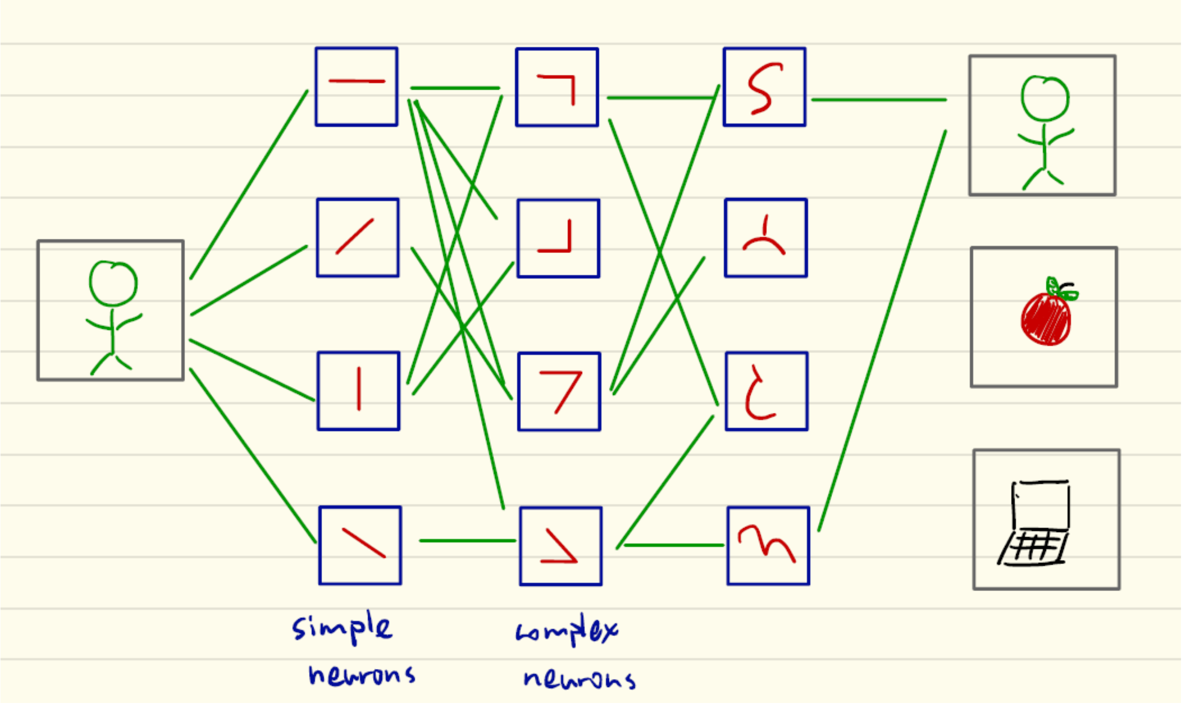
AI Lab for Wireless Communications

Week4 – Deep Learning

Speaker: Kuan-Yu Lin

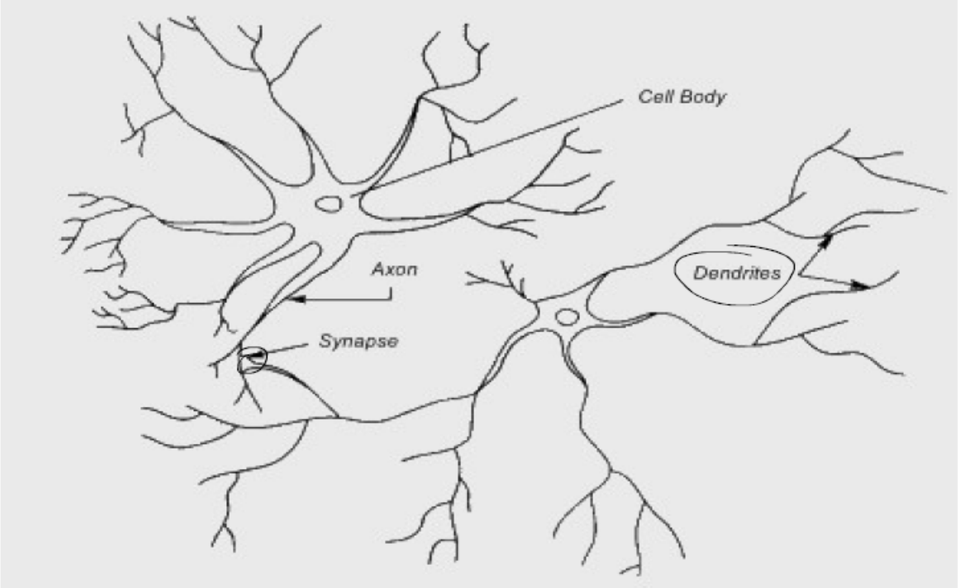
What is Deep Learning?

