Inhaltsverzeichnis

Abbildungsverzeichnis

List of Equations

Lecture 4: Dense Neural Networks

Date: 15/05/2020

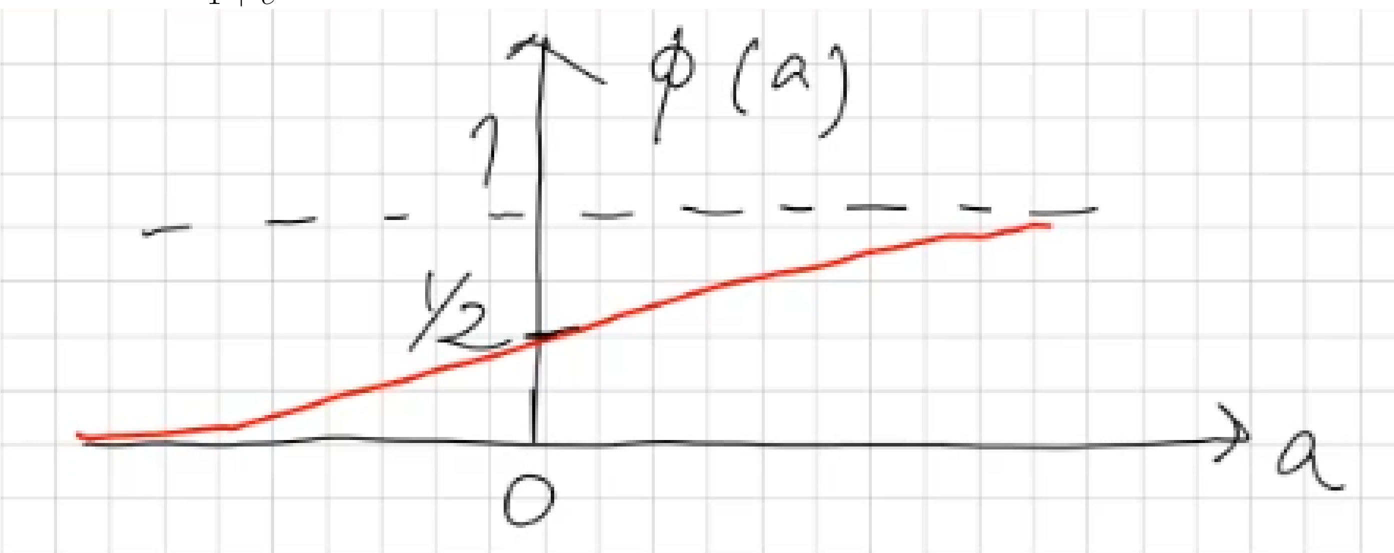
Lecturer: David Silver *By: Nithish Moudhgalya*

1.1. Activation function

Mild requirements on $\phi()$:
•nonlinear in general $\rightarrow$ fundamental
•smooth, differentiable $\rightarrow$ for training
•simple calculation $\rightarrow$ low complexity
•Slides 4-6; 4-7 activation function types

