Chapter 6: Regression

functional dependency

between features

1. Linear Regression
2. Neural Networks
3. Radial Basis Functions
4. Cross–Validation

} nonlinear regression methods

Prof. Dr. Thomas A. Runkler

Linear Regression

- determine linear dependency between $x^{(i)}$ and $x^{(j)}$ (2 features)

$$x_k^{(i)} \approx a \cdot x_k^{(j)} + b$$

- estimate parameters by minimizing

Square error → $E = \frac{1}{n} \sum_{k=1}^{n} e_k^2 = \frac{1}{n} \sum_{k=1}^{n} \left(x_k^{(i)} - a \cdot x_k^{(j)} - b \right)^2$

Error → to determine if the prediction is good or not

Linear Regression

- necessary criteria for minima of E

$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{k=1}^{n}\left(x_k^{(i)} - a \cdot x_k^{(j)} - b\right) = 0$$

insert b

$$\Rightarrow b = \bar{x}^{(i)} - a \cdot \bar{x}^{(j)}$$

$$\Rightarrow E = \frac{1}{n}\sum_{k=1}^{n}\left(x_k^{(i)} - \bar{x}^{(i)} - a(x_k^{(j)} - \bar{x}^{(j)})\right)^2$$

Linear Regression

- necessary criteria for minima of E

$$\frac{\partial E}{\partial a} = -\frac{2}{n} \sum_{k=1}^{n} (x_k^{(j)} - \bar{x}^{(j)}) \left(x_k^{(i)} - \bar{x}^{(i)} - a(x_k^{(j)} - \bar{x}^{(j)}) \right) = 0$$

$$a = \frac{\sum_{k=1}^{n} (x_k^{(i)} - \bar{x}^{(i)})(x_k^{(j)} - \bar{x}^{(j)})}{\sum_{k=1}^{n} (x_k^{(j)} - \bar{x}^{(j)})^2} = \frac{c_{ij}}{c_{jj}}$$

$c_{ij} \rightarrow$ covariance i j

$c_{jj} \rightarrow$ variance j

- regression parameters can be computed from covariance matrix

Multiple Linear Regression

(more than 2 features)

- determine linear dependency between $x^{(i)}$ and $x^{(j_1)}, \ldots, x^{(j_m)}$

$$x_k^{(i)} \approx \sum_{l=1}^{m} a_l \cdot x_k^{(j_l)} + b$$

- estimate parameters by minimizing

$$E = \frac{1}{n} \sum_{k=1}^{n} e_k^2 = \frac{1}{n} \sum_{k=1}^{n} \left(x_k^{(i)} - \sum_{l=1}^{m} a_l \cdot x_k^{(j_l)} - b \right)^2$$

Multiple Linear Regression

- necessary criteria for minima of E

$$\frac{\partial E}{\partial b} = -\frac{2}{n} \sum_{k=1}^{n} \left(x_k^{(i)} - \sum_{l=1}^{m} a_l \cdot x_k^{(j_l)} - b \right) = 0$$

$$\Rightarrow b = \bar{x}^{(i)} - \sum_{l=1}^{m} a_l \cdot \bar{x}^{(j_l)}$$

$$\Rightarrow E = \frac{1}{n} \sum_{k=1}^{n} \left(x_k^{(i)} - \bar{x}^{(i)} - \sum_{l=1}^{m} a_l \cdot (x_k^{(j_l)} - \bar{x}^{(j_l)}) \right)^2$$

Multiple Linear Regression

- necessary criteria for minima of E

$$\frac{\partial E}{\partial a_r} = -\frac{2}{n}\sum_{k=1}^{n}(x_k^{(j_r)} - \bar{x}^{(j_r)})\left(x_k^{(i)} - \bar{x}^{(i)} - \sum_{l=1}^{m} a_l \cdot (x_k^{(j_l)} - \bar{x}^{(j_l)})\right) = 0$$

$$\Rightarrow \sum_{l=1}^{m} a_l \sum_{k=1}^{n}(x_k^{(j_l)} - \bar{x}^{(j_l)})(x_k^{(j_r)} - \bar{x}^{(j_r)}) = \sum_{k=1}^{n}(x_k^{(i)} - \bar{x}^{(i)})(x_k^{(j_r)} - \bar{x}^{(j_r)})$$

$$\Leftrightarrow \sum_{l=1}^{m} a_l c_{j_l j_r} = c_{i j_r}$$ (linear equation)

