Assignment 2

Yaoding Chen

Code Available through Google Colab:

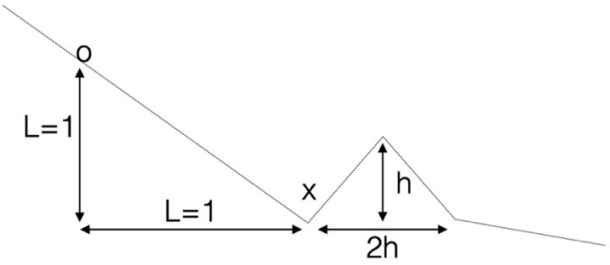
Question 1 Optimiser:
https://colab.research.google.com/drive/1fOA-eBGbcVTljZXZdjPnWKFQoQliyuQA?usp=sharing

Question 2 – Part 1 AutoEncoder:
https://colab.research.google.com/drive/1fOA-eBGbcVTljZXZdjPnWKFQoQliyuQA?usp=sharing

Question 2 – Part 2 Adversarial AutoEncoder:
https://colab.research.google.com/drive/1CfJkcDZX27Zj7qzErd4gkhkQbPjf7qpt?usp=sharing

---

Question 1



The diagram above shows a plot of a 1D function and gradient descend is applied to minimise the function at the point 'o'. There is a bump a distance L away with bump dimensions given as $h \times 2h$. Let $L=1$, $a=0.3$ and $h>a$, where a is the learning rate.

(1) what will happen if you apply standard gradient descend?

After a few steps, the cost function will converge to the local minimum x, instead of the global minimum, because the learning rate a is lower than h, which means each step is smaller than what is enough to detect another local minimum. Therefore, the function is trapped there.

(2) If you apply Adam optimisation with parameters given in the next slide, what is the max height 'h' of the bump in which the Adam optimiser will escape the local min at 'x'? Use $\mathcal{E}=0$ in instead of $\mathcal{E}=$1e-8 in your calculations.

The maximum height is 0.41. Please see the attached code.[1]

[1] https://colab.research.google.com/drive/1h6PmBW0c9vbGX0XCqCuOrGDEK7Ee09PG?usp=sharing

Question 2

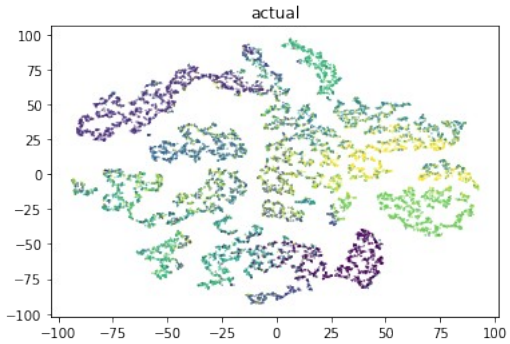
(a) Design an auto encoder to take in MNIST images with latent space dimension of $2, 16, 256$. Train auto encoder with L1-norm reconstruction loss.

Please see the attached code.[2]

Do a 2D plot of the latent space for different digits for latent space of 2. K-means clustering for latent space of dimensions $16, 256$. Use one color for each digit.

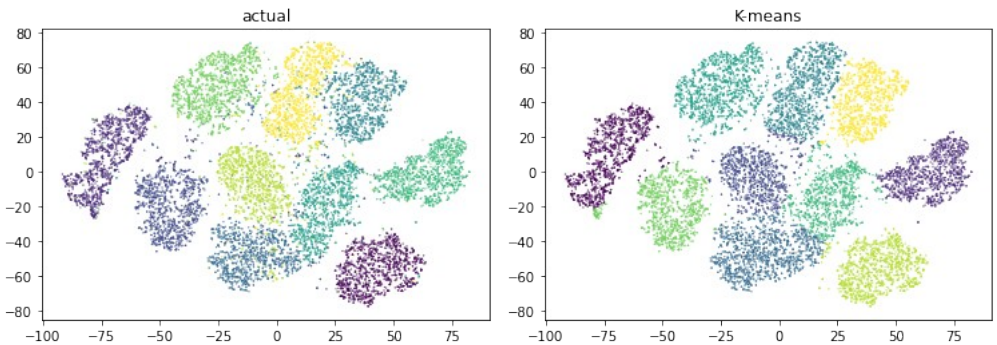
Dim =2: the result shows no clear clustering, suggesting little information learnt.

```
Autoencoder(
  (encoder): Encoder(
    (linear1): Linear(in_features=784, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=2, bias=True)
  )
  (decoder): Decoder(
    (linear1): Linear(in_features=2, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=784, bias=True)
  )
)
```



Dim = 16: the result shows clear clustering. k-mean clustering results are unstable due to random centroids, but I managed to choose one that is closest to the actual results, which is not easy to get unless trying many times.

