

# Neuronale Netze

Volker Tresp

# Einführung

- Der Entwurf eines guten Klassifikators/Regressionsmodells hängt entscheidend von geeigneten Basisfunktionen ab
- Manchmal sind geeignete Basisfunktionen (Merkmalsextraktoren) anwendungsspezifisch nahe liegend
- Generische Basisfunktionen wie Polynome, RBFs haben das Problem, dass die benötigte Anzahl mit der Eingangsdimension rapide ansteigt ( "Fluch der Dimension")
- Es sollte doch möglich sein, geeignete Basisfunktionen zu erlernen
- Dies ist genau die Grundidee hinter Neuronalen Netzen
- Bei Neuronalen Netzen kommen ganz spezielle Basisfunktionen zum Einsatz: sigmoide Basisfunktionen

## Scientists See Promise in Deep-Learning Programs



Hao Zhang/The New York Times

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese.

By JOHN MARKOFF

Published: November 23, 2012

Using an artificial intelligence technique inspired by theories about how the brain recognizes patterns, technology companies are reporting startling gains in fields as diverse as computer vision, speech recognition and the identification of promising new molecules for designing drugs.

[点击查看本文中文版。](#)

**Connect With Us on Social Media**


@nytimescience  
on Twitter.

• Science Reporters  
and Editors on Twitter


Like the science desk on  
Facebook.




The advances have led to widespread enthusiasm among researchers who design software to perform human activities like seeing, listening and thinking. They offer the promise of machines that converse with humans and perform tasks like driving cars and working in factories, [raising the specter of automated robots that could replace human workers.](#)


 FACEBOOK

 TWITTER

 GOOGLE+

 SAVE

 E-MAIL

 SHARE

 PRINT

 SINGLE PAGE

 REPRINTS

 THE WAY BACK  
WATCH TRAILER



Keith Penner

A student team led by the computer scientist Geoffrey E. Hinton used deep-learning technology to design software.

The technology, called deep learning, has already been put to use in services like Apple's Siri virtual personal assistant, which is based on Nuance Communications' speech recognition service, and in Google's Street View, which uses machine vision to identify specific addresses.

But what is new in recent months is the growing speed and accuracy of deep-learning programs, often called artificial neural networks or just "neural nets" for their resemblance to the neural connections in the brain.

"There has been a number of stunning new results with deep-learning methods," said Yann LeCun, a computer scientist at New York University who did pioneering research in handwriting recognition at Bell Laboratories. "The kind of jump we are seeing in the accuracy of these systems is very rare indeed."

Artificial intelligence researchers are acutely aware of the dangers of being overly optimistic. Their field has long been plagued by outbursts of misplaced enthusiasm followed by equally striking declines.

In the 1960s, some computer scientists believed that a workable artificial intelligence system was just 10 years away. In the 1980s, a wave of commercial start-ups collapsed, leading to what some people called the "A.I. winter."

But recent achievements have impressed a wide spectrum of computer experts. In October, for example, a team of graduate students studying with the University of Toronto computer scientist [Geoffrey E. Hinton](#) won the top prize in a contest sponsored by Merck to design software to help find molecules that might lead to new drugs.

From a data set describing the chemical structure of thousands of different molecules, they used deep-learning software to determine which molecule was most likely to be an effective drug agent.

The achievement was particularly impressive because the team decided to enter the contest at the last minute and designed its software with no specific knowledge about how the molecules bind to their targets. The students were also working with a relatively small set of data; neural nets typically perform well only with very large ones.

"This is a really breathtaking result because it is the first time that deep learning won, and more significantly it won on a data set that it wouldn't have been expected to win at," said Anthony Goldbloom, chief executive and founder of Kaggle, a company that organizes data science competitions, including the Merck contest.

# Scientists See Promise in Deep-Learning Programs

Published: November 23, 2012

(Page 2 of 2)

This summer, Jeff Dean, a Google technical fellow, and Andrew Y. Ng, a Stanford computer scientist, programmed a cluster of 16,000 computers to train itself to automatically recognize images in a library of 14 million pictures of 20,000 different objects. Although the accuracy rate was low — 15.8 percent — the system did 70 percent better than the most advanced previous one.

[点击查看本文中文版。](#)

**Connect With Us on Social Media**

@nytimescience  
on Twitter.

• Science Reporters  
and Editors on Twitter

Like the science desk on  
Facebook.



Deep learning was given a particularly audacious display at a conference last month in Tianjin, China, when [Richard F. Rashid](#), Microsoft's top scientist, gave a lecture in a cavernous auditorium while a computer program recognized his words and simultaneously displayed them in English on a large screen above his head.

Then, in a demonstration that led to stunned applause, he paused after each sentence and the words were translated into Mandarin Chinese characters, accompanied by a simulation of his own voice in that language, which Dr. Rashid has never spoken.

The feat was made possible, in part, by deep-learning techniques that have spurred improvements in the accuracy of speech recognition.

FACEBOOK

TWITTER

GOOGLE+

SAVE

E-MAIL

SHARE

PRINT

SINGLE PAGE

REPRINTS



Then, in a demonstration that led to stunned applause, he paused after each sentence and the words were translated into Mandarin Chinese characters, accompanied by a simulation of his own voice in that language, which Dr. Rashid has never spoken.

The feat was made possible, in part, by deep-learning techniques that have spurred improvements in the accuracy of speech recognition.

Dr. Rashid, who oversees Microsoft's worldwide research organization, acknowledged that while his company's new speech recognition software made 30 percent fewer errors than previous models, it was "still far from perfect."

"Rather than having one word in four or five incorrect, now the error rate is one word in seven or eight," he wrote on Microsoft's Web site. Still, he added that this was "the most dramatic change in accuracy" since 1979, "and as we add more data to the training we believe that we will get even better results."

One of the most striking aspects of the research led by Dr. Hinton is that it has taken place largely without the patent restrictions and bitter infighting over intellectual property that characterize high-technology fields.

"We decided early on not to make money out of this, but just to sort of spread it to infect everybody," he said. "These companies are terribly pleased with this."

Referring to the rapid deep-learning advances made possible by greater computing power, and especially the rise of graphics processors, he added:

"The point about this approach is that it scales beautifully. Basically you just need to keep making it bigger and faster, and it will get better. There's no looking back now."

---

## Baidu muscles in on Google's turf with Silicon Valley deep learning lab Chinese search giant beds down next to Apple in Cupertino

By [Phil Muncaster](#) • [Get more from this author](#)

Posted in [Business](#), 15th April 2013 06:00 GMT

[Free whitepaper – Hands on with Hyper-V 3.0 and virtual machine movement](#)

Chinese search giant Baidu has opened the doors to a new research facility in Google's back yard where it's hoping to tap the local talent to consolidate early mover advantage in the burgeoning field of "deep learning".

The Cupertino-based Institute of Deep Learning (IDL) is the Silicon Valley counterpart of another facility back in China dedicated to accelerating research in the emerging machine learning-related discipline.

## Neuronale Netze: wesentliche Vorteile

- Neuronale Netze sind universelle Approximatoren: jede stetige Funktion kann beliebig genau approximiert werden (mit genügend vielen sigmoiden Basisfunktionen)
- Entscheidender Vorteil Neuronaler Netze: mit **wenigen** Basisfunktionen einen sehr guten Fit zu erreichen bei gleichzeitiger hervorragender Generalisierungseigenschaften
- Besondere Approximationseigenschaften Neuronaler Netze in hohen Dimensionen



## Flexiblere Modelle: Neuronale Netze

- Auch bei einem Neuronalem Netz besteht der Ausgang (bzw. die Aktivierungsfunktion  $h(x)$  im Falle eines Perzeptrons) aus der linear-gewichteten Summation von Basisfunktionen

$$\hat{y}_i = f(\mathbf{x}_i) = \sum_{h=0}^{H-1} w_h \text{sig}(\mathbf{x}_i, \mathbf{v}_h)$$

- Beachte, dass neben den Ausgangsgewichten  $\mathbf{w}$  das Neuronale Netz auch innere Gewichte  $\mathbf{v}_h$  enthält

## Neuronale Basisfunktionen

- Spezielle Form der Basisfunktionen

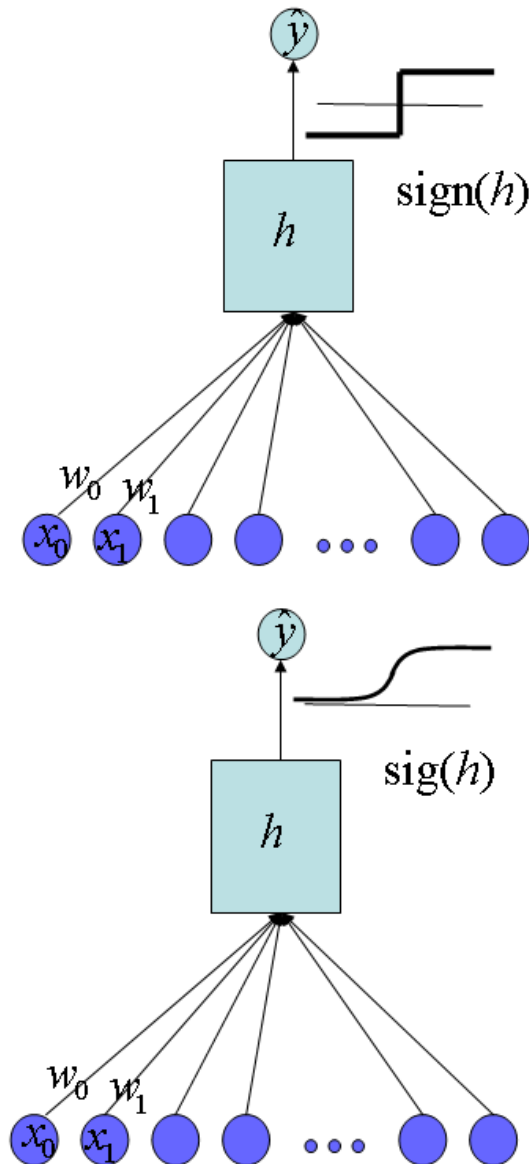
$$z_i = \text{sig}(\mathbf{x}_i, \mathbf{v}_h) = \text{sig} \left( \sum_{j=0}^{M-1} v_{h,j} x_{i,j} \right)$$

mit der *logistischen Funktion*

$$\text{sig}(in) = \frac{1}{1 + \exp(-in)}$$

- Adaption der inneren Parameter  $v_{h,j}$  der Basisfunktionen!

