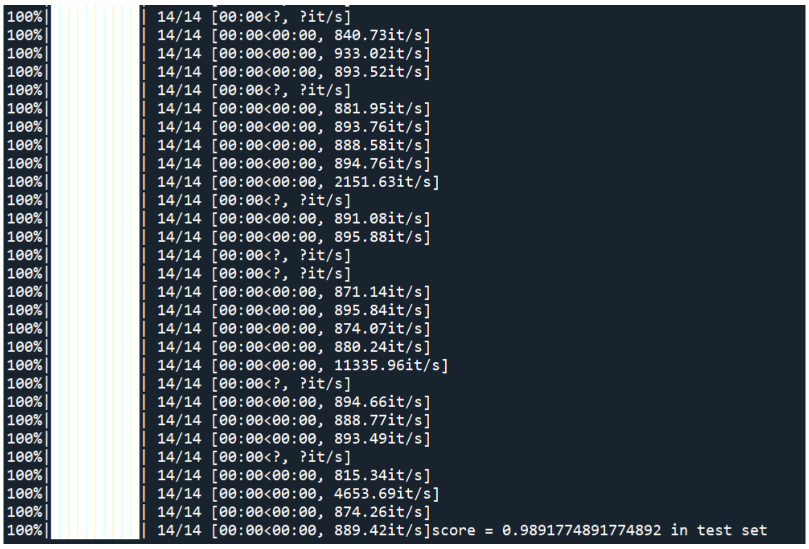
```
100%|          | 14/14 [00:00<00:00, 464.56it/s]score = 0.9826839826839827 in test set.
```

The best parameters for the model were:

hidden_dim = 16, learning_rate = 0.01, max_epoch = 50, batch_size = 128, sigma = 5

c)

Feed Forward Neural Network:



```
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 840.73it/s]
100%|          | 14/14 [00:00<00:00, 933.02it/s]
100%|          | 14/14 [00:00<00:00, 893.52it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 881.95it/s]
100%|          | 14/14 [00:00<00:00, 893.76it/s]
100%|          | 14/14 [00:00<00:00, 888.58it/s]
100%|          | 14/14 [00:00<00:00, 894.76it/s]
100%|          | 14/14 [00:00<00:00, 2151.63it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 891.08it/s]
100%|          | 14/14 [00:00<00:00, 895.88it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 871.14it/s]
100%|          | 14/14 [00:00<00:00, 895.84it/s]
100%|          | 14/14 [00:00<00:00, 874.07it/s]
100%|          | 14/14 [00:00<00:00, 880.24it/s]
100%|          | 14/14 [00:00<00:00, 11335.96it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 894.66it/s]
100%|          | 14/14 [00:00<00:00, 888.77it/s]
100%|          | 14/14 [00:00<00:00, 893.49it/s]
100%|          | 14/14 [00:00<?, ?it/s]
100%|          | 14/14 [00:00<00:00, 815.34it/s]
100%|          | 14/14 [00:00<00:00, 4653.69it/s]
100%|          | 14/14 [00:00<00:00, 874.26it/s]
100%|          | 14/14 [00:00<00:00, 889.42it/s]score = 0.9891774891774892 in test set
```

The best parameters for the model were:

hidden_dim = 32, learning_rate = 0.01, max_epoch = 200, batch_size = 128

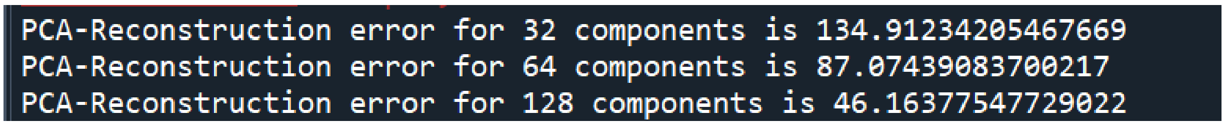
Analysis: I was able to see there has not been significant difference between the model. Also I expected more epoch I used, I will have a meaningful benefit in the Accuracy, however I could not see significant differences between parameters.

