Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Mesterséges Intelligencia és Rendszertervezés Tanszék

VIMIAC16 2025/26/I.

Neurális hálók - modelloptimalizálás

Előadó: Dr. Hullám Gábor

Mesterséges intelligencia előadássorozat

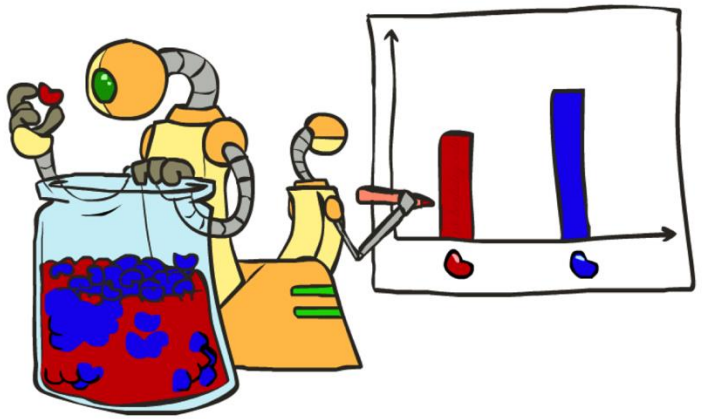
Az előadás diái az AIMA könyvre épülve (http://aima.cs.berkeley.edu) készültek a University of California, Berkeley mesterséges intelligencia kurzusának anyagainak felhasználásával (http://ai.berkeley.edu).

These slides are based on the AIMA book (http://aima.cs.berkeley.edu) and were adapted from the AI course material of University of California, Berkeley (http://ai.berkeley.edu).

Hogyan tanuljunk?

- Maximum likelihood becslés

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_i P_\theta(X_i)$$



- Maximum *feltételes* likelihood becslés

$$\theta^\star = \arg\max_{\theta} P(\mathbf{Y}|\mathbf{X},\theta)$$
$$= \arg\max_{\theta} \prod_i \underbrace{P_\theta(y_i|x_i)}$$
$$\ell(w) = \prod_i \frac{e^{w_{y_i}\cdot x_i}}{\sum_y e^{w_y \cdot x_i}}$$

$$\ell\ell(w) = \sum_i \log P_w(y_i|x_i)$$
$$= \sum_i w_{y_i}\cdot x_i - \log \sum_y e^{w_y \cdot x_i}$$

Legjobb w?

- Maximum likelihood becslés:

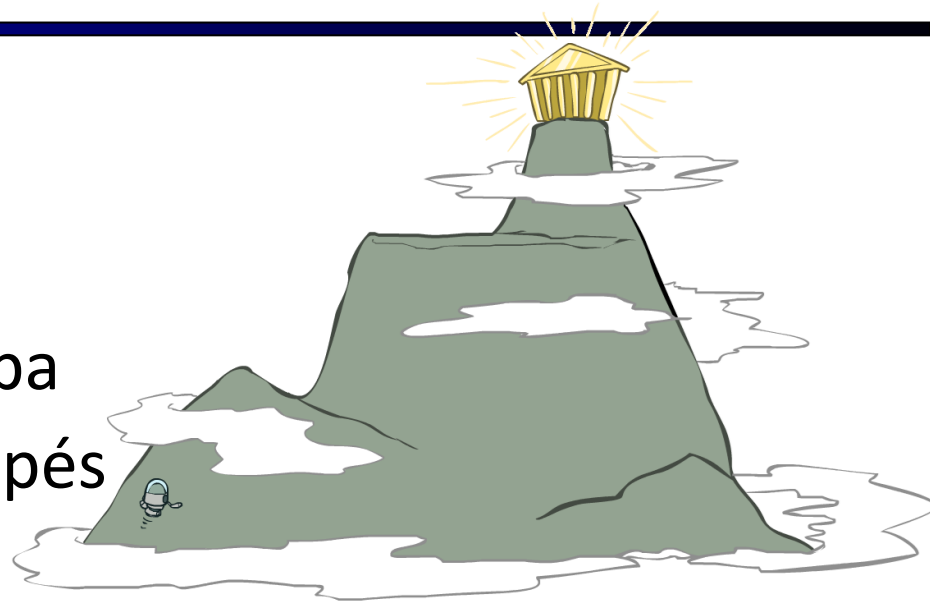
$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

$$P(y^{(i)}|x^{(i)}; w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

= Többosztályos logisztikus regresszió

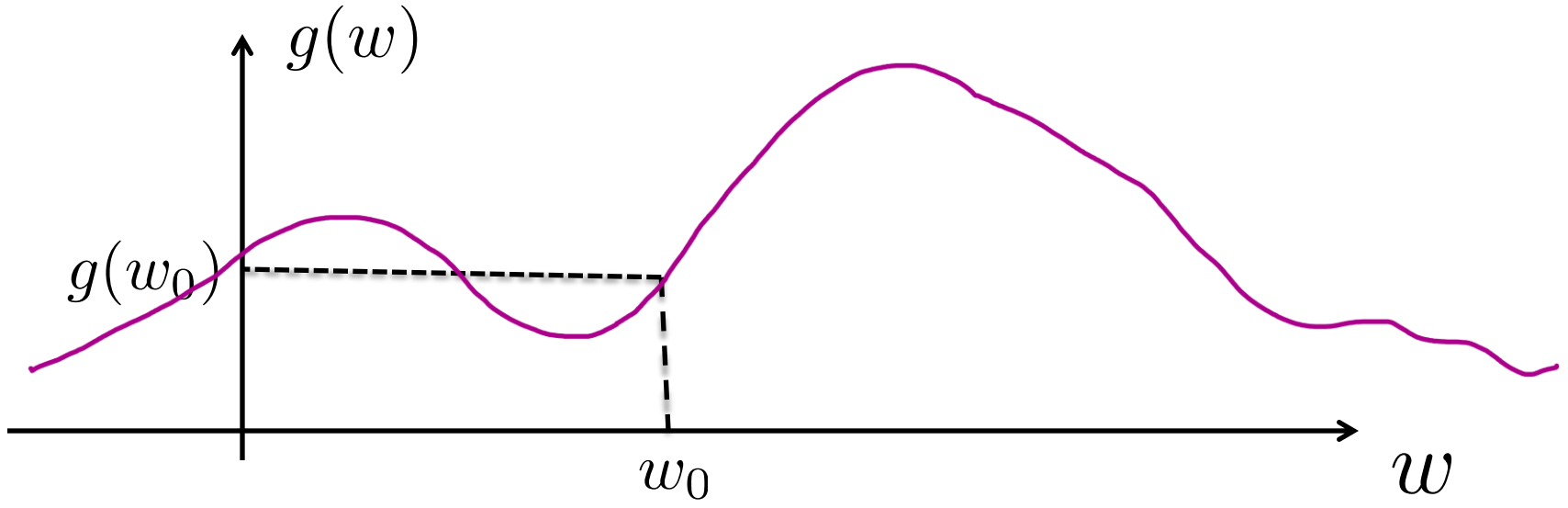
Hegymászó keresés

- egyszerű, általános ötlet
  - Kezdés bárhonnan
  - Ismétel: mozgás a legjobb szomszédos államba
  - Ha nincs jobb szomszéd, mint a jelenlegi, kilépés



- Mi különösen trükkös a hegymászó keresésnél a többosztályos logisztikai regresszió érdekében?
  - Optimalizálás folyamatos térben
  - Végtelenül sok szomszéd!
  - Hogyan lehet ezt hatékonyan csinálni?

