Training of feedforward neural networks

Johan Suykens

KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius

Lecture 3

Overview

- steepest descent
- Newton method
- Levenberg-Marquardt algorithm
- quasi-Newton method
- conjugate gradient method

- overfitting and regularization
- effective number of parameters

Learning and optimization

- Consider *off-line* learning case of

$$\min_{w_{ij}^l} E = \frac{1}{P}\sum_{p=1}^{P} E_p \quad \text{with} \quad E_p = \frac{1}{2}\sum_{i=1}^{N_L}(x_{i,p}^{desired} - x_{i,p}^{L})^2.$$

- Unconstrained nonlinear optimization problem:

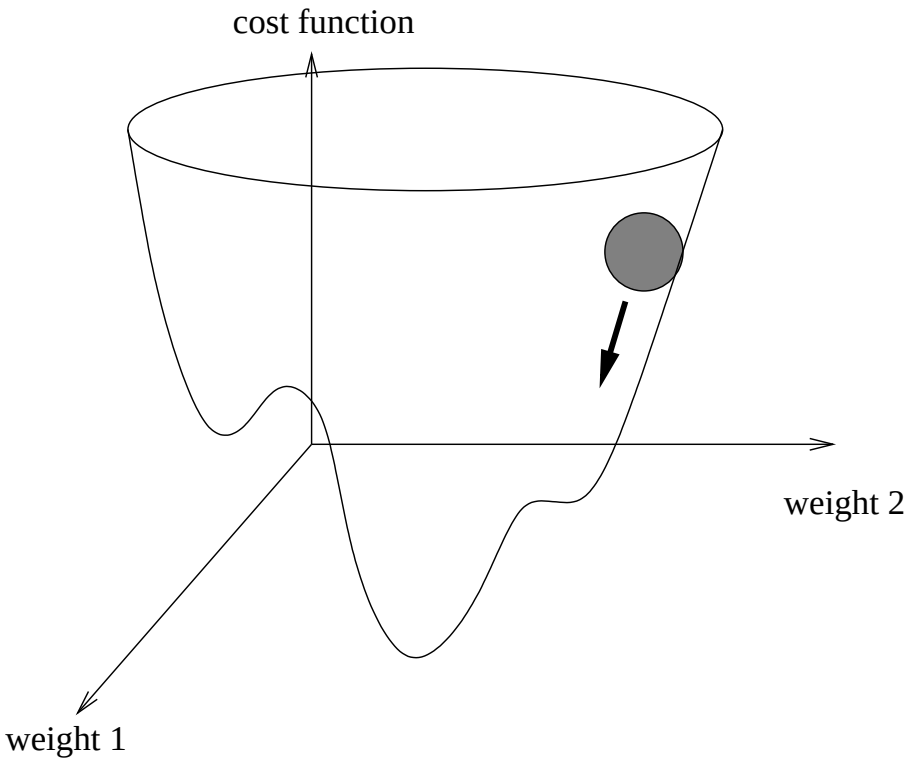
$$\boxed{\min_{x\in\mathbb{R}^n} f(x)}$$

where f denotes the cost function and x the interconnection weights. The simplest optimization algorithm is steepest descent algorithm:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

with x_k is the k-th iterate. Disadvantage: slow convergence rate.

Local optimization



Newton method

- Consider a Taylor expansion around the point x_0

$$f(x) = f(x_0) + g^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x$$

with

$$\begin{aligned} \Delta x &= x - x_0 && \text{(step)} \\ g &= \nabla f(x_0) && \text{(gradient at } x_0\text{)} \\ H &= \nabla^2 f(x_0) && \text{(Hessian at } x_0\text{)} \end{aligned}$$

-

Newton method

- Consider a Taylor expansion around the point x_0

$$f(x) = f(x_0) + g^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x$$

with

$$\begin{array}{rcll} \Delta x & = & x - x_0 & (\text{step}) \\ g & = & \nabla f(x_0) & (\text{gradient at } x_0) \\ H & = & \nabla^2 f(x_0) & (\text{Hessian at } x_0) \end{array}$$

- Optimal step

$$\frac{\partial f}{\partial(\Delta x)} = g + H\,\Delta x = 0 \;\rightarrow\; \boxed{\Delta x = -H^{-1}\, g}$$

The Newton method converges quadratically.

