FAI-HW4

waganawamegumi

May 2023

Hand-Written Part

Problem 1.

$$
\begin{aligned}
\phi'(s) =& \theta(s) + s\theta'(s) \\
\theta'(s) =& -1 \times (1 + exp(-s))^{-2} \times -1 exp(-s) \\
=& \frac{exp(-s)}{(1 + exp(-s))^2} \\
\phi'(s) =& \frac{1}{(1 + exp(-s))} + \frac{sexp(-s)}{(1 + exp(-s))^2} \\
=& \frac{1 + e^{-s} + se^{-s}}{(1 + e^{-s})^2} \\
Ans :& \frac{1 + e^{-s} + se^{-s}}{(1 + e^{-s})^2}
\end{aligned}
$$

Problem 2.

(a.)

$$
v_0 = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}
$$

$$
v_1 = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{3} \end{bmatrix}
$$

$$
v_2 = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{2} \end{bmatrix}
$$

$$v_3 = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{4} \\ \frac{1}{3} \end{bmatrix}$$

$$v_4 = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{6} \\ \frac{5}{12} \end{bmatrix}$$

$$v_5 = \begin{bmatrix} \frac{3}{8} \\ \frac{5}{24} \\ \frac{5}{12} \end{bmatrix}$$

(b.)

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0.5 \\ 0 & 0 & 0.5 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\begin{Bmatrix} 2v_1 - 2v_2 - v_3 = 0 \\ 2v_2 - v_3 = 0 \\ v_3 - v_1 = 0 \end{Bmatrix}$$

$$v_1 = v_3, \quad v_2 = \frac{v_3}{2} \Rightarrow v_1 : v_2 : v_3 = 2 : 1 : 2$$

$$normalize : (v1, v2, v3) = (\frac{2}{5}, \frac{1}{5}, \frac{2}{5})$$

$$v^* = \begin{bmatrix} \frac{2}{5} \\ \frac{1}{5} \\ \frac{2}{5} \end{bmatrix}$$

Problem 3.

L=1:

$$d^{(1)} = 100$$

$$\text{total number of weights} = 10 \times d^{(1)} = 1000$$

L=2:

$$d^{(1)} + d^{(2)} = 100$$

$$0 < d^{(1)} < 100$$

$$d^{(2)} = 100 - d^{(1)}$$

$$\text{total number of weights}$$

$$= 10d^{(1)} + d^{(1)}d^{(2)}$$

$$
\begin{aligned}
&= 10d^{(1)} + d^{(1)}(100 - d^{(1)}) \\
&= -(d^{(1)} - 55)^2 + 3025 \\
&\text{because } 0 < d^{(1)} < 100 \\
&0 \leq (d^{(1)} - 55)^2 \leq 54^2 \\
&109 \leq -(d^{(1)} - 55)^2 + 3025 \leq 3025
\end{aligned}
$$

L=3

$$
\begin{aligned}
&d^{(1)} + d^{(2)} + d^{(3)} = 100 \\
&0 < d^{(1)} \leq 98 \quad (d^{(n)} > 0) \\
&0 < d^{(2)} < 100 - d^{(1)} \\
&d^{(3)} = 100 - d^{(1)} - d^{(2)} \\
&\text{total number of weights} \\
&= 10d^{(1)} + d^{(1)}d^{(2)} + d^{(2)}d^{(3)} \\
&= 10d^{(1)} + d^{(1)}d^{(2)} + d^{(2)}(100 - d^{(1)} - d^{(2)}) \\
&= 10d^{(1)} - (d^{(2)} - 50)^2 + 2500
\end{aligned}
$$

consider following 2 cases:
(1.) $50 \leq d^{(1)} \leq 98$, $2 \leq 100 - d^{(1)} \leq 50$:

$$
\begin{aligned}
&(100 - d^{(1)} - 1 - 50)^2 \leq (d^{(2)} - 50)^2 \leq 49^2 \\
&max: 10d^{(1)} - ((100 - d^{(1)} - 1) - 50)^2 + 2500 \\
&= -(d^{(1)} - 54)^2 + 54^2 + 99 \leq 3015 \\
&min: 10d^{(1)} - 49^2 + 2500 \geq 599
\end{aligned}
$$

