University of Applied Sciences and Arts Northwestern Switzerland
School of Business

Machine Learning: Neural Networks

Motivation: Recognizing Numbers

- It is very hard to specify what makes a «2»

- It is nearly impossible to create or learn symbolic rules.

Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

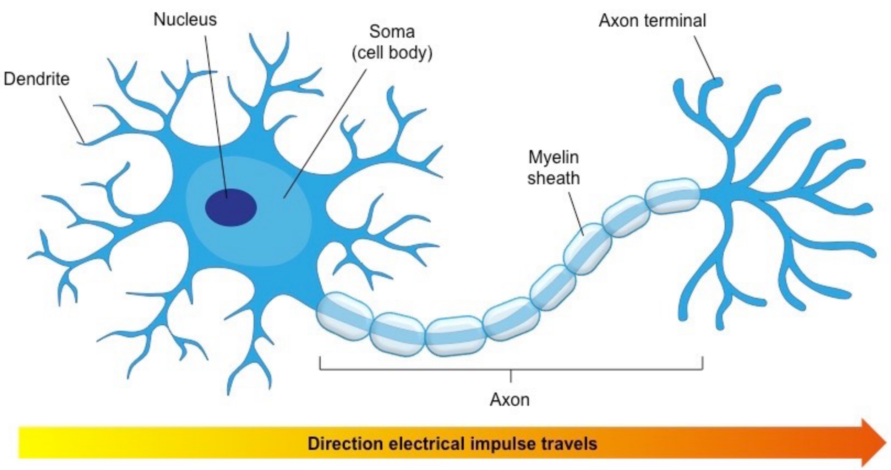
History of Artificial Neural Networks

- Creation:
  - 1890: William James - defined a neuronal process of learning
- Promising Technology:
  - 1943: McCulloch and Pitts - earliest mathematical models
  - 1954: Donald Hebb and IBM research group - earliest simulations
  - 1958: Frank Rosenblatt - The Perceptron
- Disenchantment:
  - 1969: Minsky and Papert - perceptrons have severe limitations
- Re-emergence:
  - 1985: Multi-layer nets that use back-propagation
  - 1986: PDP Research Group - multi-disciplined approach

ANN application areas ...

- Science and medicine: modeling, prediction, diagnosis, pattern recognition
- Manufacturing: process modeling and analysis
- Marketing and Sales: analysis, classification, customer targeting
- Finance: portfolio trading, investment support
- Banking & Insurance: credit and policy approval
- Security: bomb, iceberg, fraud detection
- Engineering: dynamic load schedding, pattern recognition

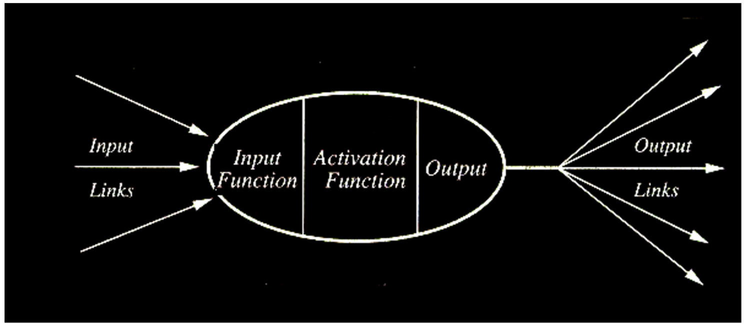
The Neuron - A Biological Information Processor

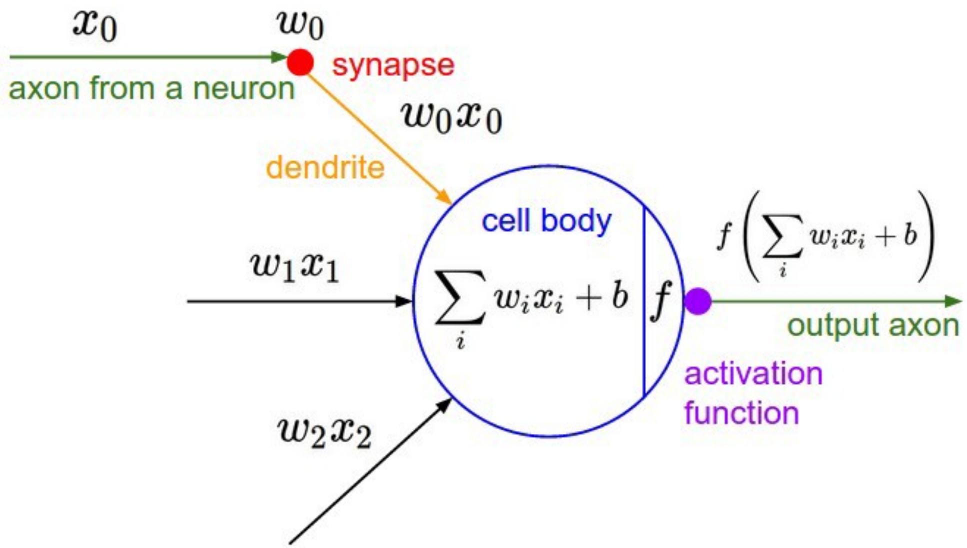


- dentrites - the receivers
- soma - neuron cell body (sums input signals)
- axon - the transmitter
- synapse - point of transmission
- neuron activates after a certain threshold is met
- Learning occurs via electro-chemical changes in effectiveness of synaptic junction.

An Artificial Neuron - The Perceptron

- input connections - the receivers
- *node*, *unit*, or PE simulates neuron body
- output connection - the transmitter
- *activation function* – when is the neuron activated
  - e.g. $f(x) = \begin{cases} 1 & if\ x > \varphi \\ 0 & otherwise \end{cases}$
- *connection weights* act as synaptic junctions
- Learning occurs via changes in value of the connection weights.





Simple Type of Neuron: Linear Neuron

- Simple but computationally limited

$$y = b + \sum_i x_i w_i$$

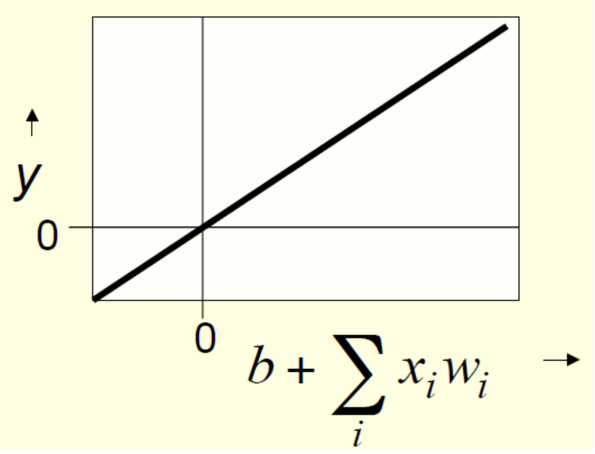
bias

i th input

output

index over input connections

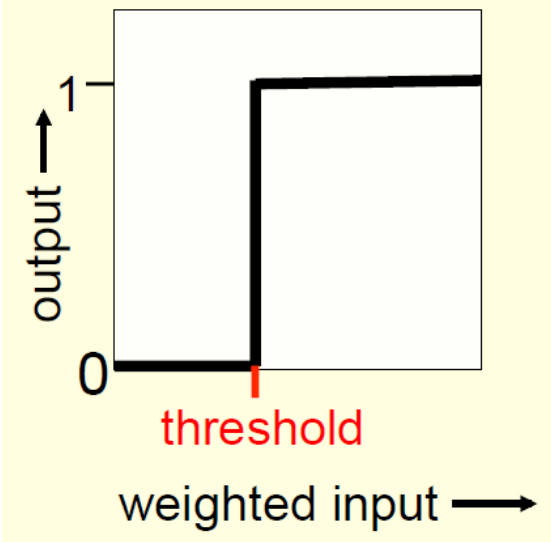
weight on i th input



Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Binary Threshold Neurons

- McCulloch-Pitts (1943)
  - First compute a weighted sum of the inputs.
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
- There are two equivalent ways to write the equations for a binary threshold neuron:



$$z = \sum_i x_i w_i \qquad\qquad z = b + \sum_i x_i w_i$$

$$\theta = -b$$

$$y = \begin{cases} 1 \text{ if } z \geq \theta \\ 0 \text{ otherwise} \end{cases} \qquad\qquad y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

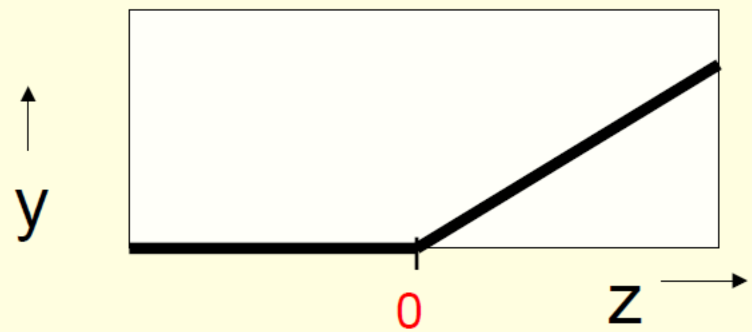
Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

Rectified Linear Neurons

- They compute a *linear* weighted sum of their inputs.
- The output is a ***non-linear*** function of the total input.

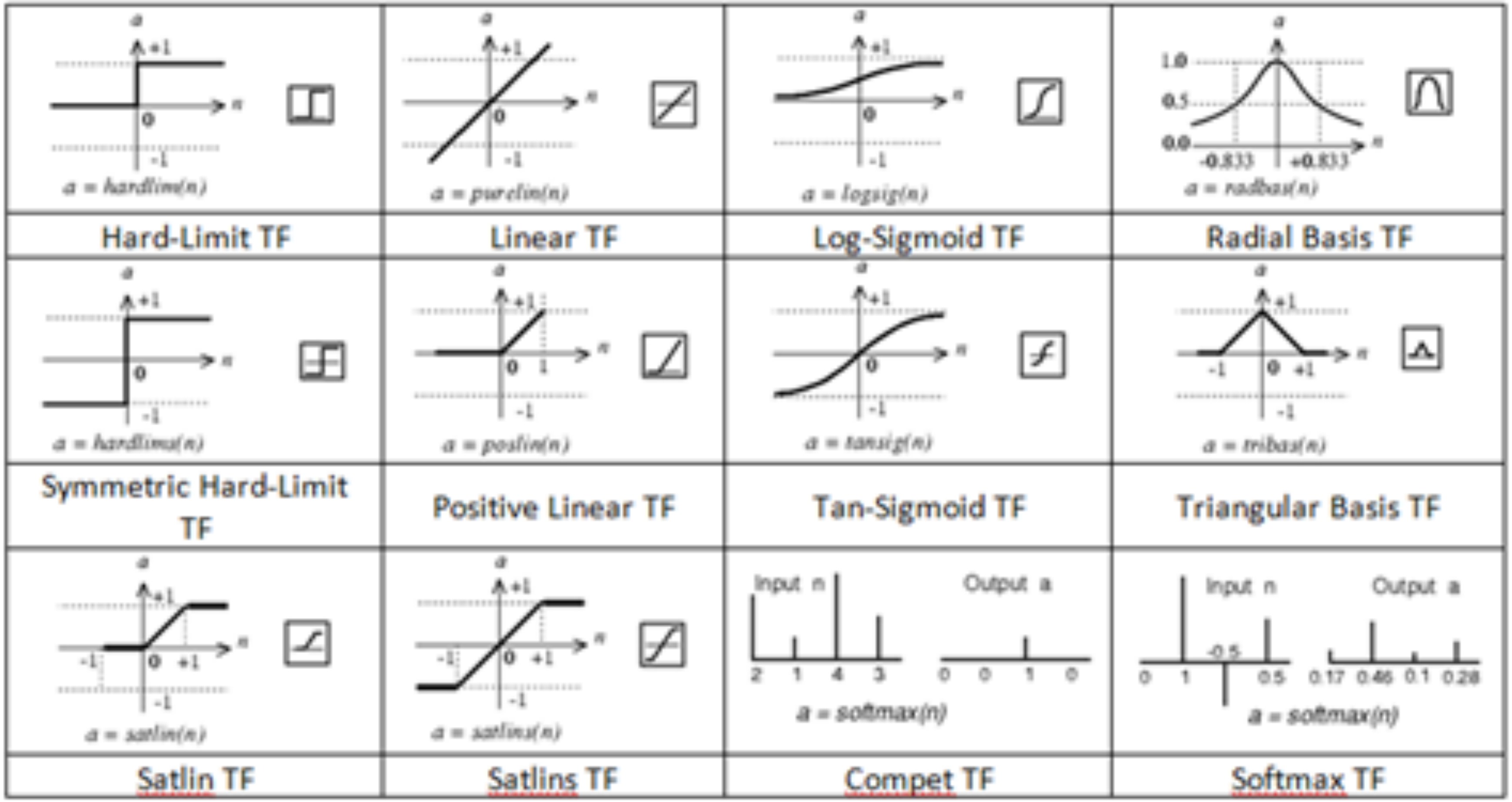
$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$