Levenberg-Marquardt method

- Impose a constraint $\|\Delta x\|_2 = 1$ leads to the Lagrangian

$$\mathcal{L}(\Delta x, \lambda) = f(x_0) + g^T \Delta x + \frac{1}{2}\Delta x^T H \Delta x + \frac{1}{2}\lambda(\Delta x^T \Delta x - 1).$$

- From the optimality conditions one obtains

$$\frac{\partial \mathcal{L}}{\partial(\Delta x)} = g + H\Delta x + \lambda \Delta x = 0 \;\rightarrow\; \boxed{\Delta x = -[H + \lambda I]^{-1} g}$$

Special cases: $\lambda = 0$: Newton method
$\lambda \gg$: Steepest descent

Quasi-Newton methods (1)

- **Direct update formulas:** From

$$f(x) = f(x_0) + g^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x$$

it follows that $\nabla_x f = g + H(x - x_0)$ or in general or

$$H d_k = y_k$$

with $d_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$. This means that there is a linear mapping between changes in the gradient and changes in position.

- Assume now that the function f is nonquadratic and B_k is an estimate for H. We have then

$$\begin{array}{lcll} B_{k+1} & = & B_k + \Delta B & (\text{Hessian update}) \\ B_{k+1} d_k & = & y_k & (\text{Quasi} - \text{Newton condition}) \end{array}$$

Quasi-Newton methods (2)

1. **Rank 1 updates**

$$\Delta B = q\, z z^T$$

where q, z follow from Quasi-Newton condition $(B_k + q\, z z^T) d_k = y_k$. Hence $z = y_k - B_k d_k$, $q = \frac{1}{z_k^T d_k}$, giving the rank 1 update formula

$$B_{k+1} = B_k + \frac{(y_k - B_k d_k)(y_k - B_k d_k)^T}{(y_k - B_k d_k)^T d_k}$$

starting with $B_0 = I$.

2. **Rank 2 updates**

$$\Delta B = q_1\, z_1 z_1^T + q_2\, z_2 z_2^T$$

where q_1, q_2, z_1, z_2 must satisfy the Quasi-Newton condition. This yields the BFGS formula (Broyden, Fletcher, Goldfarb, Shanno)

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T d_k} - \frac{(B_k d_k)(B_k d_k)^T}{(B_k d_k)^T d_k}$$

Quasi-Newton methods (3)

- **Inverse update formulas:** Instead of the Quasi-Newton condition $B_{k+1}d_k = y_k$ one takes the condition

$$d_k = R_{k+1}y_k$$

in order to avoid direct inversion of the Hessian H. A well-known procedure is the DFP formula (Davidon, Fletcher, Powell)

$$R_{k+1} = R_k + \frac{d_k d_k^T}{d_k^T g_k} - \frac{R_k y_k y_k^T R_k}{y_k^T R_k y_k}$$

- Use of Quasi-Newton methods: moderate size problems
  For large scale problems one prefers conjugate gradient algorithms

Conjugate gradient algorithms (1)

- **Case of a quadratic function:** Consider

$$f(x) = c + b^T x + \frac{1}{2} x^T A x, \qquad x \in \mathbb{R}^n$$

Given a search direction p_k in $x_{k+1} = x_k + \alpha_k p_k$, find an optimal step size α_k according to

$$\begin{array}{lll} \frac{d}{d\alpha} f(x_k + \alpha p_k)|_{\alpha=\alpha_k} = 0 & \rightarrow & [f'(x_k + \alpha_k p_k)]^T p_k = 0 \\ & \rightarrow & [g(x_{k+1})]^T p_k = 0 \end{array}$$

