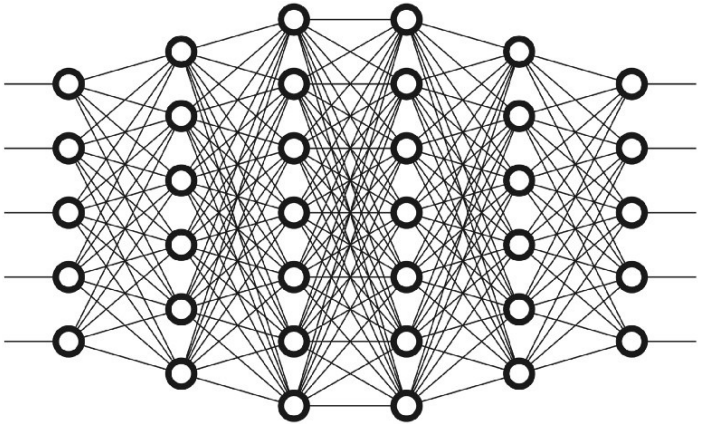
Argonne NATIONAL LABORATORY

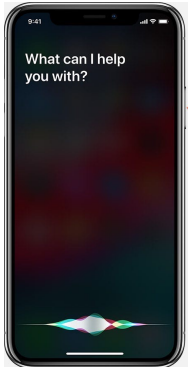
DEEP LEARNING METHODS
(TALK/HANDS-ON)



TANWI MALLICK
Assistant Computer Science Specialist
Mathematics and Computer Science Division
Argonne National Laboratory

12 August 2022

VOICE ENABLED PERSONAL ASSISTANT



"Hey Siri"

"Hey Cortana"

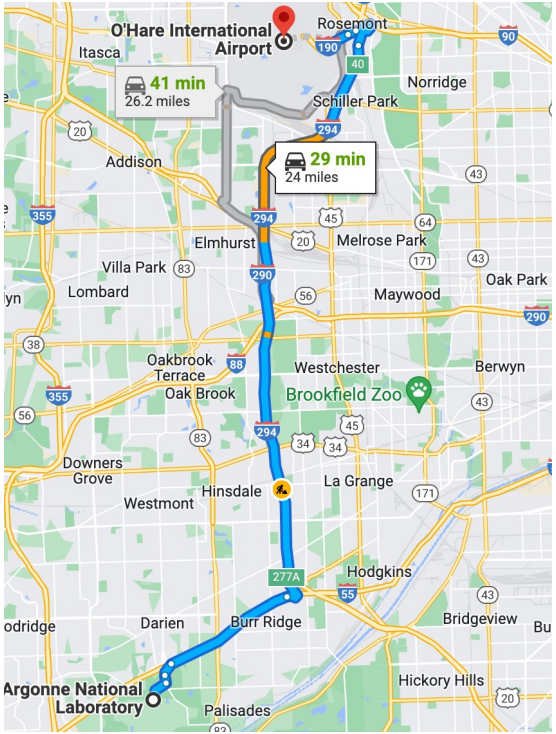
"Alexa"

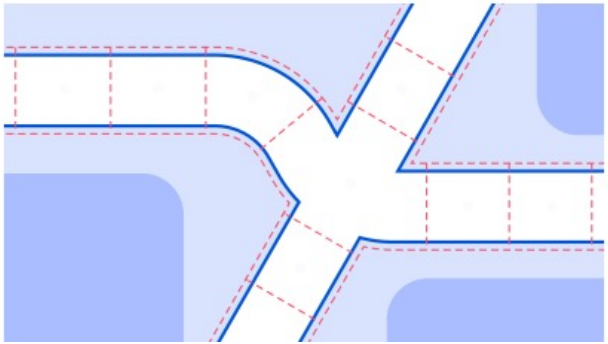
"OK Google"

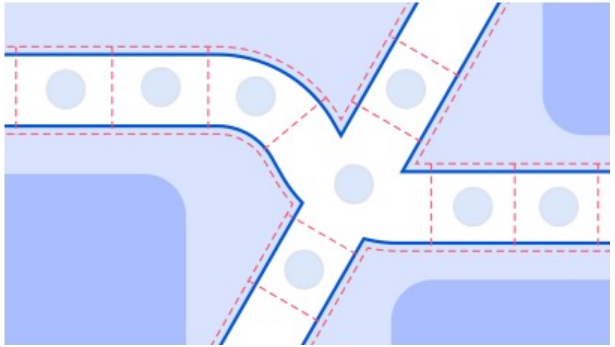
VOICE ENABLED PERSONAL ASSISTANT

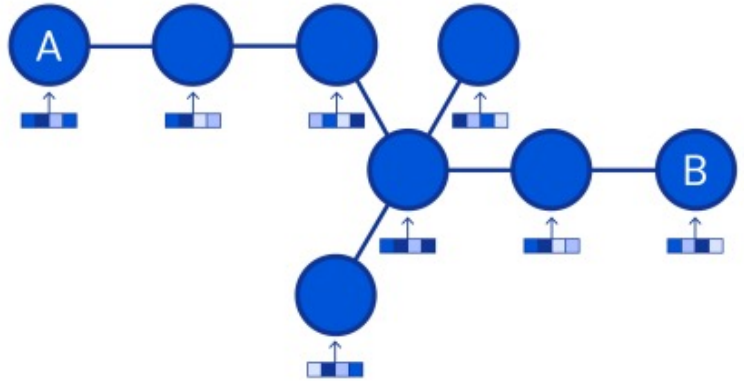
Source: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning

TRAFFIC PREDICTION



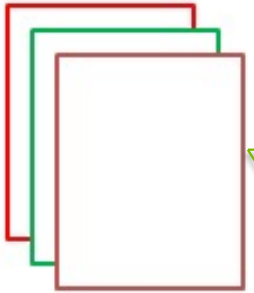






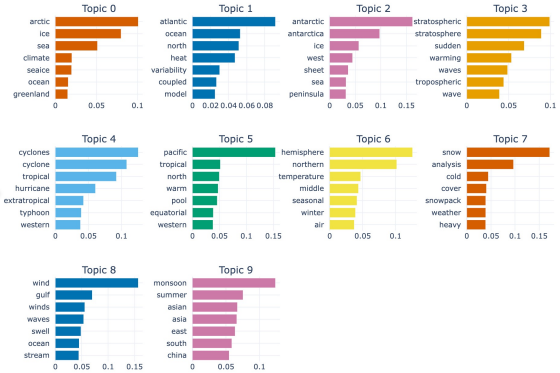
https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks
https://arxiv.org/pdf/2108.11482.pdf

NLP FOR CLIMATE RESEARCH



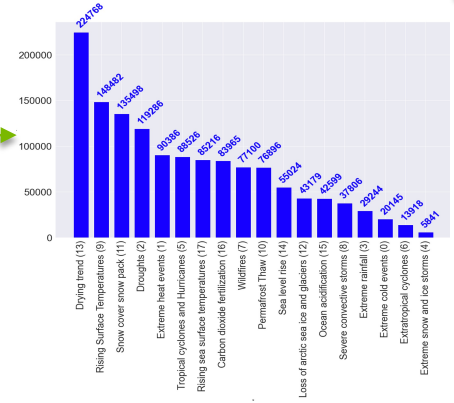
Collection of research articles

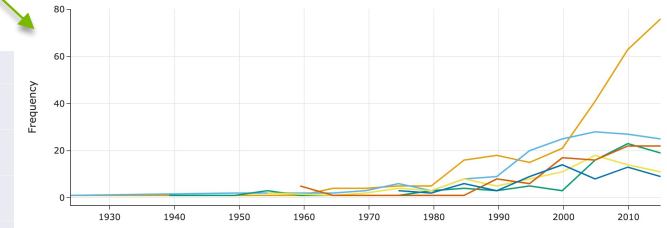
NLP based topic modeling



Distribution of words in a topics

Categorization using Weak Supervision

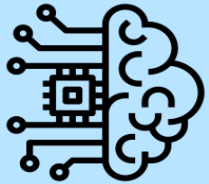




Trending topic overtime

WHAT IS DEEP LEARNING

Artificial Intelligence



Any technique that enable computer to mimic human behavior

Machine Learning

Ability to learn without explicitly programmed

Deep Learning

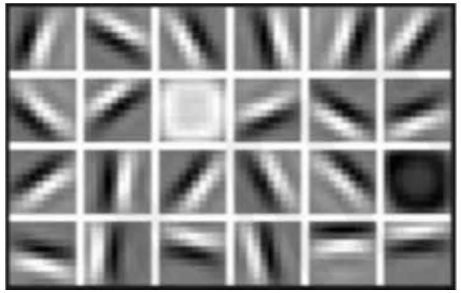
Extract pattern from directly from data

WHY DEEP LEARNING AND WHY NOW?