4)

<u>Results:</u>

b)

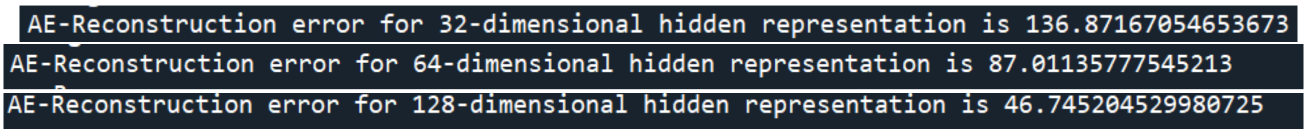
PCA:



```
PCA-Reconstruction error for 32 components is 134.91234205467669
PCA-Reconstruction error for 64 components is 87.07439083700217
PCA-Reconstruction error for 128 components is 46.16377547729022
```

d)

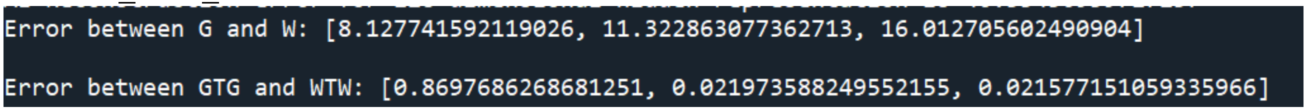
AutoEncoder(Weights shared):



```
AE-Reconstruction error for 32-dimensional hidden representation is 136.87167054653673
AE-Reconstruction error for 64-dimensional hidden representation is 87.01135777545213
AE-Reconstruction error for 128-dimensional hidden representation is 46.745204529980725
```

f)

Frobeniu_norm_error:

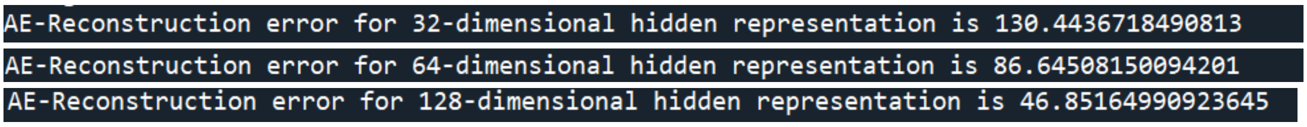


```
Error between G and W: [8.127741592119026, 11.322863077362713, 16.012705602490904]

Error between GTG and WTW: [0.8697686268681251, 0.021973588249552155, 0.021577151059335966]
```

g)

AutoEncoder(Weights Not Shared):

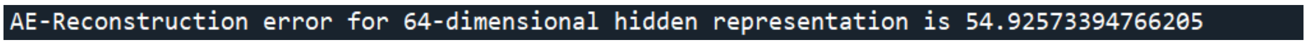


```
AE-Reconstruction error for 32-dimensional hidden representation is 130.4436718490813
AE-Reconstruction error for 64-dimensional hidden representation is 86.64508150094201
AE-Reconstruction error for 128-dimensional hidden representation is 46.85164990923645
```

h)

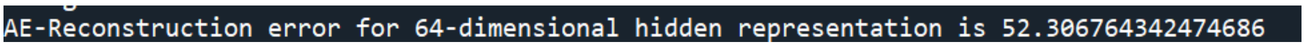
AutoEncoder(MoreLayers 64-dimensional):

64 Batches, 300 Epoch:



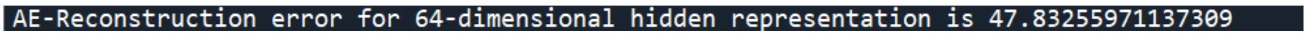
```
AE-Reconstruction error for 64-dimensional hidden representation is 54.92573394766205
```

64 Batches, 500 Epoch:



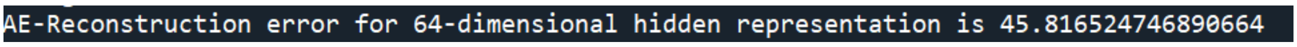
```
AE-Reconstruction error for 64-dimensional hidden representation is 52.306764342474686
```