- linear equation system can be solved by Gaussian elimination or Cramer's rule
- regression parameters can be computed from covariance matrix

Prof. Dr. Thomas A. Runkler

Pseudo Inverse

(different approach of linear regression)

- write the multiple regression problem in matrix form

$$X = \begin{pmatrix} x_1^{(j_1)} - \bar{x}^{(j_1)} & \dots & x_1^{(j_m)} - \bar{x}^{(j_m)} \\ \vdots & \ddots & \vdots \\ x_n^{(j_1)} - \bar{x}^{(j_1)} & \dots & x_n^{(j_m)} - \bar{x}^{(j_m)} \end{pmatrix}$$

$$Y = \begin{pmatrix} x_1^{(i)} - \bar{x}^{(i)} \\ \vdots \\ x_n^{(i)} - \bar{x}^{(i)} \end{pmatrix}, \quad A = \begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix}$$

$$\begin{aligned} Y &= X \cdot A \\ X^T \cdot Y &= X^T \cdot X \cdot A \\ \underbrace{(X^T \cdot X)^{-1} \cdot X^T}_{\text{pseudo inverse of } X} \cdot Y &= A \end{aligned}$$

Prof. Dr. Thomas A. Runkler

Example Multiple Regression

- data set

$$X = \begin{pmatrix} 6 & 4 & -2 \\ 2 & 1 & -1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 4 & 2 \end{pmatrix}$$

- find linear function $x^{(1)} = f(x^{(2)}, x^{(3)})$

$$x_k^{(1)} \approx \bar{x}^{(1)} + a_1(x_k^{(2)} - \bar{x}^{(2)}) + a_2(x_k^{(3)} - \bar{x}^{(3)})$$

- solution

$$\bar{x}^{(1)} = 2, \quad \bar{x}^{(2)} = 2, \quad \bar{x}^{(3)} = 0, \quad \underset{\text{Covariance}}{C} = \begin{pmatrix} 6 & 3.5 & -2.5 \\ 3.5 & 3.5 & 0 \\ -2.5 & 0 & 2.5 \end{pmatrix}$$

$$\begin{array}{ccccccccccc} c_{22}a_1 & + & c_{32}a_2 & = & c_{12} & \Leftrightarrow & 3.5\,a_1 & & = & 3.5 & \Rightarrow & a_1 = 1 \\ c_{23}a_1 & + & c_{33}a_2 & = & c_{13} & \Leftrightarrow & & 2.5\,a_2 & = & -2.5 & \Rightarrow & a_2 = -1 \end{array}$$

$$\Rightarrow x_k^{(1)} \approx 2 + (x_k^{(2)} - 2) - (x_k^{(3)} - 0) = x_k^{(2)} - x_k^{(3)}$$

Prof. Dr. Thomas A. Runkler

Example Pseudo Inverse

substract mean   substract mean

$$Y = \begin{pmatrix} 6-2 \\ 2-2 \\ 0-2 \\ 0-2 \\ 2-2 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ -2 \\ -2 \\ 0 \end{pmatrix} \qquad X = \begin{pmatrix} 4-2 & -2-0 \\ 1-2 & -1-0 \\ 0-2 & 0-0 \\ 1-2 & 1-0 \\ 4-2 & 2-0 \end{pmatrix} = \begin{pmatrix} 2 & -2 \\ -1 & -1 \\ -2 & 0 \\ -1 & 1 \\ 2 & 2 \end{pmatrix}$$

$$A = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

$$= \left(\begin{pmatrix} 2 & -1 & -2 & -1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & -2 \\ -1 & -1 \\ -2 & 0 \\ -1 & 1 \\ 2 & 2 \end{pmatrix} \right)^{-1} \begin{pmatrix} 2 & -1 & -2 & -1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 0 \\ -2 \\ -2 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 14 & 0 \\ 0 & 10 \end{pmatrix}^{-1} \begin{pmatrix} 14 \\ -10 \end{pmatrix} = \begin{pmatrix} \frac{1}{14} & 0 \\ 0 & \frac{1}{10} \end{pmatrix} \begin{pmatrix} 14 \\ -10 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Prof. Dr. Thomas A. Runkler