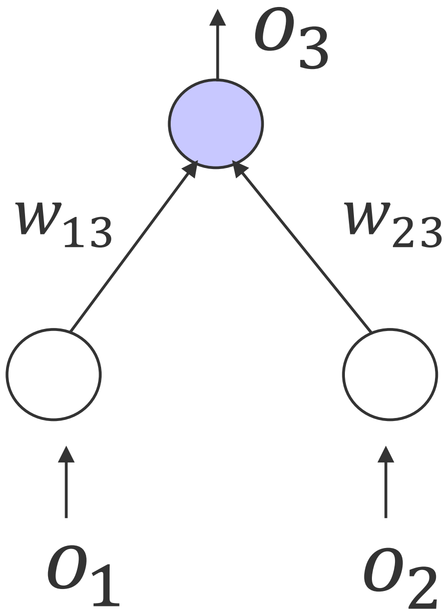
Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

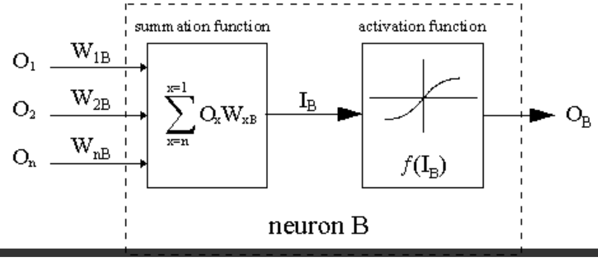
Other activation functions



An Artificial Neuron - The Perceptron

- Basic function of neuron is to sum inputs, and produce output
  - E.g. given sum is greater than threshold (McCulloch-Pitts Neuron)
- ANN node j produces an output as follows:
  1. Multiplies each component of the input pattern o_i by the weight w_{ij} of its connection ($w_{ij}o_i$)
  2. Sums all weighted inputs ($\sum_{i=1}^{n} w_{ij}o_i$)
  3. Transforms the total weighted input into the output using the activation function ($\mathrm{o_j} = f[\sum_{i=1}^{n} w_{ij}o_i]$)

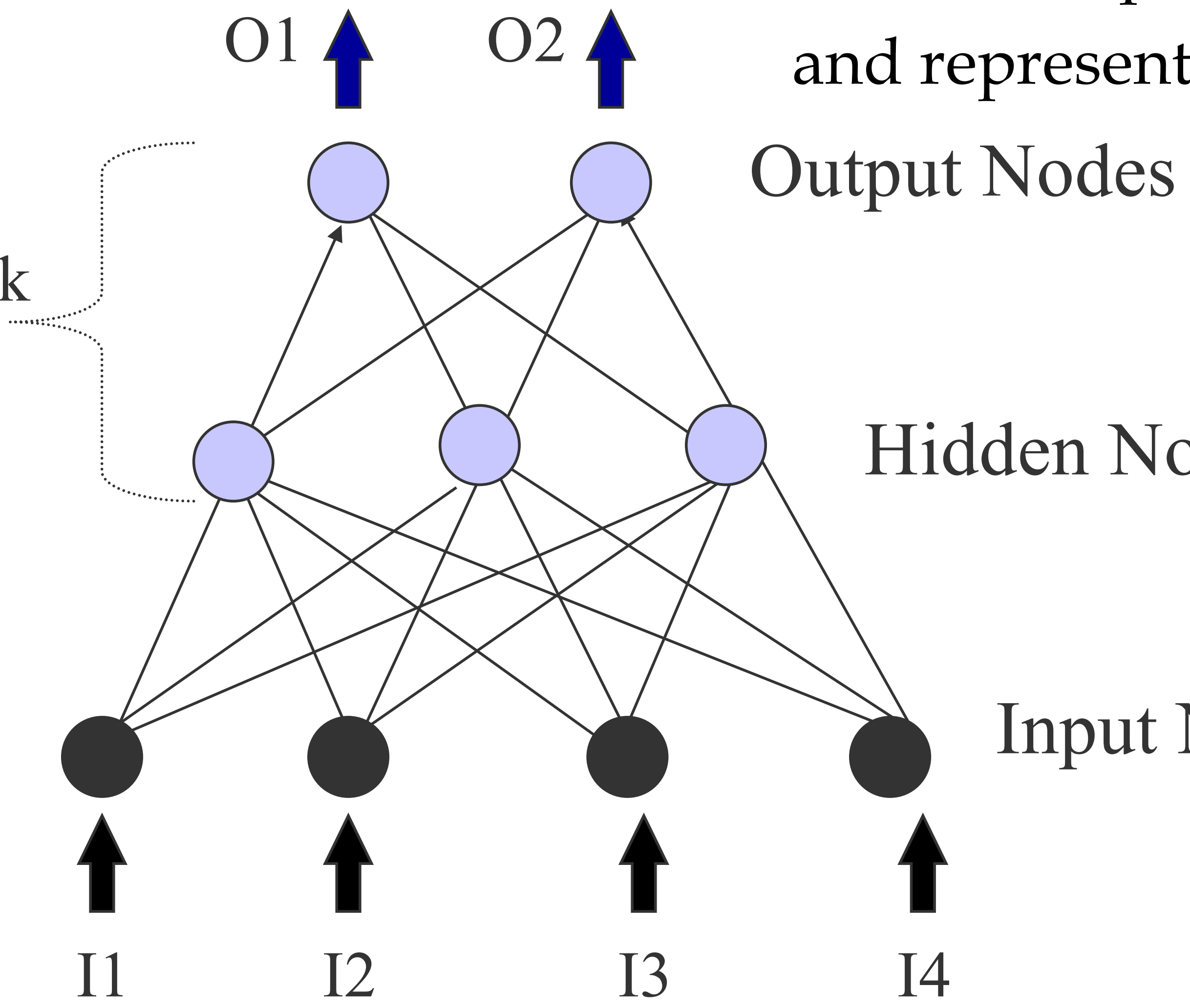




Feed-Forward Networks

"Distributed processing and representation"

3-Layer Network
has
2 active layers



Exercise

- Design a Perceptron with McCulloch-Pitts Neuron with 2 inputs and 1 output neuron(s) which operates an
  - AND
  - OR
  - XOR

Learning: Backpropagation

- **Backward Propagation of Errors,** often abbreviated as BackProp is one of the several ways in which an artificial neural network (ANN) can be trained.
- It is a supervised training scheme, which means, it learns from labeled training data.
- To put in simple terms, BackProp is like "**learning from mistakes**". The supervisor ***corrects*** the ANN whenever it makes mistakes.

The Back-propagation Algorithm

- On-Line algorithm:
  1. Initialize weights
  2. Present a pattern and target output
  3. Compute output : $o_j = f[\sum_{i=0}^{n} w_{ij} o_i]$ where $f[x] = 1/(1 + e^{-x})$
  4. Update weights : $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$

- Repeat starting at 2 until acceptable level of error

Calculating Δw_{ij}

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

- Delta Rule (Widrow-Hoff)

$$\Delta w_{ij} = \eta d_j \sum_{i=1}^{n} w_{ij} o_i$$

where $0 < \eta \leq 1$ is the learning rate (typically set = 0.1)
$d_j = t_j - o_j$ is the error signal,
t_j is the target value, and
o_j is the output value

Calculating Δw_{ij}

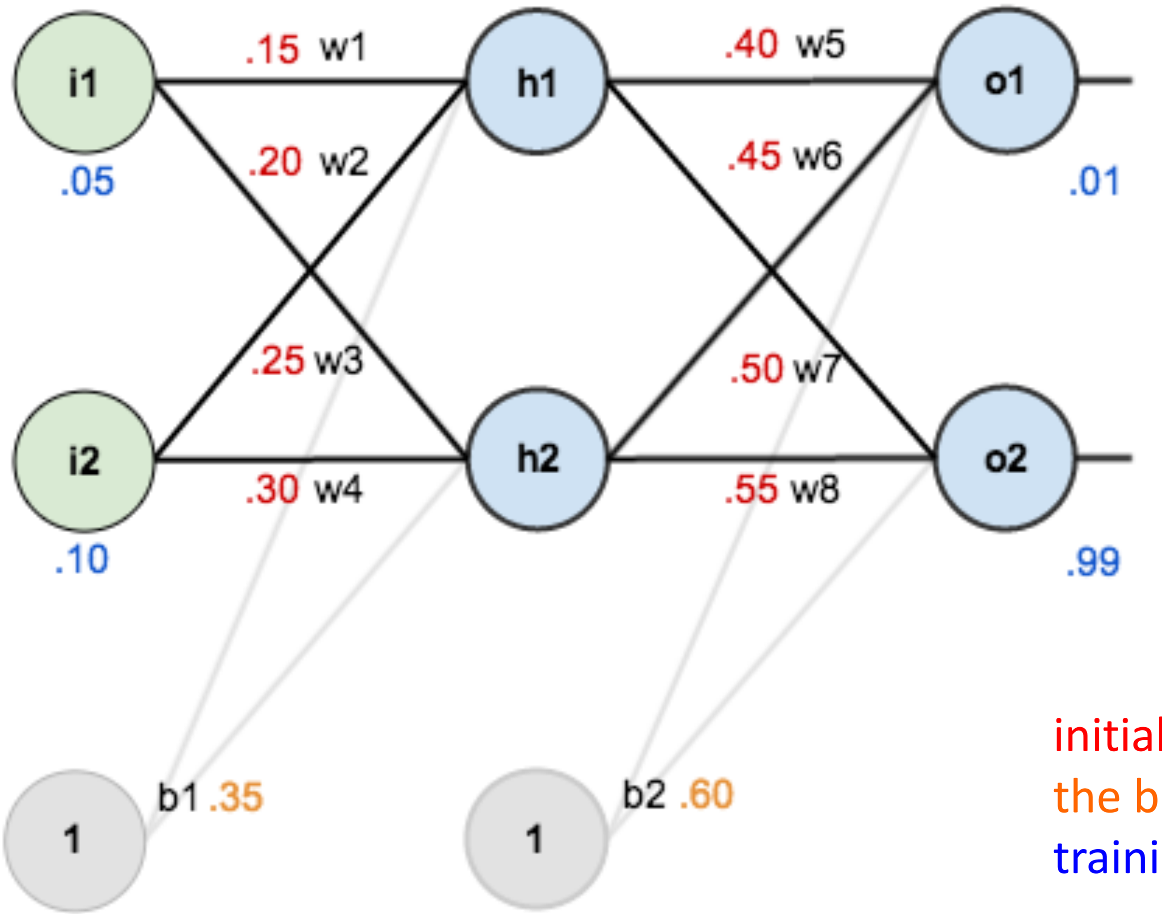
$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

