KAIST Summer Session 2018

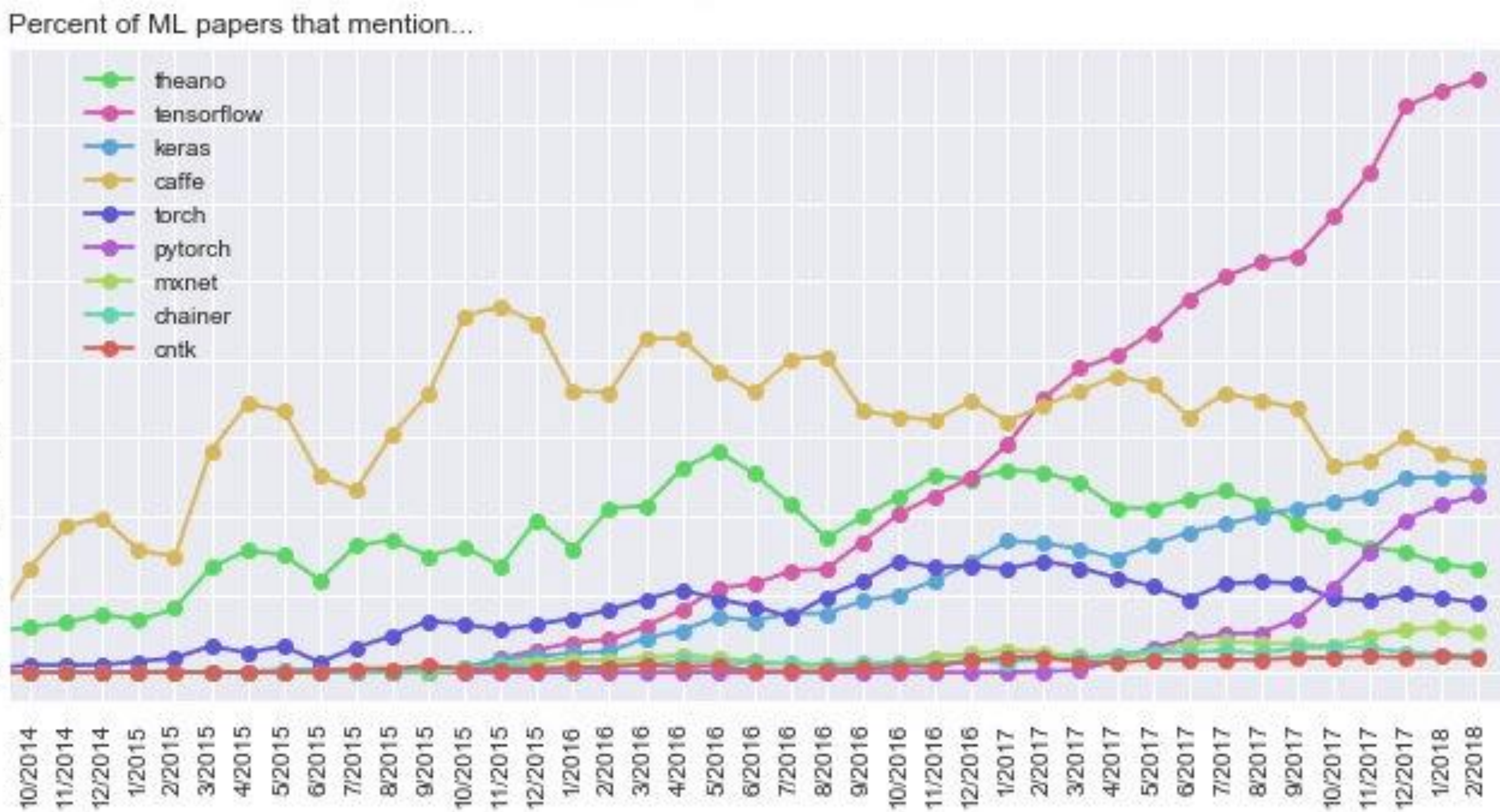
Module 3. Deep Learning with PyTorch

Regression and Neural Network

KAIST College of Business

Jiyong Park

16 August, 2018


Percent of ML papers that mention...
theano
tensorflow
keras
caffe
torch
pytorch
mxnet
chainer
cntk
0
2
4
6
8
10
12
14
10/2014
11/2014
12/2014
1/2015
2/2015
3/2015
4/2015
5/2015
6/2015
7/2015
8/2015
9/2015
10/2015
11/2015
12/2015
1/2016
2/2016
3/2016
4/2016
5/2016
6/2016
7/2016
8/2016
9/2016
10/2016
11/2016
12/2016
1/2017
2/2017
3/2017
4/2017
5/2017
6/2017
7/2017
8/2017
9/2017
10/2017
11/2017
12/2017
1/2018
2/2018

Let's Get Started

Please ensure that you are on the latest pip and numpy packages.