Conjugate gradient algorithm

$$\begin{cases} p_0 = -g_0 \\ x_{k+1} = x_k + \alpha_k p_k, \qquad \alpha_k = \frac{g_k^T g_k}{p_k^T A p_k} \\ g_{k+1} = g_k + \alpha_k A p_k \\ p_{k+1} = -g_{k+1} + \beta_k p_k, \qquad \beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \end{cases}$$

Conjugate gradient algorithms (2)

One can show that:

1. $\{p_k\}$ are conjugated with respect to A:

$$p_i^T A p_j = \delta_{ij}$$

2. α_k are such that $\frac{d}{d\alpha} f(x_k + \alpha p_k)|_{\alpha=\alpha_k} = 0$

3. The algorithm converges to the minimum in n steps.

Remark: Conjugate gradient algorithms were originally developed to solve a linear system of equations $Ax = y$ with $A = A^T > 0$ (positive definite)

Conjugate gradient algorithms (3)

- **Case of non-quadratic smooth function:**
  Conjugate gradient algorithm

$$\begin{cases} p_0 &= -g_0 \\ x_{k+1} &= x_k + \alpha_k p_k, \quad \alpha_k \text{ from } \min_{\alpha_k} f(x_k + \alpha_k p_k) \text{ (linesearch)} \\ p_{k+1} &= -g_{k+1} + \beta_k p_k \end{cases}$$

  with

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \qquad \text{(Fletcher – Reeves)}$$

$$\beta_k = \frac{g_{k+1}^T (g_{k+1} - g_k)}{g_k^T g_k} \qquad \text{(Polak – Ribiere)}$$

- Modified versions such as scaled conjugate gradient are applied with success to neural network problems (e.g. Møller 1993). For large scale neural networks, one often applies conjugate gradient methods.

Overfitting problem (1)

- Consider an example with training data generated from

$$h(x) = 0.5 + 0.4\sin(2\pi x)$$

and Gaussian noise added to data with standard deviation 0.05.
Consider 10 training data points and 90 test points in the interval [0,1].

- Model: consider polynomials of degree d with $d \in \{1, 2, ..., 10\}$
Polynomial of degree 3:

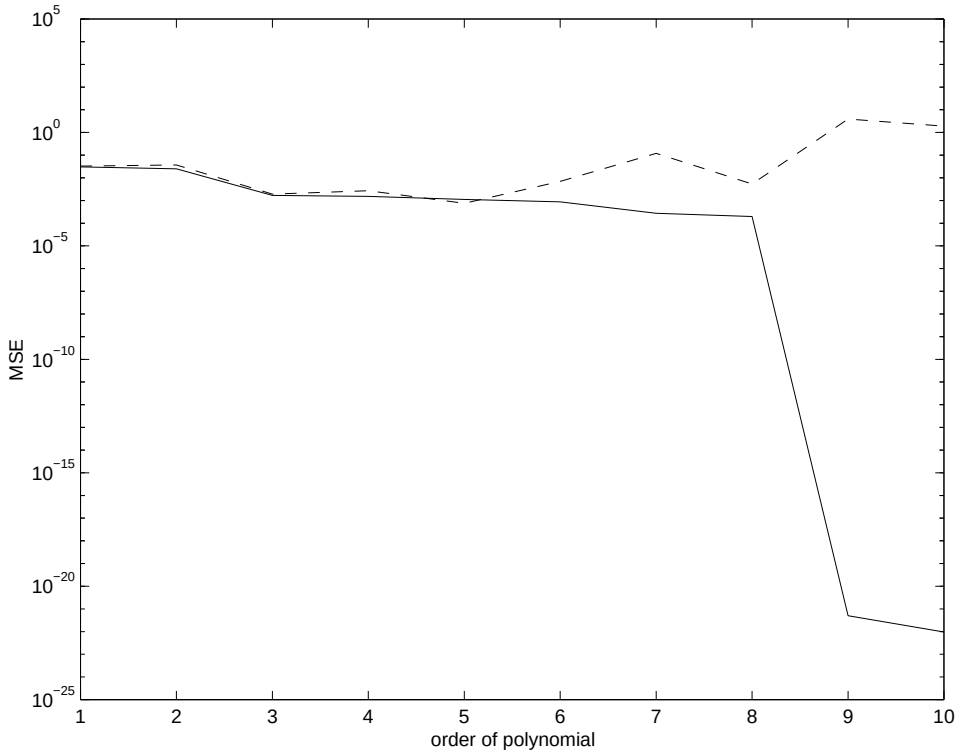
$$y = a_1 x + a_2 x^2 + a_3 x^3 + b$$

Overdetermined set of equations (data $\{x_n, y_n\}_{n=1}^{10}$)

