# Github resource located at “https://github.com/wany0011/BS6207/tree/main/assignment\_2”

# Q 1:

**Part – 1**: Based on the values of target function, its gradient values are calculated & shown below in red line, where:

1. To the left of ‘x’, it’s all “-1”
2. In between (x, H) it’s “1”
3. To the right of H, it’s “-1”

The discontinued gradient points at x and h, respective, are not defined.

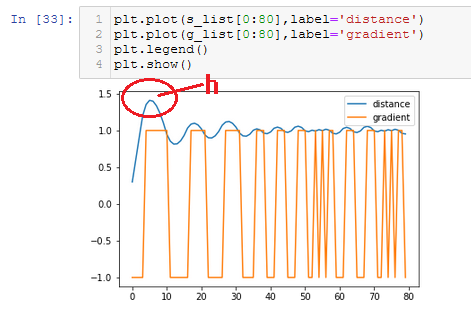
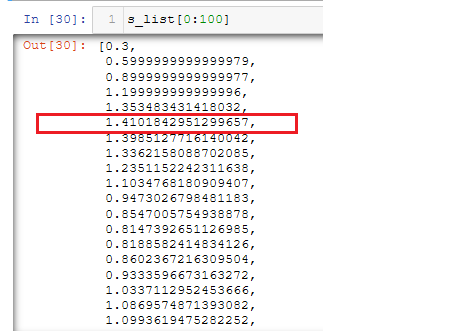
# 

# Take x as the reference point, Given the starting at L left to x with standard gradient descent, Let L = 1, a = 0.3 and h > a where a is the learning rate, the movement is illustrated here:

1. From left to right, it departs at ‘o’, along the slope towards x”, the total distance is 1
2. At the step size of 0.3, by the end of step 4: it passed-by “x”, along the slope of “x-h”, at point:
3. Give , Starting from step 5, it swung back to the left of ‘x’, along “x-o”, at point
4. At end of step 6, it returns to the point of step 3 at 0.2.
5. Now it’s an infinite loop, i.e., swinging back and forth between 3) and 4), and no way out.

**Part – 2**: Using ADAM, which is defined collectively by Root-Mean-Square Propagation and Momentum of mini-batch gradient decent), However at learning rate of 0.001, it’s no longer able to be sketched by hand. Intuitively:

1. The step-size of movement becomes variable, it keeps changing at each step, even with constant gradients.
2. The instant gradient depends on all its previous gradients at diminishing rate. (Exponentially weighted moving average)

Therefore, the intuition is the all the past gradients will add-up together and prop / push the current movement passing by the sloop of “x-h”, and that can be precisely calculated by a program – “q1.ipynb”

It shows the distance vs. gradient, and turning point located at: h = 1.410184

what happen next when the peak distance-point of h is reached? – it’s oscillating at h, just like the behaviour in the previous question, but at diminishing attenuation, and eventually converges.

Only if there is another turning point sitting above x but below h – therefore it answers the question of maximum height, that is indicate by h.

A picture containing diagram

Description automatically generated

# Q 2: ~~(take note – It looks like I misread the instruction, thus I misunderstood the definition of laten-space of 2, 16, 256, I though they refer to individual models, i.e., the very last layer of encoder – when I found this, it’s already too late for me to change back, I will redress the problem during my presentation for this assignment.)~~

* **Updates on 18-Dec (sat) Resubmission – please skip to page -7**

# Data Partitioning:

Training Dataset = 48000

Validation Dataset =12000

Test Dataset = 10000

# Try the simplest possible autoencoder (source code at “./q2\_mlp.ipynb”)

It starts simple, with a single fully-connected nerual layer as encoder and as decoder:

Diagram

Description automatically generatedIt evaluates the reconstruction-loss with latent space dimension of 2:

Diagram, table

Description automatically generated

A picture containing text, device, gauge, meter

Description automatically generatedChart

Description automatically generated

Clearly, it’s the sign of under-fitting, since it’s too simple model, which justifies more layers required.

