Neuronale Netze

Übung

Schätzung der Anzahl Fische in einem Teich

In einem Teich befinden sich N (unbekannt) Fische. Wir schätzen sie wie folgt: Wir fangen M Fische, markieren sie, und lassen sie wieder frei. Dann warten wir etwas und fangen dann n Fische und zählen, wie viele davon markiert sind (m). Jetzt schätzen wir $\boldsymbol{\hat{N} = M*n / m}$.

- Zeigen Sie: der Schätzer ist ML
- Der Schätzer ist nicht erwartungstreu.

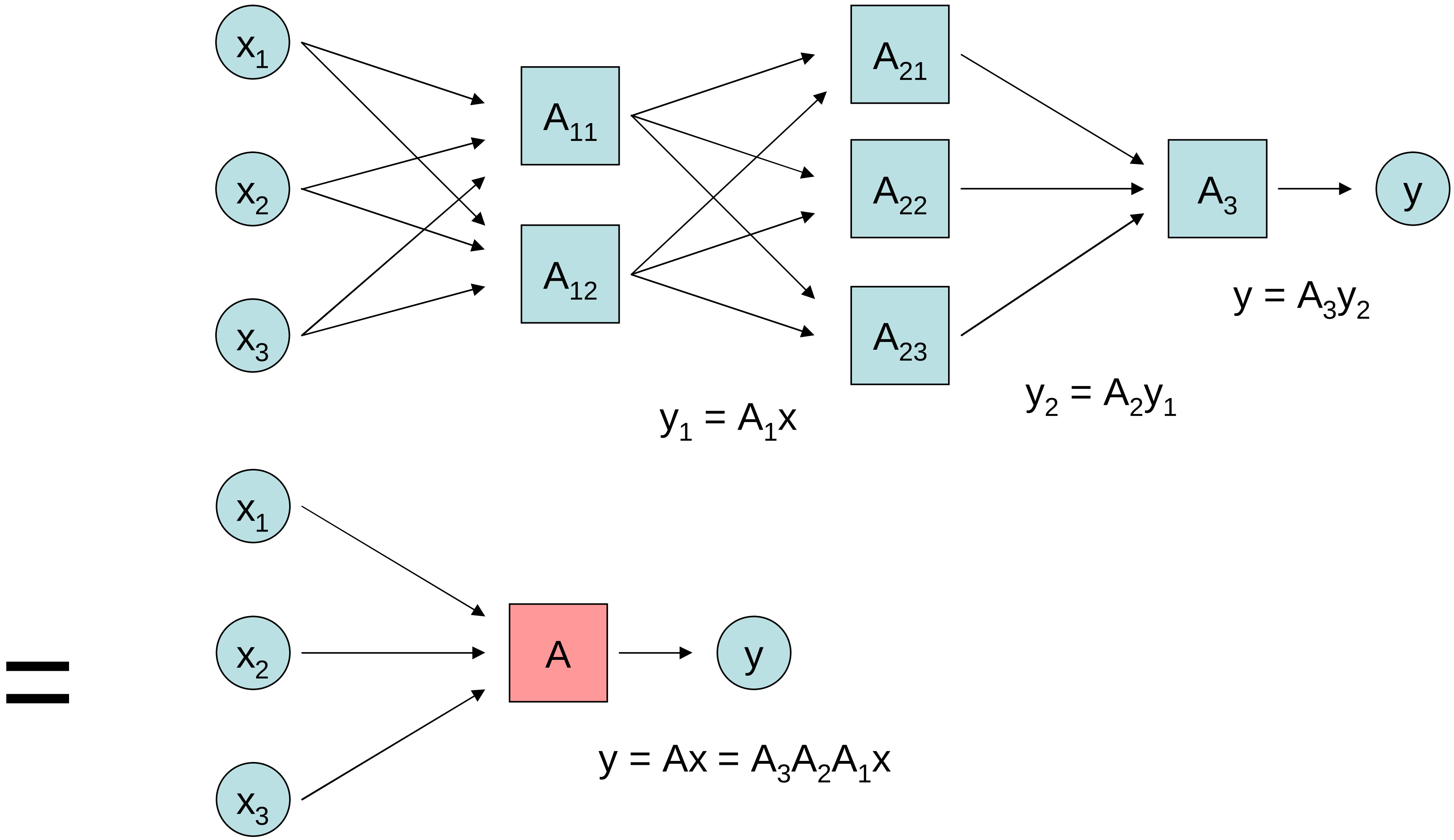
Lösung

- $\hat{N} = M*n / m$
- $\mathbf{E}\,\hat{N} = \infty \quad \Rightarrow$ nicht erwartungstreu
- $P(m|N) = (M/N)^m\,(1- M/N)^{n-m} = \max!$
- Setze $p=M/N$, $q=m/n$
- Dann $p^q\,(1-p)^{1-q} = \max$