- Backpropagation for sigmoid activation functions:

$$\mathrm{w_{ij}} = \eta \frac{\delta E}{\delta w_{ij}}$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Example Backprop



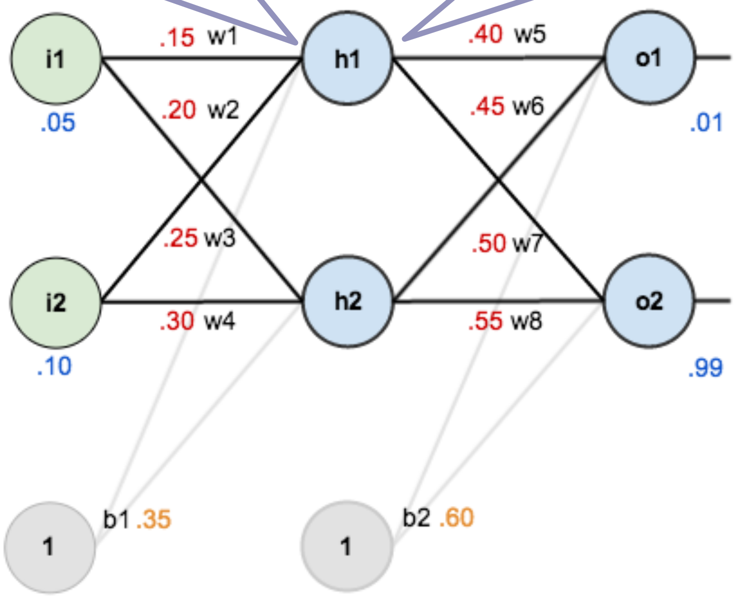
initial weights,
the biases, and
training inputs/outputs

Example Backprop: Forward Pass

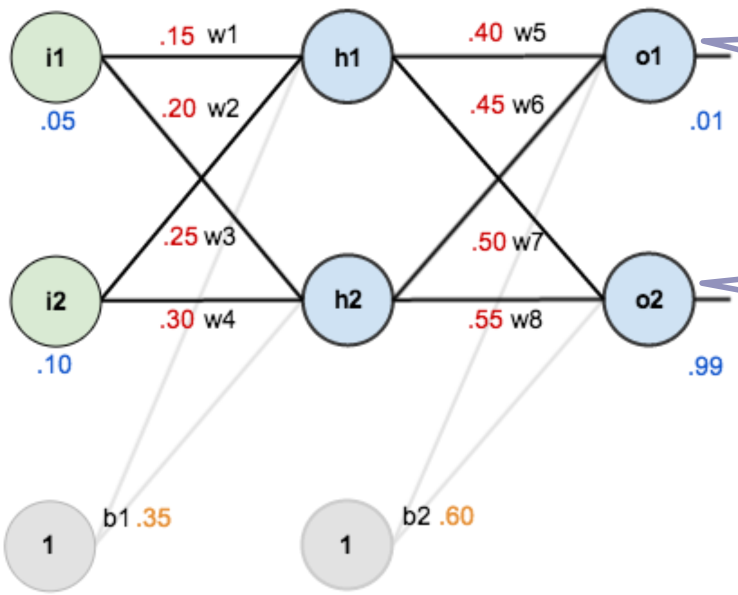
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$



Example Backprop: Forward Pass



$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

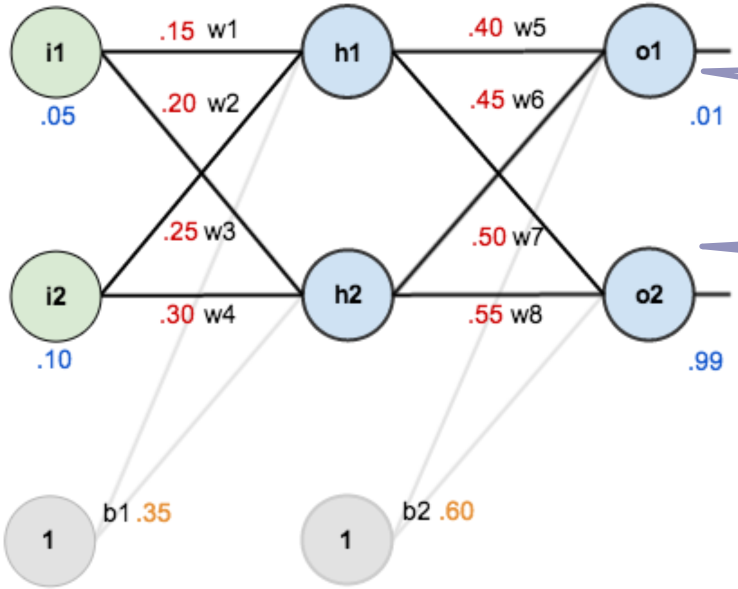
$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$out_{h2} = 0.596884378$$

Example Backprop: Error

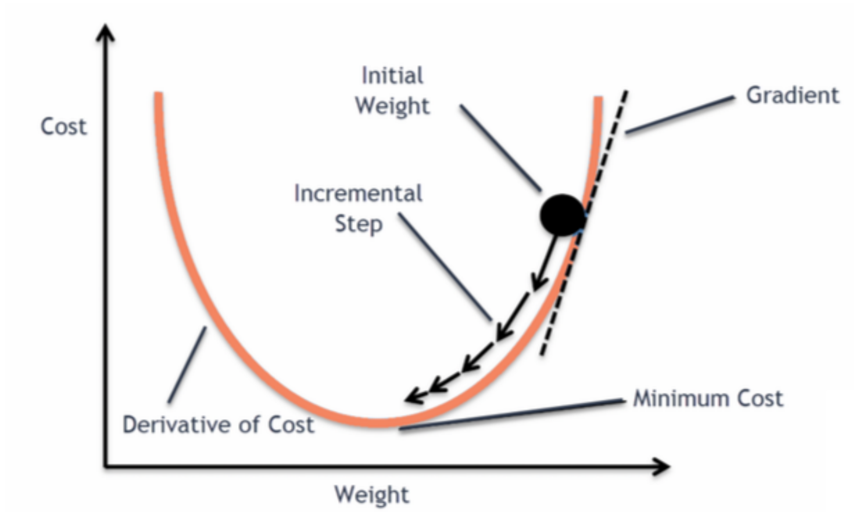
$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$



$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

$$E_{o2} = 0.023560026$$

Example Backprop: Backward Pass

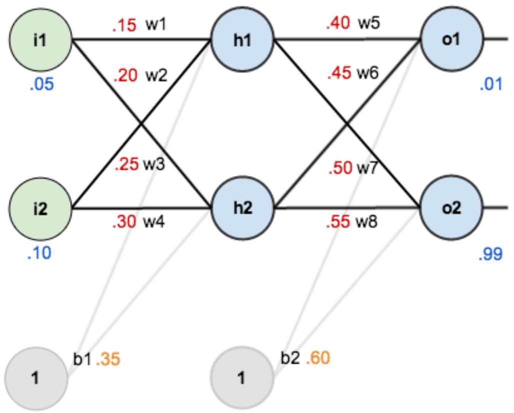


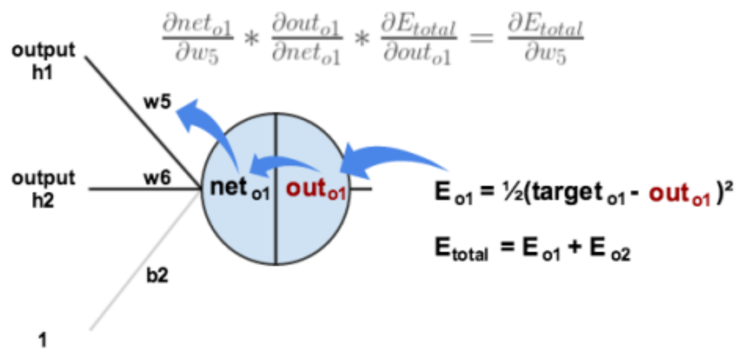
© https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f [19.05.2020]

$$\frac{\partial E_{total}}{\partial w_5}$$

Chain-Rule

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$





Example Backprop: Backward Pass

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

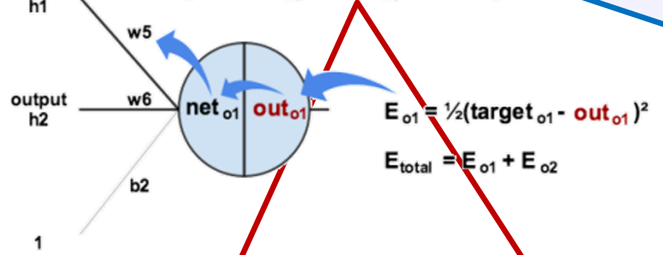
$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$



$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

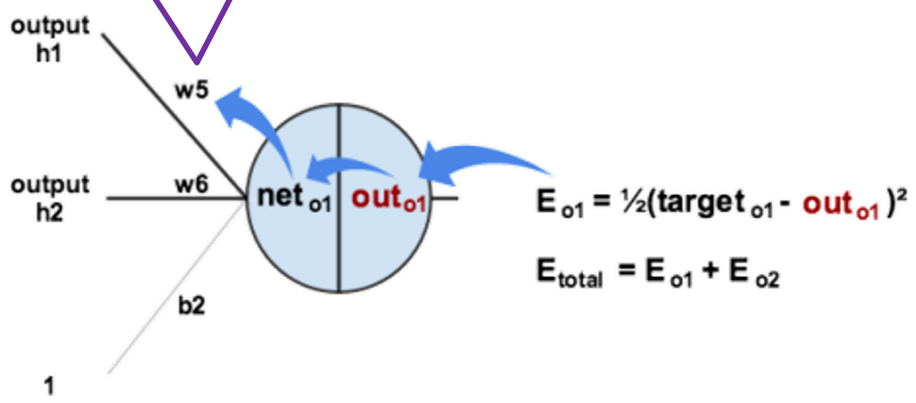
$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Example Backprop: Backward Pass