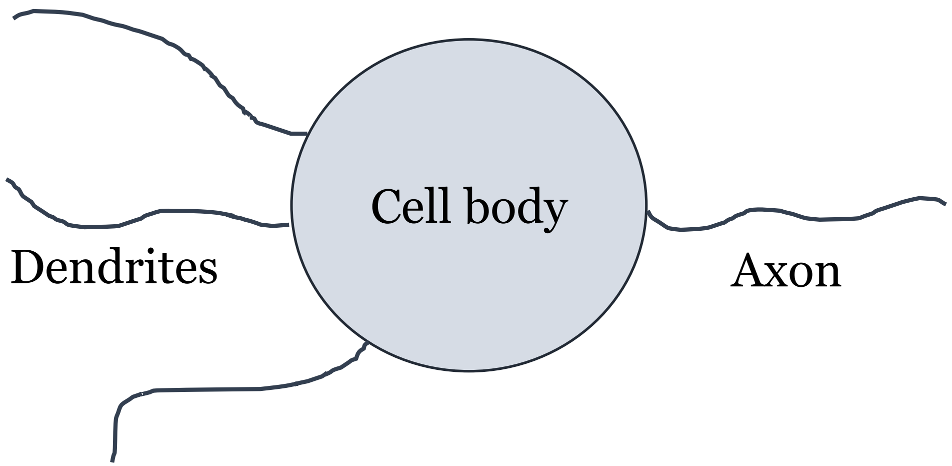
- Deep learning is a branch of ML and is based on neural network (NN)
- In recent years, deep learning is popular because of image processing.



Biological Neuron

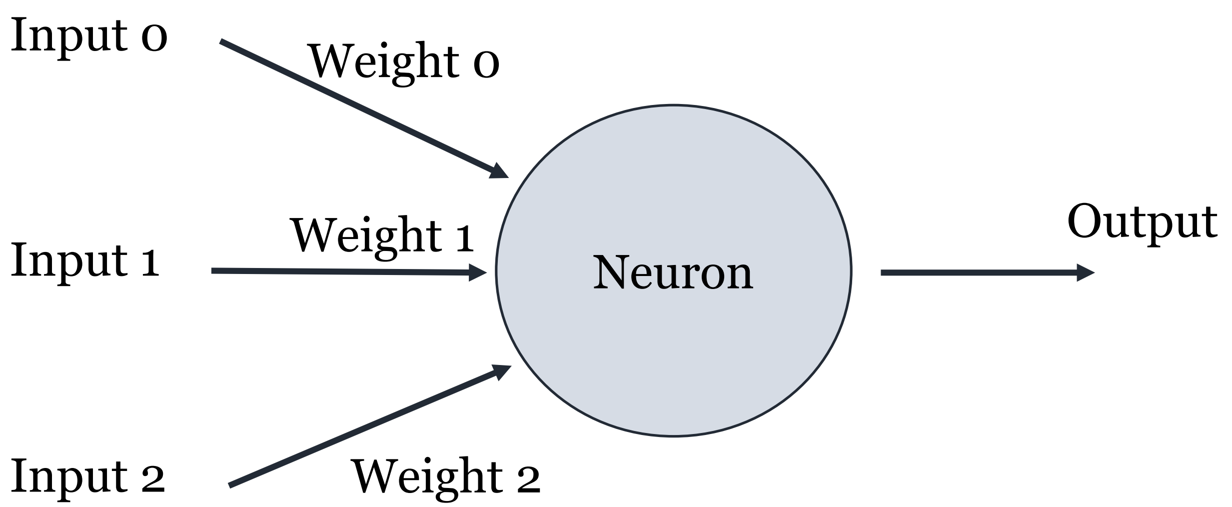
- Several neurons are connected to one another to form a neural network or a layer of a neural network





