# Assignment 2

# BS6207

## Q1.1

The standard gradient descend is an algorithm used to find the local minimum of a differentiable function. The basic idea of this method is to repetitive take steps around the current point and finally lead to the local minimum. In this diagram, we can see the point x has an abnormal trend during the learning process.

When $X_0 = 0$, $y_0 = 1$, the learning rate is 0.3. From green part, we can the slope is 1 when x in range 1 to 1+h. The slope is -1 when x is larger than 1+h. So the formular will be $\mathbf{X_n = X_{n-1} - ag}$

| | | | |
|---|---|---|---|
| x0, y0 | 0,1 | x1, y1 | 0.3, 0.7 |
| x2, y2 | 0.6, 0.4 | x3, y3 | 0.9, 0.1 |
| x4, y4 | 1.2, 0.2 | x5, y5 | 0.9, 0.1 |

*Table 1 Standard Gradient Descend*

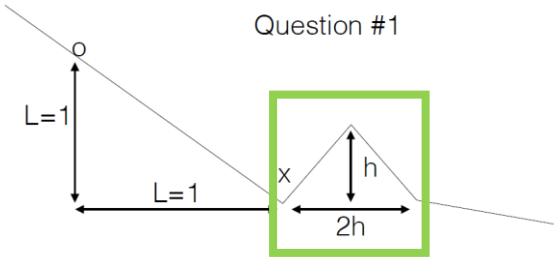


*Figure 1 plot of Question1*

## Q1.2

I implement the adam function with the parameter given in the question. In the for loop, I first set t to be next t to calculate next m_hat and v_hat. Then I check theta, if theta is smaller than 1, g will be -1, and vice versa. Finally, store all the theta -1 in to list because the slope of that range is -1. Totally I calculate for 20 times and find out the maximum value. The max x also called theta is 1.41018 so the y also called h will be 1.41018 – 1 =0.41018

```
ans = []

for i in range(20):
    t =t + 1
    if theta<1:
        g = -1
    else:
        g = 1
    m = beta1 * m + (1 - beta1) * g
    v = beta2 * v + (1 - beta2) * (g * g )
    m_hat = m / (1 - beta1 ** t)
    v_hat = v / (1 - beta2 ** t)
    theta -= lr * m_hat / (v_hat ** 0.5 + epislon)


    ans.append(theta-1)
print('Max h = ',max(ans))
```

```
Max h =  0.4101842951299657
```

*Figure 2. Adam Algorithm*

## Q2.a

In this question, I write three auto encoders which latent space dimensions are 2,16 and 256. During the model parameter selection, I use the adam as the optimizer and mean_absolute_error as the reconstruction loss function. I plot each three auto encoders' 2D plot with each color per digit and also the digit itself. Also in the dimension of 16 and 256, I used k-means to do clustering and since 16 and 256 are high dimensions. I used T-SNE to do dimension reduction and plot the 2D image.

