

Pattern Recognition IX

Neural Networks

Michal Haindl

Faculty of Information Technology, KTI
Czech Technical University in Prague
Institute of Information Theory and Automation
Czech Academy of Sciences
Prague, Czechia



<http://ro.utia.cz/>



MI/MIE-ROZ 2023-2024

© Haindl

MI-ROZ - 9

1/40

Outline

- 1 Neural Nets in PR
- 2 AI
- 3 Neural Networks
 - Neuron Models
 - Activation Functions
 - Neural Nets Properties
 - Feedback
 - Network Architecture
 - Deep NN
- 4 NN Open Problems
 - Knowledge Representation



© Haindl

MI-ROZ - 9

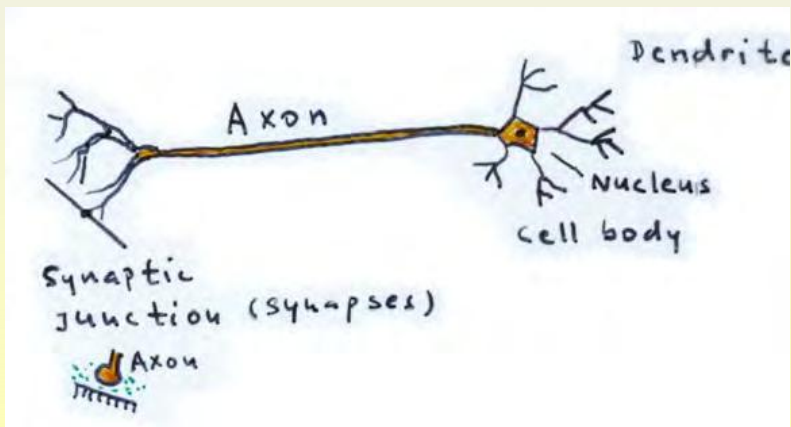
2/40

NN AI NN NN Open Problems

Neural Nets in PR

Motivation:

- human brain study
- complex cognitive tasks



© Haindl

MI-ROZ - 9

3/40

NN AI NN NN Open Problems

Neural Nets in PR 2

Problems:

- Information processing in biological systems is difficult to study.
 - physiological
 - studied since the late Middle Ages
 - structure discovered Santiago Ramon y Cajal (1852-1934, 1906 Nobel prize in medicine) & introduced the idea of neuron
 - chemical transmission of nerve signals
Sir Henry Dale (1936 Nobel prize in medicine)
 - electrical signal transmission in the nervous system
Sir John Eccles, A. L. Hodgkin, A. Huxley (1963 Nobel prize in medicine)
 - philosophical, psychological
 - mathematical analysis & computer simulations



© Haindl

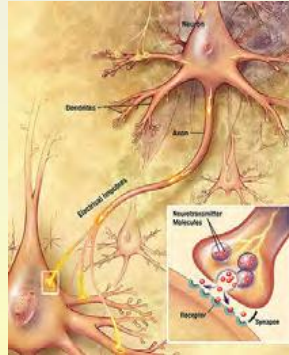
MI-ROZ - 9

4/40

Neural Nets in PR 2

Problems:

- Information processing in biological systems is difficult to study.
 - physiological
 - synaptic transmission



neurotransmitters (gap 20 – 40[nm]):

- dopamin
- noradrenalin
- serotonin

philosophical, psychological



Neural Nets in PR 2

Problems:

- Information processing in biological systems is difficult to study.
 - physiological
 - philosophical, psychological
 - sleep (REM) eliminates undesirable memory
Crick, Mitchison (1983)
 - mathematical analysis & computer simulations



Neural Nets in PR 2

Problems:

- Information processing in biological systems is difficult to study.
 - physiological
 - philosophical, psychological
 - mathematical analysis & computer simulations
 - general theory of inf. processing based on binary switching network (any imaginable computation, learning?)
W. McCulloch, W. Pitts (1943)
 - perceptron type (feed-forward) of NN F. Rosenblatt (1960)
 - Hebb's learning of McCulloch-Pitts neural network
E. Caianiello (1961)
 - analogy between McCulloch-Pitts NN and Ising model
W. Little, J. Hopfield (1978,1982)
 - deep convolutional NN + big data



Neural Nets in PR 2

- Information processing in biological systems is difficult to study.
 - physiological
 - philosophical, psychological
 - mathematical analysis & computer simulations



Neural Nets in PR 2

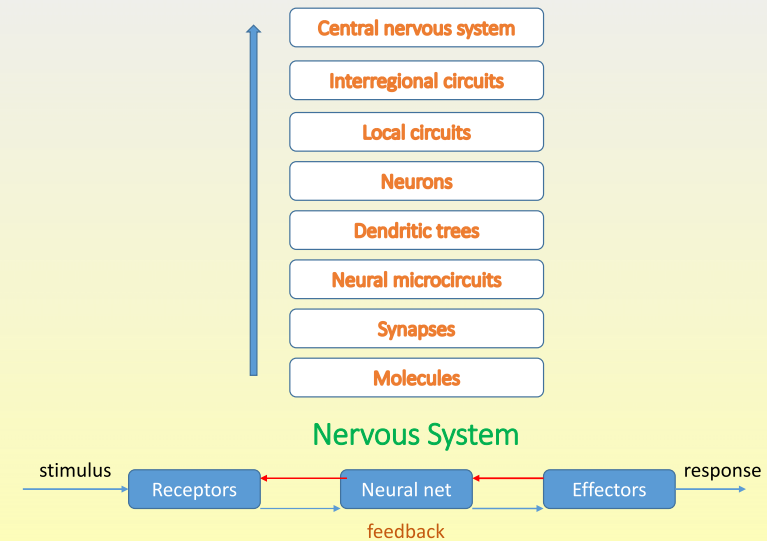
Problems:

- Information processing in biological systems is difficult to study.
 - physiological
 - philosophical, psychological
 - mathematical analysis & computer simulations
- Does the artificial neural net computing paradigm model biology or not?

