人工智慧晶片設計與應用
AI-on-Chip for Machine Learning and Inference

LAB Exercises, Spring 2020

Outline

- Labs:
  - Lab1: ML Tool Introductions and Installations (GPU)
  - Lab2: Implement Lenet-5 Model in Tensorflow (GPU)
  - Lab3: Kneron Accelerator Platform (KAP) and SDK (AI Accelerator)
  - Lab4: AI model on Kneron KAP (AI Accelerator)
  - Lab5: OpenVINO and Intel Movidius (AI Accelerator)
  - Lab6: OpenCL Exercises on CASLAB GPU (GPU)

- Projects:
  - Implement 1D PE convolution accelerator
  - Propose an application implementation using Kneron KAP and Intel Movidius

Online Resources for Basics and Terminology

- Google Machine Learning Crash Course
  - https://developers.google.com/machine-learning/crash-course/ml-intro
- Machine Learning Glossary
  - https://developers.google.com/machine-learning/glossary

e.g.,

outliers

Values distant from most other values. In machine learning, any of the following are outliers:
•**Weights** with high absolute values.
•Predicted values relatively far away from the actual values.
•Input data whose values are more than roughly 3 standard deviations from the mean.
Outliers often cause problems in model training. **Clipping** is one way of managing outliers.

Lab1

ML Tool Introductions and Installations (GPU)

TA

- 王志瑋 (Jhih-Wei, Wang)
- Email: tommywang0.tw@gmail.com

Outline

- Background Knowledge
  - Regression
  - Classification
  - Gradient Descent
  - ANN
  - Training
- Lab
  - Environment
  - Keras
  - Example
  - Practice

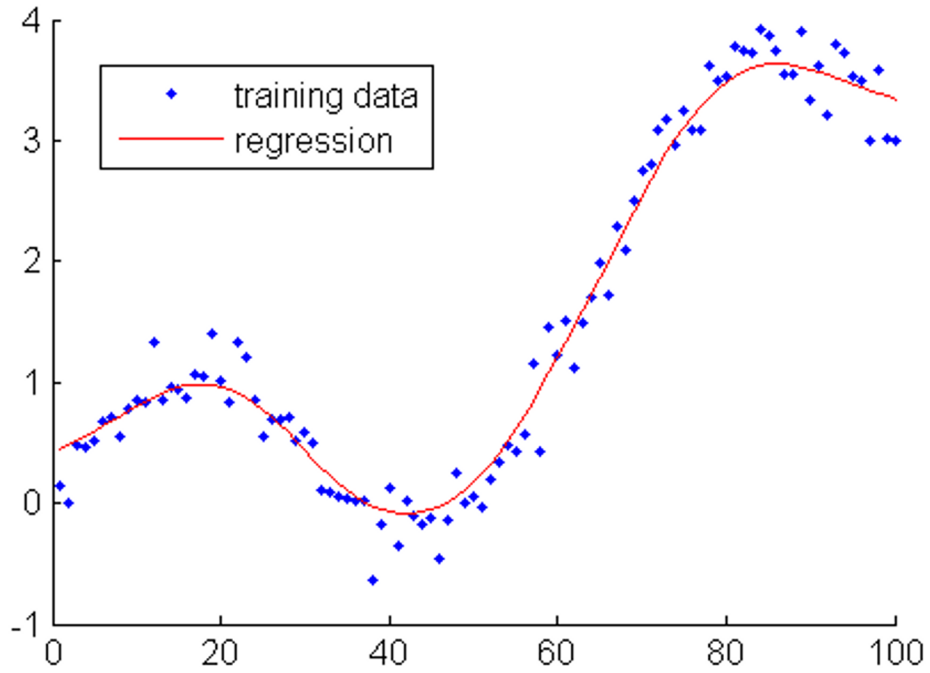
Background Knowledge

Regression

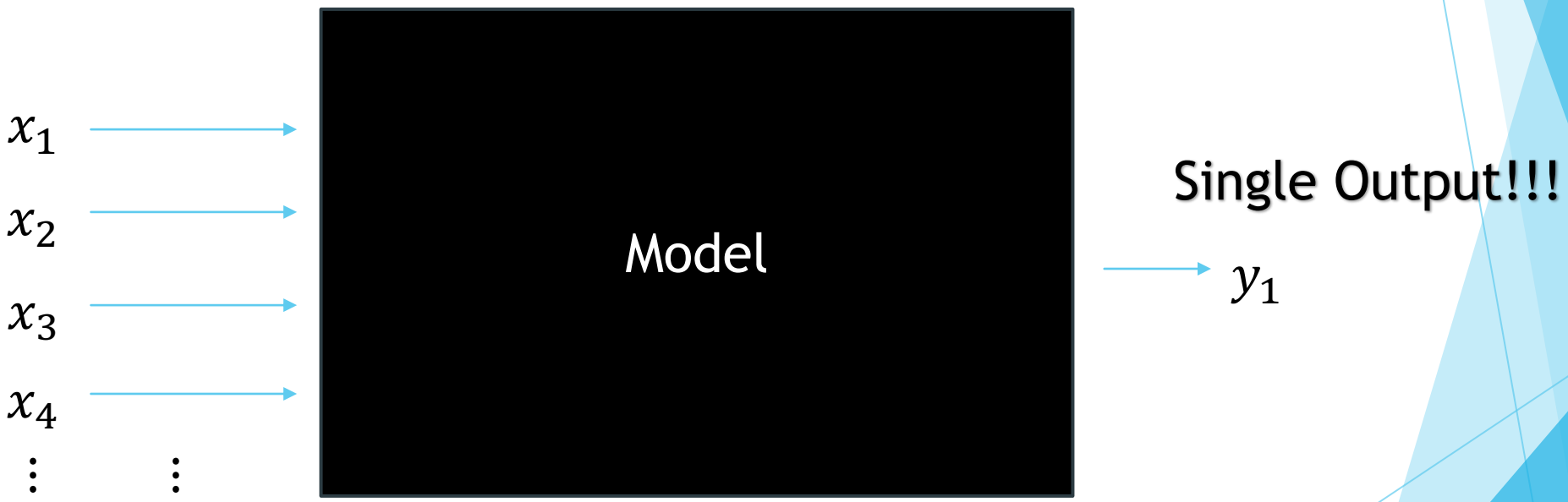
A common problem in ML

Regression

- In Machine Learning, Regression is a algorithm that can be trained to predict **real numbered outputs**; like temperature, stock price, etc.



Regression Model



Linear v.s Non-Linear

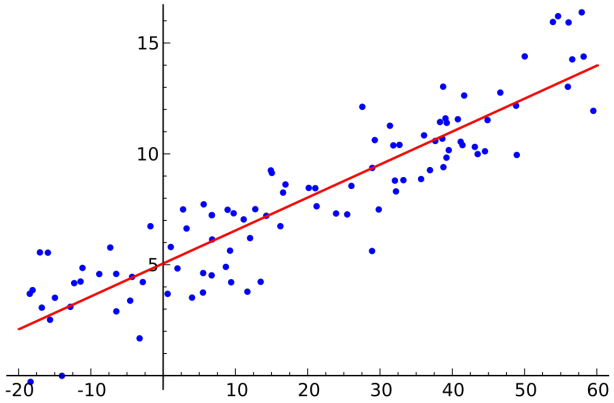
- Linearity is the property of a mathematical relationship or function which means that it can be graphically represented as a straight line. (From Wikipedia)
  - Linear: $y = b + c_1x_1 + c_2x_2 + c_3x_3$
  - Non-Linear: $y = b + c_1x_1^2 + c_2x_2 + c_3x_3$

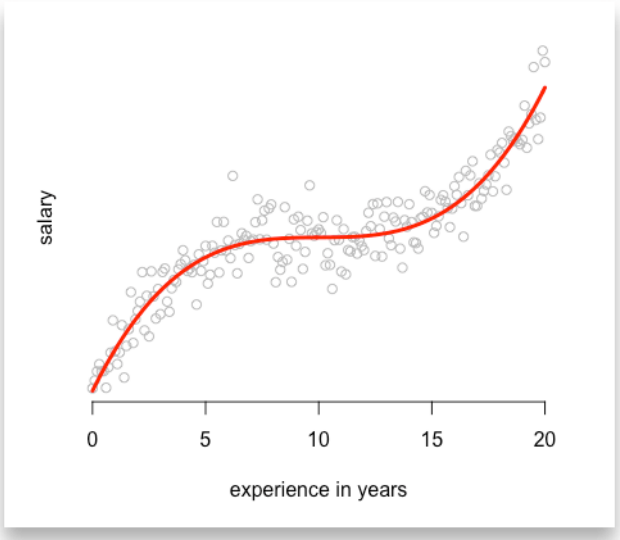
Linear v.s Non-Linear Model

- A model is linear when it's **coefficient(w)** is linear (not x).
  - $y = b + w_1x_1 + w_2x_2 + w_3x_3$ - Linear Model
  - $y = b + w_1x_1^2 + w_2x_2 + w_3x_3$ - Linear Model
  - $y = b + w_1^2x_1 + w_2x_2$ - Non-linear Model
  - $y = b + \frac{w_1x_1^2}{w_2x_2} + w_3x_3$ - Non-linear Model

