Introduction to Deep Learning

Taewook Ko

SCONE Lab.

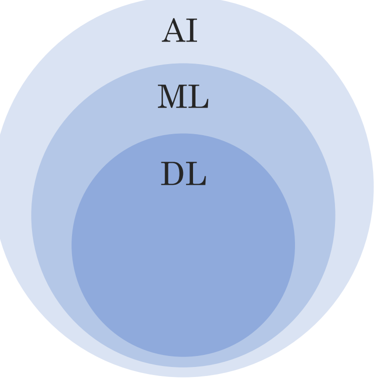
Contents

SCONE Lab.

Neural Network

SCONE Lab.

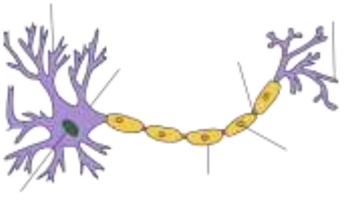
- Taxonomy
  - Artificial Intelligence (AI)
    - Anything automatically working
  - Machine Learning (ML)
    - Models with parameters
    - Parameter train (learning)
      - Logistic Regression
      - Support Vector Machine
      - Decision Tree
  - Deep Learning (DL)
    - **Neural Network**
    - Staking several layers
    - Huge number of parameters to train
      - GPT-3 175 billion



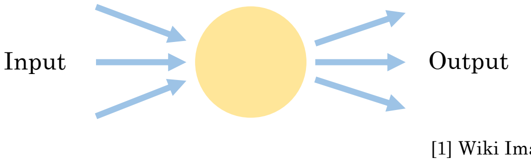
Neural Network

SCONE Lab.

- What is neural network?
  - Neuron[1]



  - Artificially mimic neuron process
    - Perceptron

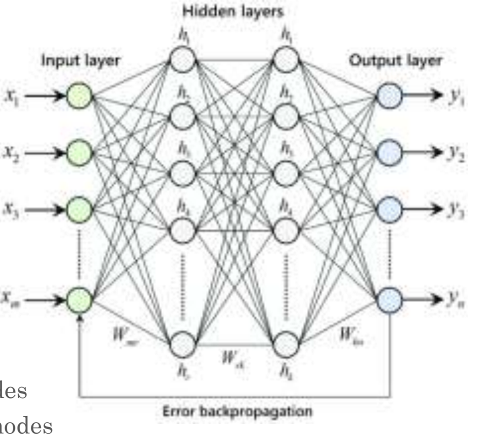


[1] Wiki Image, en.wikipedia.org/wiki

Neural Network

SCONE Lab.

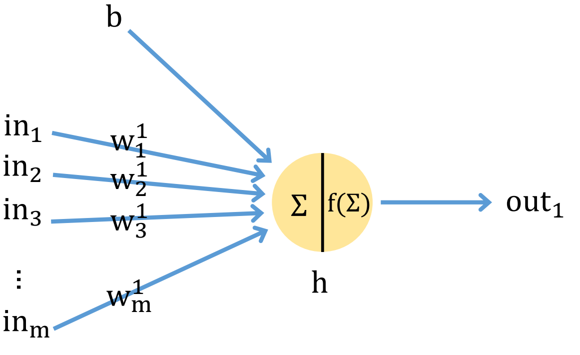
- What is neural network?
  - Neural Network Components
    - Neuron
    - Connection
    - Input layer (data)
    - Output layer (prediction)
    - Hidden layer
    - Parameters
      - Weights
      - Bias
  - Two hidden layer NN
    - First hidden layer with r nodes
    - Second hidden layer with k nodes
  - Output layer with n nodes



Neural Network

SCONE Lab.

- How the neural network work?
  - For a single neuron
  - $output = f(\sum_x x_i w_i + b)$

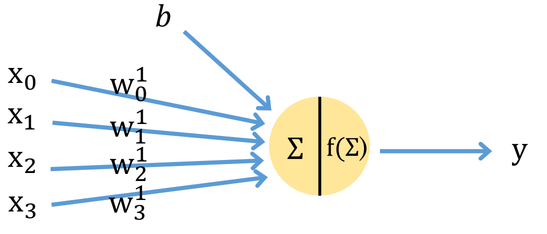


Neural Network

SCONE Lab.