# Harte und weiche (sigmoide) Übertragungsfunktion



- Zunächst wird die Aktivierungsfunktion als gewichtete Summe der Eingangsgrößen  $x_i$  berechnet zu

$$h(\mathbf{x}_i) = \sum_{j=0}^{M-1} w_j x_{i,j}$$

(beachte:  $x_{i,0} = 1$  ist ein konstanter Eingang, so dass  $w_0$  dem Bias entspricht)

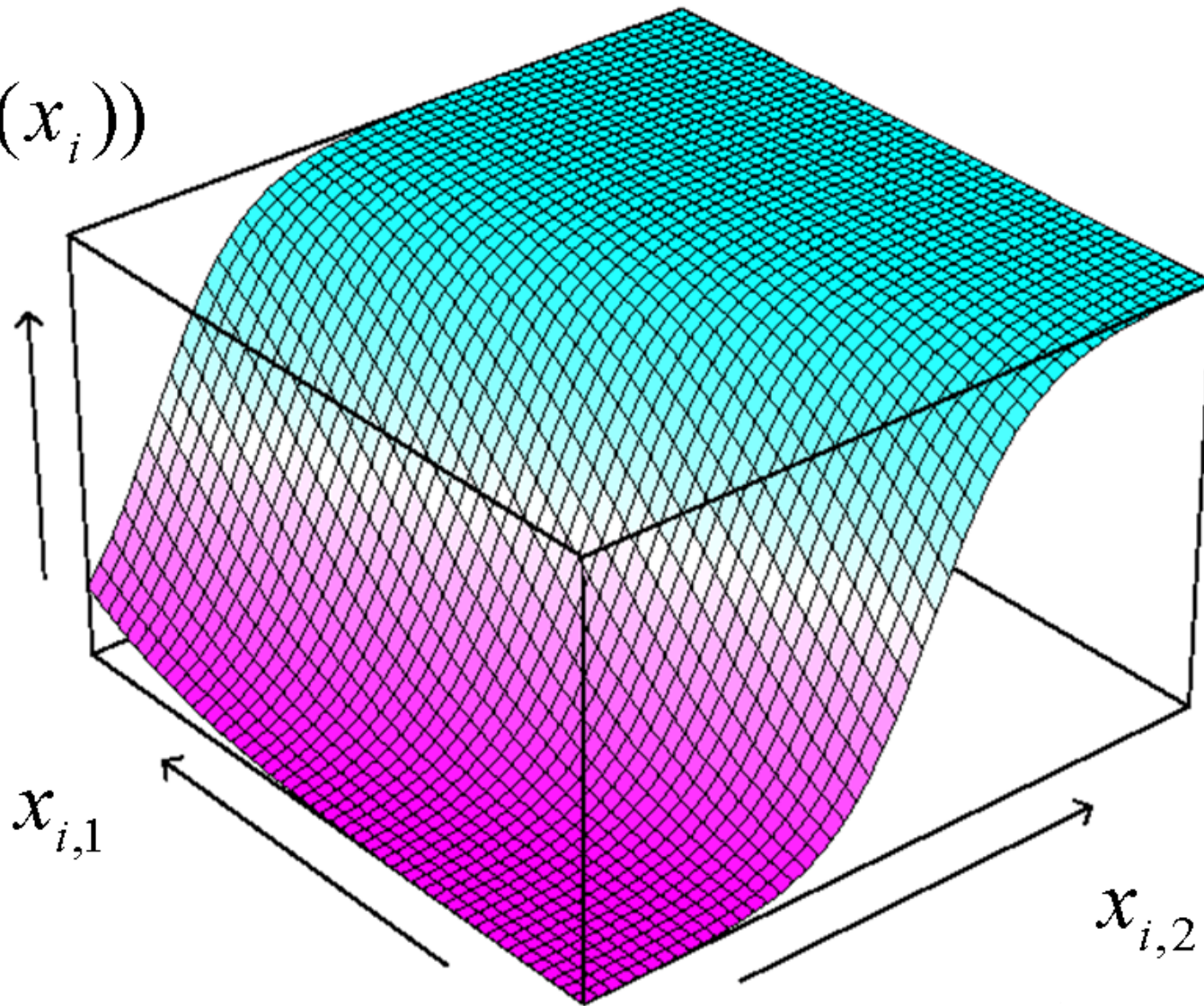
- Das sigmoide Neuron unterscheidet sich vom Perzeptron durch die Übertragungsfunktion

$$\text{Perzeptron : } \hat{y}_i = \text{sign}(h(\mathbf{x}_i))$$

$$\text{Sigmoides Neuron : } \hat{y}_i = \text{sig}(h(\mathbf{x}_i))$$

## Transferfunktion

$\text{sig}(h(x_i))$



## Charakteristische Hyperebene

- Definiere die Hyperebene

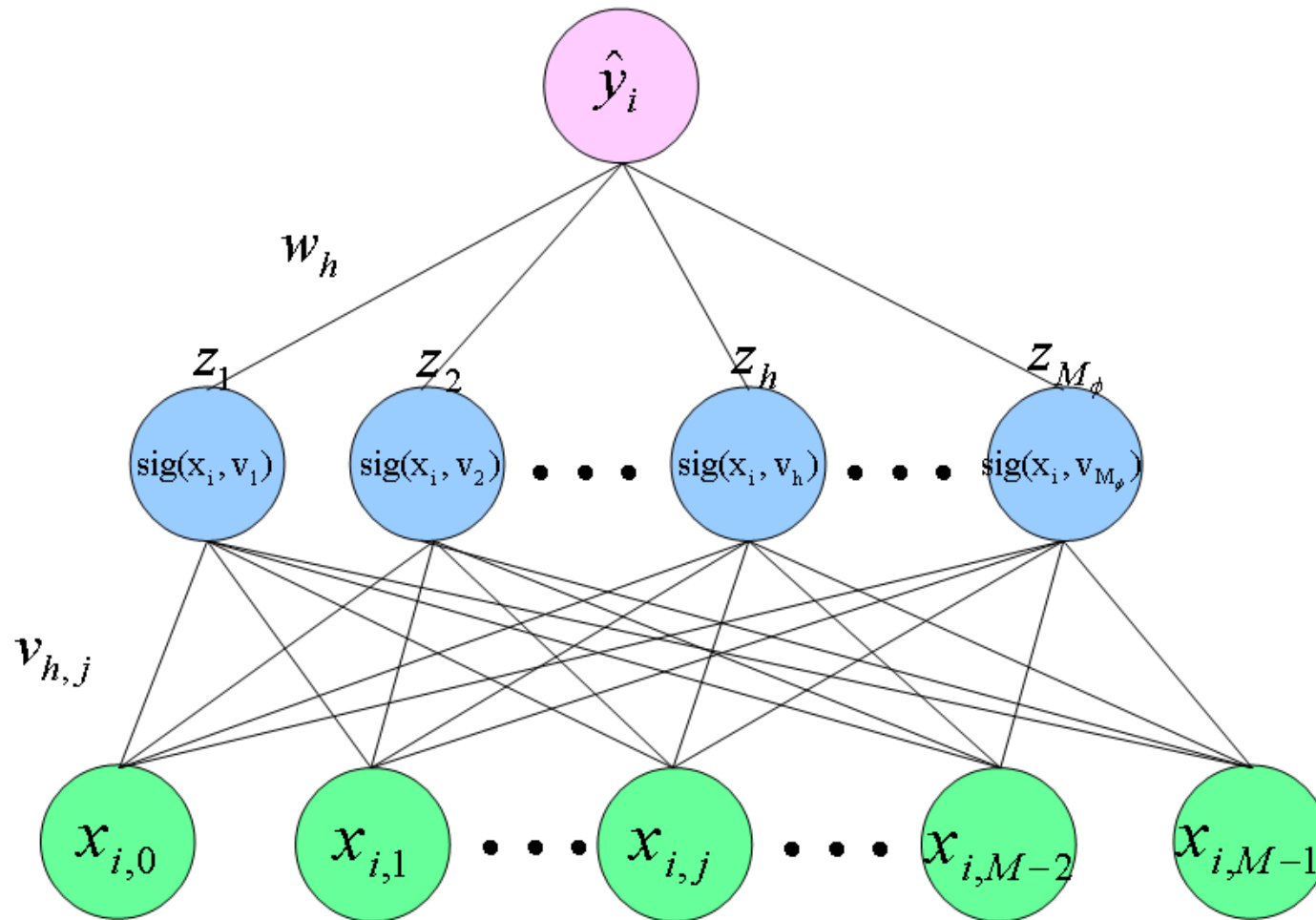
$$\text{sig} \left( \sum_{j=0}^{M-1} v_{h,j} x_{i,j} \right) = 0.5$$