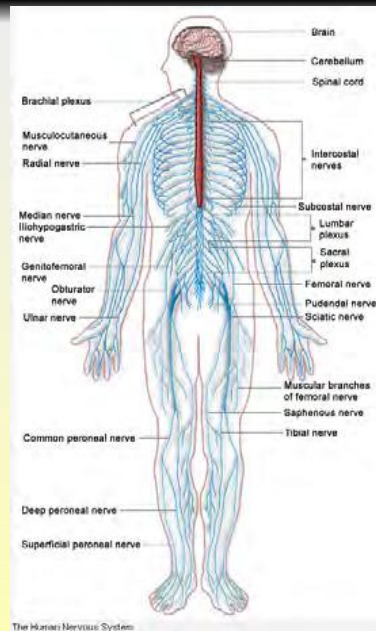


Nervous System

Structural organization levels in the brain



Nervous System



Biological System

- made up of large number (10^{10} , 10^{15} synapses) of interconnected human neurons (Purkyně cell (1839) $8 \cdot 10^4$ synaptical inputs) elemental slow processors
- ⊕ processing focus on relevant information
- ⊕ evaluation based on pattern-ness, similarities, associations, experience
- ⊕ knowledge transmitted either genetically or through learning
- ⊕ individual operations are of lesser significance in learning than spatial-temporal patterns
- ⊕ energy efficiency brain 10^{-16} J per operation per second
best computer 10^{-6} J per op. s^{-1}
AlphaGo 1202 CPUs + 176 GPUs → 1 [MW]
Lee Se-dol (Go world champion) → 20 [W]
- ⊕ face recognition 10^{-1} s
- ⊖ neuron event 10^{-3} s, silicon chip 10^{-9} s



Machine System

- ⊙ rely on speed and accuracy of execution of vast amount of instructions
- ⊖ are easily overwhelmed by task of exponential or greater complexity (most directly approached PR tasks)
- ⊖ search whole system space for acceptable solution
- ⊖ limited progress (little previously unknown knowledge)
- ⊖ few NN-based algorithms superior to those previously known



1st Generation AI - ES

AI 1956 Dartmouth College seminar (John McCarthy - the hypothesis that "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it")

Marvin Minsky MIT,

1950 (+ D. Edmuds) SNARC - first NN architecture → "NN is dead end"

1967 "Within a generation the problem of creating artificial intelligence will be substantially solved."



1st Generation AI - ES

- expert systems - based on formal logical (deductive) reasoning

AI-ES features:

- 1) store knowledge
- 2) apply stored knowledge to solve problems
- 3) acquire new knowledge through experience

ES system components:

- a) representation (symbol structures, declarative, procedural)
- b) reasoning (searching)
- c) learning

- ⊖ logical structure of a problem is known
- ⊖ parallel implementation requires large number of very complex CPU
- ⊖ 1970 first ES decline, late 1980s ES market and research interest crash



2nd Generation AI - NN

- NN (inductive learning)
 - ⊕ parallel implementation requires simple switches and adjustable channels
 - ⊕ error resistivity
 - ⊖ implementation of sufficient number of synapses
 - ⊖ learning (doesn't break problem into logical steps)

- ES × NN

- * Level of Explanation
- * Processing
- * Representation Structure



2nd Generation AI - NN

- NN (inductive learning)
 - ⊕ parallel implementation requires simple switches and adjustable channels
 - ⊕ error resistivity
 - ⊖ implementation of sufficient number of synapses
 - ⊖ learning (doesn't break problem into logical steps)
- ES × NN
 - * Level of Explanation
- ES symbolic representation
- NN parallel distributed processing model
 - * Processing
- ES sequential
- NN parallel → robust
 - * Representation Structure
- ES quasi-linguistic
- NN distributed, ad hoc, difficult to extend



2nd Generation AI - NN

- NN (inductive learning)
 - ⊕ parallel implementation requires simple switches and adjustable channels
 - ⊕ error resistivity
 - ⊖ implementation of sufficient number of synapses
 - ⊖ learning (doesn't break problem into logical steps)
- ES × NN
 - * Level of Explanation
 - * Processing
 - * Representation Structure
 - * Algorithms, Data Representation
- ES top-down
- NN bottom-up



2nd Generation AI - NN

- NN (inductive learning)
 - ⊕ parallel implementation requires simple switches and adjustable channels
 - ⊕ error resistivity
 - ⊖ implementation of sufficient number of synapses
 - ⊖ learning (doesn't break problem into logical steps)
- ES × NN
 - * Level of Explanation
 - * Processing
 - * Representation Structure

1990s cheap computers & big data ⇒ many layers (DNN)

- back-propagation (Paul Werbos 1974, Geoffrey Hinton 1986)
- deep learning (Yann LeCun, Yoshua Bengio)



Neural Nets PR Applications

NN	Algorithm	Functionality	Traditional PR
Unsupervised Learning	• ART 1, 2 • LVQ • Topologically correct mapping	Disconcerting regularities in data Classification	• Data reduction • K-means • Classification
Supervised Learning	• Back propagation • Functional link net	Learning a functional mapping from a set of examples	Non-parametric estimation

ART - adaptive resonance theory 2 layers NN + reset + gain control, LVQ - learning vector quantization

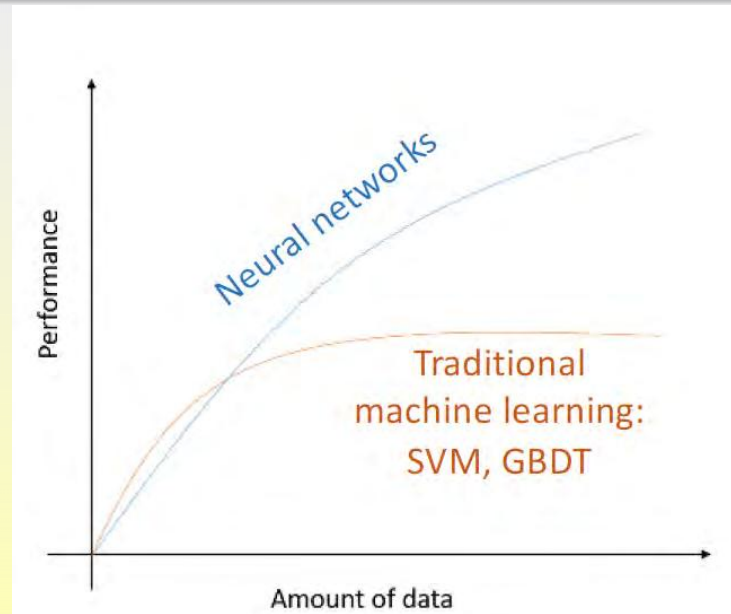


Neural Nets PR Applications 2

NN	Algorithm	Functionality	Traditional PR
Associative Memory	- Hopfield net - ART 1,2,3	Restoration Classification	Distributed matrix assoc. memories
Optimization	Hopfield & Tank approach	Gradient search	Simulated annealing



