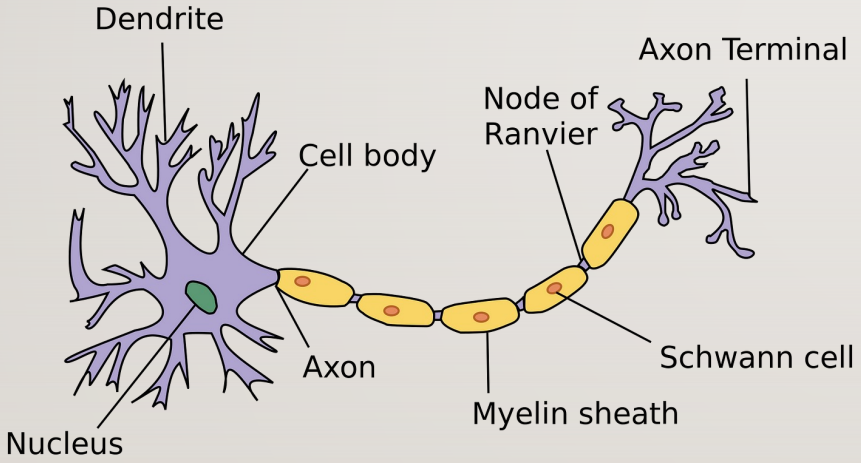
CCN MACHINE LEARNING WORKSHOP

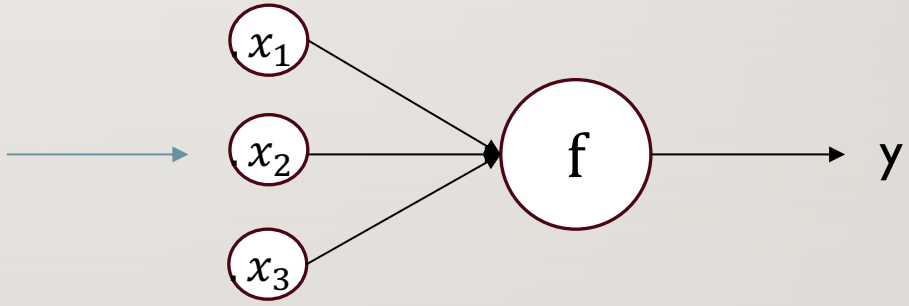
INTRODUCTION TO ANN AND CNN

Haiyan Wang

ANN

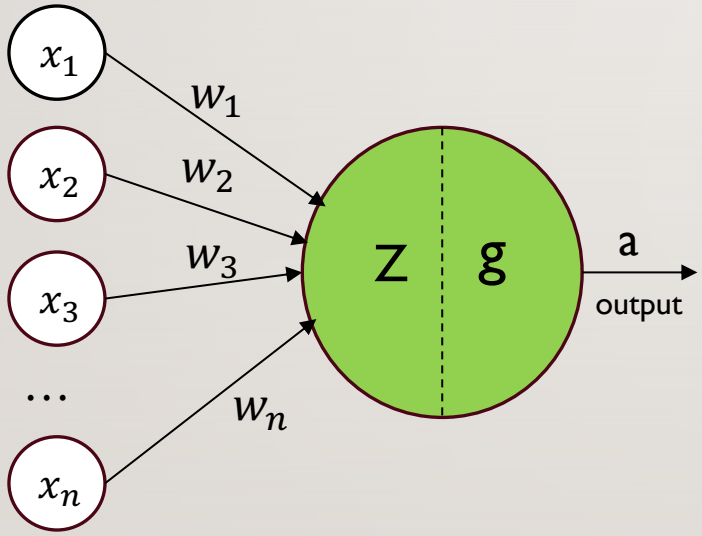
- Artificial Neural Network (ANN)
  - Use artificial neuron which loosely model the brain neuron. (ANN's history)





ANN

- What happens in the artificial neuron?