Anaconda is our recommended package manager

OS	Linux	MacOS	**Windows**	
Package Manager	**conda**	pip	Source	
Python	2.7	3.5	**3.6**	3.7
CUDA	8	9.0	9.2	**None**

Run this command:

```
conda install pytorch-cpu -c pytorch
pip3 install torchvision
```

https://pytorch.org/

(Optional) if any problems,
Try to run the anaconda prompt as administrator
In addition, type these first:

```
python –m pip install –upgrade pip
pip install –upgrade setuptools
pip install torchvision
```

Linear Regression

Statistical Style versus Machine Learning Style

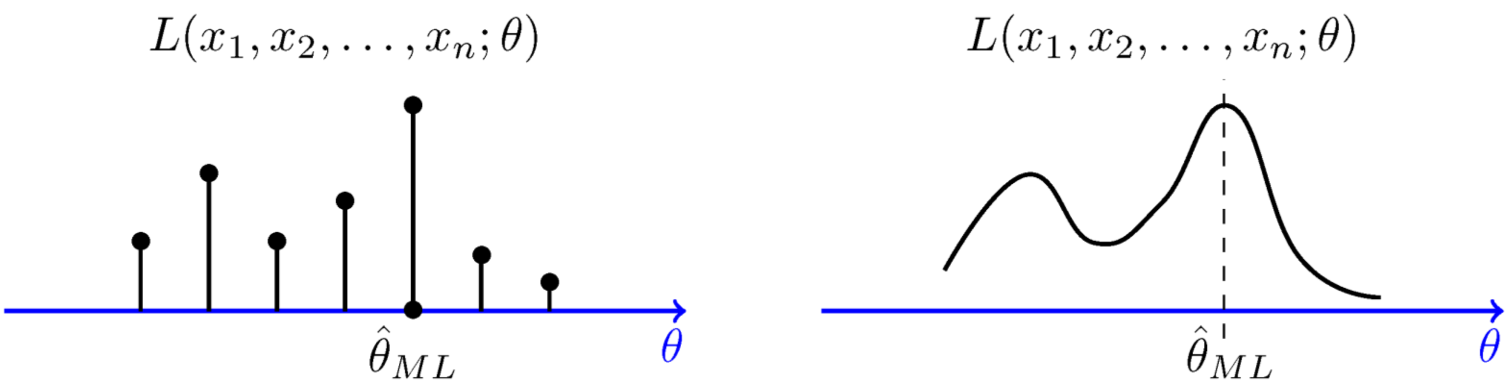
- (Statistical style) Maximum likelihood estimation (MLE)
  - MLE is same as Ordinary Least Square (OLS) under the GM assumptions