Hand engineered features are time consuming, brittle and not scalable in practice

Can we learn underlying feature directly from the data?

Low level features



Lines and edges

Mid level features

Eyes, nose, and ears

High level features

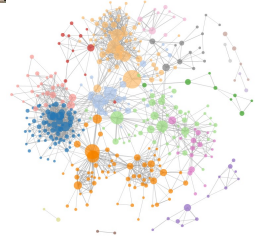
Facial structure

WHY DEEP LEARNING AND WHY NOW?

Neural networks date back decades, so why the resurgences?

1. Hardware
   - Graphics processing units (GPUs)
   - Massively parallelizable

2. Big Data
   - Large dataset
   - Easier collection and storage



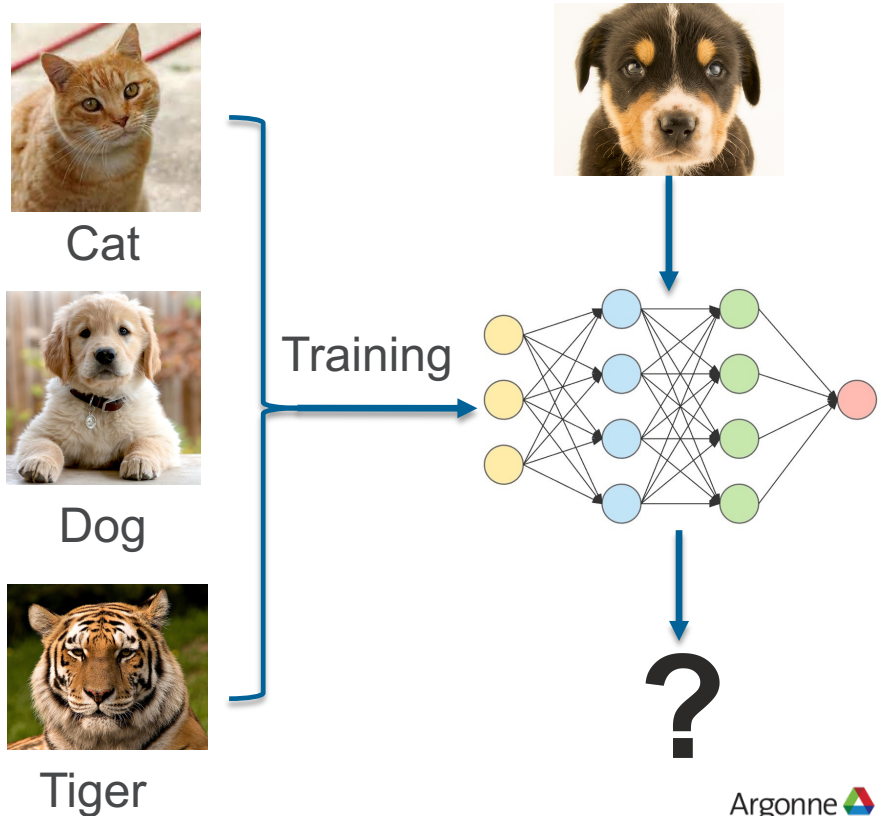
3. Software
   - New models
   - Easley usable packages

TensorFlow

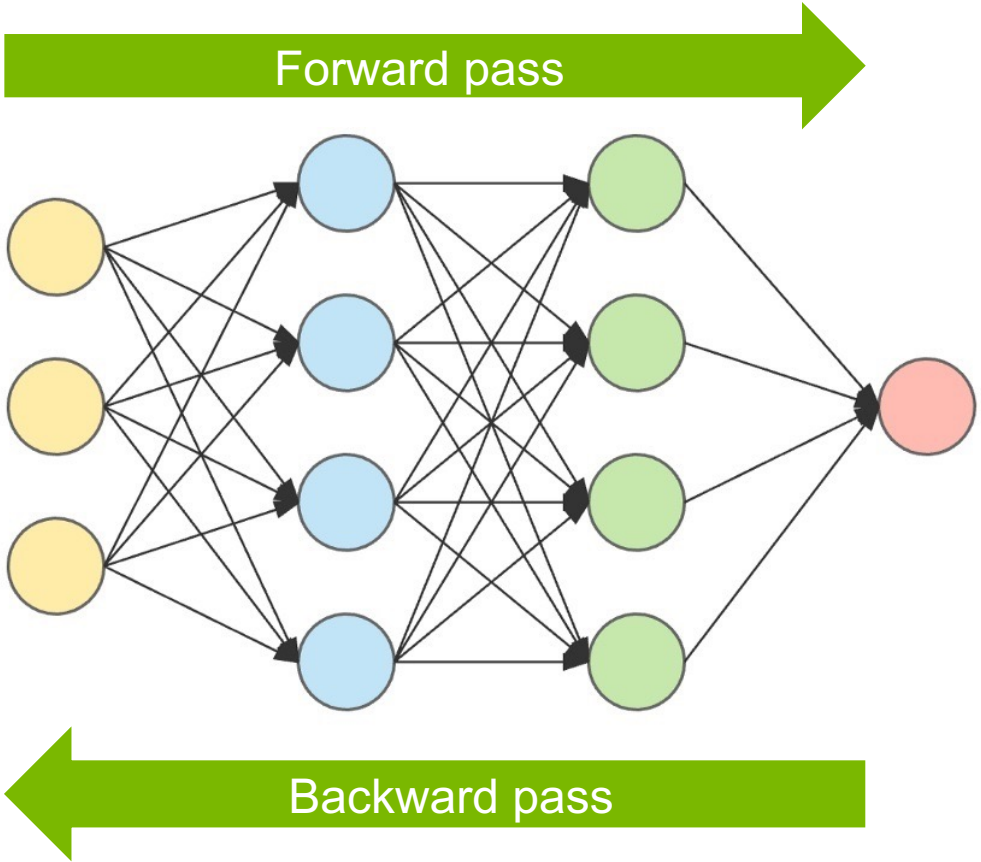
PyTorch

CATEGORIES OF LEARNING PROBLEMS OR PARADIGMS

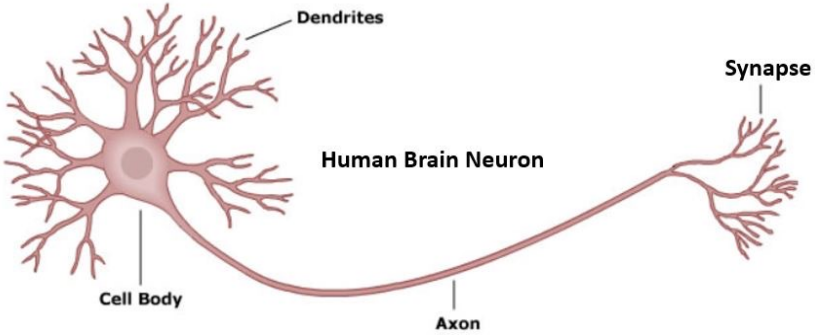
- Supervised learning (this talk)
  - Regression: output variable is continuous
  - Classification: output variable is discrete (categorical)
- Unsupervised learning
  - Clustering
  - Association
- Semi-supervised learning
- Reinforcement learning

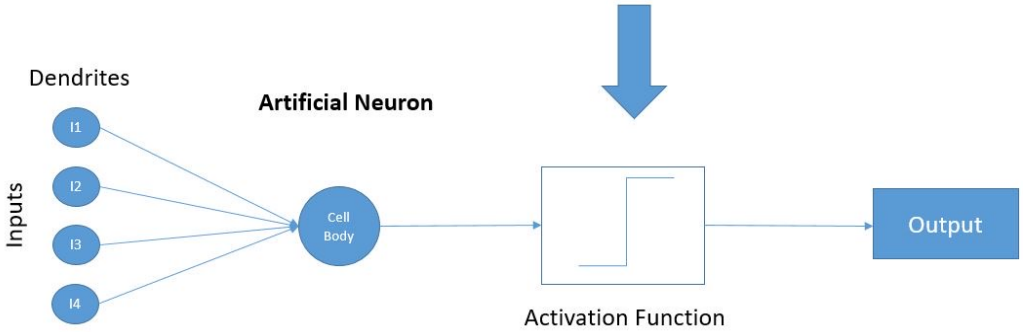


HOW DOES IT WORK?

