Deep Learning

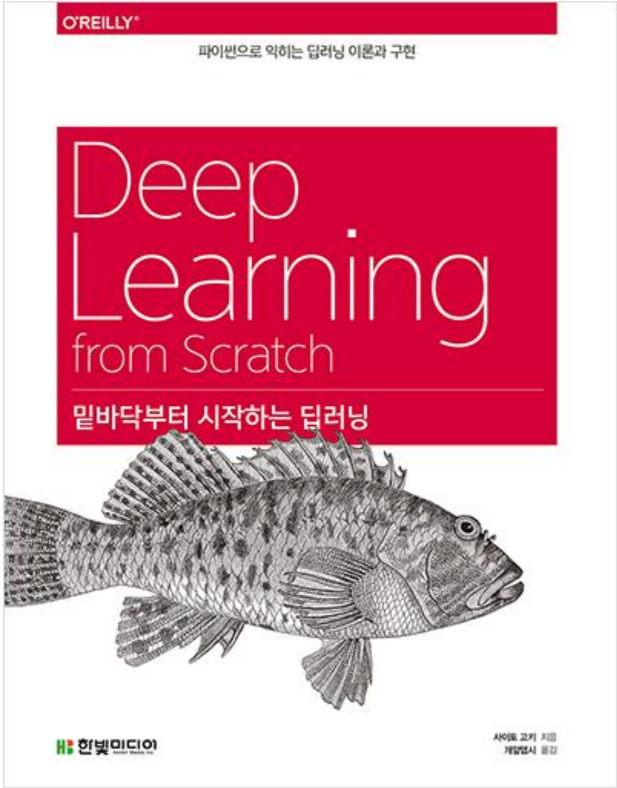
1주차

12기 이두형
12기 임효진

KU-BIG

Curriculum

1. 딥러닝 소개 및 기초 (XOR문제, 퍼셉트론, 활성화 함수 등)
2. Multi-layer Neural Network (Loss Function, Gradient Descending, Backpropagation, MNIST practice)
3. CNN 소개 및 기초 (Convolution, Padding, Stride, Pooling등 기초 개념 소개)
4. CNN 실습 (세션 후 조별 과제 부여)
5. RNN, LSTM, GRU 소개 및 기초
6. seq2seq 소개 및 실습 (세션 후 조별 과제 부여)
7. 프로젝트 발표