1.1.1. Sigmoid activation function

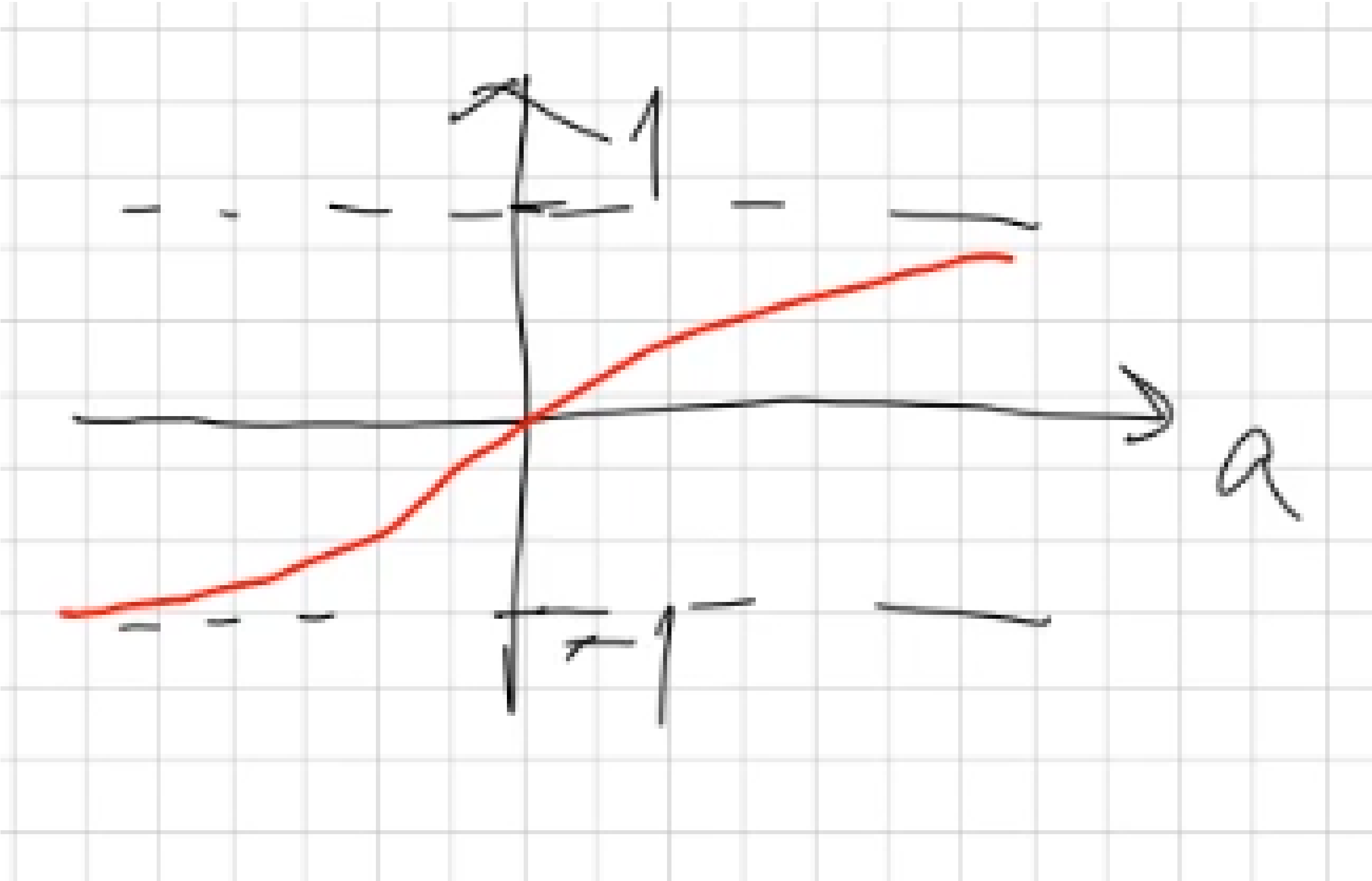
$\phi(a) = \sigma(a) = \dfrac{1}{1+e^{-a}}$



•$0 < \phi < 1 \hat{=}$ prob.
•symmetry: $\phi(-a) = 1 - \phi(a)$
•derivative: $\dfrac{d\phi(a)}{da} = ... = \dfrac{e^{-a}}{(1+e^{-a})^2} = \phi(a) \cdot \phi(-a) = \phi(a)(1-\phi(a)), \in (0,1)$
easy calculative
•widely used in conventional NN (shallow)

