university of groningen

(C) Regression, layered neural networks

- Networks of **continuous units**
- **Regression** problems
- **Gradient descent,** *backpropagation of error*
- The role of the **learning rate**
- **Online learning,** *stochastic approximation*

university of groningen

Of Neurons and Networks

biological neurons (very brief)

- single neurons
- synapses and networks
- synaptic plasticity and learning

simplified description

- **inspiration for artificial neural networks**

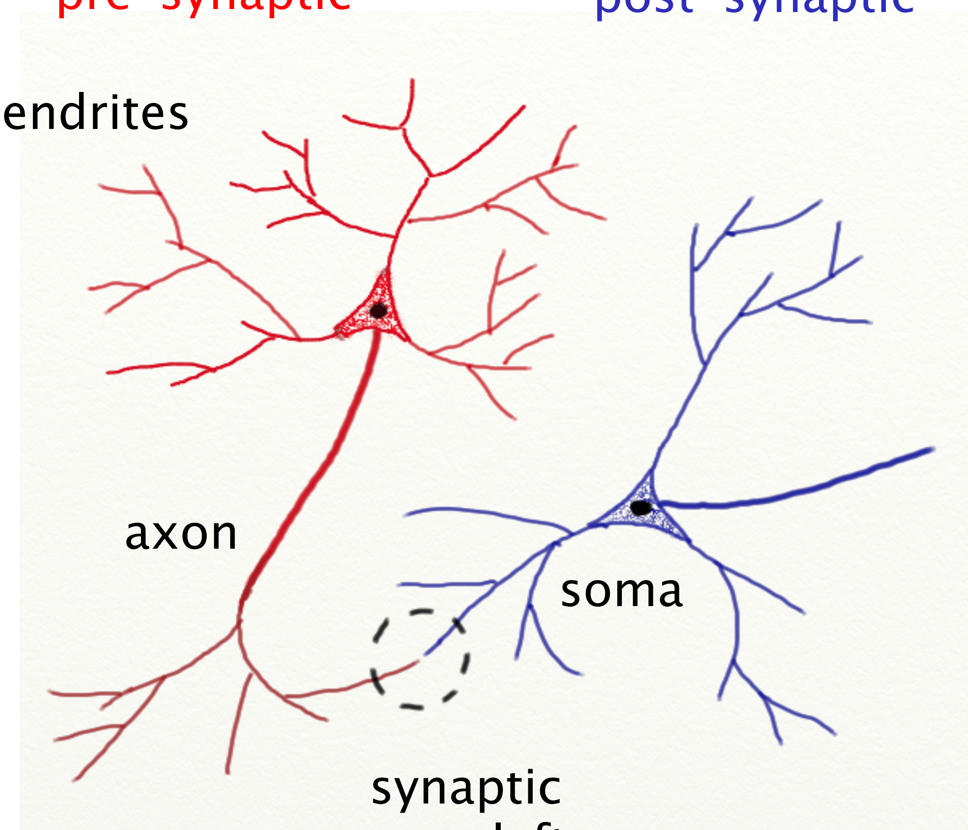
artificial neural networks

- architectures and types of networks:

  recurrent **attractor neural networks** (associative memory)

  feed-forward neural networks (classification/ regression)

Of Neurons and Networks



neurons:

highly specialized cells

- cell body **soma**
- incoming **dendrites**
- branched **axon**

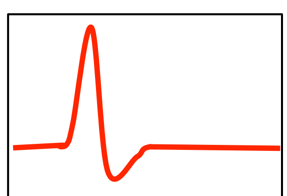
many neurons !

$\gtrsim 10^{12}$ in human cortex

highly connected !

$\gtrsim$ 1000 *neighbors*

action potentials / spikes:

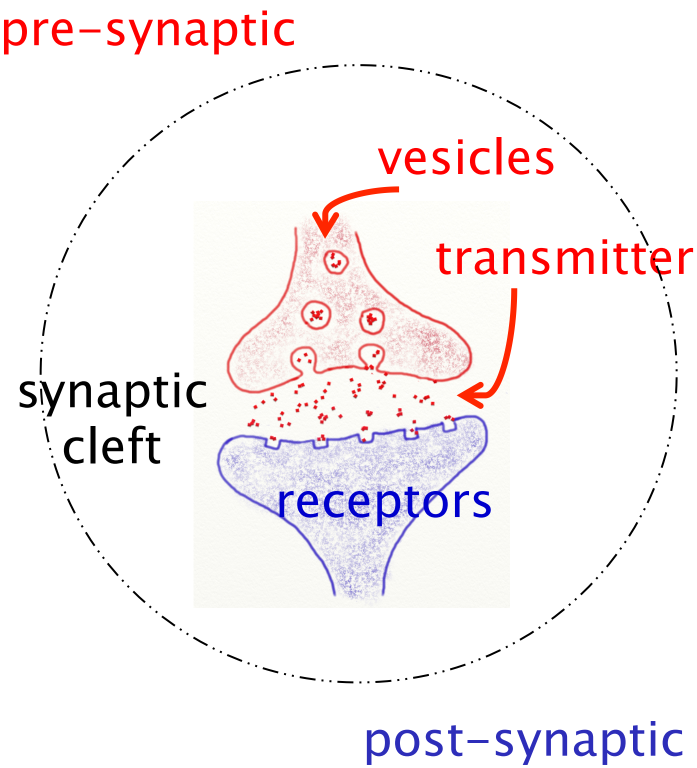


- cells generate electric pulses
- travel along the axon

synapses:

- pre-synaptic pulse arriving at
  excitatory / inhibitory synapse
  triggers / hinders
  post-synaptic spike generation

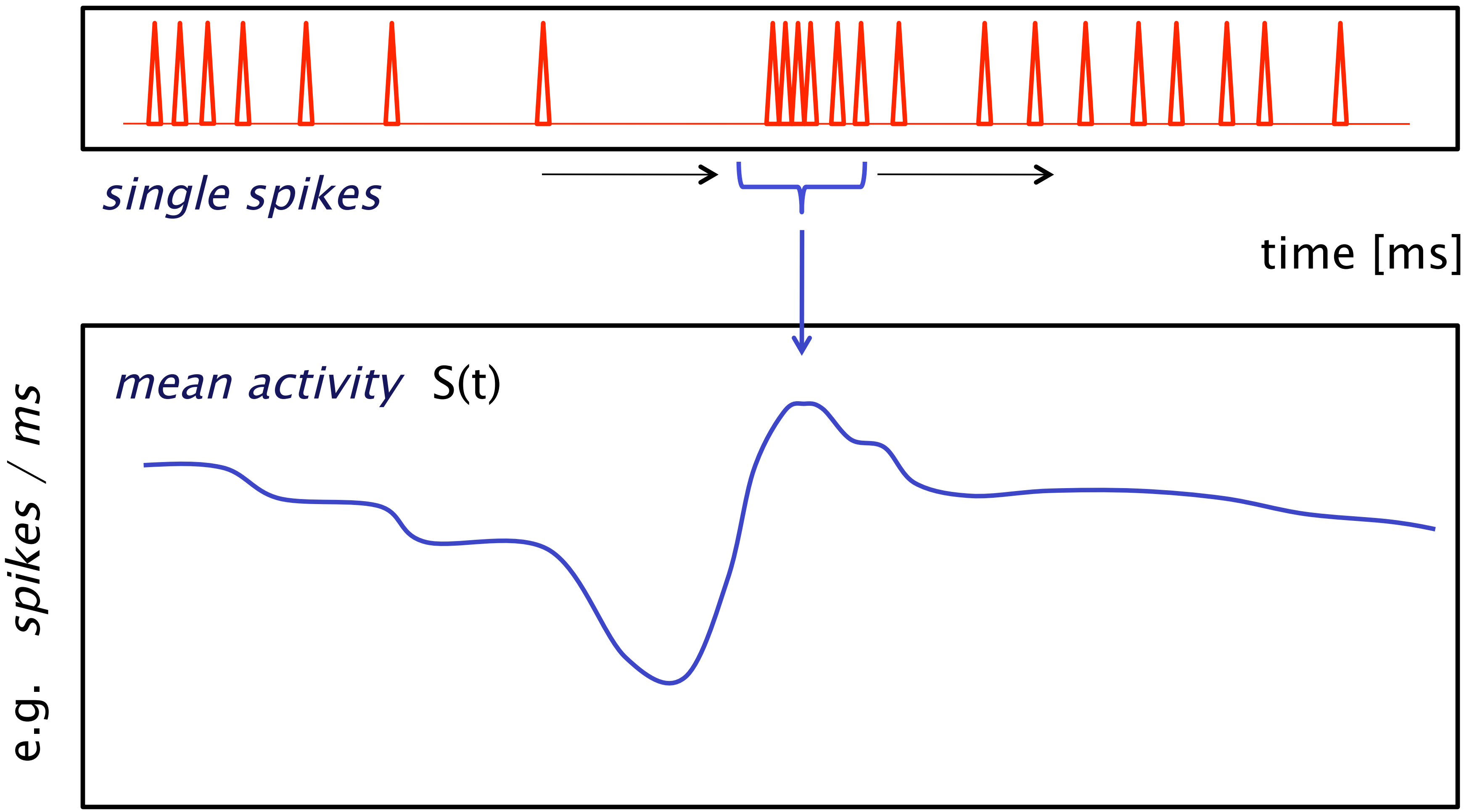
- incoming pulse
  - excitatory: increase
    the postsynaptic
    membrane potential
  - inhibitory: decrease