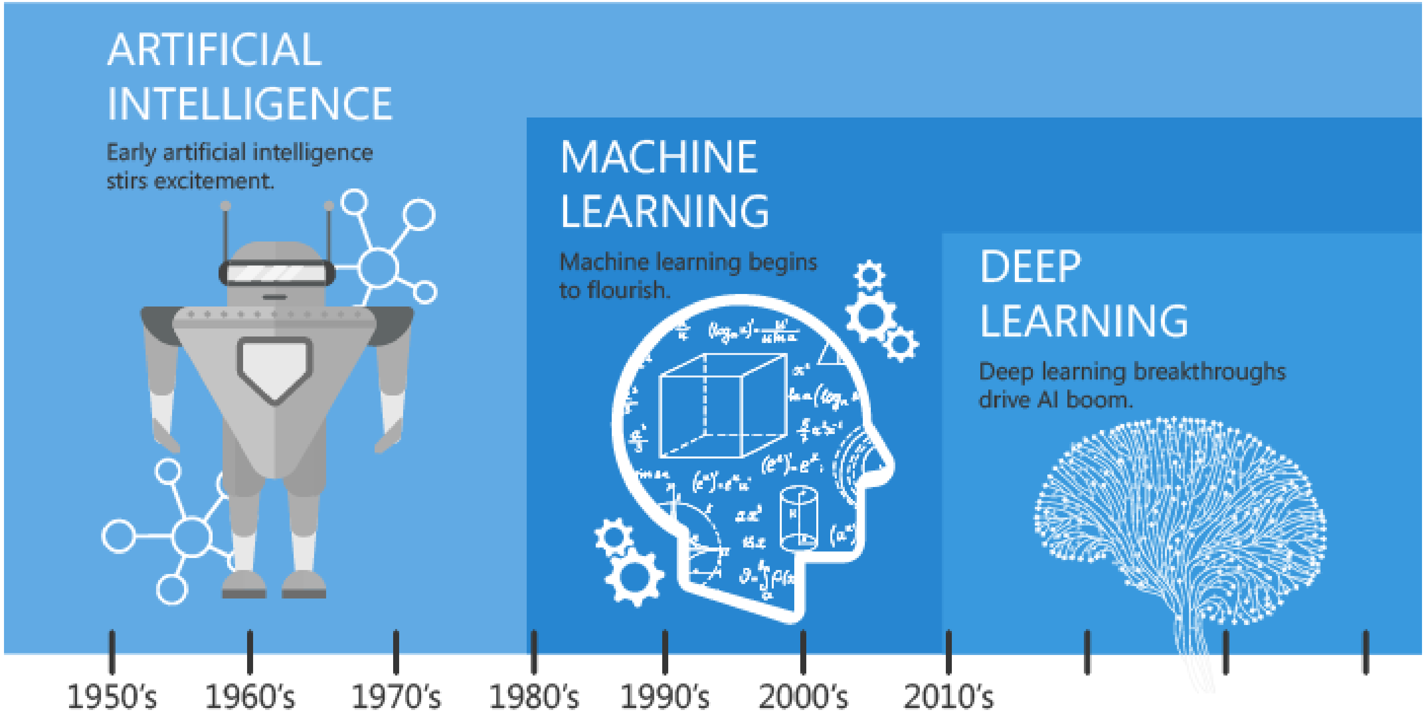


kaggle

colab

- Google Duplex: https://youtu.be/znNe4pMCsD4
- Word Lens: https://youtu.be/h2OfQdYrHRs
- Google Assistant: https://youtu.be/Pk6a6mvOoJA

- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Cars: drivable area, lane keeping
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep RL


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

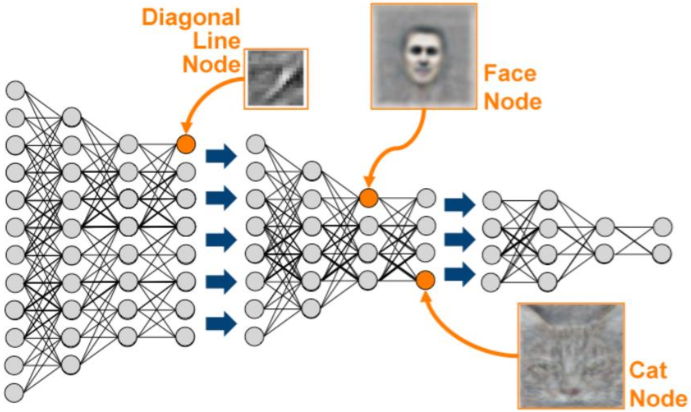
Machine Learning

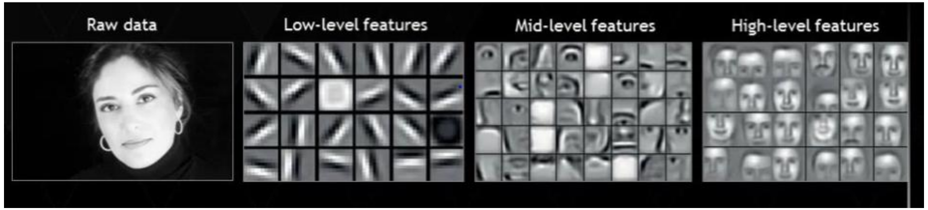
- *Input data point*
- *Examples of the expected output*
- *A way to measure whether the algorithm is doing a good job*

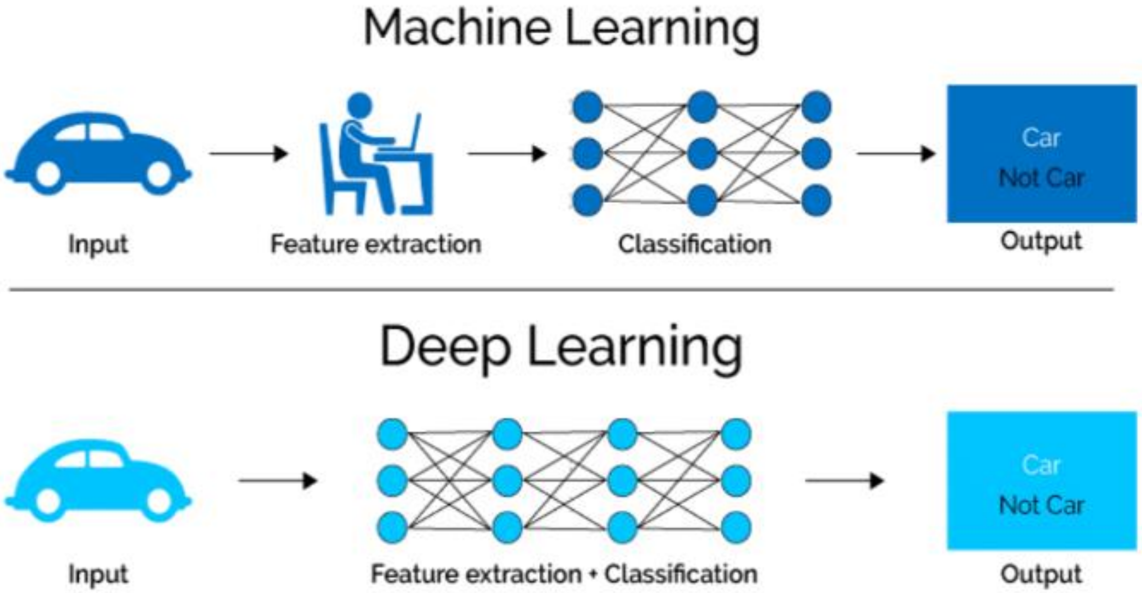
KU-BIG

Deep Learning

Deep Learning is **Representation Learning**
(aka Feature Learning)

