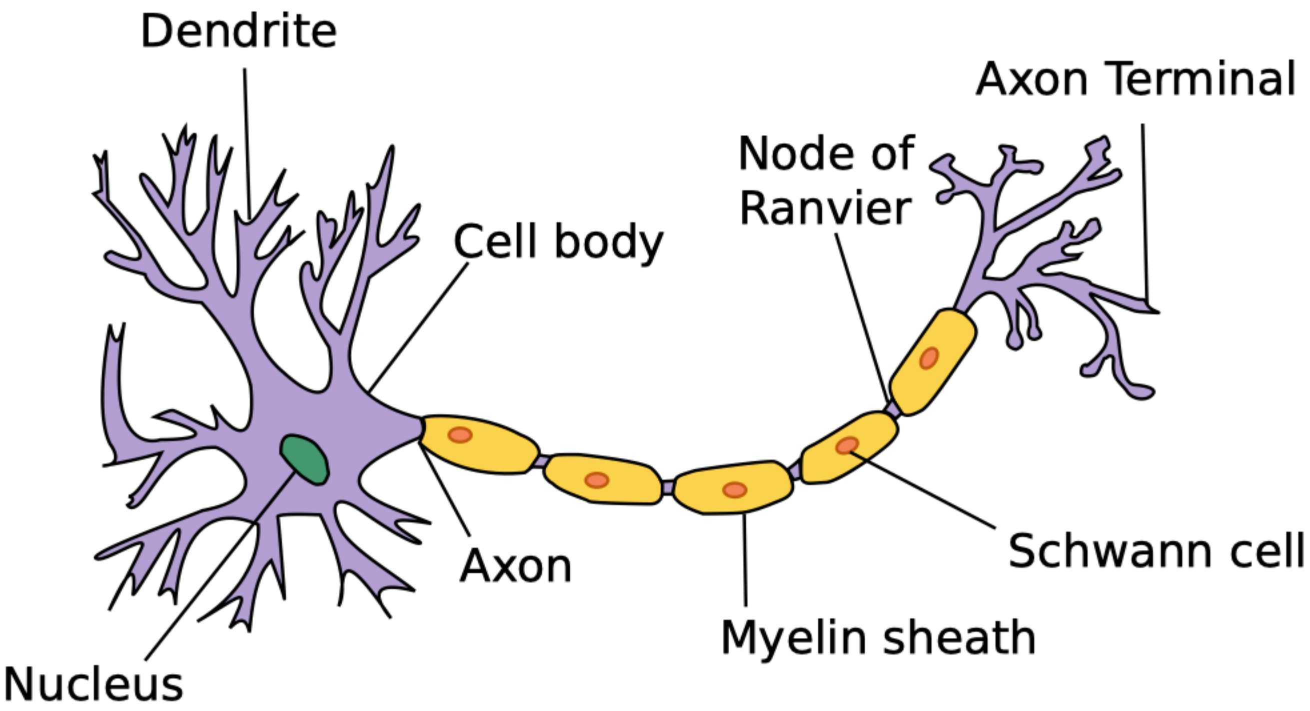
Deep Learning Basics

Lecture 2: Neural Networks & Multi-Layer Perceptron

최성준 (고려대학교 인공지능학과)

Neural Networks

- "Neural networks are computing systems vaguely inspired by the biological neural networks that constitute animal brains."



Neural Networks

- "Neural networks are computing systems vaguely inspired by the biological neural networks that constitute animal brains."

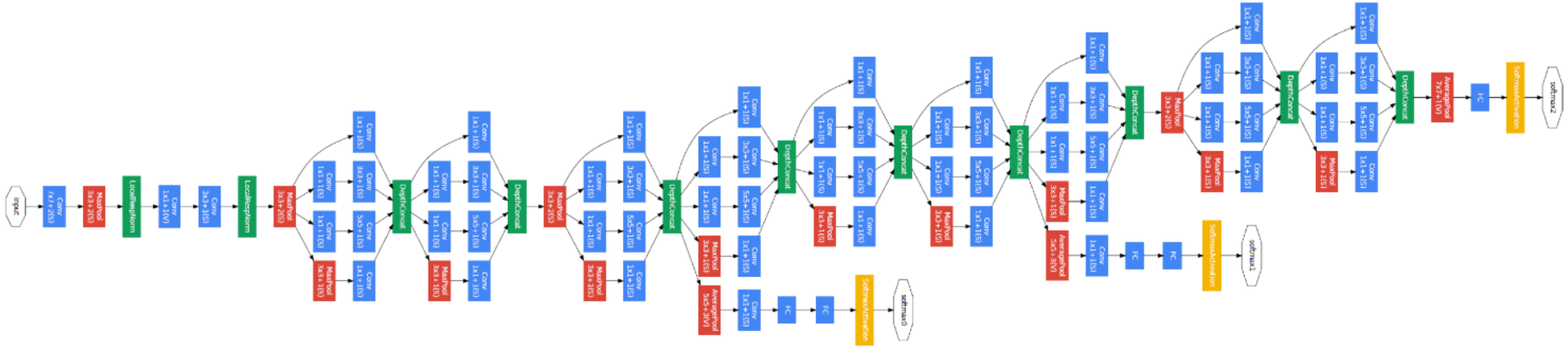
Clément Ader's **Avion III** (1897)

Write Brothers (1903)