1-D Optimalizálás

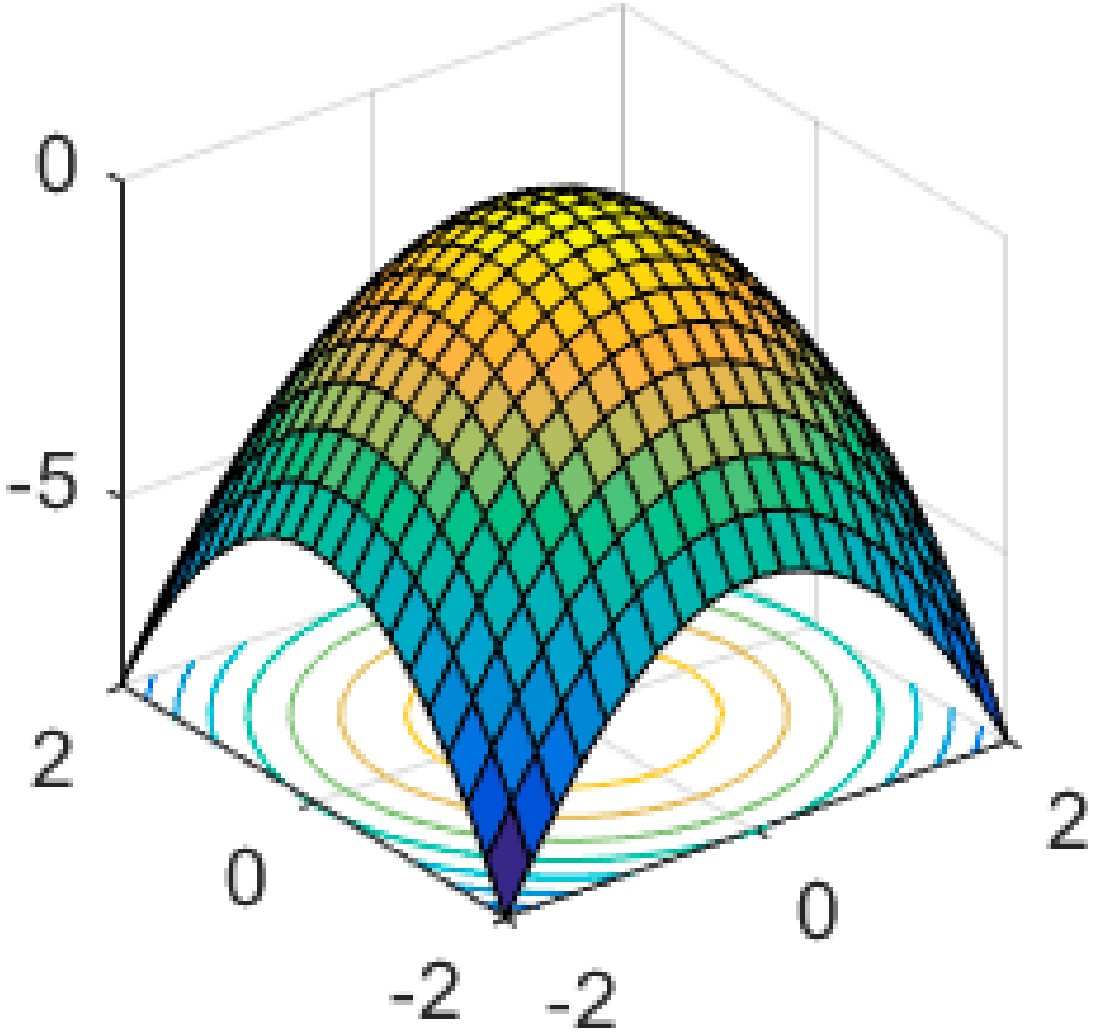


- Értékelés $g(w_0 + h)$ és $g(w_0 - h)$
  - Ezután lépés a legjobb irányba
- Vagy értékelje a származékos terméket:

$$\frac{\partial g(w_0)}{\partial w} = \lim_{h \to 0} \frac{g(w_0 + h) - g(w_0 - h)}{2h}$$

  - Megmondja, hogy melyik irányba lépjen

2-D Optimalizálás



Source: offconvex.org

Gradient Ascent

- Frissítés végrehajtása „felfelé” (hegymászás jelleggel) az egyes koordinátákhoz. Minél meredekebb a meredekség (azaz minél magasabb a deriválté), annál nagyobb a lépés az adott koordinátához
- Például fontolja meg:

$$g(w_1, w_2)$$

  - Frissítések:

$$w_1 \leftarrow w_1 + \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 + \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

