

Deep Learning Basics

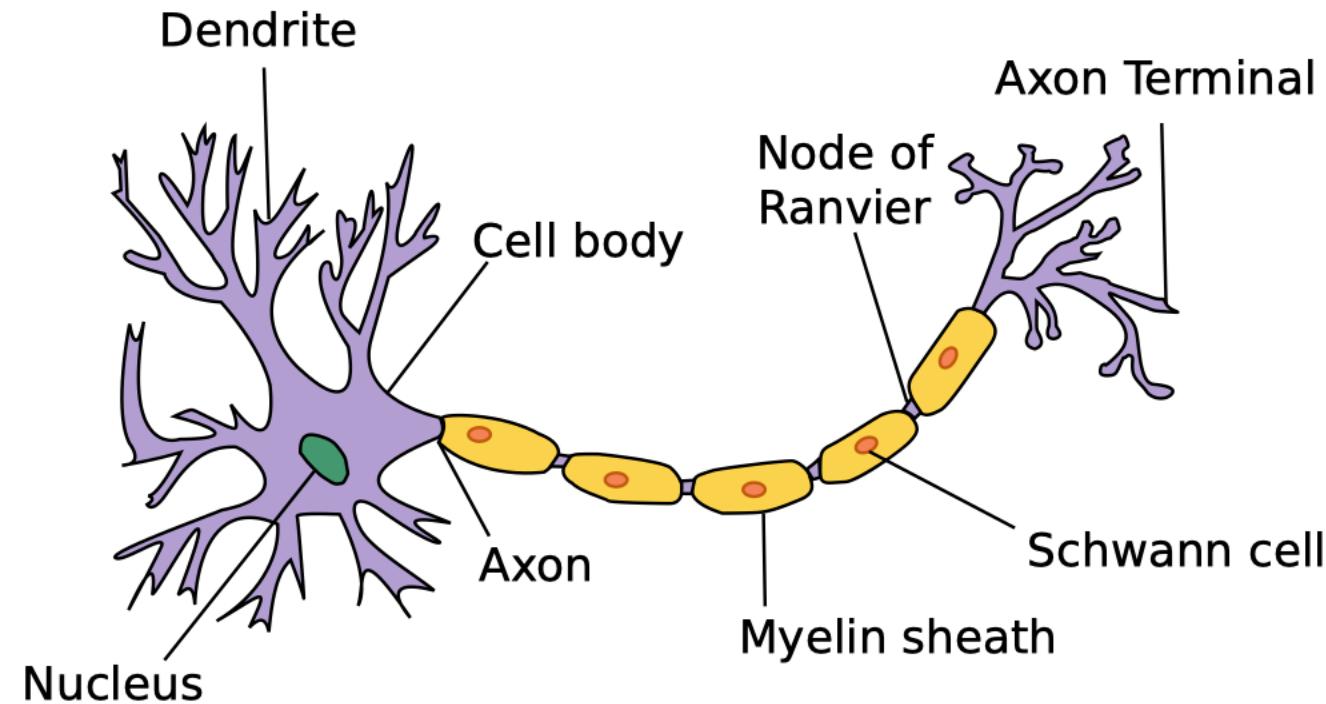
Lecture 2: Neural Networks & Multi-Layer Perceptron

최성준 (고려대학교 인공지능학과)

WARNING: 본 교육 콘텐츠의 지식재산권은 재단법인 네이버커넥트에 귀속됩니다. **본 콘텐츠를 어떠한 경로로든 외부로 유출 및 수정하는 행위를 엄격히 금합니다.** 다만, 비영리적 교육 및 연구활동에 한정되어 사용할 수 있으나 재단의 허락을 받아야 합니다. 이를 위반하는 경우, 관련 법률에 따라 책임을 질 수 있습니다.

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)



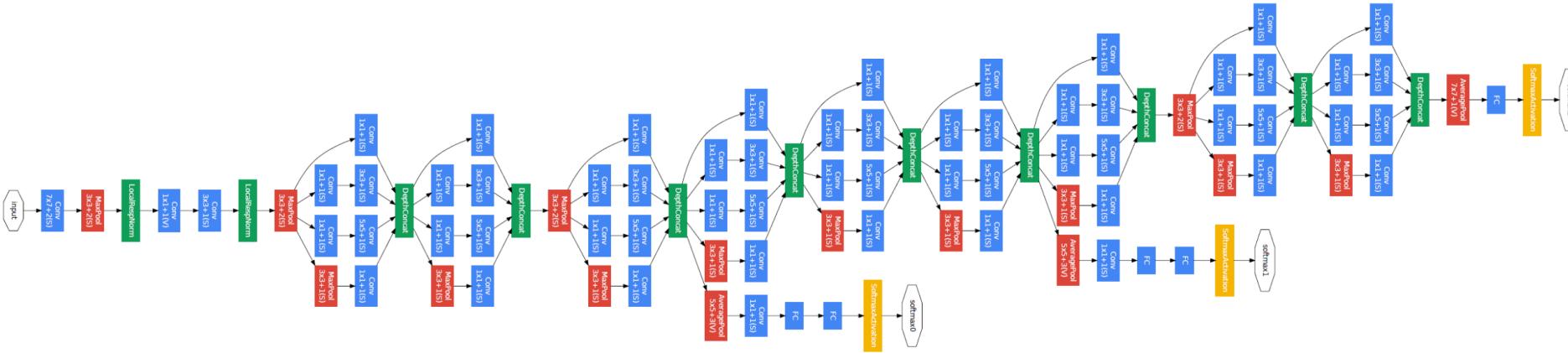
Write Brothers (1903)