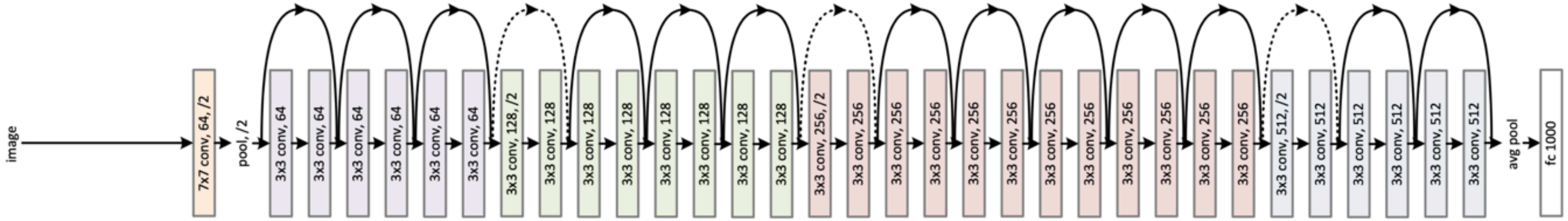
F-22 Raptor

Neural Networks

- Neural networks are function approximators that stack affine transformations followed by nonlinear transformations.



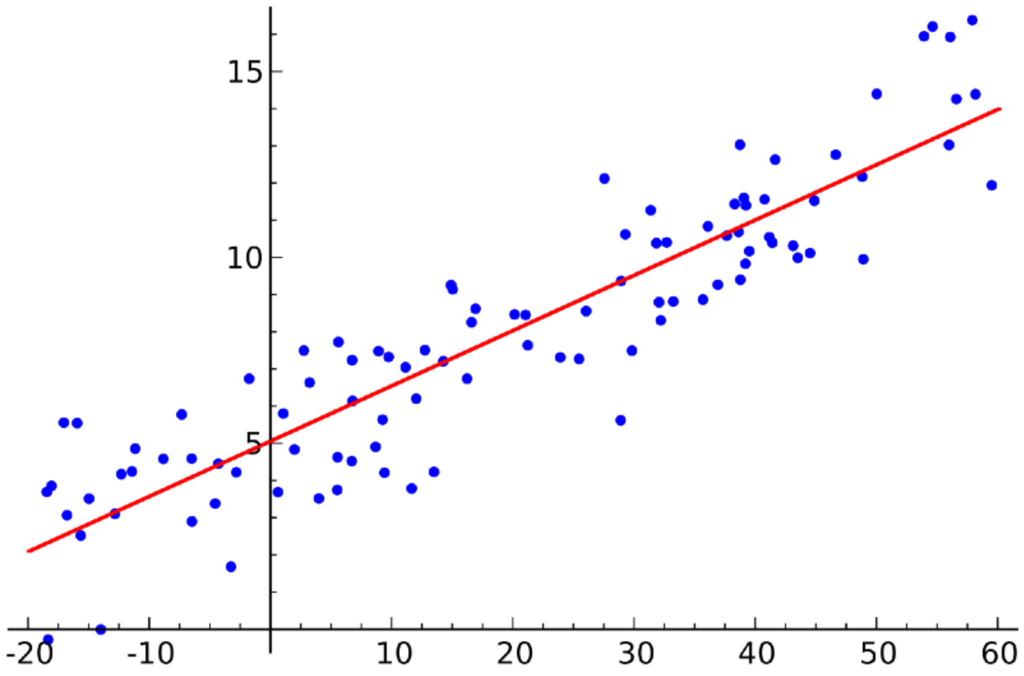
GoogLeNet



ResNet

Linear Neural Networks

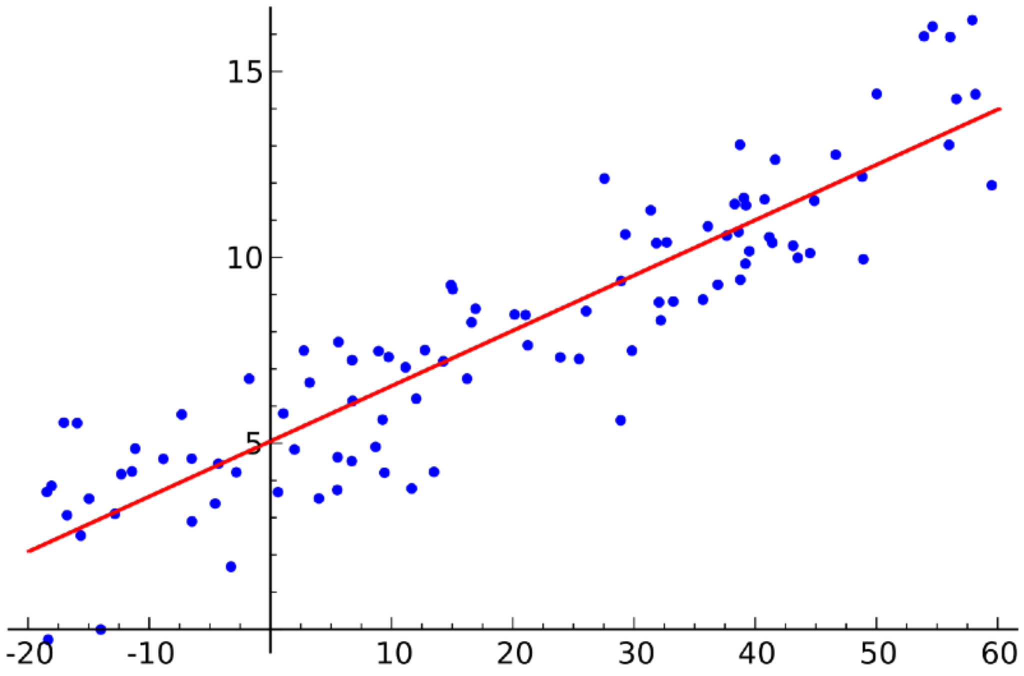
- Let's start with the most simple example.