Neural Nets PR Applications 2



Neural Nets

motivation - biological neurones
neurocomputers, connectionist networks, parallel distributed processors, intelligent computing

- nodes ~ neurones
- ↔ links ~ synapses
- → input node
- ○ output node
- ○ → hidden node (interlayer)

Net properties - activation function f , weights a_j , net architecture



Neural Nets

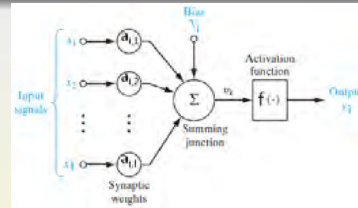
motivation - biological neurones
neurocomputers, connectionist networks, parallel distributed processors, intelligent computing
Net properties - activation function f , weights a_j , net architecture

Neural nets:

- Recurrent - $\exists$ feedback, fixed weights, input pattern X as an initial condition, net converges to the closest template stored in its memory.
- Feed-forward - synaptic connections start with random weights and change during iterative learning, in the classification stage X - input, ω - output.



Neuron Models



$$\begin{pmatrix} x_1 \circ \xrightarrow{a_{1,1}} \\ x_2 \circ \xrightarrow{a_{1,2}} \\ \vdots \\ x_I \circ \xrightarrow{a_{1,I}} \end{pmatrix} \circ \sum \xrightarrow{v_1} f(v_1) \rightarrow y_1$$

$\uparrow \vartheta_i$ bias

$$y_i = f(v_i + \vartheta_i) \quad v_i = \sum_{j=1}^I a_{i,j} x_j$$

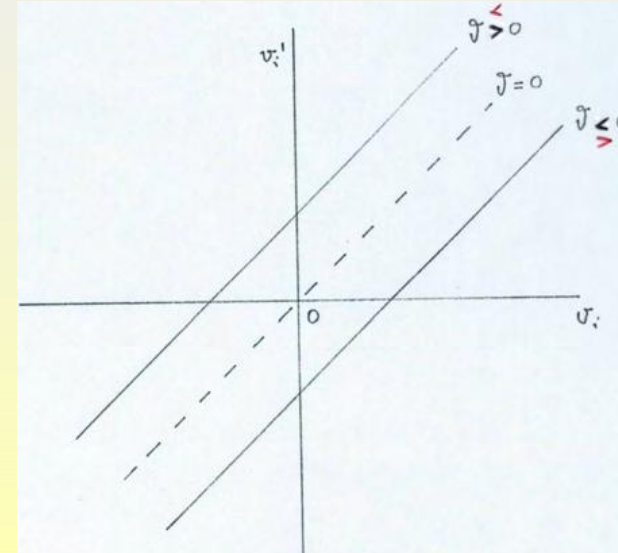
ϑ_i threshold (affine tr. on output)

$x_j \circ \rightarrow$ inputs

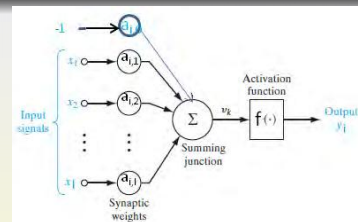


Neuron Models 2

$$y_i = f(v_i - \vartheta_i) \quad y_i = f(v_i + \vartheta_i)$$



Neuron Models 3



• equivalent model

$$\begin{pmatrix} x_0 = -1 \circ \xrightarrow{a_{1,0} = \vartheta_i} \\ x_1 \circ \xrightarrow{a_{1,1}} \\ x_2 \circ \xrightarrow{a_{1,2}} \\ \vdots \\ x_I \circ \xrightarrow{a_{1,I}} \end{pmatrix} \circ \sum \xrightarrow{v_1} f(v_1) \rightarrow y_1$$

$(x_0 = 1)$

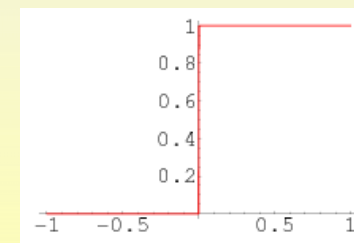
$$v_i = \sum_{j=0}^I a_{i,j} x_j \quad x_0 = -1 \quad a_{i,0} = \vartheta_i \quad y_i = f(v_i)$$



Activation Functions

- monotonic
- except Swish & ReLu saturated
- MN_∞ infinitely wide and deep networks, number of training samples $\rightarrow \infty$
- threshold (MacCulloch-Pitts model, Heaviside function)

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



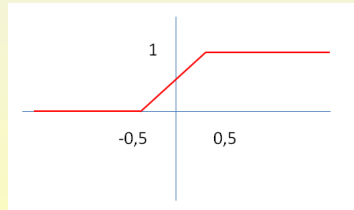
• piece-wise linear function



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function

$$f(v) = \begin{cases} 1 & \text{if } v \geq \frac{1}{2} \\ v & \frac{1}{2} > v > -\frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases}$$



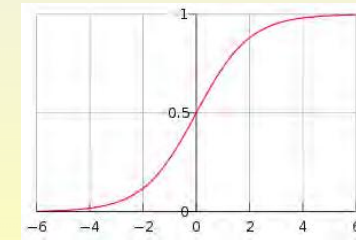
- sigmoid function (strictly increasing, smooth)
 - logistic function a slope par ($NN_\infty \rightarrow$ majority vote classifier).
 - signum function
 - hyperbolic tangent ($NN_\infty \rightarrow$ near consistent on unit circle)
 - rectifier (ReLU)



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
 - logistic function a slope par ($NN_\infty \rightarrow$ majority vote classifier).

$$f(v) = \frac{1}{1 + \exp\{-av\}} \quad \frac{\partial f(v)}{\partial v} = a f(v)(1 - f(v))$$



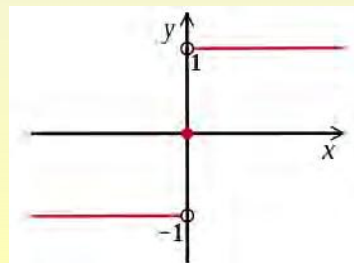
- signum function
- hyperbolic tangent ($NN_\infty \rightarrow$ near consistent on unit circle)
- rectifier (ReLU)
 - ReLU ($NN_\infty \rightarrow$ majority vote classifier)



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
 - logistic function a slope par ($NN_\infty \rightarrow$ majority vote classifier).
 - signum function