Linear Regression

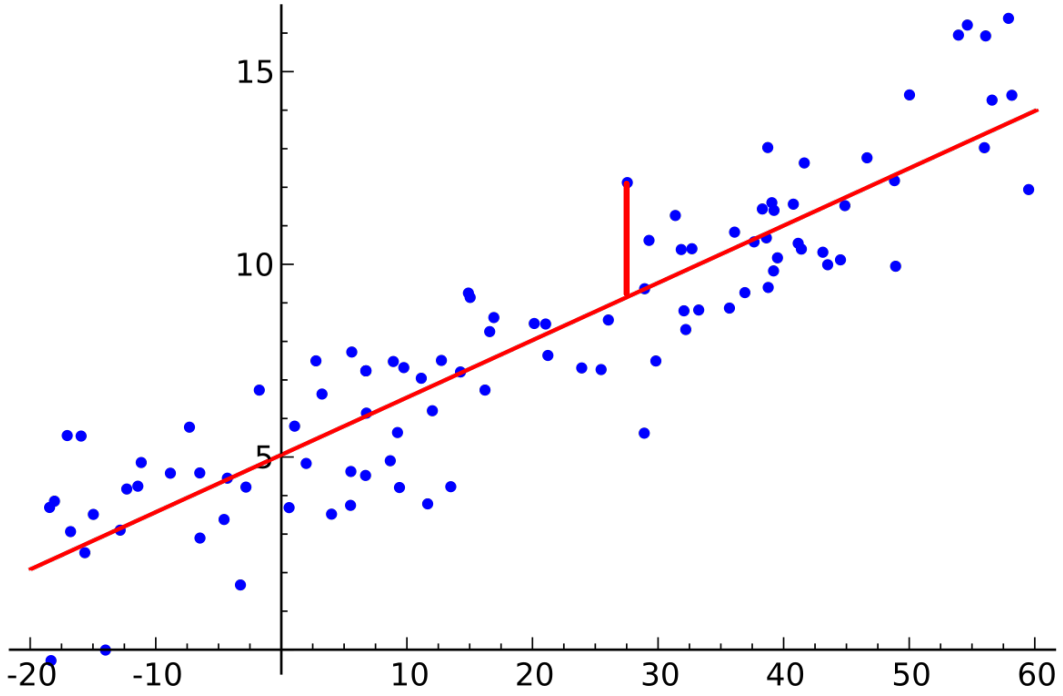
- The curve doesn't need to be a straight line.
- $y = \boldsymbol{w}\boldsymbol{x} + b$





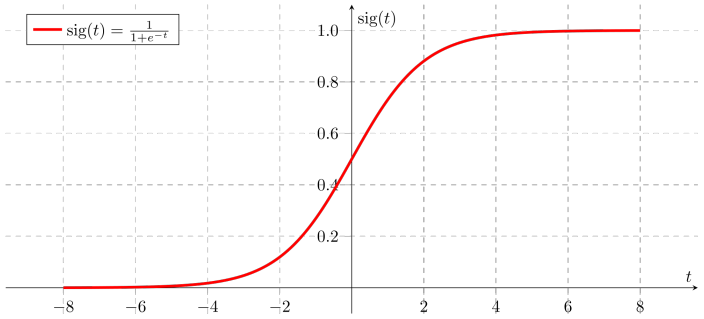
Error Function in Linear Regression

- Mean Squared Error(MSE)
- $E = \sum(y_i - \hat{y_i})^2$

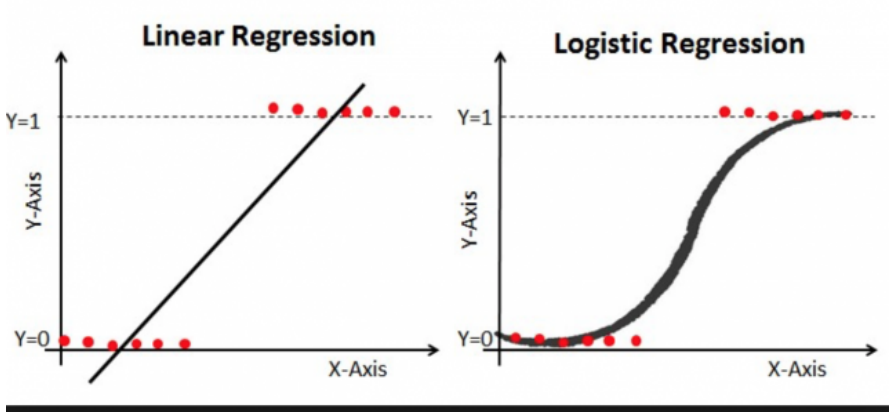


Logistic Regression

- Logistic is a kind of non-linear regression.
- It's used to solve probability problem such as binary classification.
- $y = \sigma(z), where\ z = \boldsymbol{wx} + b$
- $\sigma(z) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-\boldsymbol{z}}}$

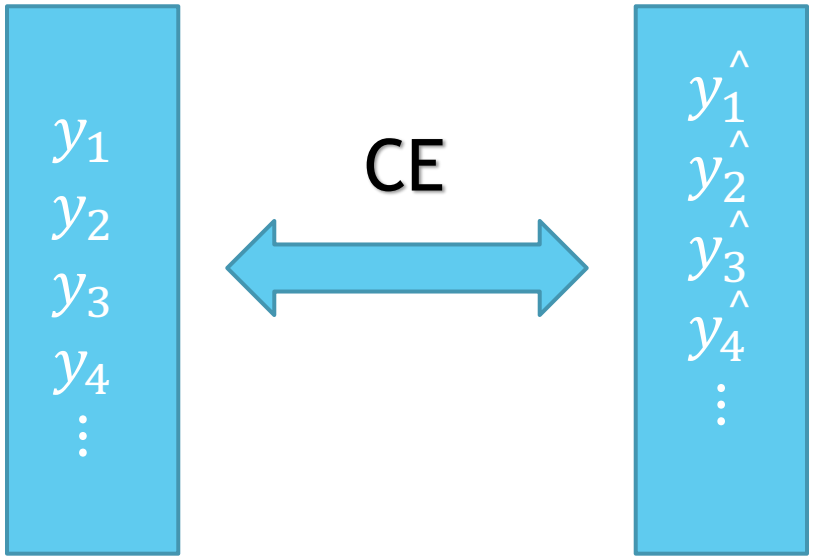


Sigmoid Function



Error Function in Logistic Regression

- Cross Entropy
- $E = -\sum(\hat{y_i}\log(y_i) - (1-\hat{y_i})\log(1-y_i))$



Why?

https://youtu.be/hSXFuypLukA?t=1305
(ML online course by HungYi Lee
Lecture 5 – Logistic Regression 21:45)

Classification

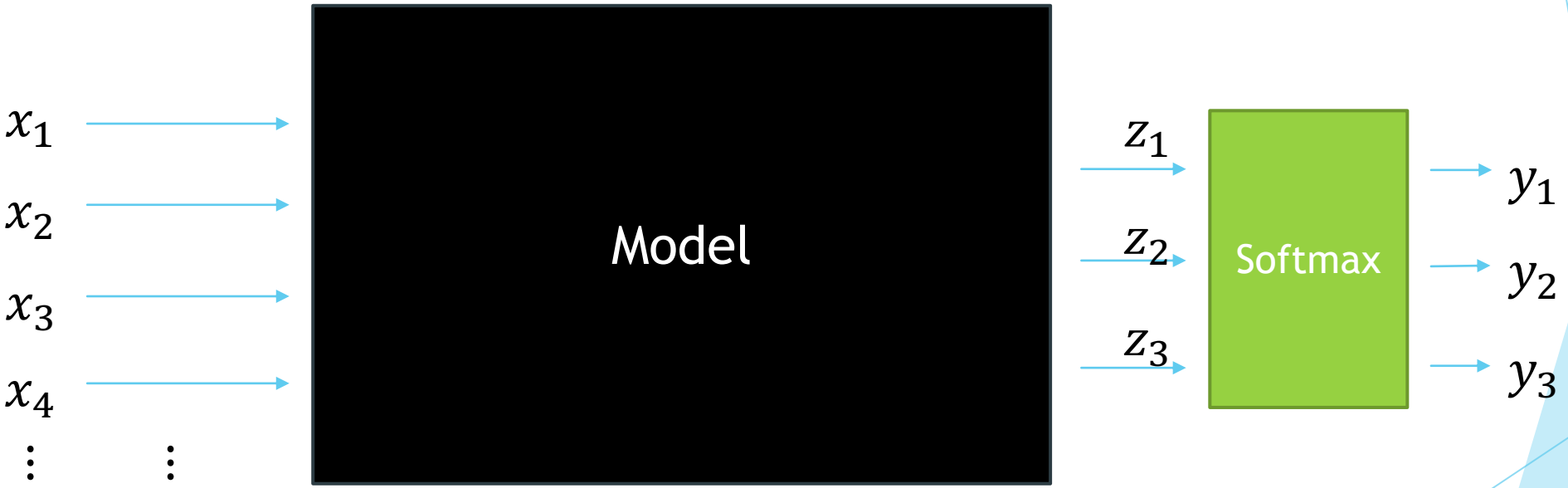
Binary Classification

- Logistic Regression
  - When output $y \geq 0.5$ … class1
  - When output $y < 0.5$ … class2

Multiple Classification

- Take 3 classes as example
  - We need 3 outputs, y_1, y_2, y_3, stand for the probability of each class.
    - $y_1 = \sigma(z_1), where\ z_1 = \boldsymbol{w_1 x} + b_1$
    - $y_2 = \sigma(z_2), where\ z_2 = \boldsymbol{w_2 x} + b_2$
    - $y_3 = \sigma(z_3), where\ z_3 = \boldsymbol{w_3 x} + b_3$
- Though each output is bounded between 0 to 1, the sum of them maybe larger or smaller than one, which disobeys the intuition of probability.
  - Solution: Softmax

Multiple Classification



Softmax

$$S(z_i) = \frac{e^{z_i}}{\sum_i^N e^{z_k}}$$

Gradient Descent

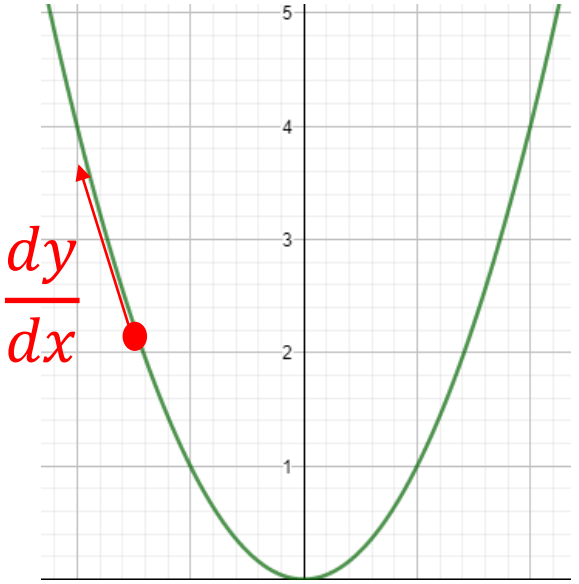
An effective algorithm to optimize ML models

Loss Function

- A.k.a Error Function
- For instance, in regression problems
  - Linear Regression: MSE ($\mathrm{L} = \sum(y_i - \hat{y_i})^2$)
  - Logistic Regression: Cross Entropy($\mathrm{L} = -\sum(\hat{y_i}\log(y_i) - (1 - \hat{y_i})\log(1 - y_i))$)