- Data: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
- Model: $\hat{y} = wx + b$
- Loss: $\text{loss} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

Linear Neural Networks

- We compute the partial derivatives w.r.t. the optimization variables.



Partial derivative

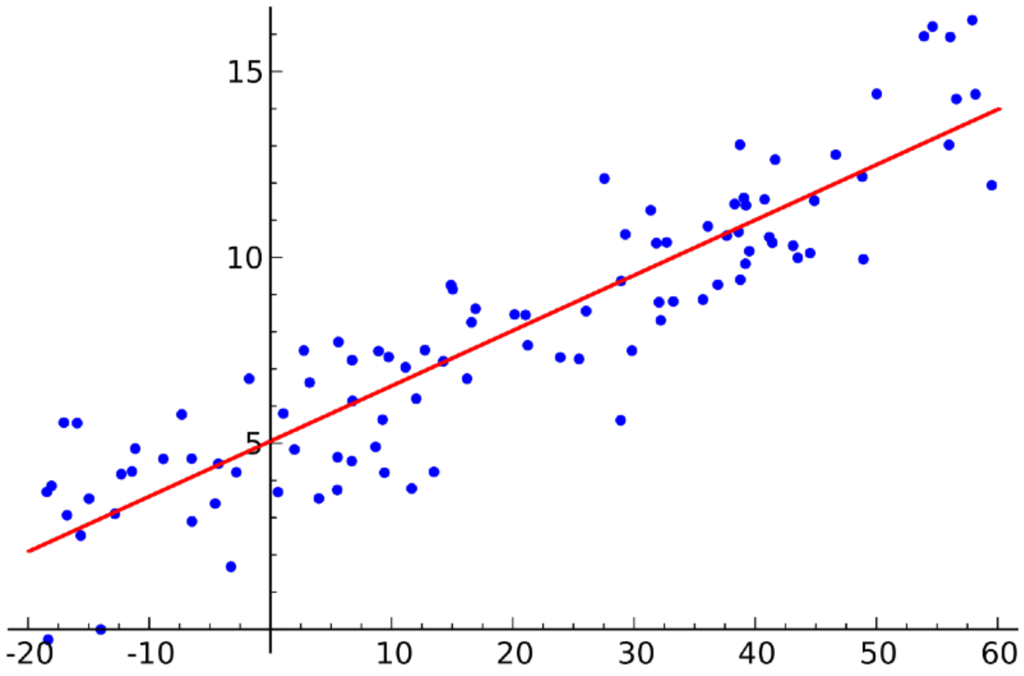
$$\boxed{\frac{\partial \text{loss}}{\partial w}} = \frac{\partial}{\partial w}\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= \frac{\partial}{\partial w}\frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i - b)^2$$

$$= -\frac{1}{N}\sum_{i=1}^{N}-2(y_i - wx_i - b)x_i$$

Linear Neural Networks

- We compute the partial derivatives w.r.t. the optimization variables.



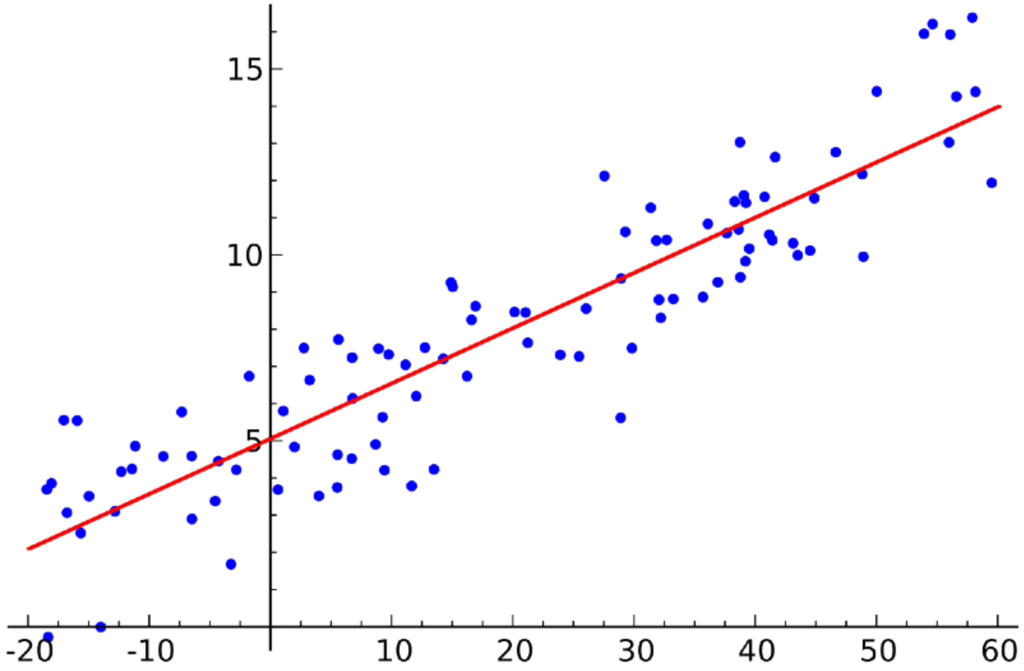
$$\frac{\partial \text{loss}}{\partial b} = \frac{\partial}{\partial b}\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= \frac{\partial}{\partial b}\frac{1}{N}\sum_{i=1}^{N}(y_i - wx_i - b)^2$$

$$= -\frac{1}{N}\sum_{i=1}^{N}-2(y_i - wx_i - b)$$

Linear Neural Networks

- Then, we iteratively update the optimization variables.