- $q \log p + (1-q) \log (1-p) = \max$
- Ableitung nach p: $q/p - (1-q)/(1-p) = 0$
- $\Rightarrow p=q \Rightarrow \hat{N}$ ist ML

Lineare Regression

- Modell: Y = bX
- ML-Schätzer für b:
- $(\mathbf{Y}-b\mathbf{X})^T(\mathbf{Y}-b\mathbf{X}) = \min!$
- Ableitung nach b (Gradient):
- $0 = 2\mathbf{X}^T\mathbf{X}b-2\mathbf{X}^T\mathbf{Y}$
- $\underline{b = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\mathbf{Y}}$

Lineares Neuronales Netz



Übung

- Gegeben sind folgende Trainingsdaten
- Aufgabe: trainiere ein entsprechendes Neuronales Netz

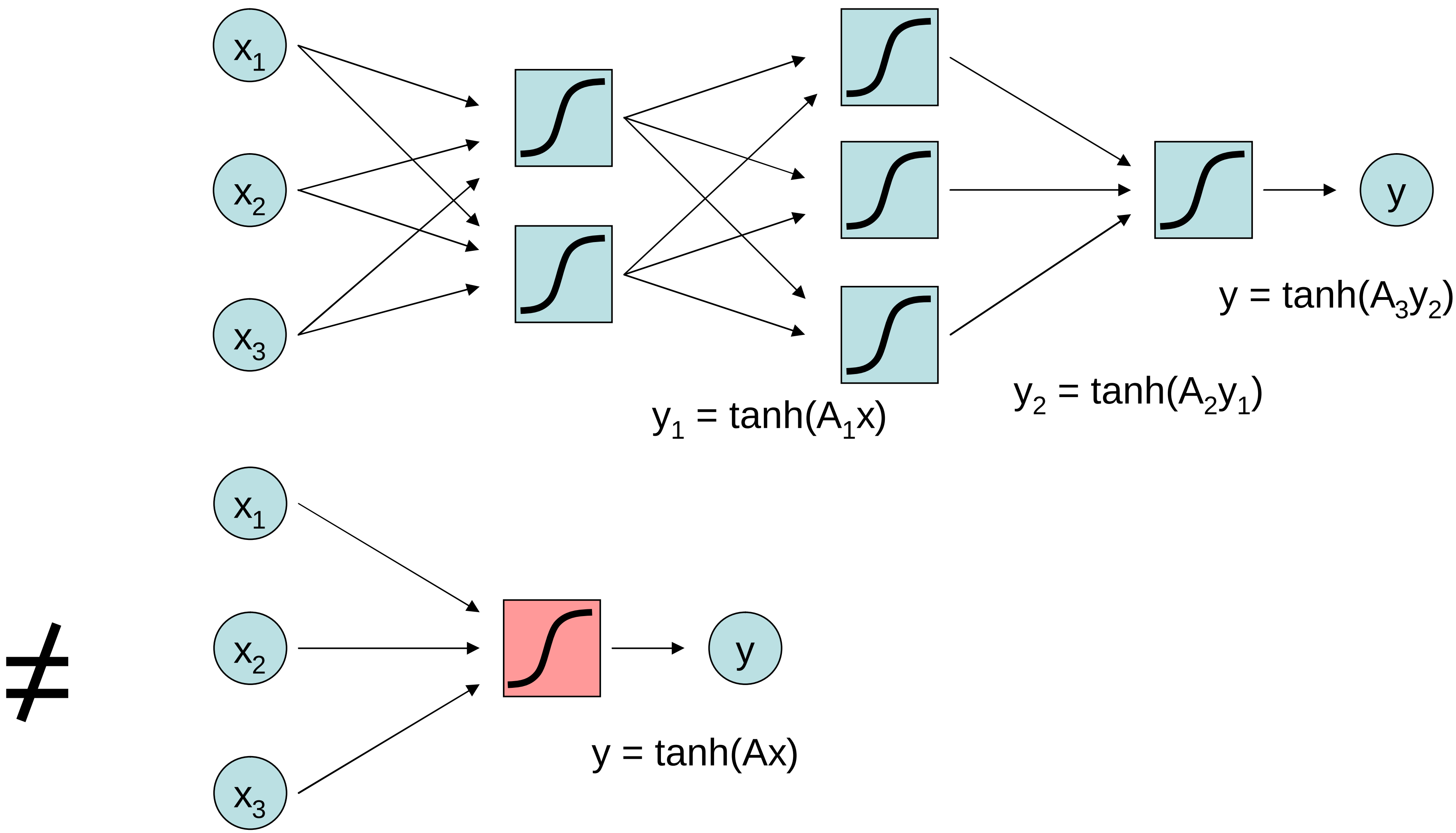
X_1	X_2	Y
0.9	3.8	-1.1
1.9	2.4	1.8
3.1	3.6	3.6
0	1.8	-0.7
1.8	3.2	0.9
3.2	2.5	5.3
3.0	3.7	3.7

