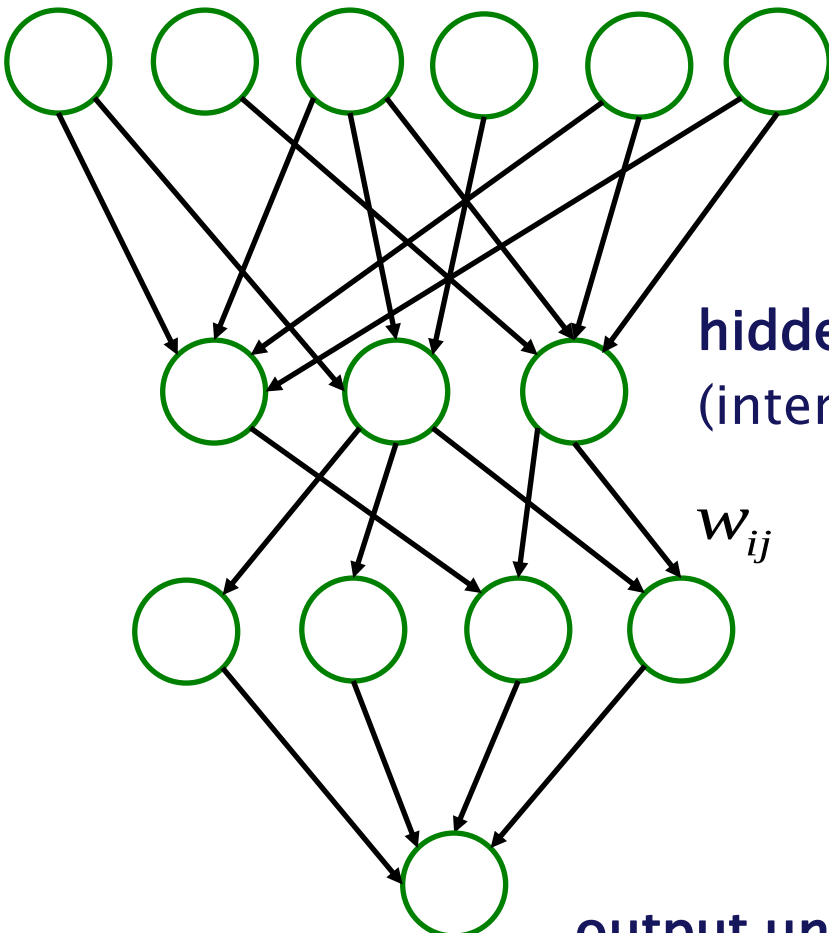
4. Gradient based Learning

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

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)