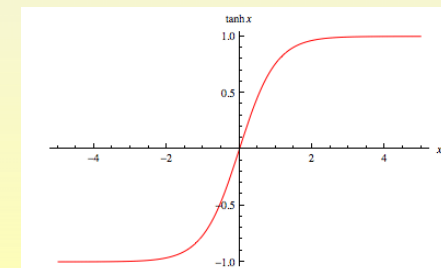
$$f(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases}$$



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
 - logistic function a slope par ($NN_\infty \rightarrow$ majority vote classifier).
 - signum function
 - hyperbolic tangent ($NN_\infty \rightarrow$ near consistent on unit circle)

$$f(v) = \tanh\left(\frac{v}{2}\right) = \frac{1 - \exp\{-v\}}{1 + \exp\{-v\}} \quad \frac{\partial f(v)}{\partial v} = \frac{1 - f^2(v)}{2}$$



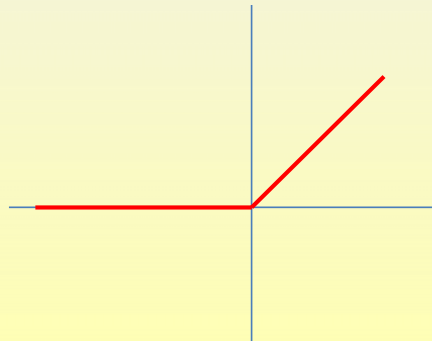
- rectifier (ReLU)



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
- rectifier (ReLU)
 - ReLu ($NN_{\infty} \rightarrow$ majority vote classifier)

$$f(v) = \begin{cases} 0 & \text{for } v < 0 \\ v & \text{for } v \geq 0 \end{cases}$$



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
- rectifier (ReLU)
 - ReLu ($NN_{\infty} \rightarrow$ majority vote classifier)
 - leaky rectifier (ReLU)

$$f(v) = \begin{cases} 0.01 & \text{for } v < 0 \\ v & \text{for } v \geq 0 \end{cases}$$

- parametric leaky rectifier (ReLU)
- Hard tanh ($NN_{\infty} \rightarrow$ near consistent on unit circle)



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
- rectifier (ReLU)
 - ReLu ($NN_{\infty} \rightarrow$ majority vote classifier)
 - leaky rectifier (ReLU)
 - parametric leaky rectifier (ReLU)

$$f(v) = \begin{cases} \alpha & \text{for } v < 0 \\ v & \text{for } v \geq 0 \end{cases}$$

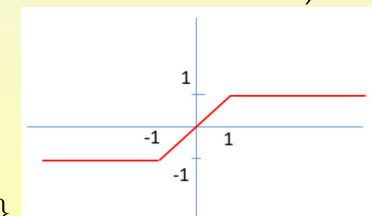
α is learned during NN training

- Hard tanh ($NN_{\infty} \rightarrow$ near consistent on unit circle)



Activation Functions

- threshold (MacCulloch-Pitts model, Heaviside function)
- piece-wise linear function
- sigmoid function (strictly increasing, smooth)
- rectifier (ReLU)
 - ReLu ($NN_{\infty} \rightarrow$ majority vote classifier)
 - leaky rectifier (ReLU)
 - parametric leaky rectifier (ReLU)
- Hard tanh ($NN_{\infty} \rightarrow$ near consistent on unit circle)



$$f(v) = \max\{\min[v, 1], -1\}$$

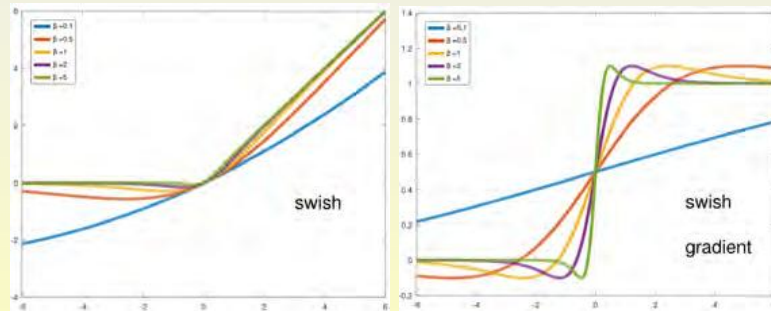


Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)

$$f(v) = \frac{v}{1 + \exp\{-\beta v\}}$$

$$\frac{\partial f(v)}{\partial v} = \frac{f(v)}{v} (1 + f(v)\beta \exp\{-\beta v\})$$



- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit

$$f(v) = \frac{v}{2} \left[1 + \operatorname{erf} \left(\beta \frac{v}{\sqrt{2}} \right) \right]$$

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\{-t^2\} dt$$

- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH
- softplus
- TanhExp
- activation functions

gradients



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit

$$f(v) = \begin{cases} v, & v \geq 0; \\ \beta \left(\exp\left\{\frac{v}{\beta}\right\} - 1 \right), & v < 0. \end{cases}$$

- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH
- softplus
- TanhExp

activation functions

gradients



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)

$$f(v) = \begin{cases} \alpha v, & v > 0; \\ \beta \exp\{v\} - \beta, & v \leq 0. \end{cases}$$

- MISH
- softplus
- TanhExp

activation functions

gradients



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH

$$f(v) = v \tanh [\log(1 + \exp\{\beta v\})]$$

- softplus
- TanhExp
- activation functions

gradients



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH
- softplus

$$f(v) = \frac{1}{\beta} \log(1 + \exp\{\beta v\})$$

- TanhExp

- activation functions

gradients



Activation Functions 2

- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH
- softplus
- TanhExp

$$f(v) = v \tanh [\exp\{\beta v\}]$$

- activation functions

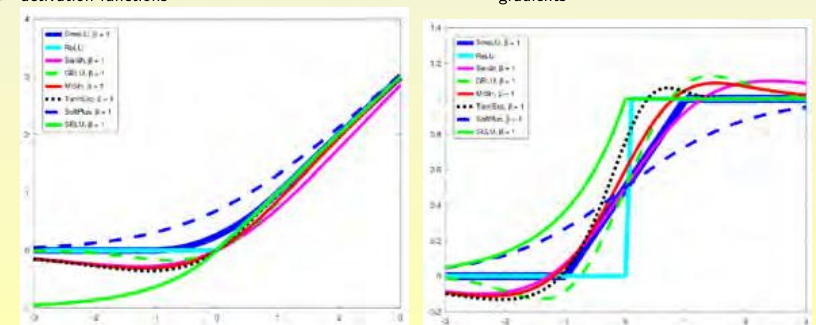
gradients



Activation Functions 2

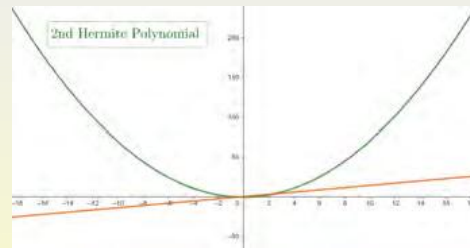
- Swish function ($NN_\infty \rightarrow$ majority vote classifier)
- GELU - Gaussian Error Linear Unit
- CELU - Continuously differentiable Exponential Linear Unit
- SELU - Scaled Exponential Linear Unit (ELU $\alpha = 1$)
- MISH
- softplus
- TanhExp
- activation functions