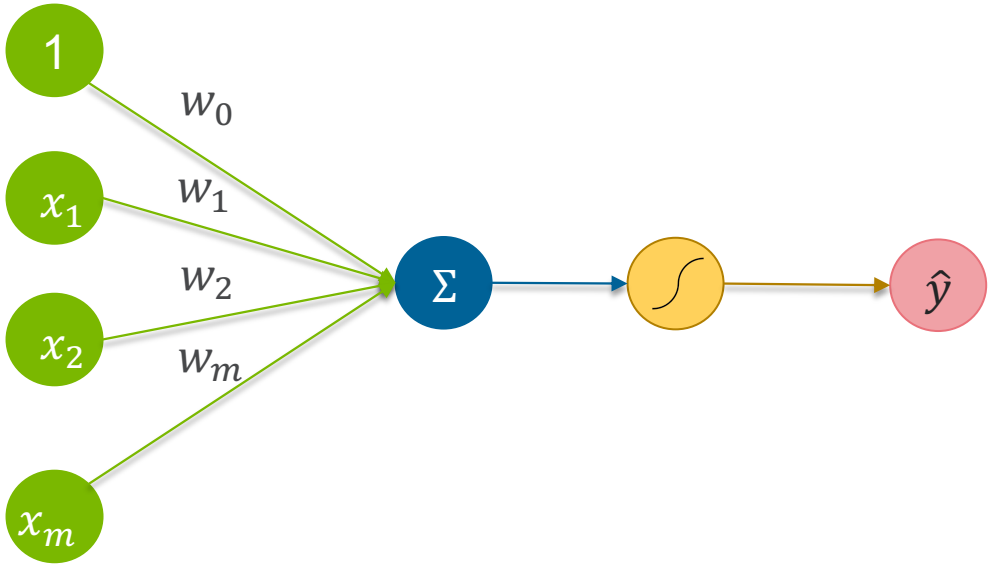


BRAIN AND NEURONS





PERCEPTRON: FORWARD PASS



Input Weight Sum Non-Linearity Output

Linear combination of inputs

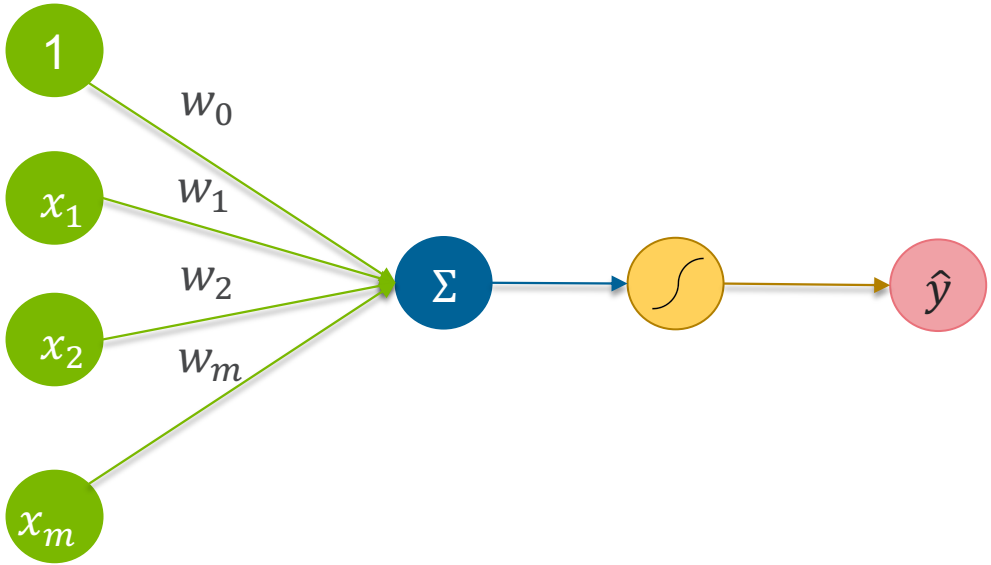
Output

$$\hat{y} = \sigma(w_0 + \sum_{i=1}^{m} w_i \, x_i)$$

Non-linear activation

Bias

PERCEPTRON: FORWARD PASS

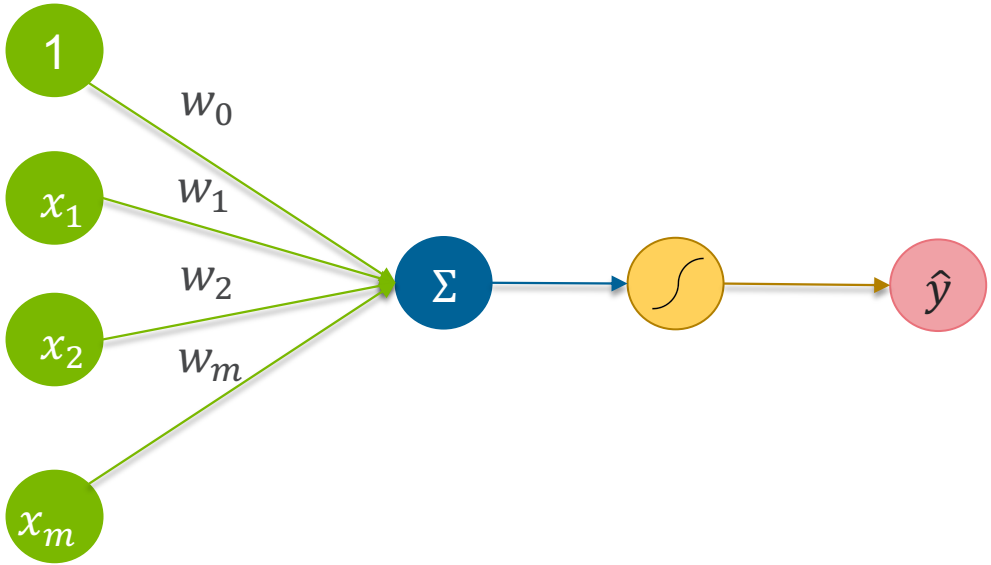


$$\hat{y} = \sigma\left(w_0 + \sum_{i=1}^{m} w_i\, x_i\right)$$

$$\hat{y} = \sigma\left(w_0 + X^T W\right)$$

where, $X = \begin{bmatrix} x_1 \\ x_2 \\ x_m \end{bmatrix}$ $W = \begin{bmatrix} w_1 \\ w_2 \\ w_m \end{bmatrix}$