Optimization

- Learning Rule
  - $w_{n+1} = w_n + \eta \Delta w_n$
  - $\Delta w_n = -\frac{\partial L}{\partial w_n}$
  - $\eta = learning\ rate$



Loss Function

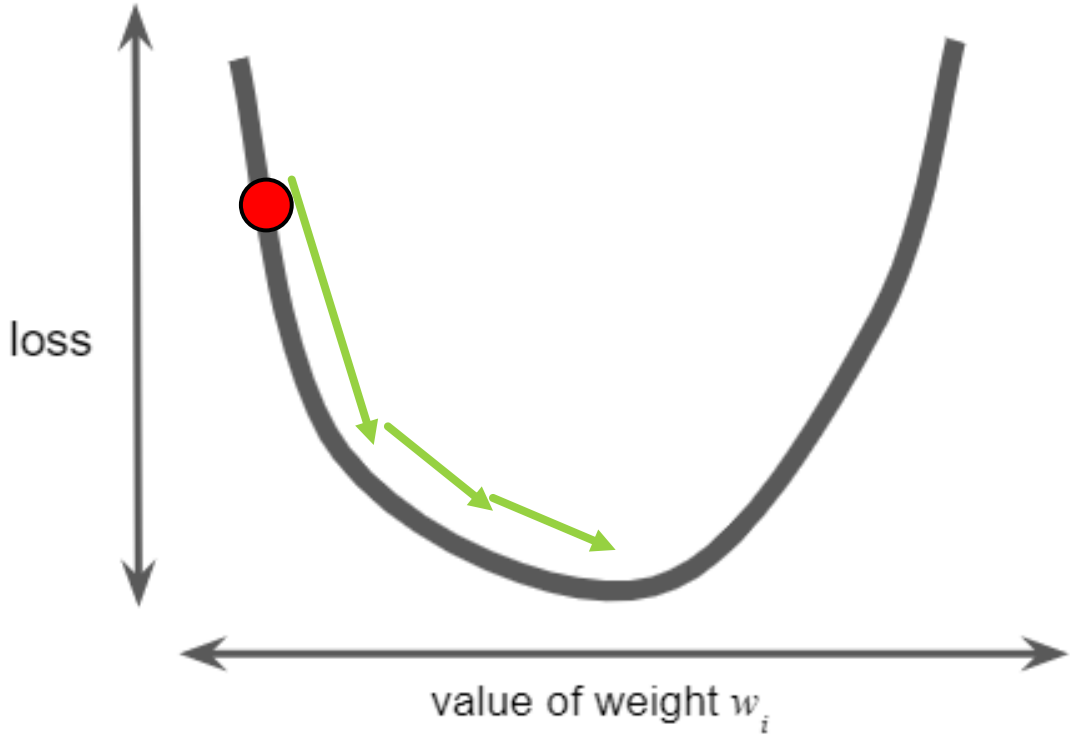
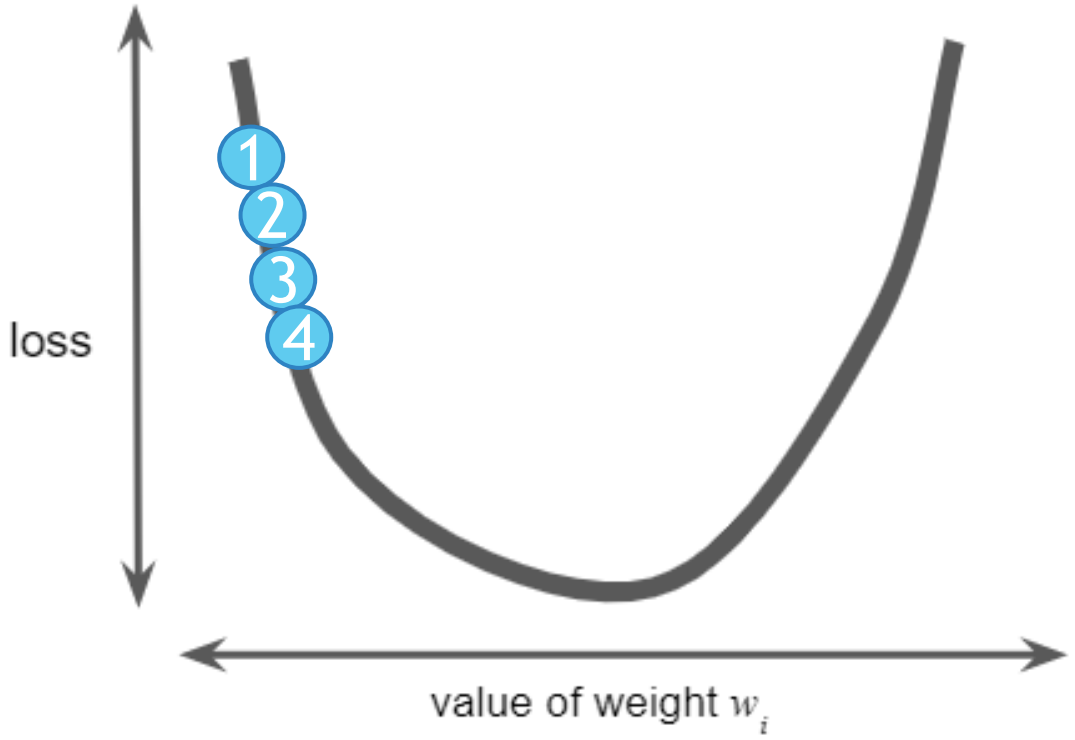


Figure from Google Crash Course

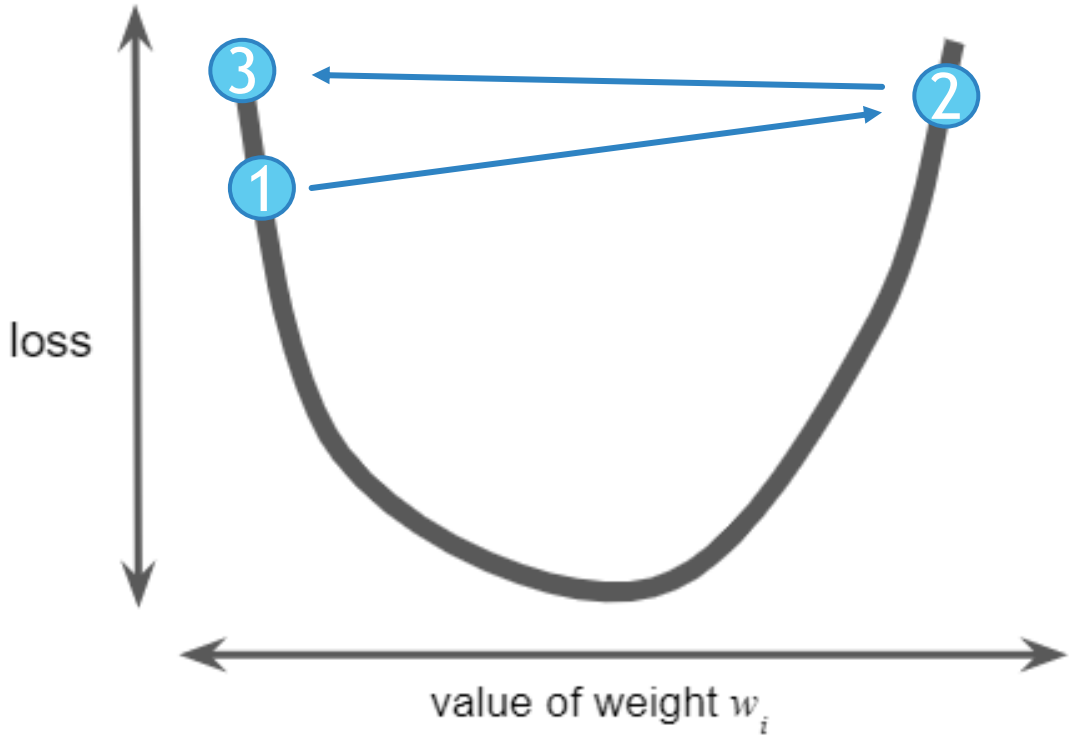
Effect by Learning Rate

Learning Rate is too small!!

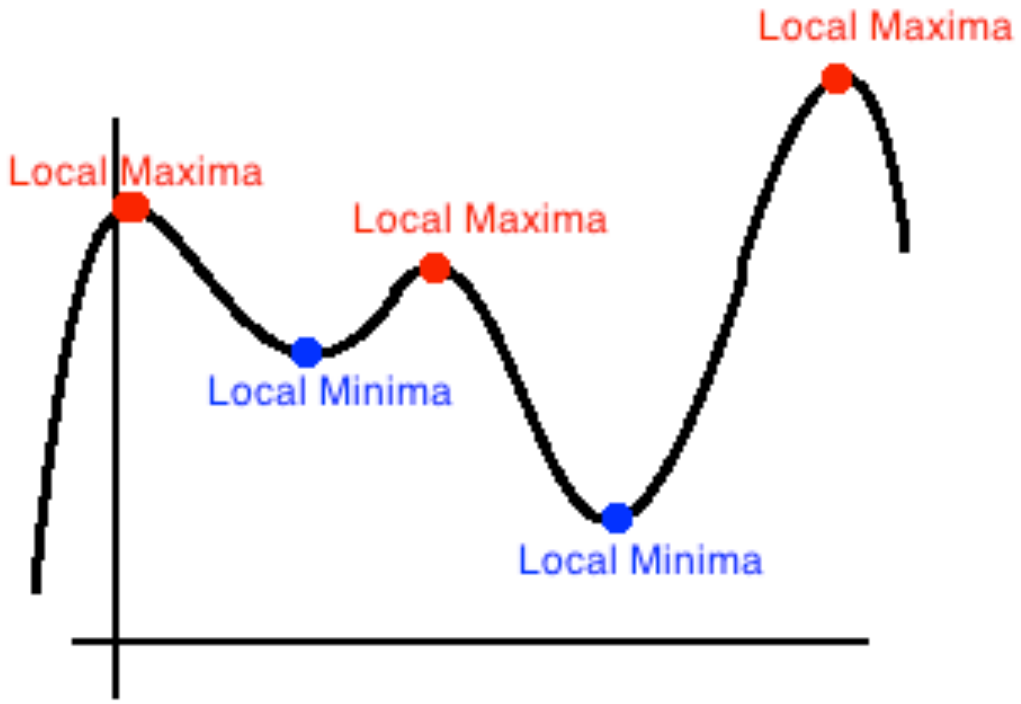


Effect by Learning Rate

Learning Rate is too big!!