F-22 Raptor

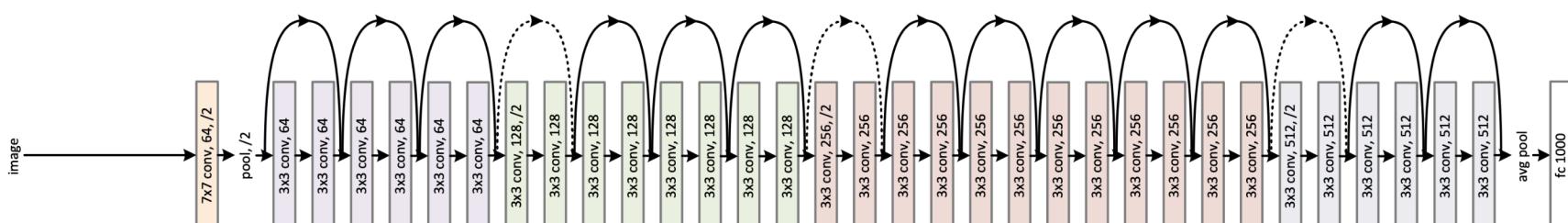
Neural Networks

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



GoogLeNet

Going Deeper with Convolutions, 2014

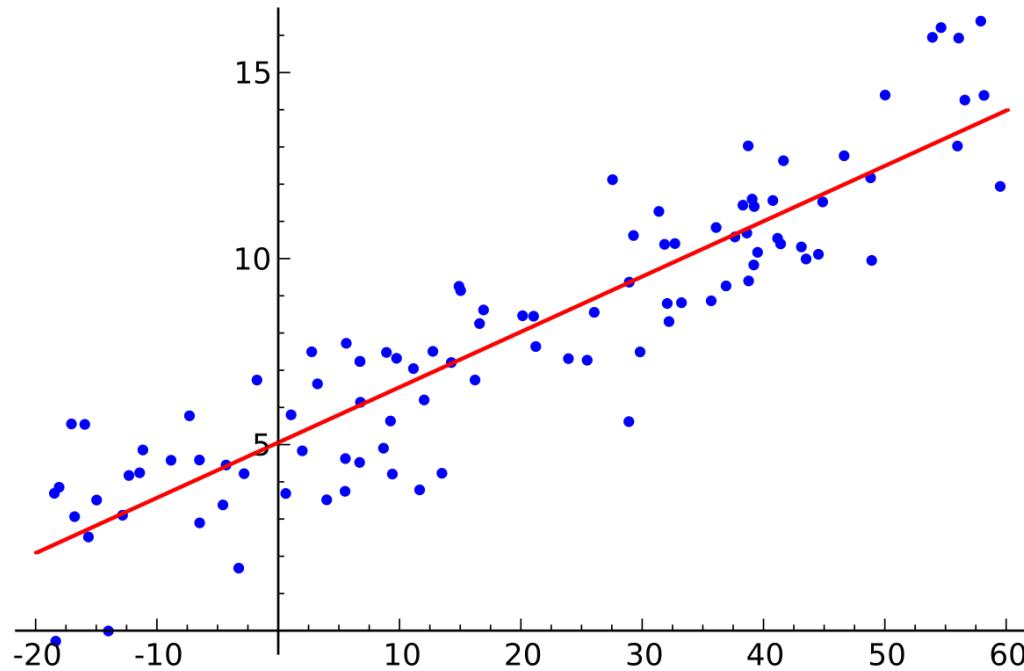


ResNet

Deep Residual Learning for Image Recognition, 2015

Linear Neural Networks

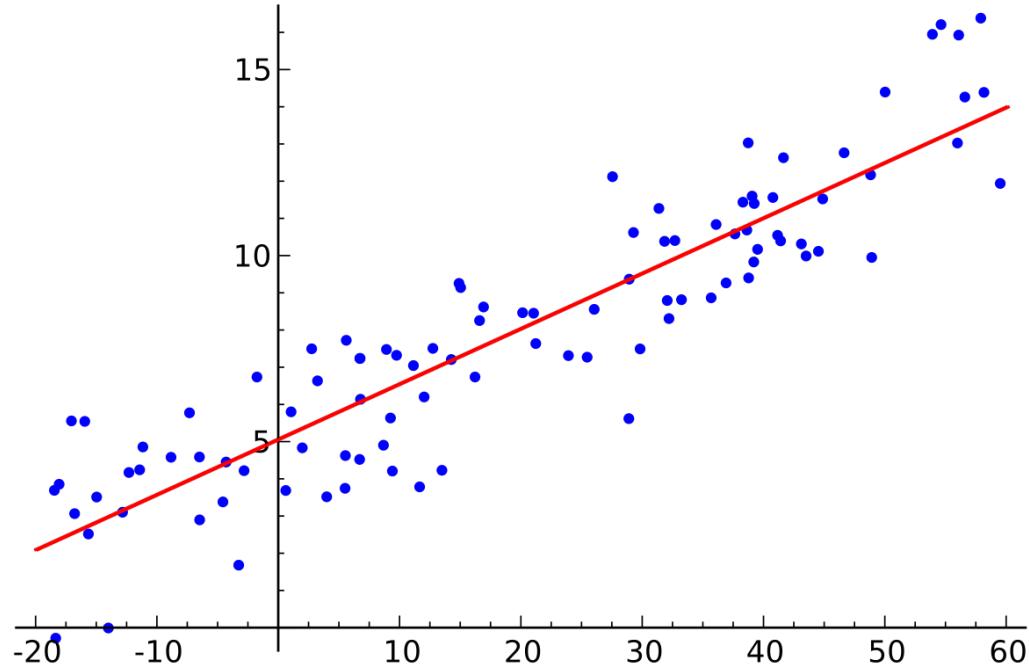
- Let's start with the most simple example.



- 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.