PERCEPTRON: FORWARD PASS



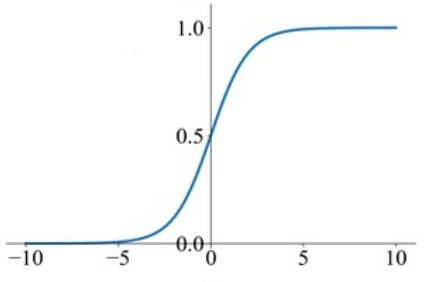
Input Weight Sum Non-Linearity Output

Activation Function

$$\hat{y} = \sigma\left(w_0 + X^T W\right)$$

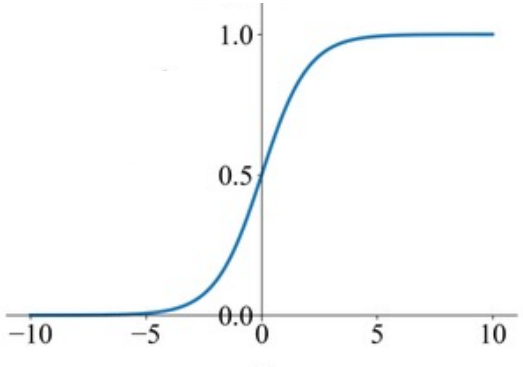
Example: Sigmoid function

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



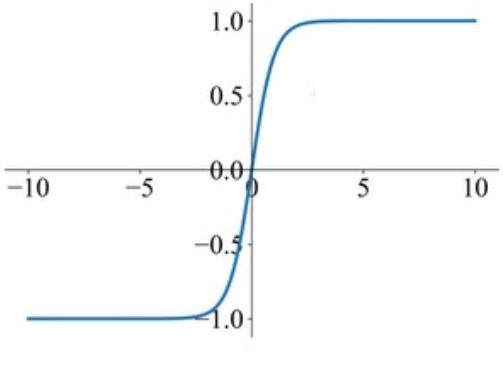
COMMON ACTIVATION FUNCTIONS

Sigmoid



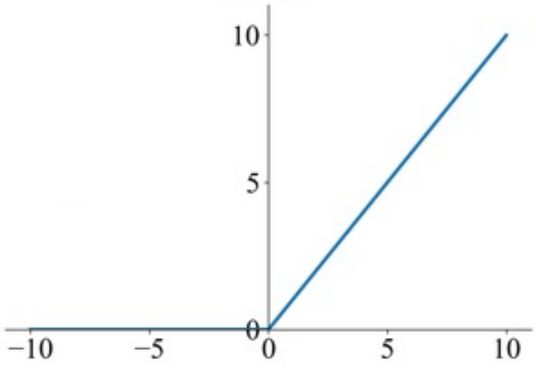
$$\sigma\,(z) = \frac{1}{1 + \ e^{-z}}$$

Tanh



$$\sigma\,(z) = \frac{e^{z} - \ e^{-z}}{e^{z} + \ e^{-z}}$$

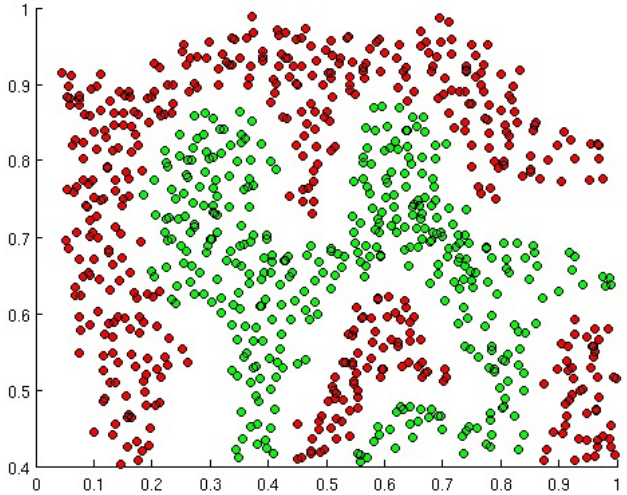
ReLu



$$\sigma\,(z) = \max(0, z)$$

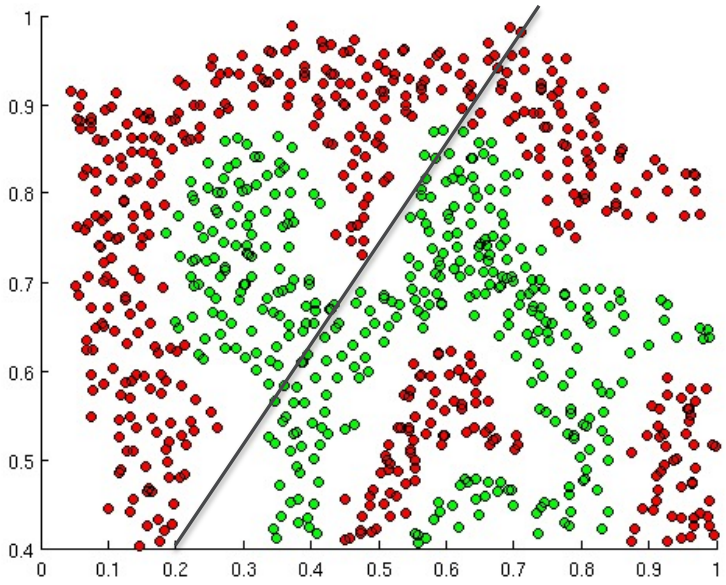
IMPORTANCE OF AN ACTIVATION FUNCTION

Introduce non-linearity into the network

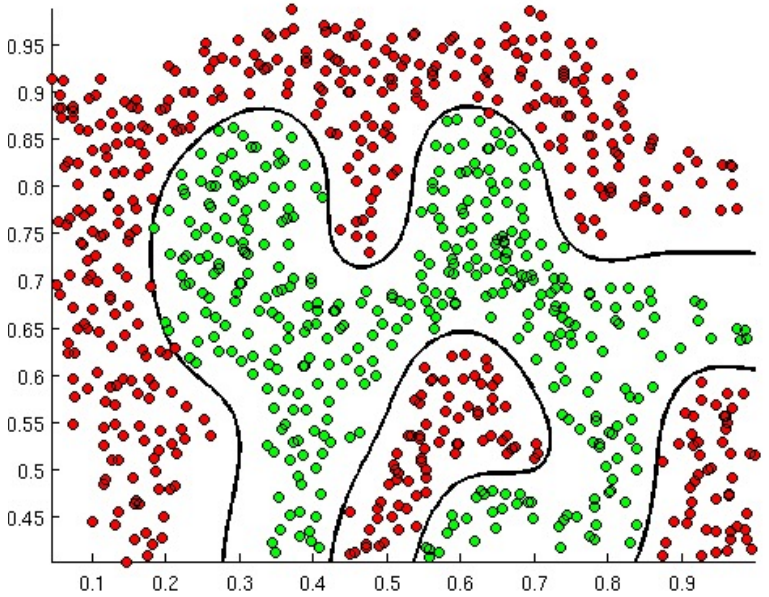


What if you want to build a neural network to separate green and red points

IMPORTANCE OF AN ACTIVATION FUNCTION



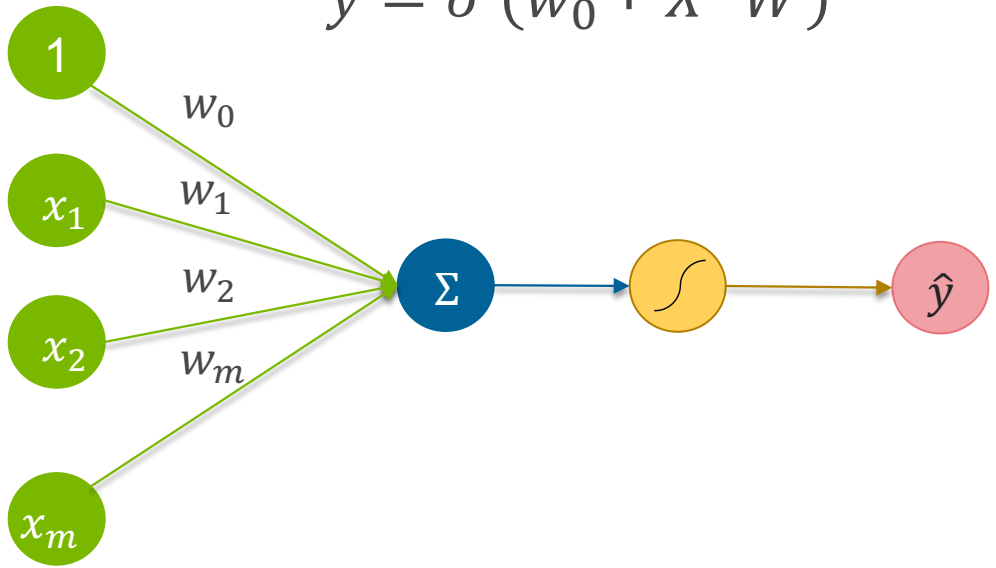
Linear activation function produce linear decision