- all or nothing response

potential exceeds **threshold** ⇨ postsynaptic neuron *fires*

potential is sub-**threshold** ⇨ **postsynaptic neuron** *rests* simplified description of neural activity: **firing rates**

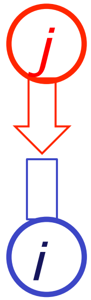


(mean) **local potential** at neuron *i* (with activity S_i)

$$\sum_j w_{ij} S_j$$ **weighted sum** of incoming activities

synaptic weights

$$w_{ij} = \begin{cases} > 0 & \text{excitatory synapse} \\ = 0 & \\ < 0 & \text{inhibitory synapse} \end{cases}$$



university of groningen

Activation Function

non–linear response: $S_i = h\left[\sum_j w_{ij} S_j\right]$

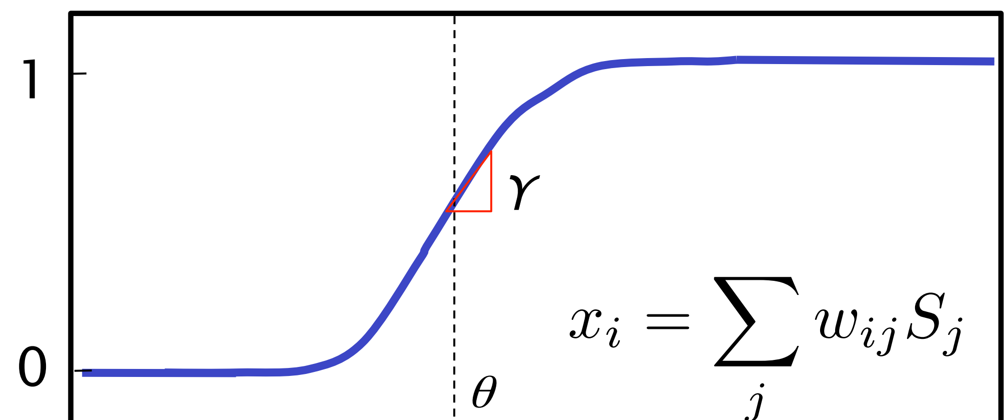
- minimal activity $h(x \to -\infty) \equiv 0$
- maximal activity $h(x \to +\infty) \equiv 1$
- monotonic increase $h'(x) > 0$

important class of fcts.: ***sigmoidal activation***

just one example: $$h(x_i) = \frac{1}{2}\Big(1 + \tanh\left[\gamma(x_i - \theta)\right]\Big)$$

gain parameter γ

local threshold θ



Activation Function

non-linear response: $S_i = g\left[\sum_j w_{ij} S_j\right]$

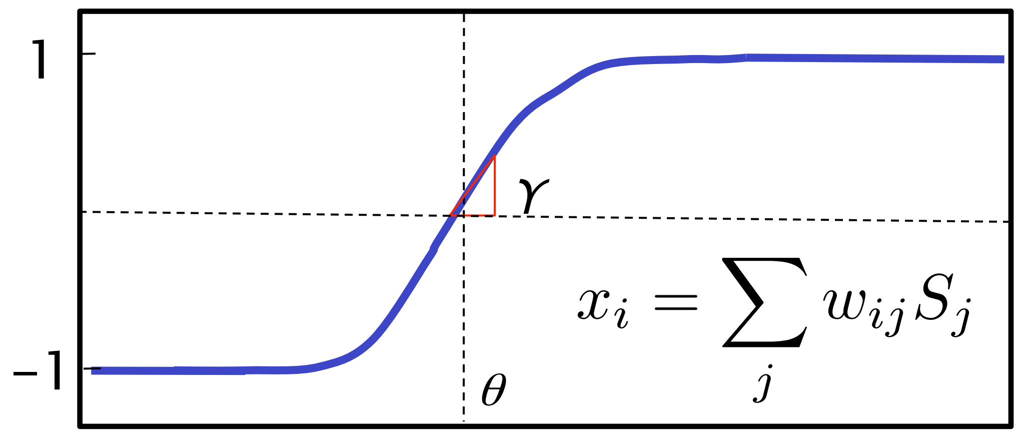
- minimal activity $g(x \to -\infty) \equiv -1$
- maximal activity $g(x \to +\infty) \equiv 1$
- monotonic increase $g'(x) > 0$

sigmoidal activation

just one example: $g(x_i) = \tanh\left[\gamma(x_i - \theta)\right]$

gain parameter γ

local threshold θ



McCulloch Pitts Neurons

an extreme case: **infinite gain** $\gamma \to \infty$

$$g(x_i) = \tanh\left[\gamma(x_i - \theta)\right] \to \operatorname{sign}\left[x - \theta\right] = \begin{cases} +1 & \text{for } x \geq \theta \\ -1 & \text{for } x < \theta \end{cases}$$

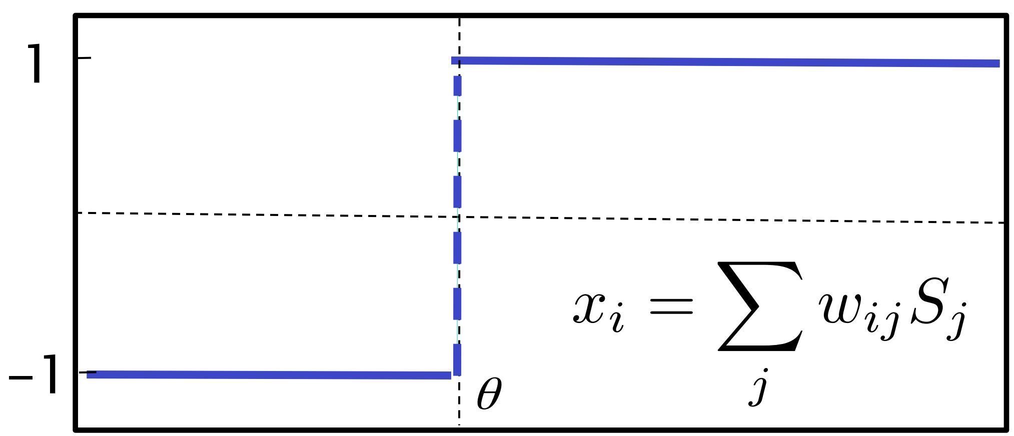
McCulloch Pitts [1943]:

model neuron is either quiescent or maximally active

do not consider graded response

local threshold θ

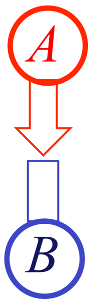
(don't confuse θ with the all-or-nothing threshold in spiking neurons)



Synaptic Plasticity

D. Hebb [1949] Hypothesis: **Hebbian Learning**

consider
- presynaptic neuron A
- postsynaptic neuron B
- excitatory synapse w_{BA}



If A and B (frequently) fire at the same time
the excitatory synaptic strength w_{AB} increases

→ memory-effect will favor joint activity in the future

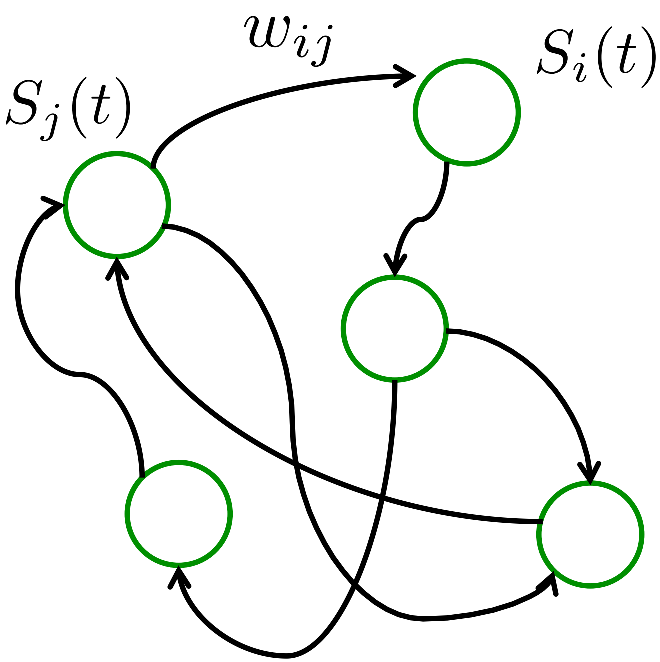
For symmetrized firing rates $-1 \leq S_A, S_B \leq +1$

change of synaptic strength $\Delta w_{BA} \propto S_A S_B$

pre-synaptic x post-synaptic activity

in the following:

- assembled from simple *firing rate neurons*
- connected by *weights*, real valued synaptic strenghts
- various architectures and types of networks
  e.g.: **attractor neural networks, recurrent networks**



dynamical systems, e.g. *Hopfield model:*

network of McCulloch Pitts neurons,

can operate as *Associative Memory*

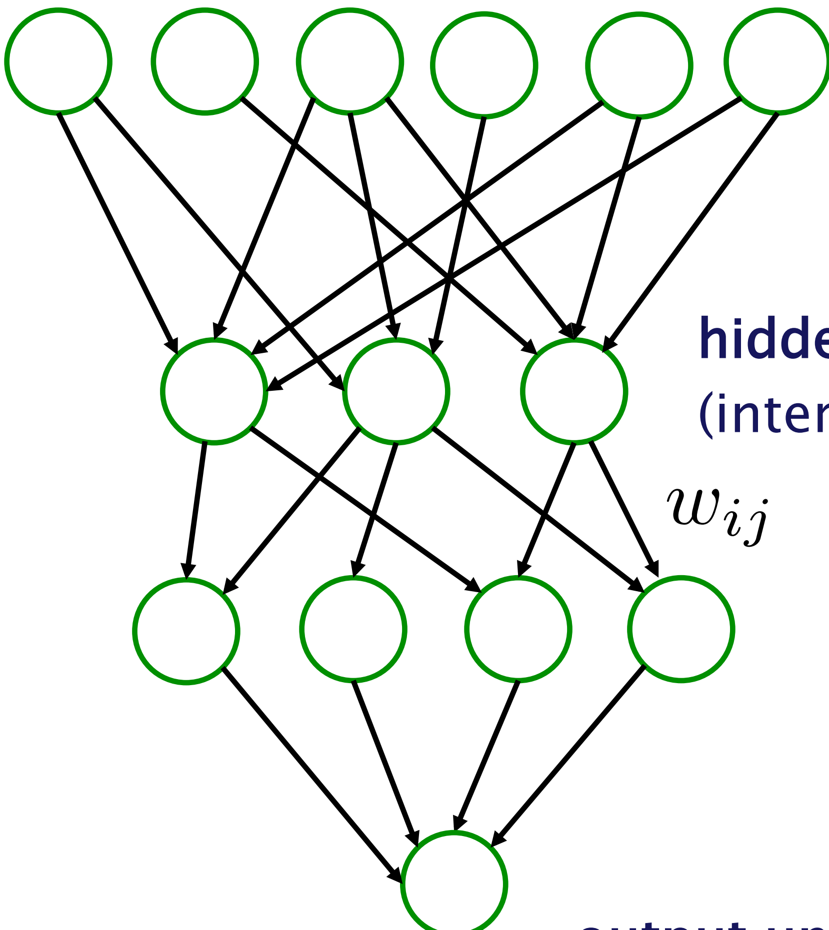
by learning of synaptic interactions

here: N=5 neurons

partial connectivity

feed–forward networks

input layer (external stimulus)



layered architecture
(here: 6–3–4–1)

directed connections
(here: only to next layer)

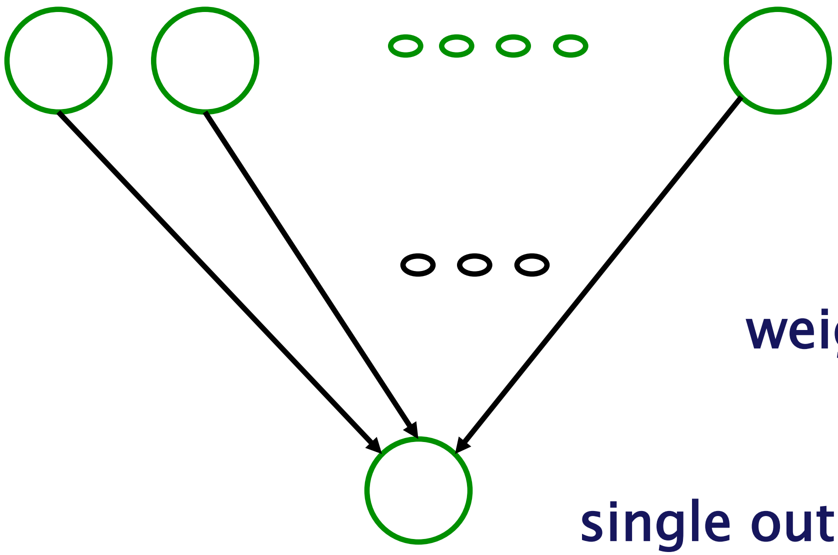
hidden units
(internal representation)

$$S_i = g\left(\sum_j w_{ij} S_j\right)$$

↑ previous layer only

output unit(s)
(function of input vector)

the perceptron revisited



input units $\xi_j \in I\!R$

weights $w_j \in I\!R, \mathbf{w} \in I\!R^N$

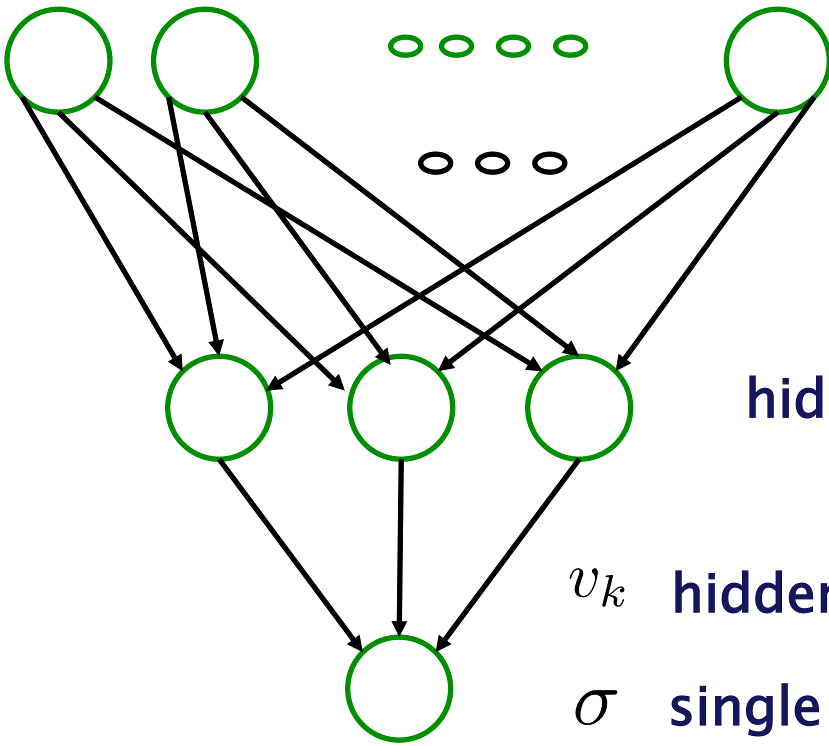
single output unit

$$S = \text{sign}\left(\sum_{j=1}^{N} w_j \xi_j - \theta\right)$$

output $=$ ***"linear separable function"*** of input variables

parameterized by the weight vector $\mathbf{w}$ and threshold θ

convergent two-layer architecture



input units $\xi_j \in I\!R,\ \boldsymbol{\xi} \in I\!R^N$

$w_j^{(k)}$ input to hidden weights

hidden layer units $S_k = g\left(\sum_j w_j^{(k)} \xi_j\right)$

v_k hidden to output weights

σ single output unit

output = **non-linear function** of input variables:

$$\sigma = g\left(\sum_{k=1}^{K} v_k S_k\right) = g\left(\sum_k v_k\, g\left(\sum_j w_j^{(k)} \xi_j\right)\right)$$

parameterized by set of all weights (and threshold)

networks of continuous nodes

continuous activation functions, e.g. $g(x) = \tanh(\gamma x)$
for all nodes in the network

given a network architecture, the weights (and thresholds) parameterize a function (input/output relation):

$$\boldsymbol{\xi} \in I\!R^N \rightarrow \sigma(\boldsymbol{\xi}) \in I\!R$$ (here: single output unit)

Learning as **regression problem**

set of examples $\{\boldsymbol{\xi}^\mu, \tau^\mu = \tau(\boldsymbol{\xi}^\mu)\}_{\mu=1}^P$ with real-valued labels τ^μ

training:
(approximately) implement $\sigma(\boldsymbol{\xi}^\mu) = \tau(\boldsymbol{\xi}^\mu)$ for all μ

generalization:
application to novel data $\sigma(\boldsymbol{\xi}) \approx \tau(\boldsymbol{\xi})$

error measure and training

training strategy: employ an **error measure**
for comparison of student/teacher outputs

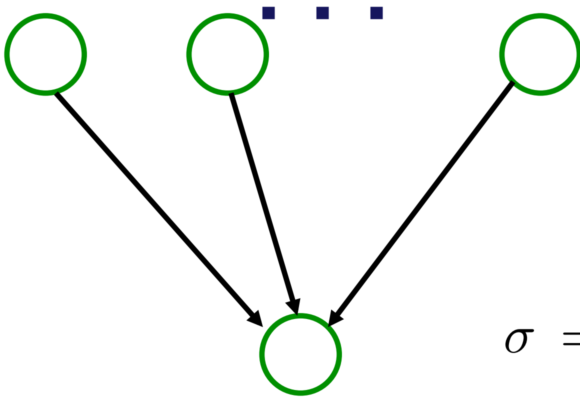
just <u>one</u> very popular and plausible choice:

quadratic deviation: $$e(\sigma,\tau) = \frac{1}{2}(\sigma-\tau)^2$$

cost function: $$E = \frac{1}{P}\sum_{\mu=1}^{P} e^{\mu} = \frac{1}{P}\sum_{\mu=1}^{P}\frac{1}{2}\Big(\sigma(\boldsymbol{\xi}^{\mu}) - \tau(\boldsymbol{\xi}^{\mu})\Big)^2$$