1.1.2. hyperbolic tangent activation function

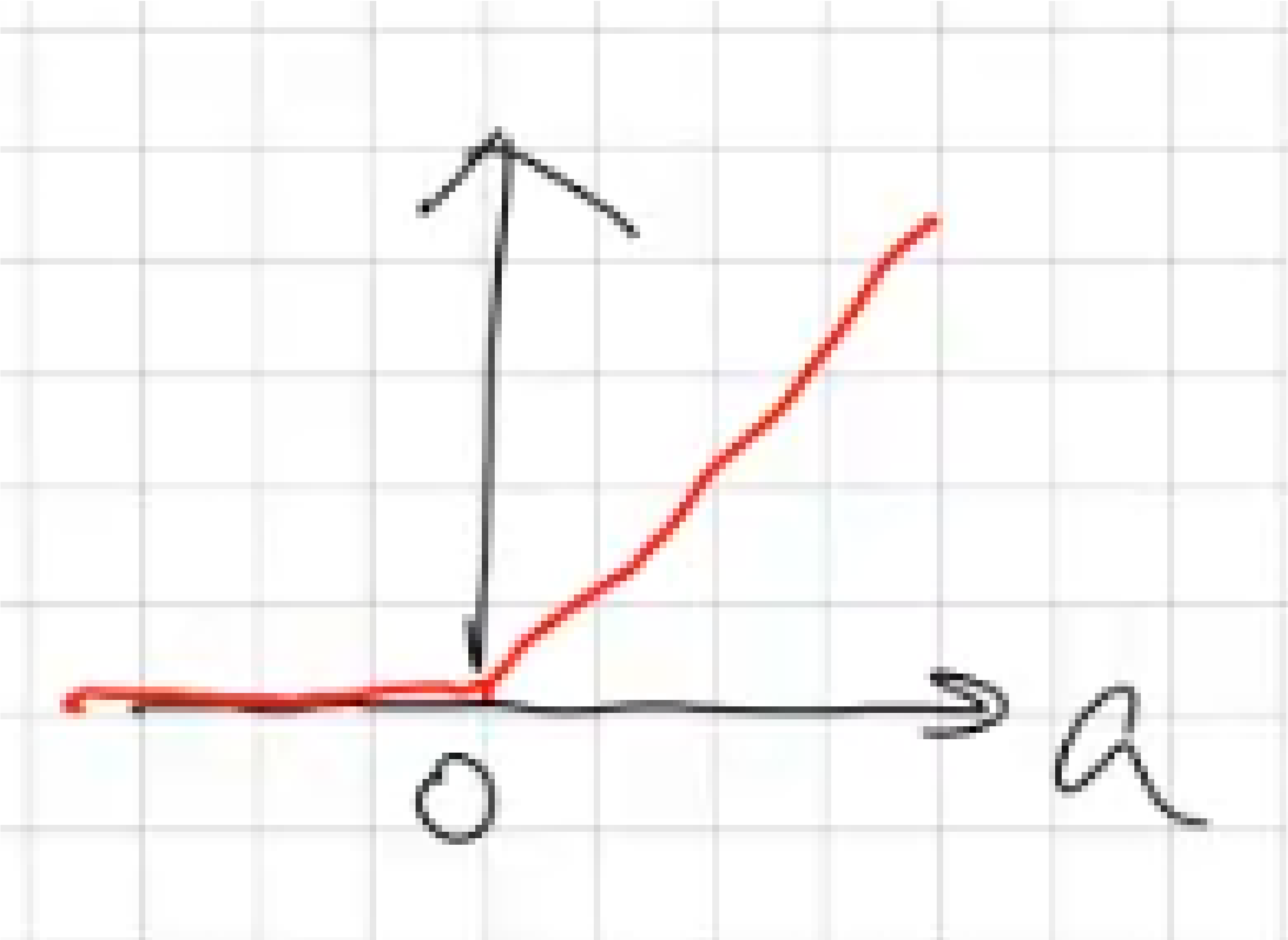
$\phi(a) = tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$



like sigmoid but another output range

1.1.3. rectifier linear unit(ReLU

$\phi(a) = ReLU(a) = max(a, 0) = \begin{cases} a & a \geq 0 \\ 0 & a < 0 \end{cases}$