Non-linear activation function can approximate arbitrarily complex function

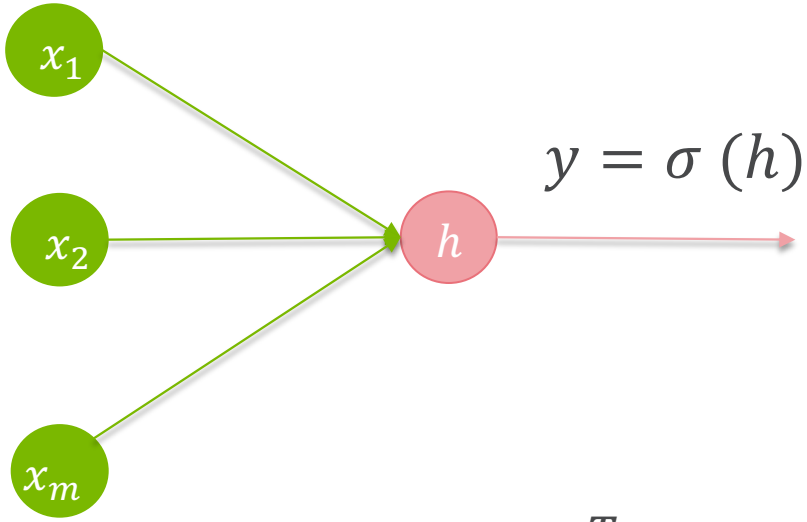
PERCEPTRON: FORWARD PASS

$$\hat{y} = \sigma\left(w_0 + X^T W\right)$$



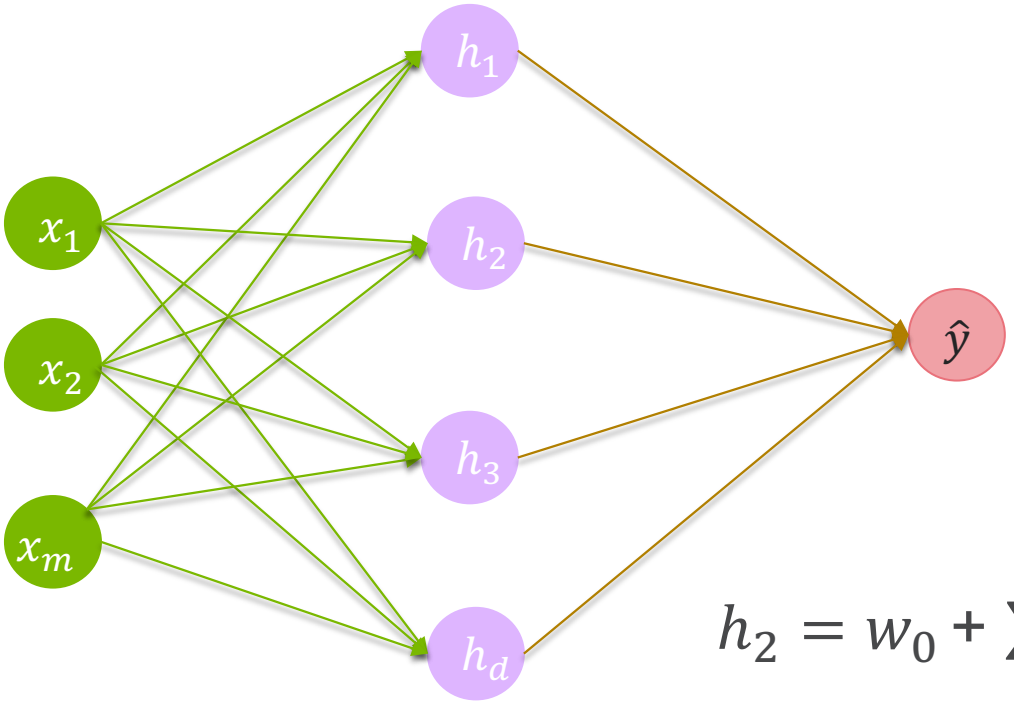
Input Weight Sum Non-Linearity Output

PERCEPTRON SIMPLIFIED



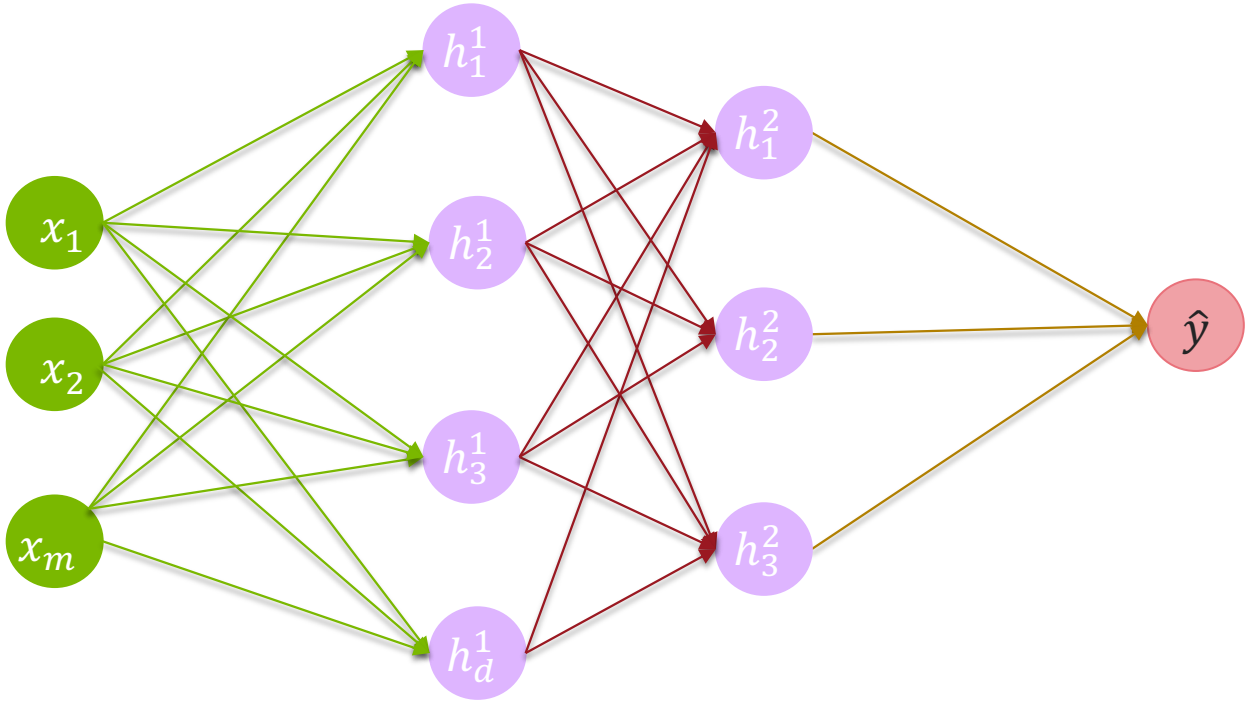
$$h = w_0 + X^T W$$

MULTILAYER PERCEPTRON



$$h_2 = w_0 + \sum_{j=1}^{m} x_j \, w_j$$

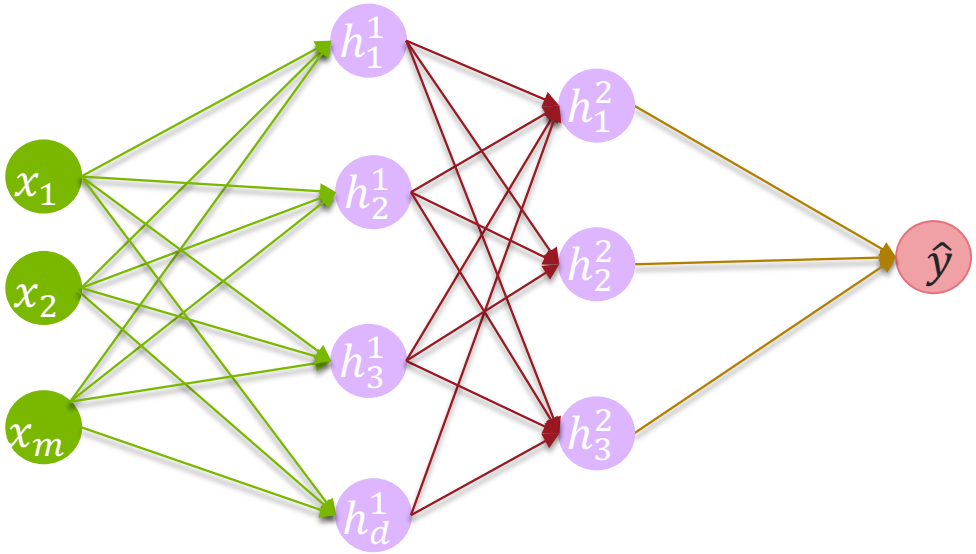
DEEP NEURAL NETWORK



Number of hidden layers > 1

BACKWARD PASS: COMPUTE LOSS

The loss of network measures the cost incurred from the incorrect prediction