- defined for a given set of example data
- guides the training process
- is a **differentiable function** of weights and thresholds
- training by **gradient descent** minimization of *E*

a single unit



$$\xi_j \in \mathbb{R},\ \boldsymbol{\xi} \in \mathbb{R}^N$$

$$\mathbf{w} \in \mathbb{R}^N$$

$$\sigma = g\left(\sum_{j=1}^{N} w_j\, \xi_j\right)$$

$$E(\mathbf{w}) = \frac{1}{P}\sum_{\mu=1}^{P} \frac{1}{2}\Big(g(\mathbf{w}\cdot\boldsymbol{\xi}^\mu) - \tau^\mu\Big)^2$$

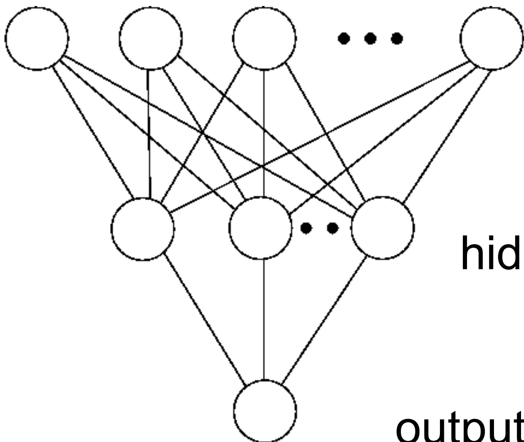
$$\frac{\partial E(\mathbf{w})}{\partial w_k} = \frac{1}{P}\sum_{\mu=1}^{P}\Big(g(\mathbf{w}\cdot\boldsymbol{\xi}^\mu) - \tau^\mu\Big)\, g'(\mathbf{w}\cdot\boldsymbol{\xi}^\mu)\ \xi_k^\mu$$

$$\nabla_w E(\mathbf{w}) = \frac{1}{P}\sum_{\mu=1}^{P}\Big(g(\mathbf{w}\cdot\boldsymbol{\xi}^\mu) - \tau^\mu\Big)\, g'(\mathbf{w}\cdot\boldsymbol{\xi}^\mu)\ \boldsymbol{\xi}^\mu$$

Backpropagation of Error

convenient calculation of the gradient in multilayer networks ($\leftarrow$ chain rule)

example: continuous two-layer network with K hidden units



inputs $\boldsymbol{\xi} \in I\!R^N$

weights $\mathbf{w}_k \in I\!R^N$, $k = 1, 2, \ldots, K$

hidden units $\sigma_k(\boldsymbol{\xi}) \; = \; g(\mathbf{w}_k \cdot \boldsymbol{\xi})$

output $\sigma(\boldsymbol{\xi}) \; = \; h\left(\sum_{j=1}^{K} v_j \, g(\mathbf{w}_j \cdot \boldsymbol{\xi})\right)$

the weigths $\mathbf{w}_k$ and v_k are used ...

– *downward* for the calculation of hidden states and output

– *upward* for the calculation of the gradient

Exercise: derive $\nabla_{\mathbf{w}_k} E$ and $\frac{\partial E}{\partial v_k}$

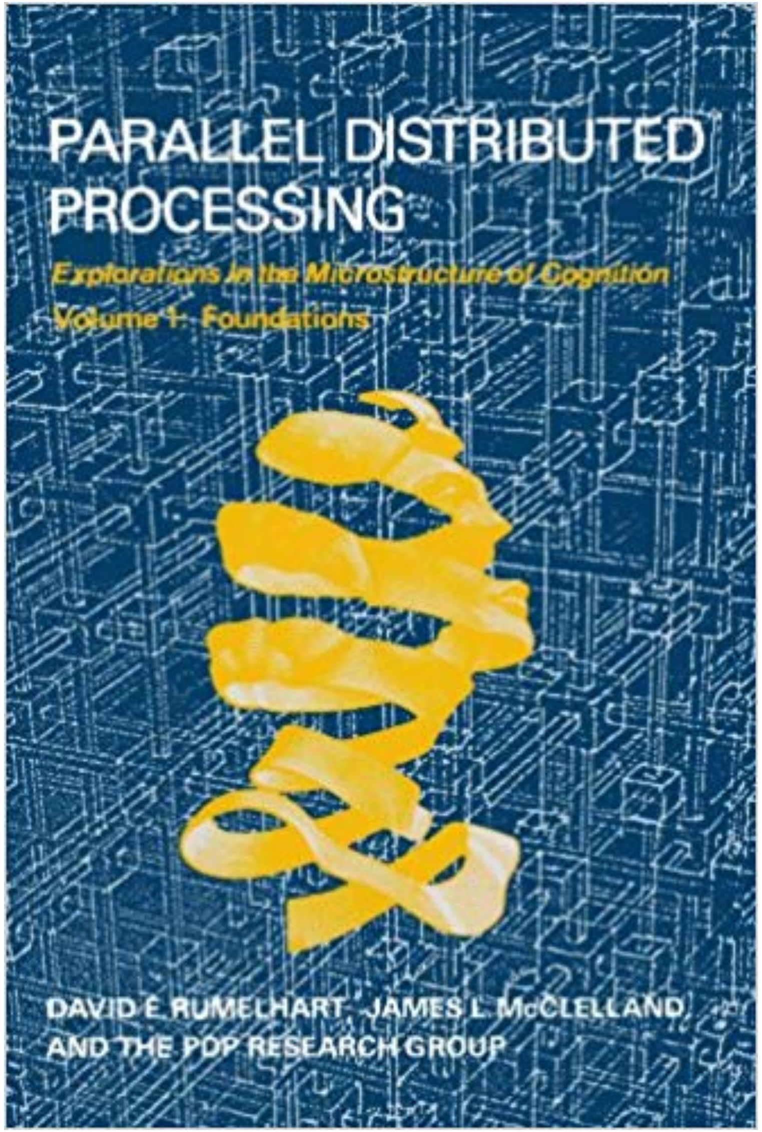
backpropagation

A.E. Bryson, Y.-C. Ho (1969)
Applied optimal control: optimization, estimation and control.
Blaisdell Publishing, p 481

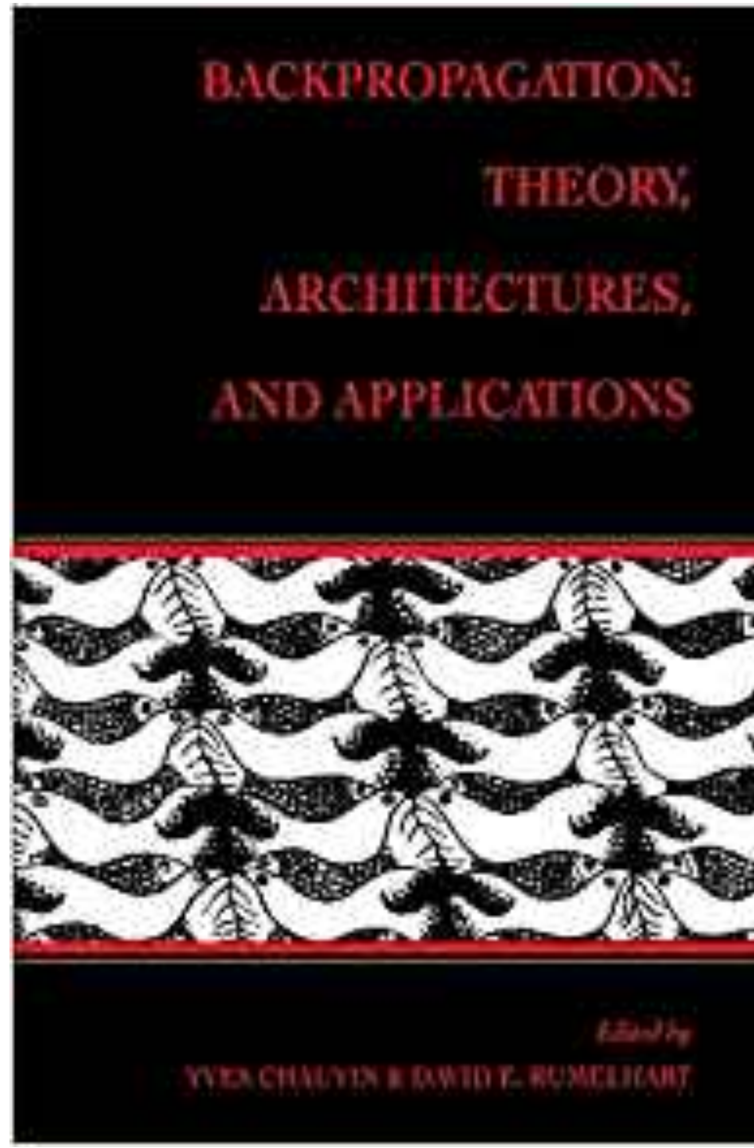
P. Werbos (1974). Beyond regression: New Tools for Prediction and Analysis in Behavorial Sciences
PhD thesis, Harvard University