•≡ diode
simple calculation
•$\dfrac{d\phi}{da} = \begin{cases} 1 & a > 0 \\ 0 & a < 0 \end{cases} = u(a), u(0) = 0$ typically used
•most popular in DNN
•Details on 4-7

1.1.4. Softmax activatoin function(classification problem)

$\phi(\underline{a} : \underline{a} = [a_i] \in \Re^c \to \Re^c$

$\phi(\underline{a}) = softmax(\underline{a}) = \begin{bmatrix} \phi_1(\underline{a}) \\ \vdots \\ \phi_c(\underline{a}) \end{bmatrix}$

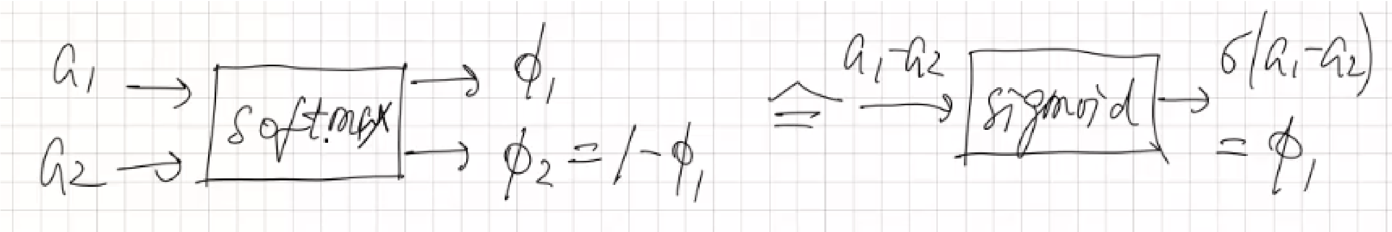
$\phi(\underline{a}) = \dfrac{a^{a_i}}{\sum_{j=1}^{c} e^{a_i}}, \in (0, 1), \sum_{i=1}^{c} \phi_i(\underline{a}) = 1$
•maps $\underline{a} \in \Re^c$ to a categorical PMF with c classes
•a_i large$\to$ $\phi_i(\underline{a})$ close to 1
•a_i small $\to$ $\phi_i(\underline{a})$ close to 0
•used in the output layer for classification problems

1.2. Special case c=2, binary classification problem

$: \phi_1(\underline{a}) = \frac{e^{a_i}}{e^{a_1}+e^{a_2}} = \frac{1}{1+e^{-(a_1-a_2)}} = \sigma(a_1-a_2)$

$\phi_2(\underline{a} = \frac{e^{a_2}}{e^{a_1}+e^{a_2}} = 1-\phi_1(\underline{a}) = \sigma(a_2-a_1)$

i.e. softmax



one sigmoid output is sufficient for binary classification instead of 2 output softmax!

Derivative of softmax:

$$\frac{\partial \phi_i(\underline{a})}{\partial a_j} = ... = \begin{cases} \phi : i(\underline{a}) \cdot (1-\phi_i(\underline{a}) & i=j \\ -phi_i(\underline{a}) \cdot \phi_j(\underline{a}) & i \neq j \end{cases}$$

•4-9 for details on usage

1.3. Chapter 4.5 Universal approximation

4.5 Universal approximation 4-10

Universal approximation theorem $\hat{=}$ existence of a solution

The **universal approximation theorem** states that a feedforward neural network with a linear output layer ($\phi_L(a) = a$) and

- *at least one hidden layer* with
- a *nonlinear* activation function

can approximate any continuous (nonlinear) function $\underline{y}(\underline{x}_0)$ (on compact input sets) to arbitrary accuracy.

Comments:

- arbitrary accuracy: with an increasing number of hidden neurons.
- valid for a wide range of nonlinear activation functions, but excluding polynomials.
- minimum requirement for universal approximation: $\mathbf{W}_2\phi_1(\mathbf{W}_1\underline{x}_0 + \underline{b}_1) + \underline{b}_2$.