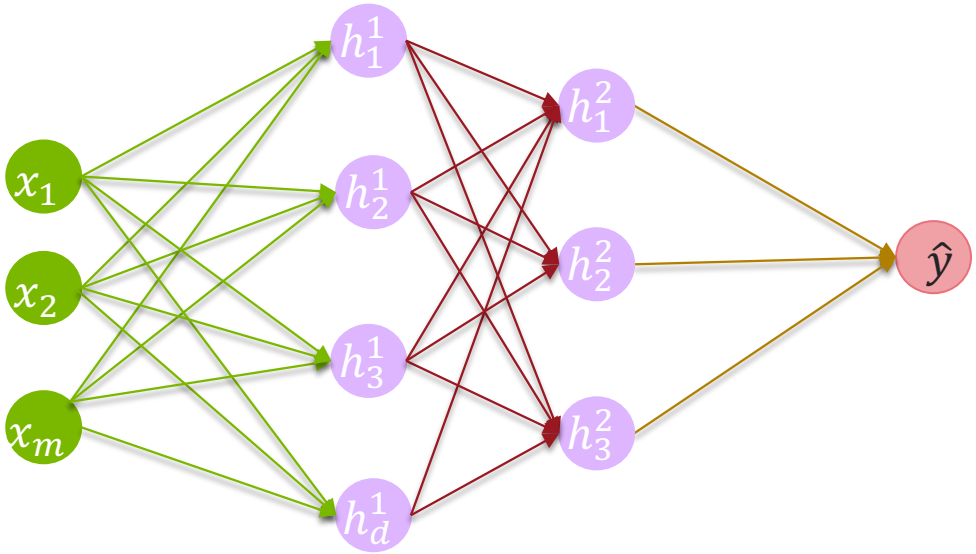


$$\mathcal{L}\left(f(x^i; W), \quad y^i\right)$$

Predicted Actual

BACKWARD PASS: COMPUTE LOSS

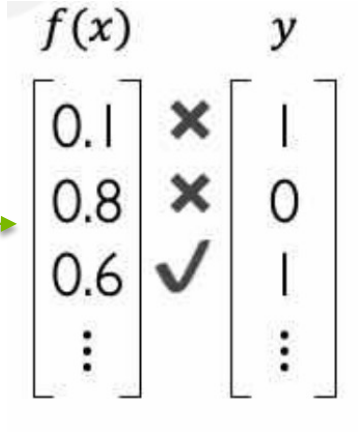
The loss of network measures the cost incurred from the incorrect predictions

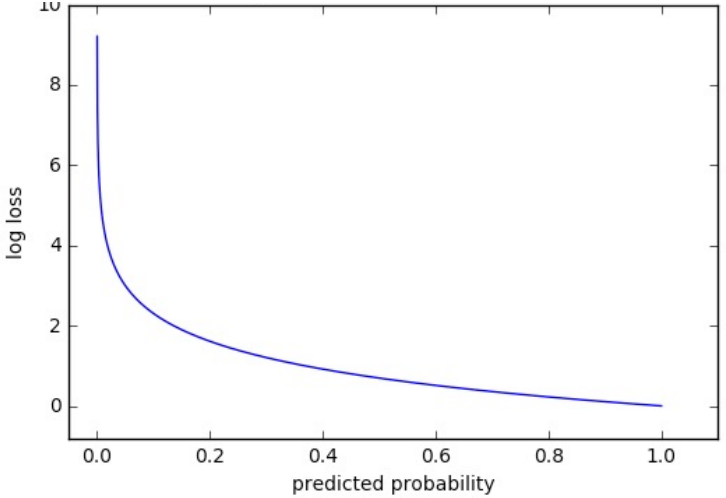


Cost function
Objective function $J(W) = \sum_{i=1}^{n} \mathcal{L}\,(\underbrace{f(x^i; W)}_{\text{Predicted}}, \quad \underbrace{y^i}_{\text{Actual}})$

BINARY CROSS ENTROPY LOSS

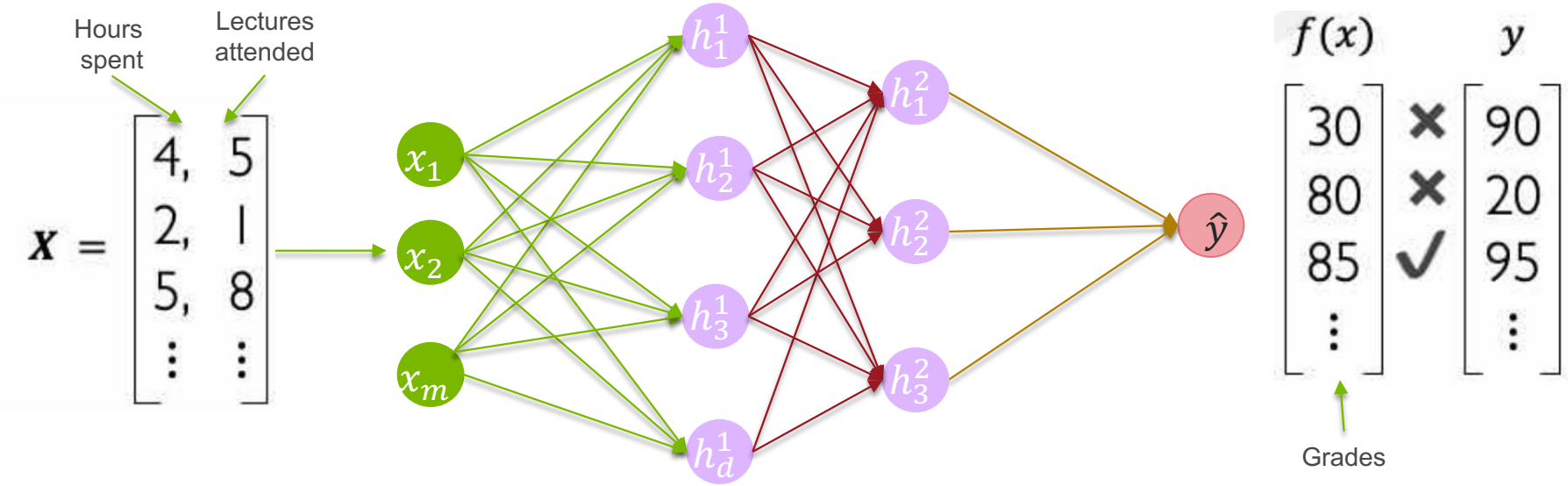
DNN





$$J(W) = -\frac{1}{n}\sum_{i=1}^{n} y^i \log(f(x^i; W)) + (1 - y^i)\log(1 - f(x^i; W))$$