D.E. Rumelhart, G.E. Hinton, R.J. Williams (1986)
Learning representations by backpropagating errors.
Nature 323 (6088): 533-536

backpropagation



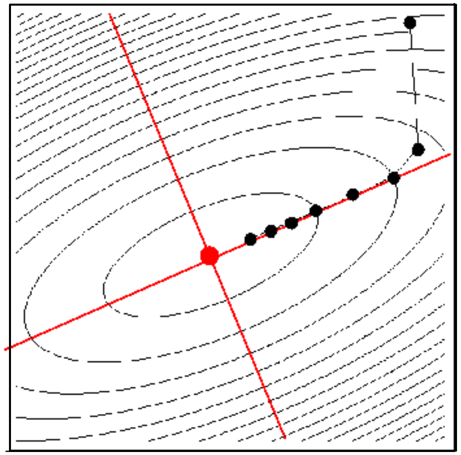
1987



1995

negative gradient gives the **direction of steepest descent** in E

simple gradient based minimization of E:



sequence $\quad \mathbf{w}_0 \to \mathbf{w}_1 \to \ldots \to \mathbf{w}_t \to \mathbf{w}_{t+1} \to \ldots$

with $\quad \mathbf{w}_{t+1} = \mathbf{w}_t - \eta \, \nabla E|_{\mathbf{w}_t}$

approaches <u>some</u> minimum of E (?)

learning rate rate η

– controls the step size of the algorithm

– has to be small enough to ensure convergence

– should be as large as possible to facilitate fast learning

assume E has a (local) minimum in $\mathbf{w}^*$, Taylor expansion in the vicinity:

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T \underbrace{\nabla E|_*}_{=0} + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) + \ldots$$

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) \qquad \nabla E|_{\mathbf{w}} \approx H^* (\mathbf{w} - \mathbf{w}^*)$$

with the positive definite **Hesse matrix** of second derivatives $\quad H^*_{ij} = \left.\frac{\partial^2 E}{\partial w_i \, \partial w_j}\right|_*$

H^* has only pos. eigenvalues $\lambda_i > 0$, orthonormal eigenvectors $\mathbf{u}_i$ (all $\lambda_i \leq \lambda_{max}$)

---

gradient descent in the vicinity of $\mathbf{w}^*$: $\quad \mathbf{w}_t = \mathbf{w}_{t-1} - \eta \, \nabla E|_{\mathbf{w}_{t-1}}$

assume E has a (local) minimum in $\mathbf{w}^*$, Taylor expansion in the vicinity:

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T \underbrace{\nabla E|_*}_{=0} + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) + \ldots$$

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) \qquad \nabla E|_{\mathbf{w}} \approx H^* (\mathbf{w} - \mathbf{w}^*)$$

with the positive definite **Hesse matrix** of second derivatives $\quad H^*_{ij} = \left. \frac{\partial^2 E}{\partial w_i \, \partial w_j} \right|_*$

H^* has only pos. eigenvalues $\lambda_i > 0$, orthonormal eigenvectors $\mathbf{u}_i$ (all $\lambda_i \leq \lambda_{max}$)

---

gradient descent in the vicinity of $\mathbf{w}^*$:

$$\mathbf{w}_t - \mathbf{w}^* \equiv \boldsymbol{\delta}_t = \boldsymbol{\delta}_{t-1} - \eta \, \nabla E|_{\mathbf{w}_{t-1}}$$
$$= [I - \eta H^*] \, \boldsymbol{\delta}_{t-1}$$

assume E has a (local) minimum in $\mathbf{w}^*$, Taylor expansion in the vicinity:

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T \underbrace{\nabla E|_*}_{=0} + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) + \ldots$$

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + \frac{1}{2} (\mathbf{w} - \mathbf{w}^*)^T H^* (\mathbf{w} - \mathbf{w}^*) \qquad \nabla E|_{\mathbf{w}} \approx H^* (\mathbf{w} - \mathbf{w}^*)$$

with the positive definite **Hesse matrix** of second derivatives $H^*_{ij} = \left. \frac{\partial^2 E}{\partial w_i \, \partial w_j} \right|_*$

H^* has only pos. eigenvalues $\lambda_i > 0$, orthonormal eigenvectors $\mathbf{u}_i$ (all $\lambda_i \leq \lambda_{max}$)

---

gradient descent in the vicinity of $\mathbf{w}^*$: $\mathbf{w}_t - \mathbf{w}^* \equiv \boldsymbol{\delta}_t = \boldsymbol{\delta}_{t-1} - \eta \nabla E|_{\mathbf{w}_{t-1}}$

$$\boldsymbol{\delta}_t \approx [I - \eta H^*] \boldsymbol{\delta}_{t-1} \approx [I - \eta H^*]^t \boldsymbol{\delta}_0$$

expansion in $\{\mathbf{u}_i\}$: $\delta_0 = \sum_i a_i \mathbf{u}_i$

$$\boldsymbol{\delta}_t \approx \sum_i a_i [I - \eta H^*]^t \mathbf{u}_i = \sum_i a_i [1 - \eta \lambda_i]^t \mathbf{u}_i$$

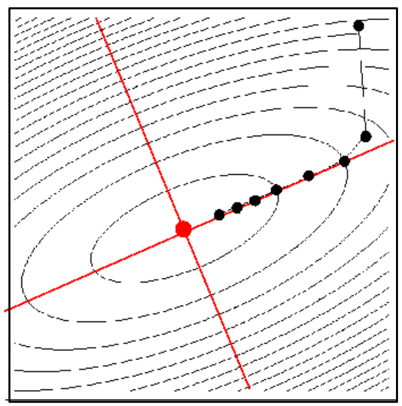
with $\mathbf{u}_j^T \mathbf{u}_k = \delta_{jk}$ we obtain

$$\boxed{|\delta_t|^2 = \sum_i a_i^2 [1 - \eta \lambda_i]^{2t}}$$

iteration approaches the minimum, $\lim_{t \to \infty} |\boldsymbol{\delta}_t| = 0$, only if $|1 - \eta \lambda_i| < 1$ for all i

condition for (local) convergence: $\boldsymbol{\eta} < \boldsymbol{\eta_{max}} = \dfrac{\mathbf{2}}{\boldsymbol{\lambda_{max}}}$

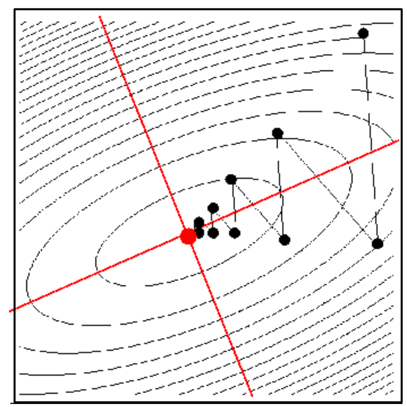
$$\eta < \frac{\eta_{max}}{2} = \frac{1}{\lambda_{max}}$$



$$1 - \eta\lambda_{max} > 0$$

smooth convergence

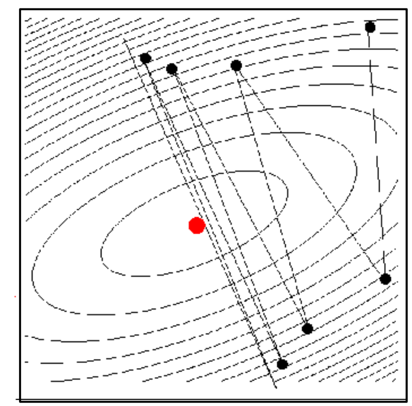
$$\frac{1}{\lambda_{max}} < \eta < \frac{2}{\lambda_{max}}$$



$$1 - \eta\lambda_{max} < 0$$

oscillations

$$\eta > \eta_{max} = \frac{2}{\lambda_{max}}$$



$$1 - \eta\lambda_{max} < -1$$

divergence

... the above considerations

- are only valid close to the minimum

  local minima can have completely different characteristics (λ_{max})

- do not concern global convergence properties

  e.g. the choice of the learning rate far from a minimum

potential problems:

- E can have (many) local minima far from global optimality
- initial conditions determine which minimum will be approached
- anistropic curvatures can cause strong oscillations
- E can have *saddle points* with $\nabla E = 0$ and/or *flat regions* with $\nabla E \approx 0$

  gradient learning can slow down drastically by, e.g., *plateau states*, see below

some modifications:

- improved gradient descent: e.g. time dependent $\eta(t)$

  momentum: $\Delta \mathbf{w}_{t+1} = -\eta \, \boldsymbol{\nabla} E + a \, \Delta \mathbf{w}_t$ “keep going”

some modifications:

- improved gradient descent: e.g. time dependent $\eta(t)$

  momentum: $\Delta \mathbf{w}_{t+1} = -\eta \, \boldsymbol{\nabla} E + a \, \Delta \mathbf{w}_t$ "keep going"

- sophisticated **optimization methods**:
  line search procedures, conjugate gradient, second order methods,

  e.g. Newton's method ("matrix update" employs H), ...

some modifications:

- improved gradient descent: e.g. time dependent $\eta(t)$

  momentum: $\Delta\mathbf{w}_{t+1} = -\eta \, \nabla E + a \, \Delta\mathbf{w}_t$ "keep going"

- sophisticated **optimization methods**:
  line search procedures, conjugate gradient, second order methods,
  e.g. Newton's method ("matrix update" employs H), ...