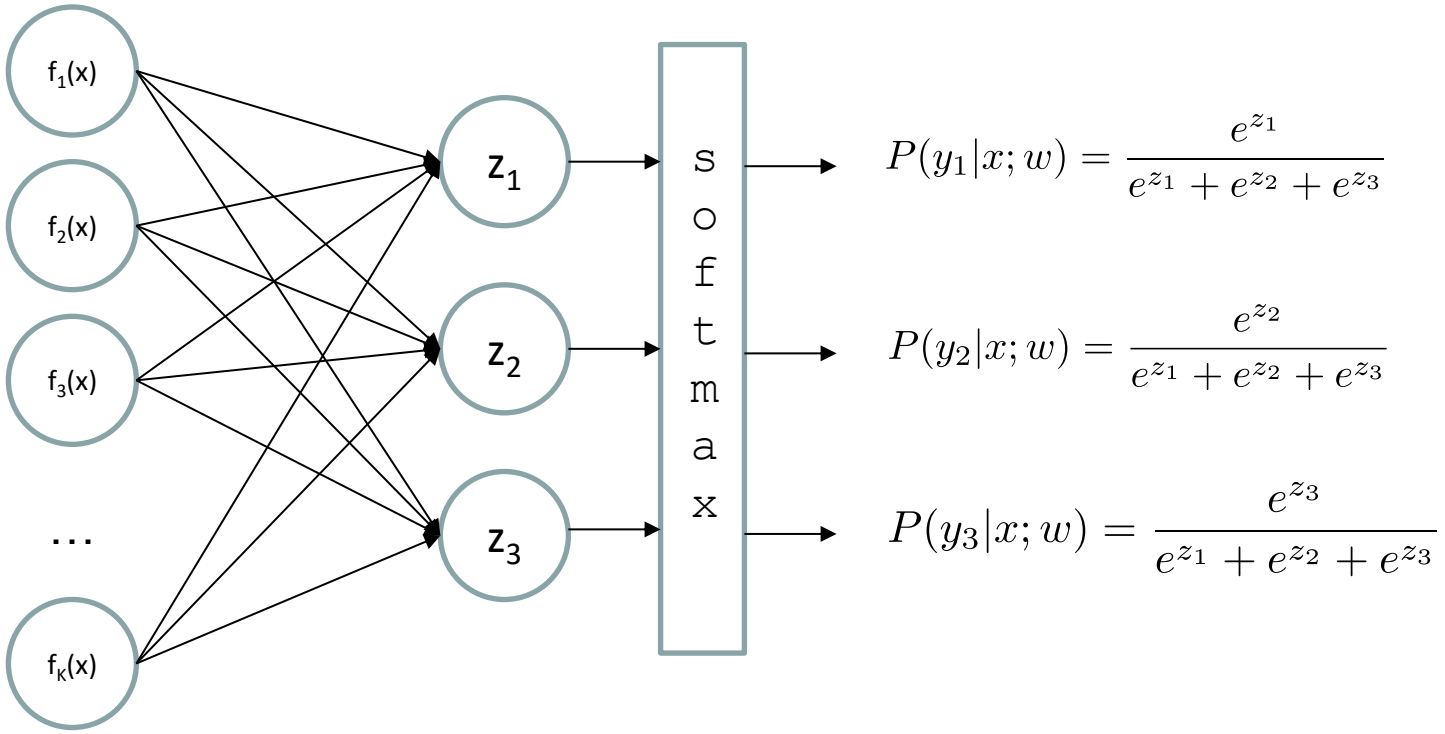
  - Frissítések vektoros jelöléssel:

$$w \leftarrow w + \alpha * \nabla_w g(w)$$

with: $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$ **= gradiens**

Multi-class Logisztikus Regresszió

- = A neurális hálózat speciális esete



$$P(y_1|x;w) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_2|x;w) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$P(y_3|x;w) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Neurális hálózatok tulajdonságai

- Tétel (univerzális függvényközelítők). Egy kétrétegű neurális hálózat, amely elegendő számú neuronnal rendelkezik, bármilyen folyamatos funkciót megközelíthet a kívánt pontossággal.
- Gyakorlati megfontolások
- Úgy tekinthető, mint a jellemzők/jegyek megtanulása
  - Számos neuronból áll 1-1 réteg
  - A túlilleszkedés veszélye fennáll
  - early stopping kell legtöbbször (egy adott ponton túl nem javul)

Univerzális függvényapproximációs tétel*

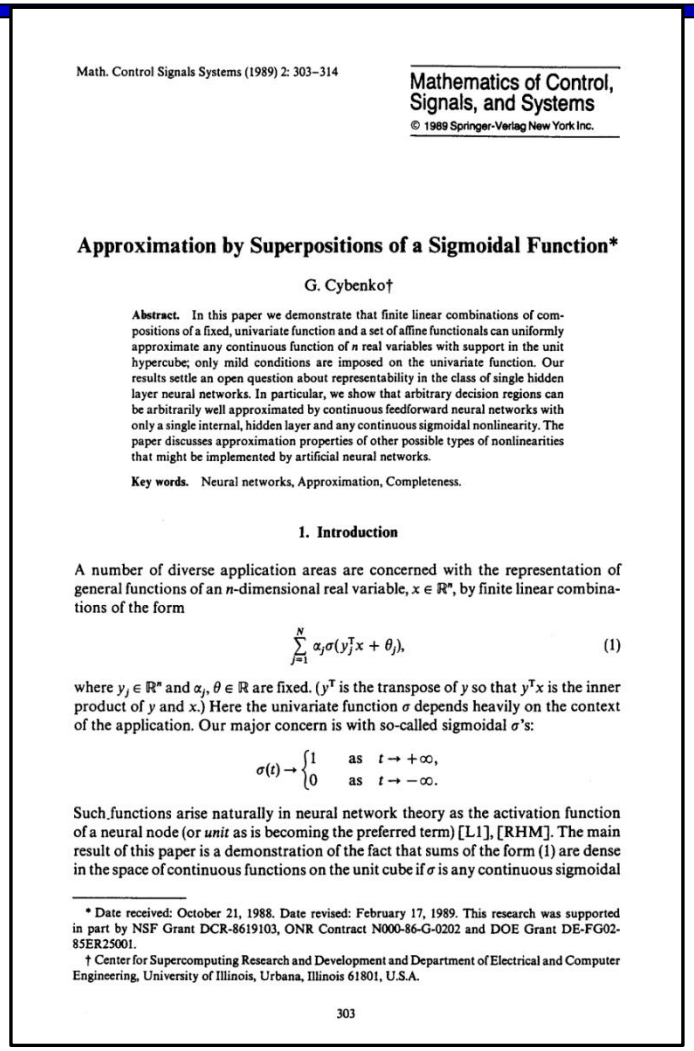
Hornik theorem 1: Whenever the activation function is *bounded and nonconstant*, then, for any finite measure μ, standard multilayer feedforward networks can approximate any function in $L^p(\mu)$ (the space of all functions on R^k such that $\int_{R^k} |f(x)|^p d\mu(x) < \infty$) arbitrarily well, provided that sufficiently many hidden units are available.

Hornik theorem 2: Whenever the activation function is *continuous, bounded and nonconstant*, then, for arbitrary compact subsets $X \subseteq R^k$, standard multilayer feedforward networks can approximate any continuous function on X arbitrarily well with respect to uniform distance, provided that sufficiently many hidden units are available.

- Szavakkal: Bármely f(x) folytonos függvény esetén, ha egy 2 rétegű neurális hálózatnak elegendő rejtett egysége van, akkor létezik a súlyok olyan megválasztása, ami lehetővé teszi f(x)-szoros megközelítését.