Nonlinear Substitution

- features can be substituted by specific functions of features
- example polynomial regression:
  multiple linear regression with features

$$x, x^2, \ldots, x^q$$

  yields polynomial coefficients $a_0, a_1, \ldots, a_p$ for

$$y \approx f(x) = \sum_{i=0}^{p} a_i x^i$$

Robust Regression

- goal: reduce sensitivity to inliers and outliers
- approach: replace square error functional
- Huber function

$$E_H = \sum_{k=1}^{n} \begin{cases} e_k^2 & \text{if } |e_k| < \varepsilon \\ 2\varepsilon \cdot |e_k| - \varepsilon^2 & \text{otherwise} \end{cases}$$

- least trimmed squares

only take the m smallest error

$$E_{LTS} = \sum_{k=1}^{m} e_k'^2$$

where

$$e_1' \leq e_2' \leq \ldots \leq e_n'$$

Prof. Dr. Thomas A. Runkler

(non-linear regression)

Universal Approximator

- continuous real-valued function f on a compact subset $U \subset R^n$

(n-d)   (1-d)

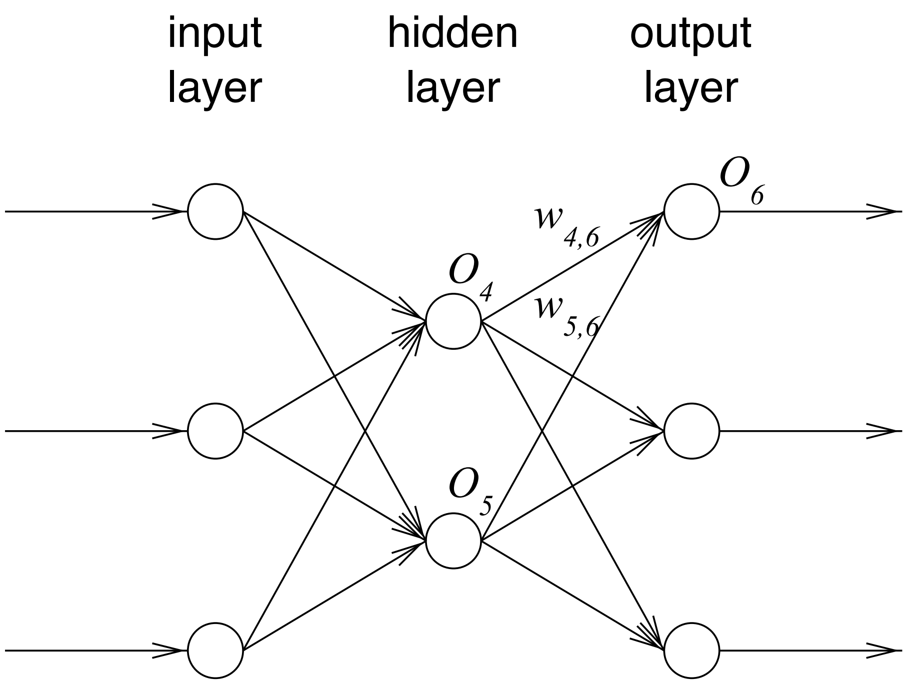
$$f : U \to R$$

- class F is universal approximator $\quad \Leftrightarrow \quad \forall \varepsilon > 0 \quad \exists f^* \in F$

$$\underbrace{|f(x) - f^*(x)|}_{\text{absolute error}} < \varepsilon \quad \forall x \in U$$

Multi Layer Perceptron (Neuron Networks)

- multi layer perceptron: directed graph
- nodes are called neurons
- $O_i \in \mathbb{R}$: output of neuron i
- $w_{ij} \in \mathbb{R}$: weight of edge from neuron i to neuron j



Multi Layer Perceptron

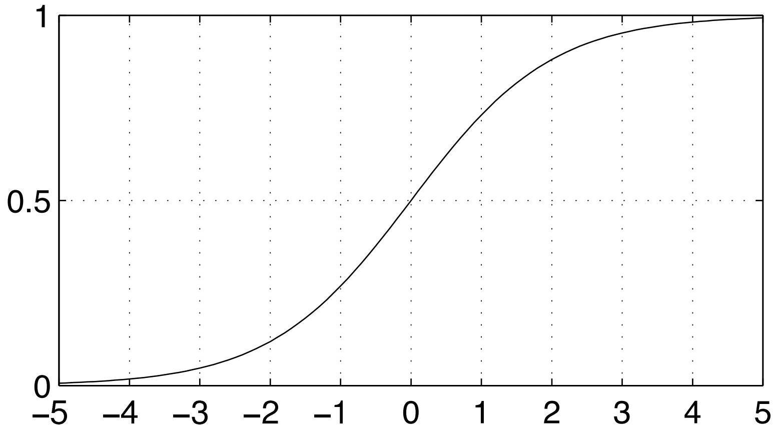
- computation of neuron output

Outputs of the previous layer

$$O_i = s(I_i), \quad I_i = \sum_j w_{ji} O_j$$

- example sigmoid function (logistic function, see Fisher–Z)