- Linear regression in a machine learning style

- Linear regression in the PyTorch way

Statistical Style versus Machine Learning Style

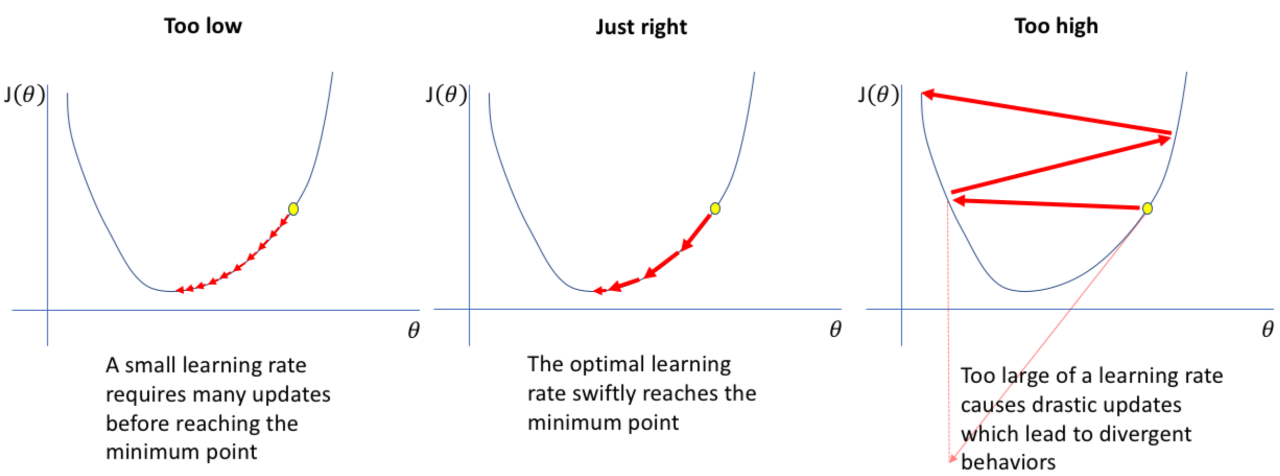
- (Statistical style) Maximum likelihood estimation (MLE)
  - Suppose that we have observed $X_1 = x_1, X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n$. The maximum likelihood estimate of θ is the value that maximizes the likelihood function, $L(x_1, x_2, \ldots, x_n\ ;\ \theta)$



  - We need to the likelihood function. What if there are many variables and it has a complex functional form?

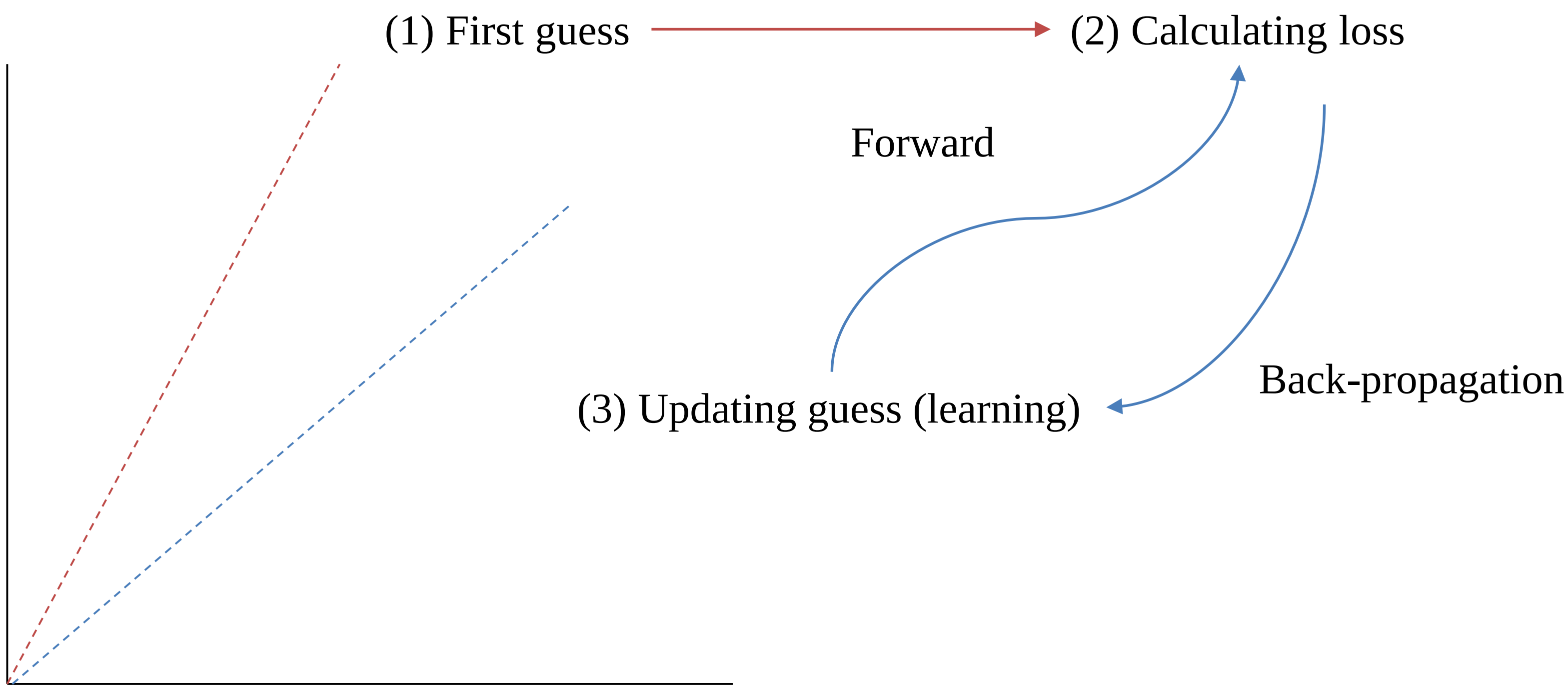
Statistical Style versus Machine Learning Style