Update

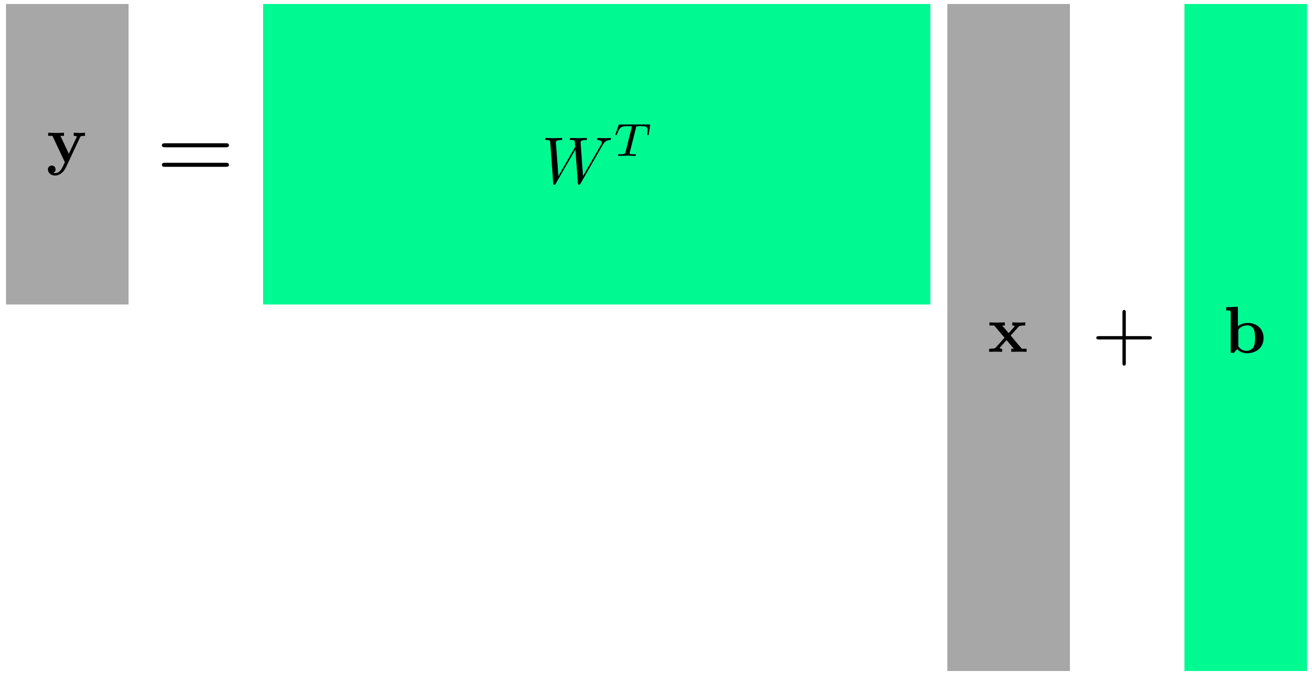
Stepsize

$$\boxed{w \leftarrow} w - \boxed{\eta} \frac{\partial \text{loss}}{\partial w}$$

$$\boxed{b \leftarrow} b - \boxed{\eta} \frac{\partial \text{loss}}{\partial b}$$

Linear Neural Networks

- Of course, we can handle multi dimensional input and output.



$$\mathbf{y} = W^T\mathbf{x} + \mathbf{b}$$

Linear Neural Networks

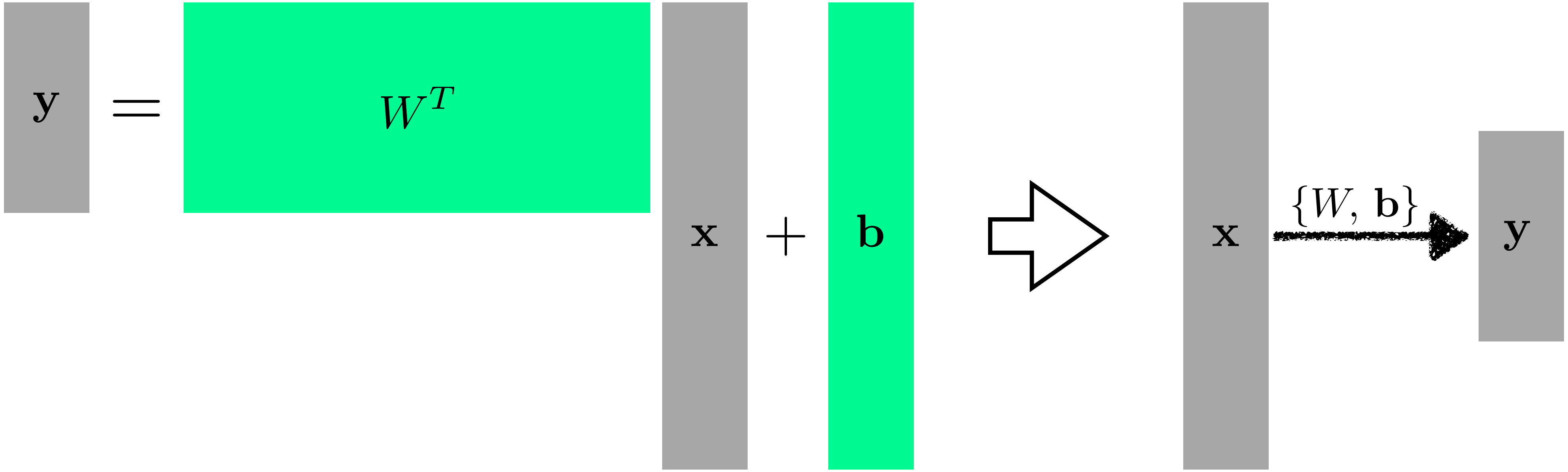
- Of course, we can handle multi dimensional input and output.