gradients



Activation Functions 3

- 2nd Hermite Polynomial



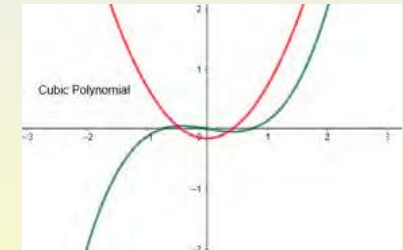
$$f(v) = \frac{v^2 - 1}{\sqrt{2}}$$

- Cubic Polynomial
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)
- Cosid ($NN_\infty \rightarrow$ majority vote classifier)
- Sin ($a = 1$ $NN_\infty \rightarrow$ near consistent on unit circle)
- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



Activation Functions 3

- 2nd Hermite Polynomial
- Cubic Polynomial



$$f(v) = \frac{v^3 + (\sqrt{6} - 3)v}{\sqrt{12}}$$

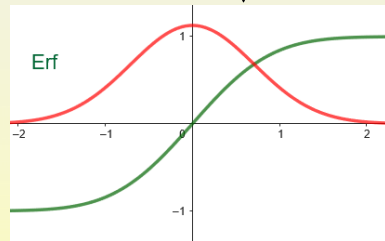
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)
- Cosid ($NN_\infty \rightarrow$ majority vote classifier)
- Sin ($a = 1$ $NN_\infty \rightarrow$ near consistent on unit circle)
- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



Activation Functions 3

- 2nd Hermite Polynomial
- Cubic Polynomial
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)

$$f(v) = \text{erf}(v) = \frac{2}{\sqrt{\pi}} \int_0^v \exp\{-t^2\} dt$$

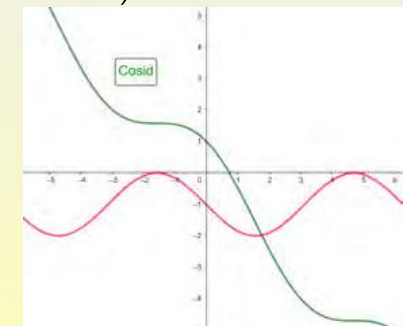


- Cosid ($NN_\infty \rightarrow$ majority vote classifier)
- Sin ($a = 1$ $NN_\infty \rightarrow$ near consistent on unit circle)
- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



Activation Functions 3

- 2nd Hermite Polynomial
- Cubic Polynomial
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)
- Cosid ($NN_\infty \rightarrow$ majority vote classifier)



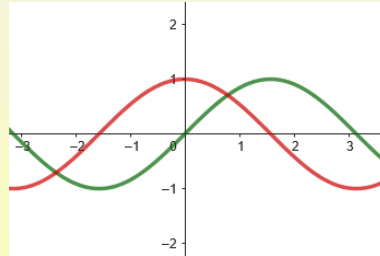
$$f(v) = \cos(v) - v$$

- Sin ($a = 1$ $NN_\infty \rightarrow$ near consistent on unit circle)
- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



Activation Functions 3

- 2nd Hermite Polynomial
- Cubic Polynomial
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)
- Cosid ($NN_\infty \rightarrow$ majority vote classifier)
- Sin ($a = 1 \text{ } NN_\infty \rightarrow$ near consistent on unit circle)



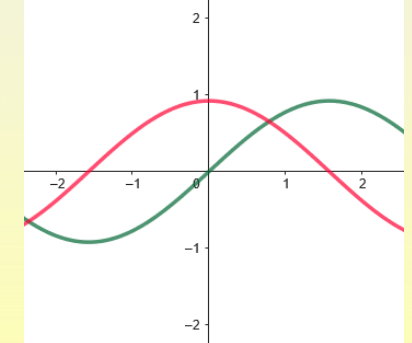
$$f(v) = \sin(av)$$

- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



Activation Functions 3

- 2nd Hermite Polynomial
- Cubic Polynomial
- Erf ($NN_\infty \rightarrow$ near consistent on unit circle)
- Cosid ($NN_\infty \rightarrow$ majority vote classifier)
- Sin ($a = 1 \text{ } NN_\infty \rightarrow$ near consistent on unit circle)
- infinite deep & wide $NN_\infty \rightarrow$ Bayes optimal classifier



$$f(v) = \frac{\sin(av)}{\sqrt{\sinh(a^2)}}$$



Activation Functions 4

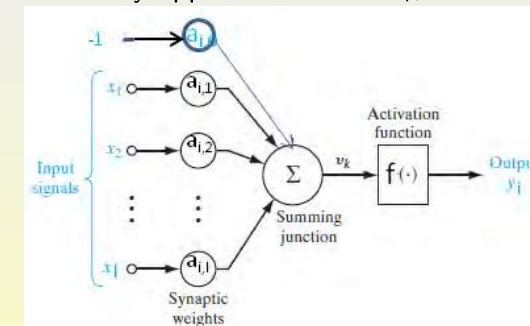
- ⊖ discontinuous gradient $\rightarrow$ several local optima, different training trajectories and predictions
- ⊕ smooth activations are superior to ReLU
- ⊙ $\beta \searrow$ accuracy traded for reproducibility

Mikes, S. - Haindl, M.: Optimal Activation Function for Anisotropic BRDF Modeling, Lisbon, Portugal, ISSN 2184-4321, ISBN 978-989-758-634-7, DOI 10.5220/0011616200003417, pp. 162 -169, 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, GRAPP 2023, SciTePress.



Neural Nets Properties

- 1 Each neuron is represented by a set of linear synaptic links, externally applied threshold ϑ_i , nonlinear activation link $f(\cdot)$.



- 2 The synaptic links of a neuron weight their resp. input signals.
- 3 The weighted sum of input signals defines the total internal activity of the neuron.
- 4 The activation link transforms the internal activity into an output (state variable of the neuron).