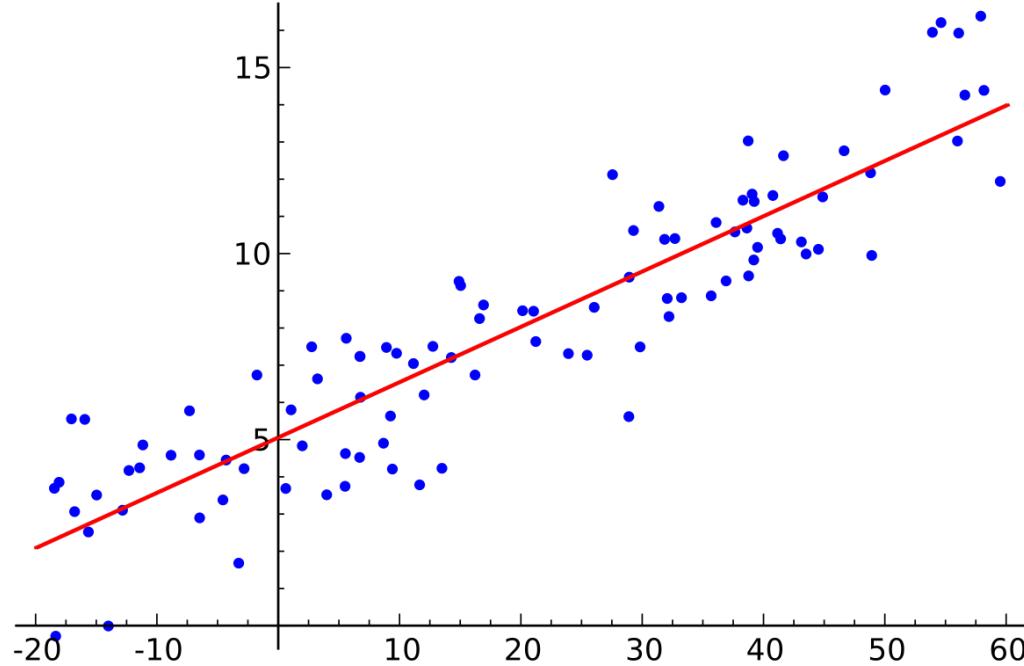


Partial derivative

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

Linear Neural Networks

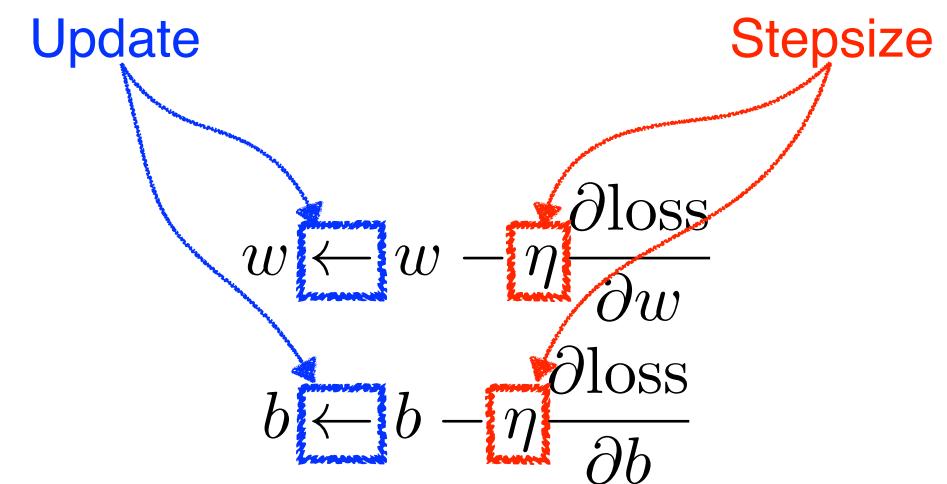
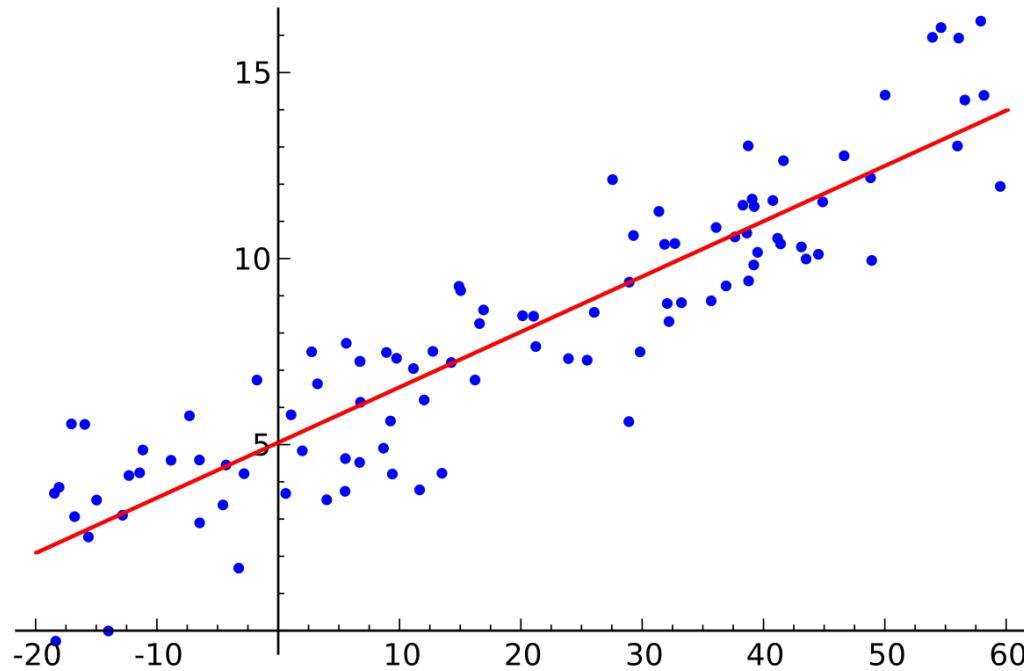