$$\begin{bmatrix} x_1 & x_1^2 & x_1^3 & 1 \\ x_2 & x_2^2 & x_2^3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{10} & x_{10}^2 & x_{10}^3 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{bmatrix}$$

Overfitting problem (2)

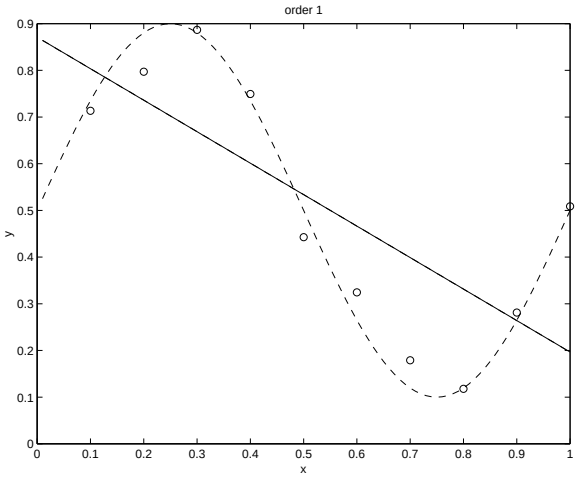
Comparison of polynomial models for training (-) and test (- -) set:



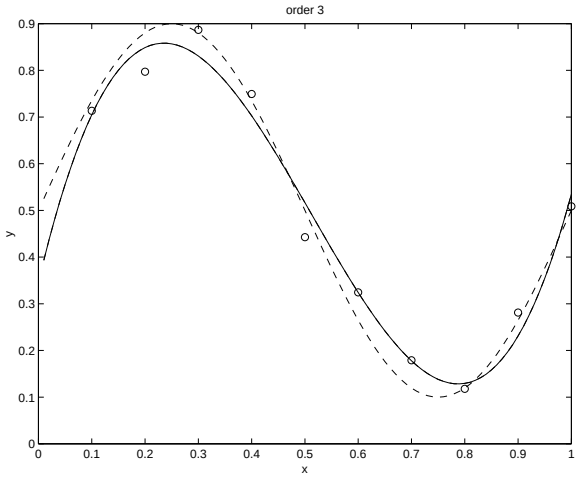
Overfitting occurs for higher degree polynomials.

Overfitting problem (3)

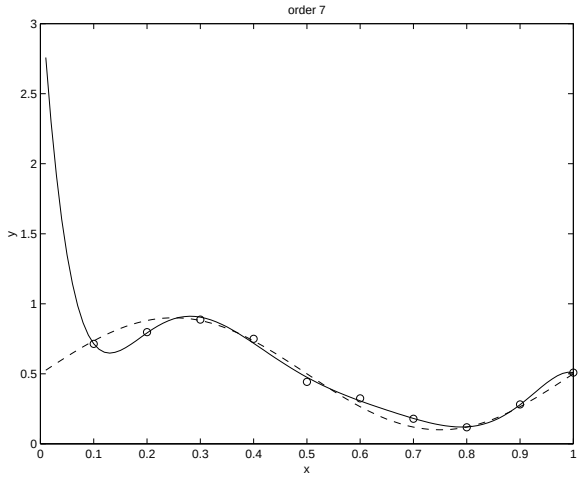
Polynomial of order 1



Polynomial of order 3



Polynomial of order 7



Overfitting occurs for higher degree polynomials (oscillations).