$$\mathbf{y} = W^T \mathbf{x} + \mathbf{b}$$

One way of interpreting a matrix is to regard it as a mapping between two vector spaces.

Linear Neural Networks

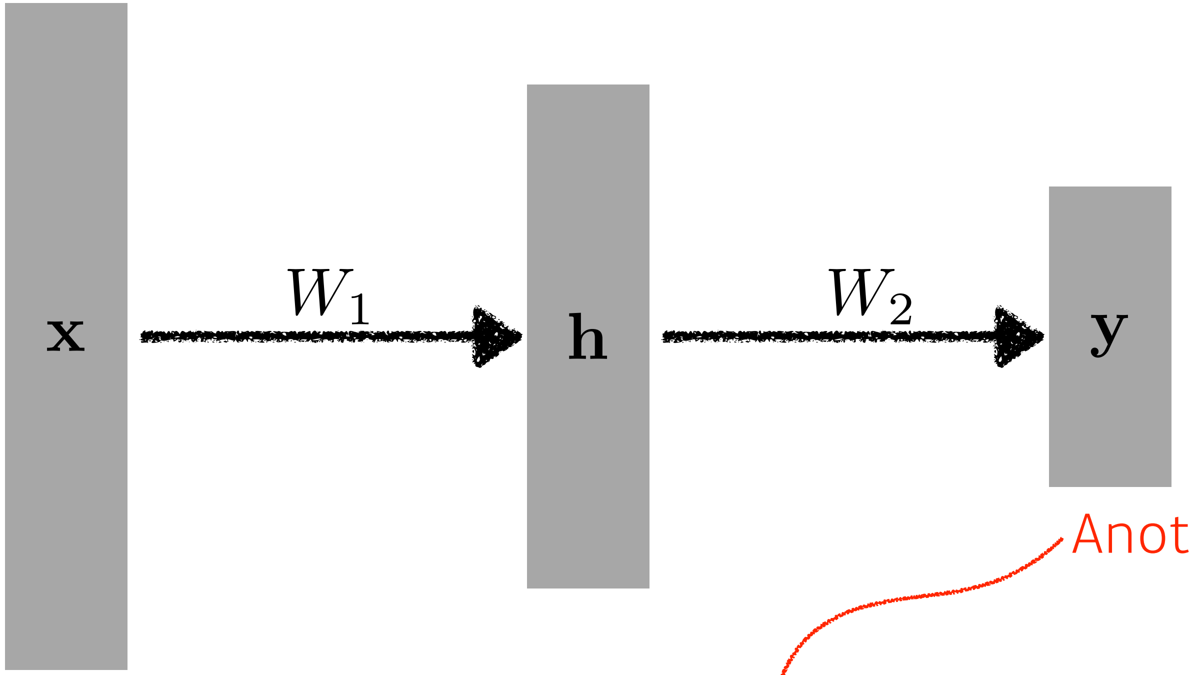
- Of course, we can handle multi dimensional input and output.



One way of interpreting a matrix is to regard it as a mapping between two vector spaces.

Beyond Linear Neural Networks

- What if we stack more?