- We compute the partial derivatives w.r.t. the optimization variables.



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

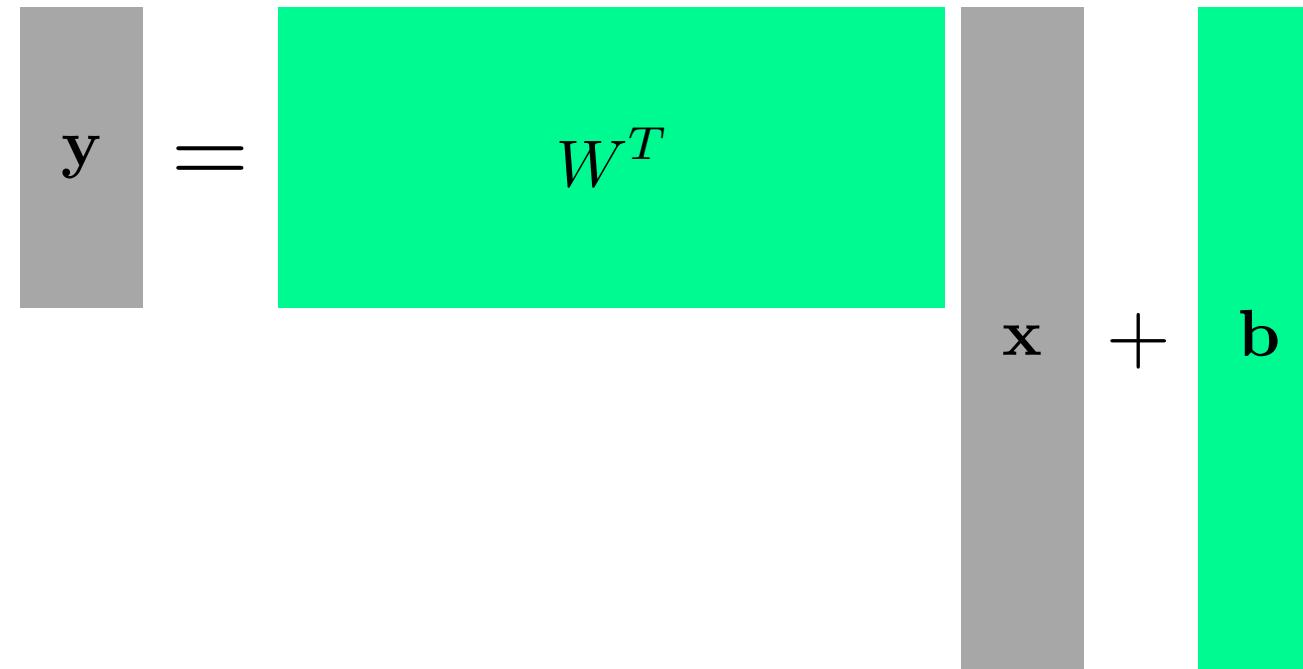
Linear Neural Networks

- Then, we iteratively update the optimization variables.