Cybenko (1989) “Approximations by superpositions of sigmoidal functions”
Hornik (1991) “Approximation Capabilities of Multilayer Feedforward Networks”
Leshno and Schocken (1991) ”Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function”

Univerzális függvényyapproximációs tétel*



Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

1. Introduction

A number of diverse application areas are concerned with the representation of general functions of an n-dimensional real variable, $x \in \mathbb{R}^n$, by finite linear combinations of the form

$$\sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j), \tag{1}$$

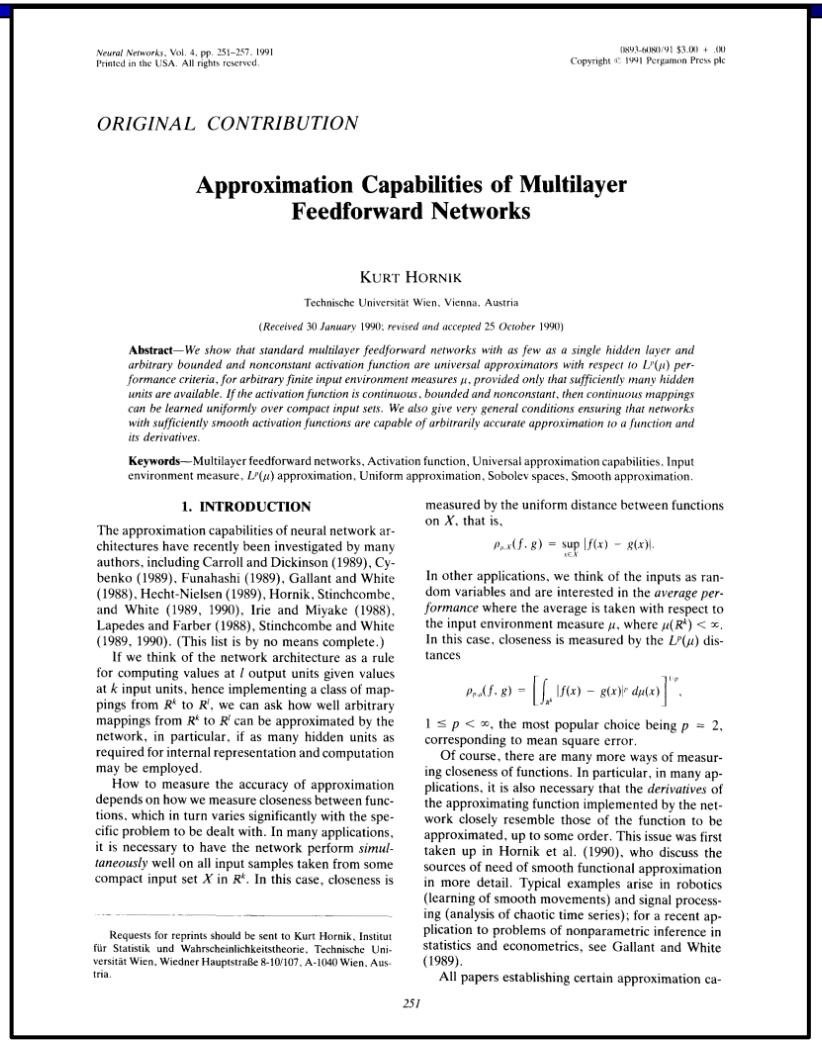
where $y_j \in \mathbb{R}^n$ and $\alpha_j, \theta \in \mathbb{R}$ are fixed. (y^T is the transpose of y so that $y^T x$ is the inner product of y and x.) Here the univariate function σ depends heavily on the context of the application. Our major concern is with so-called sigmoidal σ's:

$$\sigma(t) \to \begin{cases} 1 & \text{as} \quad t \to +\infty, \\ 0 & \text{as} \quad t \to -\infty. \end{cases}$$

Such functions arise naturally in neural network theory as the activation function of a neural node (or *unit* as is becoming the preferred term) [L1], [RHM]. The main result of this paper is a demonstration of the fact that sums of the form (1) are dense in the space of continuous functions on the unit cube if σ is any continuous sigmoidal

* Date received: October 21, 1988. Date revised: February 17, 1989. This research was supported in part by NSF Grant DCR-8619103, ONR Contract N000-86-G-0202 and DOE Grant DE-FG02-85ER25001.

† Center for Supercomputing Research and Development and Department of Electrical and Computer Engineering, University of Illinois, Urbana, Illinois 61801, U.S.A.



Neural Networks, Vol. 4, pp. 251–257. 1991
Printed in the USA.

0893-6080/91 $3.00 + .00

ORIGINAL CONTRIBUTION

Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

Technische Universität Wien, Vienna, Austria

(*Received* 30 *January* 1990; *revised and accepted* 25 *October* 1990)

Abstract—*We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^p(\mu)$ performance criteria, for arbitrary finite input environment measures μ, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.*

Keywords—Multilayer feedforward networks, Activation function, Universal approximation capabilities, Input environment measure, $L^p(\mu)$ approximation, Uniform approximation, Sobolev spaces, Smooth approximation.

1. INTRODUCTION

The approximation capabilities of neural network architectures have recently been investigated by many authors, including Carroll and Dickinson (1989), Cybenko (1989), Funahashi (1989), Gallant and White (1988), Hecht-Nielsen (1989), Hornik, Stinchcombe, and White (1989, 1990), Irie and Miyake (1988), Lapedes and Farber (1988), Stinchcombe and White (1989, 1990). (This list is by no means complete.)

If we think of the network architecture as a rule for computing values at l output units given values at k input units, hence implementing a class of mappings from R^k to R^l, we can ask how well arbitrary mappings from R^k to R^l can be approximated by the network, in particular, if as many hidden units as required for internal representation and computation may be employed.

How to measure the accuracy of approximation depends on how we measure closeness between functions, which in turn varies significantly with the specific problem to be dealt with. In many applications, it is necessary to have the network perform *simultaneously* well on all input samples taken from some compact input set X in R^k. In this case, closeness is measured by the uniform distance between functions on X, that is,

$$\rho_{\infty,X}(f, g) = \sup_{x \in X} |f(x) - g(x)|.$$

In other applications, we think of the inputs as random variables and are interested in the *average performance* where the average is taken with respect to the input environment measure μ, where $\mu(R^k) < \infty$. In this case, closeness is measured by the $L^p(\mu)$ distances