64 Batches, 700 Epoch:



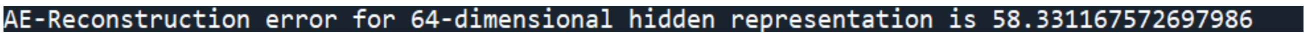
```
AE-Reconstruction error for 64-dimensional hidden representation is 47.83255971137309
```

64 Batches, 1000 Epoch:



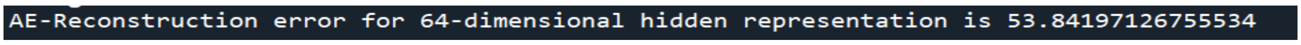
```
AE-Reconstruction error for 64-dimensional hidden representation is 45.816524746890664
```

128 Batches, 300 Epoch:



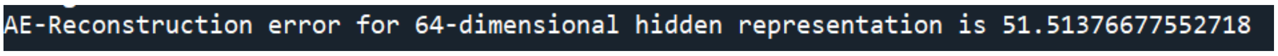
```
AE-Reconstruction error for 64-dimensional hidden representation is 58.331167572697986
```

128 Batches, 500 Epoch:



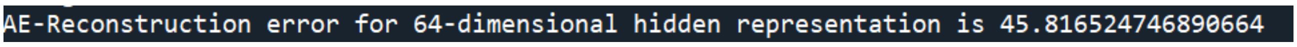
```
AE-Reconstruction error for 64-dimensional hidden representation is 53.84197126755534
```

128 Batches, 700 Epoch:



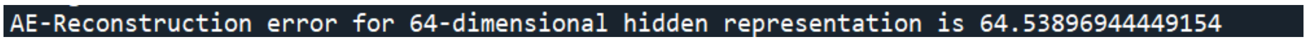
```
AE-Reconstruction error for 64-dimensional hidden representation is 51.51376677552718
```

128 Batches, 1000 Epoch:



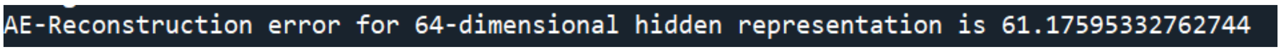
```
AE-Reconstruction error for 64-dimensional hidden representation is 45.816524746890664
```

256 Batches, 300 Epoch:



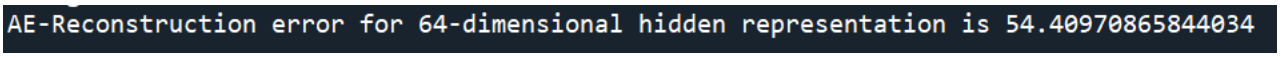
```
AE-Reconstruction error for 64-dimensional hidden representation is 64.53896944449154
```

256 Batches, 500 Epoch:



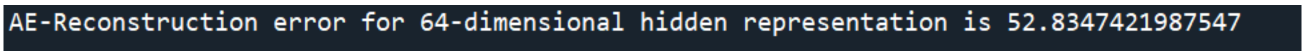
```
AE-Reconstruction error for 64-dimensional hidden representation is 61.17595332762744
```

256 Batches, 700 Epoch:



```
AE-Reconstruction error for 64-dimensional hidden representation is 54.40970865844034
```

256 Batches, 1000 Epoch:



```
AE-Reconstruction error for 64-dimensional hidden representation is 52.8347421987547
```

Analysis:

b) We can see that in general error is bigger when the dimension is lower.

e,g,h) Batch = 128, Epochs = 300

# of Dimensions / Types of Algorithms	32	64	128
PCA	134.912	87.074	46.164
AutoEncoder(Weights Shared)	136.872	87.011	46.745
AutoEncoder(Weights Not Shared)	130.444	86.645	46.852
AutoEncoder(More Layer)	NA	58.331	NA

From the table above there is a bit of PCA is slightly has less error compared to two AutoEncoder.

However, once we add more layer into AutoEncoder, we can get significantly less error compared to the other ones.

h)

Also, from the results that I got, I was able to see that less batch and more epoch gives less error. However, having less batch and more epoch is not necessarily good in all cases, because we need to also care about the time efficiency and storage usage.