Linear Neural Networks

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

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


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

Linear Neural Networks

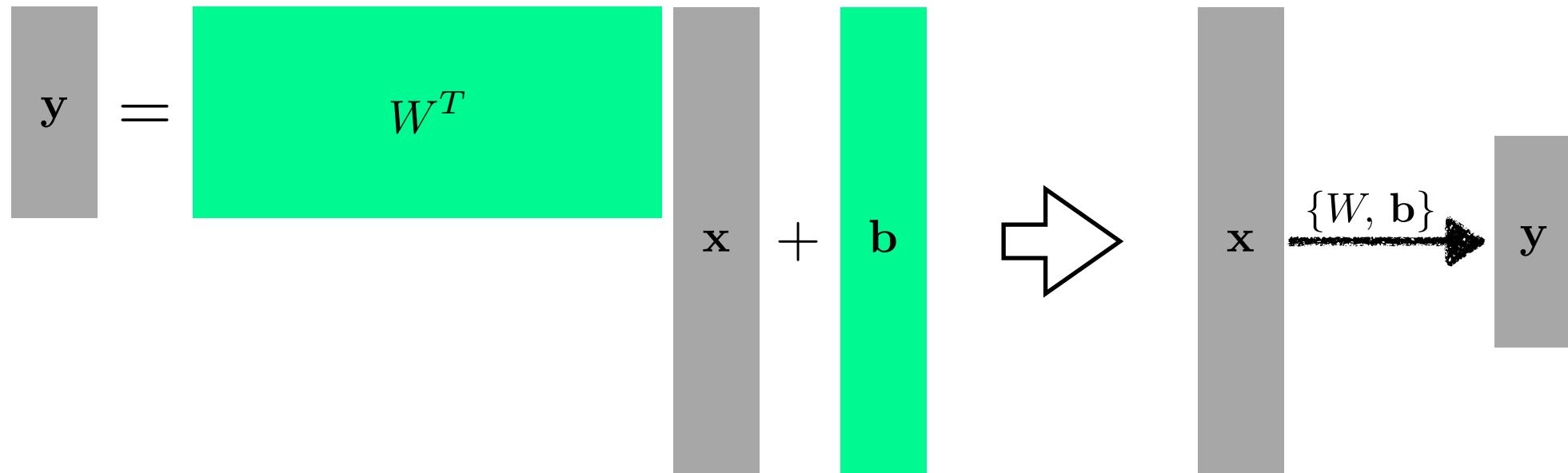
- Of course, we can handle multi dimensional input and output.

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

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

Linear Neural Networks

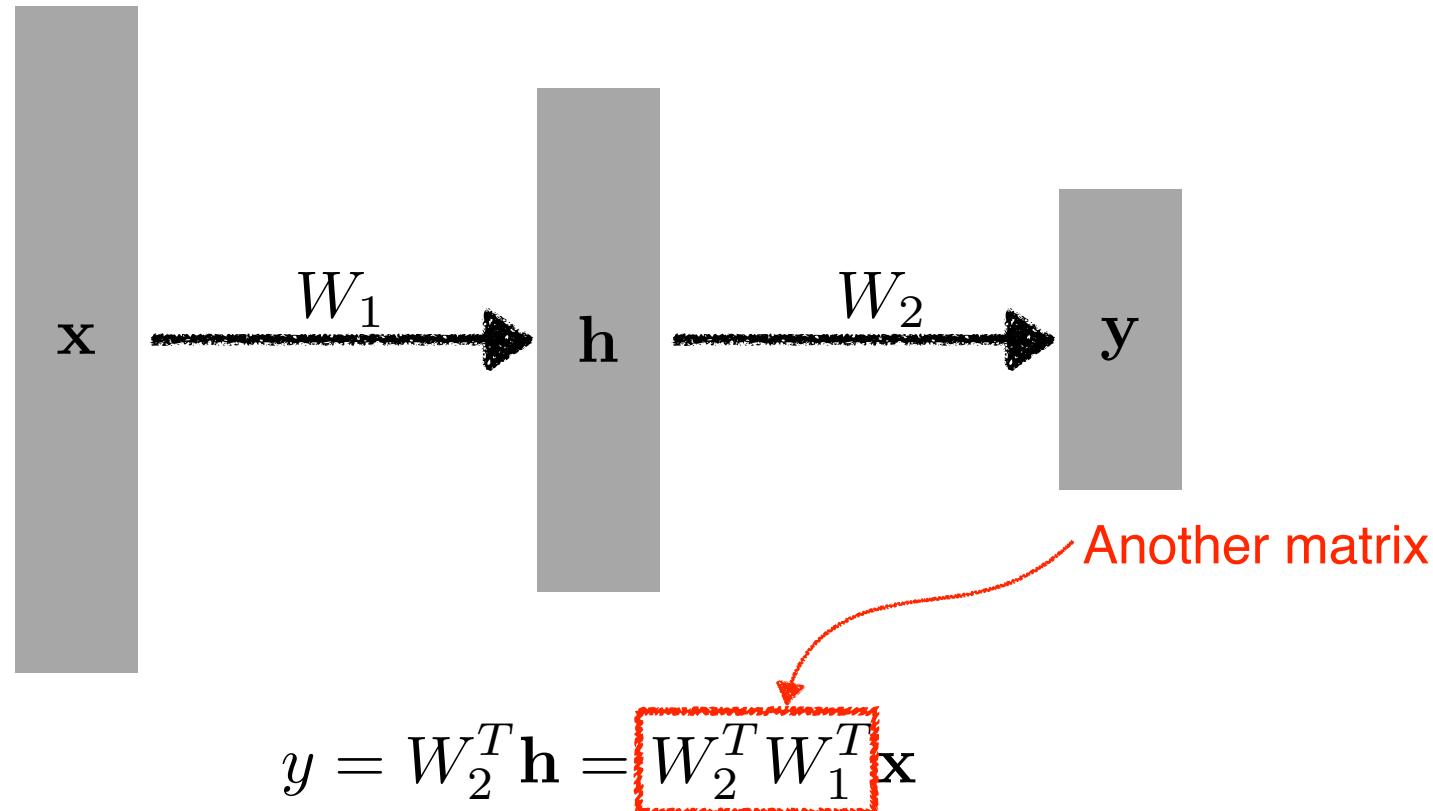
- Of course, we can handle multi dimensional input and output.