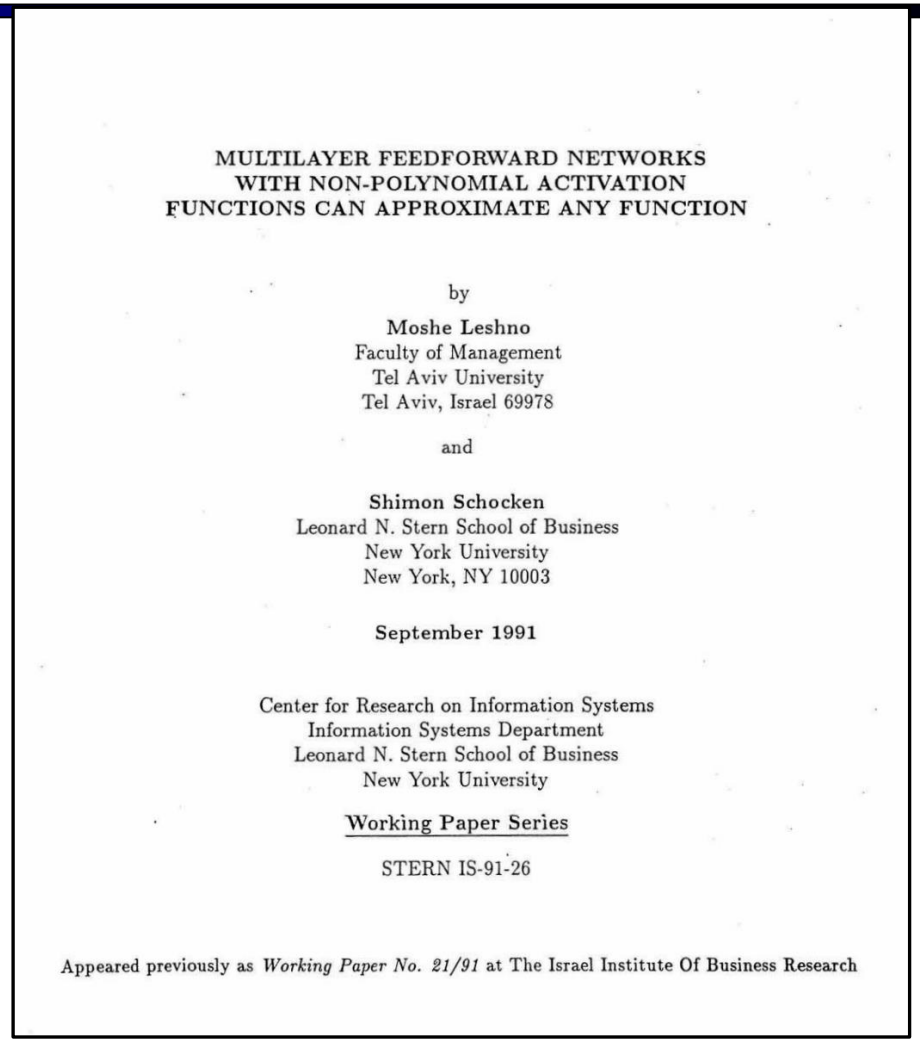
$$\rho_{p,\mu}(f, g) = \left[\int_{R^k} |f(x) - g(x)|^p \, d\mu(x) \right]^{1/p},$$

$1 \le p < \infty$, the most popular choice being $p = 2$, corresponding to mean square error.

Of course, there are many more ways of measuring closeness of functions. In particular, in many applications, it is also necessary that the *derivatives* of the approximating function implemented by the network closely resemble those of the function to be approximated, up to some order. This issue was first taken up in Hornik et al. (1990), who discuss the sources of need of smooth functional approximation in more detail. Typical examples arise in robotics (learning of smooth movements) and signal processing (analysis of chaotic time series); for a recent application to problems of nonparametric inference in statistics and econometrics, see Gallant and White (1989).

All papers establishing certain approximation ca-

Requests for reprints should be sent to Kurt Hornik, Institut für Statistik und Wahrscheinlichkeitstheorie, Technische Universität Wien, Wiedner Hauptstraße 8-10/107, A-1040 Wien, Austria.



MULTILAYER FEEDFORWARD NETWORKS WITH NON-POLYNOMIAL ACTIVATION FUNCTIONS CAN APPROXIMATE ANY FUNCTION

by

Moshe Leshno
Faculty of Management
Tel Aviv University
Tel Aviv, Israel 69978

and

Shimon Schocken
Leonard N. Stern School of Business
New York University
New York, NY 10003

September 1991

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-91-26

Appeared previously as *Working Paper No. 21/91* at The Israel Institute Of Business Research

Cybenko (1989) “Approximations by superpositions of sigmoidal functions”
Hornik (1991) “Approximation Capabilities of Multilayer Feedforward Networks”
Leshno and Schocken (1991) ”Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function”

Neural Net Demo Site

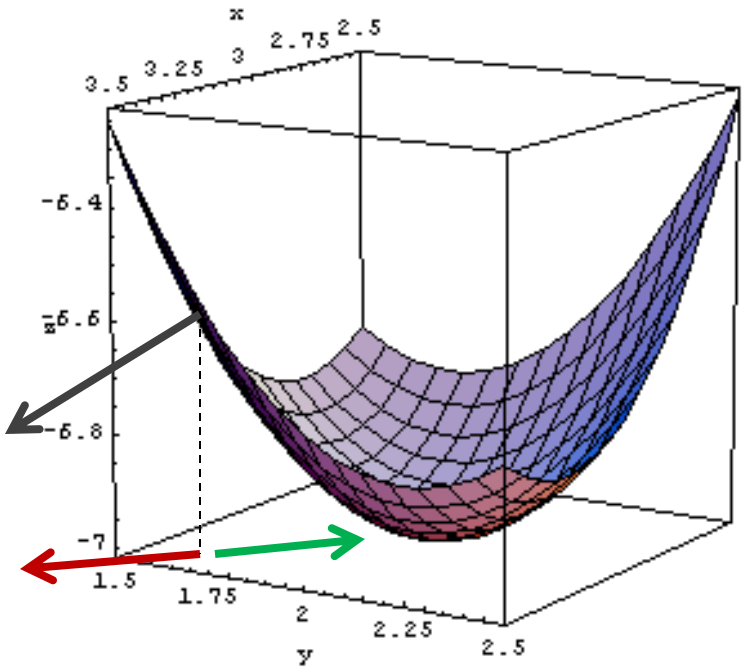
- Demo-site:
  - http://playground.tensorflow.org/

Más megközelítés: minimalizáljuk a hibát

- Minta feldolgozása: bemenet alapján a neurális hálózat kimenetének számítása (forward pass),
- Ha hiba áll elő, azaz a kimenet és az elvárt érték eltér, a súlyokat a hiba csökkentése érdekében módosítani kel.
- Ehhez ki kell számítani a hiba arányát az egyes súlyoknak megfelelően.