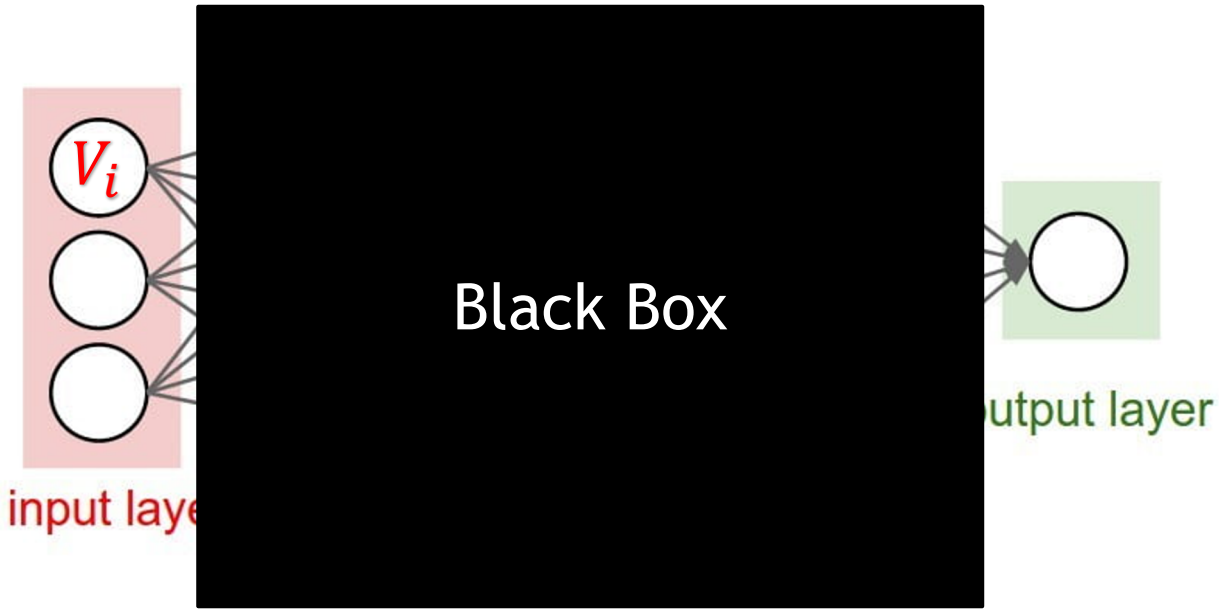


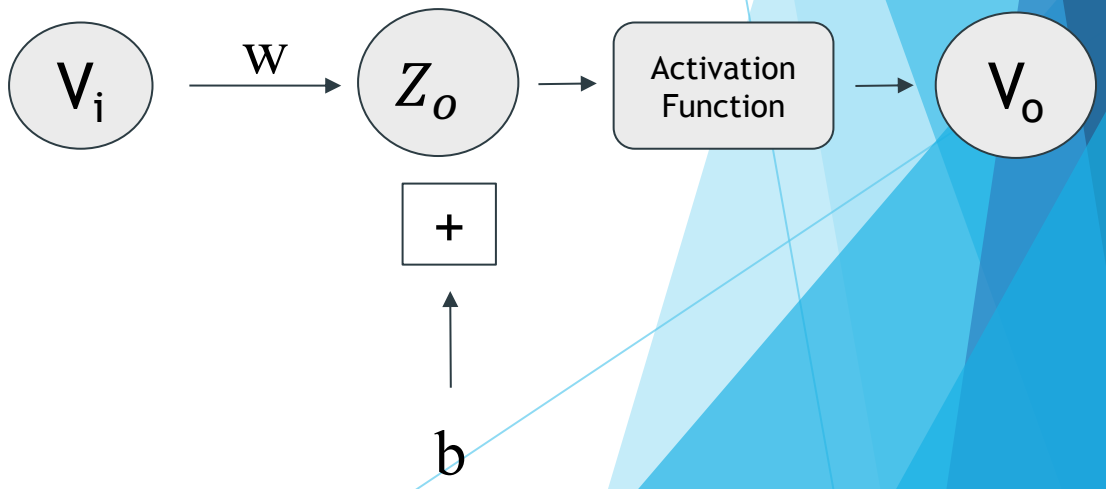
Local Minimum



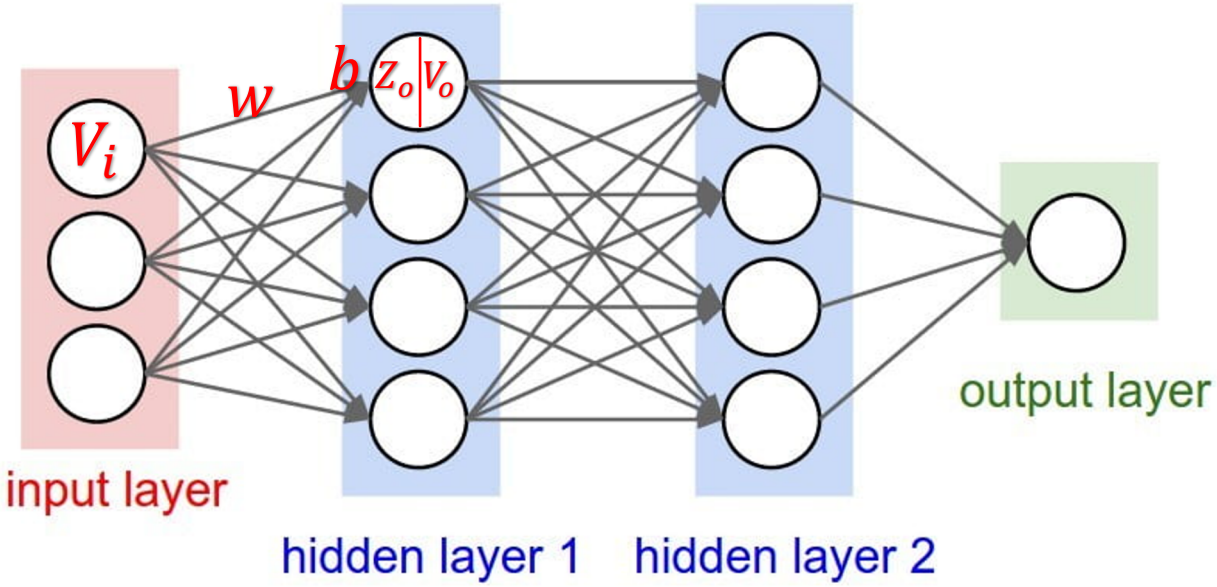
Artificial Neuron Network

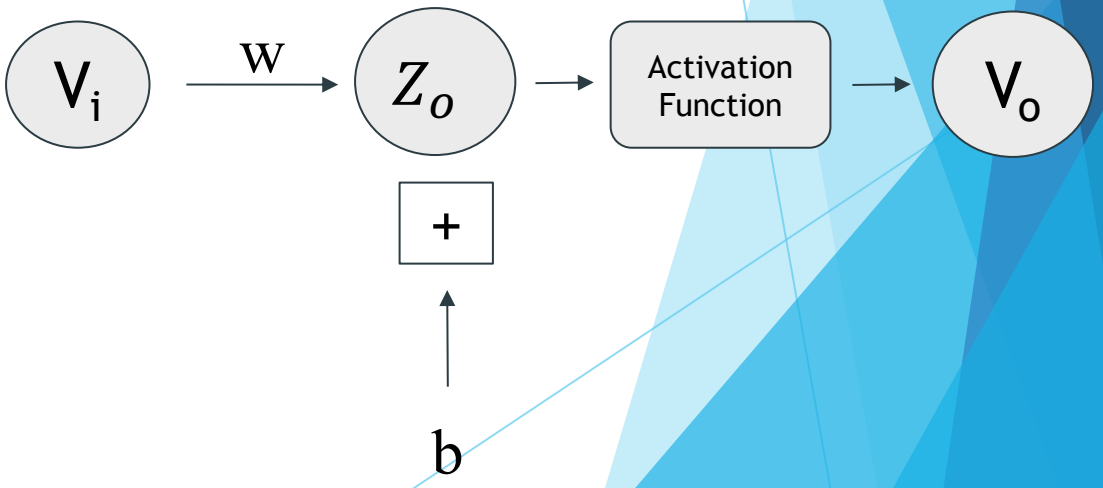
Artificial Neural Network(ANN)



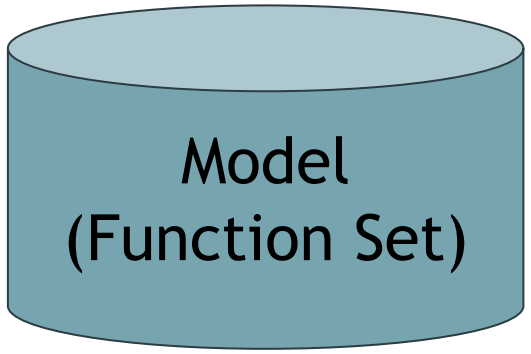


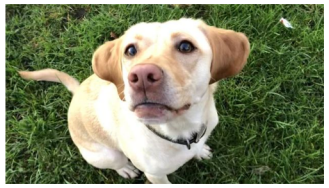
Artificial Neural Network(ANN)

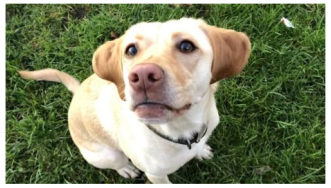




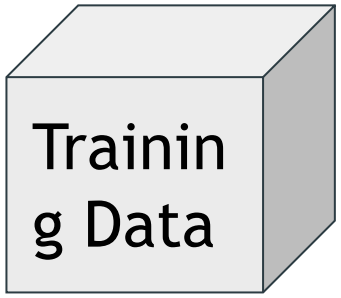
Model

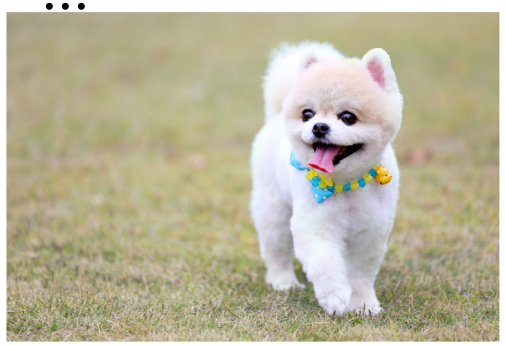


$f_1($  $) = \text{Cat}$

$f_2($  $) = \text{Dog}$ 

...