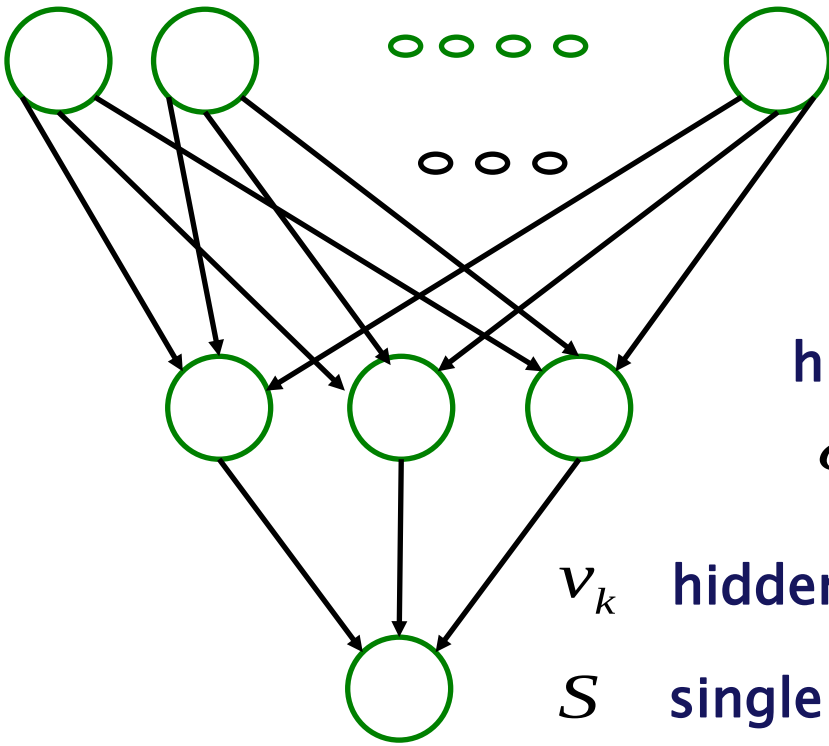
w_{ij}

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

↑ previous layer only

output unit(s)
(function of input vector)

convergent two-layer architecture



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

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

hidden layer units

$$\sigma_k = g\left(\sum w_j^{(k)} \xi_j\right)$$

v_k hidden to output weights

S single output unit

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

$$S = g\left(\sum_{k=1}^{K} v_k \, \sigma_k\right) = g\left(\sum_{k=1}^{K} v_k g\left(\sum w_j^{(k)} \xi_j\right)\right)$$

parameterized by set of all weights (and threshold)

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 continuous function:

$$\boldsymbol{\xi} \in \mathbb{R}^N \quad \rightarrow \quad \sigma(\boldsymbol{\xi}) \in \mathbb{R}$$

(here: single output unit)

Learning as **regression problem**

set of examples $\left\{ \boldsymbol{\xi}^{\boldsymbol{\mu}}, \tau(\boldsymbol{\xi}^{\boldsymbol{\mu}}) \right\}_{\mu=1}^{P}$ with cont. labels $\tau \in \mathbb{R}$

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

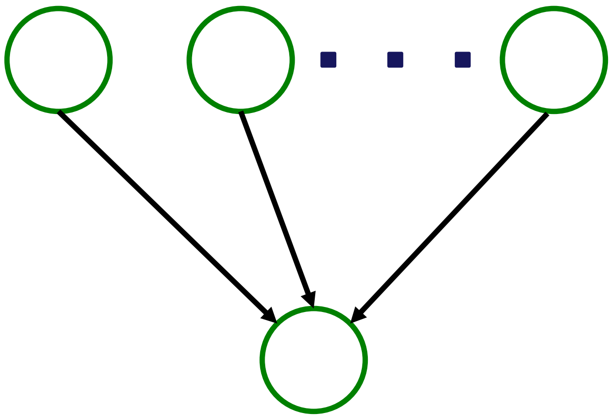
just <u>one</u> very popular and plausible choice:

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

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

- 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



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

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

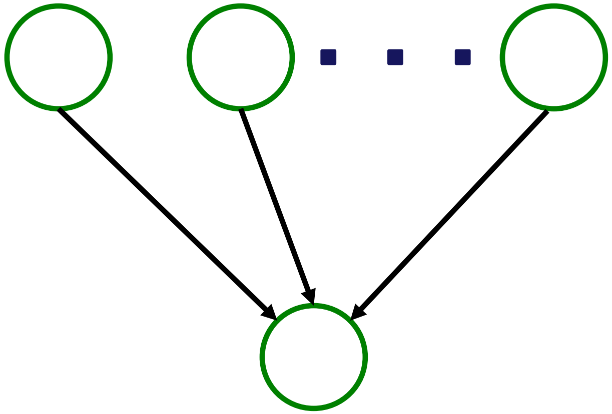
$$\sigma = g\left(\sum_{j=1}^{N} w_j \xi_j\right)$$