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W}$$

Gradient Descent



Veszteségfüggvények / Hibafüggvények

- Négyzetes hiba– regresszió

$$E(\underline{d},\underline{y}) = \frac{1}{2}\sum_i (d_i - y_i)^2$$

- Keresztentrópia –bináris osztályozás

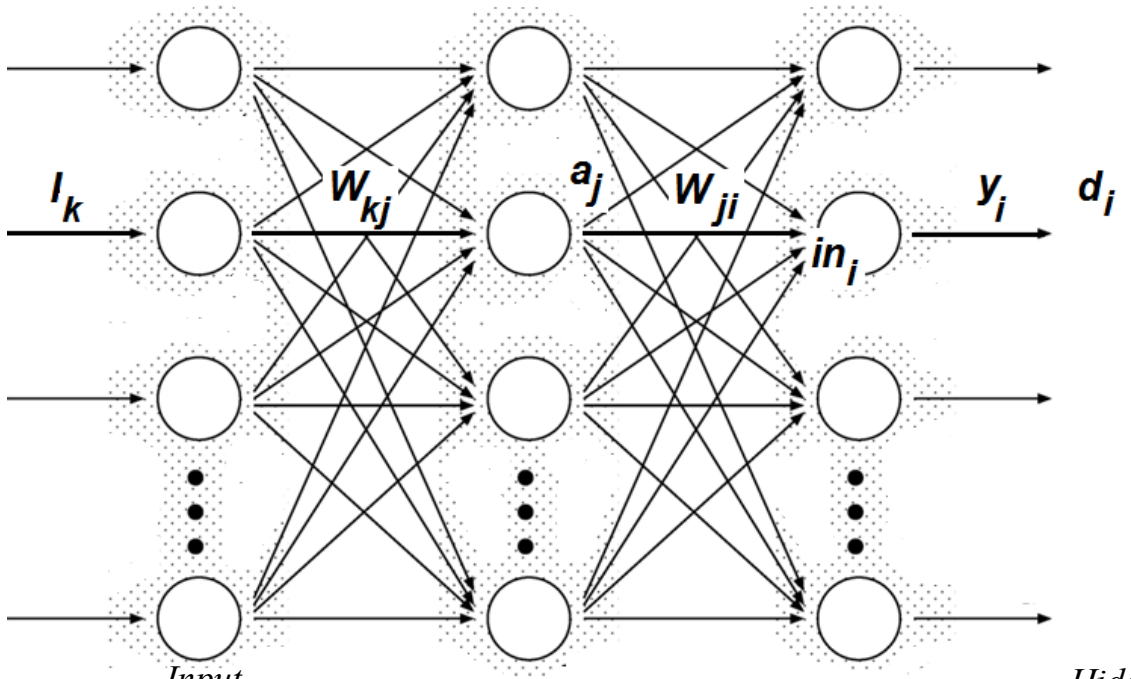
$$E(\underline{d},\underline{y}) = -\sum_i d_i \log y_i$$

- A cél olyan súlyok megtalálása, amelyeknél a hiba minimális

$$w^* = \arg\min_w \sum_{i=1}^{N} E(d_i, y_i)$$, ahol $\underline{y} = f(\underline{x},\underline{w})$

Tanítás

- Forward pass
- Backpropagation

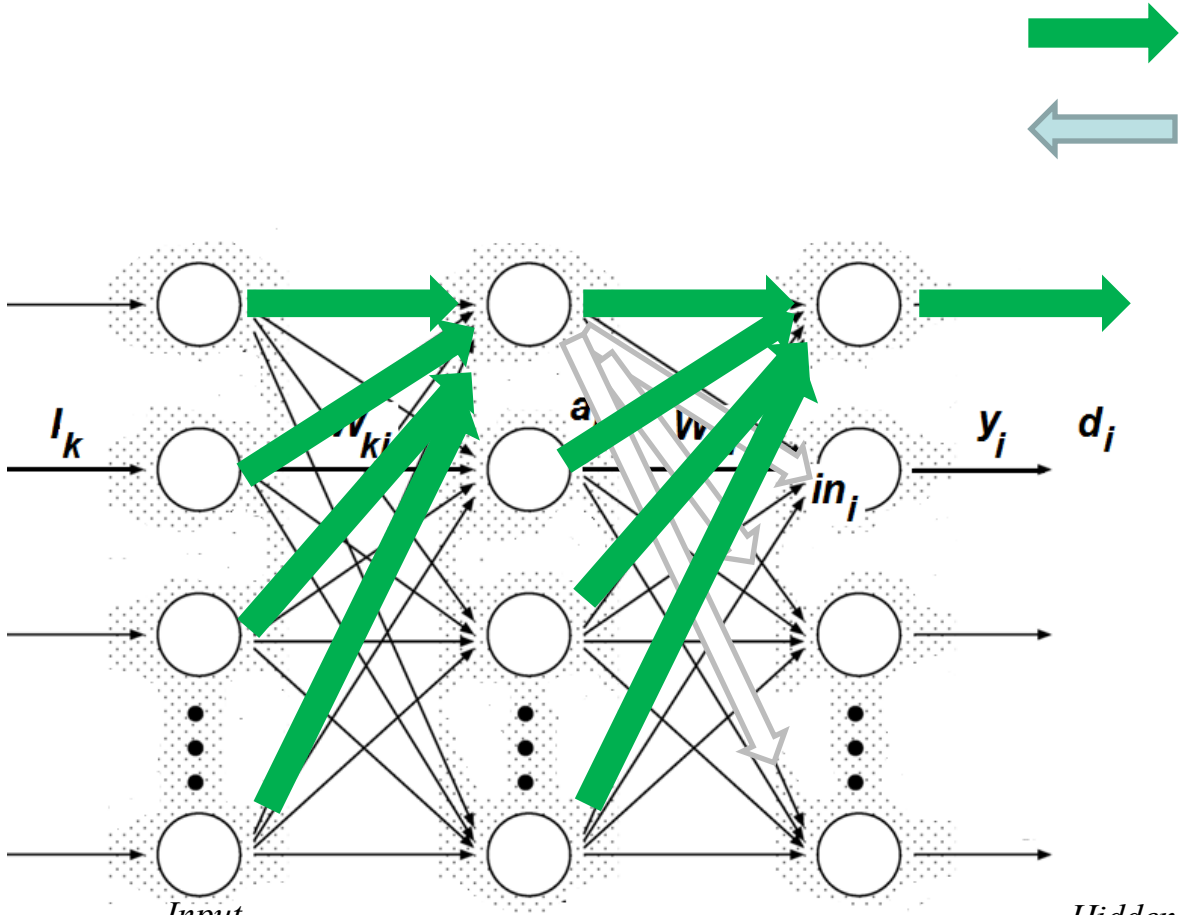


$$a_j(\underline{x}) = g(w_{k0} + \sum_{k=1}^{Input} x_k \cdot w_{kj})$$

$$y_i(\underline{x}) = g^*(w_{j0} + \sum_{j=1}^{Hidden} a_j(\underline{x}) \cdot w_{ji})$$

