Recitation 1. Your First MLP

Sarthak Bisht, Yooni Choi

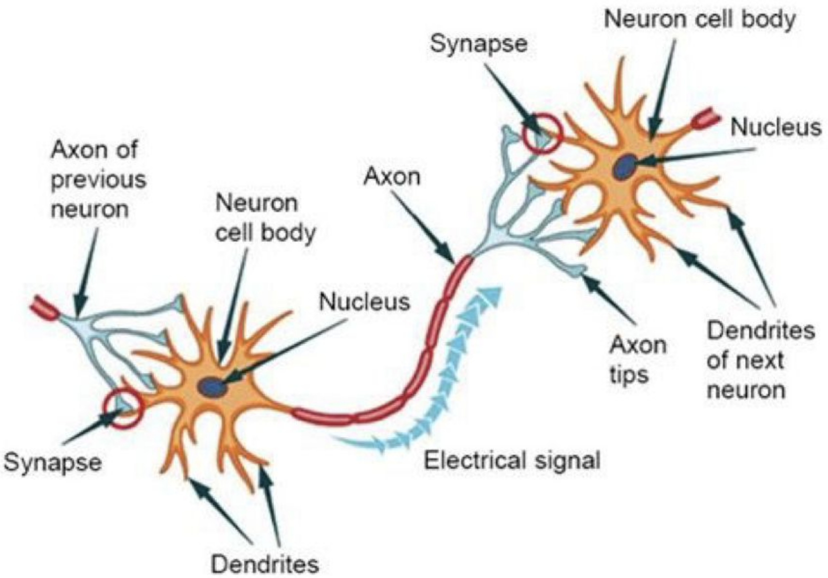
Carnegie Mellon University

Neural Networks

The brain, made up of connected neurons, are the inspirations for artificial neural networks

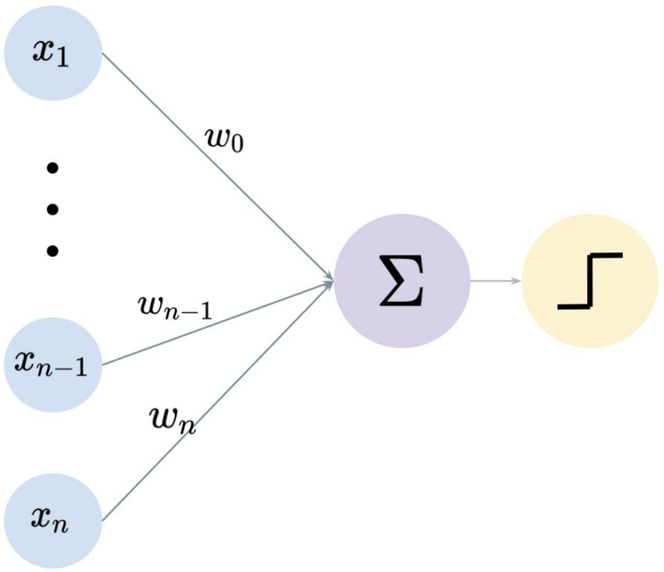
Neural Networks

- A neuron is a node with many inputs and one output
- A neural network consists of many interconnected neurons – a ‘simple’ device that receives data as the input and provides a response
- Information are transmitted from one neuron to another by electrical impulses and chemical signals



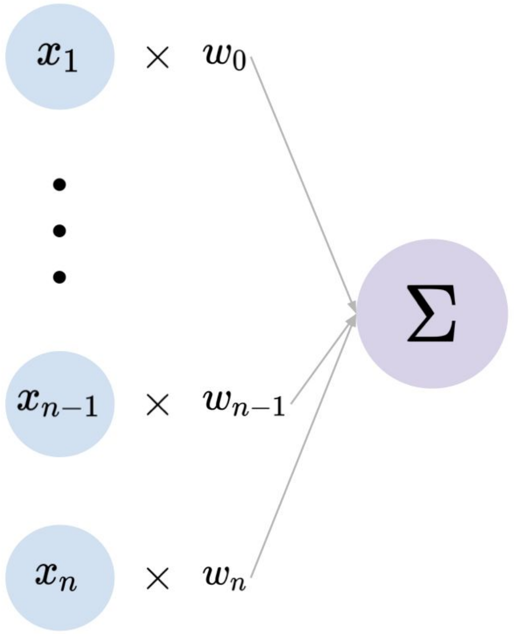
Perceptrons

- Perceptron is a single layer neural network
- The perceptron consists of 4 parts
  - Input values
  - Weights
  - Weighted sums
  - Threshold / Activation functions