Identisch mit:

$$\sum_{j=0}^{M-1} v_{h,j} x_{i,j} = 0$$

- “Teppich über einer Stufe”

# Architektur eines Neuronales Netzes



## Varianten

- Will man einen binären Klassifikator lernen, dann wendet man manchmal die sigmoide Transferfunktion auch auf das Ausgabeneuron an und berechnet als Ausgang

$$\hat{y}_i = f(\mathbf{x}_i) = \text{sig}(\mathbf{z}_i, \mathbf{w})$$

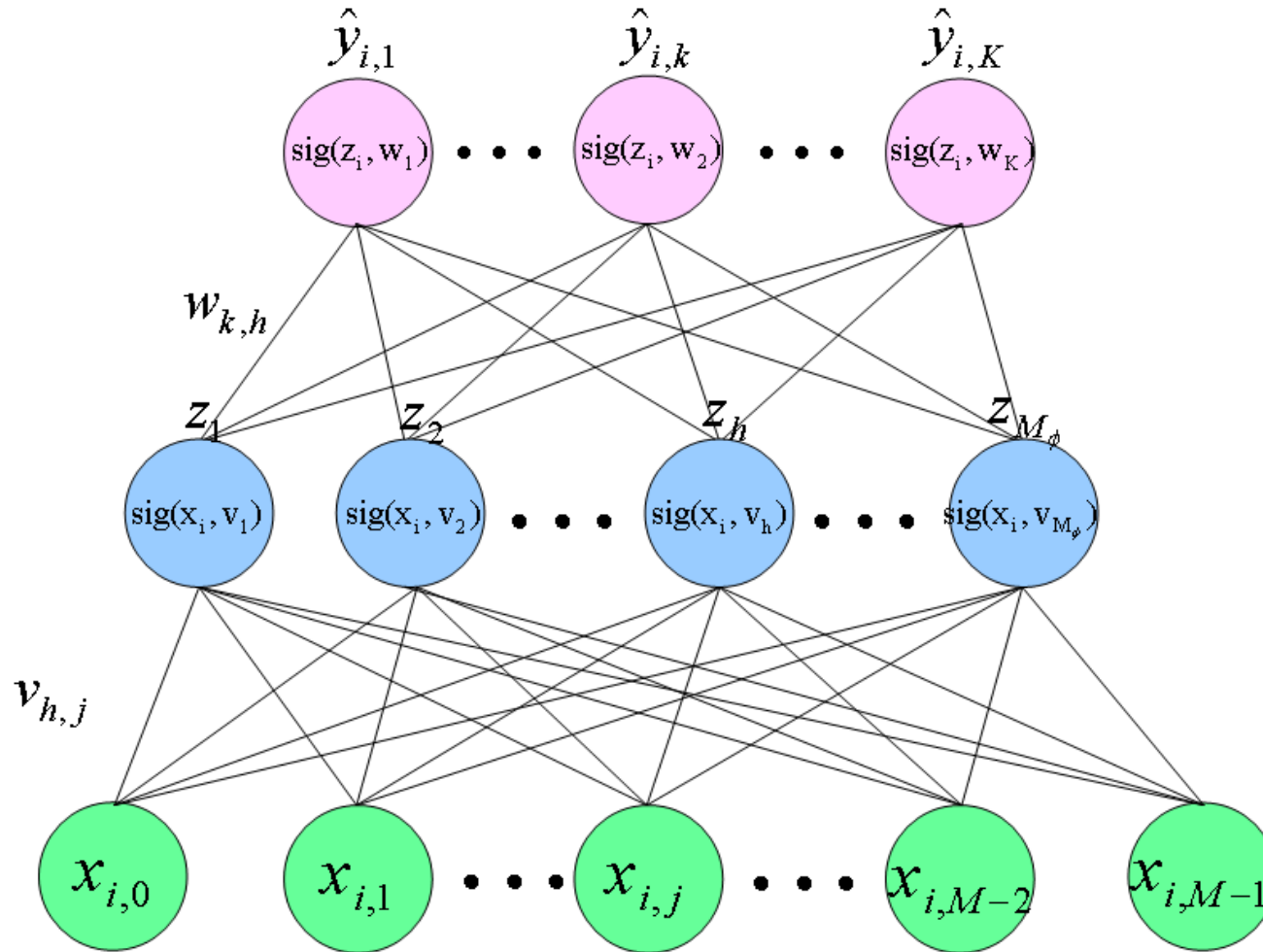
- Bei Problemen mit mehr Klassen, führt man mehrere Ausgangsneuronen ein. Zum Beispiel, um die  $K$  Ziffern zu klassifizieren

$$\hat{y}_{i,k} = f_k(\mathbf{x}_i) = \text{sig}(\mathbf{z}_i, \mathbf{w}_k) \quad k = 1, 2, \dots, K$$

und man entscheidet sich für die Klasse  $l$ , so dass  $l = \arg \max_k (\hat{y}_{i,k})$

- Das vorgestellte Neuronale Netz wird Multi Layer Perceptron (MLP) genannt

# Architektur eines Neuronales Netzes mit mehreren Ausgängen





## Lernen mit mehreren sigmoiden Ausgängen

- Ziel ist wieder die Minimierung des quadratischen Fehlers über alle Ausgänge

$$\text{cost}(\mathbf{w}, \mathbf{v}) = \sum_{i=1}^N \sum_{k=1}^K (y_{i,k} - f_k(\mathbf{x}_i, \mathbf{w}, \mathbf{v}))^2$$

- Die least-squares Lösung für  $\mathbf{v}$  kann nicht geschlossen formuliert werden
- Typischerweise werden sowohl  $\mathbf{w}$  als auch  $\mathbf{v}$  durch Gradientenabstieg optimiert

## Adaption der Ausgangsgewichte

- Für die Gradienten der Kostenfunktion nach den Ausgangsgewichten ergibt sich für Muster  $i$ :

$$\frac{\partial \text{cost}(\mathbf{x}_i, \mathbf{w}, \mathbf{v})}{\partial w_{k,h}} = -2\delta_{i,k}z_{i,h}$$

wobei

$$\delta_{i,k} = \text{sig}'(\mathbf{z}_i, \mathbf{w}_k)(y_{i,k} - f_k(\mathbf{x}_i, \mathbf{w}, \mathbf{v}))$$

das zurückpropagierte Fehlersignal ist (error backpropagation).

- Entsprechende beim ADALINE ergibt sich für den musterbasierter Gradientenabstieg:

$$w_{k,h} \leftarrow w_{k,h} + \eta \delta_{i,k} z_{i,h}$$

## Ableitung der sigmoiden Übertragungsfunktion nach dem Argument

Es lässt sich hierbei elegant schreiben

$$\text{sig}'(in) = \frac{\exp(-in)}{(1 + \exp(-in))^2} = \text{sig}(in)(1 - \text{sig}(in))$$

## Adaption der Eingangsgewichte

- Für die Gradienten der Kostenfunktion nach den Eingangsgewichten ergibt sich für Muster  $i$ :

- 

$$\frac{\partial \text{cost}(\mathbf{x}_i, \mathbf{w}, \mathbf{v})}{\partial v_{h,j}} = -2\delta_{i,h}x_{i,j}$$

wobei der zurückpropagierte Fehler ist

$$\delta_{i,h} = \text{sig}'(\mathbf{x}_i, \mathbf{v}_h) \sum_{k=1}^K w_{k,h} \delta_{i,k}$$

- Entsprechende beim ADALINE ergibt sich für den musterbasierter Gradientenabstieg:

$$v_{h,j} \leftarrow v_{h,j} + \eta \delta_{i,h} x_{i,j}$$

# Musterbasiertes Lernen

- Iteriere über alle Trainingsdaten
- sei  $\mathbf{x}_i$  der aktuelle Datenpunkt
  - $\mathbf{x}_i$  wird angelegt und man berechnet  $\mathbf{z}_i, \mathbf{y}_i$  (Vorwärtspropagation  $\leftarrow$ )
  - Durch Fehler-Rückpropagation (*error backpropagation*) werden die  $\delta_{i,h}, \delta_{i,k}$  berechnet
  - Adaptiere

$$w_{k,h} \leftarrow w_{k,h} + \eta \delta_{i,k} z_{i,h}$$

$$v_{h,j} \leftarrow v_{h,j} + \eta \delta_{i,h} x_{i,j}$$

- Alle Operationen lokal: biologisch plausibel

## Neuronales Netz und Überanpassung

- Im Vergleich zu konventionellen statistischen Modellen hat ein Neuronales Netz sehr viele freie Parameter, was zu Überanpassung führen kann
- Die beiden gebräuchlichsten Methoden damit umzugehen sind Regularisierung und Stopped-Training
- Wir diskutieren zunächst Regularisierung

## Neuronale Netze: Regularisierung

- Wir führen Regularisierungsterme ein und erhalten

$$\text{cost}^{\text{pen}}(\mathbf{w}, \mathbf{v}) = \sum_{i=1}^N \sum_{k=1}^K (y_{i,k} - f_k(\mathbf{x}_i, \mathbf{w}, \mathbf{v}))^2 + \lambda_1 \sum_{h=0}^{M_\phi-1} w_h^2 + \lambda_2 \sum_{h=0}^{H-1} \sum_{j=0}^M v_{h,j}^2$$