Inside of the neuron, there're two steps of calculation

1. **Linear calculation**:

   $z = w_1x_1 + w_2x_2 + w_3x_3 \ldots + w_nx_n + b$

   (w: weights b: bias)

2. **Non-Linear function g**:

   Activation function: a = g(z)

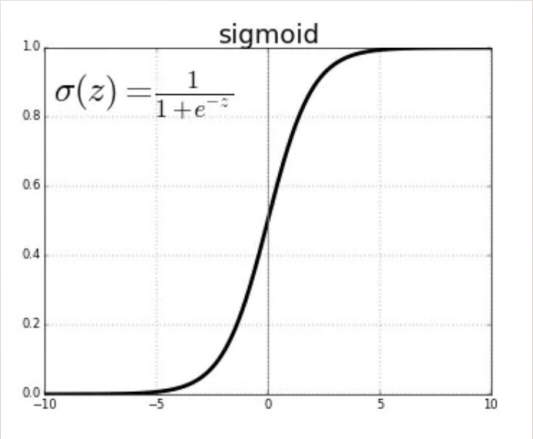
   Sigmoid, ReLu, Softmax, etc

ANN

- Activation function
  - Sigmoid
  - ReLu
  - Tanh
  - Leaky Relu
  - Softmax

ANN

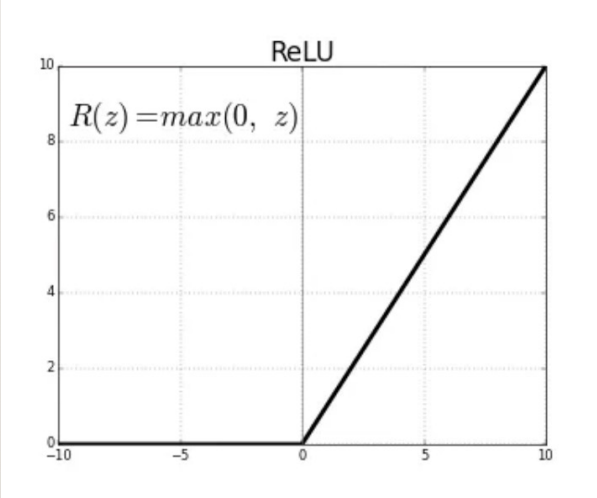
- Activation function:
  - Sigmoid function:
    - When the absolute value of z is very large, the result value is very close to 1 or 0
    - Output possibilities of an object belongs to a class
    - Normally used for output layer for binary classification



The image is from here

ANN

- Activation function:
  - Relu:
    - When z is larger or equal to 0, the result is z.
    - When z is smaller than 0, then the result is 0.
    - Normally used for hidden layers



The image is from here

ANN

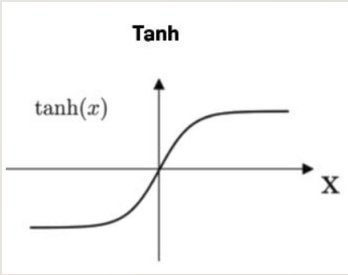
- Activation function
  - Tanh

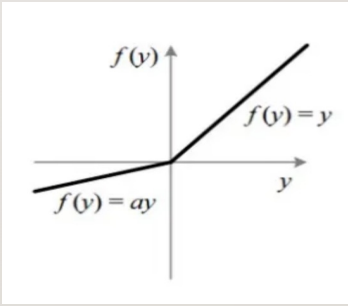
    tanh(z) = $\frac{e^z - e^{-z}}{e^z + e^{-z}}$ (-1,1)
  - Leaky Relu

    f(x) = $\begin{cases} x, when\ x > 0 \\ a * x, \quad x < 0 \end{cases}$
  - Softmax

    softmax(z_i) = $\frac{e^{z_i}}{\sum_1^k e^{z_k}}$ $(i = 1, \dots, k)$

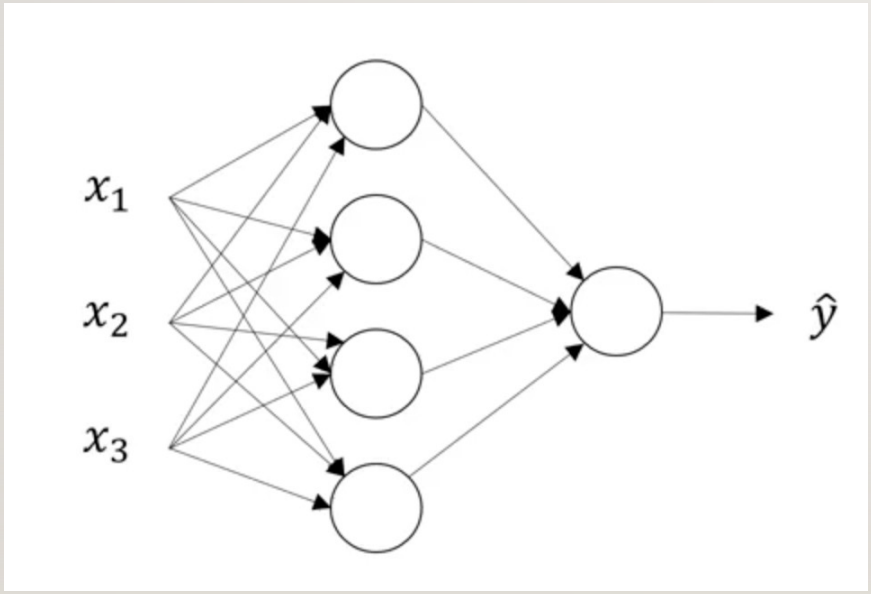
    (k is No. of the classes)