Least squares

Consider given $\mathcal{A}, \mathcal{B}$ in

$$\mathcal{A}\theta = \mathcal{B}, \qquad e = \mathcal{A}\theta - \mathcal{B}$$

with $\mathcal{A} \in \mathbb{R}^{m \times n} (m > n)$, $\mathcal{B} \in \mathbb{R}^m$. Estimate $\theta \in \mathbb{R}^n$.

Least squares:

$$\min_{\theta} J_{LS}(\theta) = \frac{1}{2} e^T e = \frac{1}{2} (\mathcal{A}\theta - \mathcal{B})^T (\mathcal{A}\theta - \mathcal{B})$$

Least squares

Consider given $\mathcal{A}, \mathcal{B}$ in

$$\mathcal{A}\theta = \mathcal{B}, \qquad e = \mathcal{A}\theta - \mathcal{B}$$

with $\mathcal{A} \in \mathbb{R}^{m \times n} (m > n)$, $\mathcal{B} \in \mathbb{R}^m$. Estimate $\theta \in \mathbb{R}^n$.
Least squares:

$$\min_{\theta} J_{LS}(\theta) = \frac{1}{2} e^T e = \frac{1}{2} (\mathcal{A}\theta - \mathcal{B})^T (\mathcal{A}\theta - \mathcal{B})$$

Condition for optimality:

$$\frac{\partial J_{LS}}{\partial \theta} = \mathcal{A}^T \mathcal{A}\theta - \mathcal{A}^T \mathcal{B} = 0$$

Solution:

$$\boxed{\theta_{LS} = (\mathcal{A}^T \mathcal{A})^{-1} \mathcal{A}^T \mathcal{B} = \mathcal{A}^{\dagger} \mathcal{B}}$$

with $\mathcal{A}^{\dagger}$ pseudo inverse matrix.

Ridge regression

Ridge regression:
Apply regularization with regularization term $\|\theta\|_2^2 = \theta^T\theta$

$$\min_{\theta} J_{ridge}(\theta) = \frac{1}{2}e^T e + \frac{1}{2}\lambda\,\theta^T\theta\,, \quad \lambda > 0$$

Ridge regression

Ridge regression:
Apply regularization with regularization term $\|\theta\|_2^2 = \theta^T\theta$

$$\min_{\theta} J_{ridge}(\theta) = \frac{1}{2}e^T e + \frac{1}{2}\lambda\,\theta^T\theta\,, \quad \lambda > 0$$

Condition for optimality:

$$\frac{\partial J_{ridge}}{\partial \theta} = \mathcal{A}^T\mathcal{A}\theta + \lambda\theta - \mathcal{A}^T\mathcal{B} = 0$$

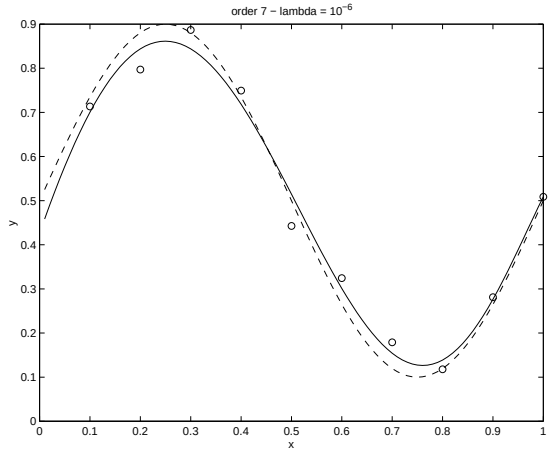
Solution:

$$\boxed{\theta_{ridge} = (\mathcal{A}^T\mathcal{A} + \lambda I)^{-1}\mathcal{A}^T\mathcal{B}}$$