- Die Adaptionsregeln ändern sich zu (mit *weight decay term*)

$$w_{k,h} \leftarrow w_{k,h} + \eta (\delta_{i,k} z_{i,h} - \lambda_1 w_{k,h})$$

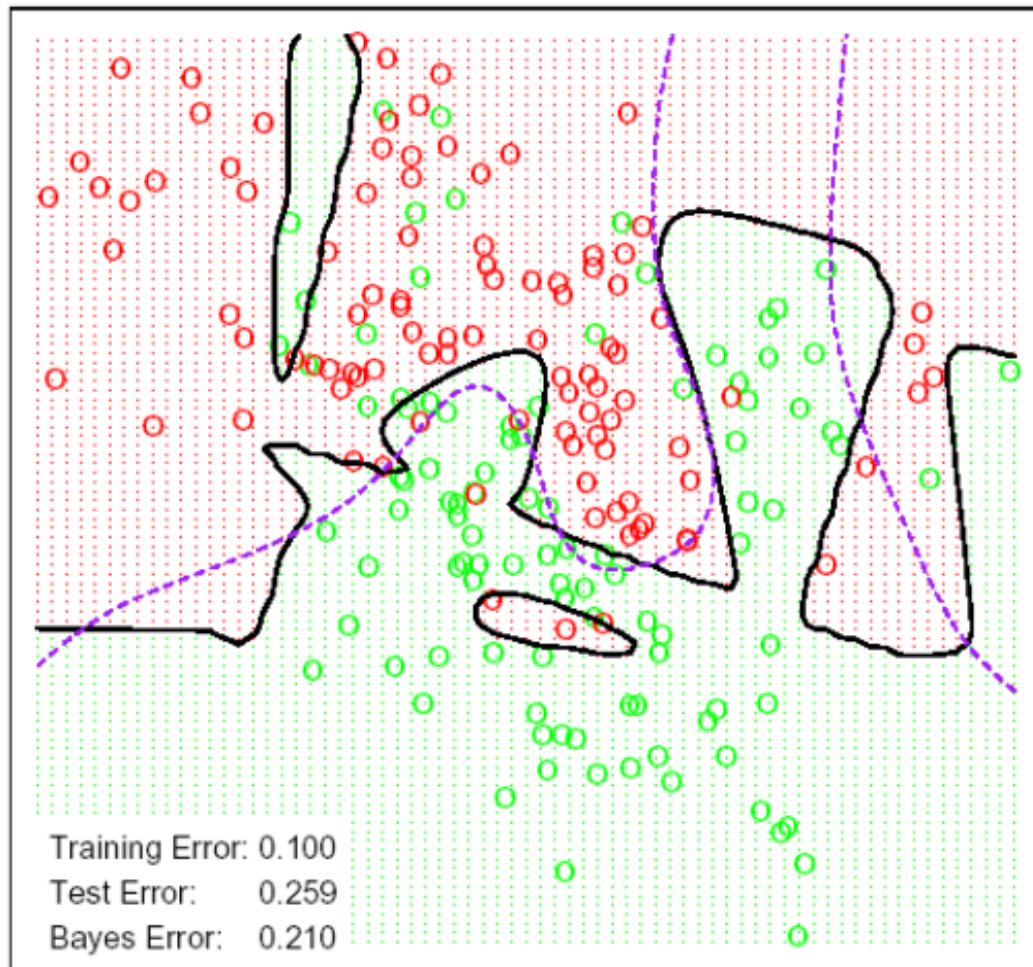
$$v_{h,j} \leftarrow v_{h,j} + \eta (\delta_{i,h} x_{i,j} - \lambda_2 v_{h,j})$$

## Künstliches Beispiel

- Daten von zwei Klassen (rote, grüne Kringel) werden generiert
- Die Klassen überlappen
- Die optimale Klassifikationsgrenze ist gestrichelt gezeigt
- Beim Neuronalen Netz ohne Regularisierung sieht man Überanpassung (stetige Kurve)



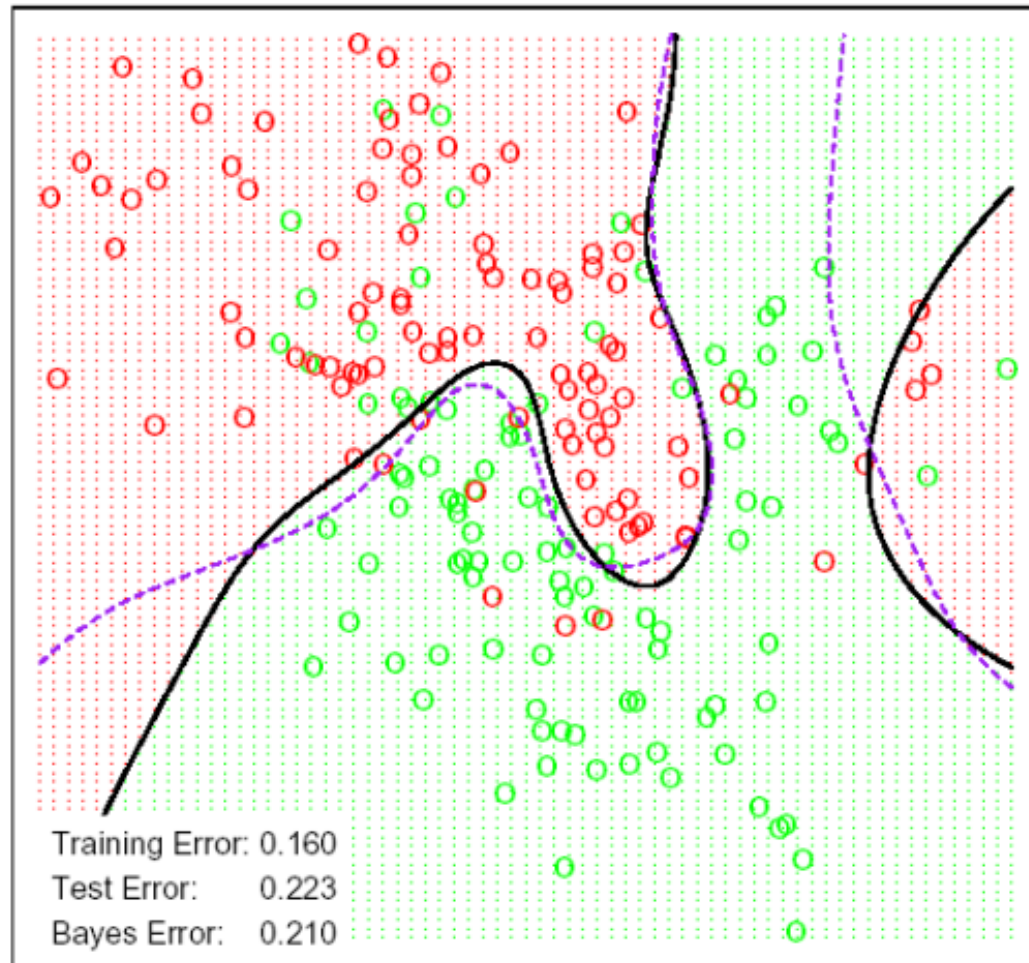
### Neural Network - 10 Units, No Weight Decay



## Künstliches Beispiel mit Regularisierung

- Mit Regularisierung ( $\lambda_1 = \lambda_2 = 0.2$ ) werden die wahren Klassengrenzen besser getroffen
- Der Trainingsfehler ist beim unregularisierten Netz kleiner, der Testfehler ist allerdings beim regularisierten Netz kleiner!

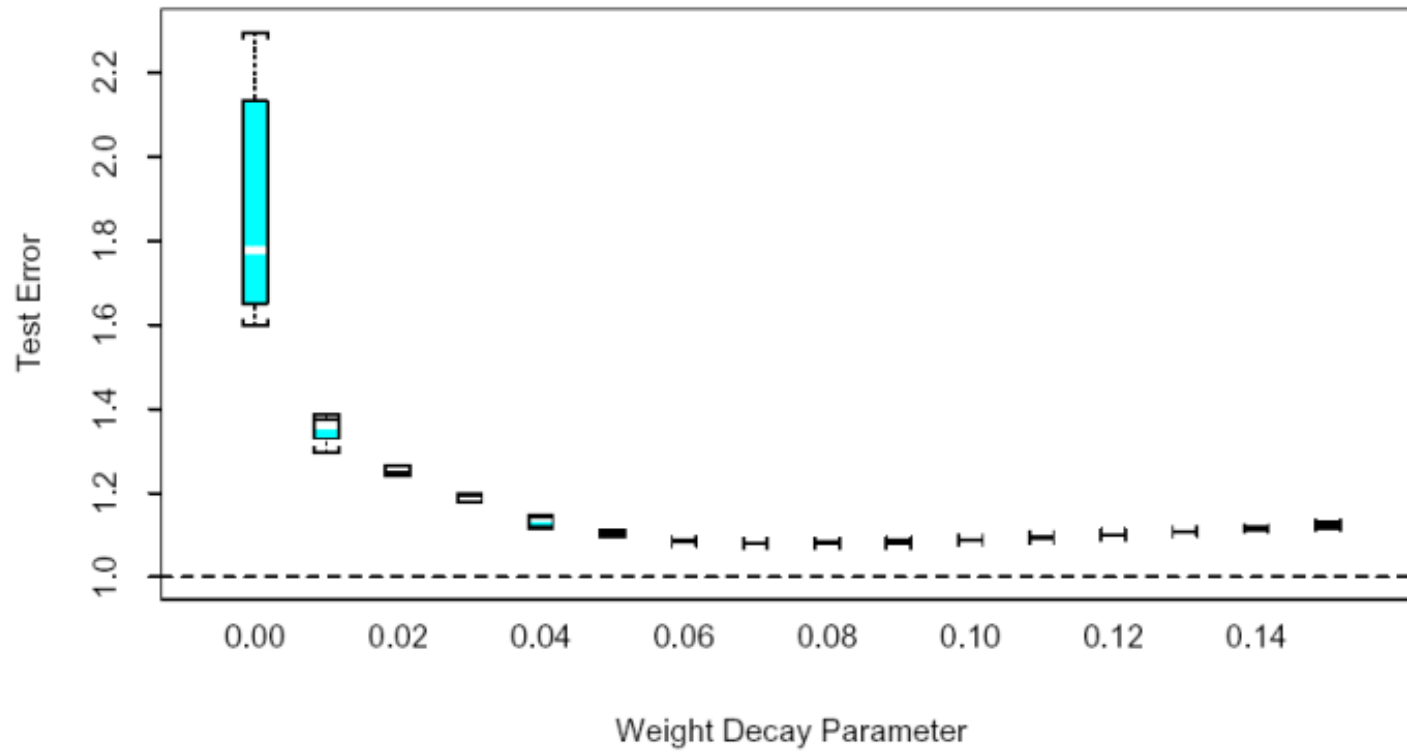
Neural Network - 10 Units, Weight Decay=0.02