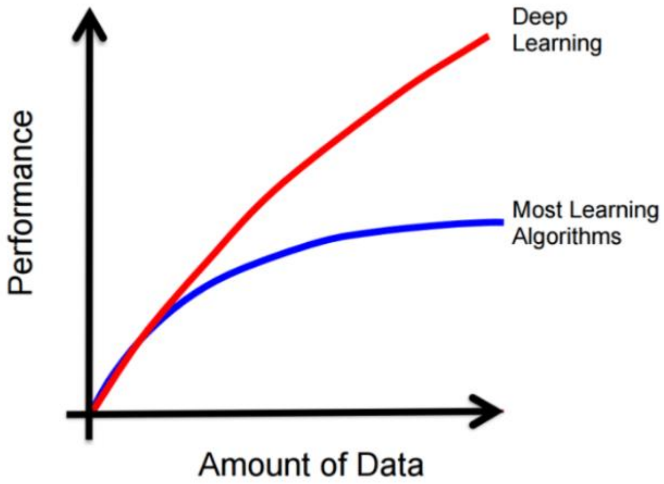





Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

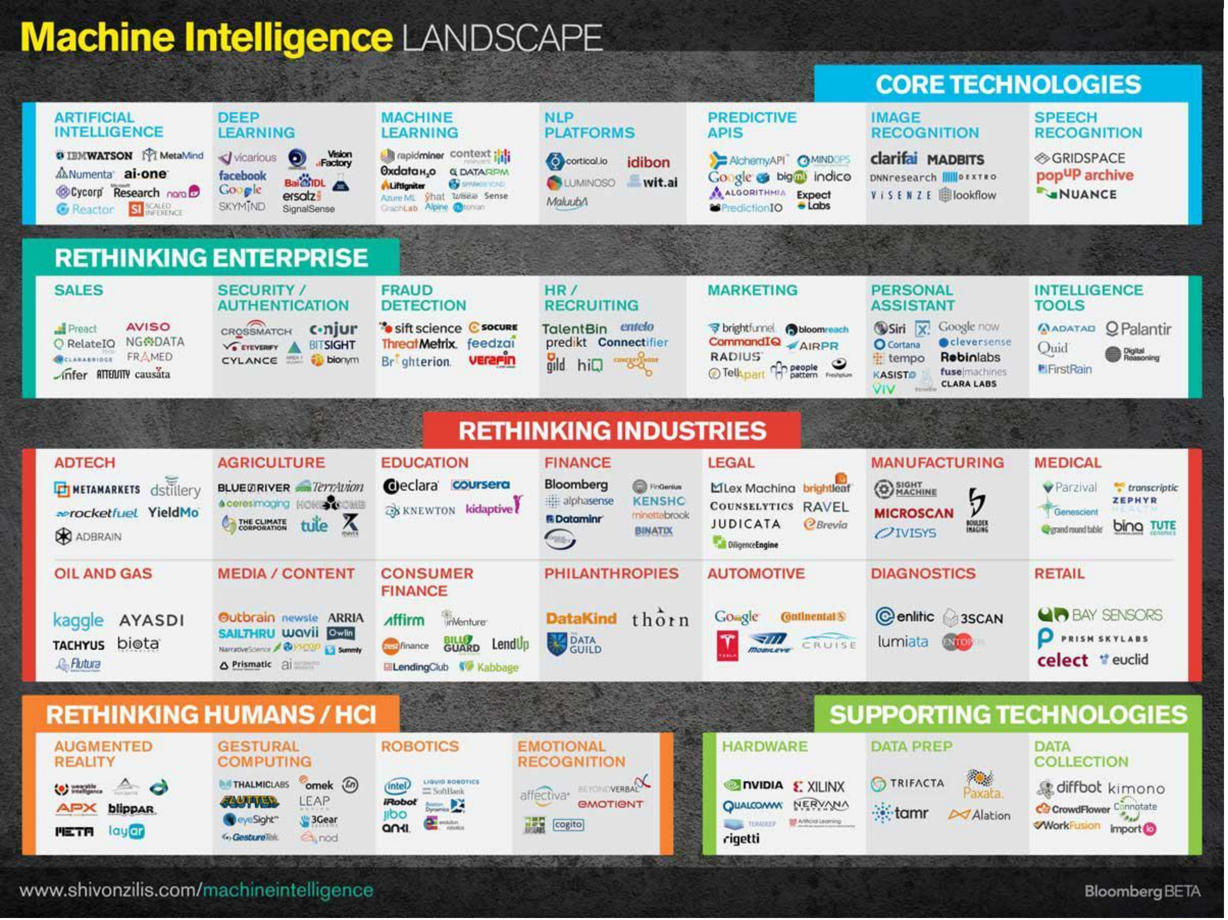
ML vs DL

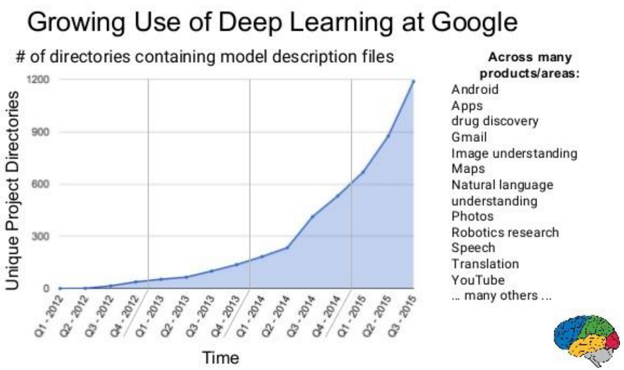
1. Data Dependency
2. Hardware Dependency
3. Execution Time
4. Interpretability