- Machine learning is to gradually train the model to minimize a loss function
  - (1) Loss function → Mean squared error (MSE) for linear regression
  - (2) Optimization → Gradient descent for back-propagation
  - (3) Learning rate



Statistical Style versus Machine Learning Style

- Forward and back-propagation

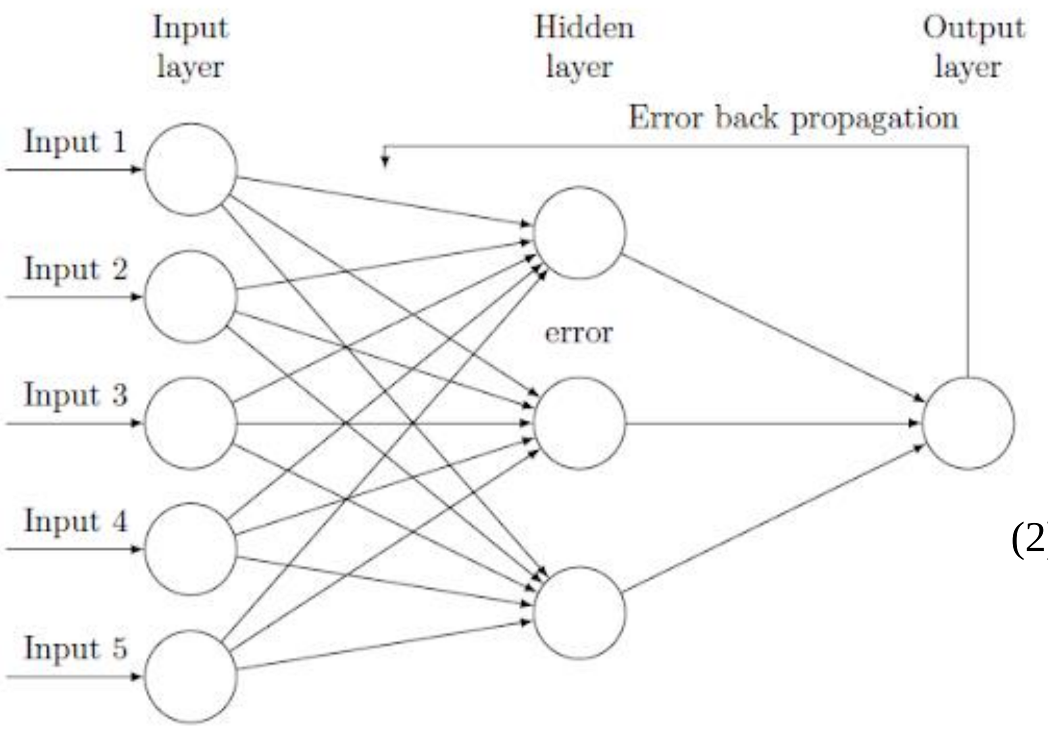


Statistical Style versus Machine Learning Style

- Forward and back-propagation in neural networks

Forward

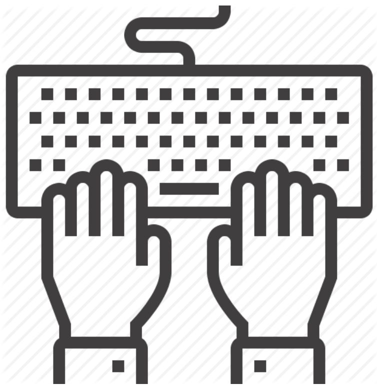
(1) First guess



(2) Calculating loss

Back-propagation

(3) Updating guess (learning)