- **different learning rates** for different weights, examples:
  - heuristics: $\eta \propto 1/N$ for input-to-hidden, $\eta \propto 1/K$ for hidden-to-output weights
  - simplified version of "matrix update" (assume H is approximately diagonal):

    update each weight w_j with a learning rate $\eta_j \propto 1 \Big/ \frac{\partial^2 E}{\partial w_j^2}$
  - learning algorithms realize *descent* in E as long as $\Delta\mathbf{w} \cdot \nabla E < 0$

some modifications:

- improved gradient descent: e.g. time dependent $\eta(t)$

  momentum: $\Delta \mathbf{w}_{t+1} = -\eta \, \nabla E + a \, \Delta \mathbf{w}_t$ "keep going"

- sophisticated **optimization methods**:
  line search procedures, conjugate gradient, second order methods,
  e.g. Newton's method ("matrix update" employs H), ...

- **different learning rates** for different weights, examples:
  – heuristics: $\eta \propto 1/N$ for input-to-hidden, $\eta \propto 1/K$ for hidden-to-output weights
  – simplified version of "matrix update" (assume H is approximately diagonal):
  update each weight w_j with a learning rate $\eta_j \propto 1 \Big/ \frac{\partial^2 E}{\partial w_j^2}$
  – learning algorithms realize *descent* in E as long as $\Delta \mathbf{w} \cdot \nabla E < 0$

- construction of alternative **well-behaved cost functions,** one example:

$$E = \sum_{\mu} \begin{cases} \gamma \, (\sigma - \tau)^2 & \text{if } \operatorname{sign}(\sigma) = \operatorname{sign}(\tau) \\ (\sigma - \tau)^2 & \text{if } \operatorname{sign}(\sigma) \neq \operatorname{sign}(\tau) \end{cases}$$ with γ increasing from 0 to 1.

small γ: emphasis on correct <u>sign</u> of the output large γ: fine tuning of σ

stochastic gradient descent

stochastic approximation (on-line gradient descent)

cost function $E = \frac{1}{P} \sum_{\mu=1}^{P} e^{\mu} \equiv \overline{e^{\mu}}$ is an **empirical average** over examples

$\rightarrow$ simple approximation of ∇E by ∇e^{μ} for one example only

- select one $\mu \in \{1, 2, \ldots, P\}$ with equal probabilty $1/P$
- single step: $\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}_t = \mathbf{w}_t - \eta \, \nabla \, e^{\mu}\big|_{\mathbf{w}_t}$

stochastic gradient descent

stochastic approximation (on-line gradient descent)

cost function $E = \frac{1}{P}\sum_{\mu=1}^{P} e^{\mu} \equiv \overline{e^{\mu}}$ is an **empirical average** over examples

$\rightarrow$ simple approximation of ∇E by ∇e^{μ} for one example only

- select one $\mu \in \{1, 2, \ldots, P\}$ with equal probabilty $1/P$
- single step: $\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}_t = \mathbf{w}_t - \eta \, \nabla \, e^{\mu}|_{\mathbf{w}_t}$

– computationally cheap compared to *off-line (batch)* gradient descent

– *intrinsic noise*, fewer problems with local minima, flat regions etc.

(when) does the procedure converge?

behavior close to a (local) minimum $\mathbf{w}^*$ of E?

averaged learning step: $\overline{\Delta \mathbf{w}} = -\eta \, \overline{\nabla \, e^\mu\big|_{\mathbf{w}}} = -\frac{\eta}{P} \sum_{\mu=1}^{P} \nabla \, e^\mu\big|_{\mathbf{w}} = -\eta \, \nabla \, E\big|_{\mathbf{w}}$

$$\overline{\Delta \mathbf{w}} = 0 \quad \text{for} \quad \mathbf{w} \rightarrow \mathbf{w}^*$$

averaged learning step: $\overline{\Delta \mathbf{w}} = -\eta \, \overline{\boldsymbol{\nabla} \, e^{\mu}|_{\mathrm{w}}} = -\frac{\eta}{P} \sum_{\mu=1}^{P} \boldsymbol{\nabla} \, e^{\mu}|_{\mathrm{w}} = -\eta \, \boldsymbol{\nabla} \, E|_{\mathrm{w}}$

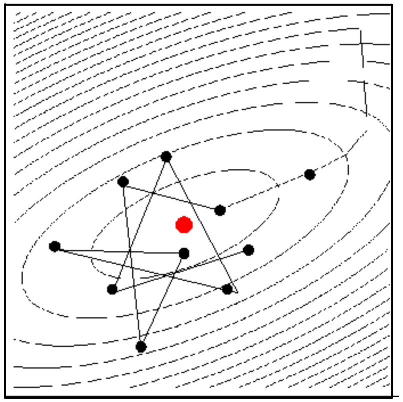
$\overline{\Delta \mathbf{w}} = 0$ for $\mathbf{w} \rightarrow \mathbf{w}^{*}$

averaged length of $\Delta \mathbf{w}$: $\overline{(\Delta \mathbf{w})^{2}} = \eta^{2} \, \overline{\left(\boldsymbol{\nabla} e^{\mu}|_{*} \right)^{2}} > 0$

(0 is possible <u>if</u> all $e^{\mu} = 0$)

for constant rate $\eta > 0$: $\lim_{t \rightarrow \infty} \left(\Delta \mathbf{w}_{t} \right)^{2} > 0$

(fluctuations remain non-zero)

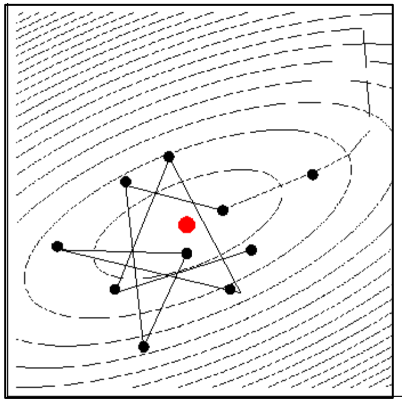


averaged learning step: $\overline{\Delta \mathbf{w}} = -\eta \, \overline{\nabla e^{\mu}|_{\mathrm{w}}} = -\frac{\eta}{P} \sum_{\mu=1}^{P} \nabla e^{\mu}|_{\mathrm{w}} = -\eta \, \nabla E|_{\mathrm{w}}$

$\overline{\Delta \mathbf{w}} = 0$ for $\mathbf{w} \to \mathbf{w}^*$

averaged length of $\Delta \mathbf{w}$: $\overline{(\Delta \mathbf{w})^2} = \eta^2 \, \overline{(\nabla e^{\mu}|_*)^2} > 0$

(0 is possible if all $e^{\mu} = 0$)



for constant rate $\eta > 0$: $\lim_{t\to\infty} (\Delta \mathbf{w}_t)^2 > 0$

(fluctuations remain non-zero)

convergence in the sense of $(\Delta \mathbf{w})^2 \to 0$ only if $\eta(t) \to 0$ for $t \to \infty$

one can show: $\lim_{t\to\infty} \sum_t \eta(t) \to \infty$ but $\lim_{t\to\infty} \sum_t \eta(t)^2 < \infty$ is required

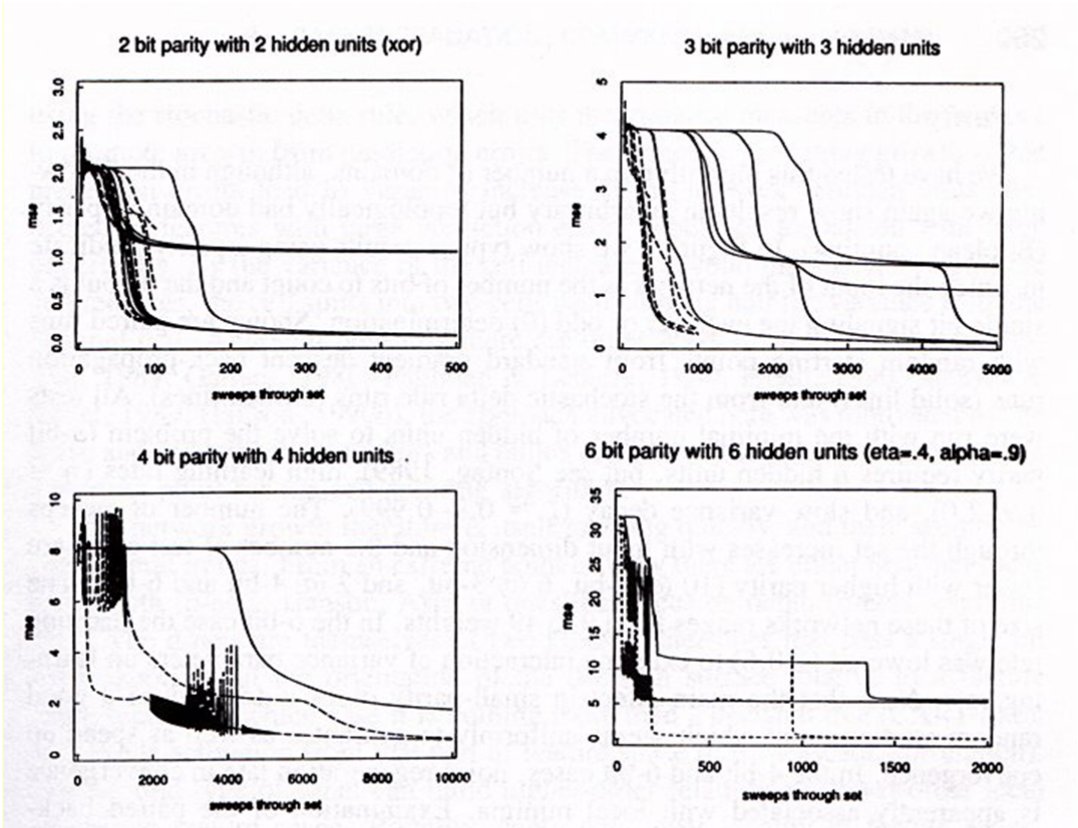
satisfied by, e.g. $\eta(t) \propto \frac{1}{t}$ for large t **learning rate schedules,** e.g. $\eta(t) = \frac{a}{b+t}$