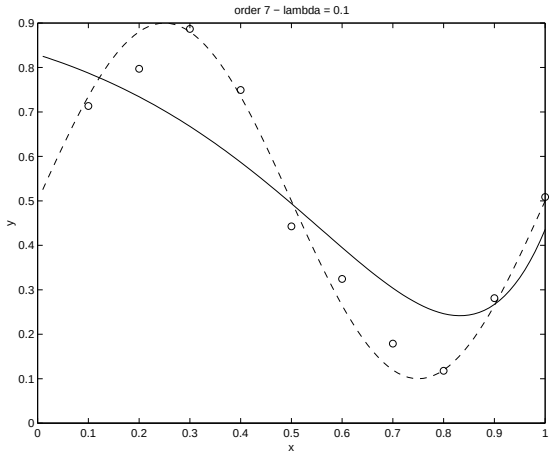
(helpful when $\mathcal{A}^T\mathcal{A}$ is ill conditioned)

Overfitting problem (4)

Regularization for polynomial of order 7: $\lambda = 10^{-6}$ (oscillation is avoided)



$\lambda = 0.1$ (too much regularization)



Training neural networks with regularization

- *Regularization:*

$$\boxed{\tilde{E} = E + \nu\,\Omega(w)}$$

with E the original cost function (e.g. MSE in backprogation), and $\Omega(w)$ the regularization term (also called weight decay term)

$$\Omega(w) = \frac{1}{2}\sum_i w_i^2$$

with ν a positive regularization constant, w the unknown weights.

- Why called weight decay? Suppose $E = 0$, then

$$\frac{dw}{d\tau} = -\eta\frac{\partial \tilde{E}}{\partial w} = -\eta\nu w$$

yielding exponentially decaying weights $w(\tau) = w(0)\exp(-\eta\nu\tau)$.

Analysis of weight decay (1)

- Consider the case of a quadratic cost function, which can be related to a Taylor expansion to the energy function

$$E(w) = E_0 + b^T w + \frac{1}{2} w^T H w$$

with

$$\tilde{E}(\tilde{w}) = E(\tilde{w}) + \nu \frac{1}{2} \tilde{w}^T \tilde{w}$$

Analysis of weight decay (1)

- Consider the case of a quadratic cost function, which can be related to a Taylor expansion to the energy function

$$E(w) = E_0 + b^T w + \frac{1}{2} w^T H w$$

with

$$\tilde{E}(\tilde{w}) = E(\tilde{w}) + \nu \frac{1}{2} \tilde{w}^T \tilde{w}$$

One has

$$\begin{aligned} \frac{\partial E}{\partial w} &= b + Hw = 0 \\ \frac{\partial \tilde{E}}{\partial \tilde{w}} &= b + H\tilde{w} + \nu\tilde{w} = 0 \end{aligned}$$

Consider eigenvalues and eigenvectors of H: $Hu_j = \lambda_j u_j$

- Expand w and $\tilde{w}$: $w = \sum_j w_j u_j, \tilde{w} = \sum_j \tilde{w}_j u_j$

Analysis of weight decay (2)

- One obtains

$$\begin{aligned} Hw - H\tilde{w} - \nu\tilde{w} &= 0 \\ \Rightarrow \quad \sum_j Hw_j u_j - \sum_j H\tilde{w}_j u_j - \nu \sum_j \tilde{w}_j u_j &= 0 \\ \Rightarrow \quad \sum_j (\lambda_j w_j - \lambda_j \tilde{w}_j - \nu\tilde{w}_j) u_j &= 0 \end{aligned}$$

- Important conclusion:

$$\boxed{\tilde{w}_j = \frac{\lambda_j}{\lambda_j + \nu} w_j}$$

If $\lambda_j \gg \nu$ then $\tilde{w}_j \simeq w_j$
If $\lambda_j \ll \nu$ then $|\tilde{w}_j| \ll |w_j|$ (suppressed components)