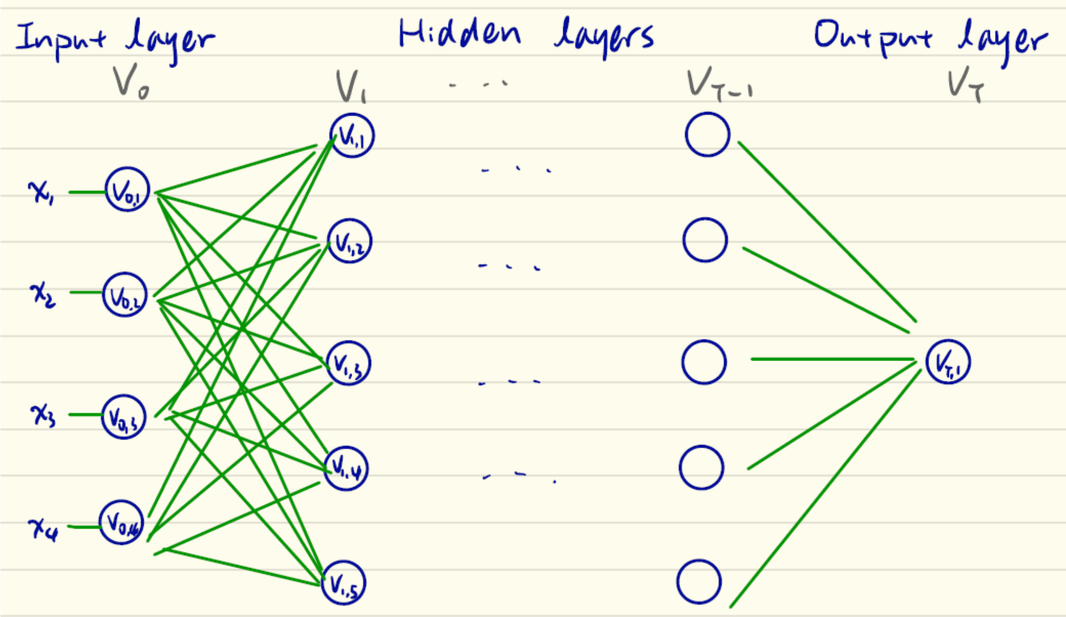
Artificial Neuron

- Artificial neuron closely mimics the characteristics of biological neuron

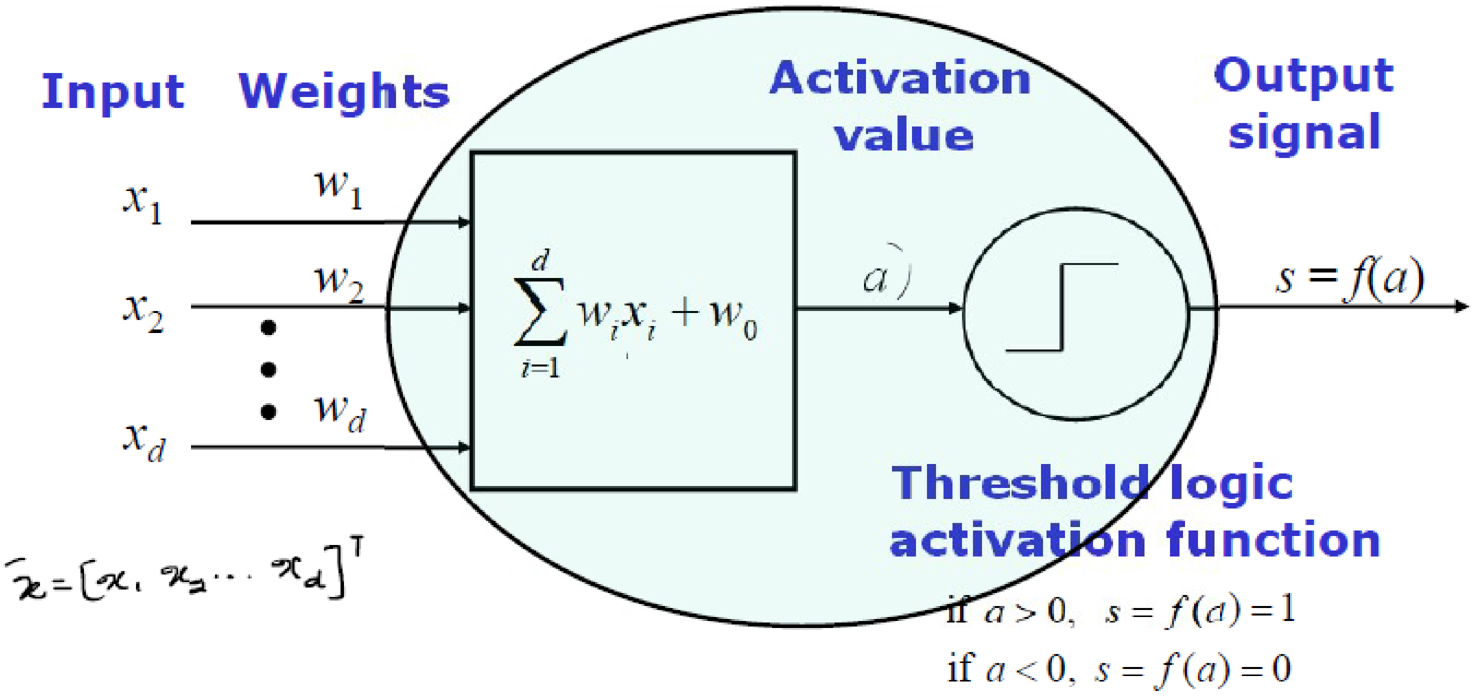


Deep Neural Network

- $x_1 \sim x_n$: Inputs, V_o: Input layer, $V_1 \sim V_{T-1}$: hidden layers, V_T: output layer
- T: Depth of the network
- Deep NN or Deep learning: if T>2
- Associate with each edge is a weight $W(V_{t,r}, V_{t+1,r}, j)$