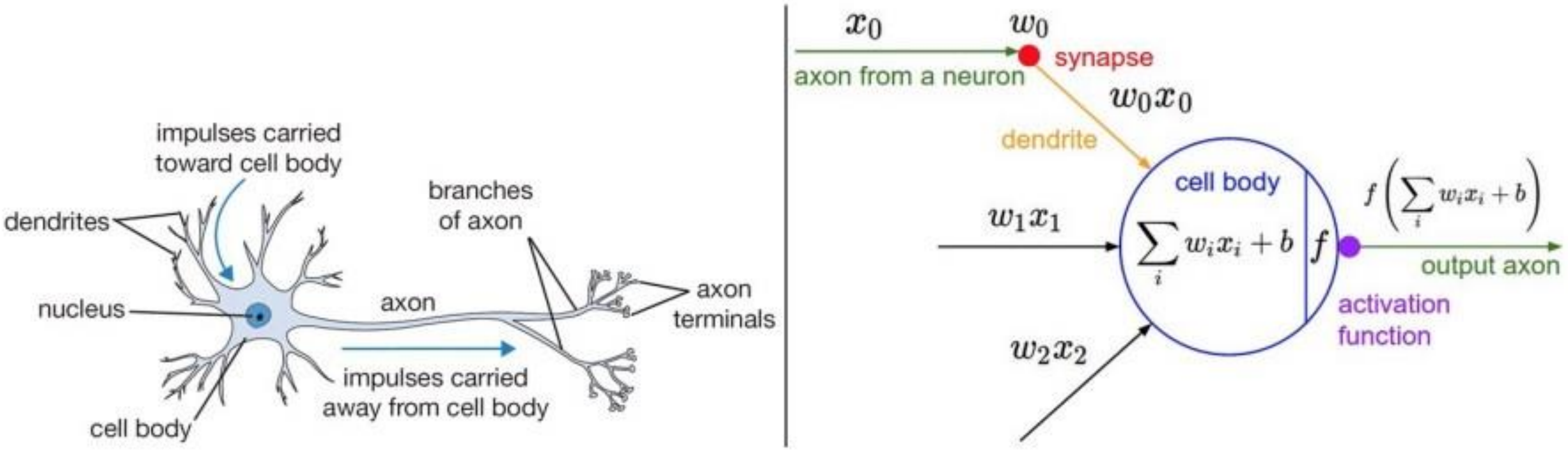
Linear Regression

M3.4 Linear Regression.ipynb

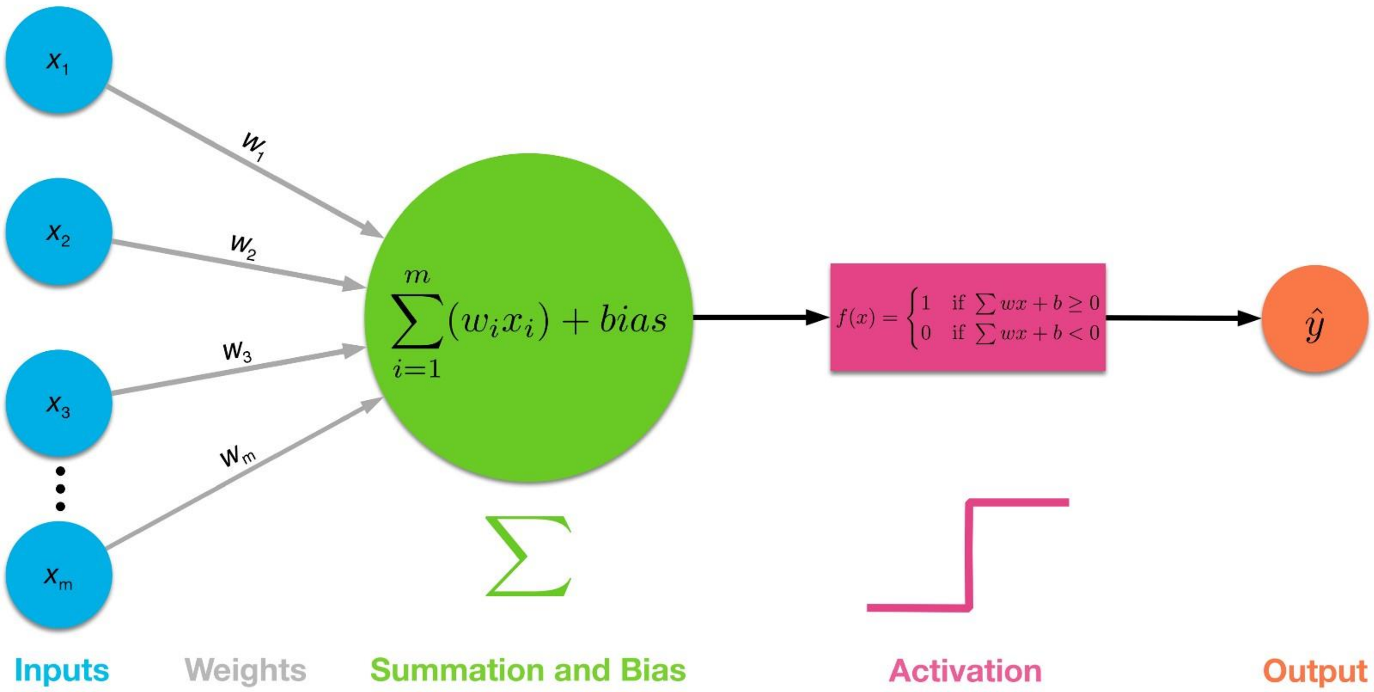
Neural Network

Neural Networks

- Artificial neural network mimics the human brain
  - ➢ Single neuron is activated (=1) or not (=0).



Linear Model + Activation = Neuron

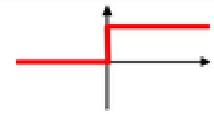
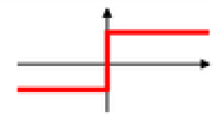
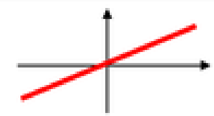
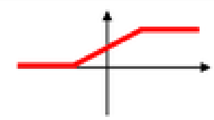
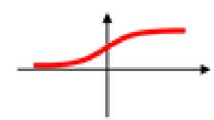
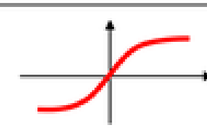