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

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

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

a single unit



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

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

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

frequent choice: g(x)=tanh(x) g'(x)=1-tanh^2(x)

$$\nabla_w E = \frac{1}{P} \sum_{\mu=1}^{P} \left(\tanh(\mathbf{w}\cdot\boldsymbol{\xi}^\mu) - \tau(\boldsymbol{\xi}^\mu)\right) \left(1\text{-}\tanh^2(\mathbf{w}\cdot\boldsymbol{\xi}^\mu)\right) \boldsymbol{\xi}^\mu$$

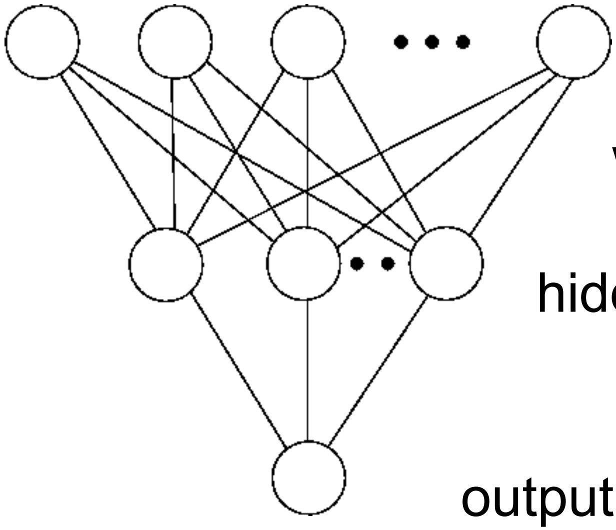
single example contribution:

$$\nabla_w e^\mu = \left(\tanh(\mathbf{w}\cdot\boldsymbol{\xi}^\mu) - \tau(\boldsymbol{\xi}^\mu)\right) \left(1\text{-}\tanh^2(\mathbf{w}\cdot\boldsymbol{\xi}^\mu)\right) \boldsymbol{\xi}^\mu$$

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

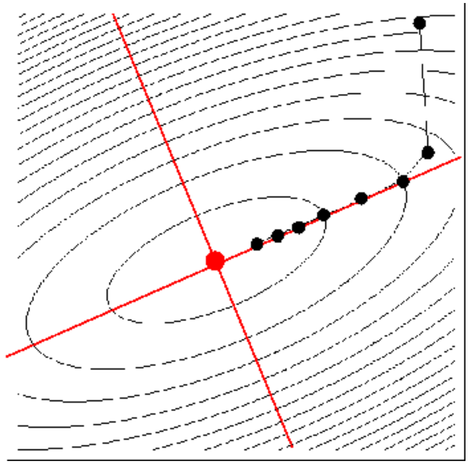
output $\sigma(\boldsymbol{\xi}) \;=\; h\left(\sum_{j=1}^{K} v_j\, g(\mathbf{w}_j \cdot \boldsymbol{\xi})\right)$

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

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



simple gradient based minimization of E:

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

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

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 $H^*_{ij} = \left. \frac{\partial^2 E}{\partial w_i \, \partial w_j} \right|_*$

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

in detail:

$$w_t = w_{t-1} - \eta \; \nabla E\,|_{w_{t-1}}$$

$$\boldsymbol{\delta}_t = \boldsymbol{\delta}_{t-1} - \eta \; \nabla E\,|_{w_{t-1}} \approx [I - \eta H^*] \quad \boldsymbol{\delta}_{t-1} \approx [I - \eta H^*]^t \boldsymbol{\delta}_o$$

$$\boldsymbol{\delta}_o = \sum_i a_i \; \mathbf{u_i} \qquad \mathrm{H}^* \; \mathbf{u}_i = \lambda_i \; \mathbf{u}_i \qquad (1 - \eta \mathrm{H}^*)\, \mathbf{u}_i = (1 - \eta \lambda_i)\, \mathbf{u}_i$$