Lösung

- X \ Y = [2.1176; -0.7737]
- [X ones(7,1)] \ Y = [2.1137; -1.0349; 0.8331]

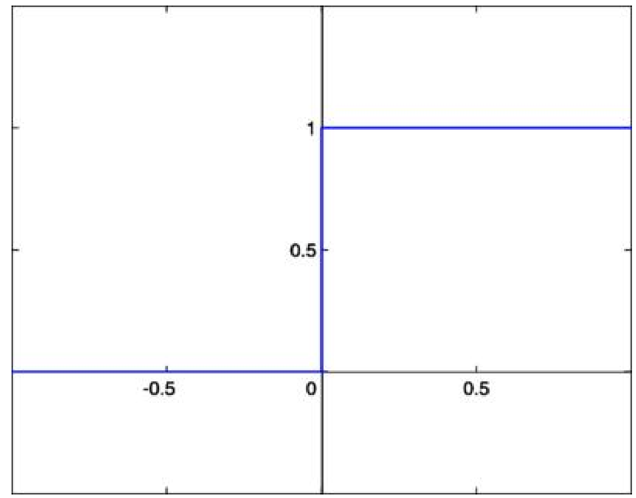
- Wahre Koeffizienten = [2; -1; 1]

Nichtlineares Neuronales Netz



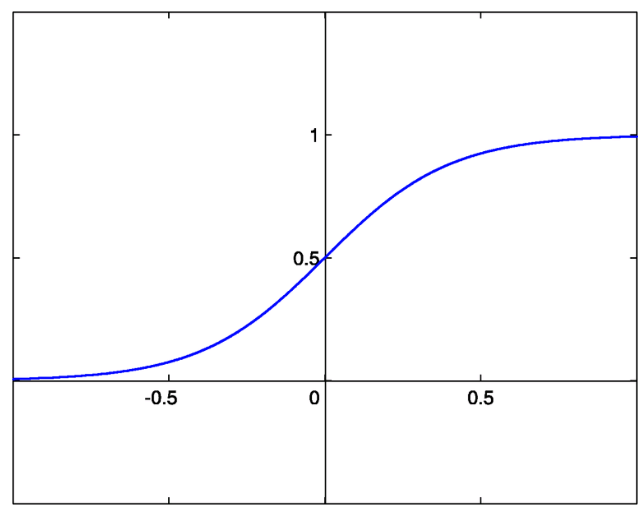
Übliche Aktivierungsfunktionen

- Linear
- Stufenfunktion



- Logistisch / TanH

$$\sigma(t) = \frac{1}{1 + e^{-\beta t}}$$



Approximationseigenschaften

Definition 2: 1. A subset S of a metric space (X, d) is *d-dense* in a subset T if for every $\epsilon > 0$ and $t \in T$ there is an $s \in S$ such that $d(s, t) < \epsilon$.