- How the neural network work?
  - Simple Example: A coffee menu classifier
    - input = [espresso, water, milk, ice]
    - output = ice americano: 0, americano: 1, ice latte: 2, latte:3

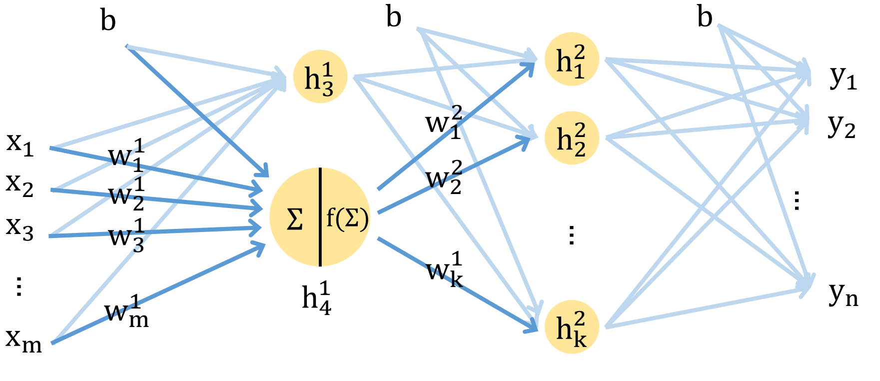
    - input = [1,1,0,1] → output = f($w_0 + w_1 + w_3 + b$) = 0
    - input = [1,0,1,0] → output = f($w_0 + w_2 + b$) = 3



Neural Network

SCONE Lab.

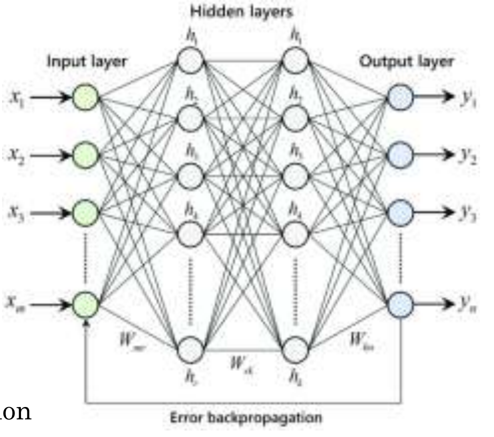
- How the neural network work?
  - Output value is another input for next layer neuron
  - $output = f(\sum_x x_i w_i + b)$



Neural Network

SCONE Lab.

- Forward Propagation
  - $Input = X \in \mathbb{R}^{1 \times m}$
    - m dimensional input
  - $H^1 = f(XW^1 + b^1)$
    - $W^1 \in \mathbb{R}^{m \times r}$, $b^1 \in \mathbb{R}^{1 \times r}$
  - $H^2 = f(H^1W^2 + b^2)$
    - $W^2 \in \mathbb{R}^{r \times k}$, $b^2 \in \mathbb{R}^{1 \times k}$
  - $Output = f(H^2W^O + b^O)$
    - $W^O \in \mathbb{R}^{k \times n}$, $b^O \in \mathbb{R}^{1 \times n}$

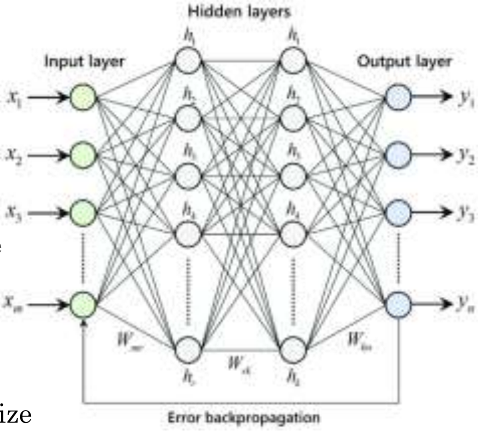


Repeating Matrix multiplication

Neural Network

SCONE Lab.