Dog

Cat

Model

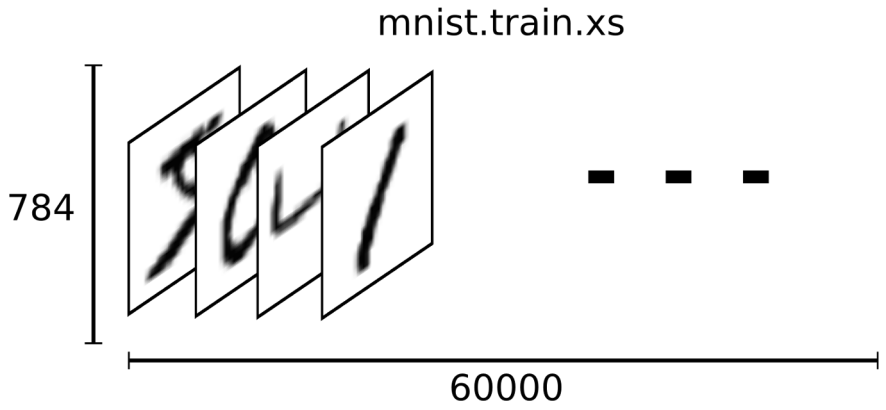
- Parameters: weights, biases
- Learning Method: Gradient Descent
- Training data
  - Teach the model what should good outputs be like

Training

Adjust parameters to optimize the model

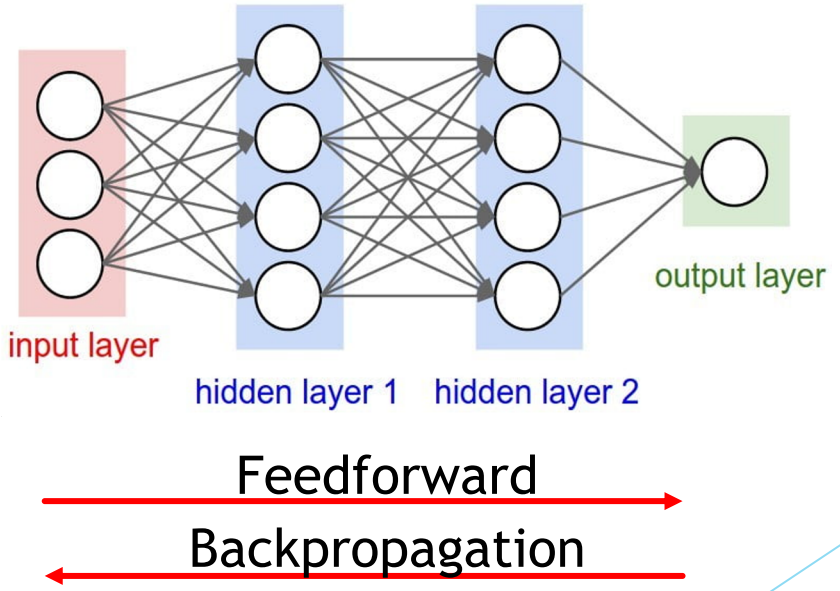
Learning Method

- Gradient Descent
  - Use the entire dataset to calculate the gradient
- Stochastic Gradient Descent
  - Use every single data to calculate the gradient
- Mini-Batch Gradient Descent
  - A compromise, calculate the gradient for every batch.



Bottleneck in Training Process

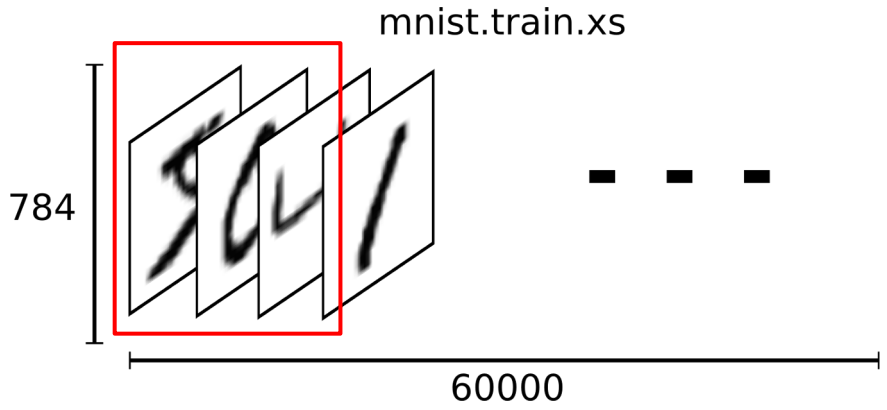
- Gradient Descent
  - Feedforward too many times
- Stochastic Gradient Descent
  - Backpropagation too many times
  - Noisy



(Trip to Taipei, Ask people)

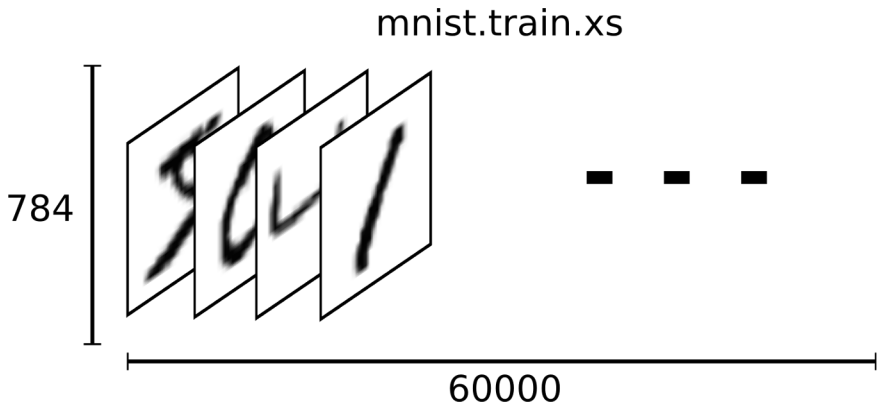
Batch size

- The amount of data we used to calculate gradient and adjust parameters for **each iteration**.



Epoch

- When we go through the whole dataset once, it’s called an epoch.



Batch Size & Epoch

- Example:
  - In MNIST Data, there are 60000 training images. If we set the batch size to 200, how many iterations we need to go through for one epoch?
  - 60000/200 = 3000

Lab

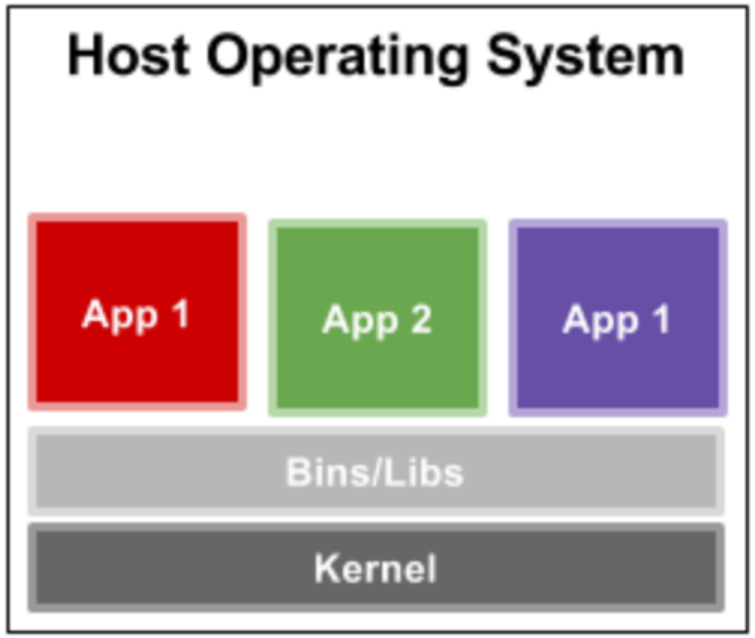
Environment

- GPU Server
- Jupyter Notebook

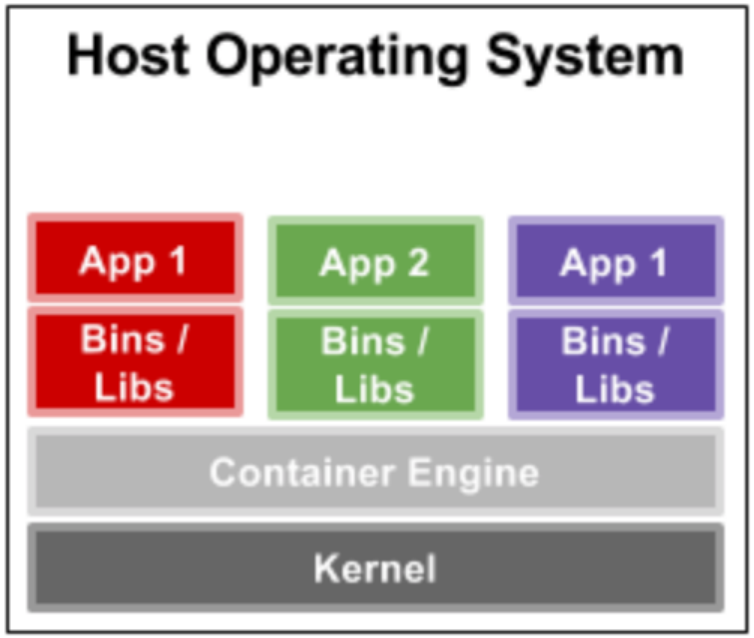
Caslab GPU Server

- With 6 Nvidia Geforce GTX 1080 Ti

Container



Native Running Applications



Containerized Applications

Container

- Already installed
  - Nvidia Driver
  - Cuda
  - Cudnn
  - python
  - pip3

Container

- Memory 4GB
- Disk 20GB
- 4 groups share 1 GPU

Connect to Container

- IP: 140.116.164.241
- Port: 80XX
  - SSH: XX = (Group Num - 1) * 2
  - Jupyter notebook: XX = SSH + 1
- For Example:
  - Group1
    - SSH: 8000
    - Jupyter Notebook: 8001
  - Group2
    - SSH: 8002
    - Jupyter Notebook: 8003
  - …