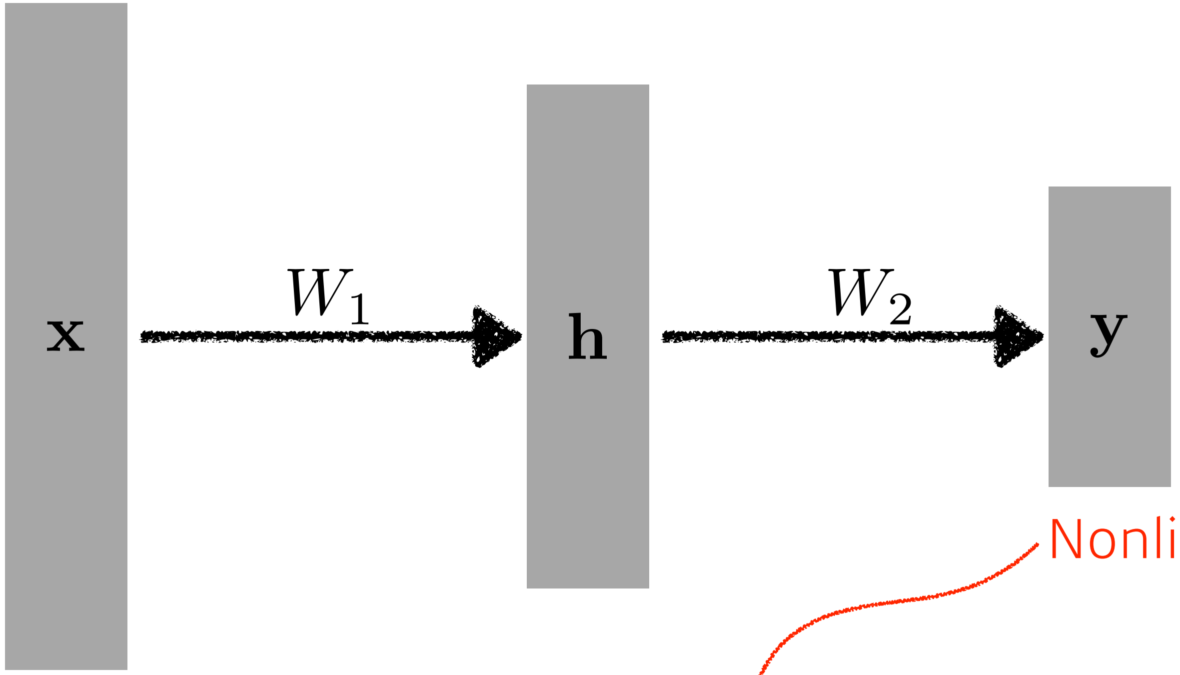


Another matrix

$$y = W_2^T \mathbf{h} = \boxed{W_2^T W_1^T} \mathbf{x}$$

Beyond Linear Neural Networks

- We need nonlinearity.



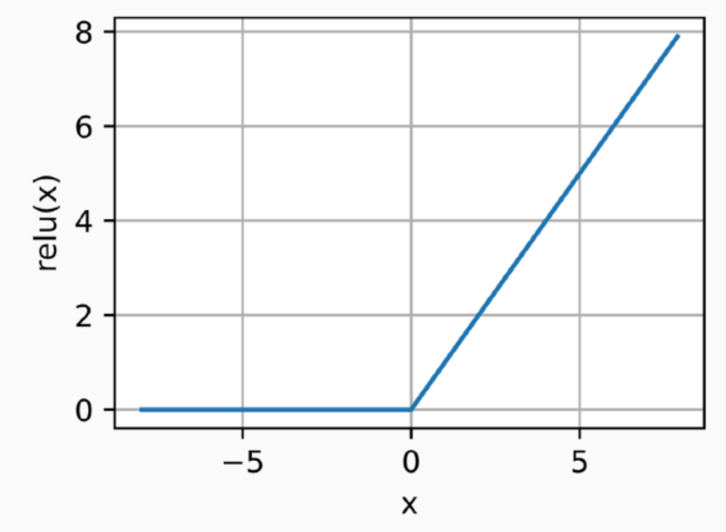
Nonlinear transform

$$y = W_2^T \mathbf{h} = W_2^T \boxed{\rho}(W_1^T \mathbf{x})$$

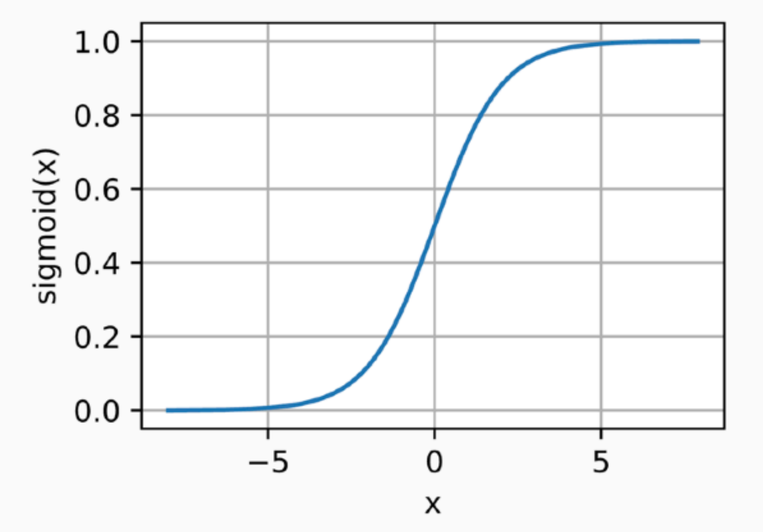
Beyond Linear Neural Networks

- Activation functions

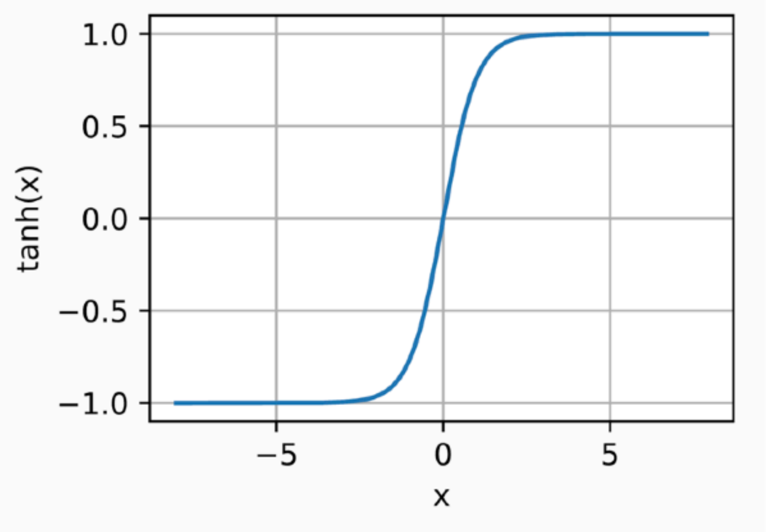
Rectified Linear Unit (ReLU)