$$s(x) = \frac{1}{1 + e^{-x}} \in (0, 1)$$



Prof. Dr. Thomas A. Runkler

Generalized Delta Rule

- input layer $p \geq 1$ neurons
- hidden layer $h \geq 1$ neurons
- output layer $q \geq 1$ neurons

topology (layers + neurons)

- training data input $O = (I_1, \ldots, I_p)^T$
- output values $O = (O_{p+h+1}, \ldots, O_{p+h+q})^T$
- training data output $O' = (O'_{p+h+1}, \ldots, O'_{p+h+q})^T$
- average quadratic output error

$$E = \frac{1}{q} \cdot \sum_{i=p+h+1}^{p+h+q} (O_i - O'_i)^2$$

- weight adaptation by gradient descent

$$\Delta w_{ij} = -\alpha(t) \cdot \frac{\partial E}{\partial w_{ij}}$$

Generalized Delta Rule

- output node:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial O_j} \cdot \frac{\partial O_j}{\partial I_j} \cdot \frac{\partial I_j}{\partial w_{ij}}$$
$$\sim \underbrace{(O_j - O'_j) \cdot s'(I_j)}_{=\delta_j^{(O)}} \cdot O_i$$

$$\Rightarrow \overset{\text{gradient}}{\Delta w_{ij}} = -\alpha(t) \cdot \delta_j^{(O)} \cdot O_i$$

- for sigmoid function

$$s'(I_j) = \frac{\partial}{\partial I_j} \frac{1}{1 + e^{-I_j}} = -\frac{1}{\left(1 + e^{-I_j}\right)^2} \cdot \left(-e^{-I_j}\right)$$
$$= \frac{1}{1 + e^{-I_j}} \cdot \frac{e^{-I_j}}{1 + e^{-I_j}} = O_j \cdot (1 - O_j)$$

Generalized Delta Rule

- hidden nodes: sum of all output gradients

$$
\begin{aligned}
\frac{\partial E}{\partial w_{ij}} &= \sum_{l=p+h+1}^{p+h+q} \frac{\partial E}{\partial O_l} \cdot \frac{\partial O_l}{\partial I_l} \cdot \frac{\partial I_l}{\partial O_j} \cdot \frac{\partial O_j}{\partial I_j} \cdot \frac{\partial I_j}{\partial w_{ij}} \\
&\sim \sum_{l=p+h+1}^{p+h+q} (O_l - O'_l) \cdot s'(I_l) \cdot w_{jl} \cdot s'(I_j) \cdot O_i \\
&= \sum_{l=p+h+1}^{p+h+q} \delta_l^{(O)} \cdot w_{jl} \cdot s'(I_j) \cdot O_i \\
&= \underbrace{s'(I_j) \cdot \sum_{l=p+h+1}^{p+h+q} \delta_l^{(O)} \cdot w_{jl}}_{=\delta_j^{(H)}} \cdot O_i
\end{aligned}
$$

$$\Rightarrow \Delta w_{ij} = -\alpha(t) \cdot \delta_j^{(H)} \cdot O_i$$

Backpropagation Algorithm

1. input: neuron numbers $p, h, q \in \{1, 2, \ldots\}$, learning rate $\alpha(t)$, training data $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$, $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^q$

2. initialize weights w_{ij} and biases b_j for $i = 1, \ldots, p$, $j = p+1, \ldots, p+h$ and for $i = p+1, \ldots, p+h$, $j = p+h+1, \ldots, p+h+q$