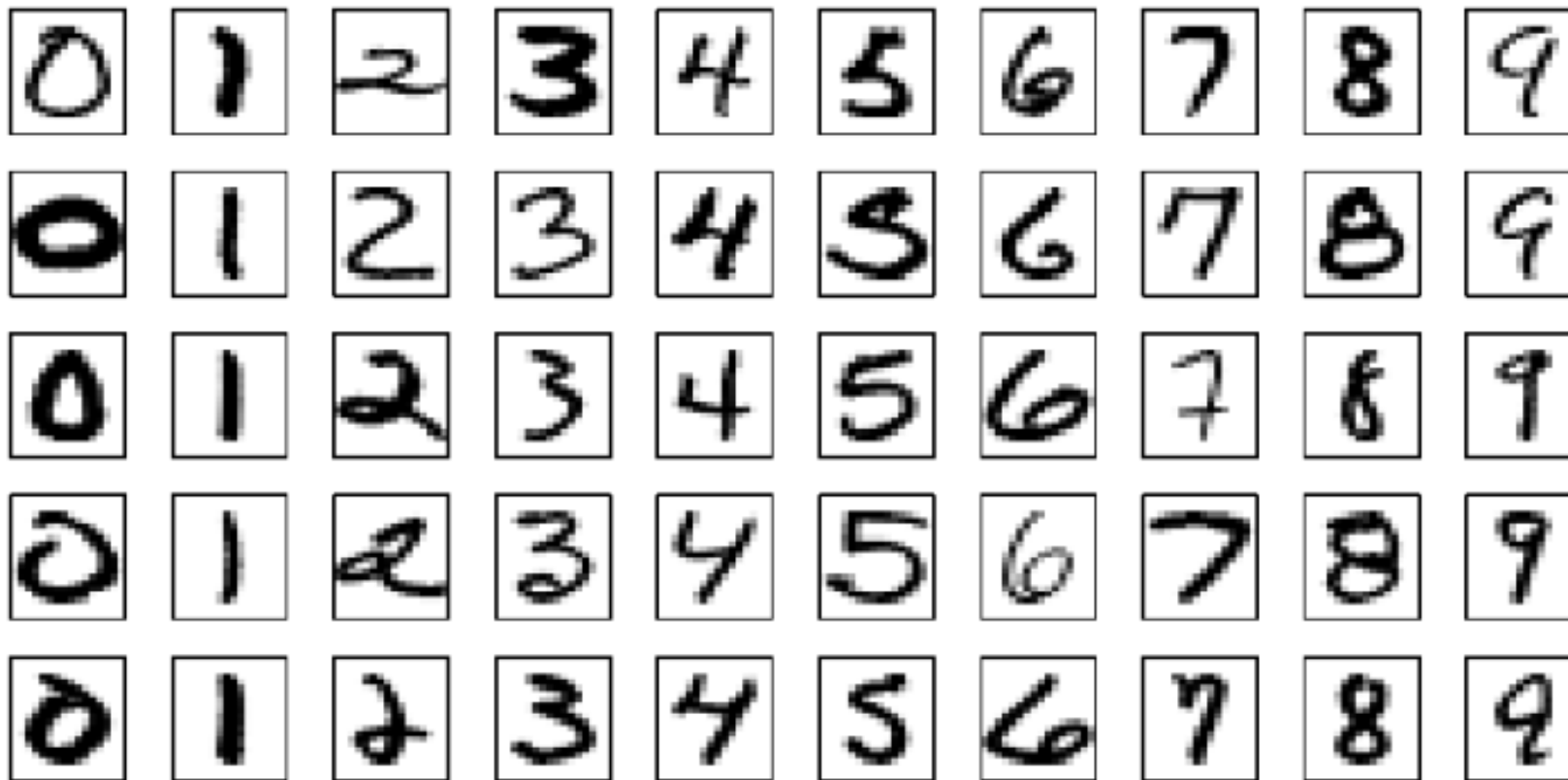
## Optimaler Regularisierungsparameter

- Der Regularisierungsparameter wird zwischen 0 und 0.15 variiert
- Die vertikale Axe zeigt den Testfehler für viele unabhängige Experimente
- Der beste Testfehler wird bei einem Regularisierungsparameter von 0.07 erzielt
- Die Variation im Testfehler wird mit größerem Regularisierungsparameter geringer

Sum of Sigmoids, 10 Hidden Unit Model

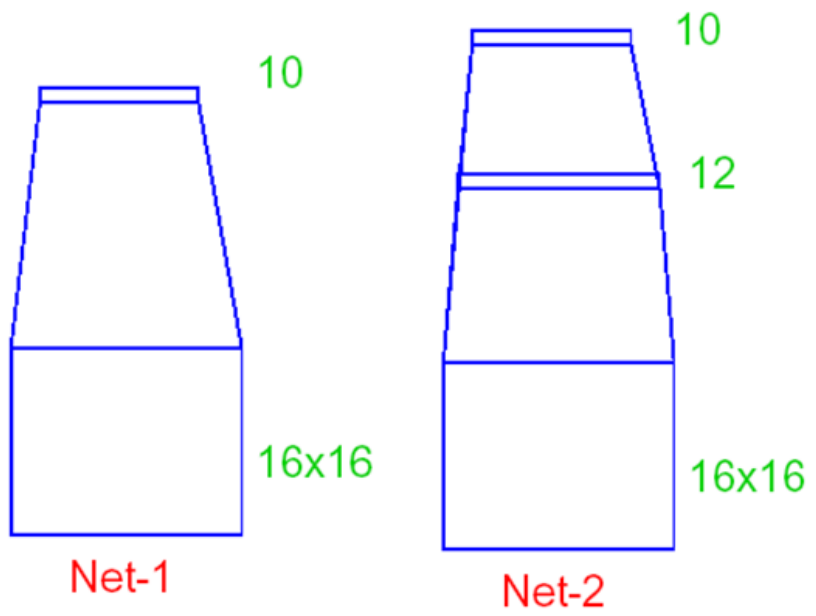


## Erkennung handgeschriebener Ziffern



## Erkennung handgeschriebener Ziffern mit Neuronalen Netzen

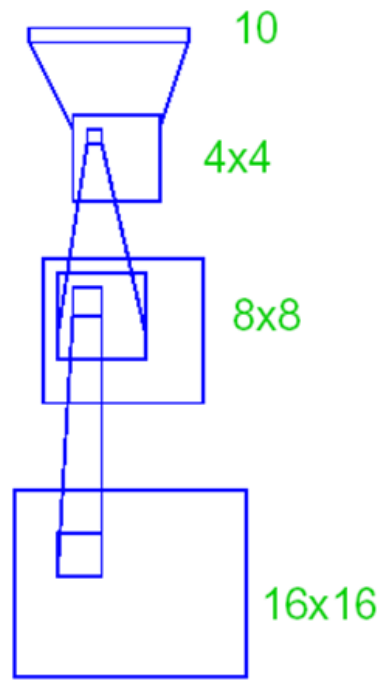
- Im Beispiel hier:  $16 \times 16$  grauwertige Bilder; 320 Ziffern im Trainingssatz, 160 Ziffern im Testsatz
- Net-1: Keine versteckte Schicht: entspricht in etwa 10 Perzeptrons, eines für jede Ziffer
- Net-2: Eine versteckte Schicht mit 12 Knoten; voll-verbunden (normales Neuronales Netz mit einer versteckten Schicht)