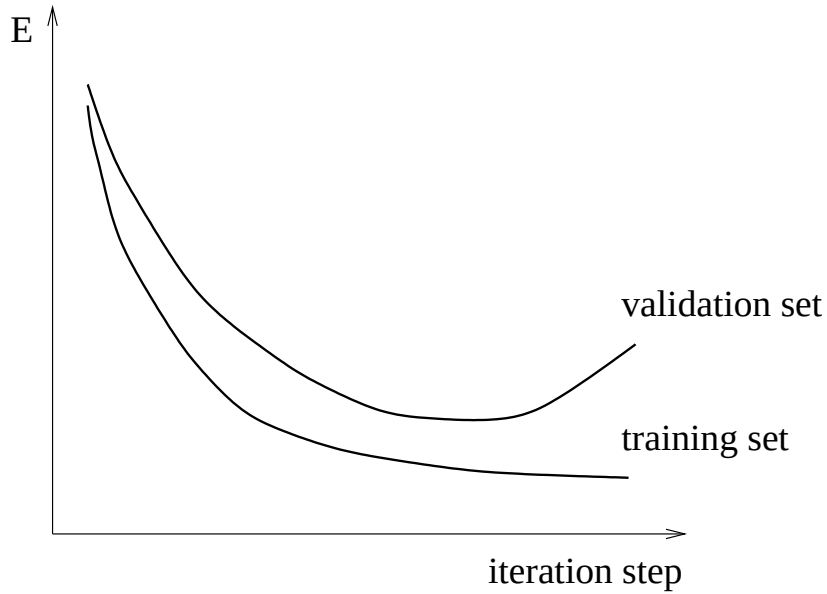
Effective number of parameters

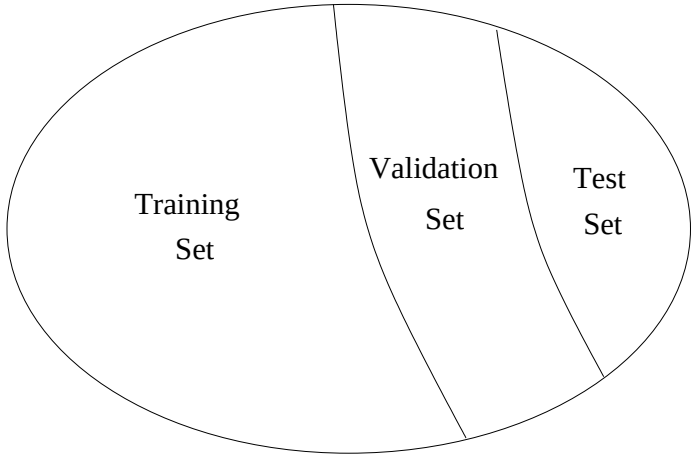
- Thanks to regularization one can implicitly work with less parameters than the number of unknown interconnection weights.

- *Effective number of parameters:*
  The number
  $$(\#\lambda_j) > \nu$$
  is related to the *effective* number of parameters

- Often the effective number of parameters is also defined as
  $$\sum_j \frac{\lambda_j}{\lambda_j + \nu}$$

Early stopping as regularization (1)





- Training set: used for training
- Validation set: used for stopping
- Test set

Stop when minimal error on validation set is reached

Early stopping as regularization (2)

- Quadratic approximation at minimum w^*: $E = E_0 + \frac{1}{2}(w - w^*)^T H (w - w^*)$ with Hessian H positive definite. Consider a simple gradient descent

$$w^{(\tau)} = w^{(\tau-1)} - \eta \nabla E$$

  with iteration step τ and learning rate η and $w^{(0)} = 0$.

- One can show that $w_j^{(\tau)} = \{1 - (1 - \eta\lambda_j)^\tau\} w_j^*$ where $w_j = w^T u_j$ with u_j, λ_j eigenvectors and eigenvalues of H respectively $Hu_j = \lambda_j u_j$. As $\tau \to \infty$ one has $w^{(\tau)} \to w^*$, provided $|1 - \eta\lambda_j| < 1$.

- If training is stopped after τ steps, one has

$$\begin{aligned} w_j^{(\tau)} &\simeq w_j^* \quad \text{when} \quad \lambda_j \gg 1/(\eta\tau) \\ |w_j^{(\tau)}| &\ll |w_j^*| \quad \text{when} \quad \lambda_j \ll 1/(\eta\tau) \end{aligned}$$

  Conclusion: $1/(\eta\tau)$ plays similar role as regularization parameter ν.