3. for each input-output vector pair (x_k, y_k), $k = 1, \ldots, n$

   (a) update the weights and biases of the output layer

   $$w_{ij} = w_{ij} - \alpha(t) \cdot \delta_j^{(O)} \cdot O_i, \qquad \begin{array}{l} i = p+1, \ldots, p+h \\ j = p+h+1, \ldots, p+h+q \end{array}$$

   $$b_j = b_j - \alpha(t) \cdot \delta_j^{(O)}, \qquad j = p+h+1, \ldots, p+h+q$$

   (b) update the weights and biases of the hidden layer

   $$w_{ij} = w_{ij} - \alpha(t) \cdot \delta_j^{(H)} \cdot O_i, \qquad \begin{array}{l} i = 1, \ldots, p \\ j = p+1, \ldots, p+h \end{array}$$

   $$b_j = b_j - \alpha(t) \cdot \delta_j^{(H)}, \qquad j = p+1, \ldots, p+h$$

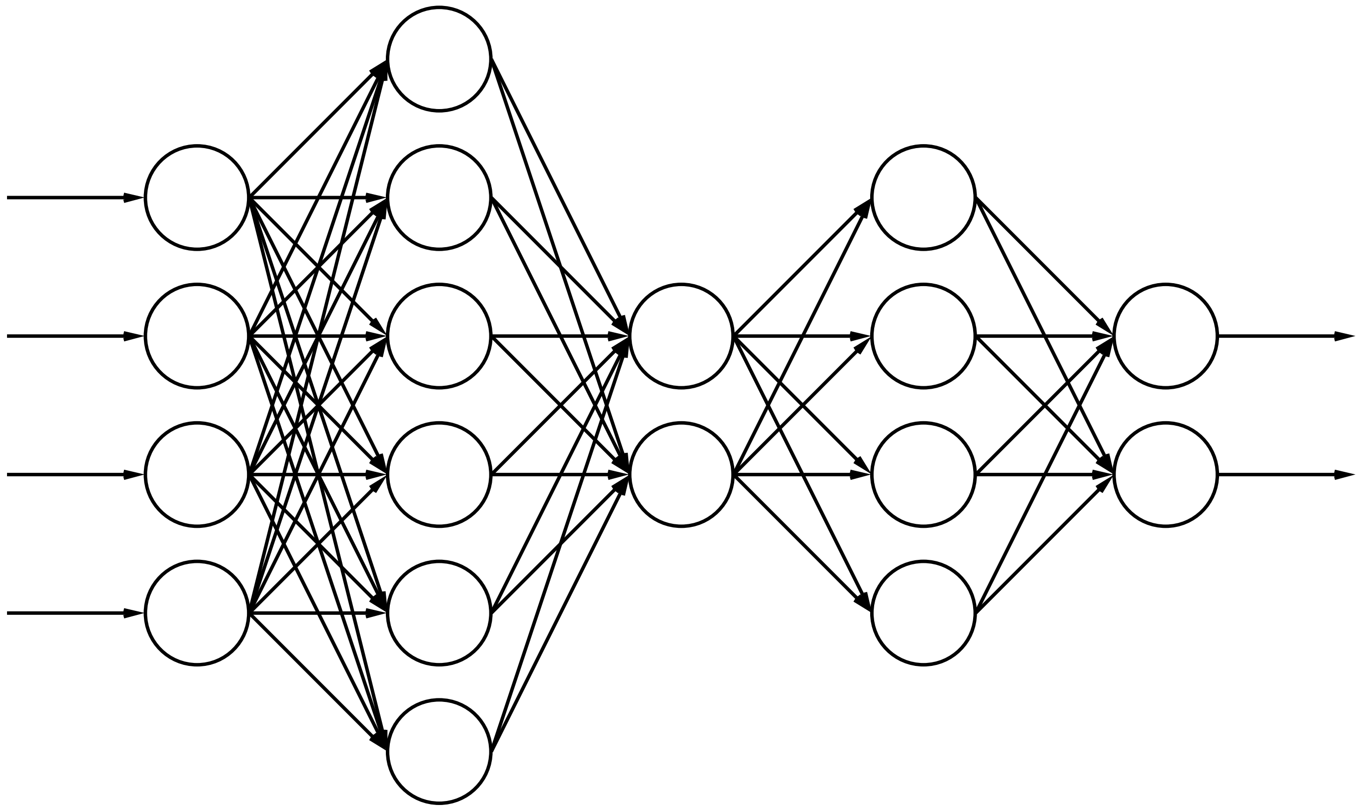
4. repeat from (3.) until termination criterion holds