## Neuronale Netze mit lokaler Verbindungsstruktur: Net-3

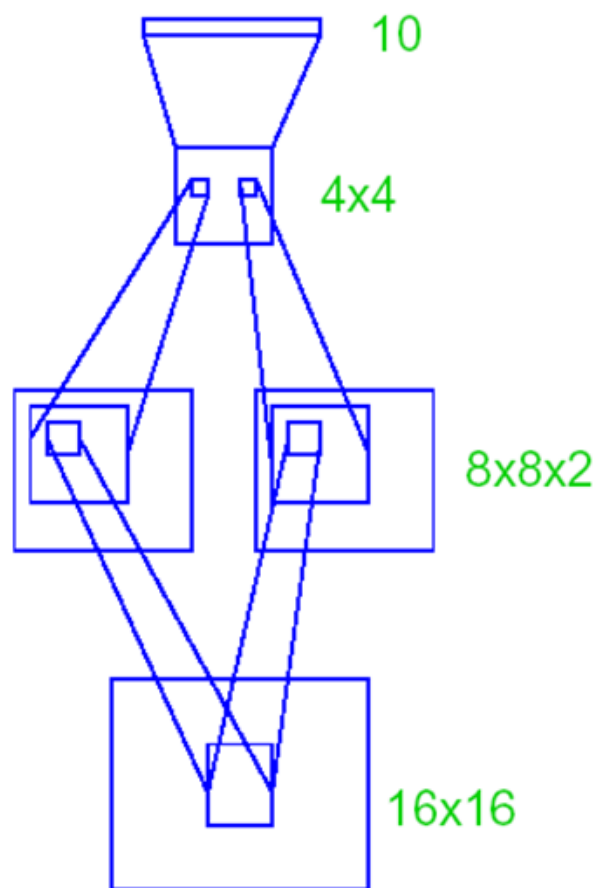
- Bei weiteren Varianten wurde die Komplexität anwendungsspezifisch eingeschränkt
- Net-3: Zwei versteckte Schichten mit lokaler Verbindungsstruktur: orientiert an den lokalen rezeptiven Feldern im Gehirn
  - Jedes der  $8 \times 8$  Neuronen in der ersten versteckten Schicht ist nur mit  $3 \times 3$  Eingangneuronen verbunden aus einem rezeptivem Feld verbunden
  - In der zweiten versteckten Schicht ist jedes der  $4 \times 4$  Neuronen mit  $5 \times 5$  Neuronen der ersten versteckten Schicht verbunden
  - Net-3 hat weniger als 50% der Gewichte als Net-2 aber mehr Neuronen



Net-3  
Local Connectivity

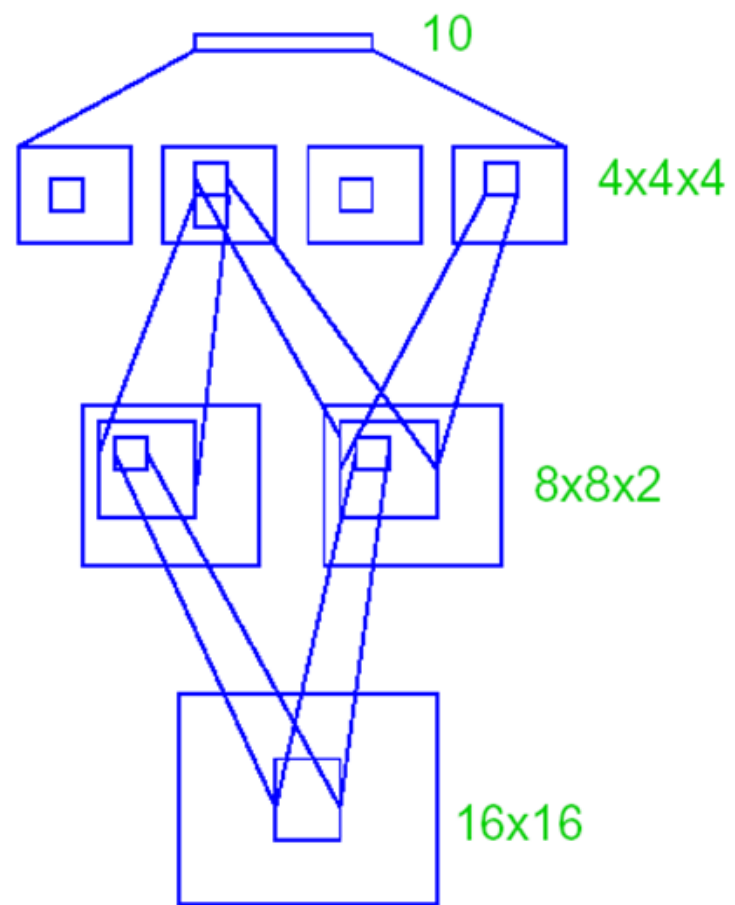
## Neuronale Netze mit Weight-Sharing (Net-4)

- Net-4: Zwei versteckte Schichten mit lokaler Verbindungsstruktur und *weight-sharing*
- Alle rezeptiven Felder im linken  $8 \times 8$  Block haben die gleichen Gewichte; ebenso alle rezeptiven Felder im rechten  $8 \times 8$  Block
- Der  $4 \times 4$  Block in der zweiten versteckten Schicht wie zuvor



## Neuronale Netze mit Weight-Sharing (Net-5)

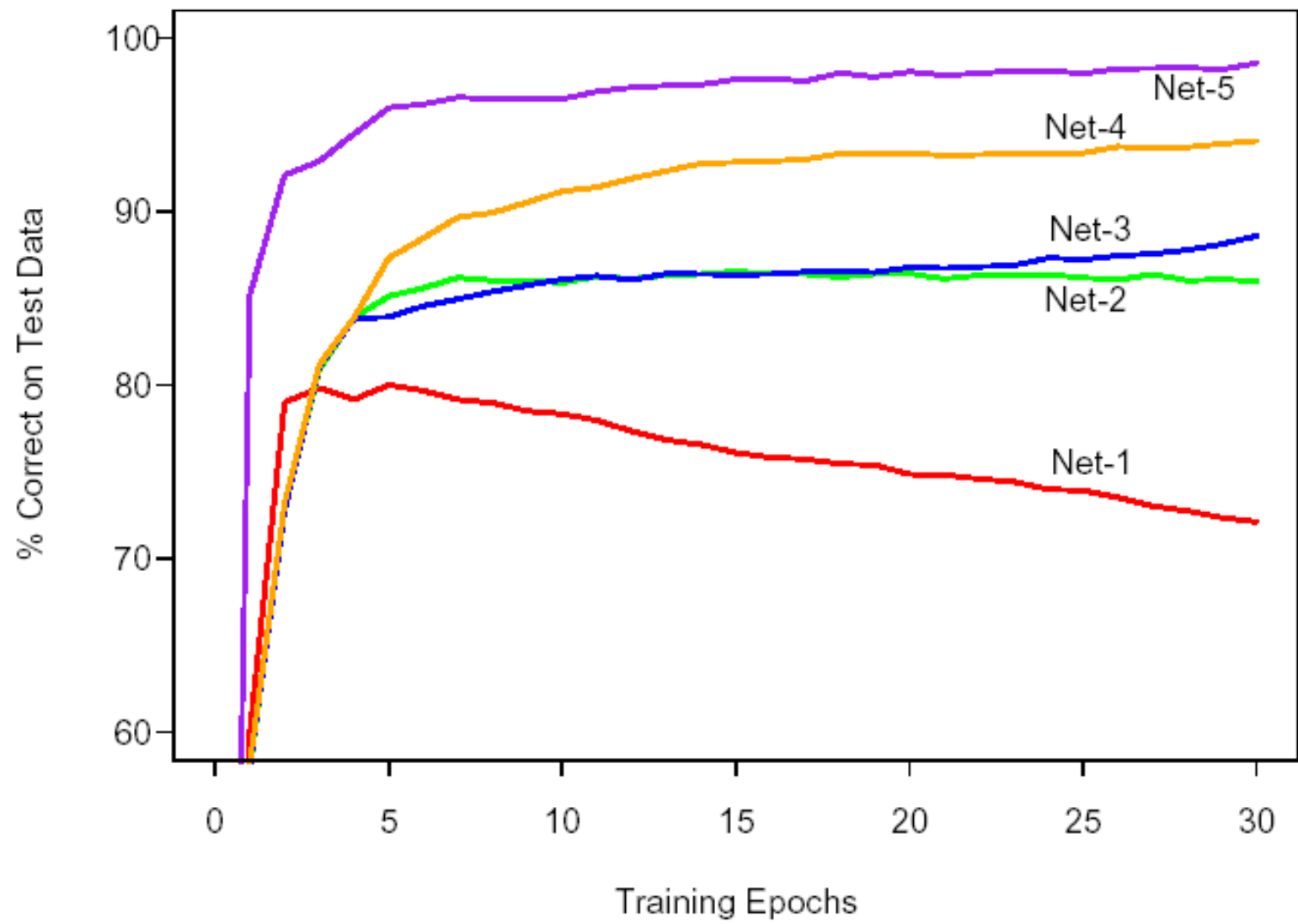
- Net-5: Zwei versteckte Schichten mit lokaler Verbindungsstruktur und zwei Ebenen von *weight-sharing*



Net-5

## Lernkurven

- Ein Trainingsepoch bezeichnet den einmaligen Durchgang durch alle Daten
- Die nächste Abbildung zeigt die Performanz auf Testdaten
- Net-1: Man sieht die Überanpassung mit zunehmenden Trainingsiterationen
- Net-5: Zeigt die besten Resultate und zeigt keine Überanpassung