Neural Nets Properties 2

- ⊕ massive parallel distributed structure
- ⊕ supervised learning
- ⊕ nonlinearity (neuron is a nonlinear device)
- ⊖ input-output mapping
- ⊕ adaptivity
- ⊕ contextual behaviour of neurons
- ⊕ fault tolerance (fault neuron → impaired quality only)
- ⊕ uniformity of analysis and design
neurons common to all NN
learning can be shared in different applications
seamless integration of NN models



NN Signal-Flow Graph

- 1 rule a signal flows only in the direction → links

- synaptic $x_j \circ \xrightarrow{a_{k,j}} \circ y_k = a_{k,j} x_j$ (linear)

- activation $x_j \circ \xrightarrow{f()} \circ y_k = f(x_j)$ (nonlinear)

- 2 rule $\circ$ signal = $\sum$ signals from incoming links

$$\begin{matrix} y_i \\ y_j \end{matrix} \searrow \nearrow \circ y_k = y_i + y_j \text{ (synaptic convergence, fan-in)}$$

- 3 rule $\circ$ signal is transmitted independently on $f_i()$ of targets

$$x_j \circ \begin{matrix} \nearrow \\ \searrow \end{matrix} \begin{matrix} x_j \\ x_j \end{matrix} \text{ (synaptic divergence, fan-out)}$$



NN Signal-Flow Graph

- 1 rule a signal flows only in the direction → links

- synaptic $x_j \circ \xrightarrow{a_{k,j}} \circ y_k = a_{k,j} x_j$ (linear)

- activation $x_j \circ \xrightarrow{f()} \circ y_k = f(x_j)$ (nonlinear)

- 2 rule $\circ$ signal = $\sum$ signals from incoming links

$$\begin{matrix} y_i \\ y_j \end{matrix} \searrow \nearrow \circ y_k = y_i + y_j \text{ (synaptic convergence, fan-in)}$$

- 3 rule $\circ$ signal is transmitted independently on $f_i()$ of targets

$$x_j \circ \begin{matrix} \nearrow \\ \searrow \end{matrix} \begin{matrix} x_j \\ x_j \end{matrix} \text{ (synaptic divergence, fan-out)}$$



NN Signal-Flow Graph

- 1 rule a signal flows only in the direction → links

- synaptic $x_j \circ \xrightarrow{a_{k,j}} \circ y_k = a_{k,j} x_j$ (linear)

- activation $x_j \circ \xrightarrow{f()} \circ y_k = f(x_j)$ (nonlinear)

- 2 rule $\circ$ signal = $\sum$ signals from incoming links

$$\begin{matrix} y_i \\ y_j \end{matrix} \searrow \nearrow \circ y_k = y_i + y_j \text{ (synaptic convergence, fan-in)}$$

- 3 rule $\circ$ signal is transmitted independently on $f_i()$ of targets

$$x_j \circ \begin{matrix} \nearrow \\ \searrow \end{matrix} \begin{matrix} x_j \\ x_j \end{matrix} \text{ (synaptic divergence, fan-out)}$$



NN Signal-Flow Graph 2

Neural network model is defined as a directed graph:

- A real state variable (input signal) x_i is associated with each $\circ_i$.
- A real-valued (synaptic) weight a_{ik} is associated with each link $\circ_i \leftarrow \circ_k$.
- A real-valued biased ϑ_i (threshold) is associated with each $\circ_i$.
- A transfer function $f_i(x_k, a_{ik}, \vartheta_i, (k \neq i))$ is defined for each $\circ_i$, which determines its state as a function of its bias, incoming links weights and states of connected nodes.



Feedback

output $\rightarrow$ input (recurrent NN)

$$\text{single-loop feedback } x_j(t) \circ \rightarrow \dot{x}_j(t) \circ \xrightarrow[A]{B} \circ y_i(t)$$

$$y_i(t) = A[\dot{x}_j(t)]$$

$$\dot{x}_j(t) = x_j(t) + B[y_i(t)]$$

$$y_i(t) = \frac{A}{1 - AB} [x_j(t)]$$

z^{-1} unit delay



Feedback - Example

$$x_j(t) \circ \rightarrow \dot{x}_j(t) \circ \xrightarrow[a]{z^{-1}} \circ y_i(t)$$

$$\frac{A}{1 - AB} = \frac{a}{1 - az^{-1}} = a \sum_{l=0}^{\infty} a^l z^{-l}$$

$$y_i(t) = a \sum_{l=0}^{\infty} a^l z^{-l} [x_j(t)] = \sum_{l=0}^{\infty} a^{l+1} x_j(t - l)$$

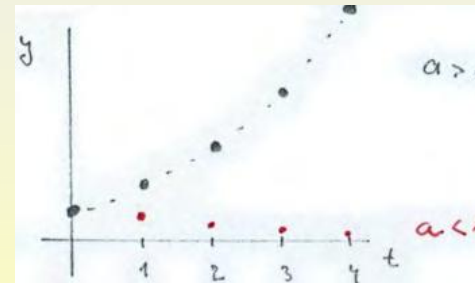
- 1 $|a| < 1 \Rightarrow y_i(t)$ exponentially convergent, stable
- 2 $|a| \geq 1 \Rightarrow y_i(t)$ divergent, system unstable

⊖ difficult dynamic analysis of feedback NN



Feedback - Example

$$x_j(t) \circ \rightarrow \dot{x}_j(t) \circ \xrightarrow[a]{z^{-1}} \circ y_i(t)$$



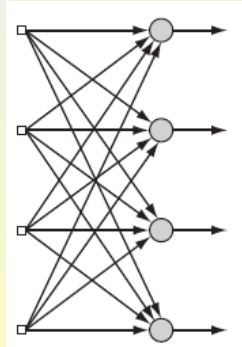
- 1 $|a| < 1 \Rightarrow y_i(t)$ exponentially convergent, stable
- 2 $|a| \geq 1 \Rightarrow y_i(t)$ divergent, system unstable

⊖ difficult dynamic analysis of feedback NN



Network Architecture

- single-layer feedforward network



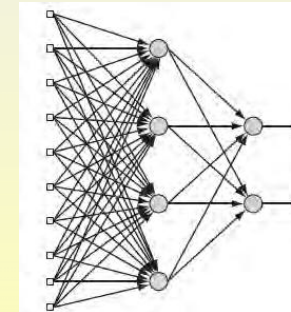
4-4

- multi-layer feedforward network
 - fully connected $\forall o_i \exists o_i \rightarrow o$