**Dimension of 2 :**

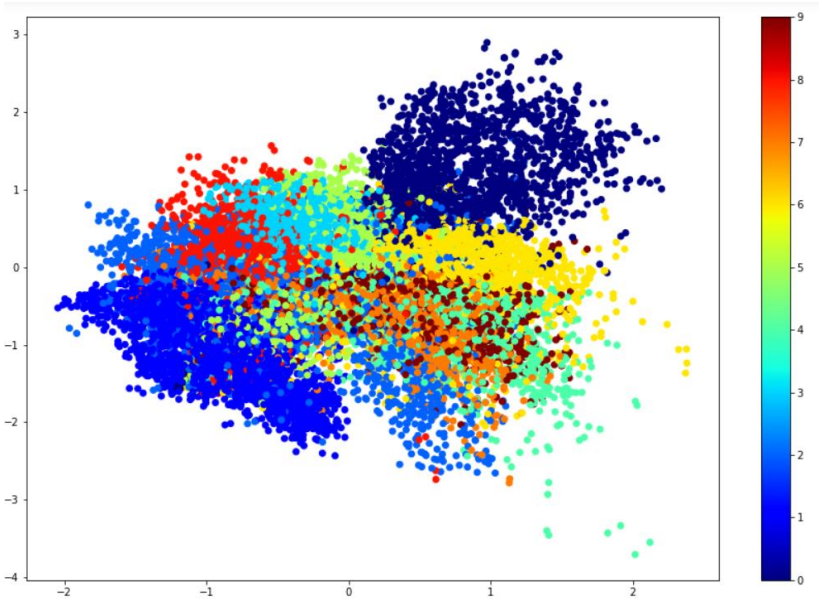

*Figure 3.*

*Figure 4.*

**Dimension of 16 :**

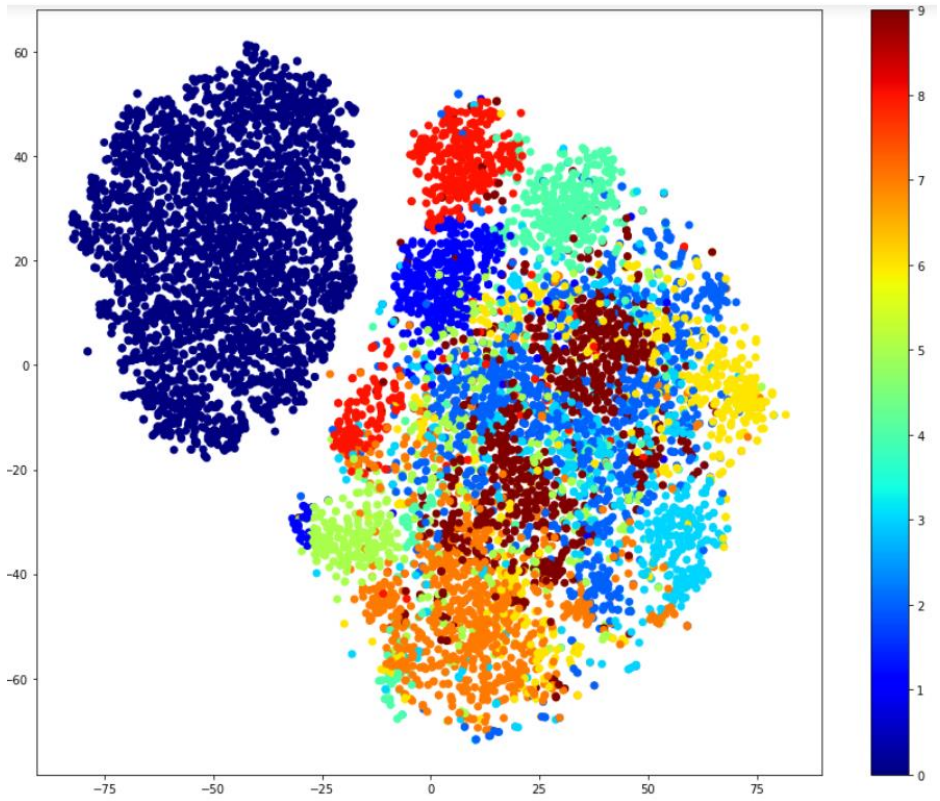


*Figure 5.*

*Figure 6.*

**Dimension of 256 :**

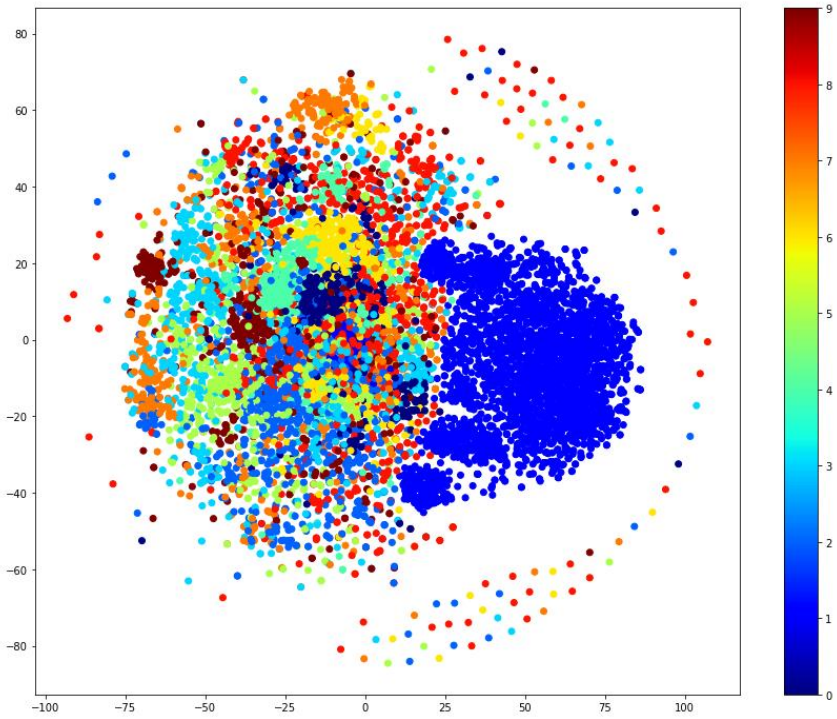


*Figure 7.*

*Figure 8.*

We can see the digit nine looks different with different latent space dimensions after reconstructed images. And higher dimension reconstructs better than a low dimension in my case.

## Q2.b

For another Neural Network ‘dis_net’, I first blur the image matrix by using Gaussian noise and clip the image between 0 and 1. And here is how it looks like.

*Figure 8.*

After feeding the noisy digits to the network and training the autoencoder with L1-norm reconstruction loss and discriminator loss. I compare the result of reconstruction with a and find that it looks similar to a's result.

*Figure 9.*