ANN

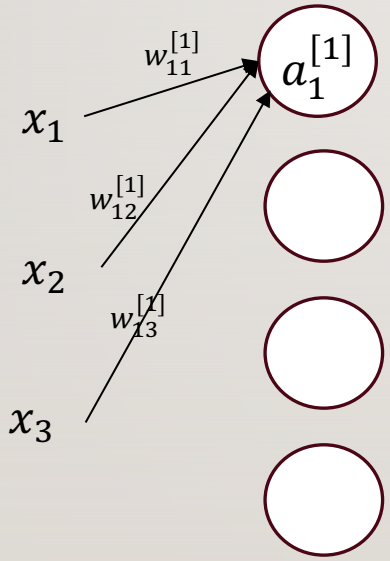
ANN With multiple layers:



Input layer: $\mathrm{X} = [x_1, x_2, x_3]$
Output layer: the last layer that generates y
Hidden layer: the rest middle layers

ANN

- ANN with multiple layers and nodes



First hidden layer:

$$z_1^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + w_{13}^{[1]} x_3 + b_1 \, , \, a_1^{[1]} = g(z_1)$$
$$z_2^{[1]} = w_{21}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{23}^{[1]} x_3 + b_2 \, , \, a_2^{[1]} = g(z_2)$$
$$z_3^{[1]} = w_{31}^{[1]} x_1 + w_{32}^{[1]} x_2 + w_{33}^{[1]} x_3 + b_3 \, , \, a_3^{[1]} = g(z_3)$$
$$z_4^{[1]} = w_{41}^{[1]} x_1 + w_{42}^{[1]} x_2 + w_{43}^{[1]} x_3 + b_4 \, , \, a_4^{[1]} = g(z_4)$$

$w_{11}^{[1]}$ *is the weight from* x_1 *to the first node* in the first hidden layer

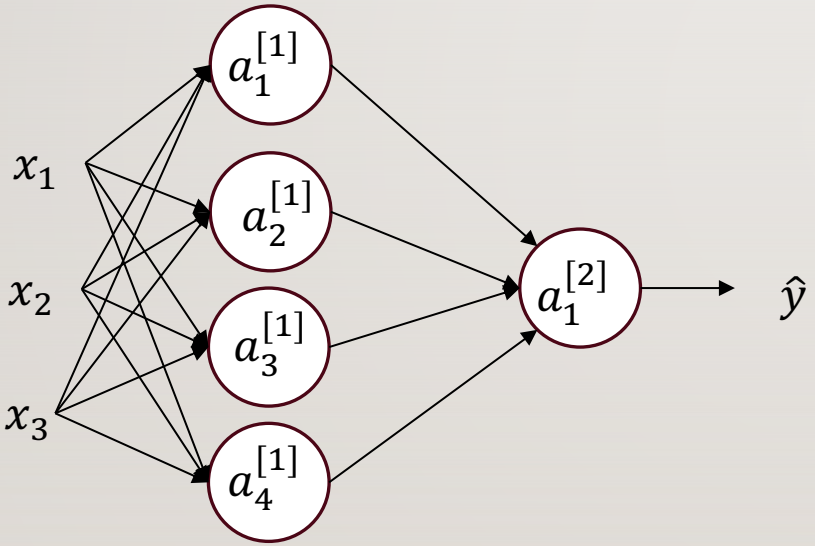
$w_{12}^{[1]}$ *is the weight from* x_2 *to the first node* in the first hidden layer

$w_{21}^{[1]}$ is the weight from x1 to the second node in the first hidden layer

b are the biases

ANN

ANN With multiple layers and nodes:



Output layer:

Layer2 weights: $[w_{11}^{[2]}, w_{12}^{[2]}, w_{13}^{[2]}, w_{14}^{[2]}]$

$$z_1^{[2]} = a_1^{[1]} w_{11}^{[2]} + a_2^{[1]} w_{12}^{[2]} + a_3^{[1]} w_{13}^{[2]} + a_4^{[1]} w_{14}^{[2]}$$

$$a_1^{[2]} = g(z_1^{[2]})$$

$$\hat{y} = a_1^{[2]}$$

COST FUNCTION

- Cost Function

  Cost function is used to find the best parameters to minimize $|\hat{y} - y|$

  - Regression

    MSE (Mean Squared Error, L2 Loss): $\frac{1}{m}\sum_1^m (y^i - \hat{y}^i)^2$ (m is the number of samples)