For learning $\mathbf{W}_l$ and $\underline{b}_l$ from D_{train}:

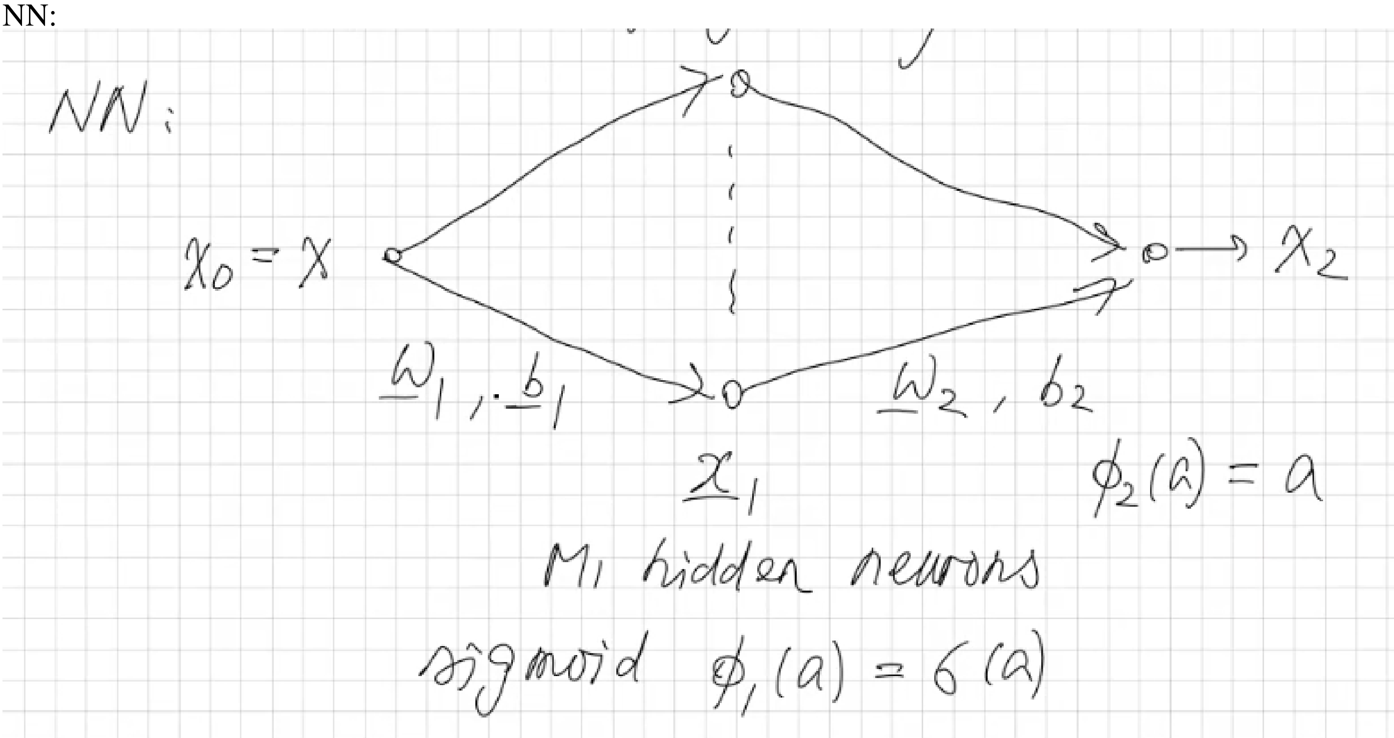
- deep networks are better than shallow ones,
- some activation functions are better than others.

$\hat{=}$ how to find a good solution.