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

$$\boldsymbol{\delta}_t^2 = \boldsymbol{\delta}_t \cdot \boldsymbol{\delta}_t = \sum_{i,j} a_i [1 - \eta \lambda_i]^t \quad a_j \; [1 - \eta \lambda_j]^t \quad \underbrace{\mathbf{u_i} \cdot \mathbf{u_j}}$$

=1 if i=j

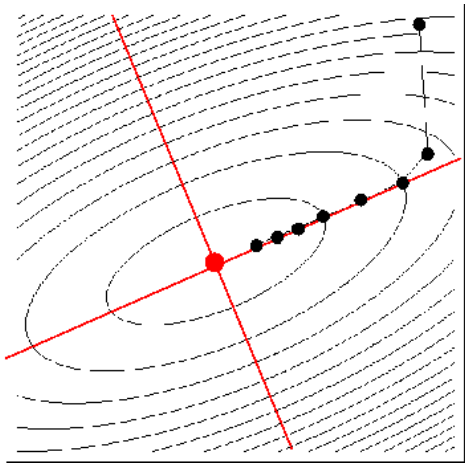
=0 else

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

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: $\boldsymbol{\eta < \eta_{max} = \dfrac{2}{\lambda_{max}}}$ (largest eigenvalue of H*)

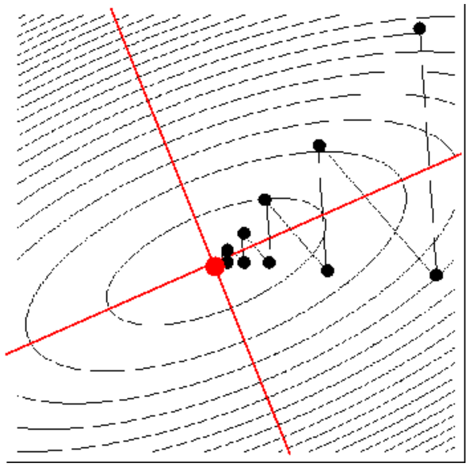
$$\eta < \frac{\eta_{max}}{2} = \frac{1}{\lambda_{max}}$$



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

smooth convergence

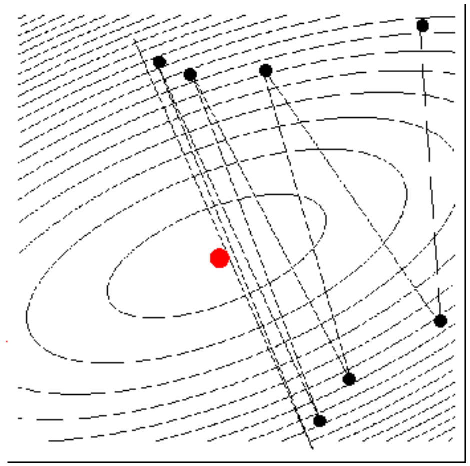
$$\frac{1}{\lambda_{max}} < \eta < \frac{2}{\lambda_{max}}$$



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

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: $\Delta \mathbf{w}_{t+1} = -\eta \nabla E + a \Delta \mathbf{w}_t$ "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 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 $\boldsymbol{\nabla} E$ by $\boldsymbol{\nabla} e^{\mu}$ for one example only

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

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

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

(when) does the procedure converge?

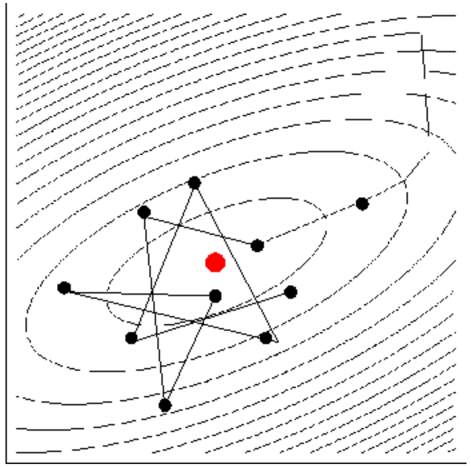
behavior close to a (local) minimum $\mathbf{w}^*$ of E?