## Statistiken

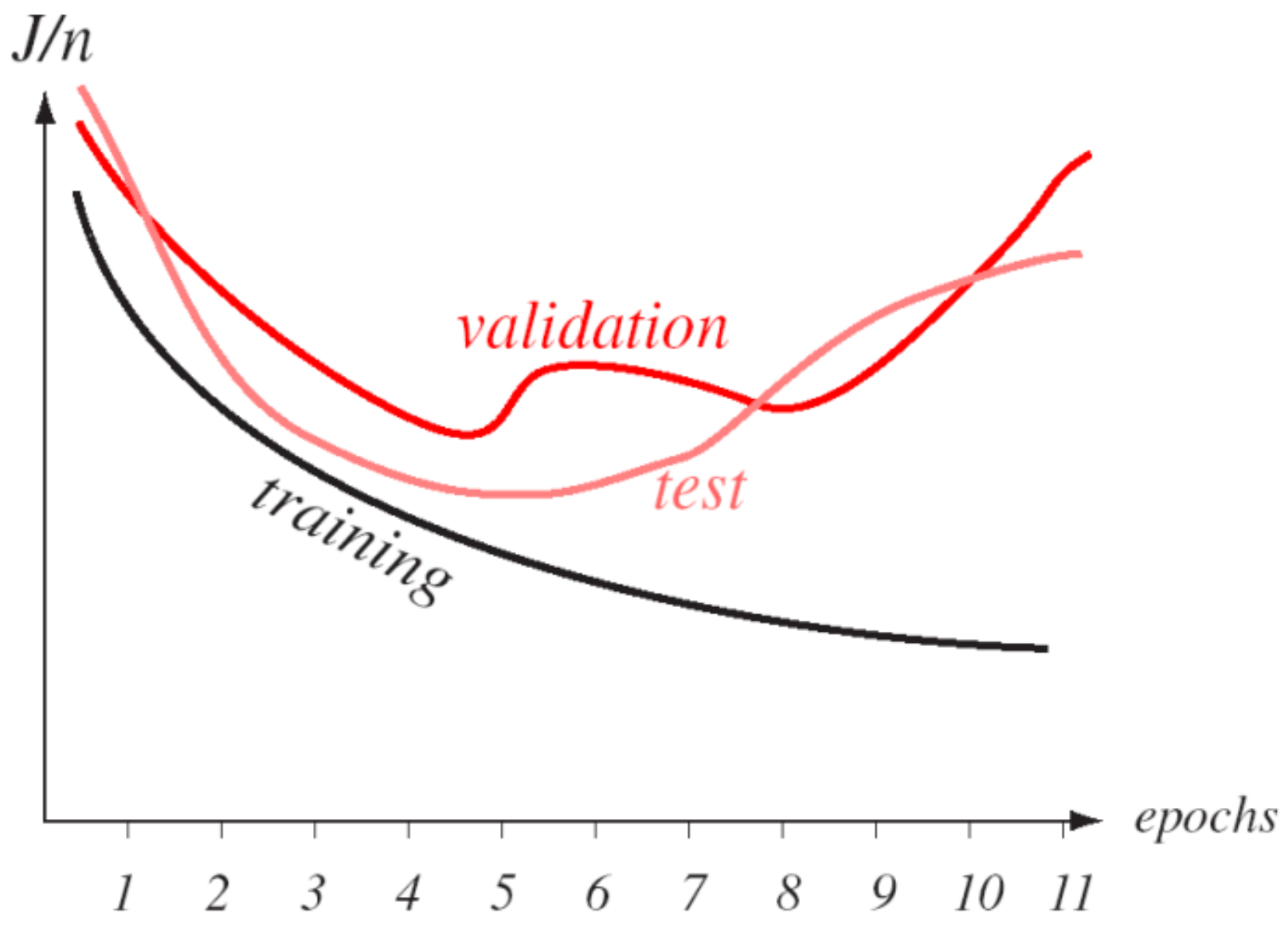
- Net-5 hat die beste Performanz. Die Anzahl der *freien* Parameter (1060) ist viel geringer als die Anzahl der Parameter *insgesamt* (5194)

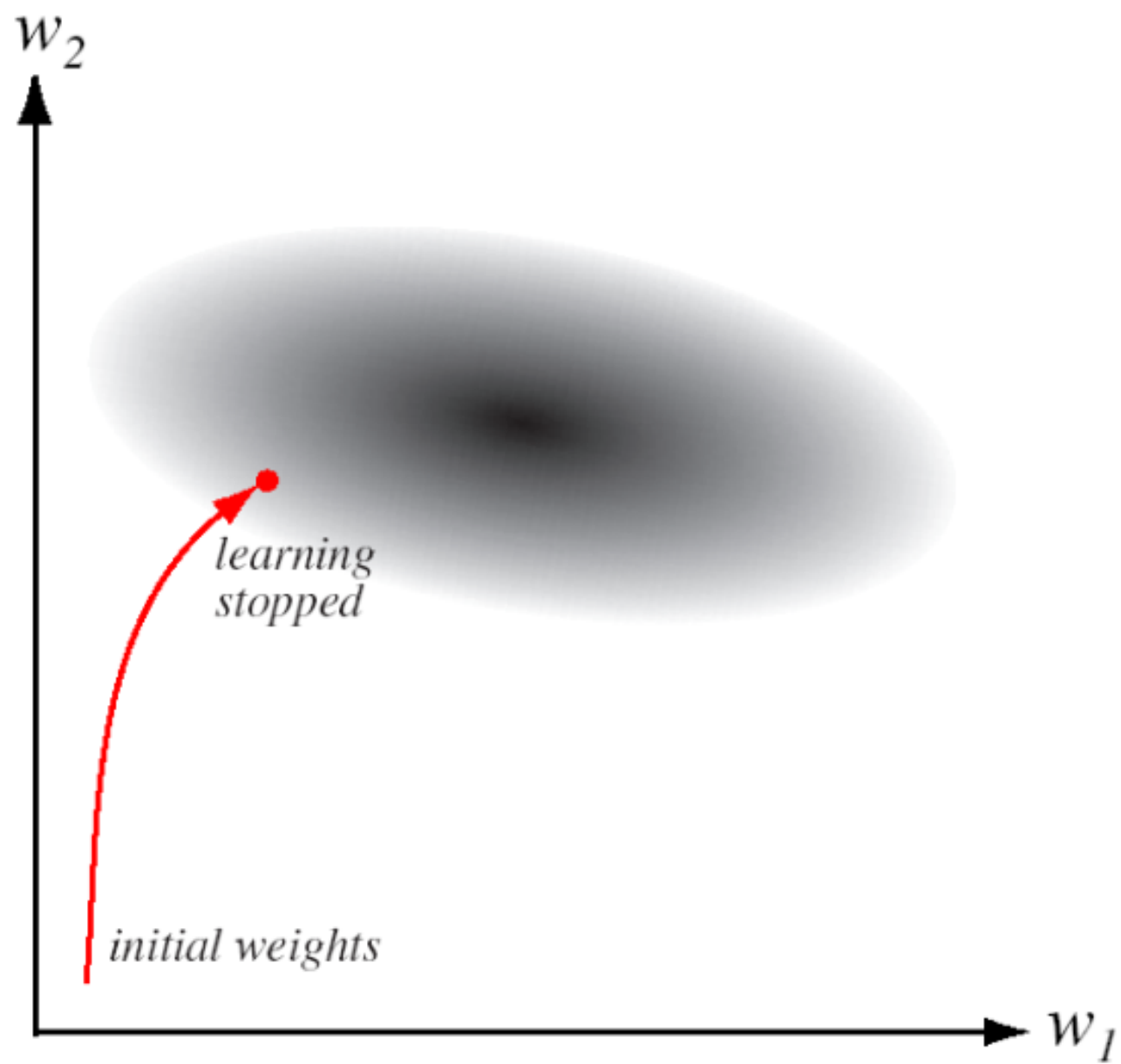
TABLE 11.1. Test set performance of five different neural networks on a handwritten digit classification example (Le Cun, 1989).

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

## Regularisierung mit Stopped-Training

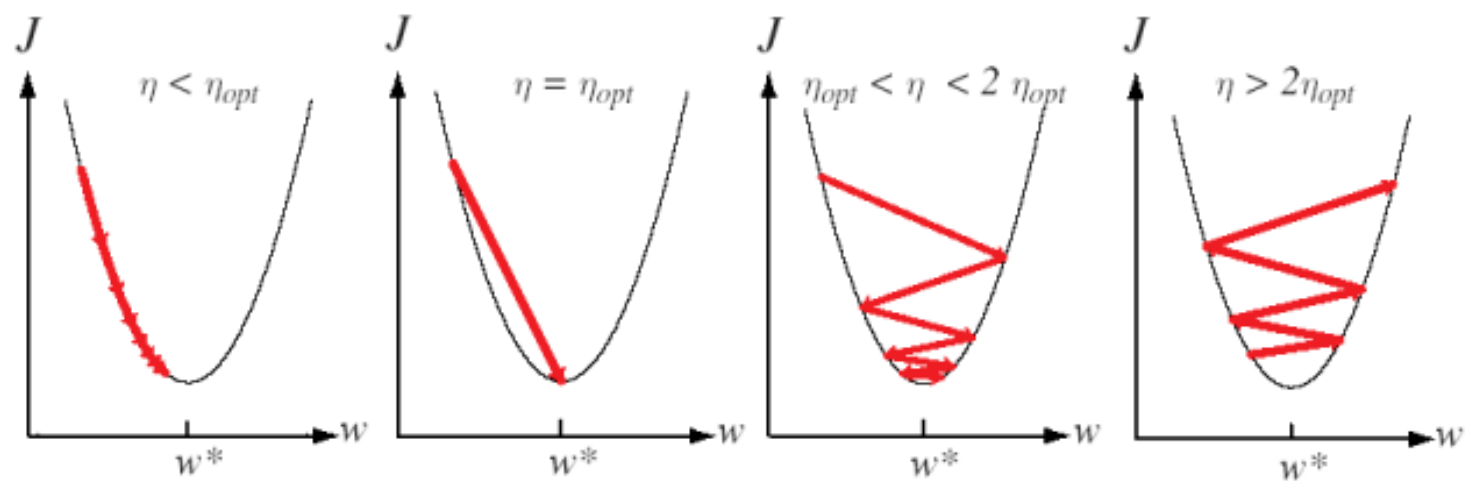
- In der nächsten Abbildung sieht man typische Verläufe für Trainingssatz und Testsatz als Funktion der Trainingsiterationen
- Wie zu erwarten nimmt der Trainingsfehler mit der Trainingszeit stetig ab
- Wie zu erwarten nimmt zunächst der Testfehler ebenso ab; etwas überraschend gibt es ein Minimum und der Testfehler nimmt dann wieder zu
- Erklärung: Im Laufe des Trainings nehmen die Freiheitsgrade im Neuronalen Netz zu; wenn zu viele Freiheitsgrade zur Verfügung stehen, sieht man Überanpassung
- Man kann ein Neuronales Netz dadurch sinnvoll regularisieren, in dem man das Training rechtzeitig beendet (Regularisierung durch Stopped-Training)