2. Let $M(R^n)$ be the set of all n-variate real-valued functions and let $C(R^n) \subseteq M(R^n)$ be the set of all *continuous* real-valued functions. A subset $F \subseteq M(R^n)$ is said to be *uniformally dense on compacta* in $C(R^n)$ or *fundamental* if for every compact subset $K \subseteq R^n$, F is *d-dense* in $C(K)$ where d is the uniform distance metric, as follows:

$$f, g \in C(K) \quad d(f, g) = \sup_{x \in K} |f(x) - g(x)| \tag{2}$$

Approximationseigenschaften 2

Theorem 1: Let f be a measurable function. $span\{f_{\mathbf{w},\theta}(x) = f(\mathbf{w} \cdot \mathbf{x} + \theta) | \mathbf{w} \in R^n, \theta \in R\}$ is fundamental in $C(R^n)$ if and only if f is not a polynomial.

Quelle:

MULTILAYER FEEDFORWARD NETWORKS
WITH NON-POLYNOMIAL ACTIVATION
FUNCTIONS CAN APPROXIMATE ANY FUNCTION

by

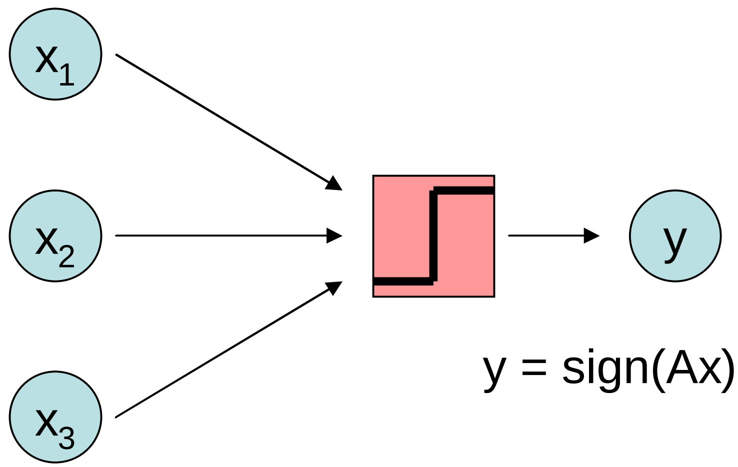
Moshe Leshno
Faculty of Management
Tel Aviv University
Tel Aviv, Israel 69978

and

Shimon Schocken
Leonard N. Stern School of Business
New York University
New York, NY 10003

September 1991

Das Perzeptron



[Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386-408]

Lernverfahren

- Online Lernen
  - Hebbsche Lernregel (1949)
    $\Delta w_{ij} = \eta \, a_i \, o_j$
    η = Lernrate
    a_i = Aktivierung des aktuellen Neurons i
    o_j = Ausgabe von Neuron j
  - Backpropagation (1974)
- Batch Lernen
  - Batch Backprop
  - 2nd Order Verfahren, z.B. Levenberg-Marquardt

Lernen als Optimierung

- Netzausgabe: $y = f(x, \theta)$
- Trainingsdaten: $(X,Y) = (x_i,y_i)_{i=1\ldots n}$
- Fehlerfunktion

$$\sum_{i=1\ldots n} [y_i - f(x_i, \theta)]^2 = \text{min!}$$

- $\Rightarrow$ viele Optimierungsalgorithmen verwendbar!

Backpropagation

- Nur ein Optimierungsverfahren
- Genauer gesagt: Gradientenverfahren mit fester Lernrate
- Andere Verfahren sind überlegen: CG, Quasi Newton (BFGS), Levenberg-Marquardt, etc.
- Levenberg-Marquardt: ersetze

$$\min_{x \in \mathbb{R}^m} \|F(x)\|_2^2$$ durch

$$\min_{x \in \mathbb{R}^m} \|F(x_k) + J(x_k)(x - x_k)\|_2^2$$

mit Trust Region (Schritt ist exakt lösbar)

Übung: Handschriftenerkennung

Datensatz

`http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`

Aktivierungsfunktion: tanh (sonst gäb‘s Probleme: welche?)

Lernverfahren: nach Wahl (EA, Hebb, o. Toolbox)

Achtung: Skalierung!

Achtung: Codierung!