(2.) $1 \leq d^{(1)} \leq 49$, $51 \leq 100 - d^{(1)} \leq 99$:

$$
\begin{aligned}
&0 \leq (d^{(2)} - 50)^2 \leq 49^2 \\
&max: 10d^{(1)} - 0 + 2500 \leq 2990 \\
&min: 10d^{(1)} - 49^2 + 2500 \geq 109
\end{aligned}
$$

min of L=3: (1.)599, (2.)109
max of L=3: (1.)3015, (2.)2990
(A) max: 3025 weights in total,.
architecture: L=2, $d^{(1)} = 55$, $d^{(2)} = 45$.
(B) min: 109 weights in total.
architecture: L=2, $d^{(1)} = 1$, $d^{(2)} = 99$

Report

(a.)pca

mean:

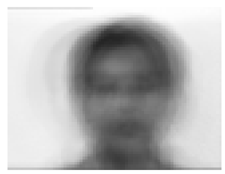


Figure 1: mean vector

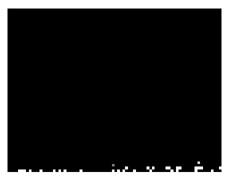


Figure 2: component 1

Figure 3: component 2

Figure 4: component 3

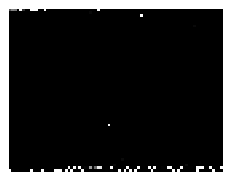


Figure 5: component 4

(b.)training curve

Autoencoder:

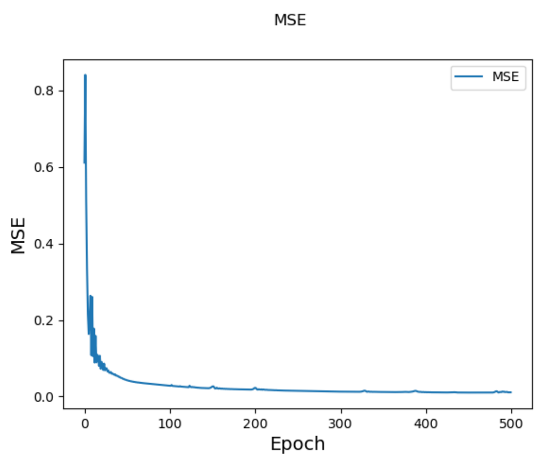


Figure 6: Autoencoder

DenoisingAutoencoder:

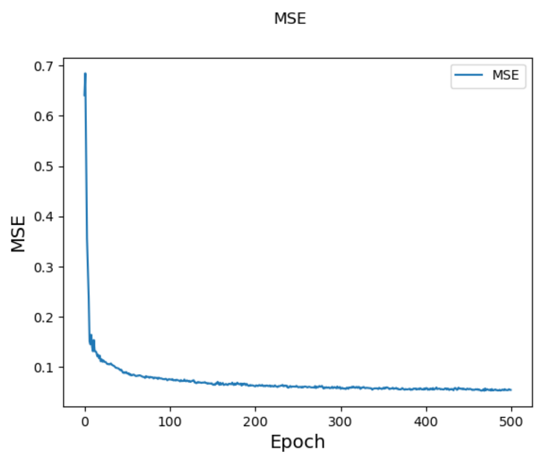
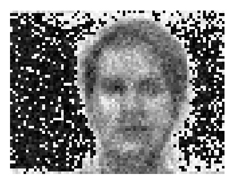


Figure 7: Deno

(c.)reconstruction

(a) pca

(b) Autoencoder



(c) Deno

Figure 8: reconstruction

Reconstruction error:
Reconstruction Loss with PCA: 3.981720845825862e-31
Reconstruction Loss with Autoencoder: 0.018090815714432912
Reconstruction Loss with DenoisingAutoencoder: 0.1829118065506724

(d.) models

original:

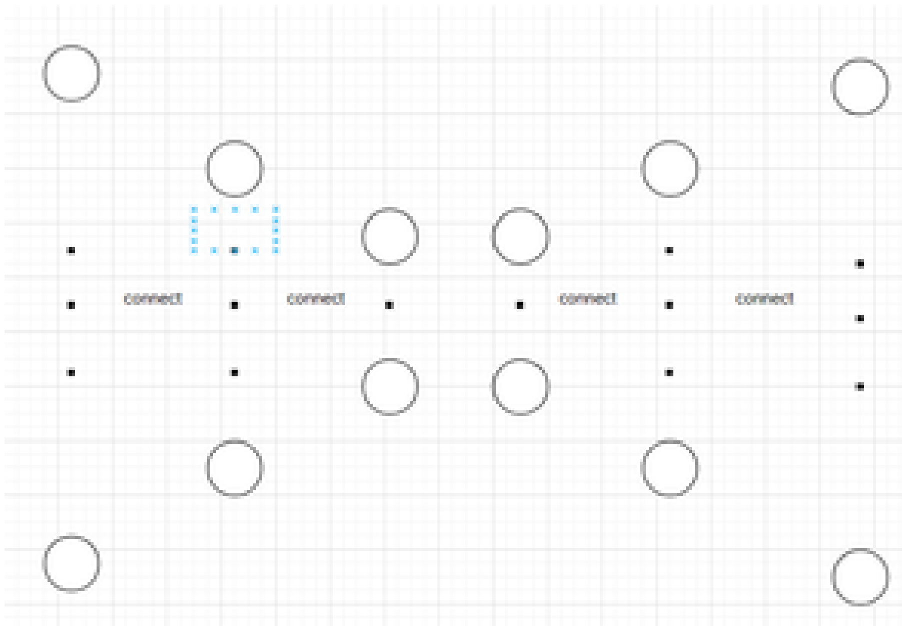


Figure 9: original

```
Acc from PCA: 0.9666666666666667
Acc from Autoencoder: 0.9
Acc from DenoisingAutoencoder: 0.8666666666666667
Reconstruction Loss with PCA: 3.981720845825862e-31
Reconstruction Loss with Autoencoder: 0.018090815714432912
Reconstruction Loss with DenoisingAutoencoder: 0.1829118065506724
```

Figure 10: performance

shallower:

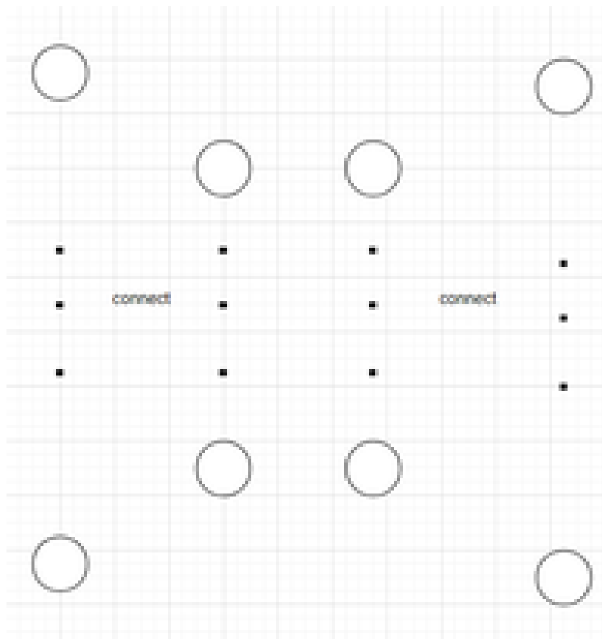


Figure 11: shallower

```
Acc from PCA: 0.9666666666666667
Acc from Autoencoder: 0.9333333333333333
Acc from DenoisingAutoencoder: 0.8666666666666667
Reconstruction Loss with PCA: 3.981720845825862e-31
Reconstruction Loss with Autoencoder: 0.012747174110231902
Reconstruction Loss with DenoisingAutoencoder: 0.02230285267564073
```

Figure 12: performance

deeper:

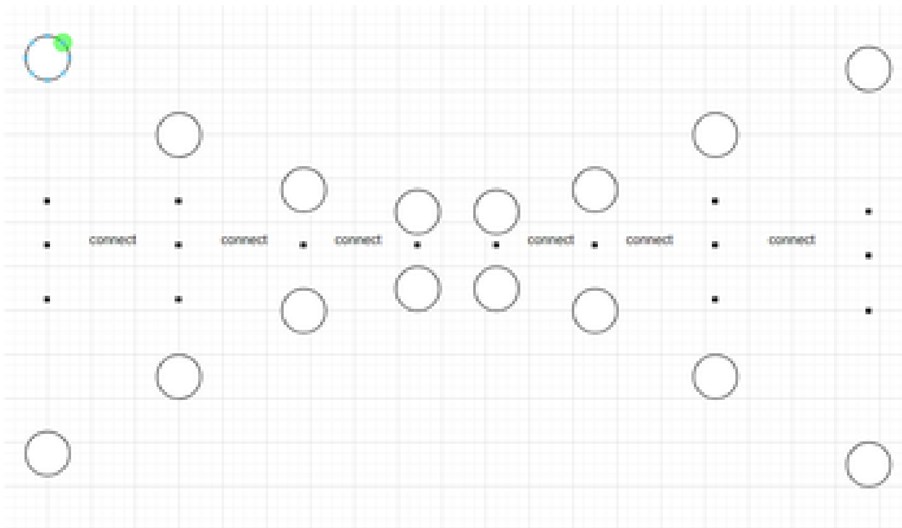
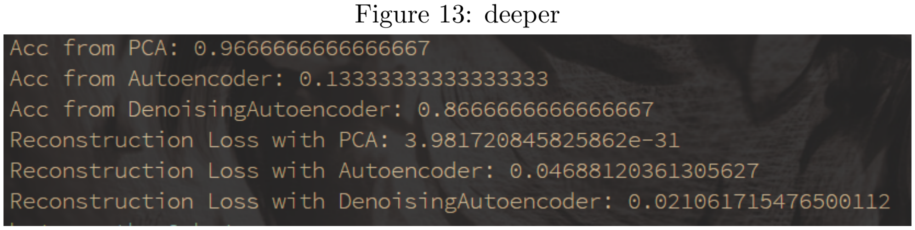