- Notations
  - W^l: weight matrix of l-th layer
    - $W^l \in \mathbb{R}^{d_1 \times d_2}$
    - d_i :# of layer nodes
  - b^l: bias vector of l-th layer
    - $b^1 \in \mathbb{R}^{1 \times d2}$
  - H^l: hidden representation
    - $H^l \in \mathbb{R}^{1 \times d_2}$
  - $Output$: desire output shape
    - Prediction value
    - Percentage

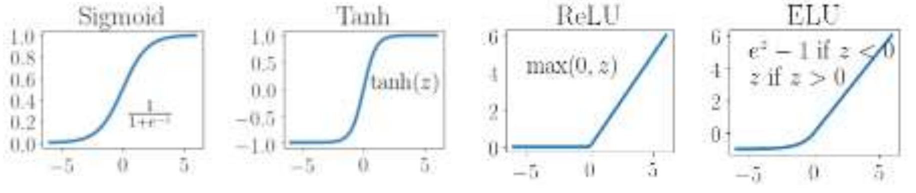


# of layer nodes = dimension size

Neural Network

SCONE Lab.

- Activation Function
  - $output = f(\sum_i x_i w_i + b)$
  - Neural networks are the process of repeating matrix multiplication
  - No difference from linear algebraic models
    - Linear regression / SVM
  - Activation function is the key which makes the difference!
    - Non-linear function
    - Gives non-linearity characteristic to the model



[2] Johnson, N. S., et al. 2020

Parameter Train

SCONE Lab.

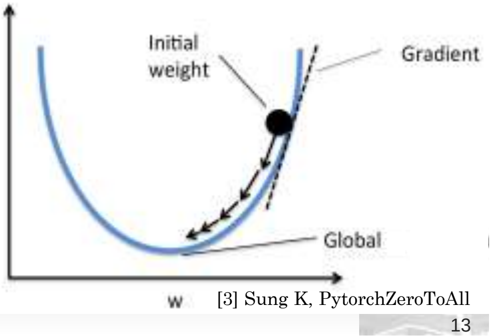
- Neural network
  - Input features are fed into the neural network
  - Get the output after forward propagation
  - Output should be similar to the ground-truth
    - ex. Cat and Dog image classification
    - Dog → forward propagation → Dog: 99% Cat: 1%
    - Cat → forward propagation → Dog: 2% Cat: 98%

- Train the neural network parameters $\theta(W^l, b^l)$
  - To make proper output

Parameter Train

SCONE Lab.

- How to train the parameters
  - Gradient Decent algorithm [studied in calculus class]
  - Loss function
    - $Loss = (\hat{y} - y)^2 = (x * w + b - y)^2$
  - Want to minimize the loss
    - Find the global minimum value of the loss function
  - Derivate on parameters
    - Gradient $\frac{\partial loss}{\partial w}$
      - Direction to reducing loss



[3] Sung K, PytorchZeroToAll

Parameter Train

SCONE Lab.

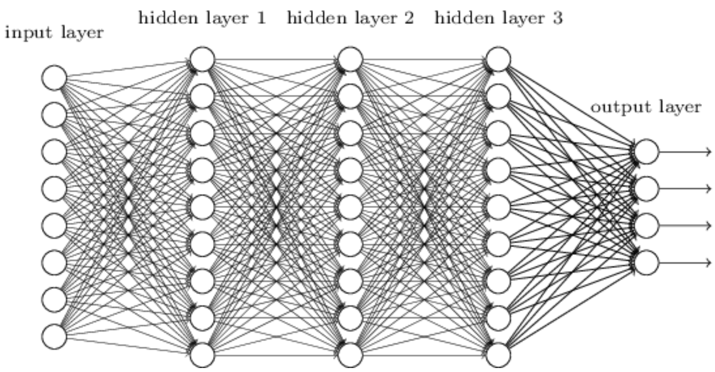
- Update Rule
  - $w_i := w_i - \alpha \frac{\partial loss}{\partial w_i}$
  - w_i : parameter
  - $\frac{\partial loss}{\partial w_i}$: gradient on parameter w_i
  - α : learning rate, learning step

  - Expected to get smaller loss with newly update parameter w_i
  - Update the parameter to the direction to reduce loss
  - Loss is a function of parameters (Outcome of forward pass)

Parameter Train

SCONE Lab.