Learning Rate

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

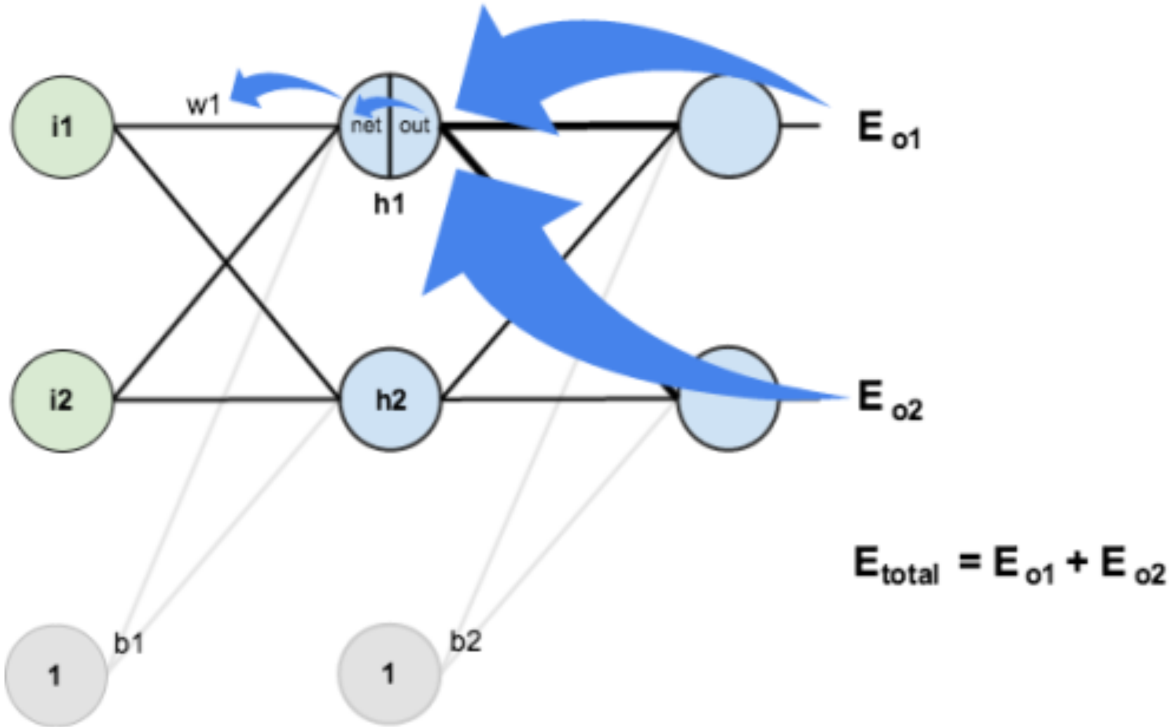


Example Backprop: Backward Pass

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

↓

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Example Backprop: Backward Pass (Hidden Layer)

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

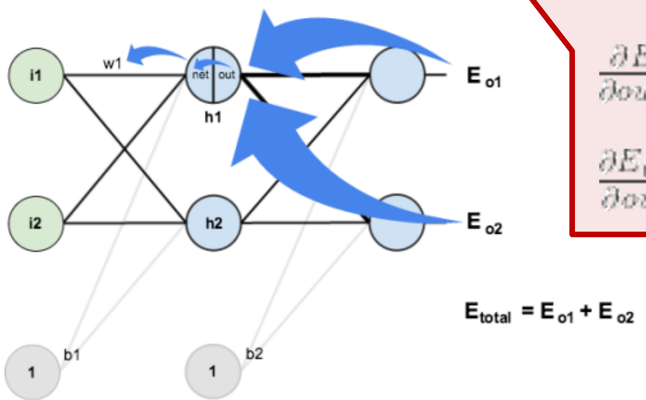
$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$



Example Backprop: Backward Pass (Hidden Layer)

$$net_{h1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

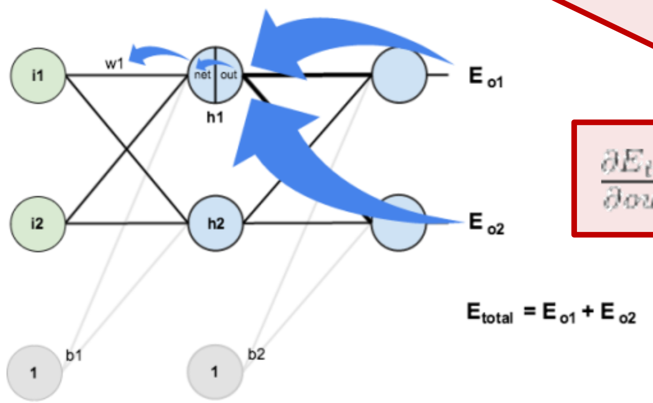
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

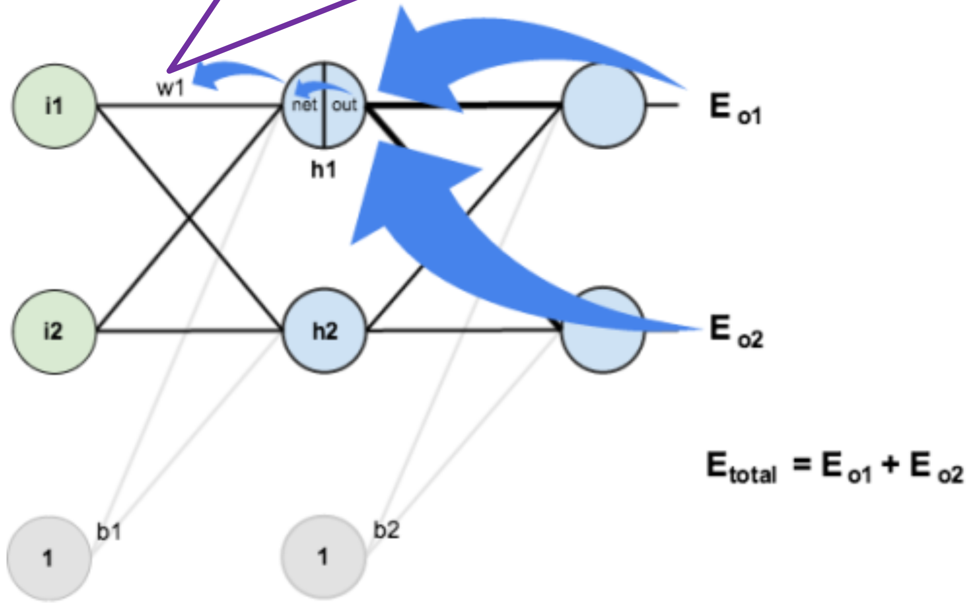
$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

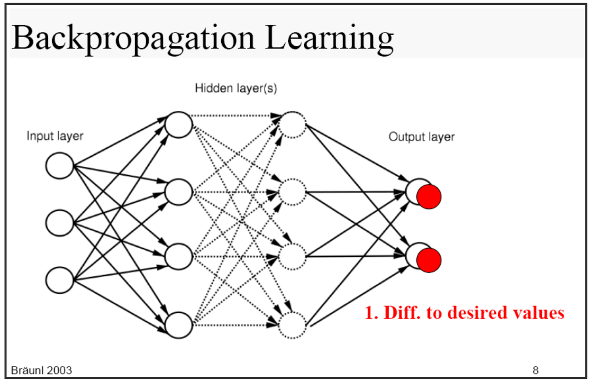


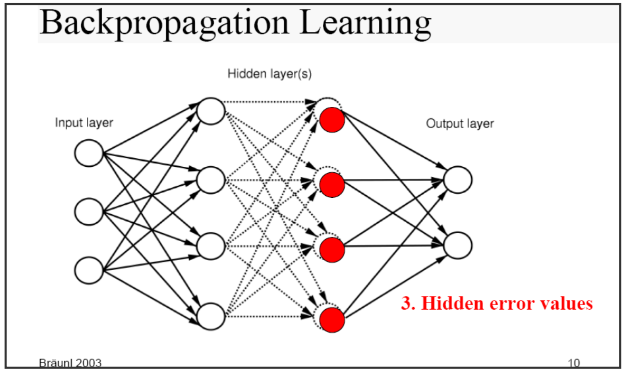
Example Backprop: Backward Pass (Hidden Layer)

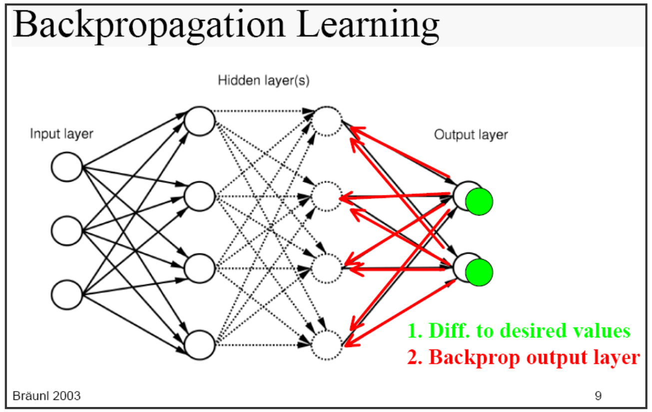
$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

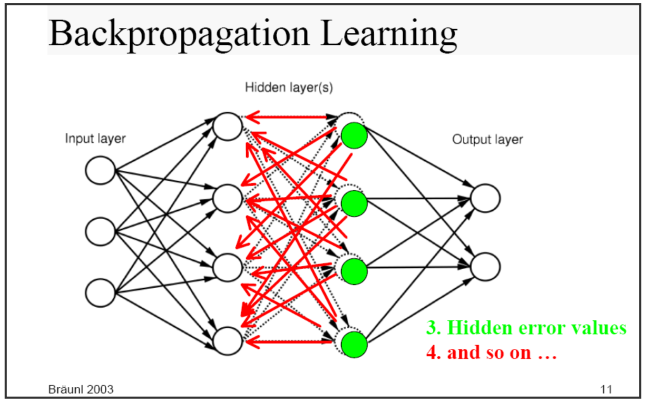


Visualization of the Backprop-Learning









 © http://web.cecs.pdx.edu/~mperkows/CLASS_479/2011.ALL_LECTURES/2011-0480.Neural-Networks.ppt. [04.06.2018]

Generalization – A Probabilistic Guarantee

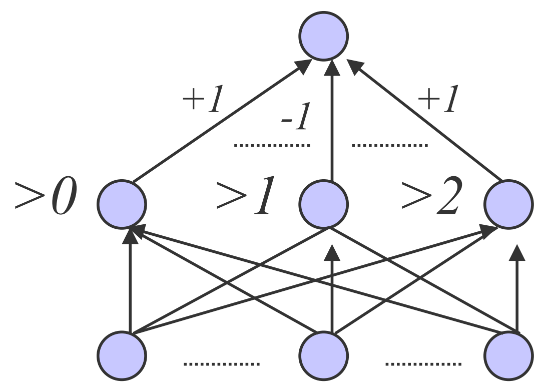
- N = # hidden nodes   m = # training cases
- W = # weights ϵ = error tolerance (< 1/8)
- Network will generalize with 95% confidence if:
  1. Error on training set < $\epsilon/2$
  2. $$m > O\left(\frac{W}{\epsilon}\log_2\frac{N}{\epsilon}\right) \approx m > \frac{W}{\epsilon}$$
- Based on PAC theory → provides a good rule of practice.
- If m is given then hidden nodes can be estimated!