    MAE (Mean absolute Error, L1 Loss): $\frac{1}{m}\sum_1^m |y^i - \hat{y}^i|$

  - Categorical

    Binary Cross Entropy (Log Loss) $-\frac{1}{m}\sum_1^m (y^i \log \hat{y}^i + (1 - y^i)\log(1-\hat{y}^i))$

    Categorical Cross Entropy $-\frac{1}{m}\sum_1^m \sum_1^k y^i \log \hat{y}^i$ (k is the class No.)

FORWARD AND BACKWARD PROPAGATION

- **Forward propagation**

  Processing input data in each layer in order and get output data

  (x, W, b are vectors here, ex. x = $[x_1, x_2, x_3]$, b = $[b_1, b_2, b_3]$)

  $z^{[1]} = W^{[1]}x + b^{[1]} \rightarrow a^{[1]} = g^{[1]}(z^{[1]}) \rightarrow$

  $z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \rightarrow a^{[2]} = g^{[2]}(z^{[2]}) \rightarrow \hat{y}$ (ex. Only 1 hidden layer here but can be many layers)

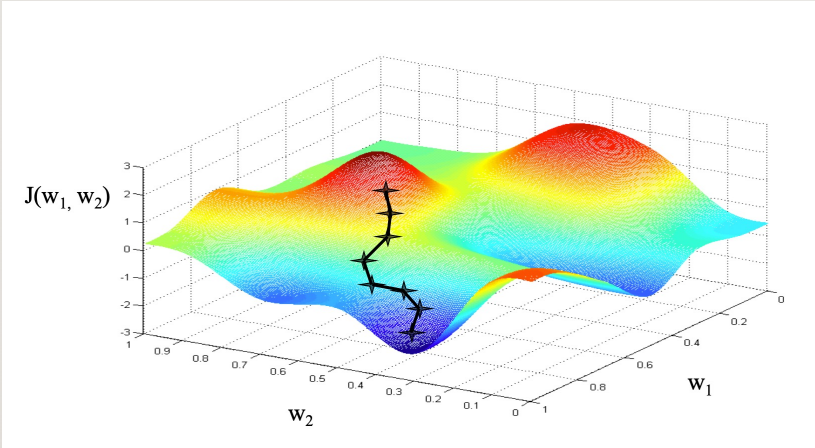
- **Backward propagation**

Use loss function $J(a^{[2]}, y)$ to calculate backward to find out the best parameters that can minimize the prediction error, using partial derivative.

$$\frac{\partial J}{\partial w^{[1]}}, \frac{\partial J}{\partial b^{[1]}} \leftarrow \frac{\partial J}{\partial z^{[1]}} \leftarrow \frac{\partial J}{\partial a^{[1]}} \leftarrow \frac{\partial J}{\partial w^{[2]}}, \frac{\partial J}{\partial b^{[2]}} \leftarrow \frac{\partial J}{\partial z^{[2]}} \leftarrow \frac{\partial J}{\partial a^{[2]}}$$

GRADIENT DECENT

- Gradient Decent (with two parameters)



From Andrew Ng, Machine Learning

- Optimizer

  By adjusting the model's parameters (weights and biases) during gradient decent so it can reach the goal faster

ANN DEMO

- Data: Brain Tumor images (2D data)

  In our coding demo, the 2D data are flattened into 1D as input

- Use Tensorflow

  Tensorflow is a type of Machine Learning framework

  ```
  import tensorflow as tf
  tf.keras.models.Sequential()
  tf.keras.layers.Dense(num_nodes, activation_function)
  ```

CNN

- Convolutional Neural Network
  - How to process images better?