Neuron with Threshold Logic Activation Function



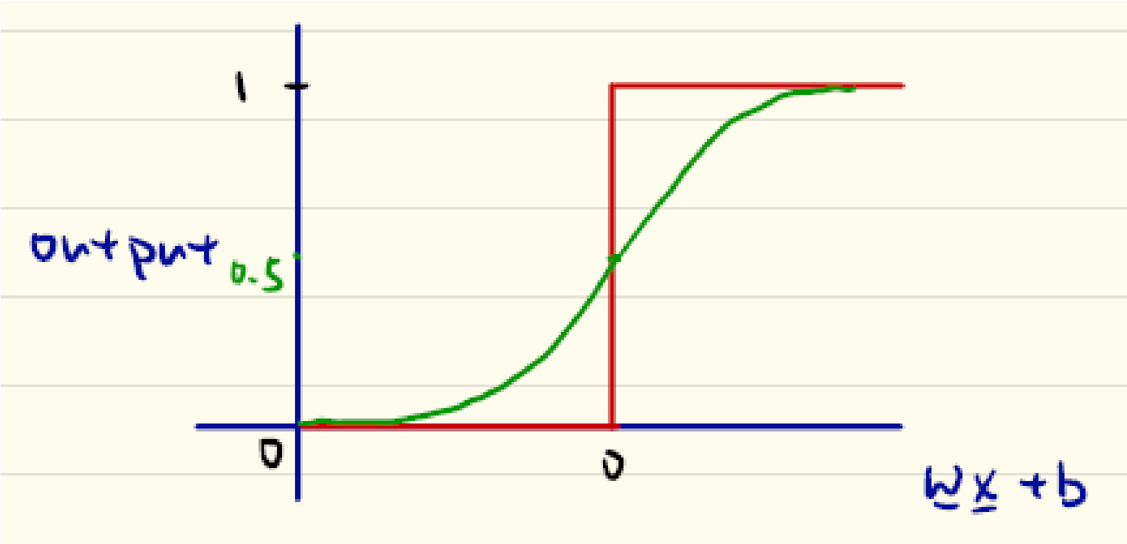
W.S.McCulloch and W.Pitts. A logival calculus of the ideas imminent in nervous activity. 1943.

Activation Function

- Sigmoid function
- Hyperbolic tangent function
- ReLu (Rectified Linear Units)

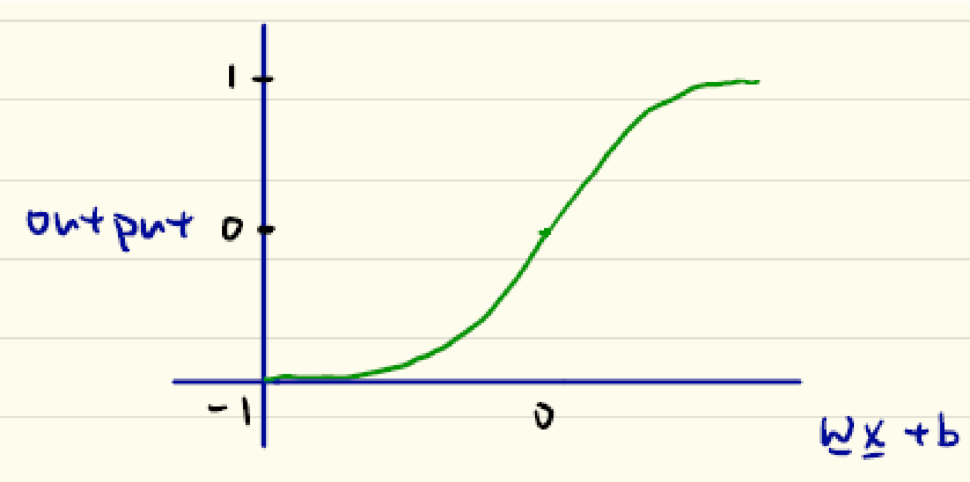
Sigmoid Function

- $\sigma(z) = \frac{1}{1+e^{-z}}$
- It outputs soft value in (0,1)
- $\sigma(z) \rightarrow 0 \; as \; z \rightarrow -\infty$
- $\sigma(z) = \frac{1}{2}$ if $\mathrm{z} = 0$
- $\sigma(z) \rightarrow 1 \; as \; z \rightarrow \infty$