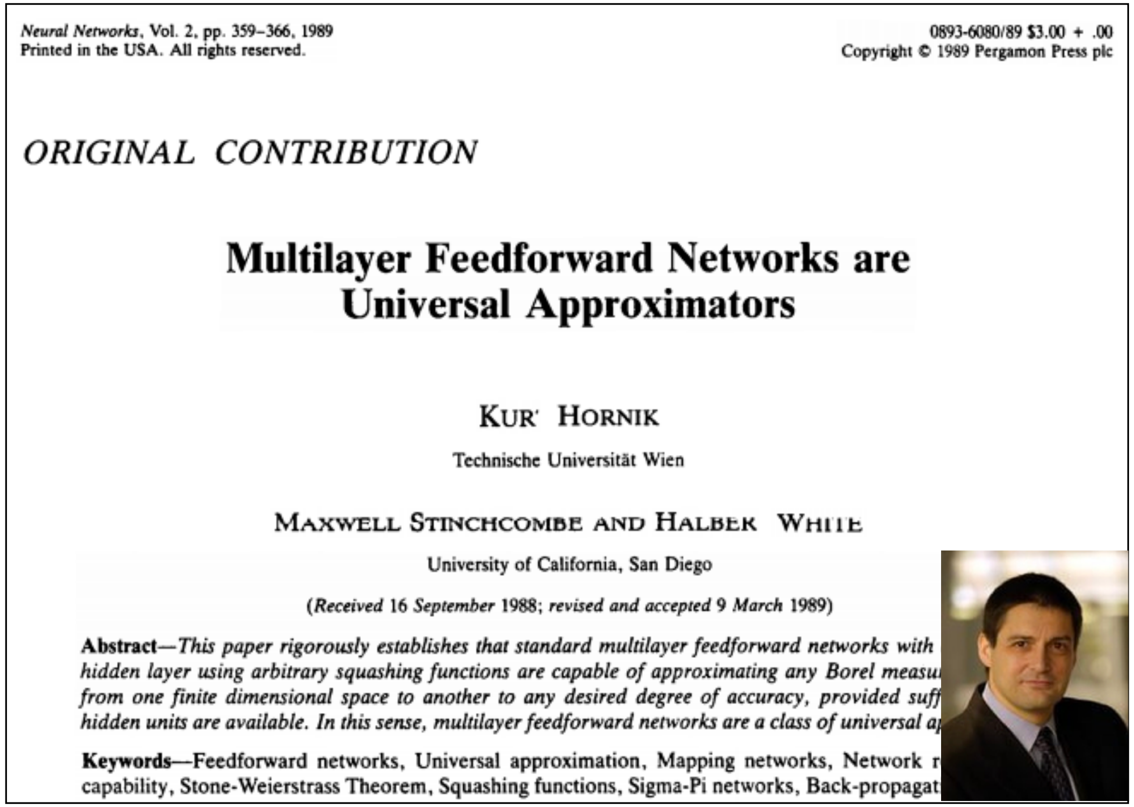
Sigmoid



Hyperbolic Tangent



Beyond Linear Neural Networks


Neural Networks, Vol. 2, pp. 359–366, 1989
Printed in the USA. All rights reserved.

0893-6080/89 $3.00 + .00
Copyright © 1989 Pergamon Press plc

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK
Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE
University of California, San Diego

(*Received 16 September 1988; revised and accepted 9 March 1989*)

Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with hidden layer using arbitrary squashing functions are capable of approximating any Borel measu from one finite dimensional space to another to any desired degree of accuracy, provided suff hidden units are available. In this sense, multilayer feedforward networks are a class of universal ap*

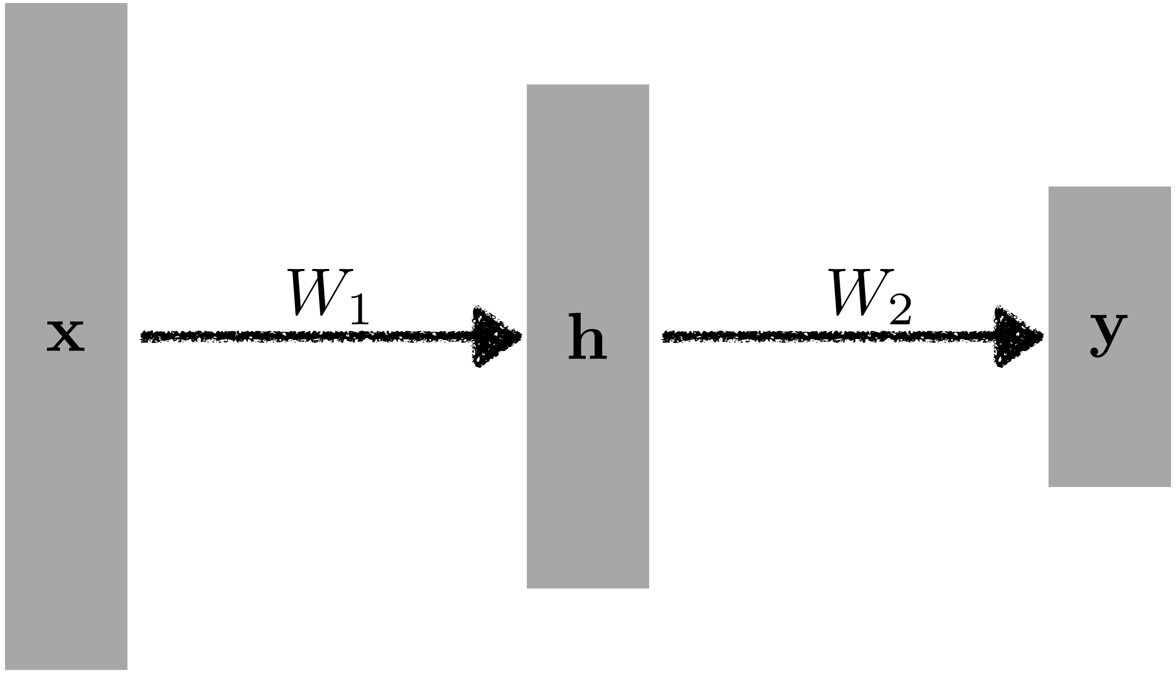
Keywords—Feedforward networks, Universal approximation, Mapping networks, Network r capability, Stone-Weierstrass Theorem, Squashing functions, Sigma-Pi networks, Back-propagat