Network Architecture

- single-layer feedforward network
- multi-layer feedforward network
 - fully connected $\forall o_i \exists o_i \rightarrow o$

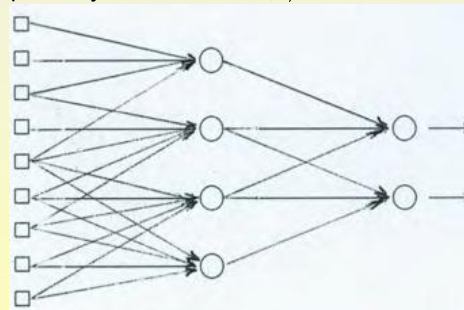


9-4-2



Network Architecture

- single-layer feedforward network
- multi-layer feedforward network
 - fully connected $\forall o_i \exists o_i \rightarrow o$
 - partially connected $\exists o_i \nrightarrow o$



9-4-2

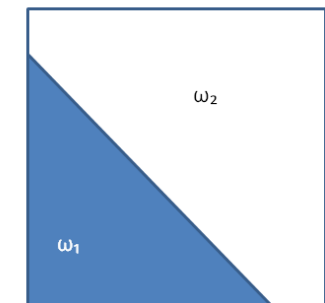


Network Architecture

- single-layer feedforward network
- multi-layer feedforward network
 - fully connected $\forall o_i \exists o_i \rightarrow o$

Multilayer perceptron

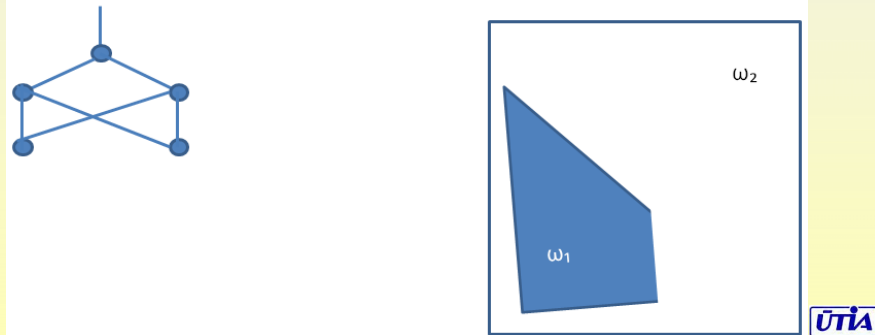
1 layer



Network Architecture

- single-layer feedforward network
- multi-layer feedforward network
 - fully connected $\forall o_i; \exists o_i \rightarrow o$

Multilayer perceptron 2 layers



Network Architecture

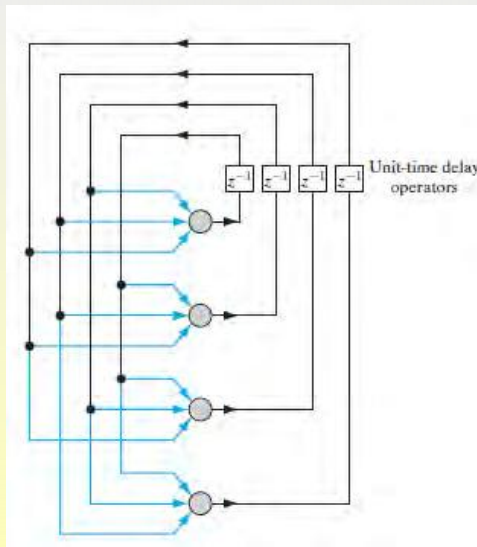
- single-layer feedforward network
- multi-layer feedforward network
 - fully connected $\forall o_i; \exists o_i \rightarrow o$

Multilayer perceptron 3 layers



Network Architecture 2

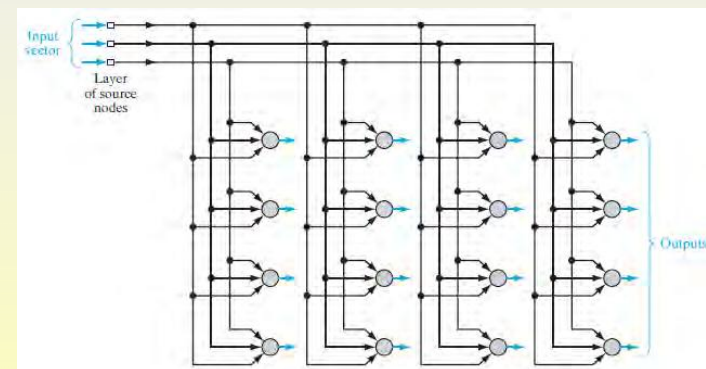
- recurrent network $\exists$ feedback



- lattice

Network Architecture 2

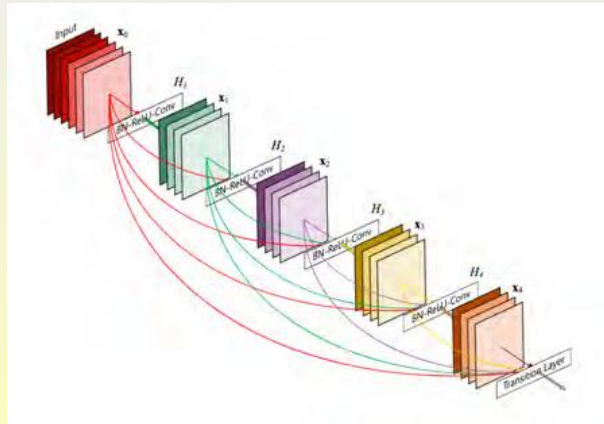
- recurrent network $\exists$ feedback
- lattice



- dense net
- Siamese net

Network Architecture 2

- recurrent network $\exists$ feedback
- lattice



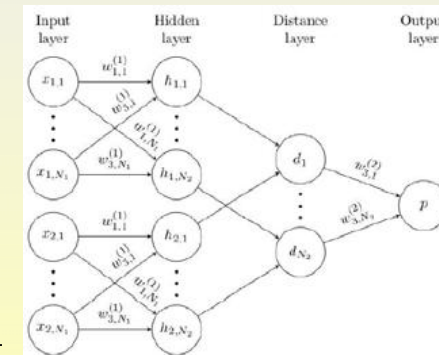
- dense net connects each layer to every other layer in a feed-forward fashion (reduces no. of parameters)