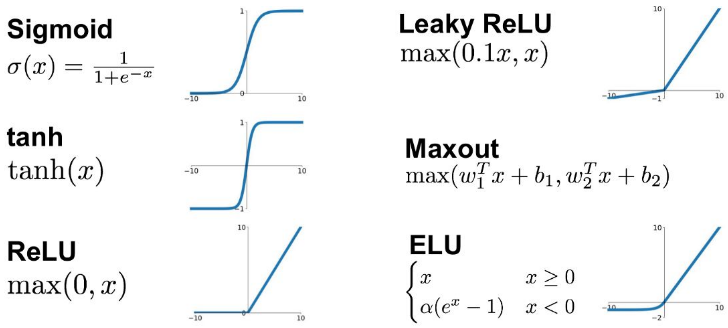
Perceptrons

- Perceptron is a single layer neural network
- The perceptron consists of 4 parts
- The perceptron works on the following steps:
  - Multiply all inputs with their weights
  - Add all multiplied values (weighted sum)



Perceptrons

- Perceptron is a single layer neural network
- The perceptron consists of 4 parts
- The perceptron works on the following steps:
  - Multiply all inputs with their weights
  - Add all multiplied values (weighted sum)
  - Apply the weighted sum to activation function



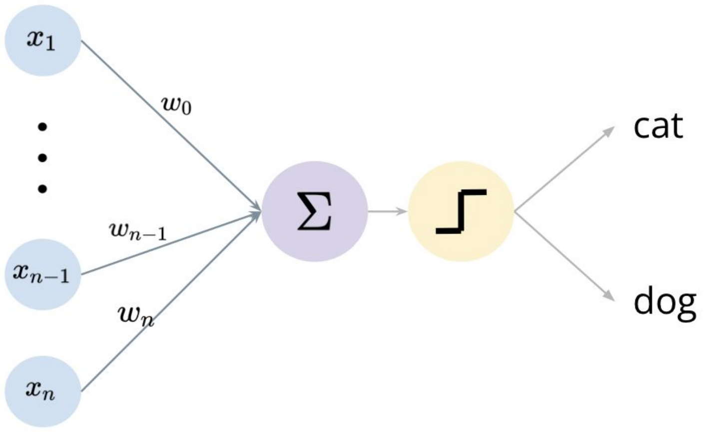
Carnegie Mellon University

Perceptrons

- Perceptron is usually used to classify the data into two parts

  (**Linear Binary Classifier**)

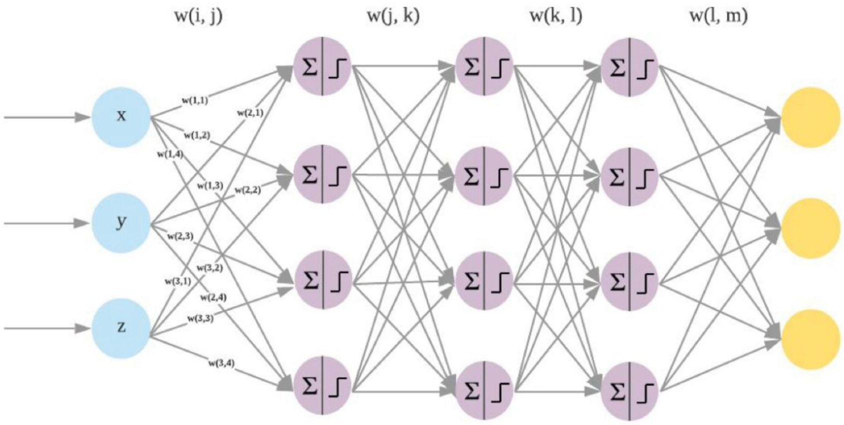
  - **Weights** shows the strength of the particular node
  - **Activation functions** are used to map the input between the required values