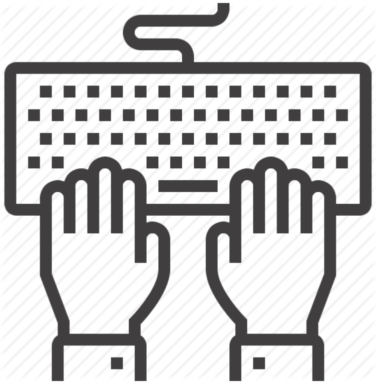
Linear Model

Activation Function

Linear Model + Activation = Neuron

- Activation function

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

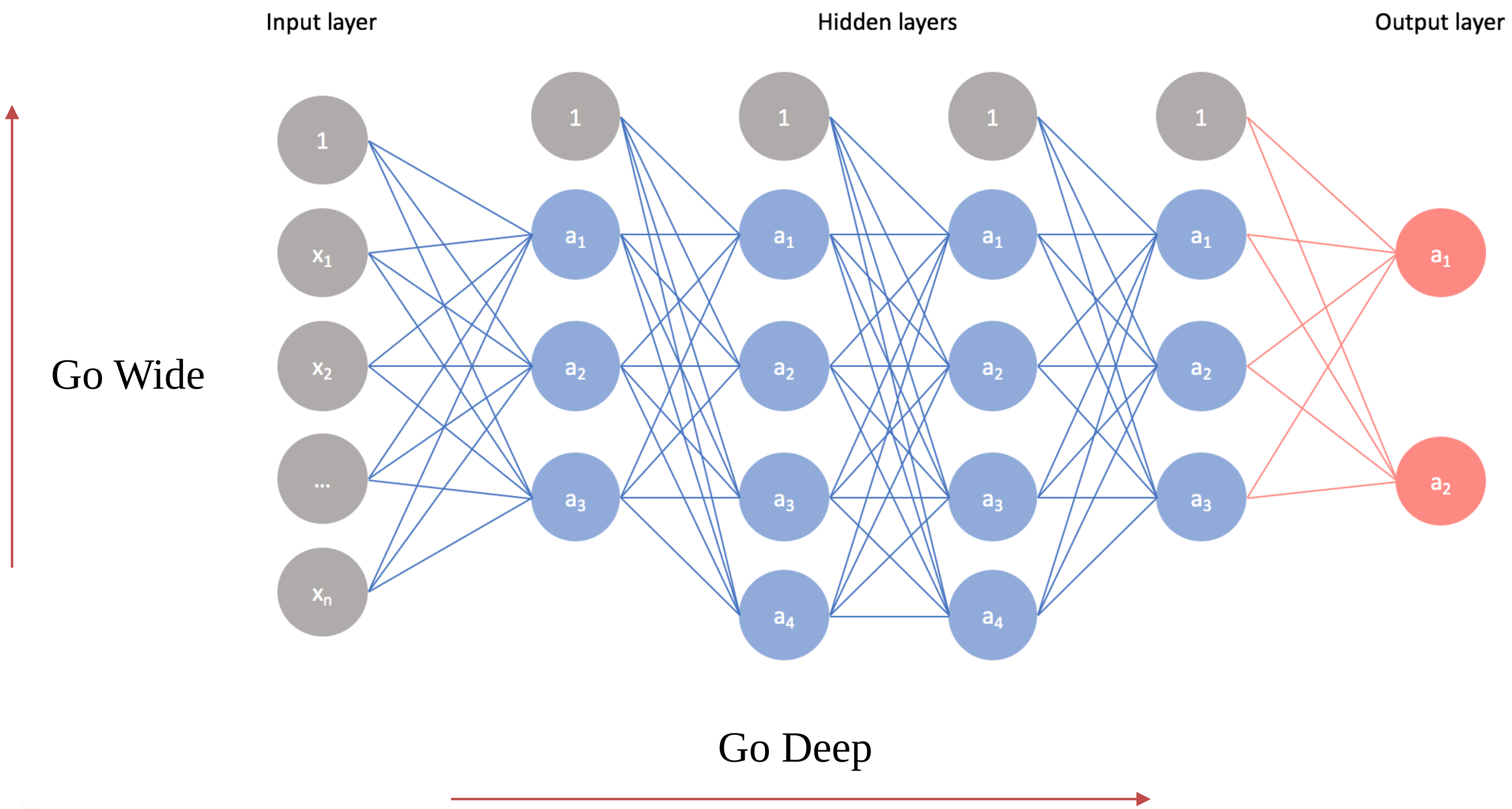


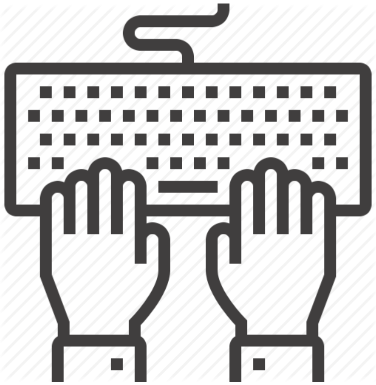
Neural Network

M3.4 Neural Network.ipynb

Go Wide & Go Deep

- Neural networks consist of multiple neurons at multiple layers



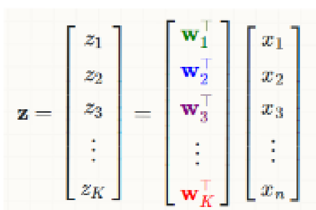


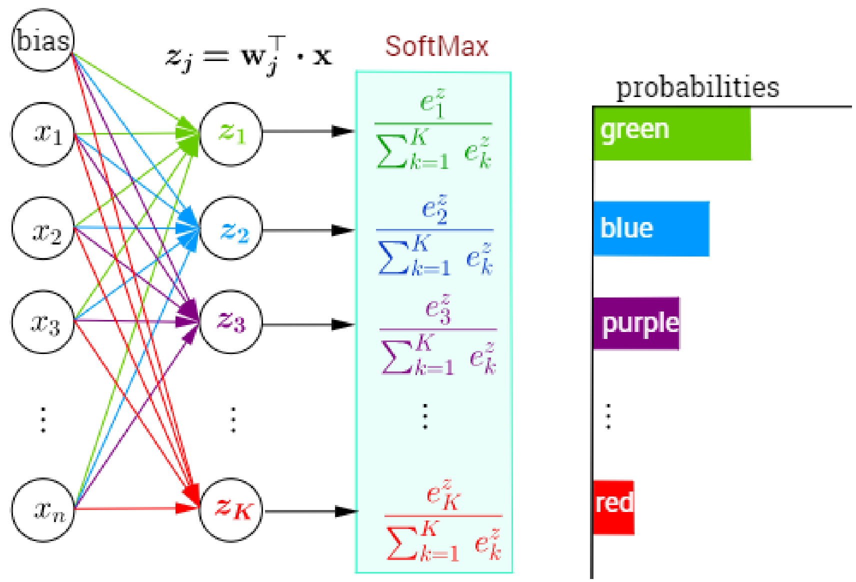
Neural Network

M3.4 Neural Network.ipynb

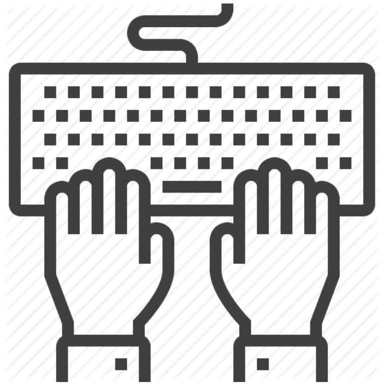