- Siamese net



Network Architecture 2

- recurrent network $\exists$ feedback
- lattice
- dense net



- Siamese net twin networks with distinct inputs, joined by a power function at the top



Network Architecture 3 -GAN

Generative Adversarial Networks

- two simultaneous neural network models:

- generative nn

$$X, \omega_{real} \rightarrow \text{GENnn} \rightarrow \hat{X}, \omega_{synthetic}$$

- discriminative nn

$$\left. \begin{matrix} X, \omega_{real} \\ \hat{X}, \omega_{synthetic} \end{matrix} \right\} \rightarrow \text{DISnn} \rightarrow \begin{cases} \omega_{real} \\ \omega_{synthetic} \end{cases}$$

- GAN training - alternately updating the parameters of the generator and the discriminator



Network Architecture 3 -GAN

Generative Adversarial Networks

- two simultaneous neural network models:

- generative nn

$$X, \omega_{real} \rightarrow \text{GENnn} \rightarrow \hat{X}, \omega_{synthetic}$$

- discriminative nn

$$\left. \begin{matrix} X, \omega_{real} \\ \hat{X}, \omega_{synthetic} \end{matrix} \right\} \rightarrow \text{DISnn} \rightarrow \begin{cases} \omega_{real} \\ \omega_{synthetic} \end{cases}$$

- GAN training - alternately updating the parameters of the generator and the discriminator

GAN:

⊕ if simulation-to-realism gap is small



NN Problems

- huge data** to train
- catastrophic forgetting** new task → forgets all old ones
- unsustainable energy trajectory** computational power required to train the biggest NN systems doubles every 3 months
- black box** performance not understandable, parameters unexplainable
- brittleness** e.g., NN recognize a car, but fails in 97% if the car is on side
- embedded bias** racial / gender etc.
- explainability** poor trust
- Bayes optimality** NN training consistency (to minimize the probability of misclassification for arbitrary data distribution)
- unecological** GPT-3 (Generative Pre-trained Transformer 3) training 502 CO₂ ton = 91 human life years = 8 cars lifetime inc. fuel



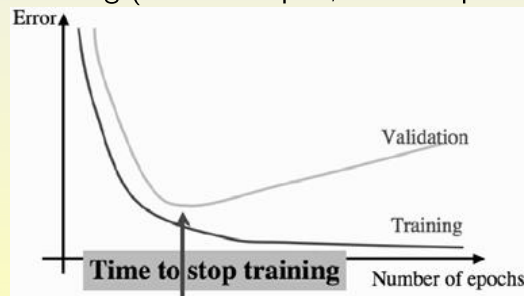
NN Problems -cont

- What is the best architecture (no. of layers, no. of connections, type of activation function, type of updating - synchronous / asynchronous / deterministic / stochastic)?
- Net programming (learning / predesigning)? Learning (no. of samples, no. of repetitions).
- How is a specific information related with the weights?
- What can the various types of network do (how many diff. tasks, how well, how fast, how robust)?
- How can a net be built in hardware?



NN Problems -cont

- What is the best architecture (no. of layers, no. of connections, type of activation function, type of updating - synchronous / asynchronous / deterministic / stochastic)?
- Net programming (learning / predesigning)? Learning (no. of samples, no. of repetitions).



- How is a specific information related with the weights?
- What can the various types of network do (how many diff. tasks, how well, how fast, how robust)?



NN Problems -cont

- What is the best architecture (no. of layers, no. of connections, type of activation function, type of updating - synchronous / asynchronous / deterministic / stochastic)?
- Net programming (learning / predesigning)? Learning (no. of samples, no. of repetitions).
- How is a specific information related with the weights?
- What can the various types of network do (how many diff. tasks, how well, how fast, how robust)?
- How can a net be built in hardware?



NN Problems -cont

- What is the best architecture (no. of layers, no. of connections, type of activation function, type of updating - synchronous / asynchronous / deterministic / stochastic)?
- Net programming (learning / predesigning)?
Learning (no. of samples, no. of repetitions).
- How is a specific information related with the weights?
- What can the various types of network do (how many diff. tasks, how well, how fast, how robust)?
- How can a net be built in hardware?



NN Problems -cont

- What is the best architecture (no. of layers, no. of connections, type of activation function, type of updating - synchronous / asynchronous / deterministic / stochastic)?
- Net programming (learning / predesigning)?
Learning (no. of samples, no. of repetitions).
- How is a specific information related with the weights?
- What can the various types of network do (how many diff. tasks, how well, how fast, how robust)?
- How can a net be built in hardware?



Network Architecture 3

- How many hidden layers are needed?
- How many neurons should be in each layer?

of hidden neurons too small ⇒

⊖ the accurate mapping impossible

of hidden neurons too large ⇒

⊖ more different solutions

⊖ the network will fail in the operational stage



Network Architecture 4

Approaches:

- Start from larger than necessary topology, learn the desired mapping, prune.
 - ⊖ unnecessary large net
 - ⊖ pruning may get stuck in suboptimal net topology
- Start with small network and grow the network until a solution can be found.
 - ⊕ finds the smallest possible net solving the task
 - ⊖ necessary to retrain the complete network after the addition of each single new neuron



Knowledge Representation

knowledge $\rightarrow$ free NN parameters

⊖ knowledge representation inside NN little understood (complicated interactions)

good representation $\Rightarrow$ good solution



Knowledge Representation - Rules

Rules (common-sense):

- 1 Similar inputs (similar classes) $\rightarrow$ similar representation (i.e. classified as belonging to the same category).
- 2 Item classified as different categories should have widely different representation.
- 3 Important feature $\Rightarrow$ a large number of its representation neurons.
- 4 Prior information and invariances should be built into the NN design to simplify the network design (avoiding to learn them).
prior information
 - ⊖ ad-hoc procedures (e.g. locally connected net, constrains on weights)



Knowledge Representation - Rules

Rules (common-sense):

- 1 Similar inputs (similar classes) $\rightarrow$ similar representation (i.e. classified as belonging to the same category).
- 2 Item classified as different categories should have widely different representation.
- 3 Important feature $\Rightarrow$ a large number of its representation neurons.
- 4 Prior information and invariances should be built into the NN design to simplify the network design (avoiding to learn them).
prior information
 - ⊖ ad-hoc procedures (e.g. locally connected net, constrains on weights)

invariance

- invariance by NN structure (e.g. $a_{ij} = a_{ji}$ rotational inv.)
- invariance by training (object in different transformations)
- invariant feature space