- Chain Rule
  - Hundreds, millions of parameters contributes the loss function
  - Need to calculate gradient of each parameters
  - Use chain rule

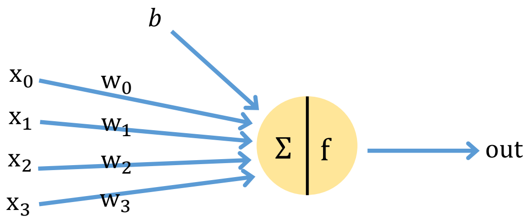


Parameter Train

SCONE Lab.

- Chain Rule
  - $f = f(g), \quad g = g(x)$
  - $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \times \frac{\partial g}{\partial x}$
  - $\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial out} \times \frac{\partial out}{\partial in} \times \frac{\partial in}{\partial w_0}$
  - $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial out} \times \frac{\partial out}{\partial in} \times \frac{\partial in}{\partial w_1}$
  - $\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial out} \times \frac{\partial out}{\partial in} \times \frac{\partial in}{\partial w_2}$
  - $\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial out} \times \frac{\partial out}{\partial in} \times \frac{\partial in}{\partial w_3}$

    $= 1 \times f'(out) \times x_3$
  - $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial out} \times \frac{\partial out}{\partial in} \times \frac{\partial in}{\partial b}$



$in = \sum x_i w_i + b$

$out = f(\sum_x x_i w_i + b)$

$L = out - y$

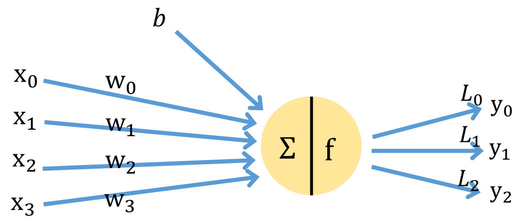
Parameter Train

SCONE Lab.

- Chain Rule
  - $f = f(g), \quad g = g(x)$
  - $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \times \frac{\partial g}{\partial x}$

  - $\frac{\partial L_0}{\partial w_0} = \frac{\partial L_0}{\partial \text{out}} \times \frac{\partial \text{out}}{\partial \text{in}} \times \frac{\partial \text{in}}{\partial w_0}$
  - $\frac{\partial L_1}{\partial w_0} = \frac{\partial L_1}{\partial \text{out}} \times \frac{\partial \text{out}}{\partial \text{in}} \times \frac{\partial \text{in}}{\partial w_0}$
  - $\frac{\partial L_2}{\partial w_0} = \frac{\partial L_2}{\partial \text{out}} \times \frac{\partial \text{out}}{\partial \text{in}} \times \frac{\partial \text{in}}{\partial w_0}$

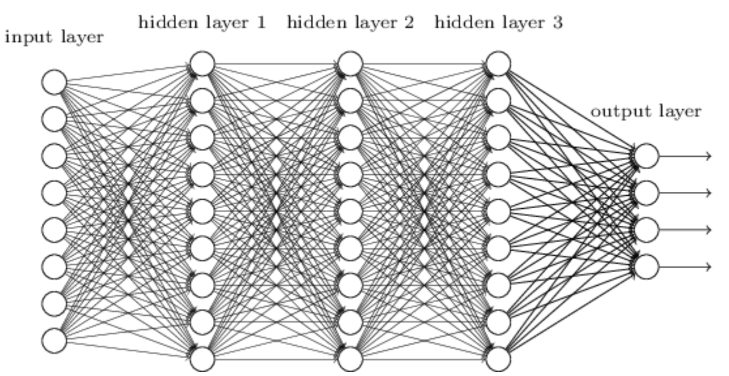
  - $\frac{\partial L}{\partial w_0} = \frac{\partial L_0}{\partial w_0} + \frac{\partial L_1}{\partial w_0} + \frac{\partial L_2}{\partial w_0}$
  - $w_0 = w_0 - \alpha \frac{\partial L}{\partial w_0}$



Parameter Train

SCONE Lab.