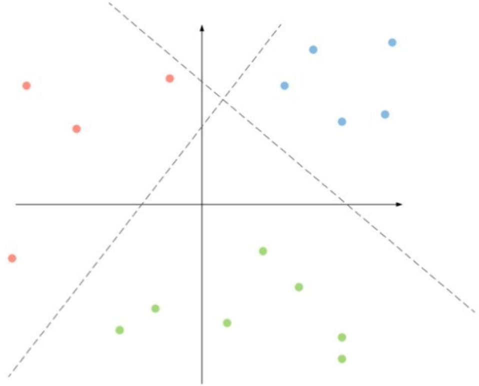


Multilayer Perceptrons

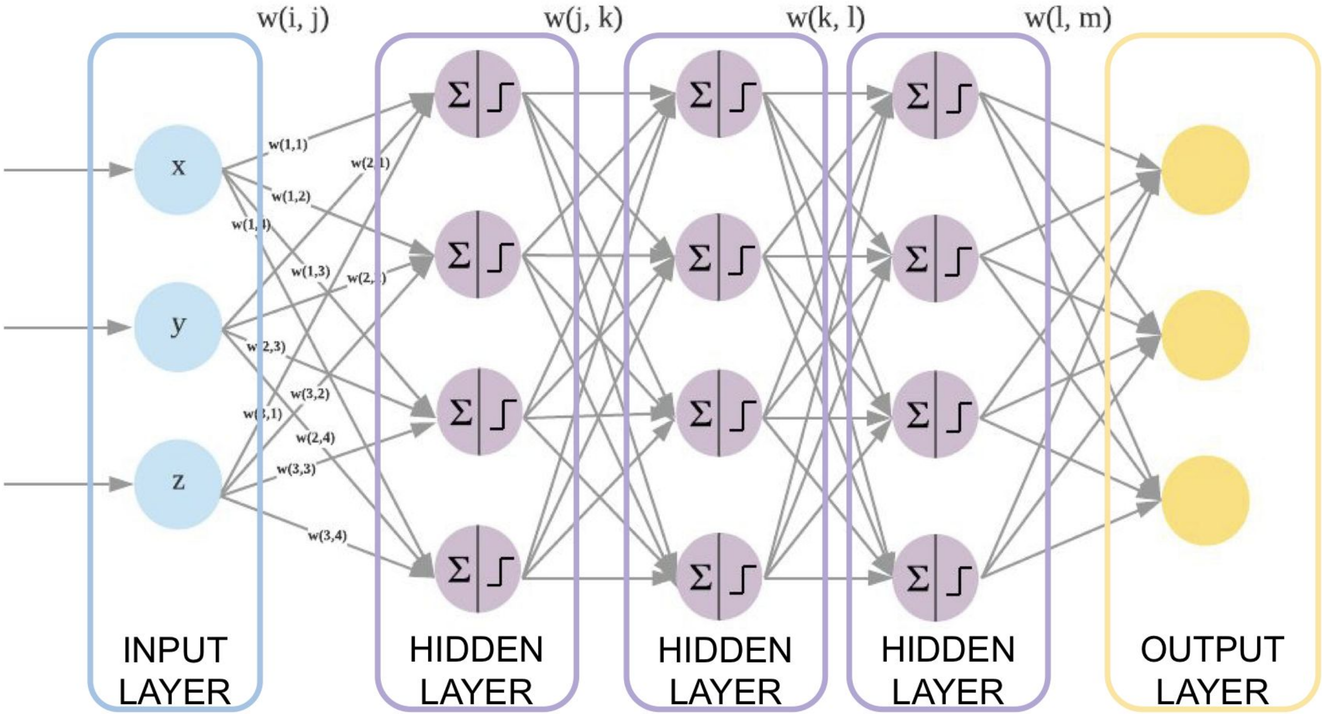
What if we want to be able to distinguish between more classes?

- Introduce more perceptrons!



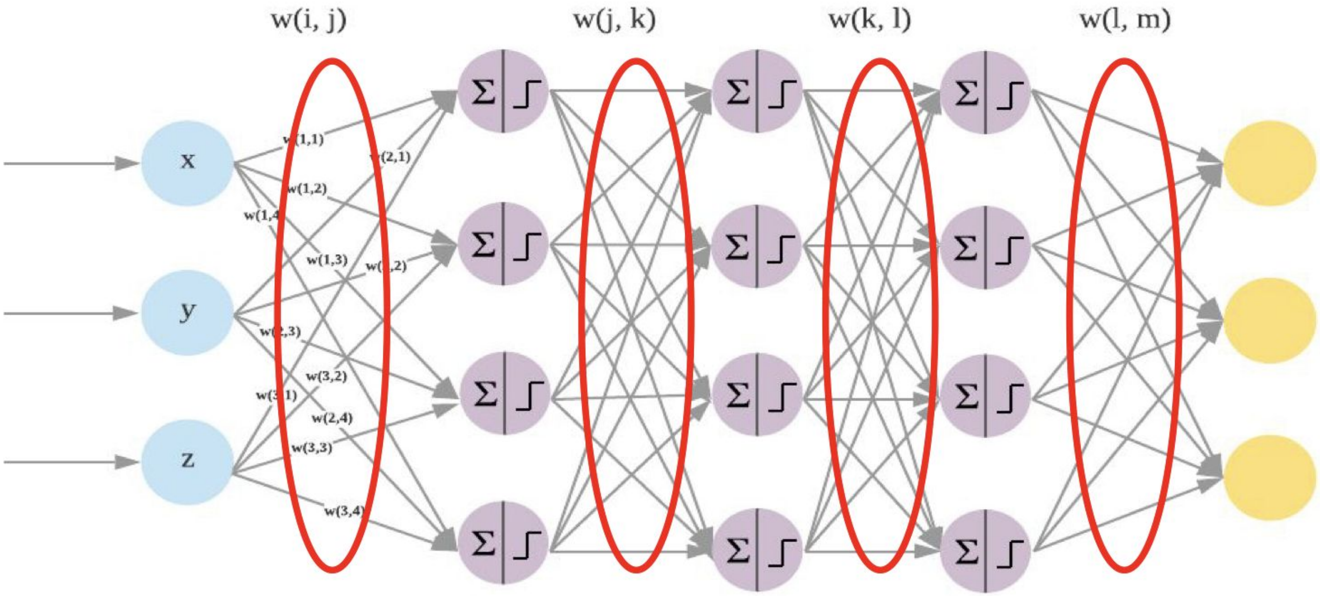


Multilayer Perceptrons



In order to correctly classify, the network must be **learned**

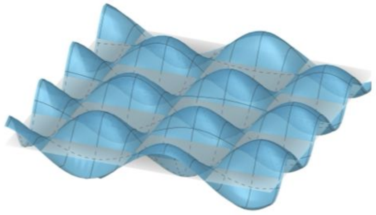
What do we need to learn?