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

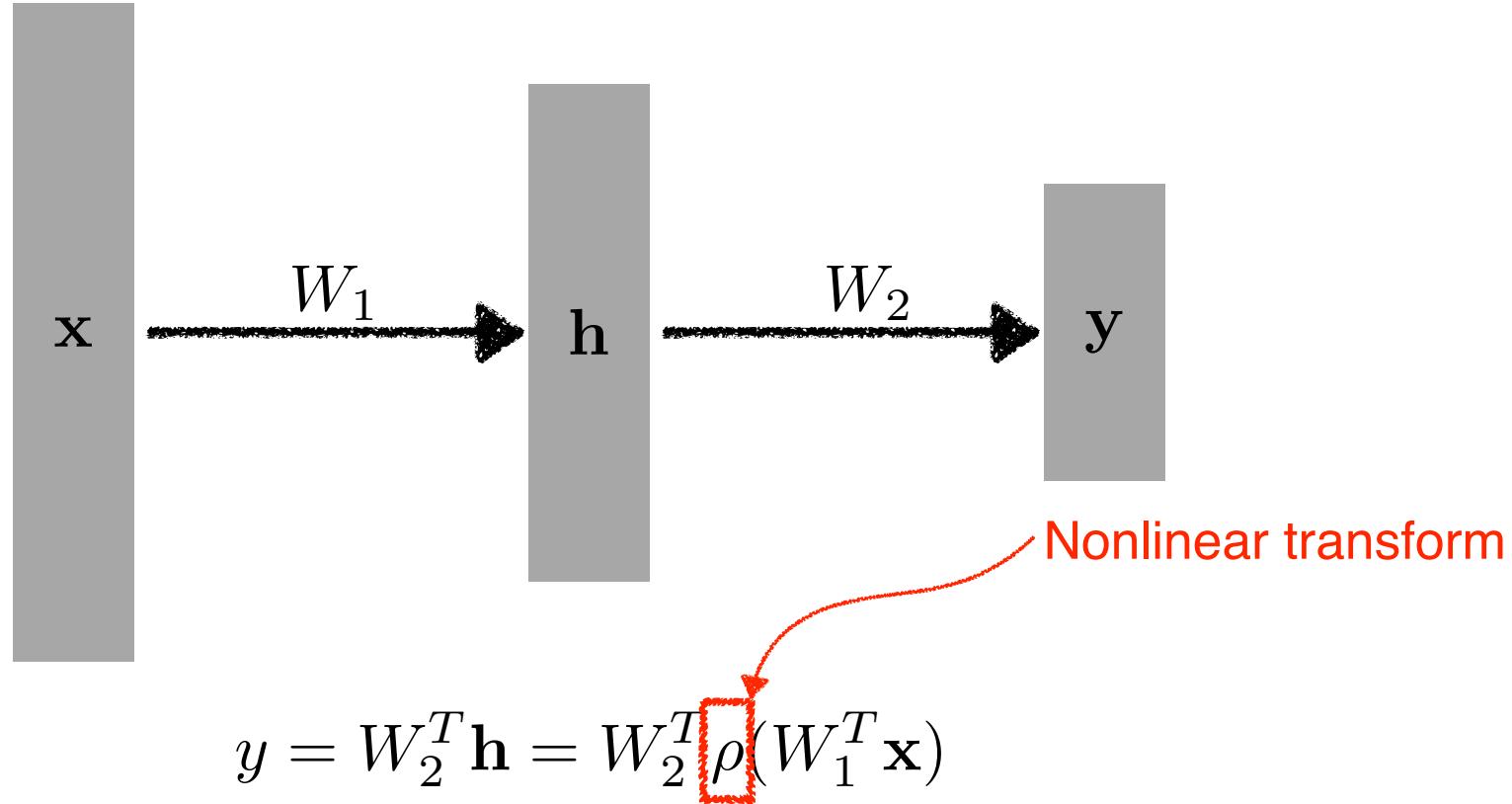
Beyond Linear Neural Networks

- What if we stack more?