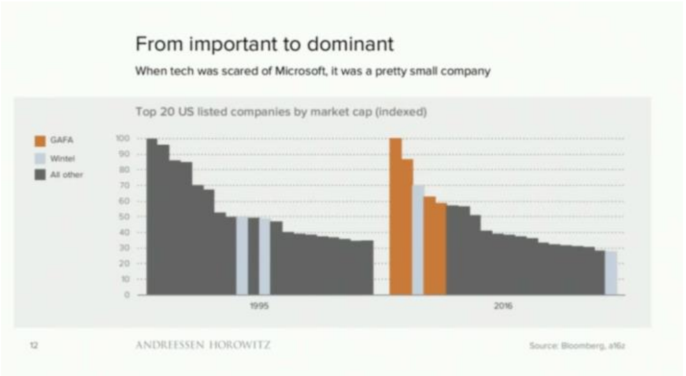


Why now?

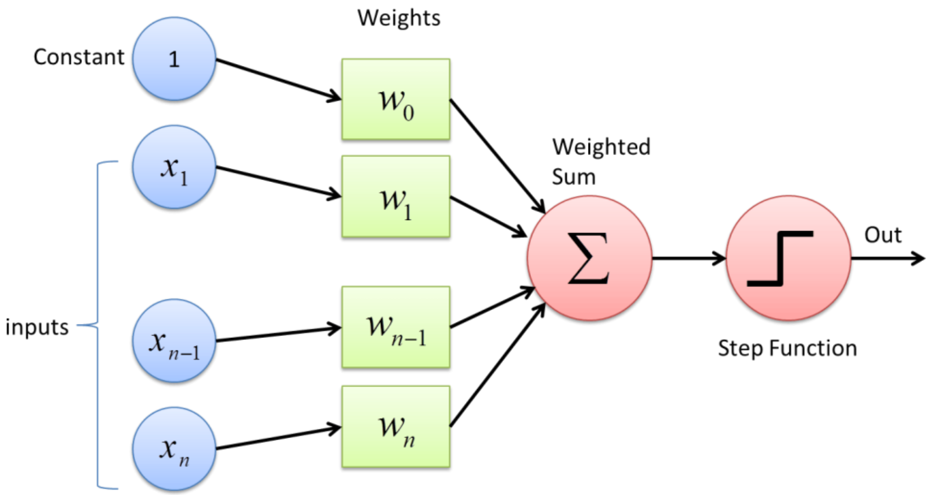
- Big Data
  - Larger Datasets
  - Easier Collection & Storage
- Hardware
  - GPU
  - Massively Parallelizable
- Software
  - Improved Techniques
  - Toolboxes
  - Communities


Machine Intelligence LANDSCAPE
CORE TECHNOLOGIES
ARTIFICIAL INTELLIGENCE
DEEP LEARNING
MACHINE LEARNING
NLP PLATFORMS
PREDICTIVE APIS
IMAGE RECOGNITION
SPEECH RECOGNITION
RETHINKING ENTERPRISE
SALES
SECURITY / AUTHENTICATION
FRAUD DETECTION
HR / RECRUITING
MARKETING
PERSONAL ASSISTANT
INTELLIGENCE TOOLS
RETHINKING INDUSTRIES
ADTECH
AGRICULTURE
EDUCATION
FINANCE
LEGAL
MANUFACTURING
MEDICAL
OIL AND GAS
MEDIA / CONTENT
CONSUMER FINANCE
PHILANTHROPIES
AUTOMOTIVE
DIAGNOSTICS
RETAIL
RETHINKING HUMANS / HCI
AUGMENTED REALITY
GESTURAL COMPUTING
ROBOTICS
EMOTIONAL RECOGNITION
SUPPORTING TECHNOLOGIES
HARDWARE
DATA PREP
DATA COLLECTION
www.shivonzilis.com/machineintelligence
Bloomberg BETA

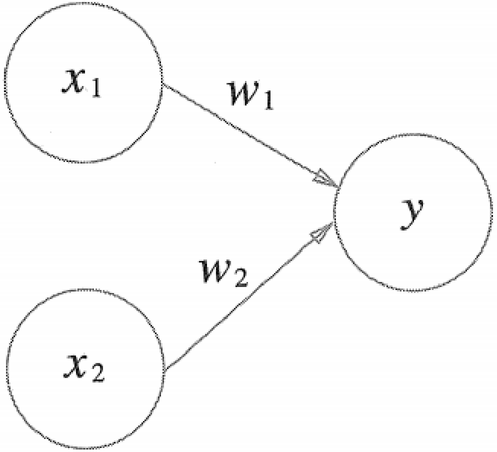

Growing Use of Deep Learning at Google
# of directories containing model description files
Unique Project Directories
1200
900
600
300
0
Q1 - 2012
Q2 - 2012
Q3 - 2012
Q4 - 2012
Q1 - 2013
Q2 - 2013
Q3 - 2013
Q4 - 2013
Q1 - 2014
Q2 - 2014
Q3 - 2014
Q4 - 2014
Q1 - 2015
Q2 - 2015
Q3 - 2015
Time
Across many products/areas:
Android
Apps
drug discovery
Gmail
Image understanding
Maps
Natural language understanding
Photos
Robotics research
Speech
Translation
YouTube
... many others ...