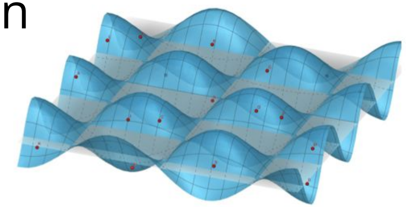
The parameters
(or the weights)

How do we learn?

- Actual function that we are trying to model:
  - Note: We don’t know the actual function



- We only have several sample data points on this function



- Our goal:
  - **Estimate the function with the given samples**

How do we learn?

- A measurement of **error**
  - How much off is the **network output** with respect to the **desired output**

$$Loss(W) = \frac{1}{N} \sum_i div(f(X_i, W), d_i)$$

For each sample
MLP
Network Output
Desired Output
Number of samples
Divergence function
Sample value
Current weights of estimated function

- Our goal (more specifically):
  - Minimize the loss

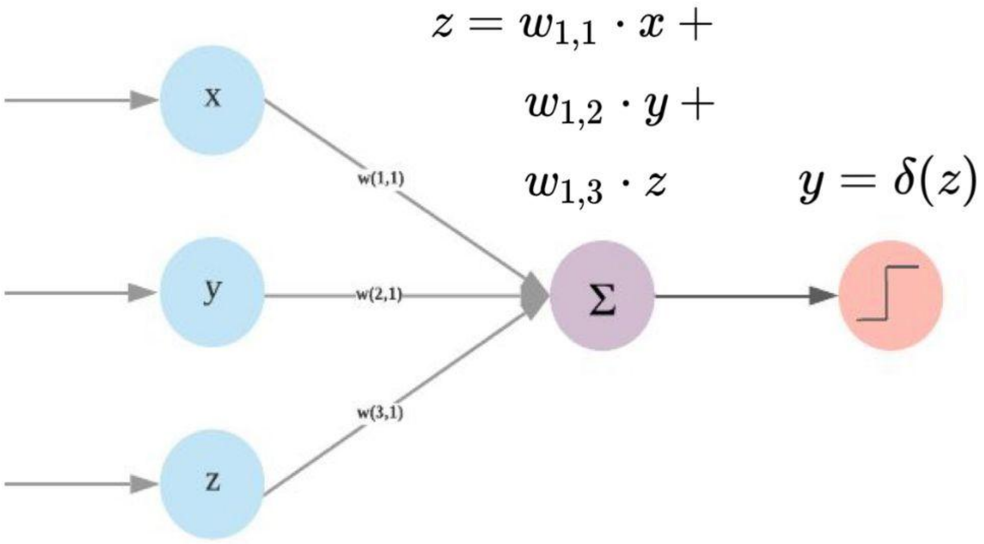
$$\hat{W} = \arg\min_W \; Loss(W)$$

How do we learn?