5. output: w_{ij}, b_j

Prof. Dr. Thomas A. Runkler

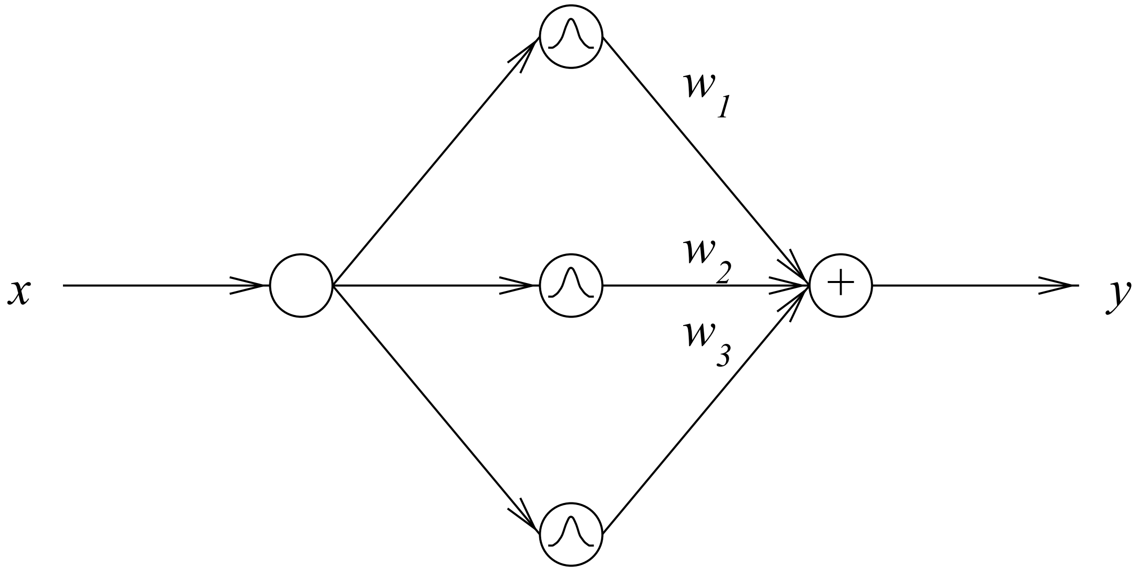
Deep Neural Network

Ex. neuron networks with 5 layers



Prof. Dr. Thomas A. Runkler

Radial Basis Functions (Powell '85)



output component $y \in \mathrm{IR}$ is computed by superposition of c radial basis functions (RBF) of the input $x \in \mathrm{IR}^p$

$$y = \sum_{i=1}^{c} w_i \cdot e^{-\left(\frac{\|x-\mu_i\|}{\sigma_i}\right)^2}$$

RBF Training

- training of RBF network using
  - input data $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ and
  - output data $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}$
- training of the centers $\mu_i \in \mathbb{R}^p$ and variances $\sigma_i > 0$
  - clustering methods (c–means, self organizing map)
  - gradient descent (*)
  - competitive learning
- training of the weights $w_i \in \mathbb{R}$
  - pseudo inverse (*)

RBF Training: Gradient Descent

- error function

$$E = \frac{1}{n} \sum_{k=1}^{n} \left(\sum_{i=1}^{c} w_i \, e^{-\left(\frac{\|x_k - \mu_i\|}{\sigma_i}\right)^2} - y_k \right)^2$$

$$\frac{\partial E}{\partial \mu_i} = \frac{4 w_i}{n \sigma_i^2} \sum_{k=1}^{n} \left(\sum_{j=1}^{m} w_j \, e^{-\left(\frac{\|x_k - \mu_j\|}{\sigma_j}\right)^2} - y_k \right) \|x_k - \mu_i\| \, e^{-\left(\frac{\|x_k - \mu_i\|}{\sigma_i}\right)^2}$$

$$\frac{\partial E}{\partial \sigma_i} = \frac{4 w_i}{n \sigma_i^3} \sum_{k=1}^{n} \left(\sum_{j=1}^{m} w_j \, e^{-\left(\frac{\|x_k - \mu_j\|}{\sigma_j}\right)^2} - y_k \right) \|x_k - \mu_i\|^2 \, e^{-\left(\frac{\|x_k - \mu_i\|}{\sigma_i}\right)^2}$$