MEAN SQUARE ERROR



$$J(W) = -\frac{1}{n}\sum_{i=1}^{n}(y^i - f(x^i; W))^2$$

LOSS OPTIMIZATION

Want to find network weights that achieve the lowest loss

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f(x^i; W), \quad y^i \right)$$

$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

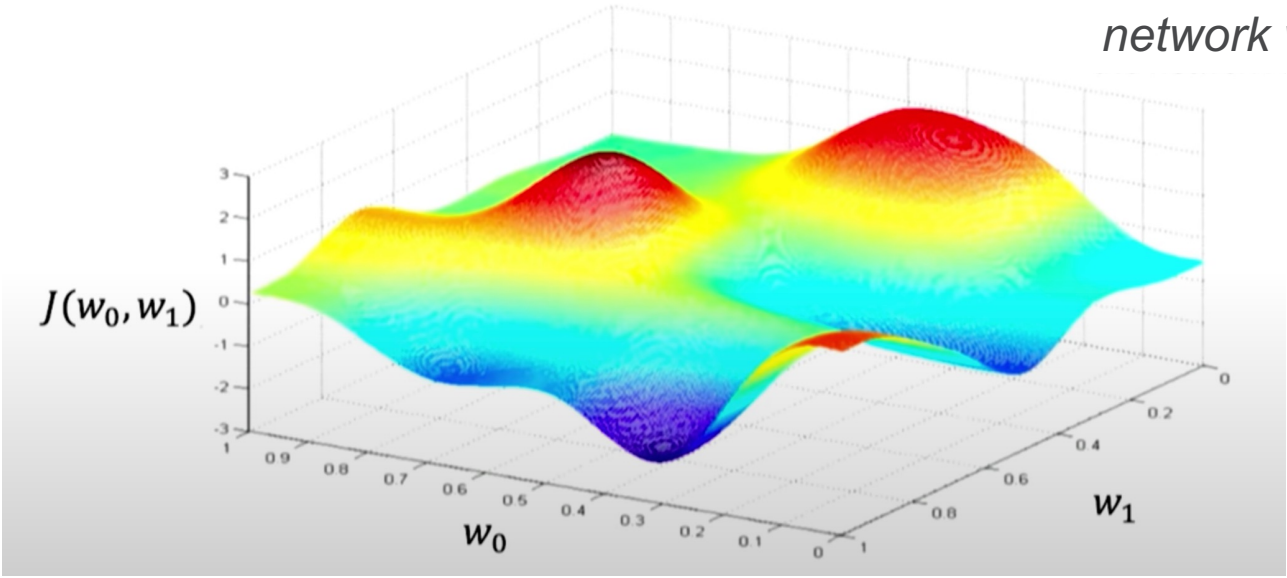
Remember:

$W = \{W^0, W^1, ..\}$

LOSS OPTIMIZATION

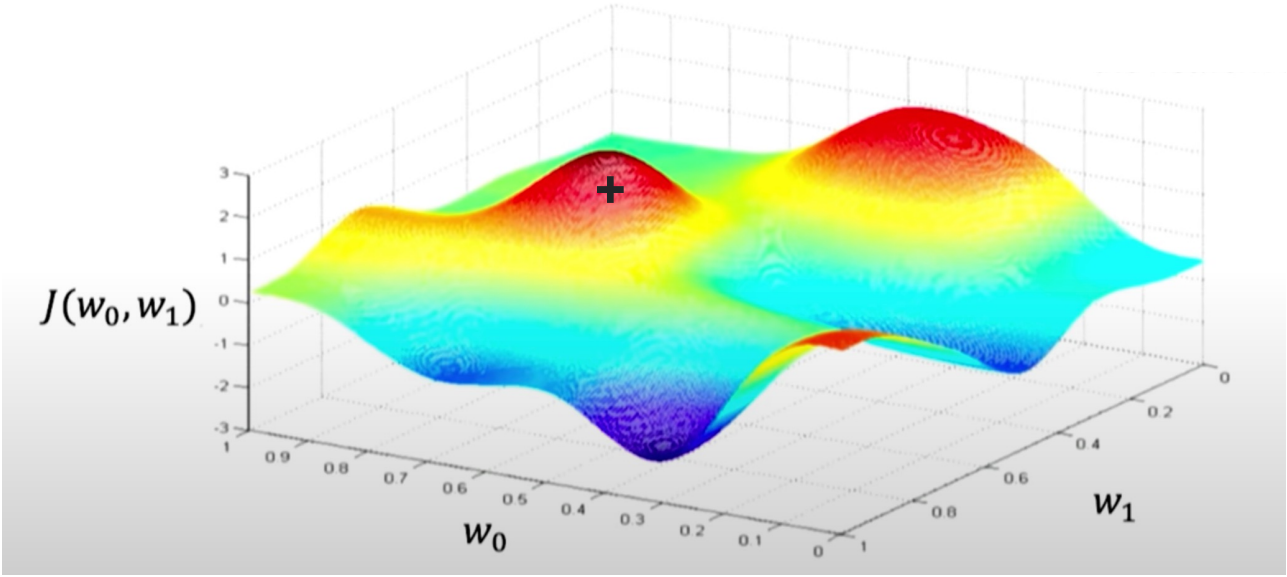
$$W^* = \underset{W}{\operatorname{argmin}}\, J(W)$$

Remember:
Loss is function of network weights



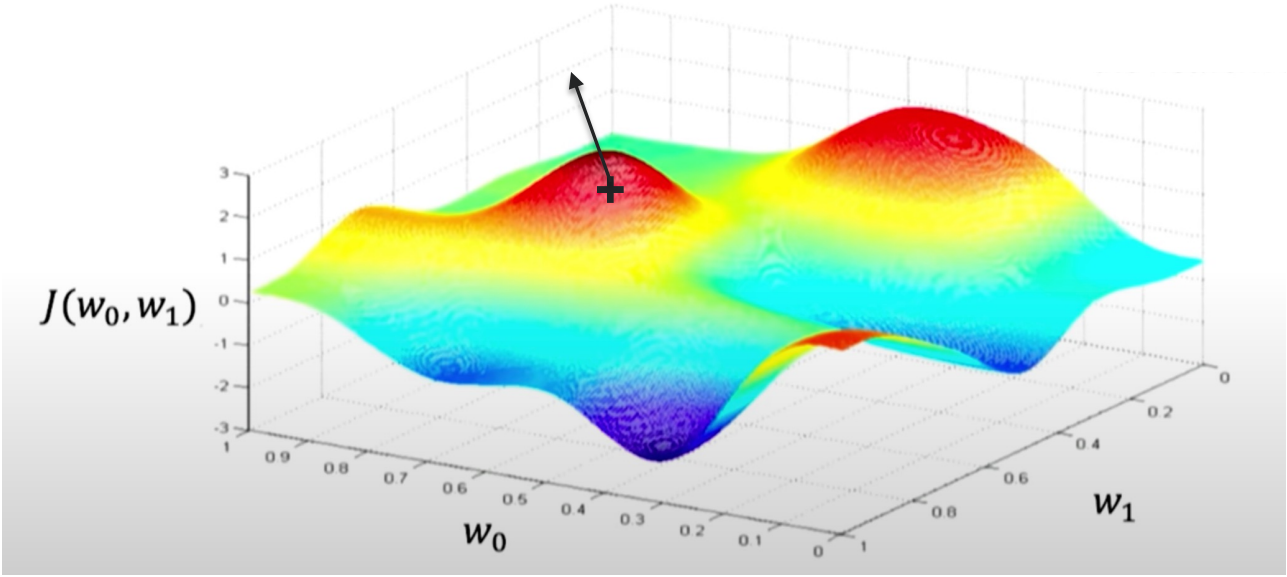
LOSS OPTIMIZATION

Randomly pick the initial (W_0, W_1)