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

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

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

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



generic behavior

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

(fluctuations remain non-zero)

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

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

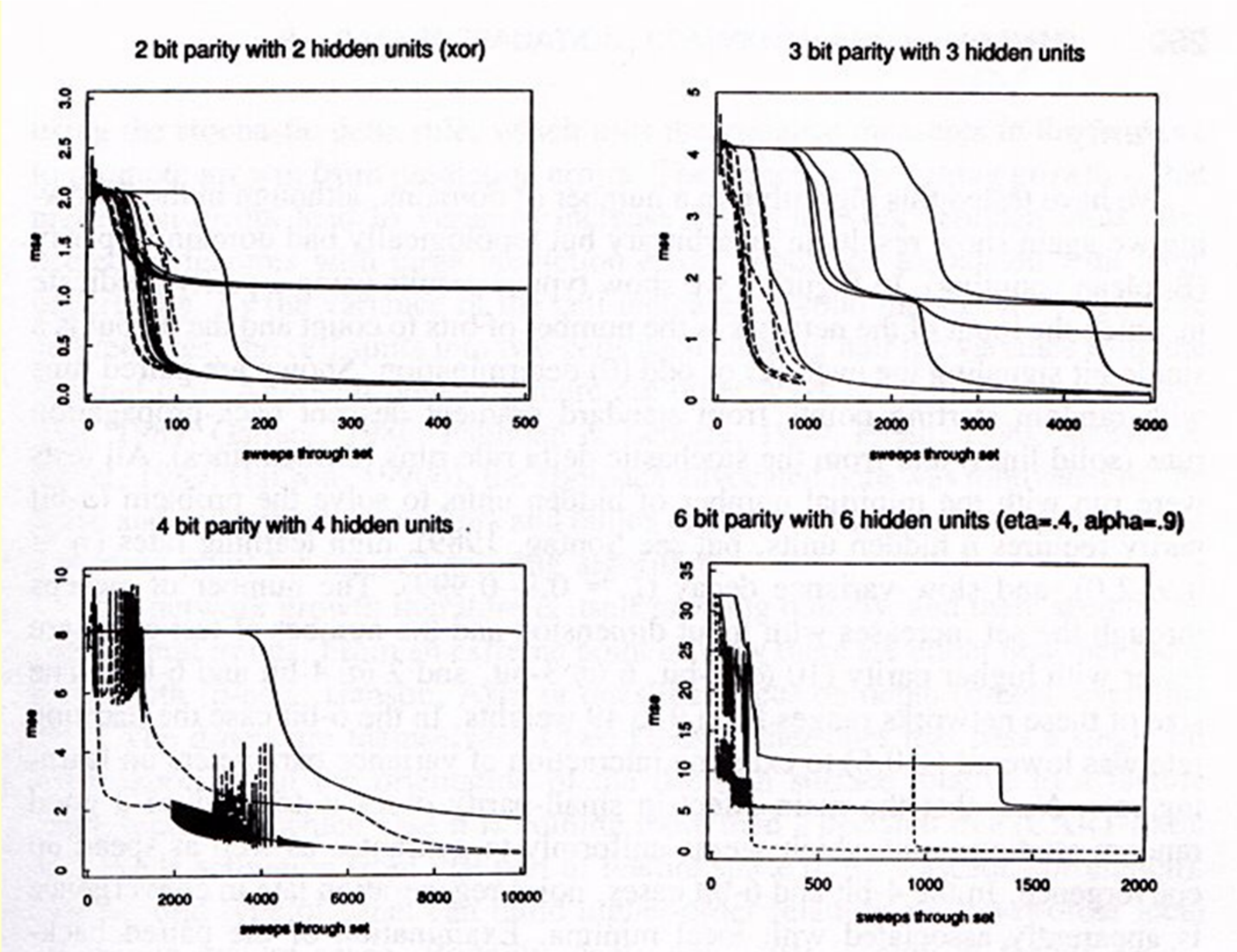
satisfied by, e.g. $\eta(t) \propto \frac{1}{t}$ for large t **learning rate schedules,** e.g. $\eta(t) = \frac{a}{b+t}$

alternative: retain finite learning rate and fluctuations, average weights over time