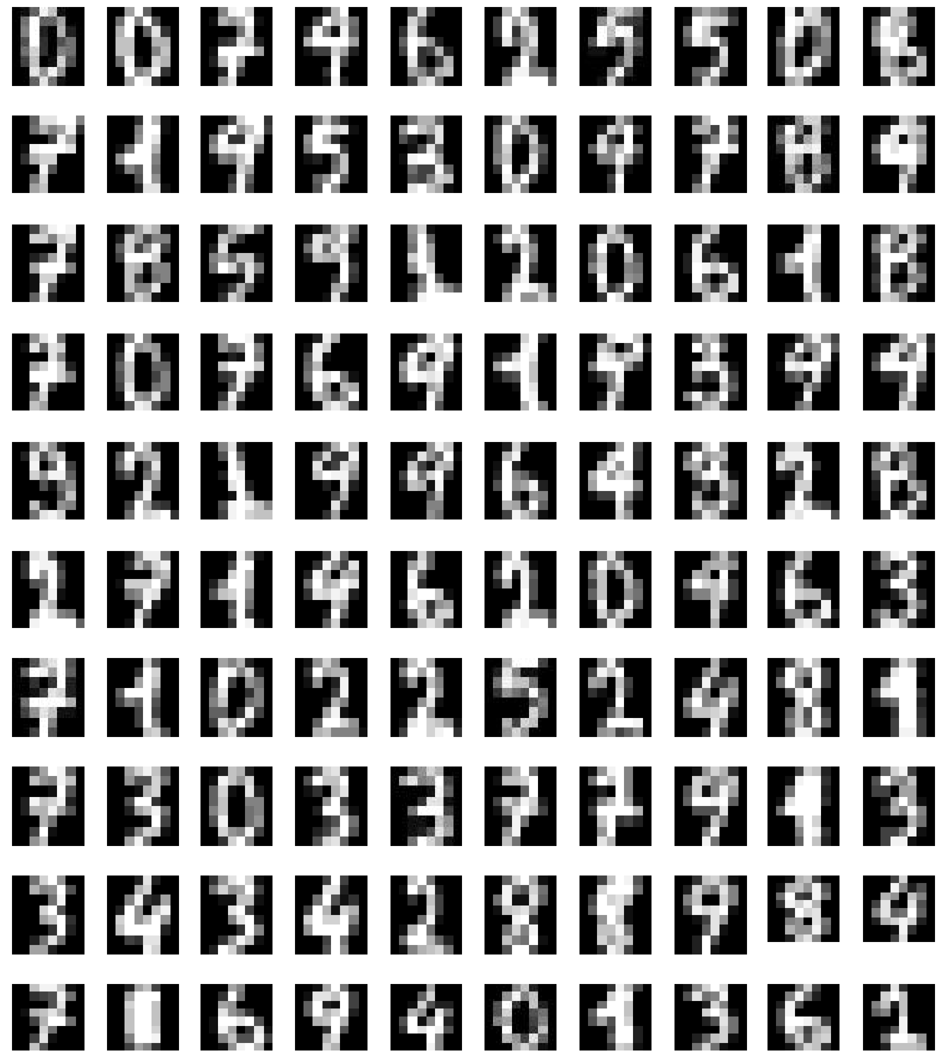
Trainingsmenge: bitte aufteilen in Training/Validierung (2:1)