Beyond Linear Neural Networks

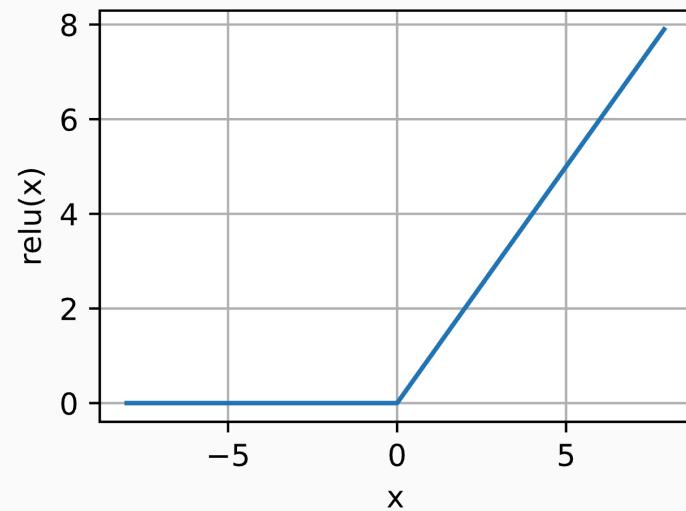
- We need nonlinearity.



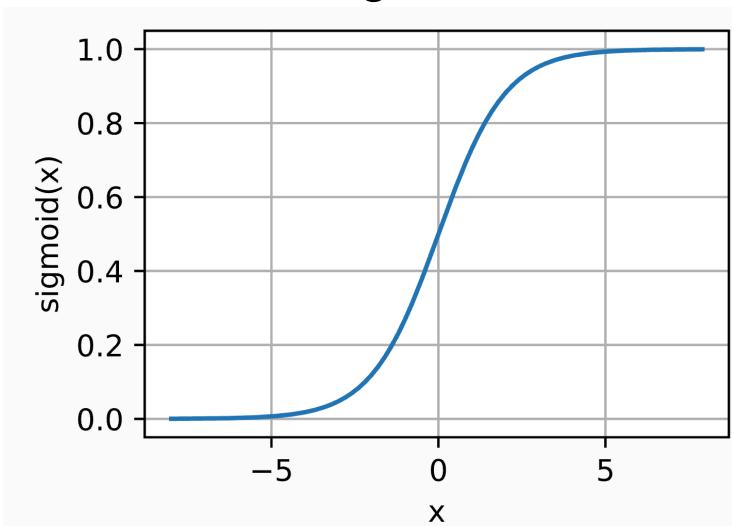
Beyond Linear Neural Networks

- Activation functions

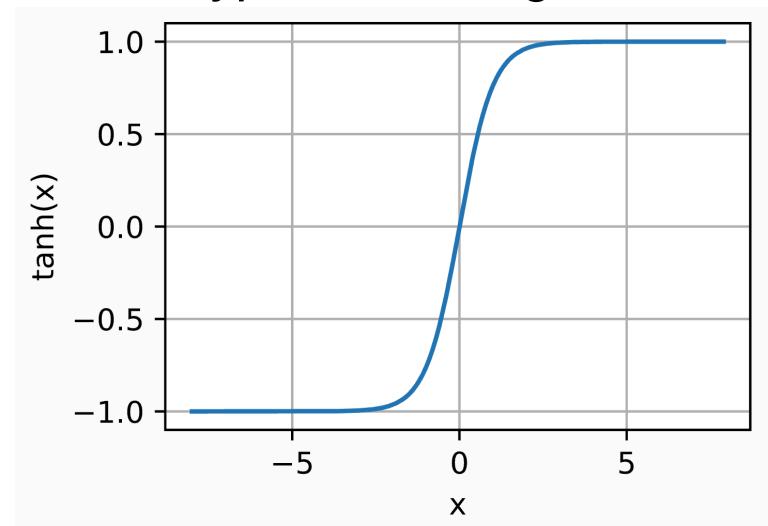
Rectified Linear Unit (ReLU)



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

KUR' HORNIK

Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERK 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 measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficient hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.

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



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 .