Connect to Container - Linux

```
ssh user@140.116.164.241 -p 8000
```

Default password: aoc2020

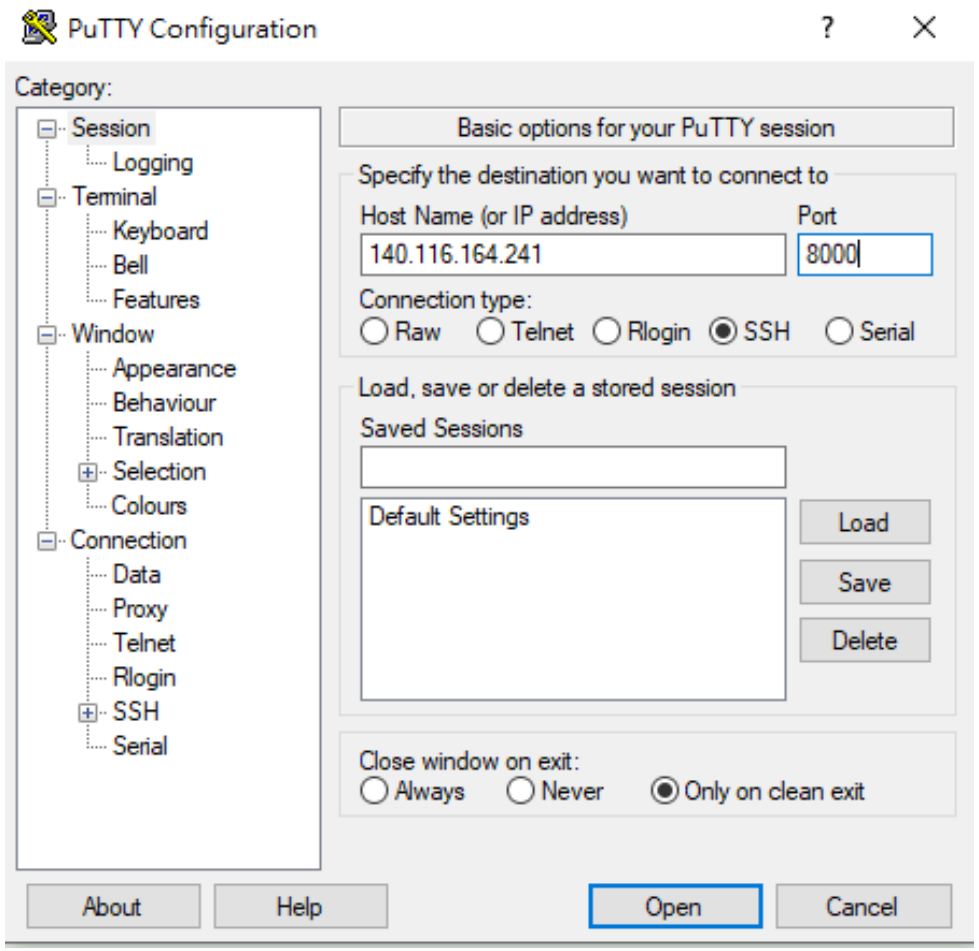
```
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

3 packages can be updated.
1 update is a security update.


##################################################
#                                                #
#    Dear classmates, to assure that everyone    #
#    can access GPU fairly, you have to add      #
#    some limitation in your code. For more      #
#    information, please read README.txt in      #
#                           ^^^^^^^^^^           #
#    your home directory.                        #
#    (Path: /home/user/README.txt)               #
#                                                #
##################################################


Last login: Tue Mar  3 06:10:35 2020 from 127.0.0.1
user@AIONCHIP1:~$
```

Connect to Container - Windows



Username: user
Password: aoc2020

Password

- $ passwd

Jupyter notebook

- Login to your container account
- $ jupyter notebook

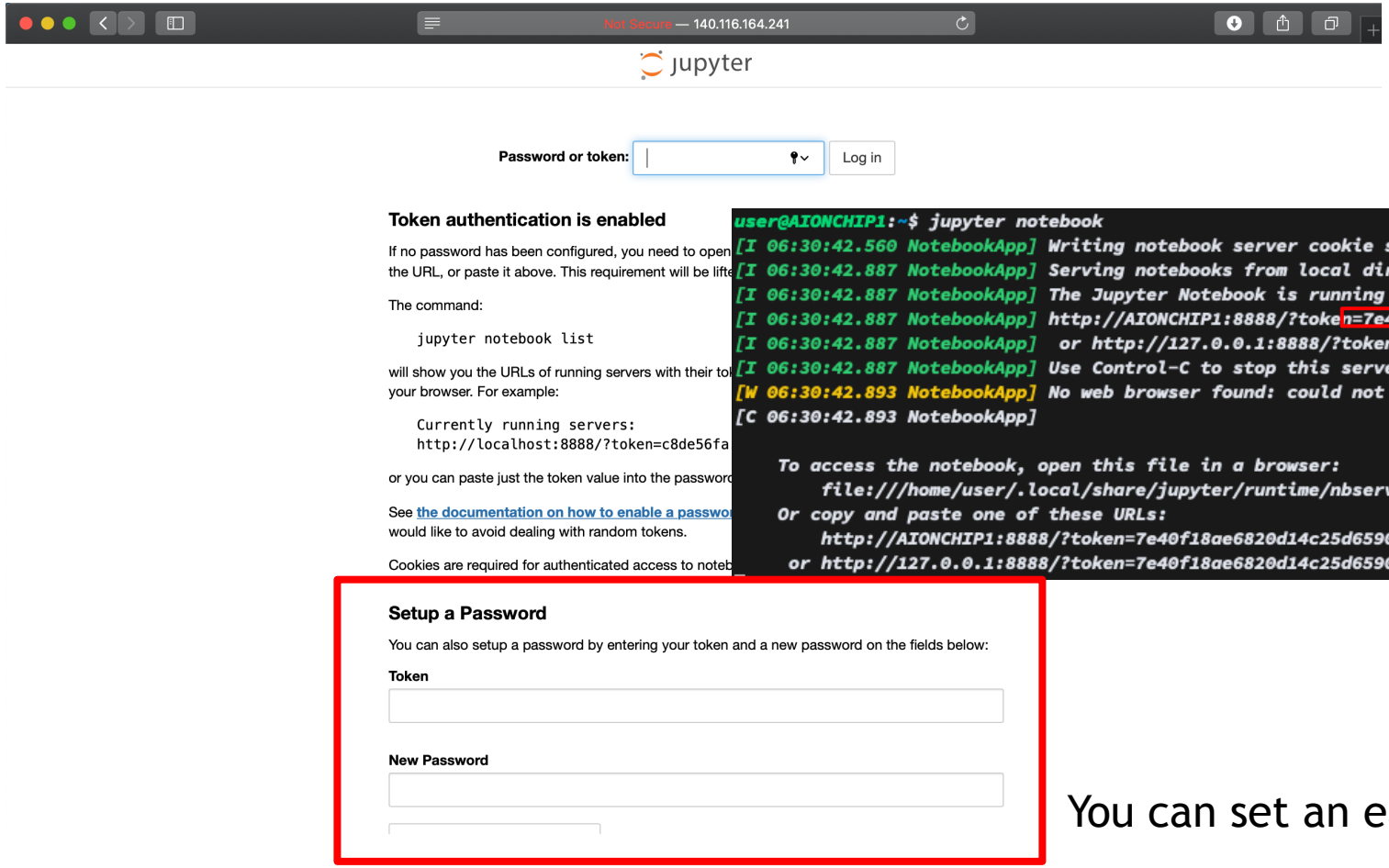
```
user@AIONCHIP1:~$ jupyter notebook
[I 06:30:42.560 NotebookApp] Writing notebook server cookie secret to /home/user/.local/share/jupyter/runtime/notebook_cookie_secret
[I 06:30:42.887 NotebookApp] Serving notebooks from local directory: /home/user
[I 06:30:42.887 NotebookApp] The Jupyter Notebook is running at:
[I 06:30:42.887 NotebookApp] http://AIONCHIP1:8888/?token=7e40f18ae6820d14c25d6590eece4913b244419990a77420
[I 06:30:42.887 NotebookApp]  or http://127.0.0.1:8888/?token=7e40f18ae6820d14c25d6590eece4913b244419990a77420
[I 06:30:42.887 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 06:30:42.893 NotebookApp] No web browser found: could not locate runnable browser.
[C 06:30:42.893 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/user/.local/share/jupyter/runtime/nbserver-937-open.html
    Or copy and paste one of these URLs:
        http://AIONCHIP1:8888/?token=7e40f18ae6820d14c25d6590eece4913b244419990a77420
     or http://127.0.0.1:8888/?token=7e40f18ae6820d14c25d6590eece4913b244419990a77420
```

Jupyter notebook

- Open a web browser
  - http://140.116.164.241:{your jupyter notebook port}
  - For example
    - Group1: http://140.116.164.241:8001

Jupyter notebook



You can set an easy password for quick login

Environment Variable

- Do the command below every time you open a new terminal
  - $ export LD_LIBRARY_PATH=/usr/local/cuda-10.0/lib64

Remember to do it before you start jupyter notebook!

Limit GPU usage

- keras

```
import keras
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
tf_config = tf.ConfigProto()
tf_config.gpu_options.per_process_gpu_memory_fraction = 0.5 # 50% GPU memory usage at most
tf_config.gpu_options.allow_growth = True
set_session(tf.Session(config=tf_config))
```

- Tensorflow

```
import tensorflow as tf
tf_config = tf.ConfigProto()
tf_config.gpu_options.per_process_gpu_memory_fraction = 0.5 # 50% GPU memory usage at most
tf_config.gpu_options.allow_growth=True
session = tf.Session(config=tf_config)
```

Keras

The python deep learning library(framework)

Commonly used ML frameworks

TensorFlow

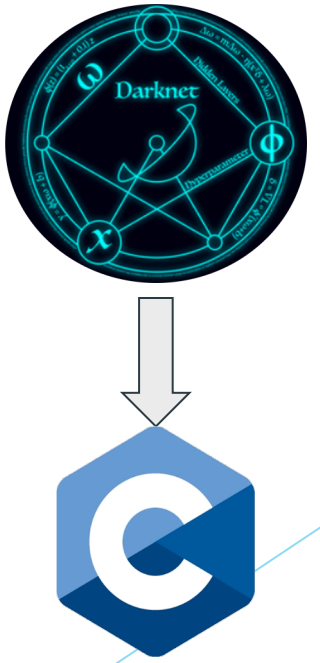
Keras

PyTorch

Caffe2

Framework and Programming Language

- A framework is built on top of a programming language to aid in a certain type of computer program, such as AI, web server, image processing etc.