- gradient descent

$$\Delta \mu_i = -\alpha(t) \cdot \frac{\partial E}{\partial \mu_i}$$

$$\Delta \sigma_i = -\alpha(t) \cdot \frac{\partial E}{\partial \sigma_i}$$

RBF Training: Pseudo Inverse

- training of the weights $w_i \in \mathrm{I\!R}$, $i = 1, \ldots, c$ using input data $X = \{x_1, \ldots, x_n\} \subset \mathrm{I\!R}^p$ and output data $Y = \{y_1, \ldots, y_n\} \subset \mathrm{I\!R}$
- hidden layer outputs

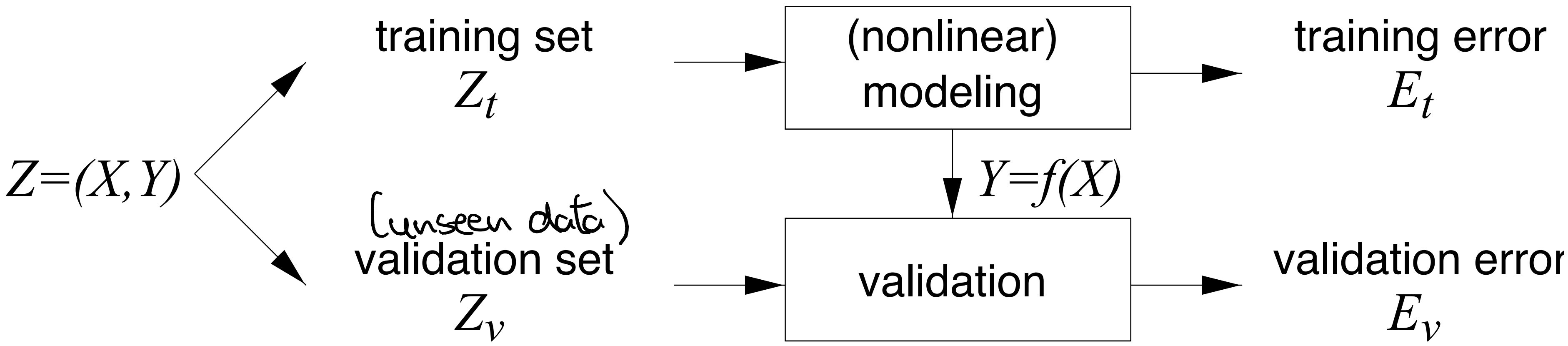
$$U = \begin{pmatrix} e^{-\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2} & \ldots & e^{-\left(\frac{x_1-\mu_c}{\sigma_c}\right)^2} \\ \vdots & & \vdots \\ e^{-\left(\frac{x_n-\mu_1}{\sigma_1}\right)^2} & \ldots & e^{-\left(\frac{x_n-\mu_c}{\sigma_c}\right)^2} \end{pmatrix}$$

- determine parameters by pseudo inverse ($Y = (y_1 \ldots y_n)^T$, $W = (w_1 \ldots w_c)^T$)

$$Y = U \cdot W \quad \Rightarrow \quad W = (U^T \cdot U)^{-1} \cdot U^T \cdot Y$$

Prof. Dr. Thomas A. Runkler

(Nonlinear) Modeling



Prof. Dr. Thomas A. Runkler

Cross Validation

- k-fold cross-validation
  - randomly partition Z into k pairwise disjoint and (almost) equally sized subsets $Z_1, \ldots, Z_k$
  - for each subset Z_i train with the remaining $k-1$ subsets Z_j and compute validation error on Z_i

$$E_{vi} = \frac{1}{|Z_i|} \sum_{(x,y)\in Z_i} \|y - f_i(x)\|^2$$

  - compute k-fold cross-validation error

$$E_v = \frac{1}{k} \sum_{i=1}^{k} E_{vi}$$

- Leave one out $=$ n-fold cross-validation (only one single data vector is retained for validation)

Training and Validation Error

- number of free parameters of the regression model: d
- plausibility $E_v \approx E_t$ for $d \to 0$ or $n \to \infty$
- estimates:

$$
\begin{aligned}
E_v &\approx \frac{1 + d/n}{1 - d/n}\, E_t \\
E_v &\approx (1 + 2d/n)\, E_t \\
E_v &\approx \frac{1}{(1 - d/n)^2}\, E_t
\end{aligned}
$$

Prof. Dr. Thomas A. Runkler