    Edge detection (Ex. Sobel filter)

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

CNN

- Convolutional Neural Network
  - Sobel filter

(original, x-direction
y-direction, xy-direction)

CNN

- Convolutional Neural Network
  - Convolution layer
  - Pooling layer
  - Fully connected layer

CNN

- Convolutional Neural Network
  - Convolution layer (Animation)

    Apply filter to the image matrix, (the filter values are parameters)

    Ex: $30x(-1)+0x0+0x1+30x(-1)+0x0+0x1+30x(-1)+30x0+0x1 = -70$

30	0	0	0
30	0	0	0
10	0	0	0
10	0	0	0

*

-1	0	1
-1	0	1
-1	0	1

=

-70	0

CNN

- Convolutional Neural Network
  - Convolution layer (Animation)

    Apply filter to the image matrix,

    Ex: $0x(-1)+0x0+0x1+0x0+0x(-1)+0x0+0x1+0x0+ 0x(-1)+0x0+0x1+0x0 = 0$

30	0	0	0
30	0	0	0
30	0	0	0
30	0	0	0

*

-1	0	1
-1	0	1
-1	0	1

=

-70	0

CNN

- Convolutional Neural Network
  - Convolution layer (Animation)

    Apply filter to the image matrix, Ex:

    Ex. $30x(-1)+0x0+0x1+10x(-1)+0x0+0x1+10x(-1)+30x0+0x1 = -50$

30	0	0	0
30	0	0	0
10	0	0	0
10	0	0	0

*

-1	0	1
-1	0	1
-1	0	1

=

-70	0
-50	

CNN

- Convolutional Neural Network
  - Convolution layer (Animation)

    Apply filter to the image matrix,

    Ex: $0x(-1)+0x0+0x1+0x0+0x(-1)+0x0+0x1+0x0+\ 0x(-1)+0x0+0x1+0x0 = 0$

30	0	0	0
30	0	0	0
30	0	0	0
30	0	0	0

*

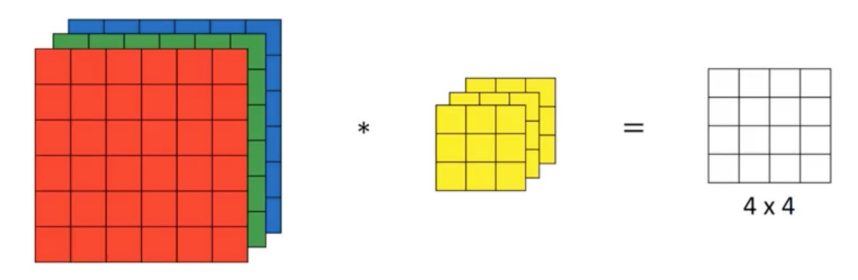
-1	0	1
-1	0	1
-1	0	1

=

-70	0
-50	0

CNN

- Convolutional Neural Network
  - Convolution layer (Animation)



- Kernel (or filter) : ex. 3x3, 5x5, 7x7 matrix
- Padding: add 0s to the edges of image
- Stride: step size to move to the right and down
- No. of filters

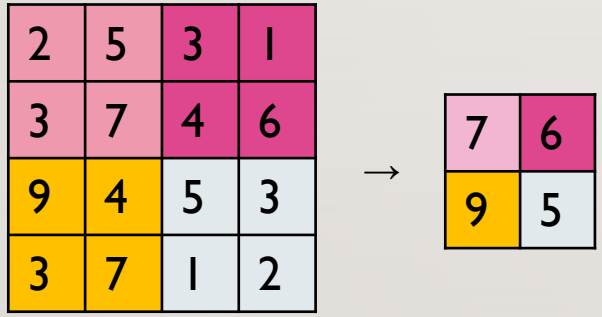
0	0	0	0	0
0	3	7	2	0
0	1	2	9	0
0	6	3	8	0
0	0	0	0	0

Padding

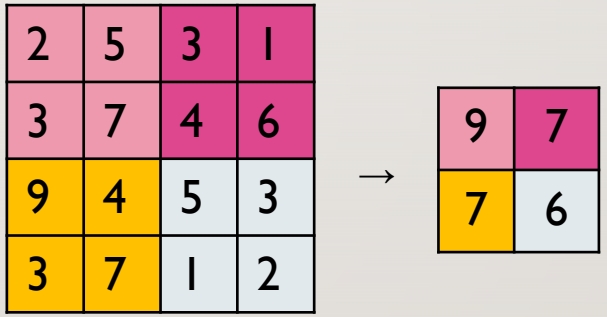
CNN

- Convolutional Neural Network
  - Convolution layer
  - Pooling layer: Max Pooling, Average Pooling

Max Pooling



Ave Pooling



  - Fully connected layer

CNN

- Why does CNN works well?

Andrew Ng, Deep Learning

- Use TensorFlow to build CNN

In our coding demo, we use CNN on the same data to detect brain tumor

CNN

- Future workshop

  Use 3D, 4D MRI data in Neural Network

  Welcome any idea, suggestions for next workshop!

- Any questions?

haiyanwang@mednet.ucla.edu