From important to dominant
When tech was scared of Microsoft, it was a pretty small company
Top 20 US listed companies by market cap (indexed)
GAFA
Wintel
All other
100
90
80
70
60
50
40
30
20
10
0
1995
2016
12
ANDREESSEN HOROWITZ
Source: Bloomberg, a16z

Perceptron



Weights
Constant
1
w_0
x_1
w_1
Weighted
Sum
Σ
Out
inputs
x_{n-1}
w_{n-1}
Step Function
x_n
w_n



$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

KU-BIG

- AND

x_1	x_2	y
0	0	0
1	0	0
0	1	0

- OR

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

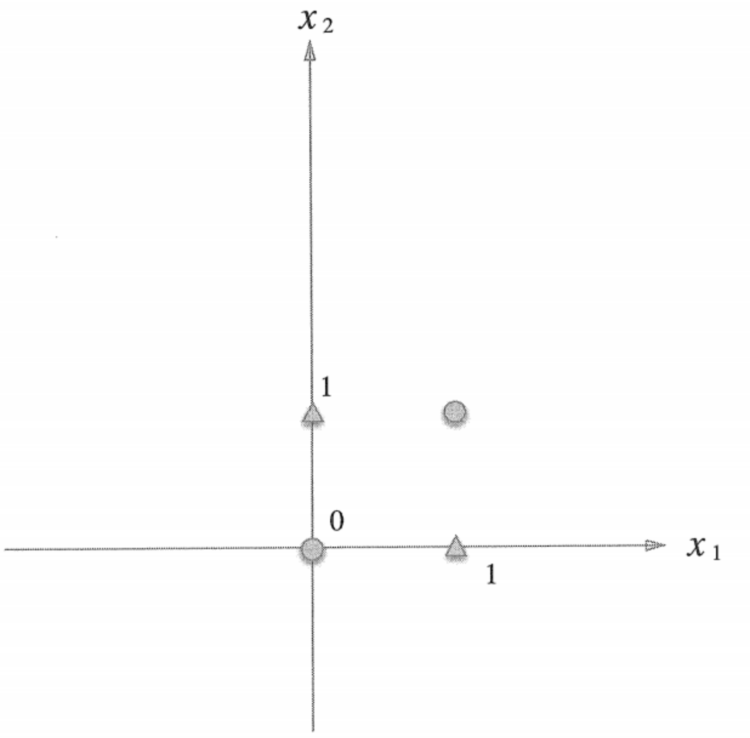
- NAND

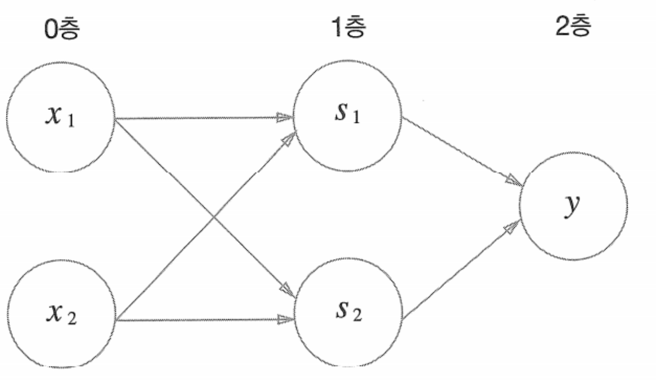
x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