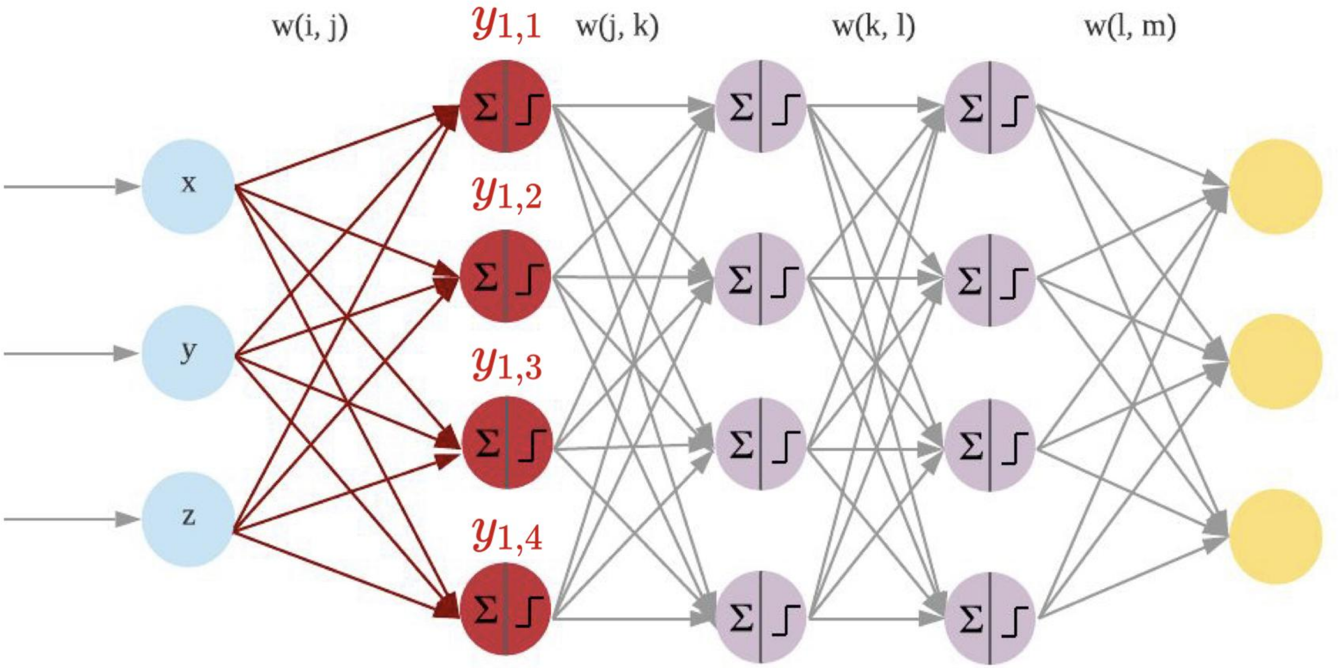
- Gradient Descent

Forward Pass

- For each single perceptron:

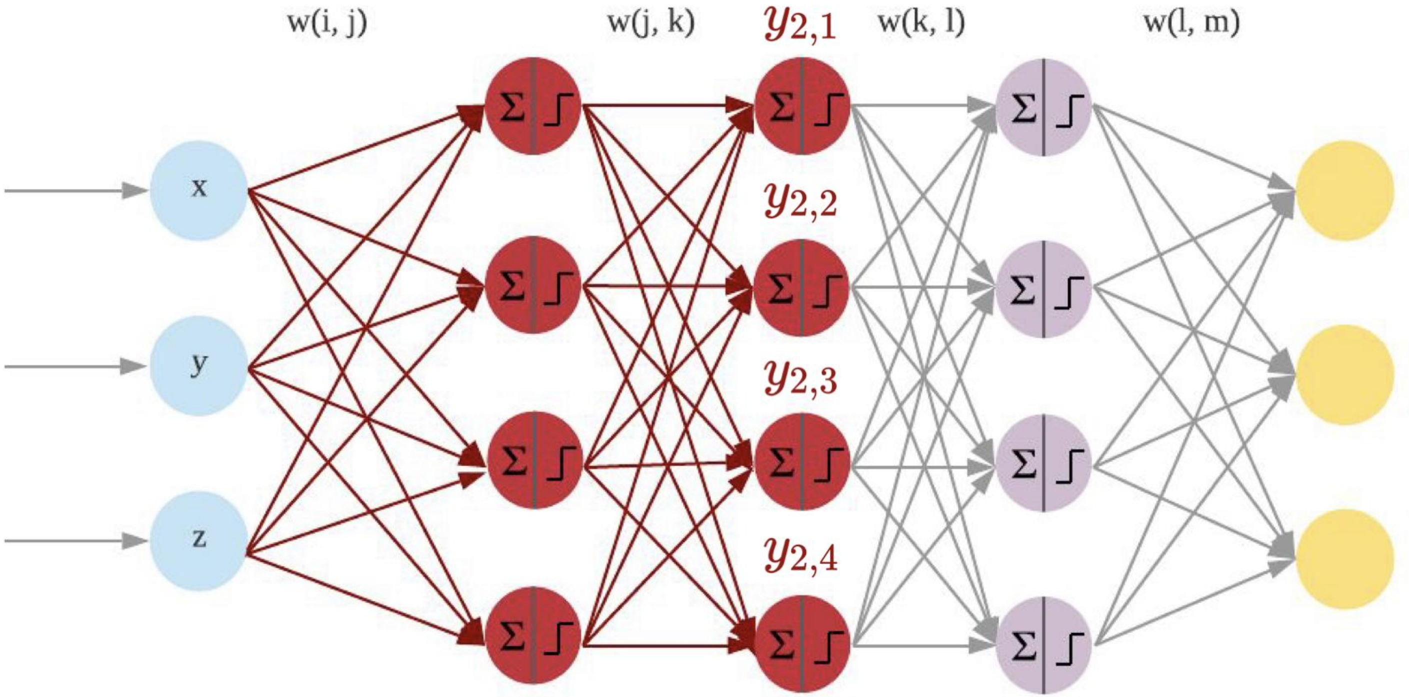


Forward Pass

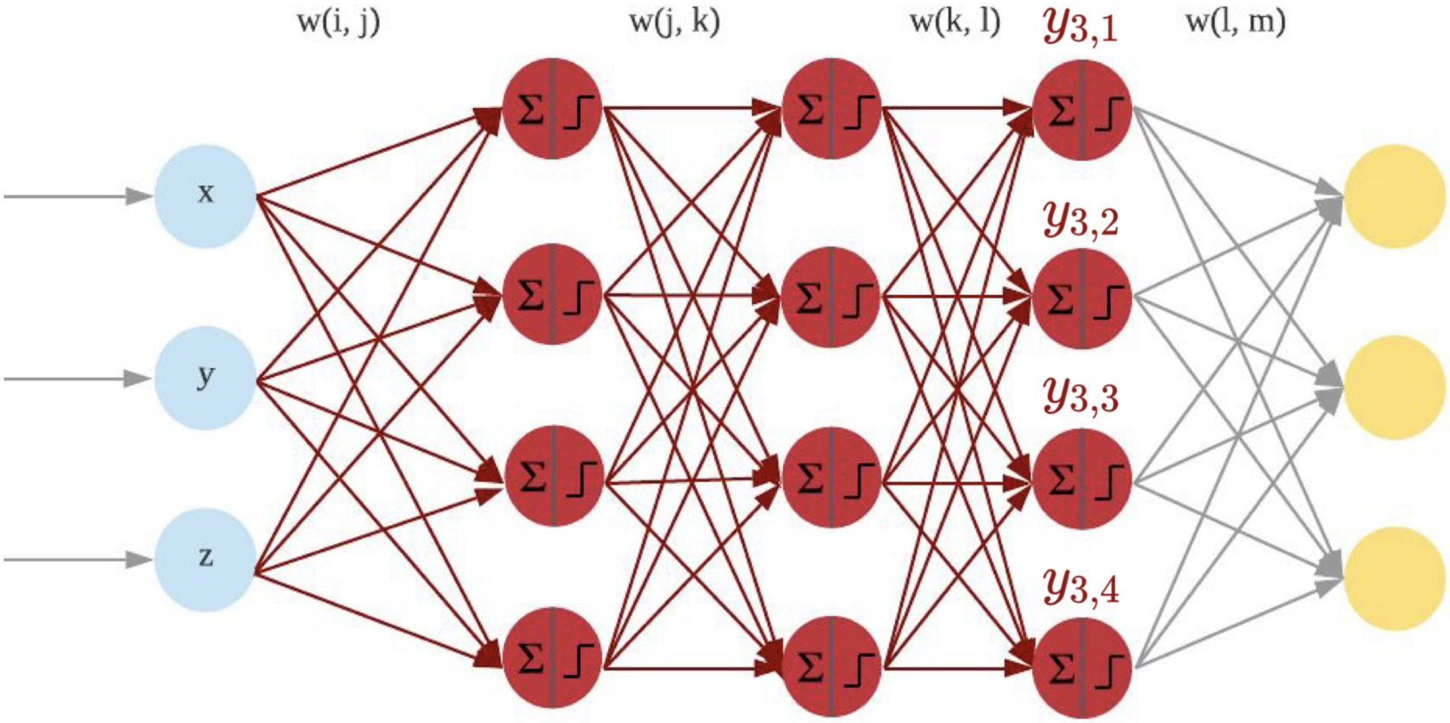


Carnegie Mellon University

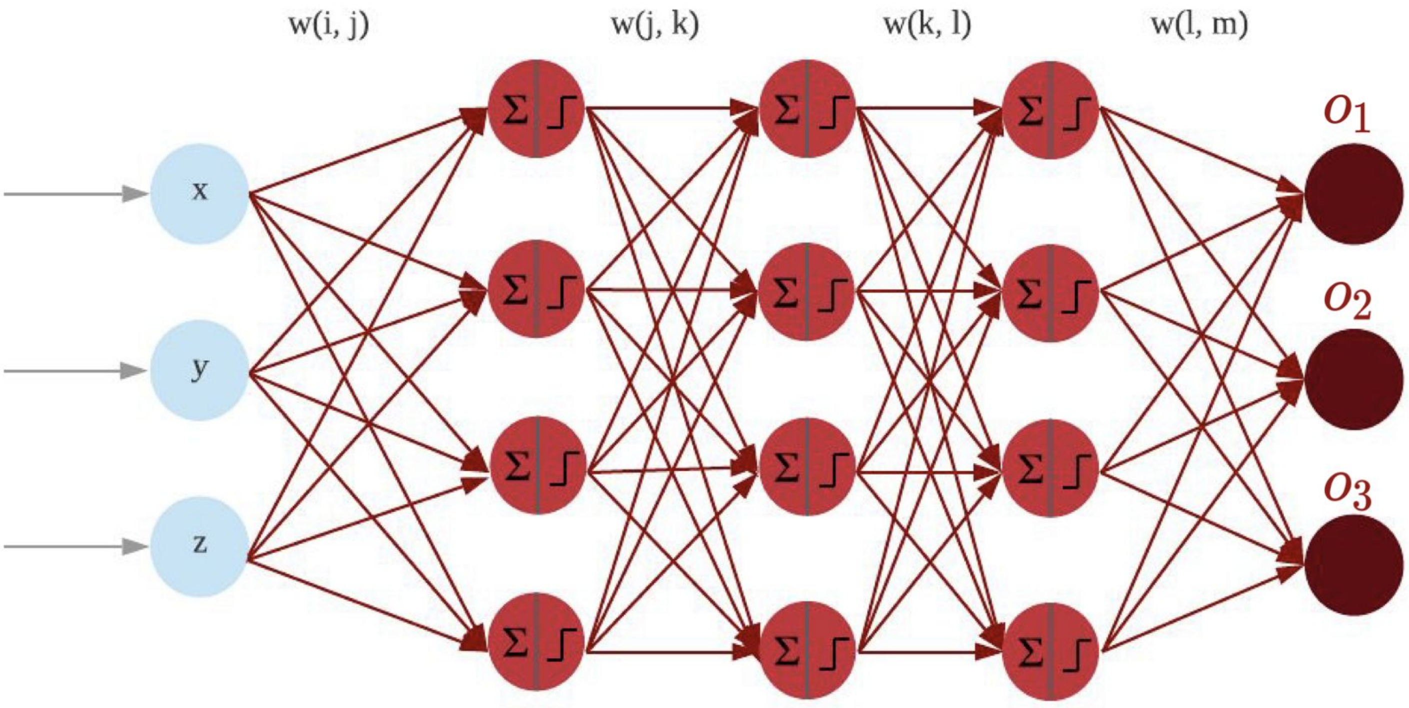
Forward Pass



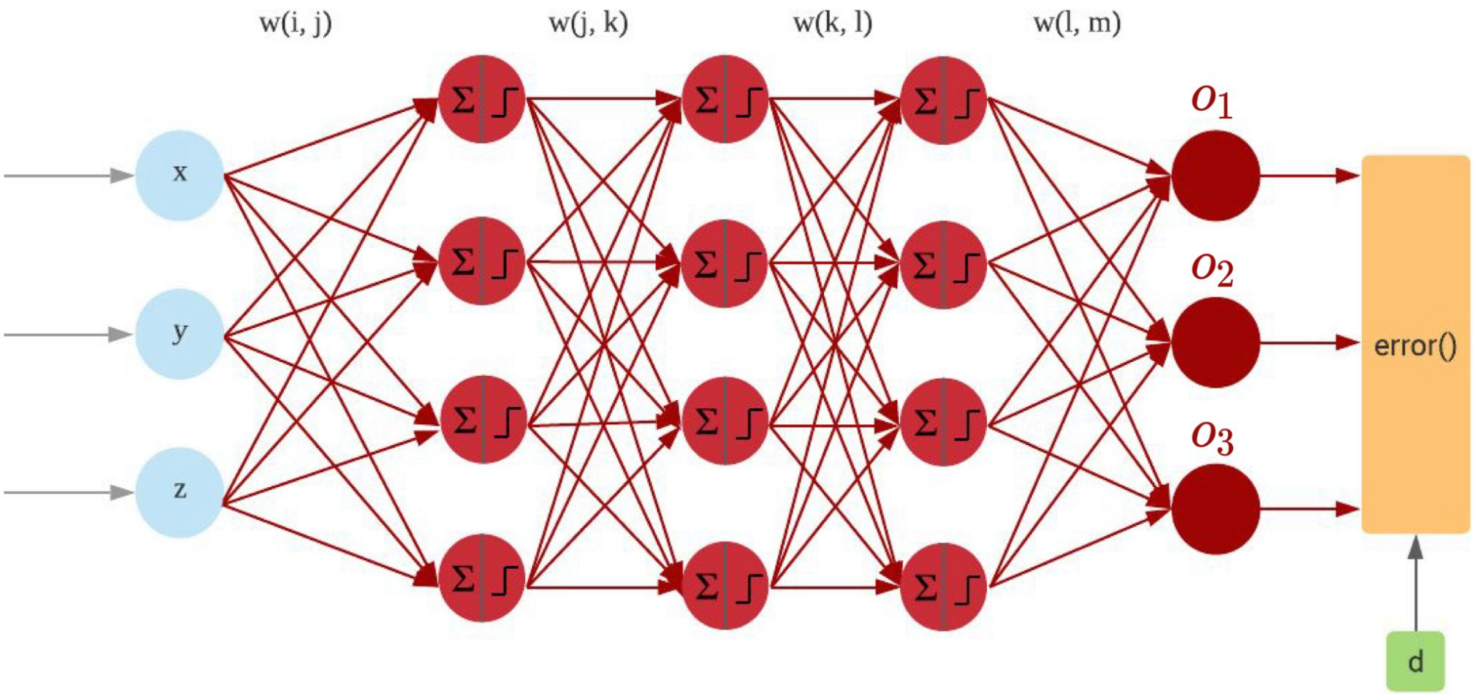
Forward Pass



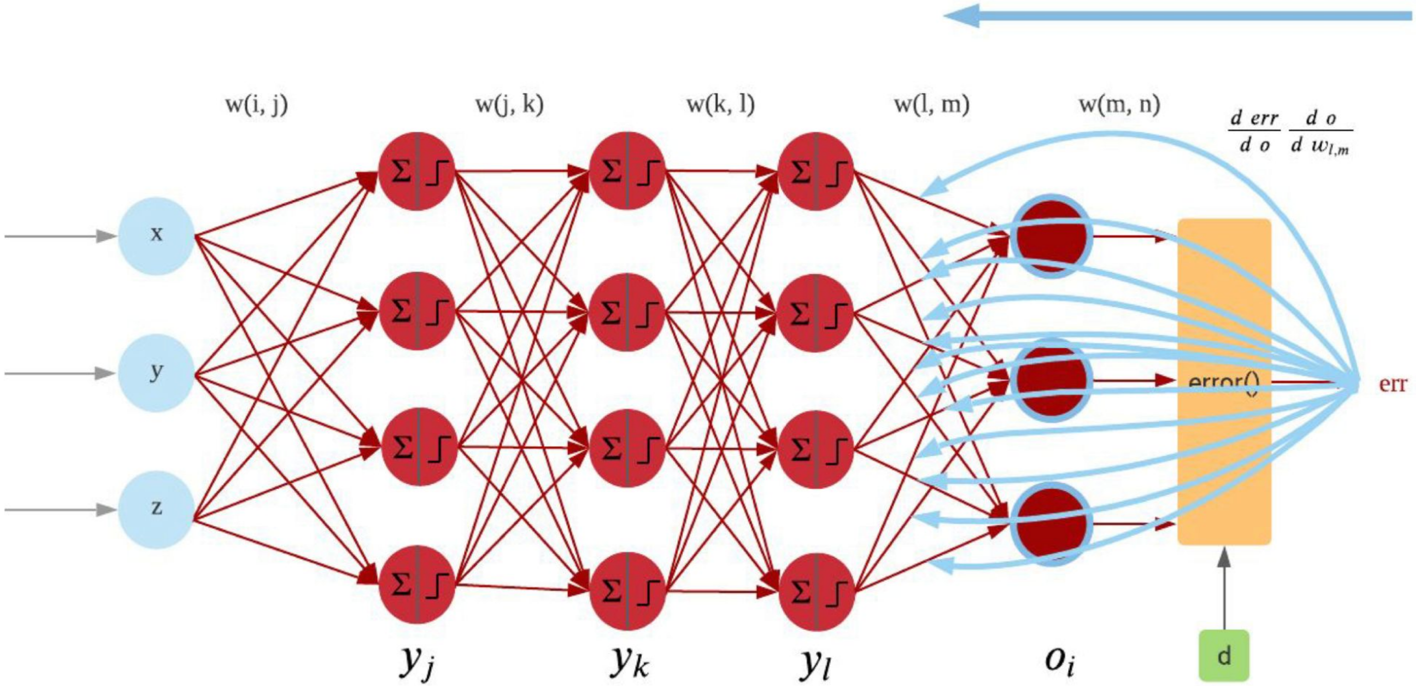
Forward Pass



Error



Backpropagation



All gradients of weights w.r.t error are calculated

Carnegie Mellon University

Colab Exercise: Using MLP - MNIST classification

Using MLP: MNIST classification

Stretching on the recitation 0L, we will explore MNIST classification using MLP in this