English
There is a single hidden layer feedforward network that approximates any measurable function to any desired degree of accuracy on some compact set K.
Math
For every function g in M^r there is a compact subset K of R^r and an $f \in \sum^r(\Psi)$ such that for any $\epsilon > 0$ we have $\mu(K) < 1 - \epsilon$ and for every $X \in K$ we have $|f(x) - g(x)| < \epsilon$, regardless of Ψ, r, or μ.

Caution: It only guarantees the existence of such networks.

Multi-Layer Perceptron

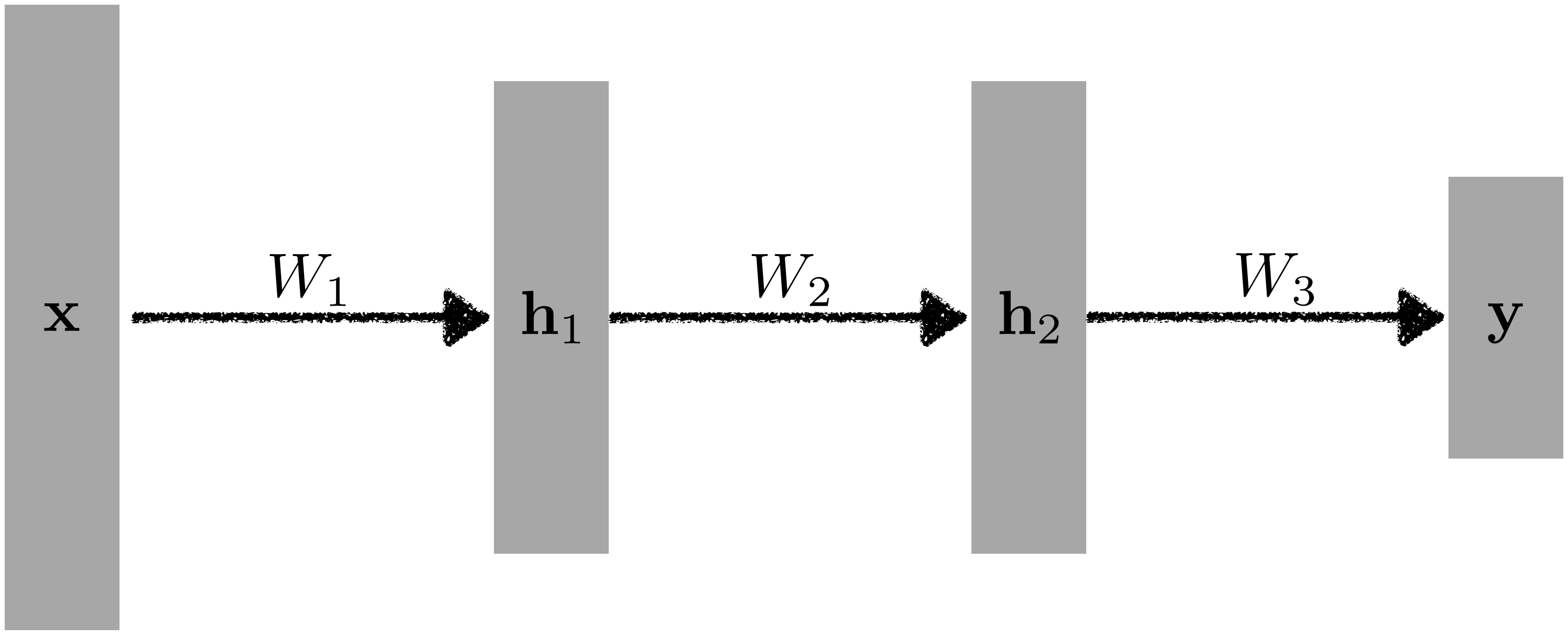
- This class of architectures are often called multi-layer perceptrons.



$$y = W_2^T \mathbf{h} = W_2^T \rho(W_1^T \mathbf{x})$$

Multi-Layer Perceptron

- Of course, it can go deeper.



$$y = W_3^T \mathbf{h}_2 = W_3^T \rho(W_2^T \mathbf{h}_1) = W_3^T \rho(W_2^T \rho(W_1^T \mathbf{x}))$$

Multi-Layer Perceptron

- What about the loss functions?

True target

Predicted output

Regression Task

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D}(\boxed{y_i^{(d)}} - \boxed{\hat{y}_i^{(d)}})^2$$

Classification Task

$$\text{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D} y_i^{(d)} \log \hat{y}_i^{(d)}$$

Probabilistic Task

$$\text{MLE} = \frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D} \log \mathcal{N}(y_i^{(d)}; \hat{y}_i^{(d)}, 1)$$ (=MSE)

Thank you for listening