Hyperbolic Tangent Function

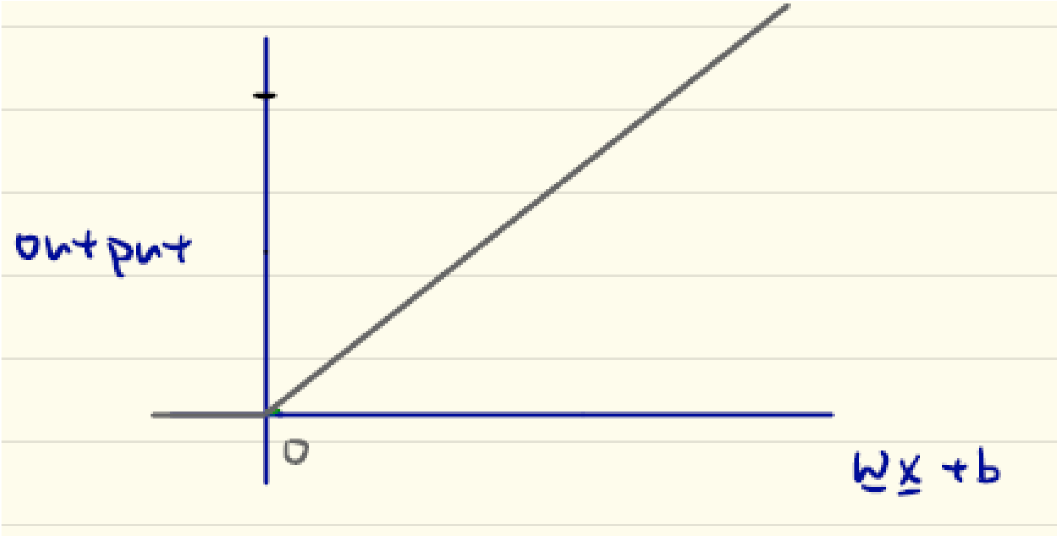
- $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Very similar to sigmoid, but its range is (-1,1)



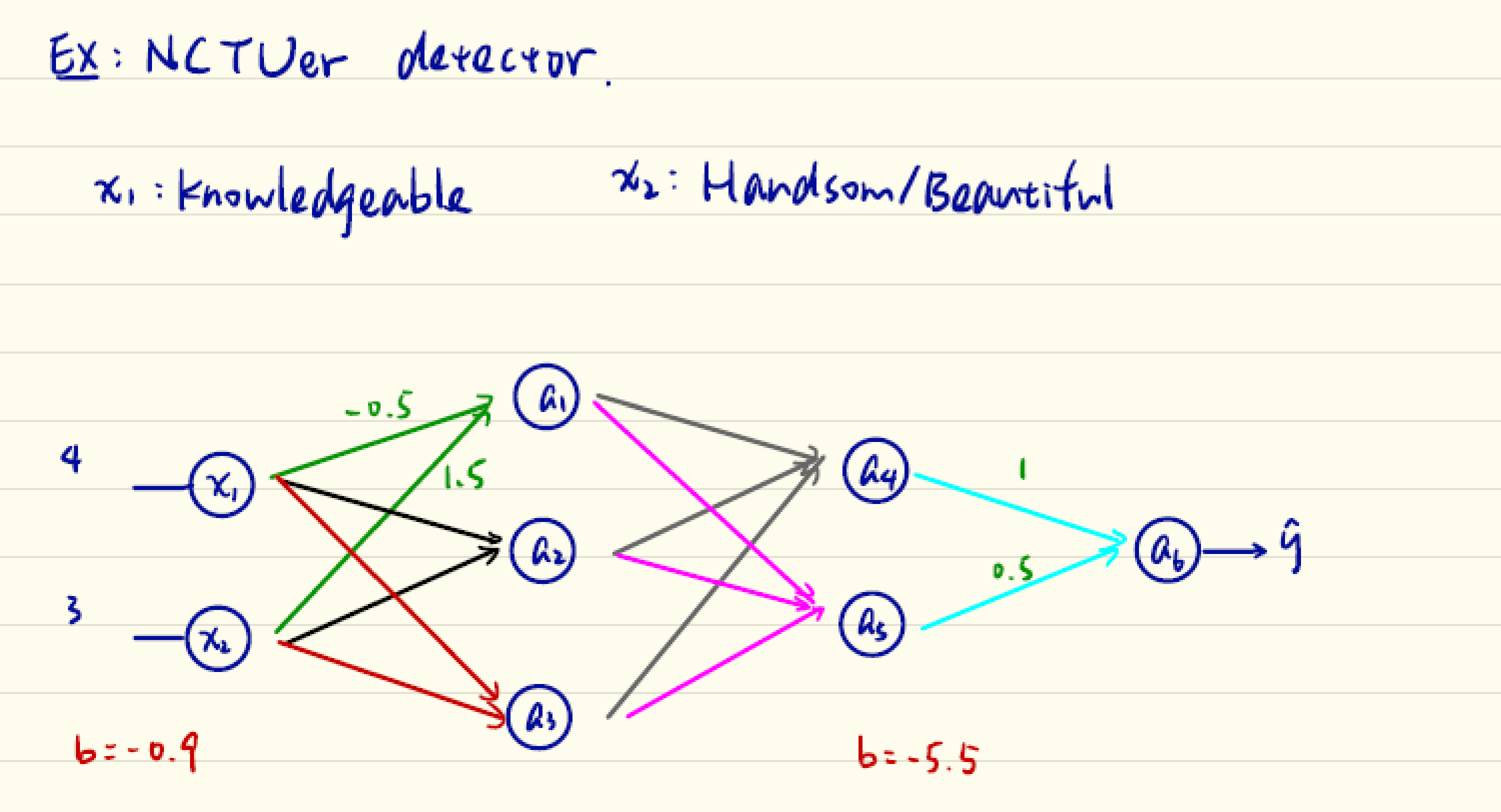
- Issue in sigmoid and Tanh: They saturate!