Forward pass

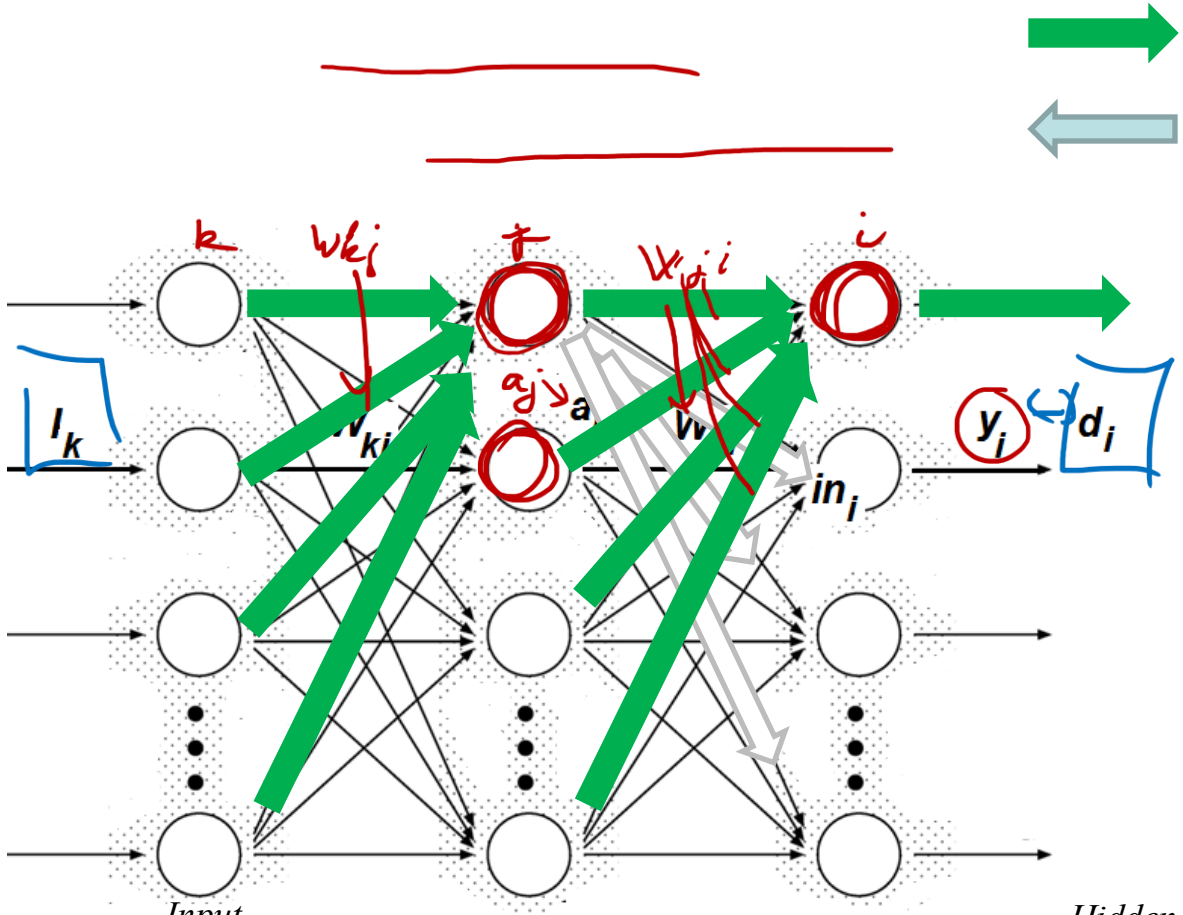
- Forward pass



$$a_j(\underline{x}) = g(w_{k0} + \sum_{k=1}^{Input} x_k \cdot w_{kj}) \qquad y_i(\underline{x}) = g^*(w_{j0} + \sum_{j=1}^{Hidden} a_j(\underline{x}) \cdot w_{ji})$$

Forward pass

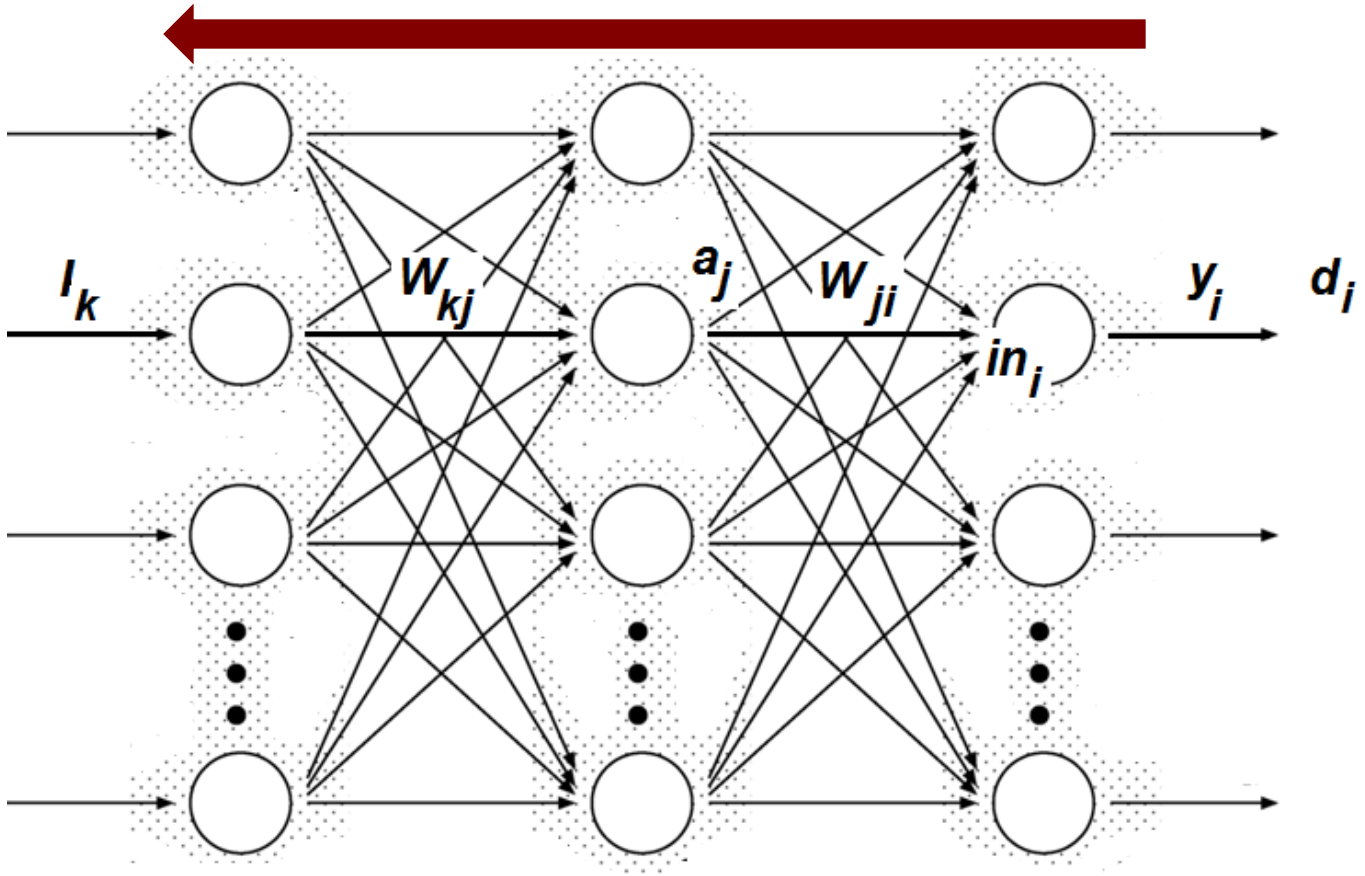
- Forward pass



$$a_j(\underline{x}) = g(w_{k0} + \sum_{k=1}^{Input} x_k \cdot w_{kj})$$

$$y_i(\underline{x}) = g^*(w_{j0} + \sum_{j=1}^{Hidden} a_j(\underline{x}) \cdot w_{ji})$$