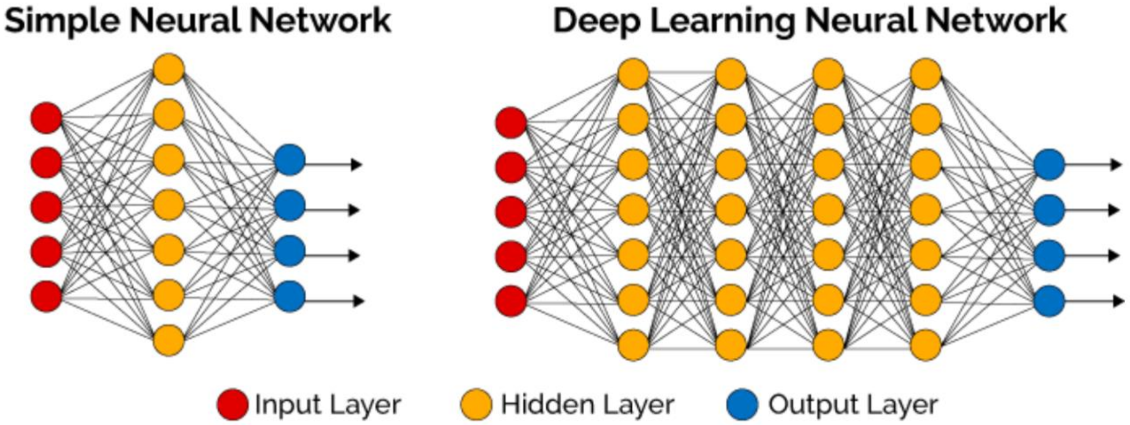
- XOR

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

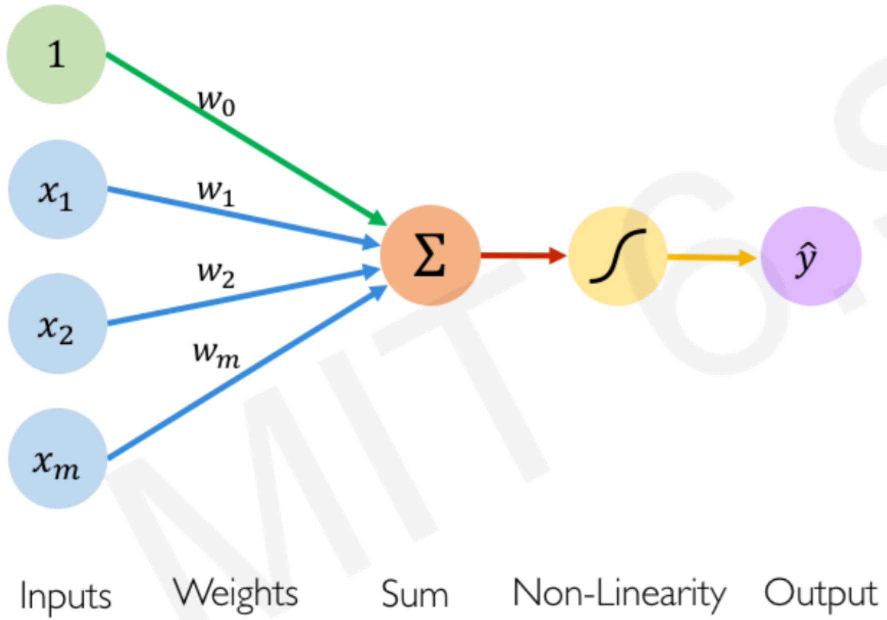
$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases}$$




0층
1층
2층
x_1
x_2
s_1
s_2
y


Simple Neural Network
Deep Learning Neural Network
Input Layer
Hidden Layer
Output Layer

The Perceptron: Forward Propagation



Output

Linear combination of inputs

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

Non-linear activation function

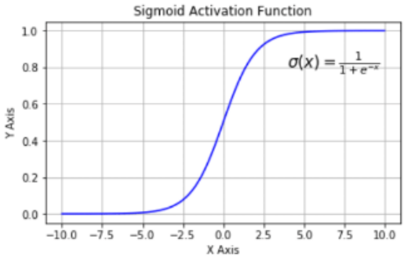
Bias

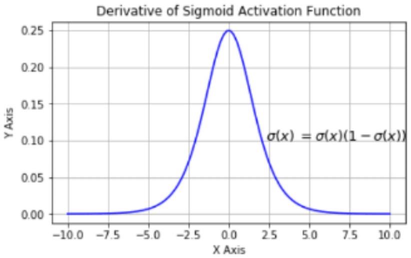
$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Dense layer from scratch

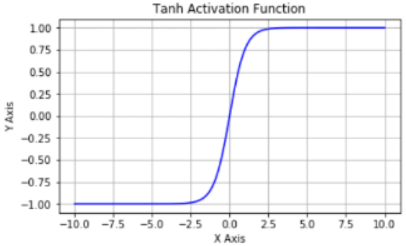
```
class MyDenseLayer(tf.keras.layers.Layer):
  def __init__(self, input_dim, output_dim):
    super(MyDenseLayer, self).__init__()

    # Initialize weights and bias
    self.W = self.add_weight([input_dim, output_dim])
    self.b = self.add_weight([1, output_dim])

  def call(self, inputs):
    # Forward propagate the inputs
    z = tf.matmul(inputs, self.W) + self.b

    # Feed through a non-linear activation
    output = tf.math.sigmoid(z)

    return output
```

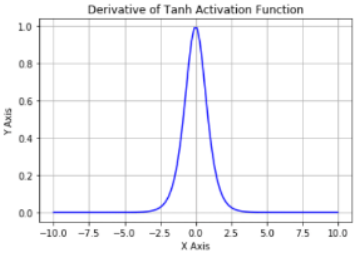




Sigmoid `tf.math.sigmoid(z)`

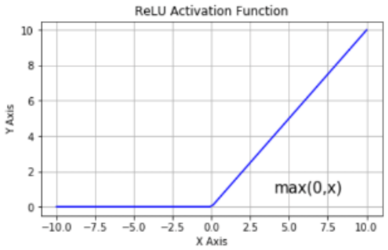
- Vanishing gradients
- Not zero centered