ReLu (Rectified Linear Units)

- $\sigma(z) = \max(0, z)$
- Super simple, do not saturate
- Most widely used for hidden layers



Example



EX: NCTUer detector.
x1 : Knowledgeable
x2 : Handsom/Beautiful
4
3
x1
x2
-0.5
1.5
a1
a2
a3
a4
a5
a6
1
0.5
ŷ
b=-0.9
b=-5.5

Example

Two dense hiden layers with ReLu neurons

$$z = \begin{bmatrix} -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} - 0.9 = 1.6$$

$$a_1 = \max(0, 1.6) = 1.6$$

suppose we have done calculation and obtained $a_4 = 2.5$ & $a_5 = 2$

$$a_6 = \begin{bmatrix} 1 & 0.5 \end{bmatrix} \begin{bmatrix} 2.5 \\ 2 \end{bmatrix} - 5.5 = -2$$

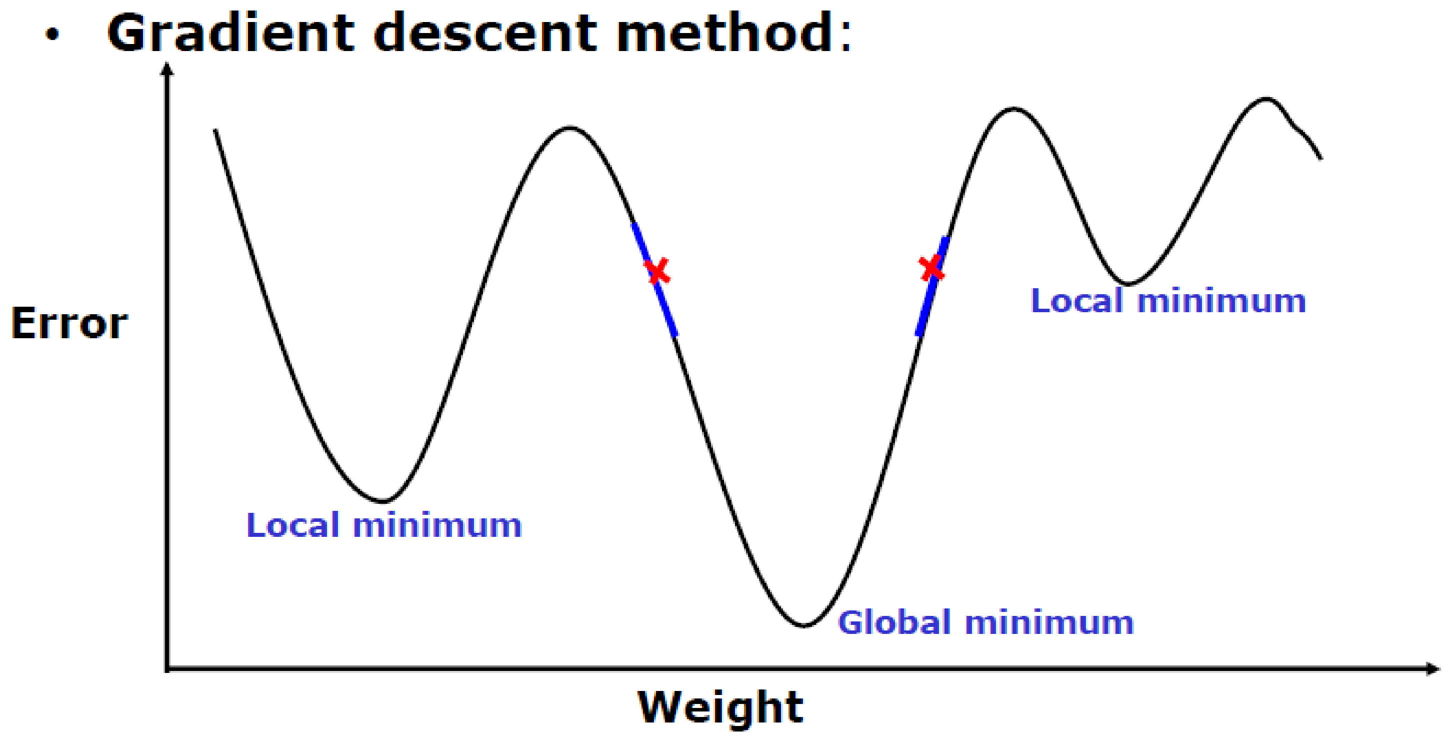
The output layer is sigmoid : $\hat{y} = \frac{1}{1 + e^{-2}} \approx 0.1192$

This indicates that the neural net estimates that there is an

11.92 % chance that this person is a NCTUer.

Parameter(Weights) Learning

- Gradient Descent Method, Stochastic Gradient Descent is often be used.



Implement

Procedure

- Build model
- Compile model
- Training/Fitting the model with training data
- Predict with test data
- Calculate the error

Build Model

- Define your model

  tf.keras.Sequential()

- Adding layers in your model

  model.add(...)

  - You can try different layers in library

    tf.keras.layers.experimental.preprocessing.
    Rescaling(scale=1,input_shape = (7,))
    tf.keras.layers.xxx

- Check your model

  model.summary()

Compile Model

- Given lose function, optimization way and the metrics you would like to observe.

    model.compile(loss=‘ xxx‘, optimizer=xxx , metrics=xxx)

    Optimizer=tf.keras.optimizer.Adam(learning_rate=0.01)

Training/Fitting the model with training data

- Given the training data, testing data and epochs. We start to train the model which we defined.
- We calculate the validation error at the same time, but it does nothing in our training progress.
- history = model.fit(train_y, train_m_OneHot, epochs=xx, validation_data=(test_y, test_m_OneHot), verbose=xx)

Observation & Demo

- Observe training results
  - plt.plot(history.history['accuracy'])
  - plt.plot(history.history['val_accuracy'])
  - plt.legend(['training', 'validation'], loc = 'upper left')
  - plt.show()

- Demo - BLER under $SNR = 0 \sim 7$