Multi-class Output

- Softmax function
  - ➢ $softmax\ (neuron\ j) = \frac{e^{z_j}}{\sum_K e^{z_k}}$





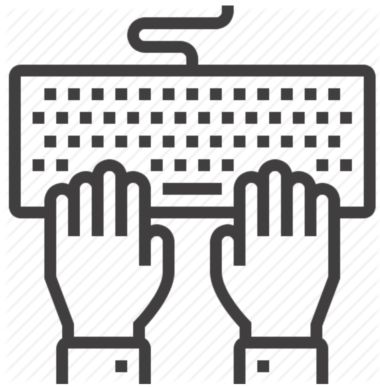
- Cross-entropy loss function for multi-class classification
  - ➢ Cross-entropy loss function = $-\sum_K \log\{softmax\ (neuron\ j)\}$



Neural Network

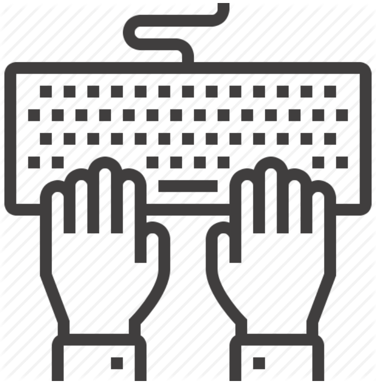
M3.4 Neural Network.ipynb

Projects for Neural Networks



MNIST Classifier using Neural Network

M3.4 Neural Network_MNIST.ipynb



CIFAR10 Classifier using Neural Network

M3.4 Neural Network_CIFAR10.ipynb

End of Document