## Optimierung der Lernrate $\eta$

- Die Konvergenz kann durch die Größe von  $\eta$  beeinflusst werden
- In der nächsten Abbildung sieht man, dass wenn die Lernrate zu klein gewählt wird, es sehr lange dauern kann, bis das Optimum erreicht wird
- Wird die Lernrate zu gross gewählt, sieht an oszillatorisches Verhalten oder sogar instabiles Verhalten



**FIGURE 6.16.** Gradient descent in a one-dimensional quadratic criterion with different learning rates. If  $\eta < \eta_{opt}$ , convergence is assured, but training can be needlessly slow. If  $\eta = \eta_{opt}$ , a single learning step suffices to find the error minimum. If  $\eta_{opt} < \eta < 2\eta_{opt}$ , the system will oscillate but nevertheless converge, but training is needlessly slow. If  $\eta > 2\eta_{opt}$ , the system diverges. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

## Zusammenfassung

- Neuronale Netze sind sehr leistungsfähig mit hoher Performanz
- Das Training ist aufwendig, aber man kann mit einiger Berechtigung sagen, dass das Neuronale Netz tatsächlich etwas “lernt”, nämlich die optimale Repräsentation der Eingangsdaten in der versteckten Schicht
- Die Vorhersage ist schnell berechenbar
- Neuronale Netze sind universelle Approximatoren
- Neuronale Netze besitzen sehr gute Approximationseigenschaften
- Nachteil: das Training hat etwas von einer Kunst; eine Reihe von Größen müssen bestimmt werden (Anzahl der versteckten Knoten, Lernraten, Regularisierungsparameter, ....)
- Nachteil: Ein trainiertes Netz stellt ein lokales Optimum dar; Nicht-Eindeutigkeit der Lösung

- Aber: In einem neueren Benchmarktest hat ein (Bayes'sches) Neuronales Netz alle konkurrierenden Ansätze geschlagen!



## Zeitreihenmodellierung

- Die nächste Abbildung zeigt eine Finanzdatenzeitreihe (DAX)
- Andere Zeitreihen von Interesse: Energiepreise, Stromverbrauch, Gasverbrauch, ...

## DAX Performance-Index

01.06.07 17:45 Uhr

• 7.987,85

+1,33 % [+104,81]

Enthaltene Werte:

30

Tages-Vol.:

9,12 Mrd.

Typ: Index

Börse: XETRA



## Neuronale Netze in der Zeitreihenmodellierung

- Sei  $x_t, t = 1, 2, \dots$  die zeitdiskrete Zeitreihe von Interesse (Beispiel: DAX)
- Sei  $u_t, t = 1, 2, \dots$  eine weitere Zeitreihe, die Informationen über  $x_t$  liefert (Beispiel: Dow Jones)
- Der Einfachheit halber nehmen wir an, dass  $x_t$  und  $u_t$  skalar sind; Ziel ist die Vorhersage des nächsten Werts der Zeitreihe
- Wir nehmen ein System an der Form

$$x_t = f(x_{t-1}, \dots, x_{t-M_x}, u_{t-1}, \dots, u_{t-M_u}) + \epsilon_t$$

mit i.i.d. Zufallszahlen  $\epsilon_t, t = 1, 2, \dots$ . Diese modellieren unbekannte Störgrößen

## Neuronale Netze in der Zeitreihenmodellierung (2)

- Wir modellieren durch ein Neuronales Netz

$$f(x_{t-1}, \dots, x_{t-M_x}, u_{t-1}, \dots, u_{t-M_u})$$

$$\approx f_{NN}(x_{t-1}, \dots, x_{t-M_x}, u_{t-1}, \dots, u_{t-M_u})$$

und erhalten als Kostenfunktion

$$\text{cost} = \sum_{t=1}^N (x_t - f_{NN}(x_{t-1}, \dots, x_{t-M_x}, u_{t-1}, \dots, u_{t-M_u}))^2$$

- Das Neuronale Netz kann nun wie zuvor mit Backpropagation trainiert werden
- Das beschriebene Modell ist ein NARX Modell: **N**onlinear **A**uto **R**egressive Model with **e**xternal inputs. Andere Bezeichnung: TDNN (time-delay neural network)

## Rekurrente Neuronale Netze

- Wenn  $x_t$  nicht direkt gemessen werden kann, sondern nur verrauschte Messungen zur Verfügung stehen (Verallgemeinerung des linearen Kalman Filters), reicht einfaches Backpropagation nicht mehr aus, sondern man muss komplexere Lernverfahren verwenden:
- Backpropagation through time (BPTT)
- Real-Time Recurrent Learning (RTRL)
- Dynamic Backpropagation

## Deep Learning Recipee (Hinton 2013)

- Use feed-forward neural nets with several big hidden layers composed of rectified linear units:  $\max(0, x)$ .
- Regularize it with dropout: In a large feedforward neural network, overfitting can be greatly reduced by randomly omitting half of the hidden units on each training case
- If the input is spatial, make the first few layers convolutional with overlapping max-pooling ( Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.)

## Android Server Architektur zur Spracherkennung (2013)

- Teilproblem: Modellierung von  $P(State|Frame)$ , um dann mit Bayes Formel

$$P(Frame|State)$$

zu berechnen. Wird dann benötigt im HMM des Spracherkennungsmoduls. Klassischer Ansatz: Gaußschen Mischdichten

- Architektur: 4-10 Schichten, 1000-3000 Knoten/Schicht, rectified linear units, vollverbunden, softmax Ausgang. Training mit cross-entropy Kostenfunktion (siehe Vorlesung: Lineare Klassifikatoren). 2-3 Wochen Training auf GPUs!
- Auf einem neuen Android geschieht die servergestützte Spracherkennung über so ein Neuronales Netz
- Ebenso im offline Modus wird ein kleineres Neuronales Netz eingesetzt: lokal auf dem Smart Phone!

# APPENDIX



## Appendix: Approximationsgenauigkeit Neuronaler Netze

- Dies ist der entscheidende Punkt: *wie viele* innere Knoten sind notwendig, um eine vorgegebene Approximationsgenauigkeit zu erreichen?
- Die Anzahl der inneren Knoten, die benötigt werden, um eine vorgegebene Approximationsgenauigkeit zu erreichen, hängt von der Komplexität der zu approximierenden Funktion ab.
- Barron führt für das Maß der Komplexität der zu approximierenden Funktion  $f(x)$  die Größe  $C_f$  ein, welche definiert ist als

$$\int_{\mathbb{R}^d} |w| |\tilde{f}(w)| dw = C_f,$$

wobei  $\tilde{f}(w)$  die Fouriertransformation von  $f(x)$  ist.  $C_f$  bestraft im Besonderen hohe Frequenzanteile.

- Die Aufgabe des Neuronalen Netzes ist es, eine Funktion  $f(x)$  mit Komplexitätsmaß  $C_f$  zu approximieren.

- Der Eingangsvektor  $x$  ist aus  $\mathbb{R}^d$ , das Neuronale Netz besitzt  $n$  innere Knoten und die Netzapproximation bezeichnen wir mit  $f_n(x)$
- Wir definieren als Approximationsfehler  $AF$  den mittleren quadratischen Fehler zwischen  $f(x)$  und  $f_n(x)$

$$AF = \int_{B_r} (f(x) - f_n(x))^2 \mu(dx). \quad (1)$$

$\mu$  ist ein beliebiges Wahrscheinlichkeitsmaß auf der Kugel  $B_r = \{x : |x| \leq r\}$  mit Radius  $r > 0$

- Barron hat nun gezeigt, dass für jede Funktion, für welche  $C_f$  endlich ist und für jedes  $n \geq 1$  ein Neuronales Netz mit einer inneren Schicht existiert, so dass für den nach letzten Gleichung definierten Approximationsfehler  $AF_{Neur}$  gilt

$$AF_{Neur} \leq \frac{(2rC_f)^2}{n}. \quad (2)$$

- Dieses überraschende Resultat zeigt, dass die Anzahl der inneren Knoten *unabhängig* von der Dimension  $d$  des Eingangsraumes ist.

- Man beachte, dass für konventionelle Approximatoren mit festen Basisfunktionen der Approximationsfehler  $AF_{kon}$  exponentiell mit der Dimension ansteigt :

$$AF_{kon} \propto (1/n)^{2/d}.$$

- Für gewisse Funktionenklassen kann  $C_f$  exponentiell schnell mit der Dimension anwachsen, und es wäre somit nicht viel gewonnen. Barron zeigt jedoch, dass  $C_f$  für große Klassen von Funktionen nur recht langsam mit der Dimension (z.B. proportional) wächst.
- Quellen: Tresp, V. (1995). Die besonderen Eigenschaften Neuronaler Netze bei der Approximation von Funktionen. *Künstliche Intelligenz*, Nr. 4.

A. Barron. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Trans. Information Theory*, Vol. 39, Nr. 3, Seite 930-945, Mai 1993.