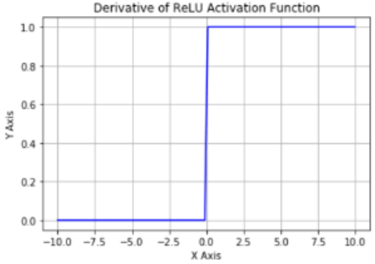




Tanh `tf.math.tanh(z)`

- Vanishing gradients

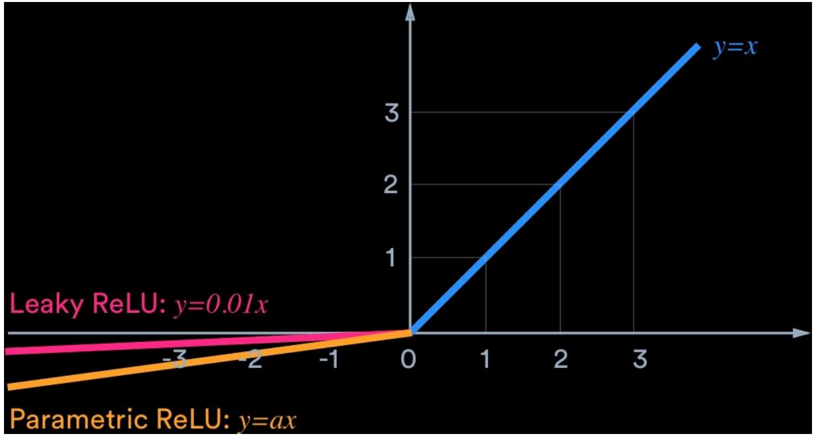




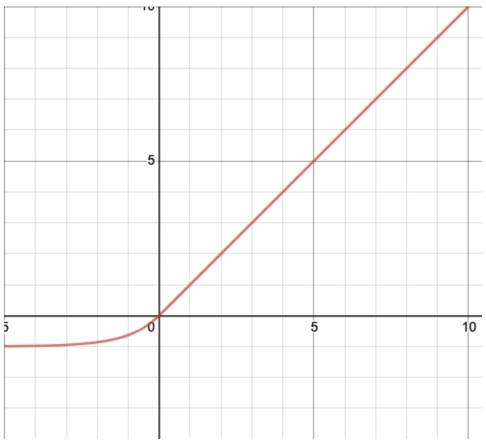
ReLU `tf.nn.relu(z)`

- Not zero centered

- Leaky RELU/Parametric RELU



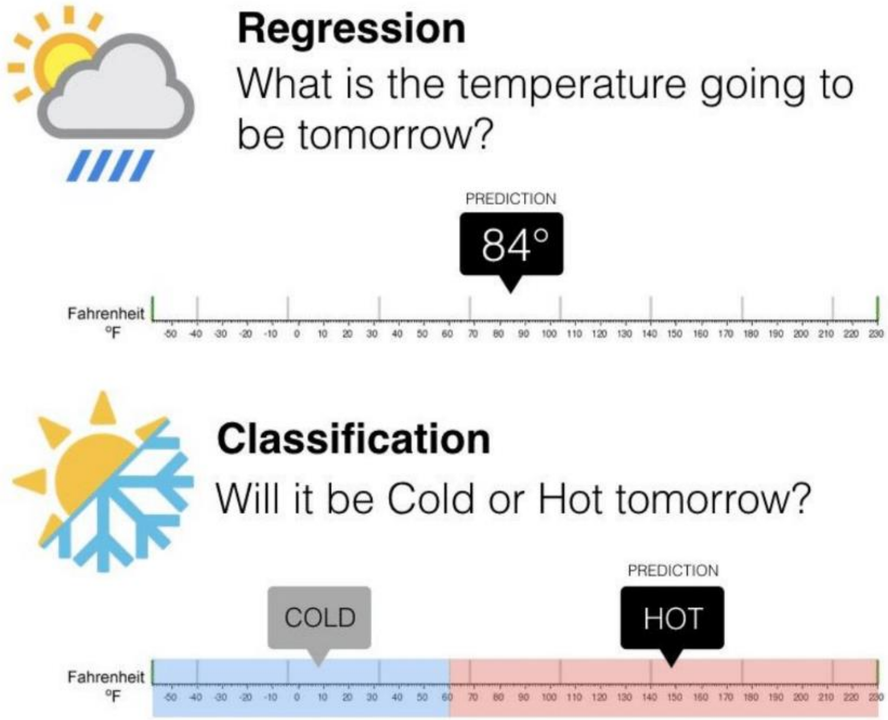
- ELU



- Maxout

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

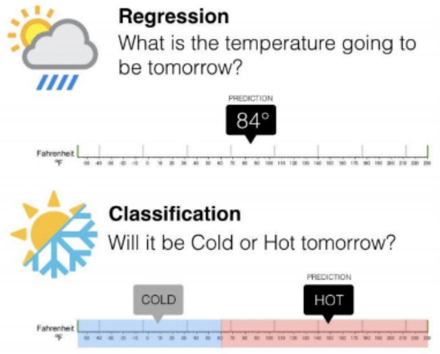
Regression vs Classification



$f(x) = x$ for all elements x in M.[1]

$$softmax(x) = \frac{x_i}{\sum_{j=0}^{k} e^{x_j}} \quad (i = 0, 1, \ldots k)$$

Loss Functions



- Loss function quantifies gap between prediction and ground truth
- For regression:
  - Mean Squared Error (MSE)
- For classification:
  - Cross Entropy Loss