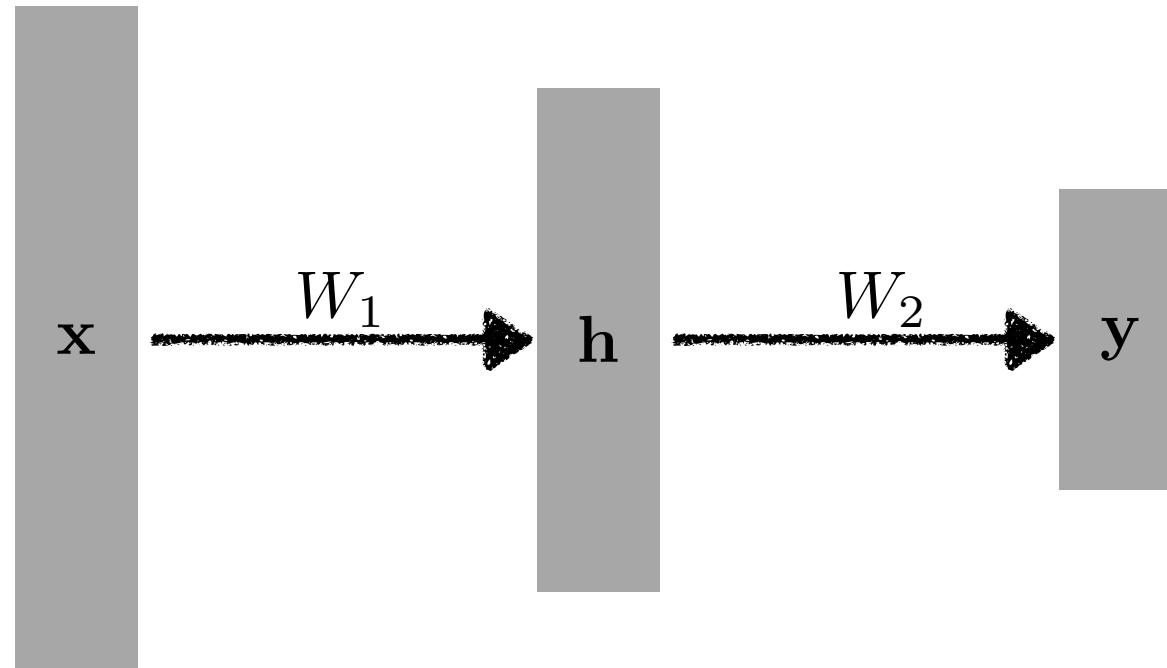
Math

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

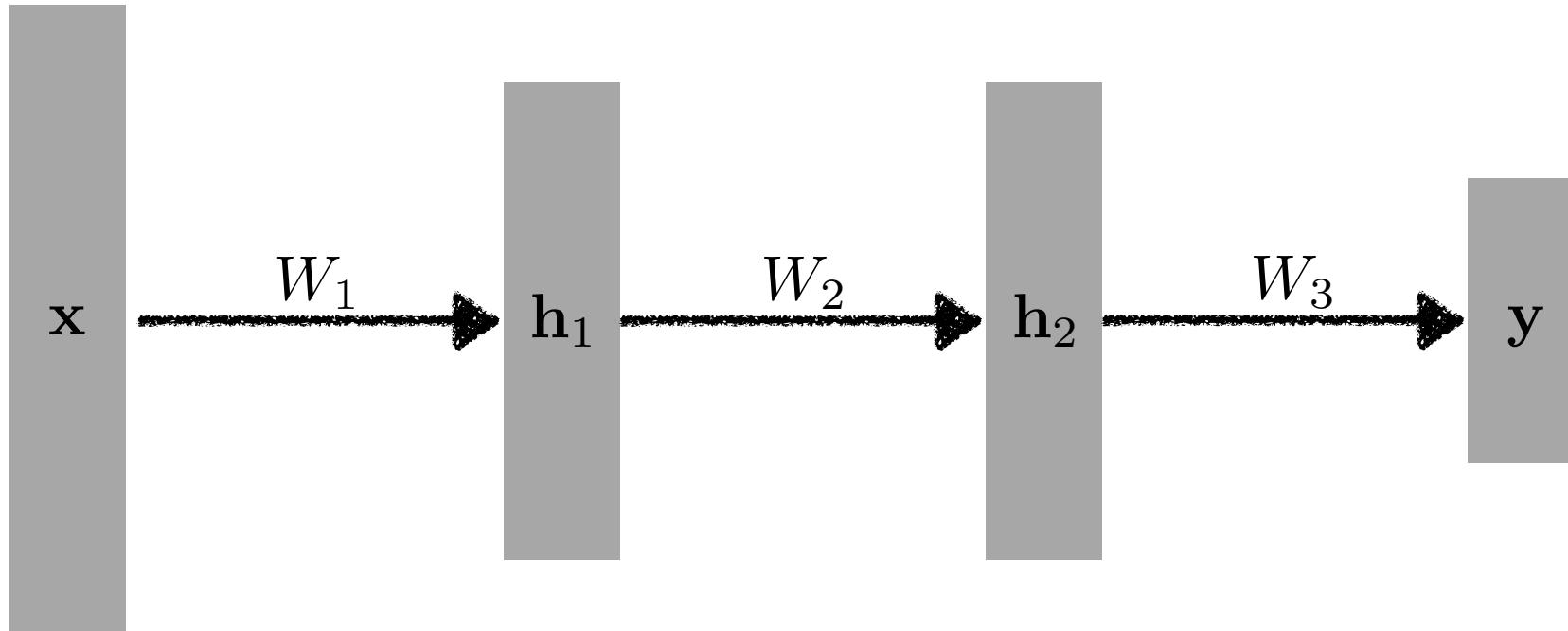
- This class of architectures are often called multi-layer perceptrons.



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

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?

Regression Task

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


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

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

Thank you for listening