Generalization: 20-bit parity problem

- 20-20-1 net has 441 weights $(n+1)^2 weights$
- For 95% confidence that net will predict with $\leq \epsilon = 0.1$, we need this amount of training examples

$$m > \frac{W}{\epsilon} = \frac{441}{0.1} = 4410$$

Parity bit value



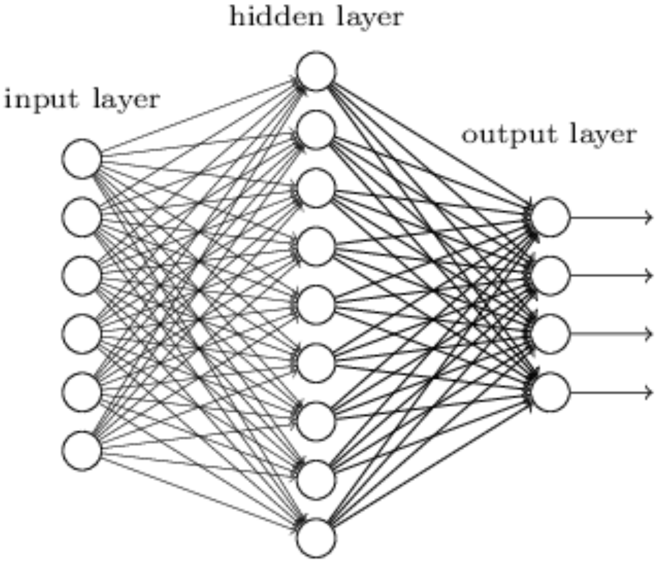
n bits of input

2^n possible examples

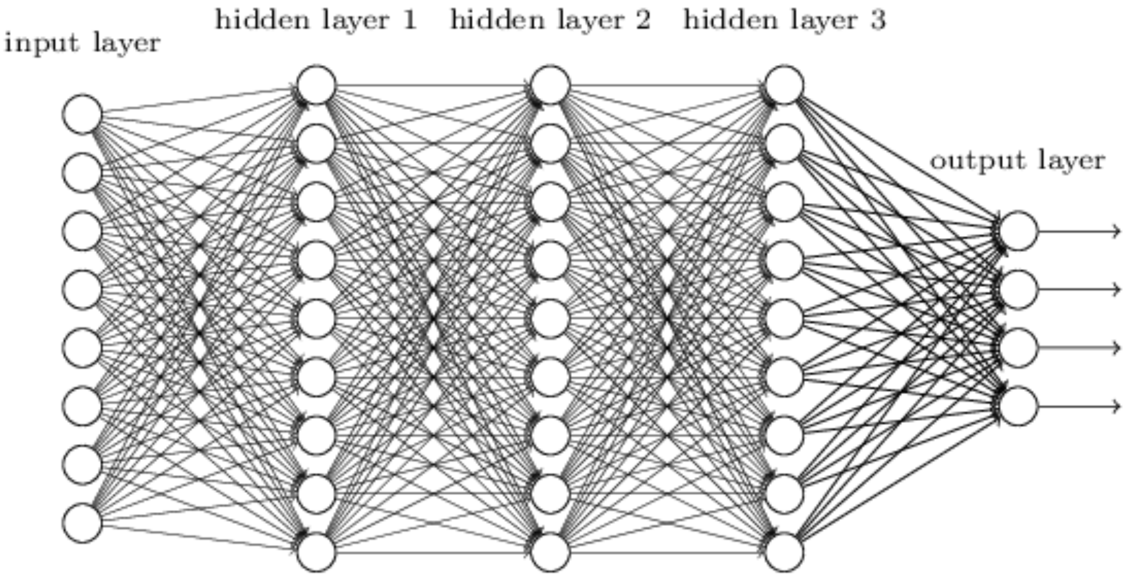
NETWORK TYPES

Deep Neural Networks

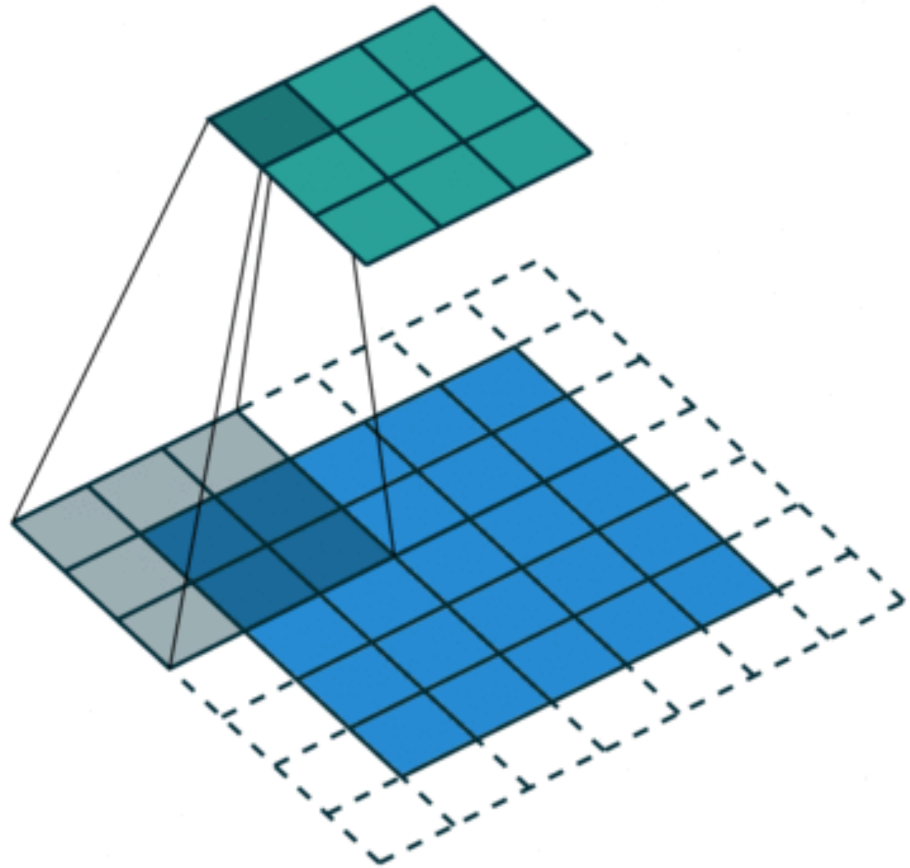
"Non-deep" feedforward neural network



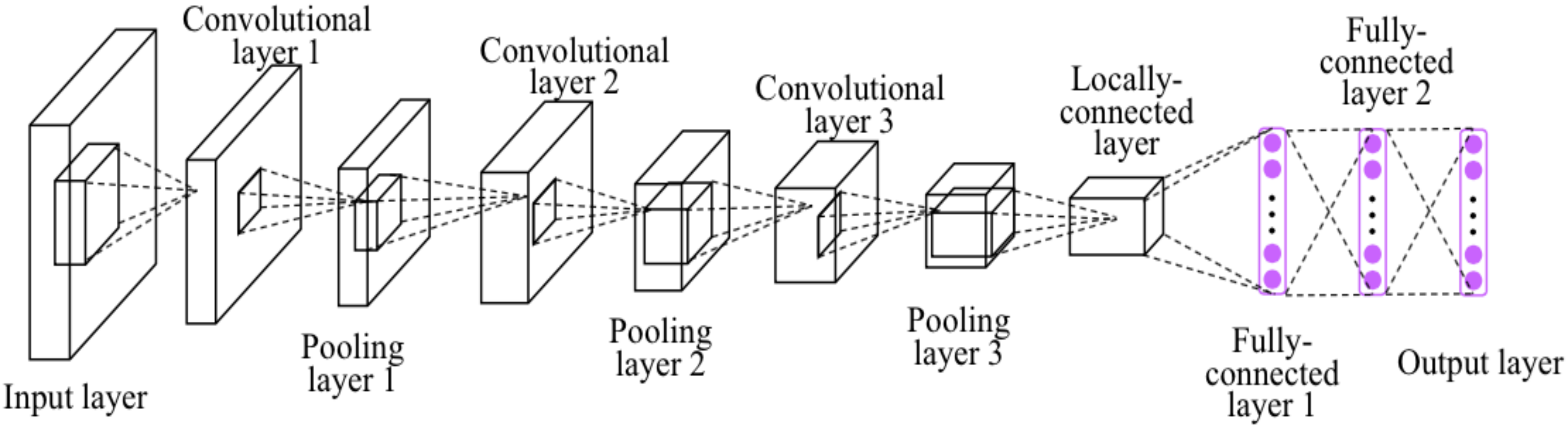
Deep neural network



Convolutional Neural Networks (CNN)

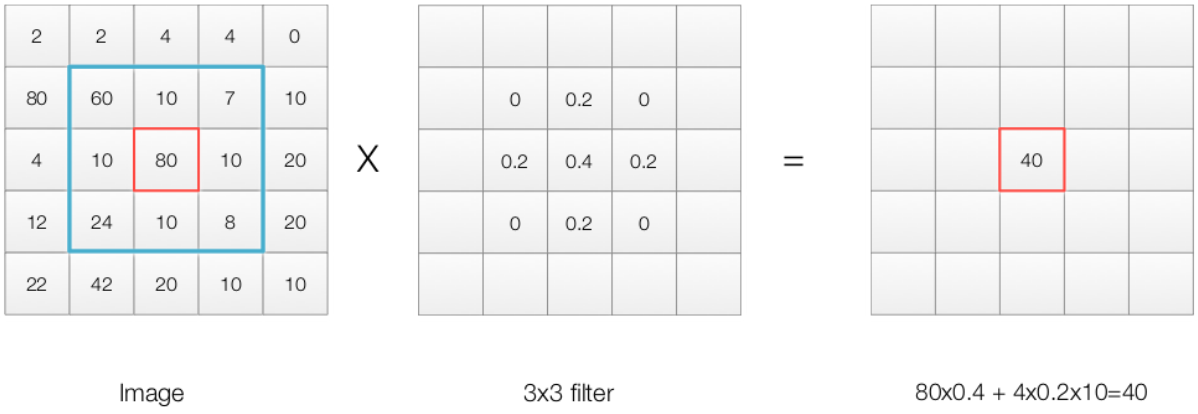


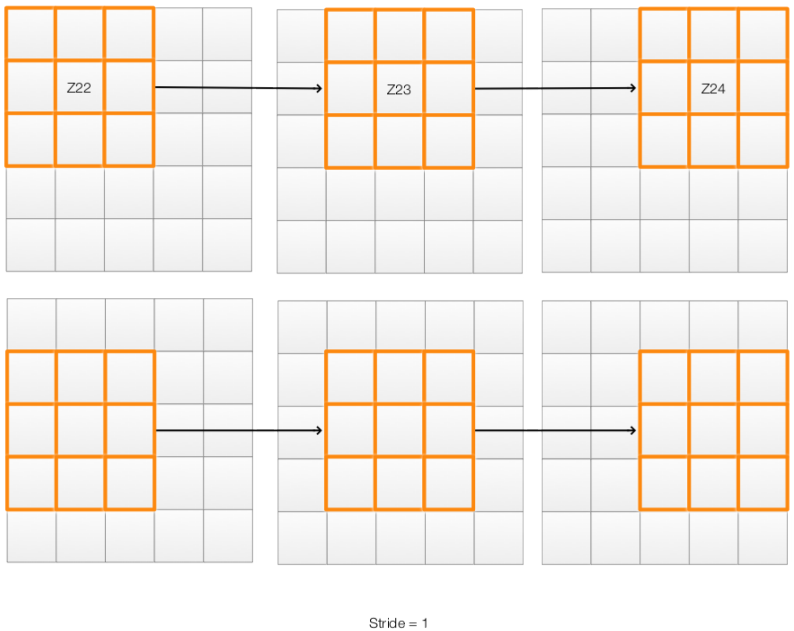
Convolutional Neural Networks (CNN)