Backpropagation



$$W_{k,j} \leftarrow W_{k,j} - \alpha \frac{\partial E}{\partial W_{k,j}} \qquad W_{j,i} \leftarrow W_{j,i} - \alpha \frac{\partial E}{\partial W_{j,i}} \qquad E = \frac{1}{2}\sum_i (d_i - y_i)^2$$

Backpropagation

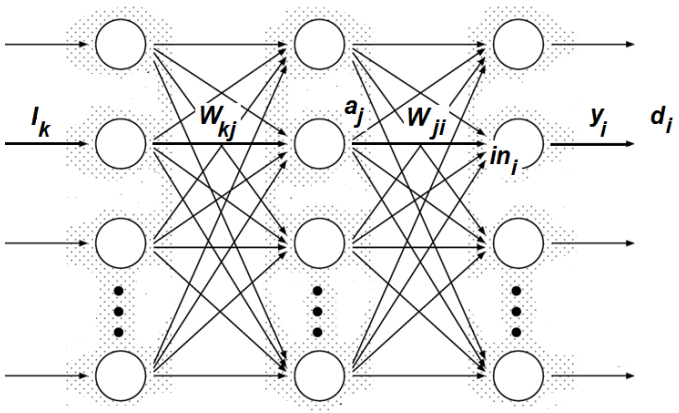
$$E = \frac{1}{2}\sum_i (d_i - y_i)^2 = \frac{1}{2}\sum_i Err_i \qquad W_{j,i} \leftarrow W_{j,i} - \alpha \frac{\partial E}{\partial W_{j,i}}$$

$$E(\mathbf{W}) = \frac{1}{2}\sum_i (d_i - g(\sum_j W_{j,i}\, a_j))^2 = \frac{1}{2}\sum_i (d_i - g(\sum_j W_{j,i}\, g(\sum_k W_{k,j}\, I_k)))^2$$

$$\frac{\partial E}{\partial W_{j,i}} = -a_j (d_i - y_i) g'(\sum_j W_{j,i}\, a_j) = -a_j (d_i - y_i) g'(in_i) = -a_j\, \Delta_i$$

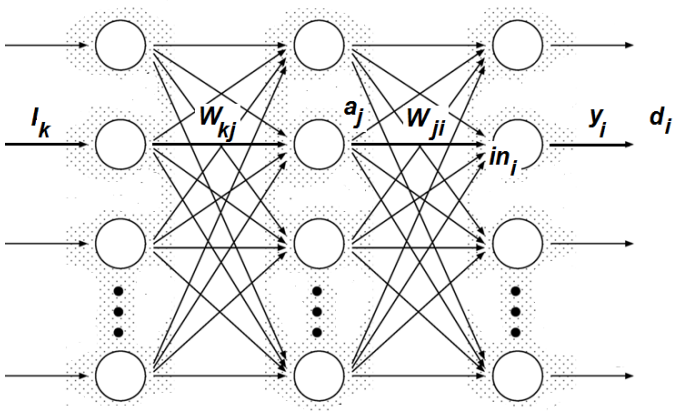
$$W_{j,i} \leftarrow W_{j,i} - \alpha \frac{\partial E}{\partial W_{j,i}} = W_{j,i} + \alpha\, a_j\, Err_i\, g'(in_i)$$

$$\Delta_i = Err_i\, g'(in_i)$$



$$W_{j,i} \leftarrow W_{j,i} + \alpha\, a_j\, \Delta_i$$

Backpropagation



$$\frac{\partial E}{\partial W_{j,i}} = -a_j (d_i - y_i) g'(\sum_j W_{j,i} a_j) = -a_j (d_i - y_i) g'(in_i) = -a_j \Delta_i$$

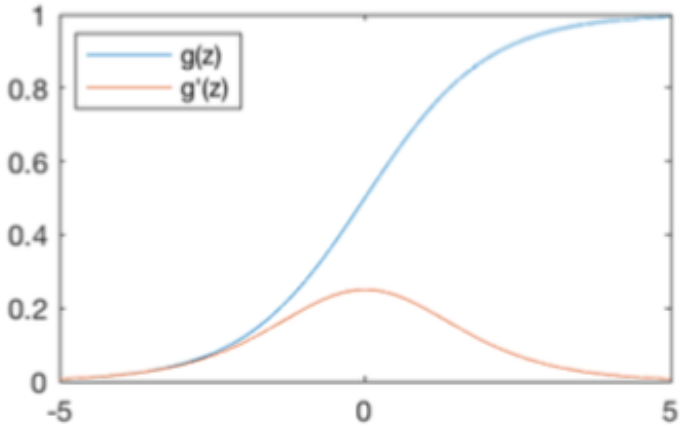
$$\frac{\partial E}{\partial W_{k,j}} = -I_k \Delta_j$$

$$W_{k,j} \leftarrow W_{k,j} - \alpha \frac{\partial E}{\partial W_{k,j}}$$

$$W_{k,j} \leftarrow W_{k,j} + \alpha I_k \Delta_j \qquad \Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i$$

Gyakori aktivációs függvények

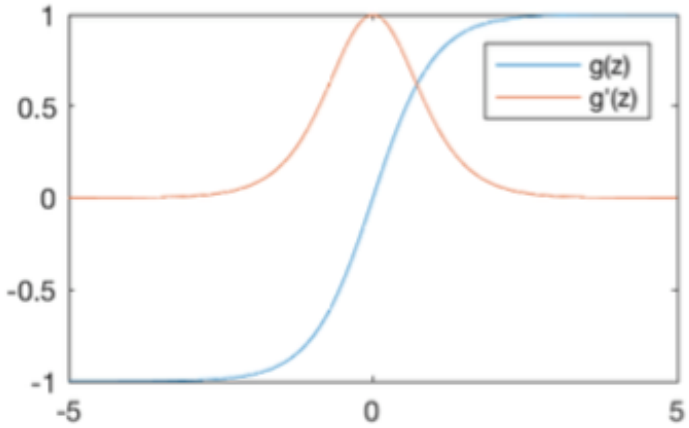
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

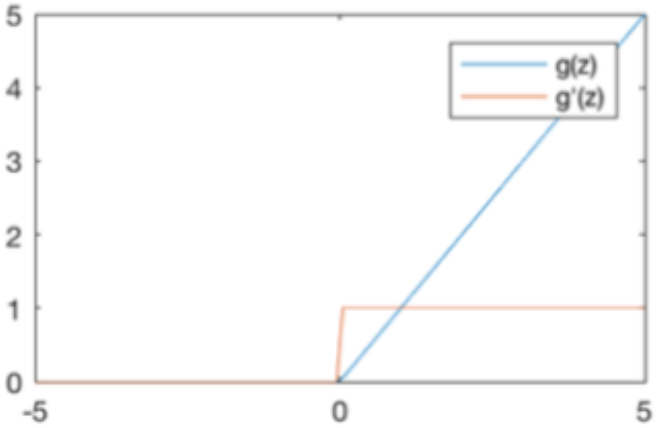
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

[source: MIT 6.S191 introtodeeplearning.com]