- Chain Rule
  - Hundreds, millions of parameters contributes the loss function
  - Need to calculate gradient for each parameters
  - Use chain rule



Parameter Train

SCONE Lab.

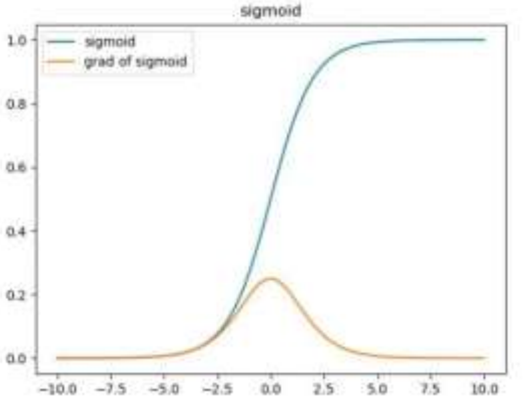
- Back Propagation
  - Calculate loss (prediction, ground-truth)
  - Calculate gradients with chain rule
  - Updated parameters with updating rule

- Gradient Vanishing problem
  - Chain rule
    - Repeatedly multiply gradients
    - Gradients are small values
    - For deep layer, gradients will be very small
      - $\frac{\partial y}{\partial x_1} = \frac{\partial f}{\partial x_l} \times \frac{\partial x_l}{\partial x_{l-1}} \times \frac{\partial x_{l-1}}{\partial x_{l-2}} \times \frac{\partial x_{l-2}}{\partial x_{l-3}} \times \ldots \times \frac{\partial x_2}{\partial x_1}$
      - *There is no parameter update and training for deep layer NN*
  - This neural network idea was proposed in 80's
    - The gradient vanishing issue brought AI winter

Parameter Train

SCONE Lab.

- Activation Functions
  - Sigmoid
    - Maximum derivative value of sigmoid is less than 1
    - $\sigma(x) = \frac{1}{1+e^{-x}}$
    - $\sigma'(x) = \sigma(x)(1- \sigma(x))$

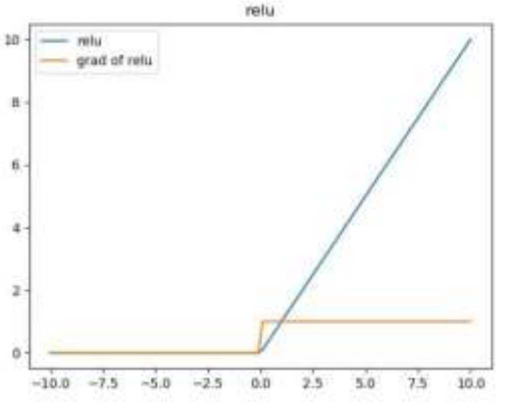


Parameter Train

SCONE Lab.

- Activation Functions
  - ReLU
    - $R(x) = \max(0, x)$
    - $R'(x) = 0 \; or \; 1$

Gradients are not drastically reduced
Large values can get gradients

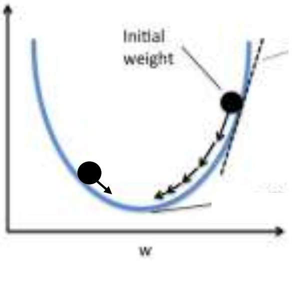


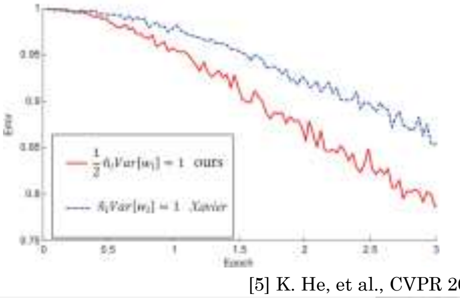
[4] N. Vinod and G. Hinton, ICML. 2010.