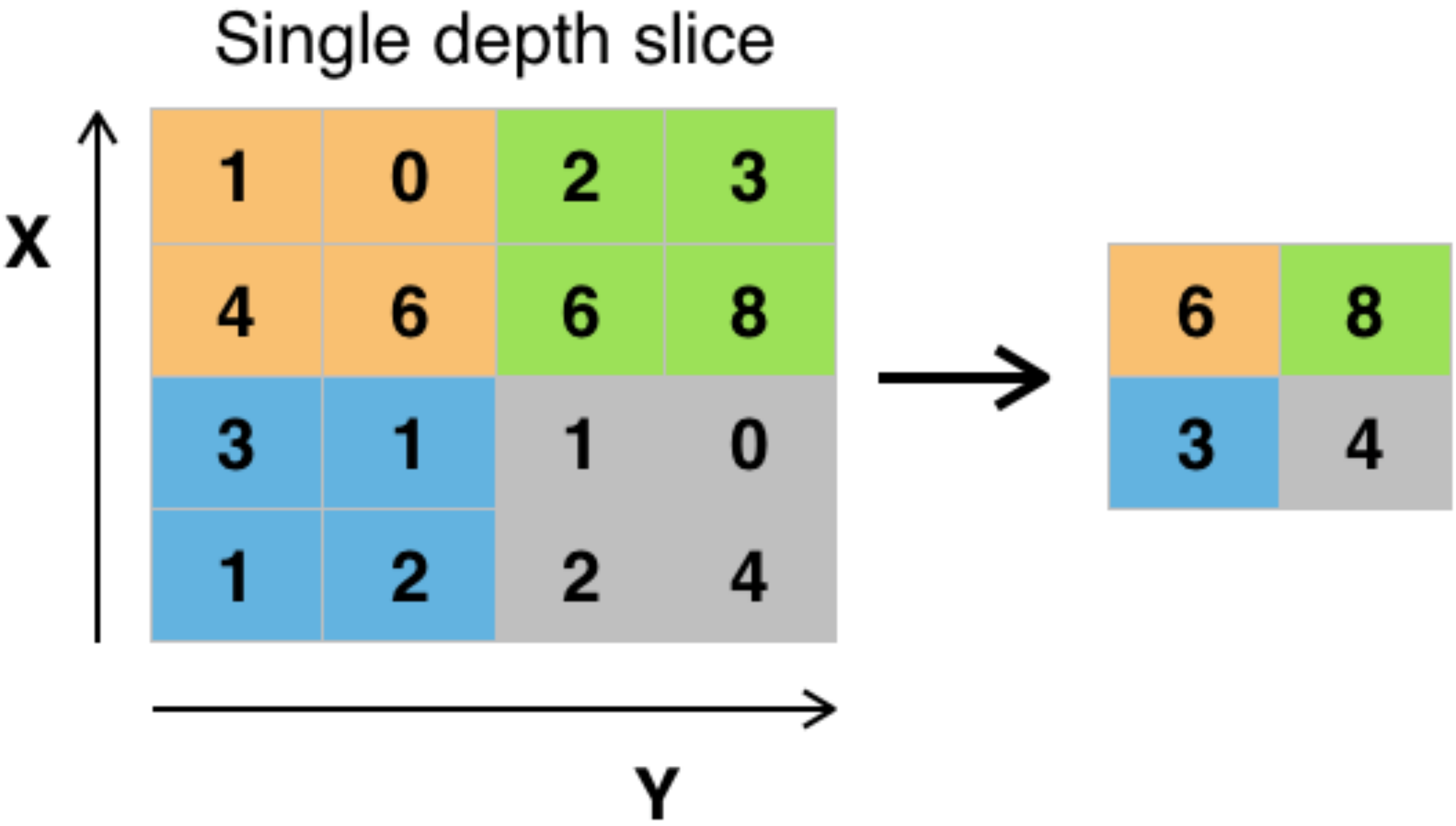
© http://personal.ie.cuhk.edu.hk/~ccloy/project_target_code/images/fig3.png [19.05.2020]

Convolutional Neural Networks (CNN)





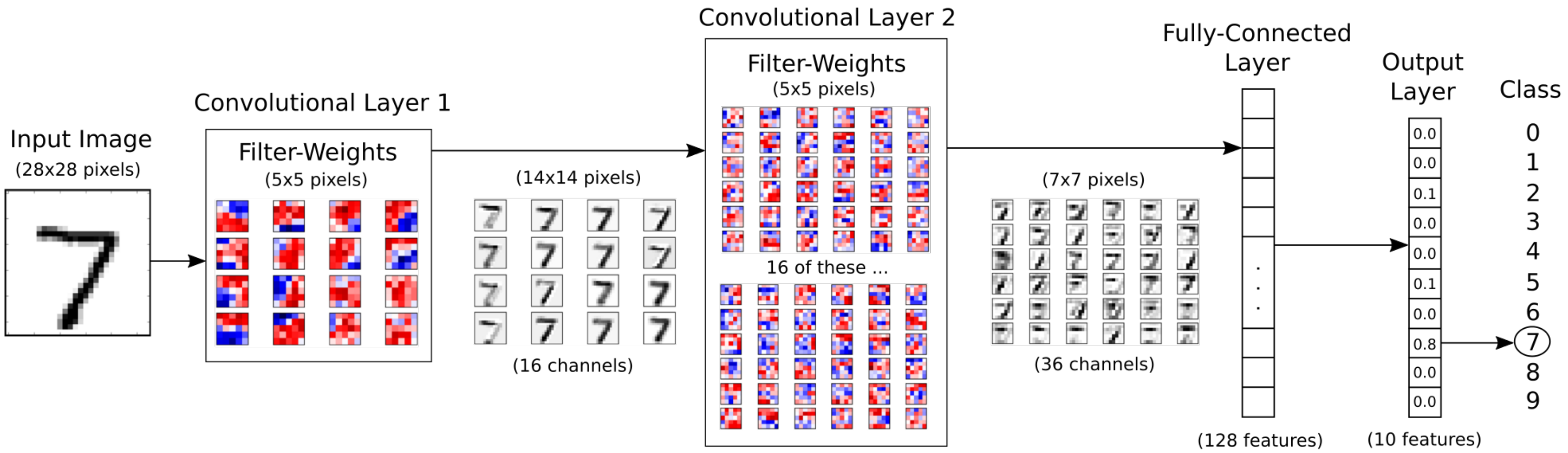
Convolutional Neural Networks (CNN)



Example of Maxpool with a 2x2 filter and a stride of 2

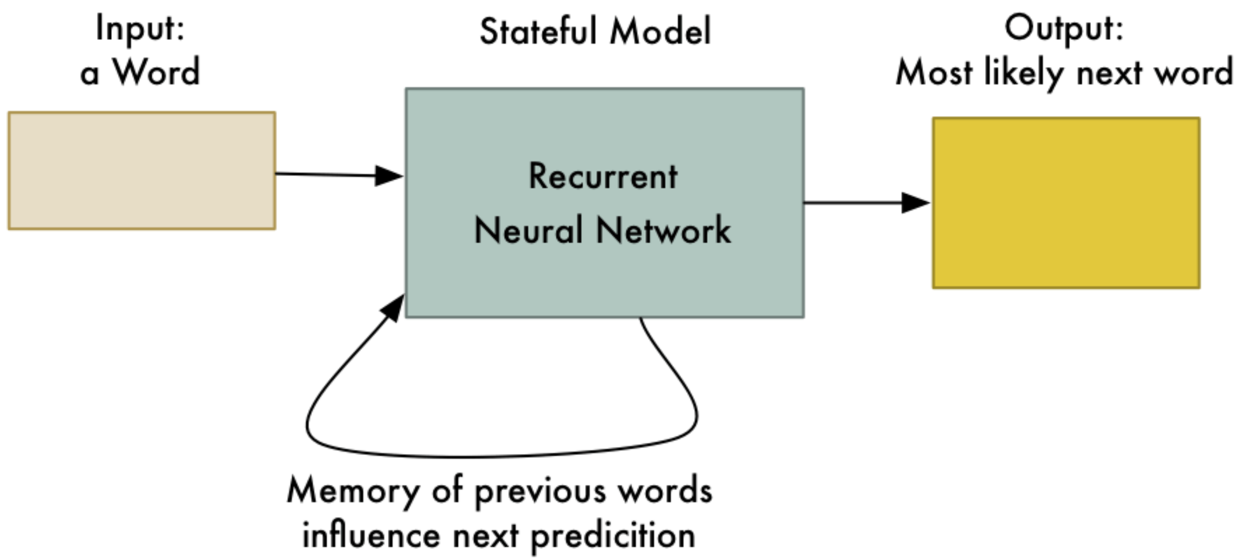
CNN: Learning MNIST

http://yann.lecun.com/exdb/mnist/



https://colab.research.google.com/github/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb#scrollTo=Q7kAPMNP9FZK [19.05.2020]

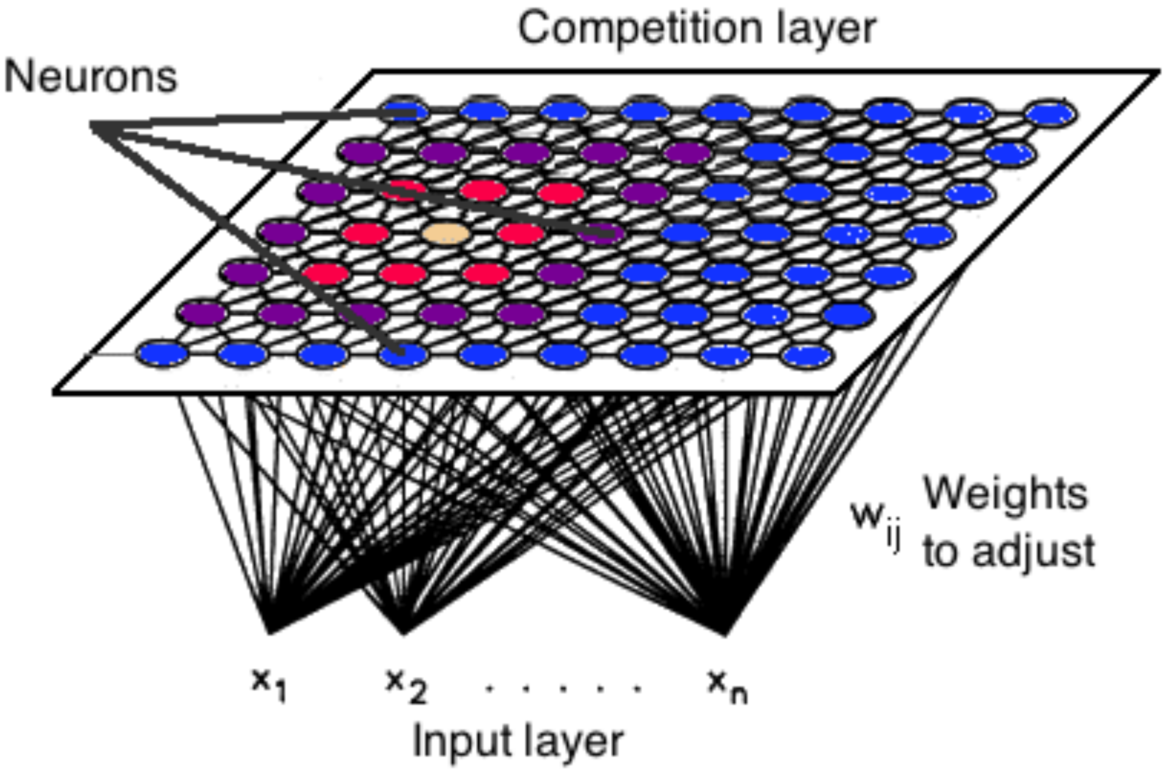
Recurrent Neuronal Network (RNN)