Early Stopping verwenden und MSEs aufzeichnen

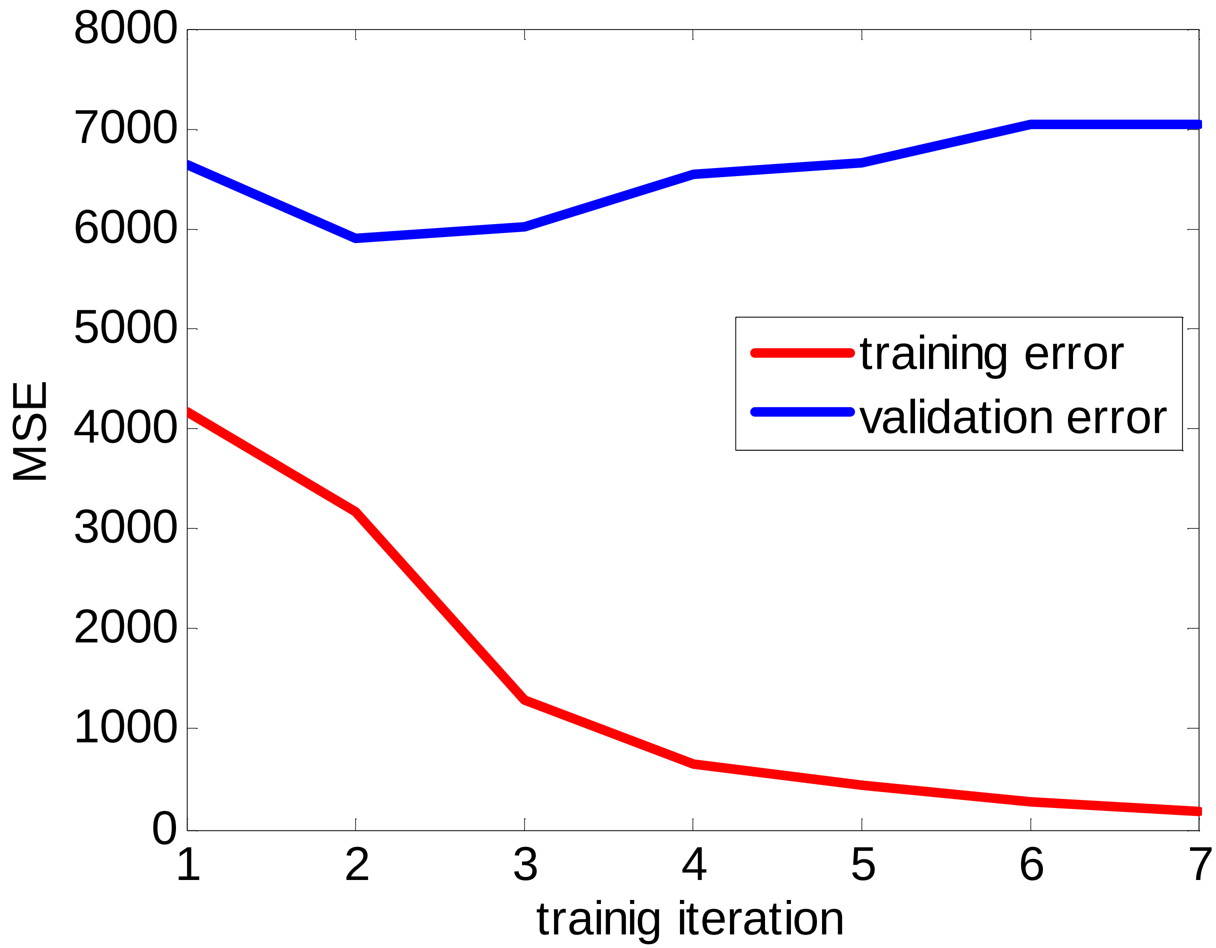
Auswertungskriterium: korrekte Erkennung der Testmenge

Die Daten (erste 100 Beispiele)

0	0	7	4	6	2	5	5	0	8
7	1	9	5	3	0	4	7	8	4
7	8	5	9	1	2	0	6	1	8
7	0	7	6	9	1	9	3	9	4
9	2	1	9	9	6	4	3	2	8
2	7	1	4	6	2	0	4	6	3
7	1	0	2	2	5	2	4	8	1
7	3	0	3	3	7	7	9	1	3
3	4	3	4	2	8	8	9	8	4
7	1	6	9	4	0	1	3	6	2



Training mit Early Stopping



2000 training samples, 1823 validation samples, Levenberg-Marquardt
Network architecture: 20 hidden tanh units, 10 tanh output units

Beispielergebnisse

weight decay	#hidden units	%correct on test data
10^{-3}	10	90
10^{-3}	5	87
1	5	86
1	10	92
1	15	94
1	20	93

Regularisierung

- Lernen als Optimierung: finde ML-Schätzer $\hat{\theta}$ in y = f(x, θ)
- ML ist manchmal ungünstig!
- $\Rightarrow$ Regularisiere θ
- Occam‘s Razor: bevorzuge einfache Modelle

Bayes Regel

- Occam‘s Razor für Wahrscheinlichkeiten:

$$P(H \mid D) = \frac{P(D \mid H) P(H)}{P(D)}$$

Prior

Posterior

- Übliches ML: maximiere P(D|H)
- Regularisiertes ML: maximiere P(D|H)P(H) (weight decay)

Weight Decay

- Annahme: die Gewichte haben einen normalverteilten Prior
- Übung: Leite die Weight Decay Kostenfunktion her und stelle sie auf!
- Lösung:

$$\exp(-a\|\theta\|^2)\,\exp(-\|Y-f(X,\theta)\|^2) = \max!$$

$$\Leftrightarrow\ a\|\theta\|^2 + \|Y-f(X,\theta)\|^2 = \min!$$

Übung

- Trainiere ein NN zur Erkennung von handgeschriebenen Ziffern mit Weight Decay