alternative: averages of $\mathbf{w}$ over recent (or all) gradient steps

Plateau states

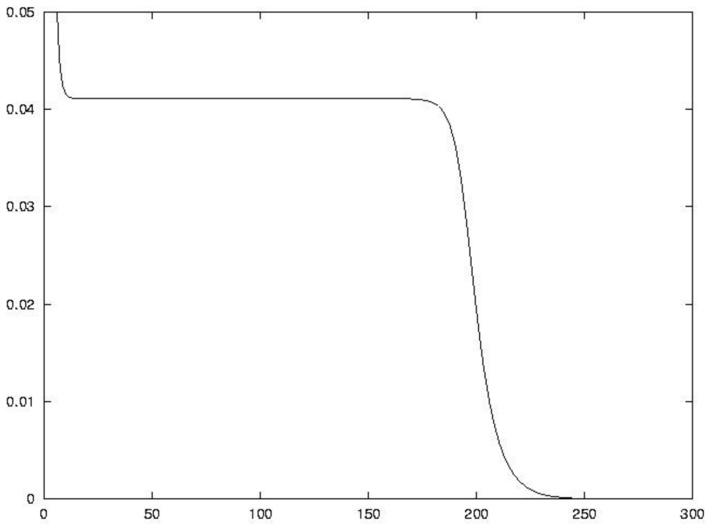
frequent observation:

training of multilayer networks is delayed by *quasi-stationary plateaus*



(S.J. Hanson, in: Y. Chauvin and D.E. Rummelhart, *Backpropagation: Theory, Architectures, and Applications*, 1995) example: a two-layer network trained from reliable, perfectly realizable data by on-line gradient descent

ε_g



number of examples $P/(KN)$

- fast initial decrease of ε_g
- fast asymptotic decrease of $\varepsilon_g \to 0$ (here: matching complexity)
- plateau state: **unspecialized** h.u. with $\mathbf{w}_k \sim \mathbf{w}_o + noise$ have all obtained some (the same) information about the unknown rule

occurence of plateaus relates to symmetries:

the network output is invariant under **permutations of hidden units**

perfectly symmetric state corresponds to a flat region (saddle) in E

successful learning requires **specialization** and can be delayed significantly

analysed in depth in the statistical physics community (1990's)

problem re-discovered in deep learning

university of groningen

Shallow and deep networks

shallow and deep architectures

- ***shallow networks***

  frequently used: input-hidden-output architectures, e.g. *N-M-1*

  often shown to be ***universal approximators / classifiers***

  easy to implement

  efficient, fast training, e.g. by backpropagation

  examples: Committee/Parity Machine

  Extreme Learning Machine

  Radial Basis Function Networks

  special case: ***Reservoir Computing***

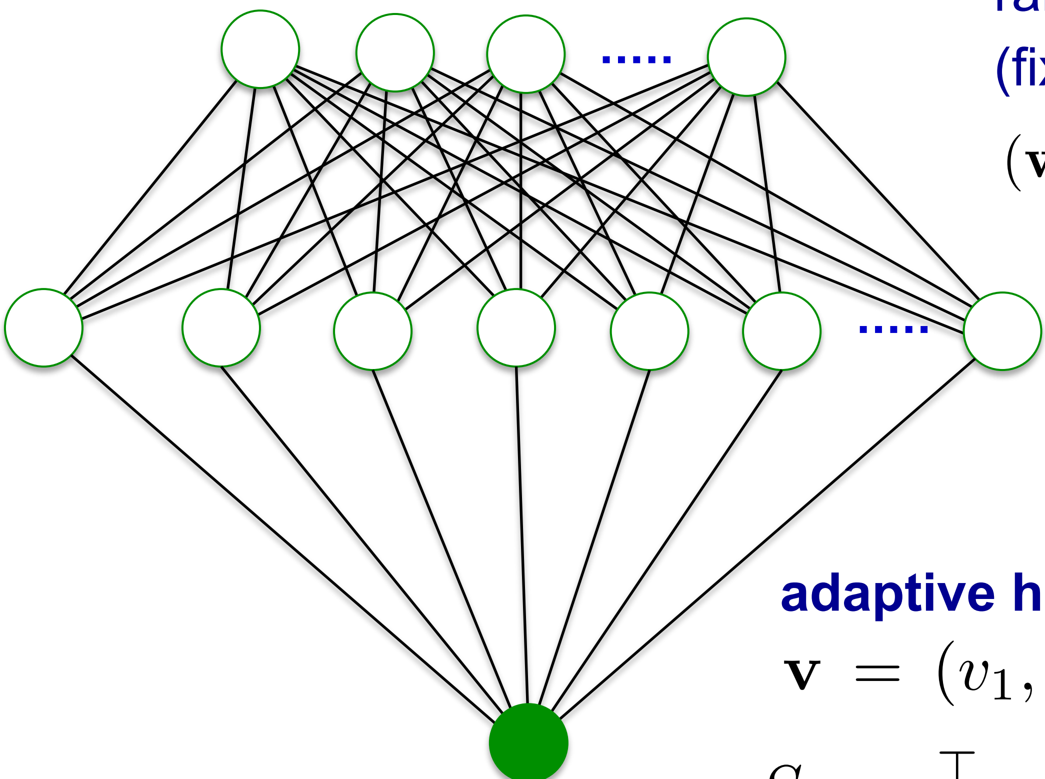
  replace hidden layer by a dynamical network with intra-layer connections and/or internal dynamics

- ***deep networks*** (at a glimpse)

  deep learning, convolutional neural networks

Extreme Learning Machine (ELM)

input: N-dim. feature vectors $\mathbf{x}$



random input-to-hidden weights
(fixed, non-adaptive)

$$(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M)^\top \in I\!R^{M\times N}$$

hidden layer: M units (e.g. M>N)

e.g. sigmoidal $\sigma_j = g\left(\mathbf{w}_j^\top \mathbf{x}\right)$

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_M)^\top$$

adaptive hidden-to-output weights

$$\mathbf{v} = (v_1, v_2, \ldots, v_M)^\top$$

$S = \mathbf{v}^\top \boldsymbol{\sigma}$ **linear output**

training (hidden-to-output only!) by **regression** w.r.t. given targets

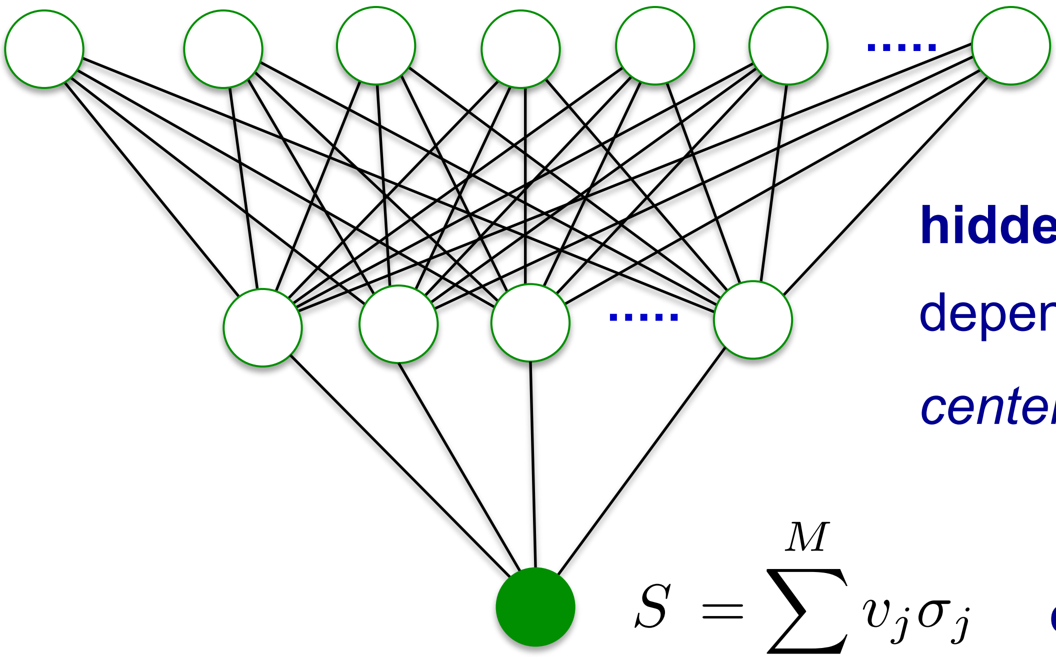
e.g. least square solution obtained as Moore-Penrose pseudoinverse