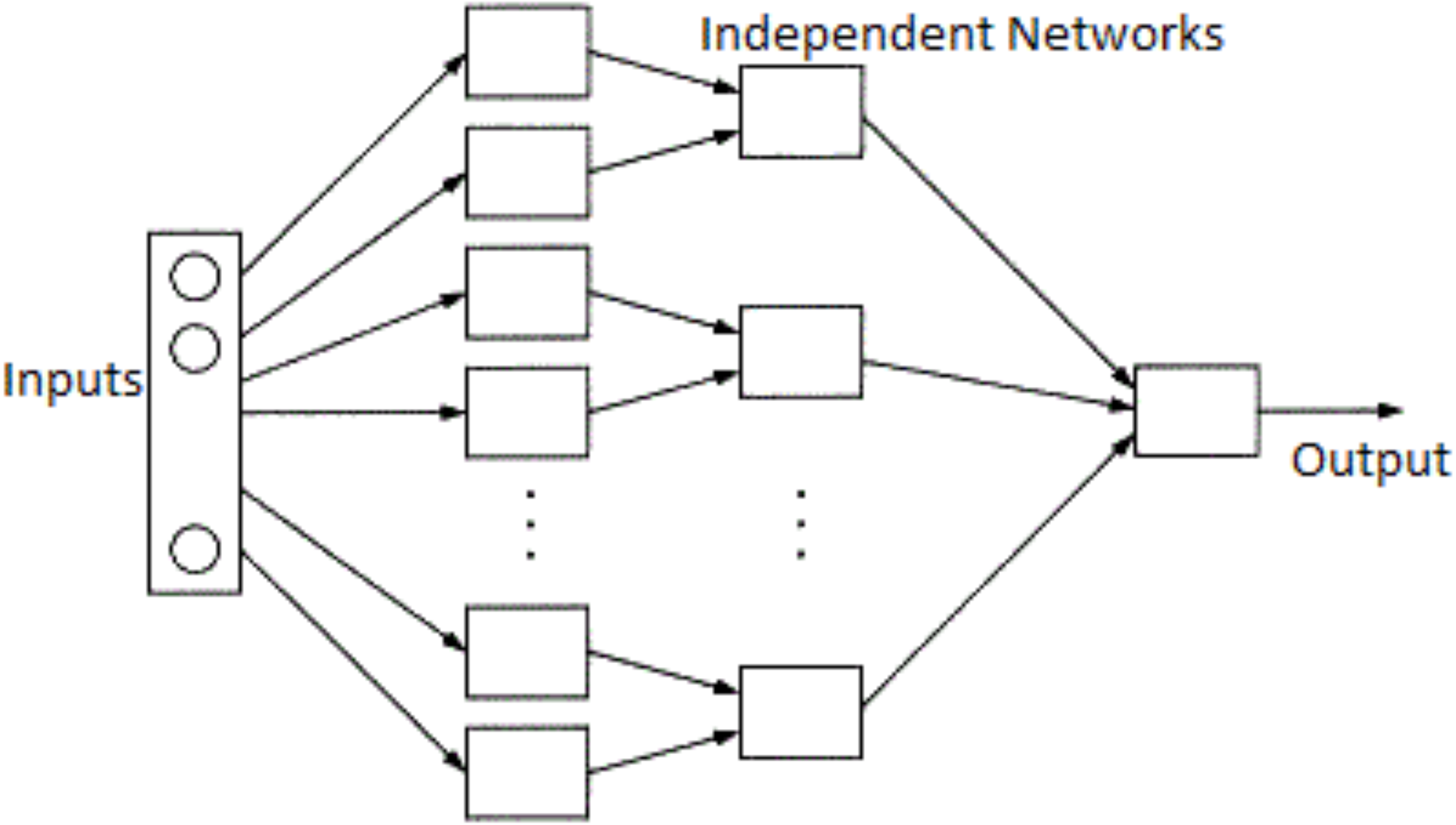
Output so far:

Machine

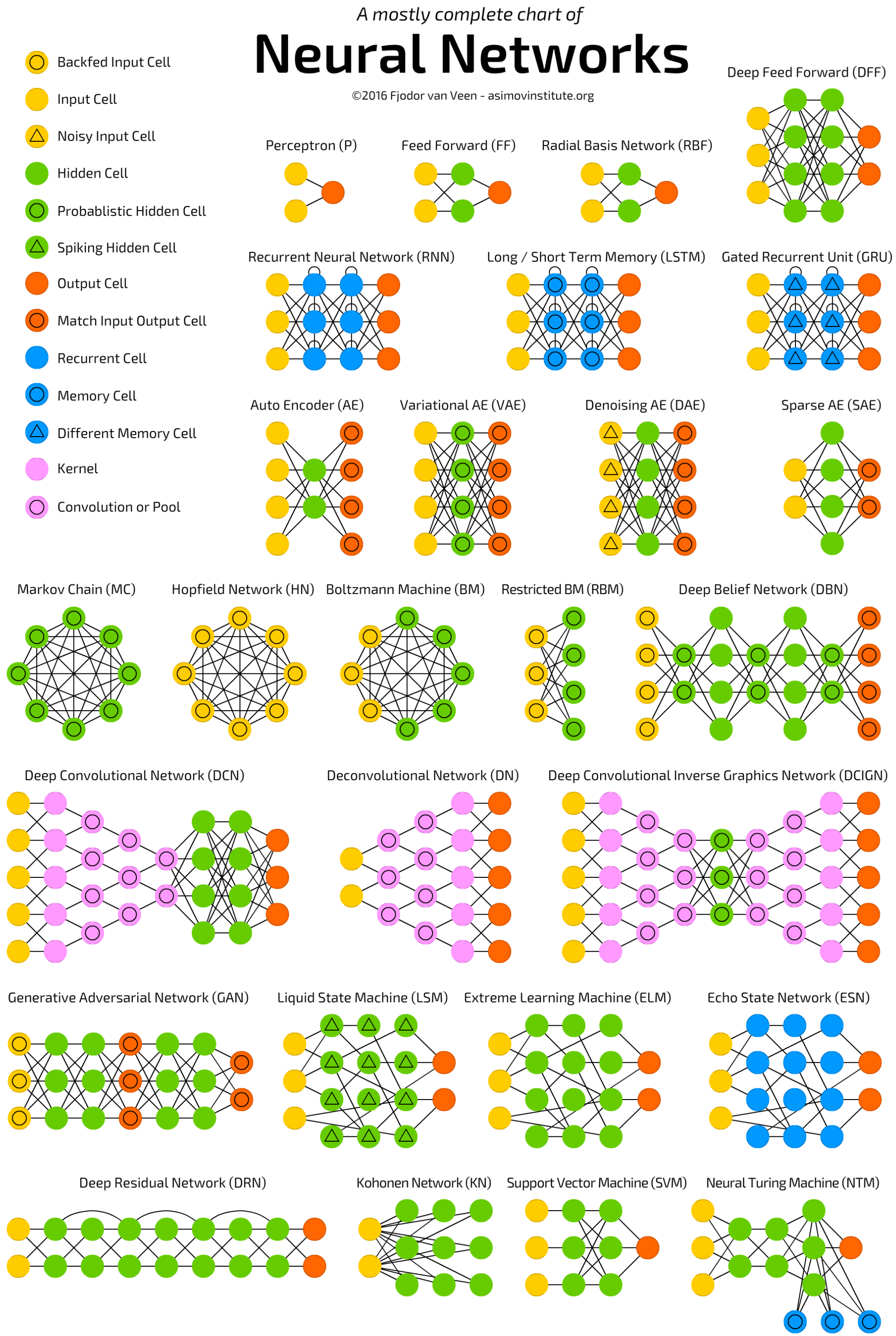
Kohonen Maps



Modular Neural Networks (The Hit!)



 © https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/ [04.06.2018]


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

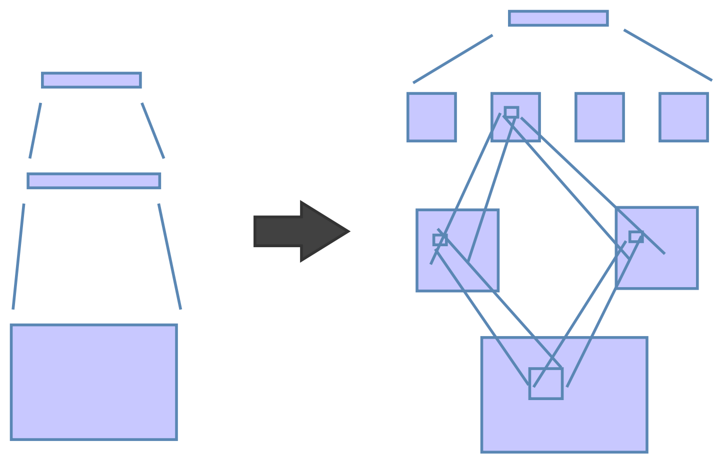
NETWORK DESIGN

Architecture of the network

- How many nodes?
- Determines number of network weights
- How many layers?
- How many nodes per layer?
  - Input Layer ____ Hidden Layer ____ Output Layer ____

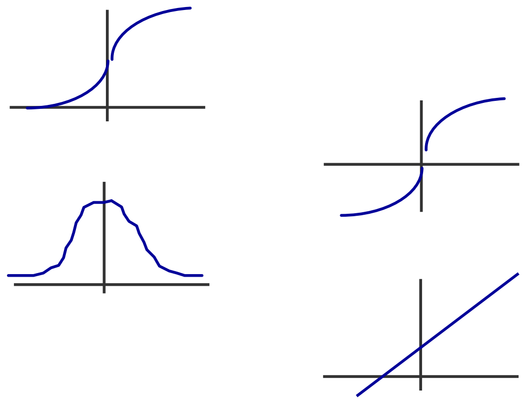
Architecture of the network: Connectivity

- Concept of model or hypothesis space
- Constraining the number of hypotheses:
  - selective connectivity
  - shared weights
  - recursive connections



Structure of artificial neuron nodes

- Choice of input integration:
  - summed, squared and summed
  - multiplied
- Choice of activation (transfer) function:
  - sigmoid (logistic)
  - hyperbolic tangent
  - Guassian
  - linear
  - soft-max



Selecting a Learning Rule (Optimizer)

- Generalized delta rule (steepest descent)
- Momentum descent
- Advanced weight space search techniques
- Global Error function can also vary
  - Normal
  - quadratic
  - cubic