Figure 13: deeper



```
Acc from PCA: 0.9666666666666667
Acc from Autoencoder: 0.13333333333333333
Acc from DenoisingAutoencoder: 0.8666666666666667
Reconstruction Loss with PCA: 3.981720845825862e-31
Reconstruction Loss with Autoencoder: 0.04688120361305627
Reconstruction Loss with DenoisingAutoencoder: 0.021061715476500112
```

Figure 14: performance

discussion:
Compare to the original one:
1. shallower model shows lower loss rate on Autoencoder, but shows higher loss rate on Denoising Autoencoder. I think the shallower model cause less information loss, making the loss of Autoencoder lower, but it is also weaker for filtering the noise, making the loss of DenoisingAutoencoder higher.
2. deeper model shows extremely higher loss on Autoencoder, and it performs normal with DenoisingAutoencoder. I think the deeper model caused too much information loss, which makes the autoencoder can't compress the features well. However, the original database might be quite similar to a Gaussian distribution, so the model can still capture the key features which the distribution is needed.

Summary:
Deeper model somtimes perform worse. It's important to select a proper input and output variables. Adding noise might improve the generalization by a lot.

(e.) optimizers

Denoising Autoencoder

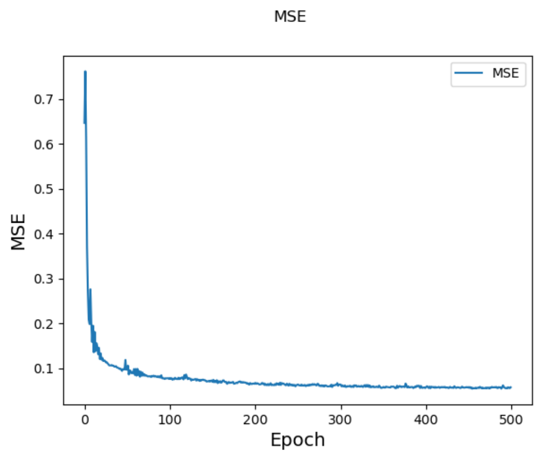


Figure 15: ADAM

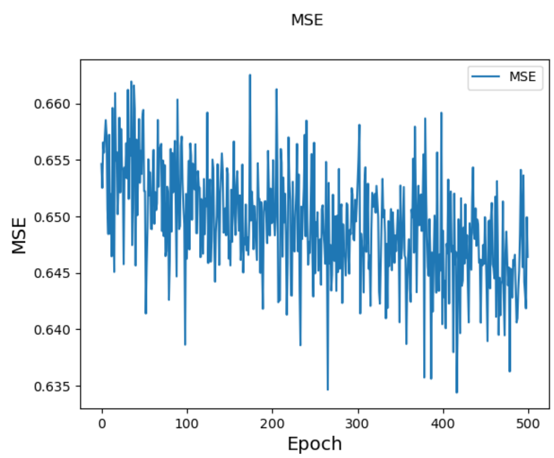


Figure 16: SGD

Discussion:
1. Comparing to SGD, the training curve of ADAM optimizer is more smooth. I think the reason is that SGD is influenced a lot by the fixed learning rate. The fixed length of step might be too long, making it fail to find the converging point again and again. The ADAM optimizer adjust the learning rate dynamically in case the converging point is missed.
2. In this task, ADAM optimizer converges much more quickly than SGD optimizer does. I think one of the reason is that the target function behind

this task is relatively steep, so the ADAM optimizer can quickly find out the converge point. If the target curve(or plane) is relatively gentle, SGD might performs better than ADAM.