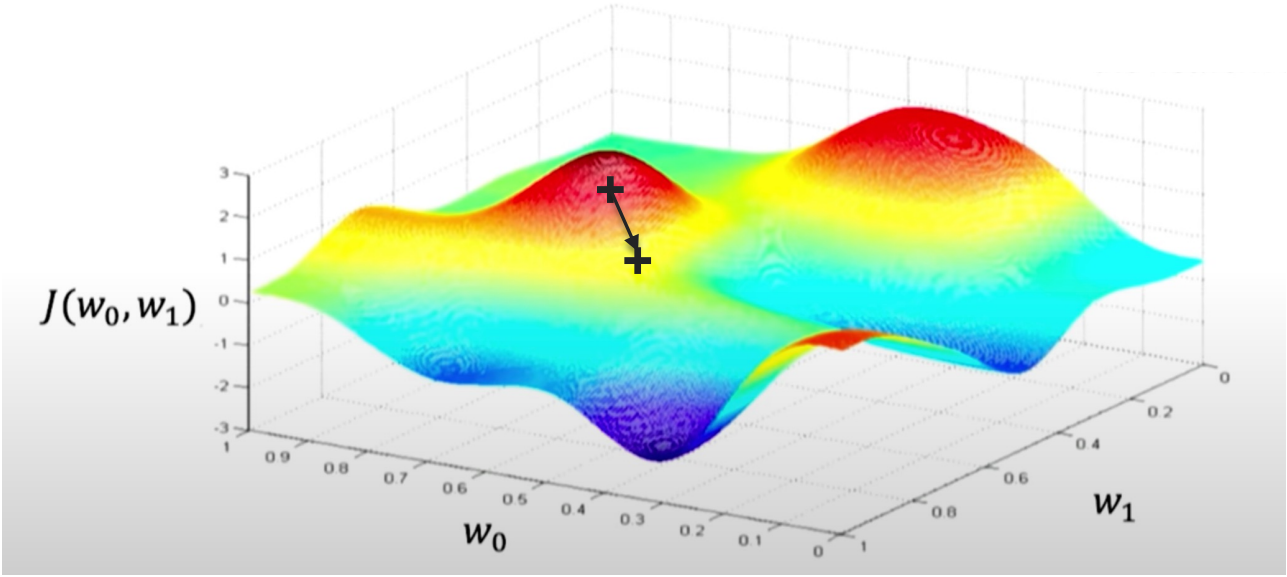
LOSS OPTIMIZATION

Compute gradient $\frac{\partial J(W)}{\partial W}$



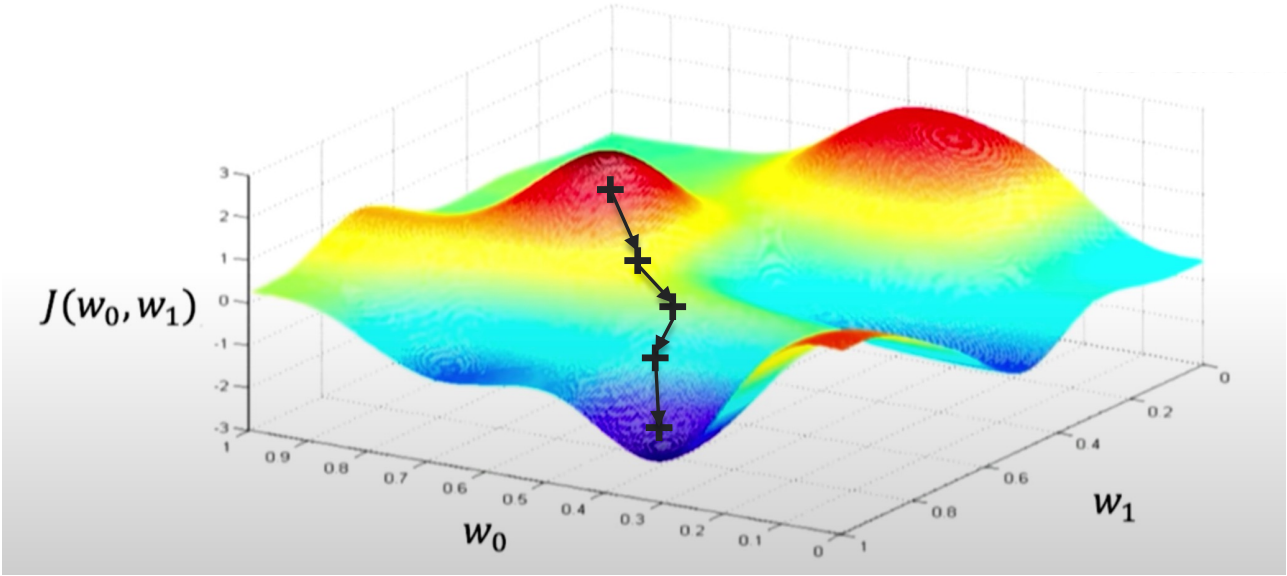
LOSS OPTIMIZATION

Take a step in the opposite direction of the gradient



LOSS OPTIMIZATION

Repeat until convergence

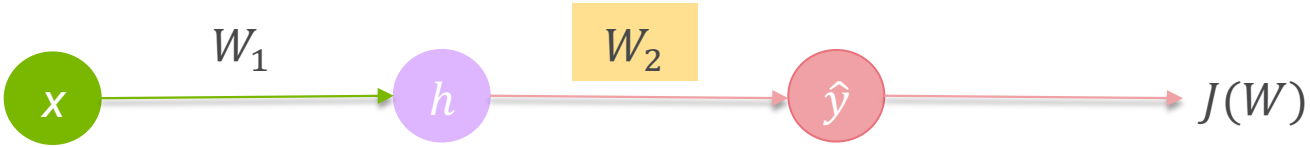


GRADIENT DESCENT

Algorithm

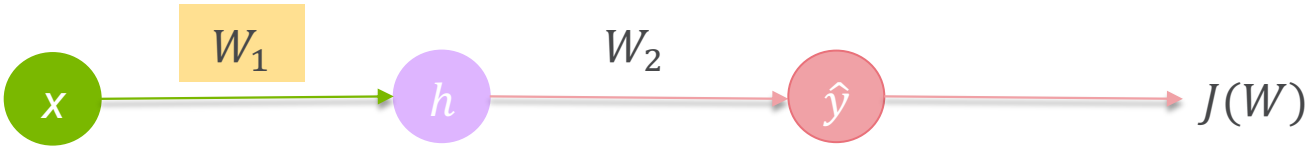
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

BACKPROPAGATION



$$\frac{\partial J(W)}{\partial W_2} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial W_2}$$

BACKPROPAGATION

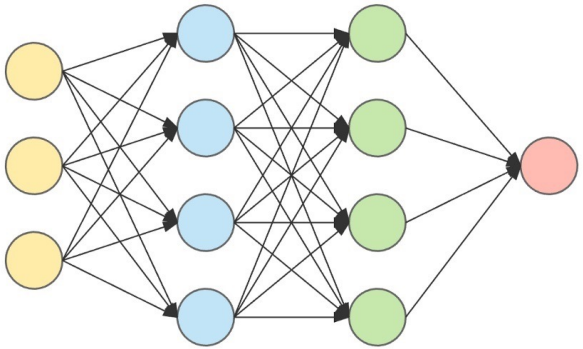


$$\frac{\partial J(W)}{\partial W_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial W_1}$$

$$\frac{\partial J(W)}{\partial W_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h} * \frac{\partial h}{\partial W_1}$$

Repeat this for every weight of the network using gradient from previous layer

Train a Neural Network




Argonne
NATIONAL LABORATORY