Parameter Train

SCONE Lab.

- Initialization
  - Initializing parameters
    - Start with small number sampled from gaussian distribution
  - Things to read
    - Xavier Weight Initialization [Xavier et al, ICML2010]
    - Normalized Weight Initialization [Xavier et al, ICML2010]
    - He Weight Initialization [He et al, CVPR2015]



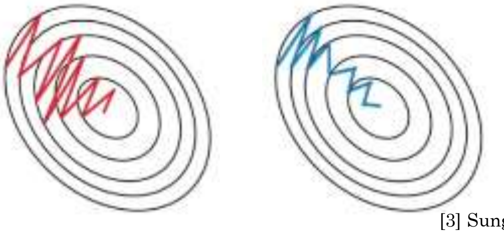


[5] K. He, et al., CVPR 2015

Parameter Train

SCONE Lab.

- Update Rule
  - Stochastic Gradient Decent
    - Cannot load all training dataset at once
    - Train with some batch of train data (Called mini-batch learning)
    - Calculated gradients for batch data
      - It is not the exact gradient to the global minimum
    - Momentum update
      - $W = W - \alpha v_w$
      - $v_{dw} = \beta v_{dw} + (1 - \beta)dW$

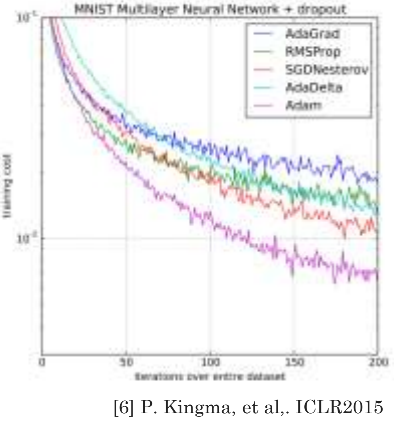


[3] Sung K, PytorchZeroToAll

Parameter Train

SCONE Lab.

- Update Rule
  - Things to read
    - RMSProp
      - Exponentially weighted moving average
    - AdaGrad [JMLR2011]
      - Change learning rate
    - ADAM [ICLR2015]
      - RMSProp + AdaGrad



[6] P. Kingma, et al,. ICLR2015

Parameter Train

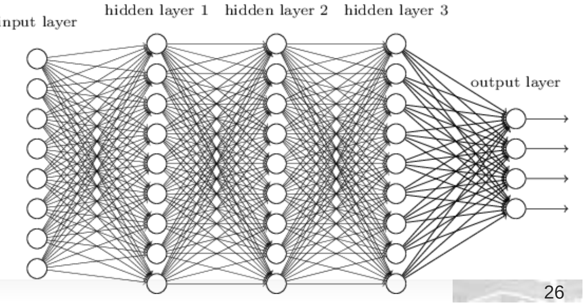
SCONE Lab.

- Loss Functions
  - Mean squared error
    - $L = (\hat{y}_i - y_i)^2$
  - Mean absolute error
    - $L = |\hat{y}_i - y_i|$
  - Binary Cross-Entropy
    - $L = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - y_i))$
  - Cross-Entropy
    - $L = y_i log(\hat{y}_i)$
  - Hinge Loss
    - $L = \max(0, \mathrm{y} - \hat{\mathrm{y}} + 1)$

Parameter Train

SCONE Lab.

- Neural Network overview
  - Network design
    - Input, output
    - Layer, node
  - Initialize parameters
    - Initializing
  - Forward Propagation
    - Activation functions
    - Normalization
    - Regularization
  - Calculate loss
    - Loss functions
  - Back Propagation
    - Update rule
    - Learning rate



Reference

SCONE Lab.

[1] https://en.wikipedia.org/wiki/Neuron

[2] Johnson, N. S., et al. "Invited review: Machine learning for materials developments in metals additive manufacturing." *Additive Manufacturing* 36 (2020): 101641.

[3] https://github.com/hunkim/PyTorchZeroToAll

[4] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *Icml*. 2010.

[5] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*. 2015.

[6] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).