# Try sophisticated CNN mode (source code at “./q2.ipynb”)

### **Reconstruction only**

### Laten-space dimension set to 2:

* Total Classification-loss (test) = 0.07422441244125366
* Randomly regenerate samples from latent space.

A picture containing grater, kitchenware, microphone, grate

Description automatically generated

* Plot of the 2-D location of the original images mapping out to the latent space.

Chart, scatter chart

Description automatically generated

* T-SNE visualization of the latent space.

Chart, map, scatter chart

Description automatically generated

1. Applying K-Means to the encoded layer for test-data, with Confusion matrix generated, followed by accuracy of prediction.

Graphical user interface, text

Description automatically generated

### **With either random noises or gaussian-noises added to individual input image, accordingly the loss function is defined as sum of reconstruction loss and denoising loss.**

### With Laten-space dimension set to 2:

* Classification-loss (test) = 0.07392493635416031

# Text, letter Description automatically generated

From left-to-right

1. Original image -> corrupted image -> reconstructed image
2. Dot-scatter plot of the 2-D latent space
3. T-SNE of the 2-D latent space (in place of K-Mean)

Map

Description automatically generatedChart, scatter chart

Description automatically generatedText

Description automatically generated

### With Laten-space dimension set to 16

* Total Classification (test) = 0.03253877907991409

Chart, map, scatter chart

Description automatically generatedText

Description automatically generated

### With Laten-space dimension set to 256

* Visualize coding of images in the latent-space:

Text

Description automatically generated

* Total Classification -lost (test) = 0.02503991685807705

Map

Description automatically generatedText

Description automatically generated

**Re-submission on 18-Dec (Sat)**

Firstly, establish baseline of K-Mean with original images, Since K-Mean is un-supervised learning, a purity function is required to find the best possible match of individual digits. And the accuracy of K-Mean with the 10000 original test-images is **0.5726**

Next, clustering accuracy at respective encoder dimension (with no random noise added) is summarized as follows, please refer to appendix for the detailed confusion matrix.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Encoder Dimension** | **With Gaussian Noise** | **Model Accuracy (Loss of MAE for Test)** | **Clustering accuracy at Encoder layer** |
| **1** | **2** | **N** | **0.07385** | **0.6431** |
| **2** | **16** | **N** | **0.03016** | **0.6795** |
| **3** | **256** | **N** | **0.01688** | **0.6664** |

Lastly, the summary report (with no random noise added) is tabulated below. Again, please refer to appendix for the detailed confusion matrix.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Encoder Dimension** | **With Gaussian Noise** | **Model Accuracy (Loss of MAE for Test)** | **Clustering accuracy at Encoder layer** |
| **4** | **2** | **Y** | **0.** **07386** | **0.7674** |
| **5** | **16** | **Y** | **0.03393** | **0.6238** |
| **6** | **256** | **Y** | **0.02522** | **0.6360** |

Observation:

1. Measured by K-Meaning clustering at encoder-layer:

* Compared the k-mean clustering, the autoencoder approach performs better in accuracy that using the original images.
* When the encoder increases its dimensions, the accuracy drops.
* With random noise added at each iteration, the accuracy of autoencoder improves.

Explanation:

1. Although data representation in high dimension, but the support manifolds can be in low dimension, with no exception to image data having low dimensional support, i.e., the actual structure/pattern can be in very low dimension.
2. With noise added at each iteration of training progress, that basically stretches the actual data-distribution all over the high-dimensional space, since the decoder remains unchanged to such un-generalizable random noise, it helps generation, with many noisy points surrounding actual data point.

# [Appendix]

* Benchmark accuracy by K-Mean clustering:

A picture containing application

Description automatically generated

* Graphical user interface, application

  Description automatically generatedAccuracy with encoder under the predefined conditions (No noise):

A picture containing treemap chart

Description automatically generated

A picture containing treemap chart

Description automatically generated

* Accuracy with encoder under the predefined conditions (With noise)

**A picture containing Teams

Description automatically generatedQr code

Description automatically generated**

**A picture containing Teams

Description automatically generated**