Mean Squared Error

Prediction

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Ground Truth

Cross Entropy Loss

Classes   Prediction

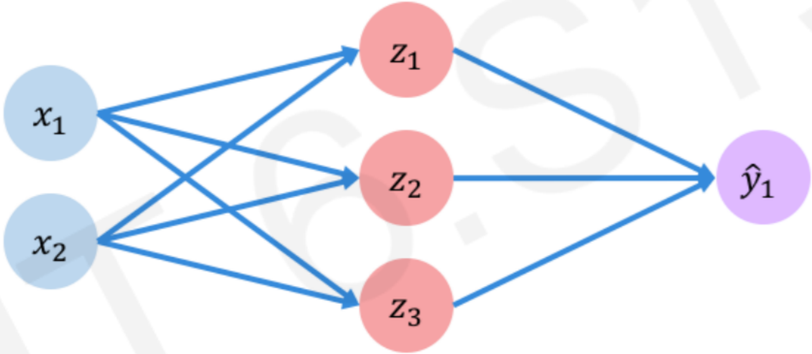
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Ground Truth {0,1}

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$\begin{array}{ccc} f(x) & & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} & \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}$$

Final Grades (percentage)

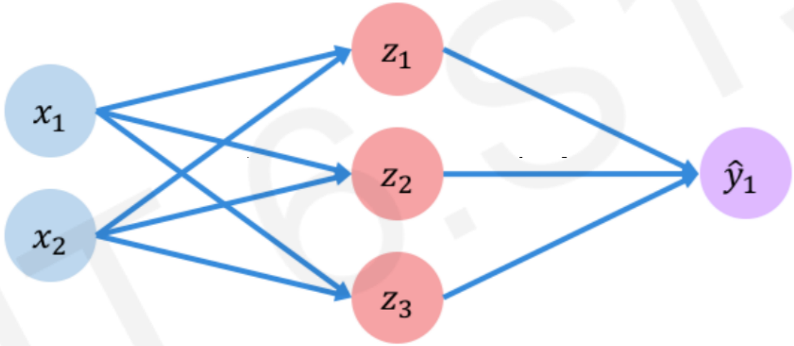
$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)}; \boldsymbol{W}\big)}_{\text{Predicted}}\Big)^2$$

```
loss = tf.reduce_mean( tf.square(tf.subtract(y, predicted)) )
```

Binary Cross Entropy Loss

Cross entropy loss can be used with models that output a probability between 0 and 1

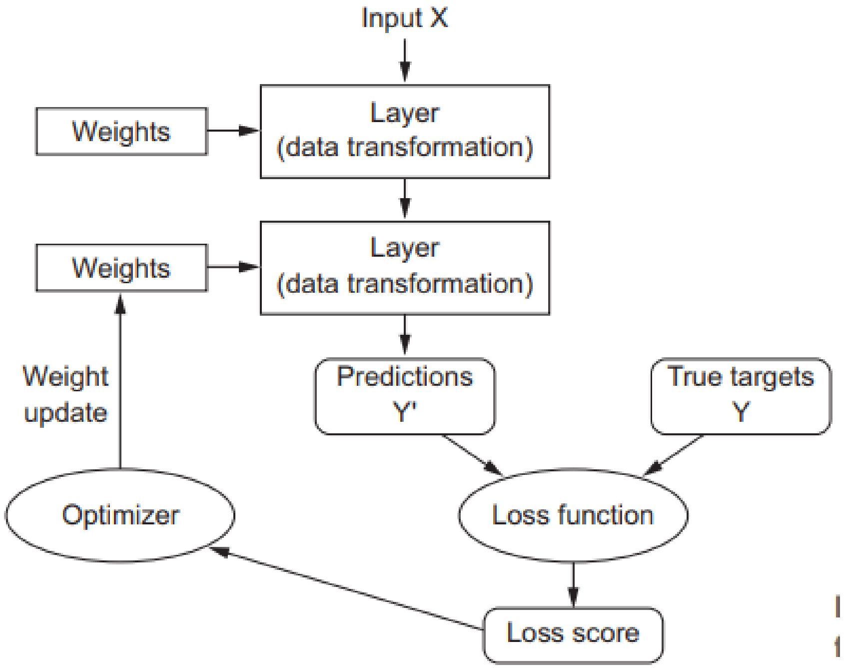
$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$\begin{array}{ccc} f(x) & & y \\ \begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} & \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \end{array}$$

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1 - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

```
loss = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(y, predicted) )
```


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function
Loss score