Extreme Learning Machine (ELM)

- Huang et al. (*IJCNN* 2004): concept and name,
  see original (and later) publications provided in *Nestor*
- triggered numerous publications, even specialized journals and
  ELM conferences
- serious, on-going **debate about originality** of the concept, see
  Wikipedia entry and the Comment by Wang and Wan in *IEEE TNN* (2008)
  see also: http://elmorigin.weebly.com. One example early paper with
  similar ideas: Schmidt, Kraaijveld, Duin, *ICPR* 1992

- conceptual **similarity to SVM** is discussed in, e.g., Frenay and Verleysen:
  Using SVMs with randomised feature spaces: an extreme learning approach

Radial Basis Functions (RBF) networks



input: N-dim. feature vectors $\mathbf{x}$

hidden layer: M units, activation (*) depends on distance of $\mathbf{x}$ from *center* $\mathbf{c_i}$: $\sigma_i = g(|\mathbf{x} - \mathbf{c}_i|)$

$$S = \sum_{j=1}^{M} v_j \sigma_j$$ e.g. linear output unit

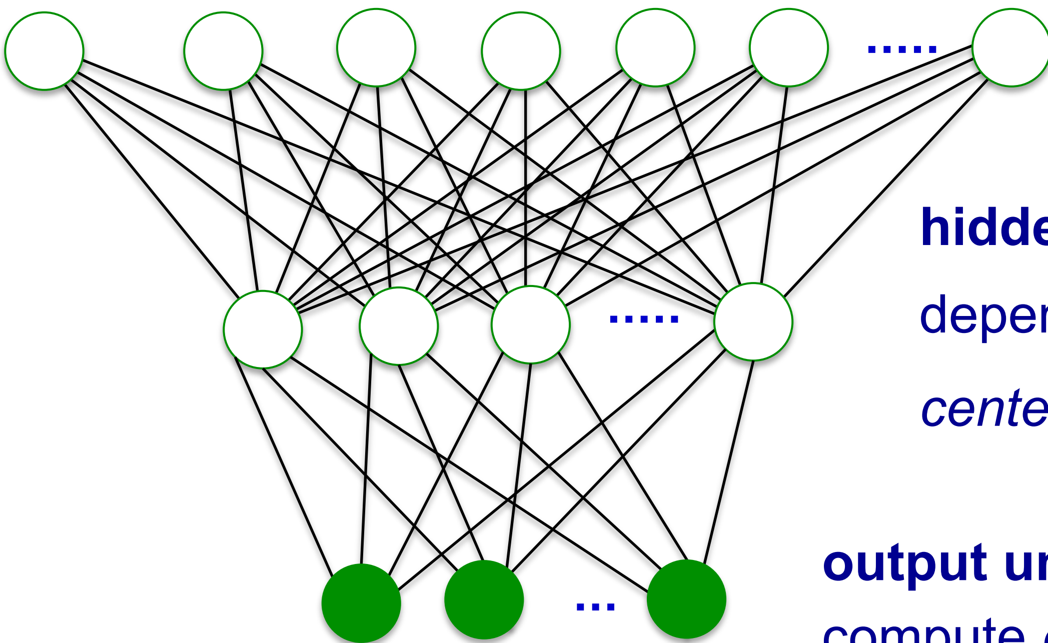
adaptive: *centers* $\mathbf{c_i}$ (e.g. by unsupervised vector quantization)

weights $\mathbf{v}$ (e.g. by least squares regression for given centers)

* example: $\sigma_i = \exp\left[-\beta\left(\mathbf{x}-\mathbf{c_i}\right)^2\right]$ unnormalized (local)

beyond RBF: $\sigma_i = \dfrac{\exp\left[-\beta\left(\mathbf{x}-\mathbf{c_i}\right)^2\right]}{\sum_{j=1}^{M}\exp\left[-\beta\left(\mathbf{x}-\mathbf{c_j}\right)^2\right]}$ normalized (constant total activation)

RBF classifier



input: N-dim. feature vectors $\mathbf{x}$

hidden layer: M units, activation (*) depends on distance of $\mathbf{x}$ from *center* $\mathbf{c_i}$: $\sigma_i = g(|\mathbf{x} - \mathbf{c}_i|)$

output units represent **C classes,** compute ***class-membership scores***

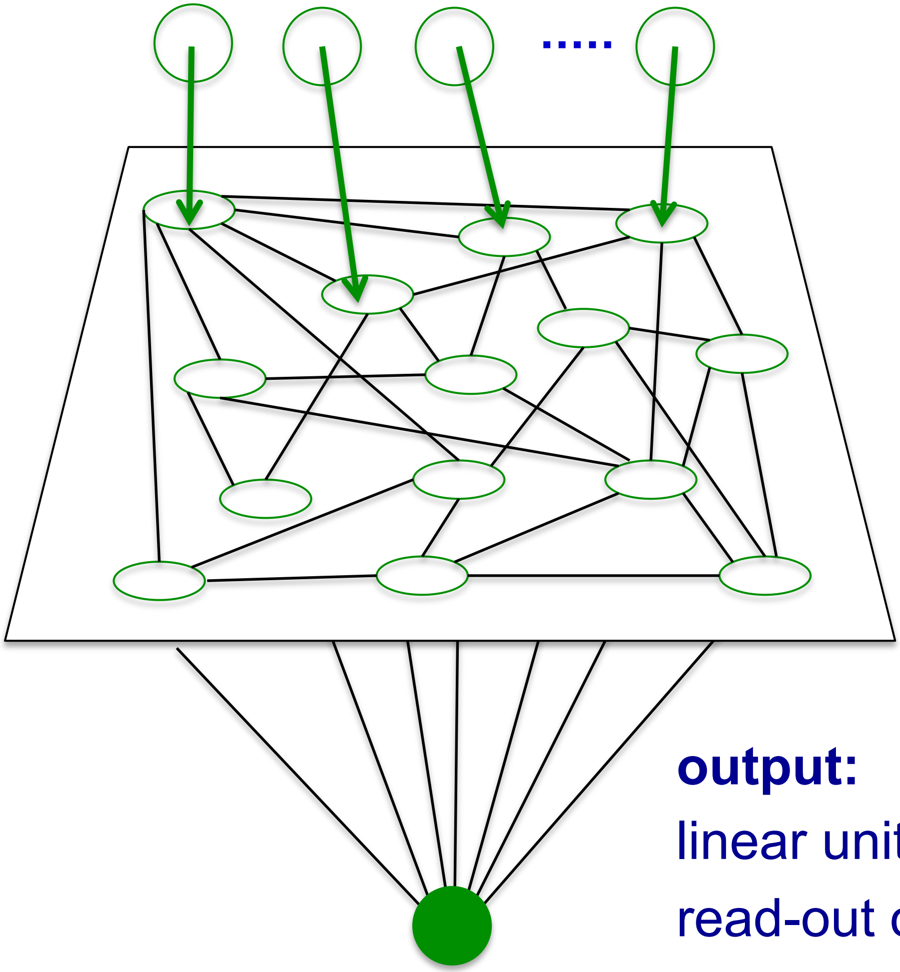
adaptive hidden-to-output weights (C *pseudo-regression* problems) or fixed*, pre-wired* function

assign input to class with **maximum score**

very similar concept: Learning Vector Quantization

[**RBF-networks**: see book by Bishop for detailed discussion and references]

Reservoir Computing



input: enforce (initial) state in the reservoir network (or a subset of units)

recurrent network as *reservoir:*
fixed random connections, represents inputs by different internal states

- leaky integrator units
  liquid state machine
- sparsely connected attractor net
  echo-state networks

output:
linear unit with **adaptive weights**
read-out of the reservoir state

regression training: comparison with target output for a given set of input/output examples

most prominent examples in the literature:
(see *Nestor* for original publications and review articles)

echo-state networks	[Jaeger 2001]
liquid state machines	[Natschlaeger et al. 2002]
decorrelation-backpropagation	[Steil 2004]

see also: http://reservoir-computing.org feed-forward networks with a large (?) number of layers and units

combination of several concepts / methods / tricks

training became feasible due to ...

increased **computational power** (backpropagation of error)

sparse connectivity (e.g. **convolutional networks**)

weight sharing and ***pooling***

availability of **huge data sets**

simplified transfer functions („rectified linear units" $g(x)=max\{0,x\}$)

efficient **regularization** techniques (e.g. „dropout")

main application areas with excellent performance:

data with spatial / temporal structure

image (faces, digits, scenes) classification / recognition

Goodfellow, Bengio, Courville: **Deep Learning, 2016**

The fishermen in the north of Spain
have been using Deep Networks for centuries.
Their contribution should be recognized...

Javier Movellan

From a discussion about the origins of the term "Deep Networks" in the Connectionists mailing list
http://dove.ccs.fau.edu/dawei/ICM/connectionists.html