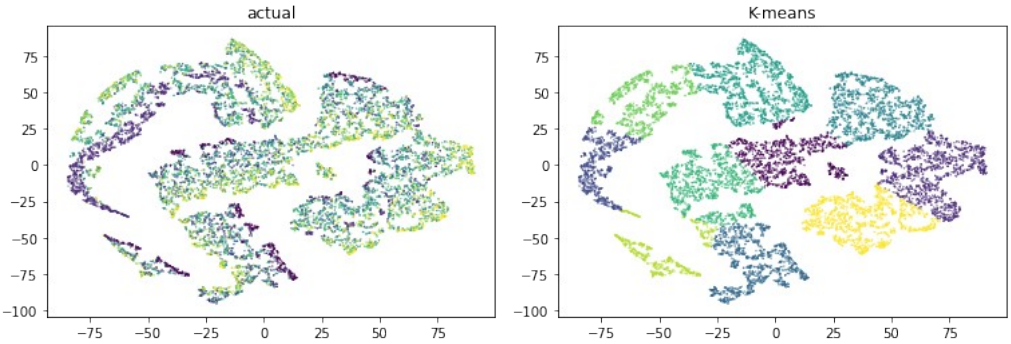
```
Autoencoder(
  (encoder): Encoder(
    (linear1): Linear(in_features=784, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=16, bias=True)
  )
  (decoder): Decoder(
    (linear1): Linear(in_features=16, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=784, bias=True)
  )
)
```



[2] https://colab.research.google.com/drive/1fOA-eBGbcVTljZXZdjPnWKFQoQliyuQA?usp=sharing

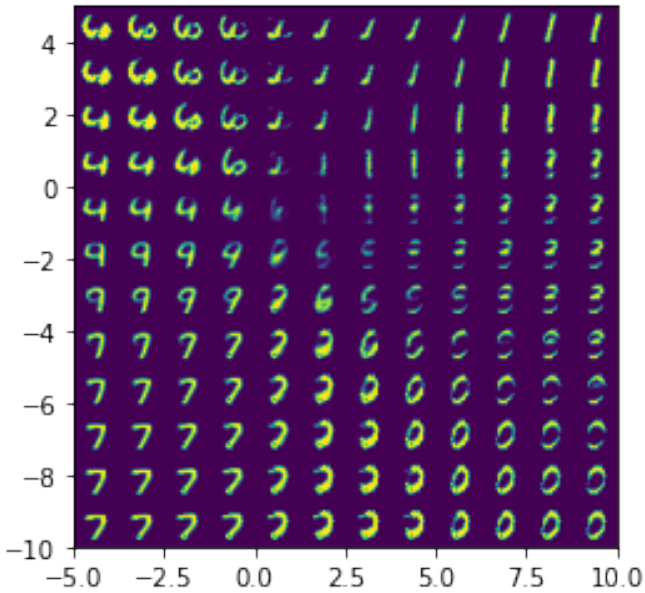
Dim = 256: it seems that little information is learnt and correctly summarised in the latent space, and the k-means results are poor, although it shows some kind of shapes we expect as a result of auto-encoder learning.

```
Autoencoder(
  (encoder): Encoder(
    (linear1): Linear(in_features=784, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=256, bias=True)
  )
  (decoder): Decoder(
    (linear1): Linear(in_features=256, out_features=512, bias=True)
    (linear2): Linear(in_features=512, out_features=784, bias=True)
  )
)
```



Report all results. What do you notice about the reconstructed images?

The images reconstructed in the latent layer seem to be a blurred, distorted version of the original images, which may indicate certain information loss.



(b) Design another neural network “dis_net” to discriminate between blur images and clear images. Blur images can be generated by taking the original MNIST data and do some gaussian blur.

Train autoencoder with L1-norm reconstruction loss + discriminator loss. Make reconstructed images as clear as possible, that is, the auto encoder will need to be trained so that “dis_net” score it as a clear image.
Compare results between (a) and (b)

Here we need to design a generative adversarial autoencoder.[3] And we implement this according to https://blog.paperspace.com/adversarial-autoencoders-with-pytorch/ The latent space dimension is set to 16. And the image is denoised after processing by the net.

I haven't finished up my own code here, [4] and the current version is just a walkthrough of the tutorial article. I will try to integrate the code with previously defined auto-encoder during the holiday, whether it is to be marked or not.

[3] https://www.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/
[4] https://colab.research.google.com/drive/1CfJkcDZX27Zj7qzErd4gkhkQbPjf7qpt?usp=sharing