notebook. Most of the contents are adopted from Recitation 0L notebook, but this recitation will focus more on the MLP model implementation part.

We're going to use the MNIST dataset which consists of handwritten digits 0-9 and use a neural network, specifically MLP, to classify them.

```
[ ] !pip install -q torchsummaryX
```

```
[ ] import torch
    import torchvision
    import matplotlib.pyplot as plt
    from torchsummaryX import summary
    import sklearn
    import sklearn.metrics
    from tqdm.auto import tqdm
    device = 'cuda' if torch.cuda.is_available() else 'cpu'
    print("Device: ", device)
```

```
Device:  cpu
```

https://colab.research.google.com/drive/1gSjoUsmPxRjH3bzEkkmCZYYhPG_M_rMp?authuser=1#scrollTo=DsNhXR25mCmq

Carnegie Mellon University

Deep Learning Pipeline

MNIST Dataset

Features (Image Pixels)	Labels (digits)
B x 1 x 28 x 28 = B x 784	{0, 1, 2, …, 9} = 10
⋮	⋮

Network Arch: MLP

Optimizer: Adam

Loss: Cross Entropy

Output: Arg Max

LibriSpeech Dataset

Features (MFCCs)	Labels (Phonemes)
N x T x 26 = B x 26	CMUdict = 40
⋮	⋮

Network Arch: MLP

Optimizer: Adam

Loss: Cross Entropy

Output: Arg Max

Speech Recognition