Tensorflow 2.0

tf.keras is Tensorflow's high-level API for building and training deep learning models. - From Tensorflow official website

Programming

Building blocks

Example

ANN for Handwriting Recognition

Required Packages

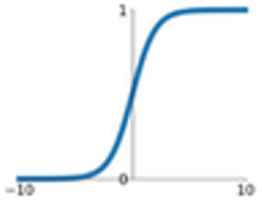
- tensorflow-gpu (1.14)
  - $ pip3 install tensorflow-gpu
- keras
  - $ pip3 install keras
- numpy
  - $ pip3 install numpy
- matplotlib
  - $ pip3 install matplotlib

Activation Function

Activation Functions

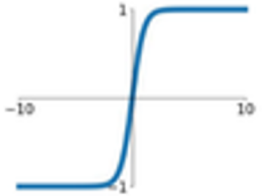
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



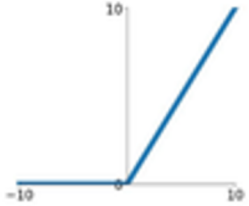
tanh

$$\tanh(x)$$



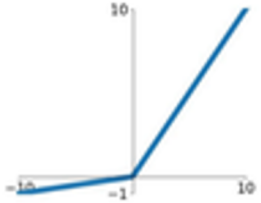
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

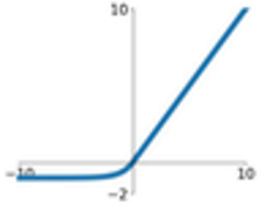


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Numpy APIs

- np.astype()

```
import numpy as np

a_f = np.array([1.5, 1.7, 5.4, 4.3])
a_i = a_f.astype('int32')

print("a_float: ",a_f)
print("a_int: ",a_i)
```

```
a_float:  [1.5 1.7 5.4 4.3]
a_int:  [1 1 5 4]
```

Numpy APIs

- np.reshape()

```
import numpy as np
n = np.array([[[1,1,1,1,1],[2,2,2,2,2]],
              [[3,3,3,3,3],[4,4,4,4,4]],
              [[5,5,5,5,5],[6,6,6,6,6]]])
print("The shape of n:",n.shape)
print("n_[3,2,5]:\n",n)
print("n_[3,10]:\n",n.reshape(3,10))
```

```
The shape of n: (3, 2, 5)
n_[3,2,5]:
 [[[1 1 1 1 1]
  [2 2 2 2 2]]

 [[3 3 3 3 3]
  [4 4 4 4 4]]

 [[5 5 5 5 5]
  [6 6 6 6 6]]]
n_[3,10]:
 [[1 1 1 1 1 2 2 2 2 2]
 [3 3 3 3 3 4 4 4 4 4]
 [5 5 5 5 5 6 6 6 6 6]]
```

One hot encoding

keras.utils.to_catergorical()

```
import keras
import numpy as np

n = np.array([1,0,5,4,3])
n_OneHot = keras.utils.to_categorical(n,6) #6種 output
print(n_OneHot)
```

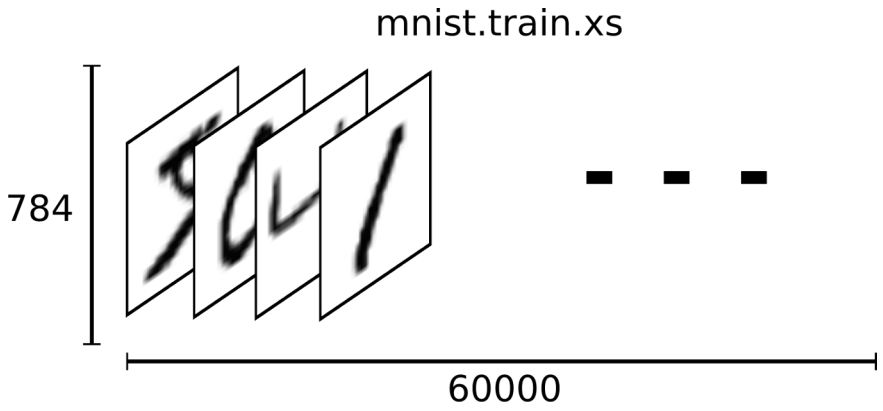
```
[[0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0.]]
```

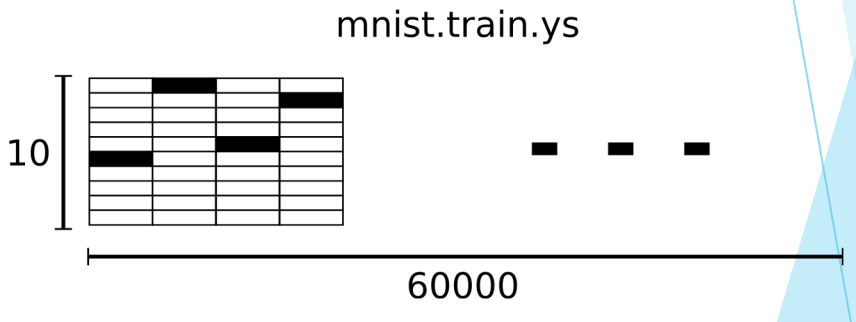
MNIST Dataset

The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems.

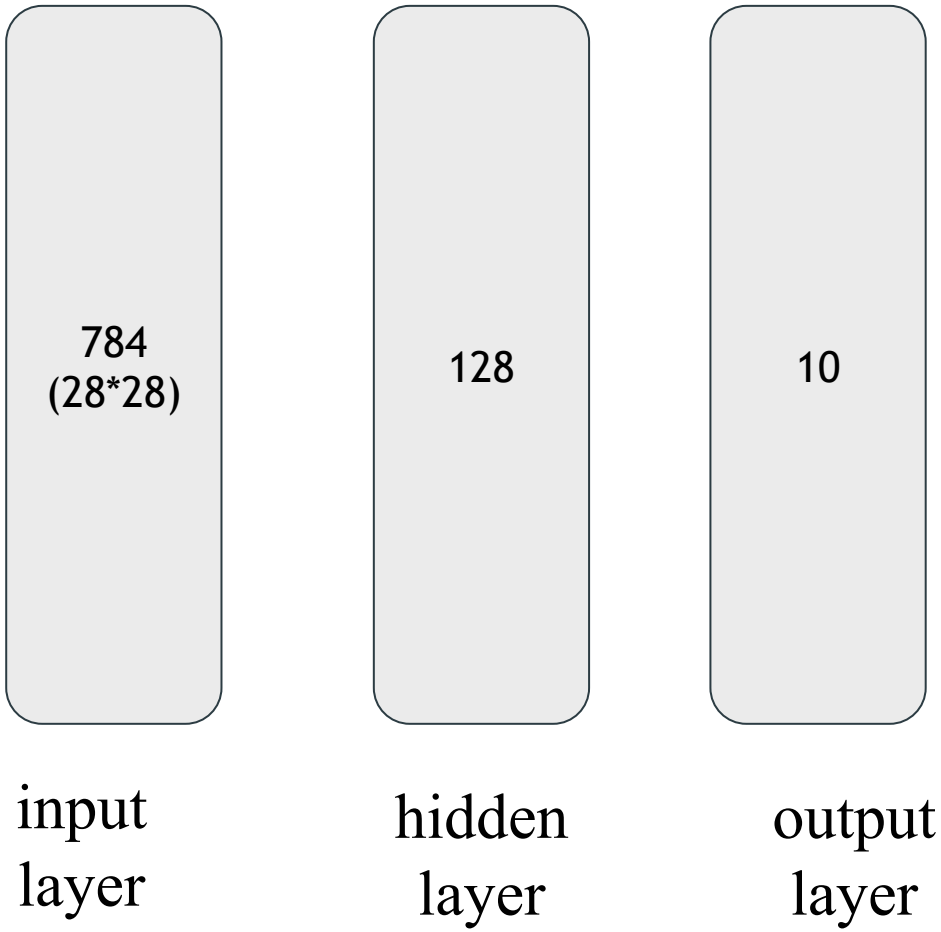
Image size:
28*28

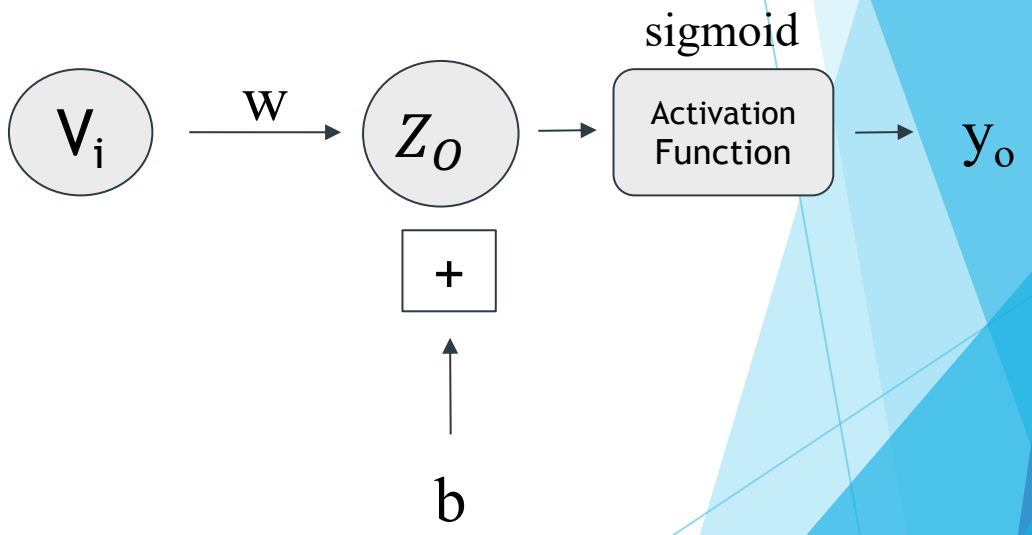
MNIST dataset





Model Architecture





Training Hyperparameters

Epochs = 20

Optimizer = Stochastic Gradient Descent (SGD)