NETWORK TRAINING

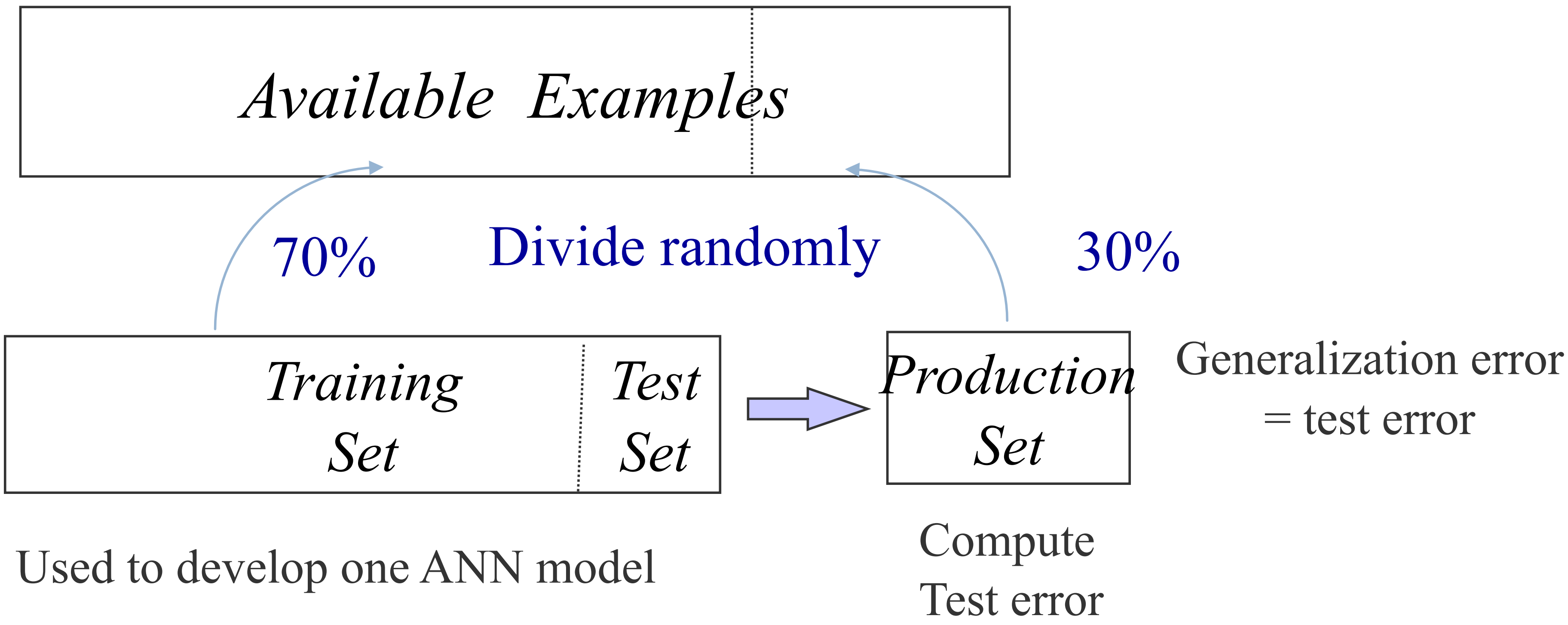
How do you ensure that a network has been well trained?

- Objective: To achieve good generalization accuracy on new examples/cases
- Establish a maximum acceptable error rate
- Train the network using a validation test set to tune it
- Validate the trained network against a separate test set which is usually referred to as a production test set

Network Training

Approach #1: Large Sample

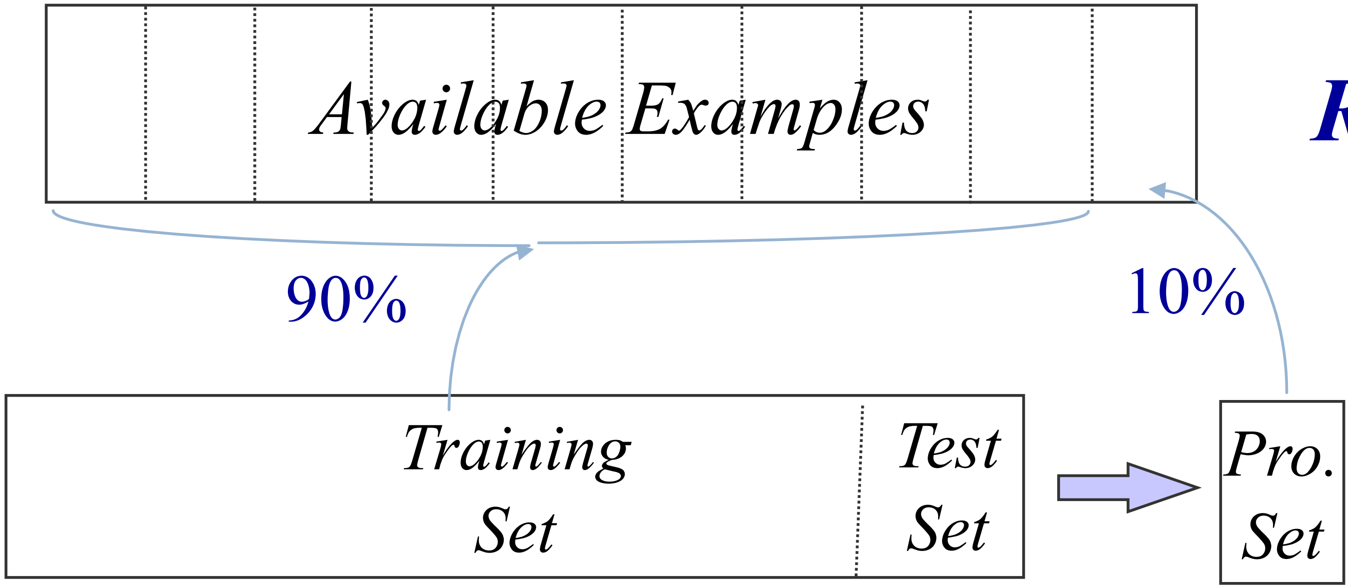
When the amount of available data is large ...



Network Training

Approach #2: Cross-validation

When the amount of available data is small ...



Repeat 10 times

Generalization error determined by mean test error and stddev

Used to develop 10 different ANN models

Accumulate test errors

Network Training: Mastering ANN Parameters

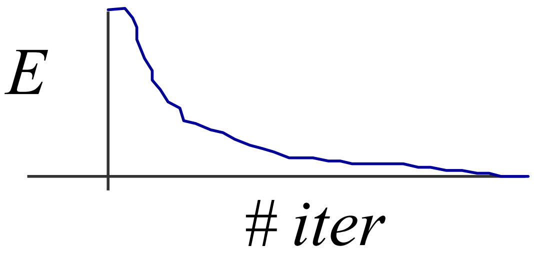
- Typical Range

♦ learning rate - η	0.1	0.01 - 0.99
♦ momentum - α	0.8	0.1 - 0.9
♦ weight-cost - λ	0.1	0.001 - 0.5

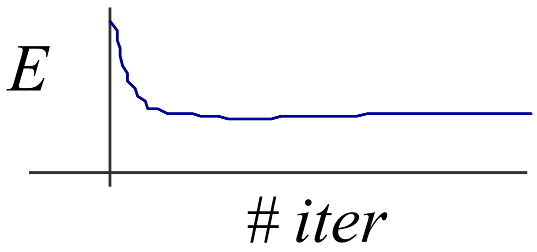
- Fine tuning : adjust individual parameters at each node and/or connection weight
  - automatic adjustment during training

Typical Problems During Training

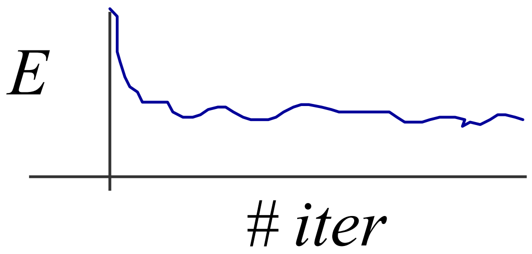
Would like:



Steady, rapid decline in total error

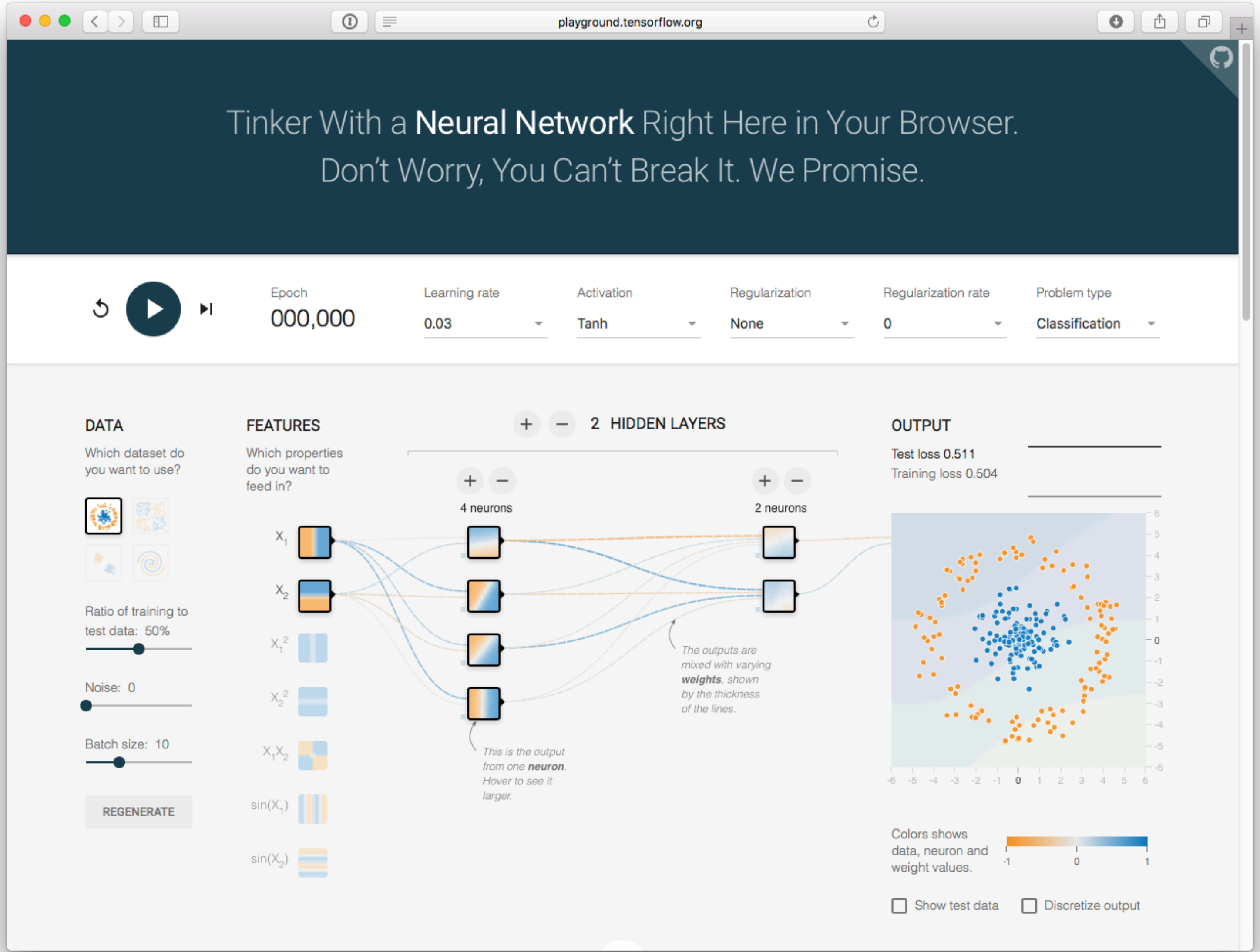


Seldom a local minimum
- reduce learning or momentum parameter



Reduce learning parms.
- may indicate data is not learnable

Playground



http://playground.tensorflow.org/ [09.06.2017]

Neural Networks vs. Decision Trees/Rules

- Classification with Neural Networks is very good
- Decisions with neural networks are not comprehensible
- Decision Trees and Rules are often more trustworthy
  - Decisions with trees and rules are comprehensible and explainable
  - One can see which rules are applied to make a decision
- In applications, in which trust in the decision or explainability is important, people prefer decision trees or rules