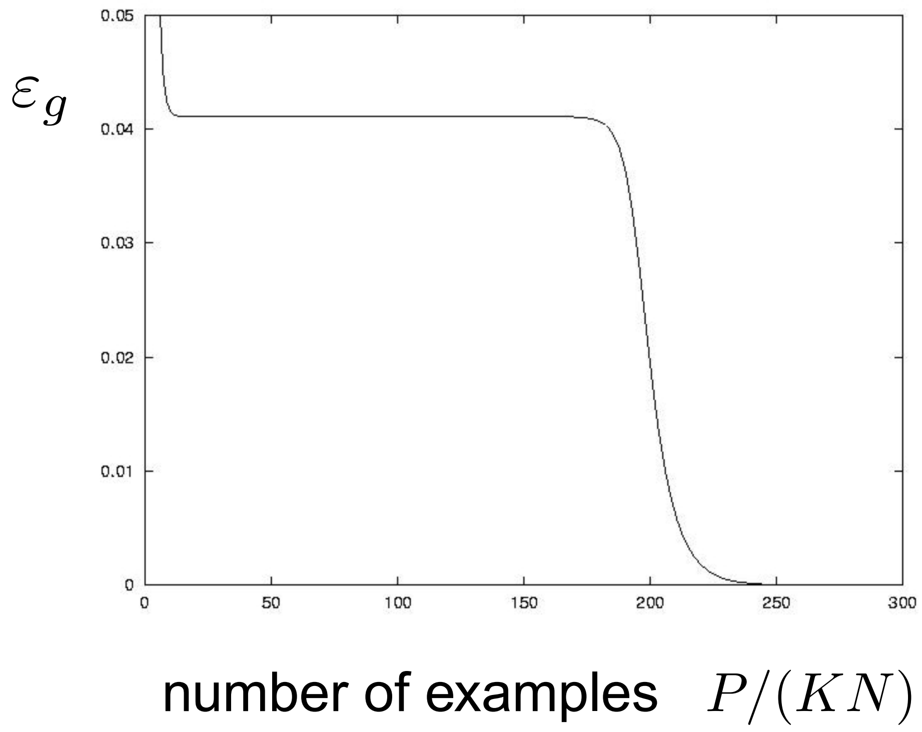
Plateau states

frequent observation:

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



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



- fast initial decreas of ε_g
- fast asymptotic decrease of $\varepsilon_g \rightarrow 0$
  (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

math. analysis: D. Saad and S. Solla (1995), M. Biehl, P.Riegler, C. Wöhler (1996)

Remarks

- the extension to several output units $\{\sigma^l\}_{l=1}^{L}$ is non-trivial

  the choice of cost function $E = \dfrac{1}{2} \sum_{l=1}^{L} \left(\sigma^l - \tau^l\right)^2$ seems plausible

  but a generalized metric $E = \dfrac{1}{2} \sum_{k,l} (\sigma^k - \tau^k)\, A_{kl}\, (\sigma^l - \tau^l)$

  with a suitable $(L \times L)$-matrix A might be more appropriate

- appoximation of a classifier by regression training:

  replace $S^\mu = \pm 1$ by $\sigma^\mu = \pm R \quad |\, R\,| < 1$,

  take $S = \mathrm{sign}(\sigma^\mu)$ as classification output after training

  Note:

  – the decrease of ε_g can be slower than *necesary*

  – for very steep activation functions or $|\, R\,| \approx 1$, the gradient becomes non-informative as $\boldsymbol{\nabla} E \approx 0$