+

1.3.1. E4.3 Regression with 1 hidden layer

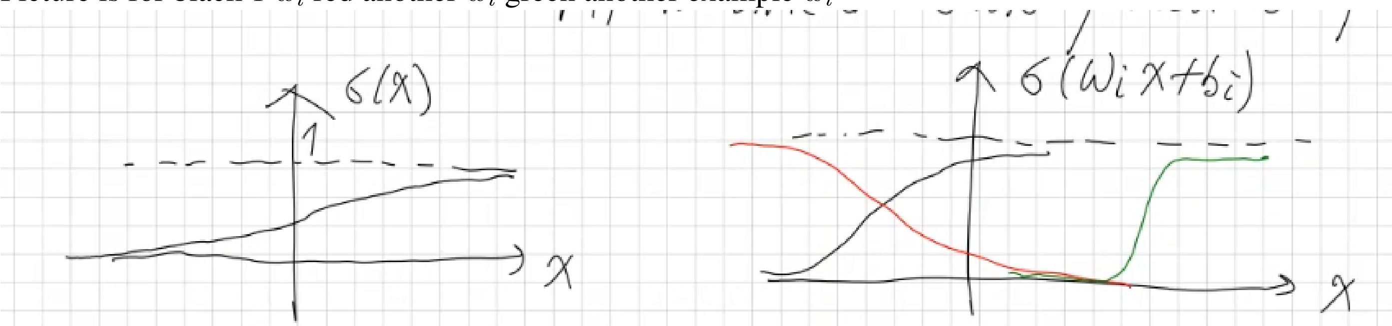
True function: $f_0(x)$
Given: $x(n)$ and noisy function $y(n) = f_(x(n)) + z(n), 1 \leq n \leq N, z(n)$ is the noise
NN:



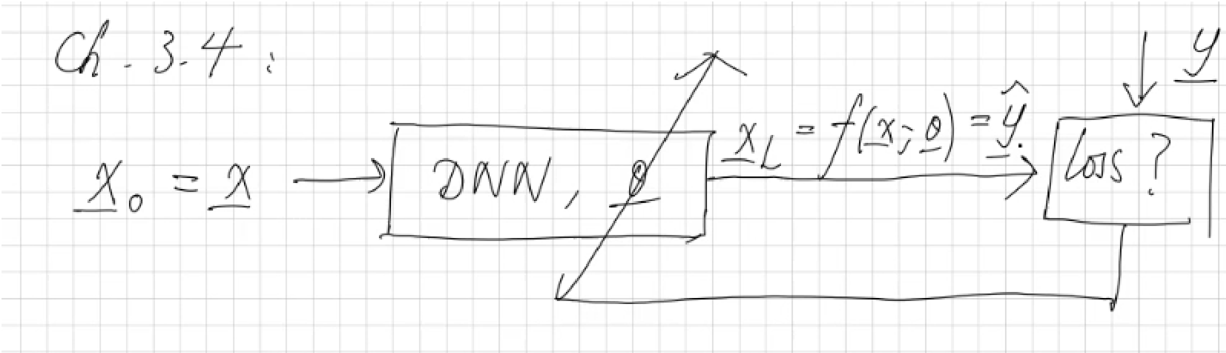
i.e. $x_2 = f(x, \underline{\theta}) = \underline{w}_2^T \sigma(\underline{w}_1 x + \underline{b}_1)$, column times row times scalar
$= \sum_{i=1}^{N_1} w_{2,i} \cdot \underbrace{\sigma(w_{1,i} x + b_{1,i})}_{M_1 \text{ nonllinear basis functions of } x} + b_2$

$\sigma(w_i x + b_i)\sigma(w_i(x + \dfrac{b_i}{w_i}))$

new center at $\dfrac{-b_i}{w_i}$ and new slope value w_i
Picture is for black 1 w_i red another w_i green another example w_i



1.4. Chapter 4.4 - 4.6 Loss and cost function



Review chapter 3.4

$\underset{\underline{\theta}}{min} L(\underline{\theta}) = \frac{1}{N} = \sum_{n=1}^{N} (l(\underline{x}(n), y(n)_i \underline{\theta}$ cost dunction for d_{train}

$l(\underline{x}, \underline{y}_j \underline{\theta}) = -ln(q(\underline{y}|\underline{x}_j \underline{\theta})$ loss for one pair $(\underline{x}, \underline{y})$