Learning Rate = 0.01

Batch Size = 200

Activation Function1: Sigmoid

Activation Function2: Softmax

Note: In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

Check if GPU is available

- If output false, check if the environment variable is set.

```
import tensorflow as tf
print(tf.test.is_gpu_available())
```

Code (0/8) – limit GPU usage

```
import keras
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
tf_config = tf.ConfigProto()
tf_config.gpu_options.per_process_gpu_memory_fraction = 0.5 # 50% GPU memory usage at most
tf_config.gpu_options.allow_growth = True
set_session(tf.Session(config=tf_config))
```

Code (1/8) - Import Libraries

```
%matplotlib inline
import keras
from keras.layers import Dense
from keras import Sequential
import numpy as np
import matplotlib.pyplot as plt
import keras.datasets.mnist as mnist

# Define hyperparameter
EPOCHS = 20
LEARNING_RATE = 0.01
BATCH_SIZE = 200
```

Code(2/8) - Read MNIST dataset

```
# Read MNIST dataset
# mnist.load_data() method returns a tuple. Take a look here if you are confused about it
# https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
plt.imshow(x_train[0], "gray")
```

Code(3/8) - Data Preprocessing

```
# Data preprocess
# Change data type from uint8 to float32
# Because we is going to do normalization
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
# Normalization
x_train = x_train/255
x_test = x_test/255
# Flatten the array
x_train = x_train.reshape(x_train.shape[0], 28*28)
x_test = x_test.reshape(x_test.shape[0], 28*28)
# One-hot encode the labels
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

Code(4/8) - Build the Model

```
# Build the model
# Input shape needs to be specified in the first layer
# https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
my_model = Sequential()
my_model.add(Dense(input_shape = (784, ), units = 128, activation = "sigmoid"))
my_model.add(Dense(units = 10, activation = "softmax"))
my_model.summary()
```

Code(5/8) - Configure and Fit the Model

```
# Configure the model for training
my_model.compile(optimizer=keras.optimizers.SGD(learning_rate=LEARNING_RATE),
                 loss="categorical_crossentropy", metrics=["acc"])

# Feed the input data to this model and start training
Train_history = my_model.fit(x=x_train, y=y_train,
                         batch_size=BATCH_SIZE,
                         epochs=EPOCHS,
                         verbose=1,
                         validation_split=0.2)
```

Code(6/8) - Plot the Training Process

```
# Plot the training process
# Plot Accuracy curve
xaxis = np.linspace(1, EPOCHS, EPOCHS)
plt.figure()
plt.plot(Train_history.history["acc"])
plt.title("Training Process - Acc")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.xticks(xaxis)
plt.ylim(0, 1)
plt.show()
# Plot Loss curve
plt.figure()
plt.plot(Train_history.history["loss"])
plt.title("Training Process - Loss")
plt.xlabel("Epoch")
plt.xticks(xaxis)
plt.ylabel("Loss")
plt.show()
```

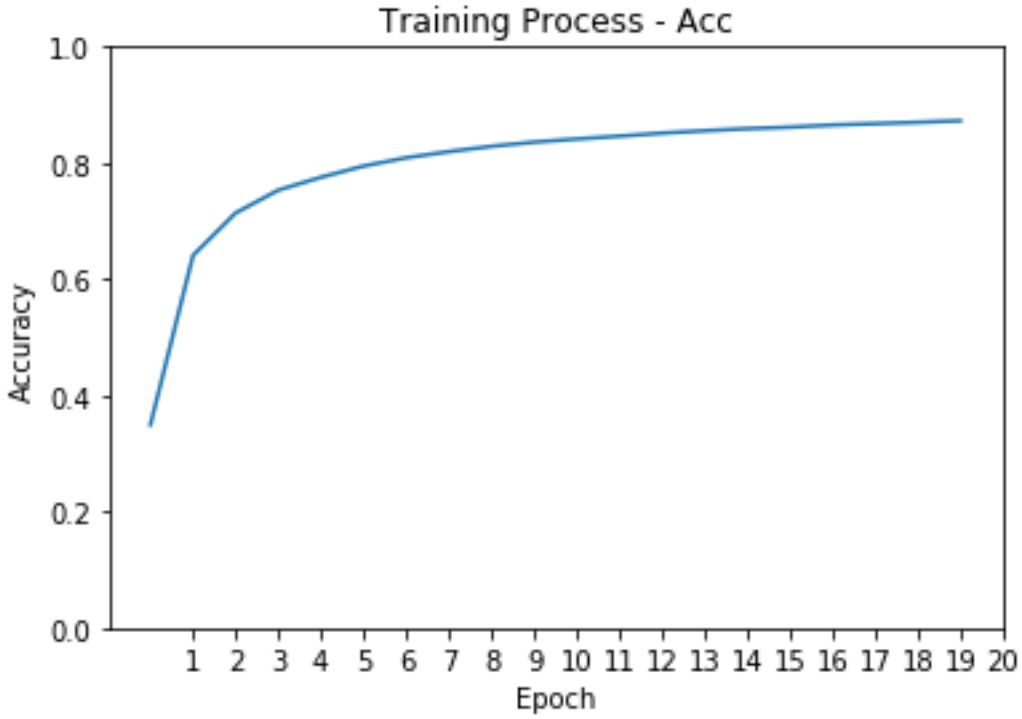
Code(7/8) - Evaluate the Model

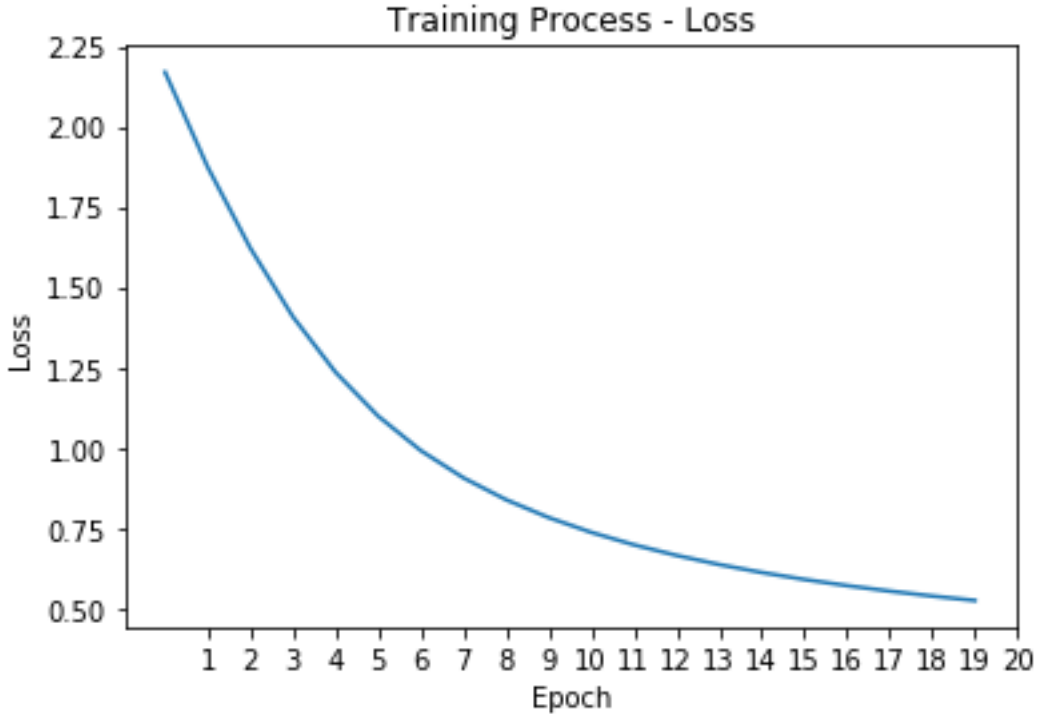
```
# Evaluation
Test_result = my_model.evaluate(x = x_test, y = y_test, batch_size = x_test.shape[0])
print("Loss on testing set: %f, Accuracy on testing set: %f" % (Test_result[0], Test_result[1]))
```

Code(8/8) - Save the Model

```
# Save the model
my_model.save("MyFirstModel.h5")
```

Result

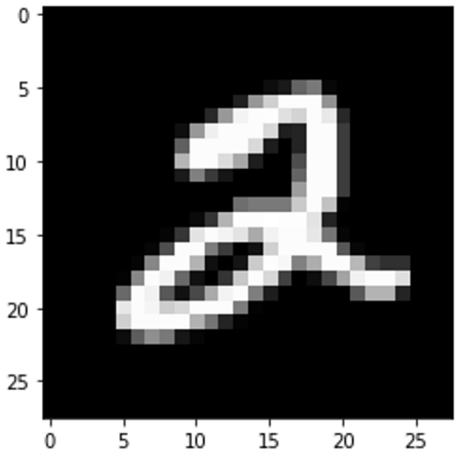





```
Loss on testing set: 0.491744, Accuracy on testing set: 0.883600
```

Reuse the model

Create another file




```
%matplotlib inline
import tensorflow as tf
import tensorflow.keras.datasets.mnist as mnist
import matplotlib.pyplot as plt
import numpy as np
retreived_model = tf.keras.models.load_model("./MyFirstModel.h5")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
plt.figure()
plt.imshow(x_train[5], "gray")
test_image = x_train[5]
test_image = test_image.reshape(1,784)
prediction = retreived_model.predict(test_image)
print(np.argmax(prediction))
```

Output: 2

Practice

Training Issues

Part1

- Watch these two videos
  - https://www.youtube.com/watch?v=L8unuZNpWw8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=15
  - https://www.youtube.com/watch?v=Ky1ku1miDow&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=17
- Reproduce the experiments in the videos and organize them into a report.
  - Screenshot of your experiment
  - Modified hyperparameters
  - Effects by the modifications and the reasons
  - The model summary and result of your best try on MNIST data
  - What you clarified and still confused about

Part2

- From those videos, there are some important concepts we've not covered yet. Do some simple research about the three topics below and explain what they are as detailly as you can. Bonus point will be given if you cover more than the three topics(Whatever you just learned about deep learning when during the research).
  - Overfitting and Underfitting
  - Gradient Vanishing
  - Dropout
  - (Bonus topics): Batch Normalization, Adam, RMSProp, momentum…

Thank you