Prognose von Zeitreihen mit Hilfe von künstlichen neuronalen Netzen am Beispiel von Börsenprognosen

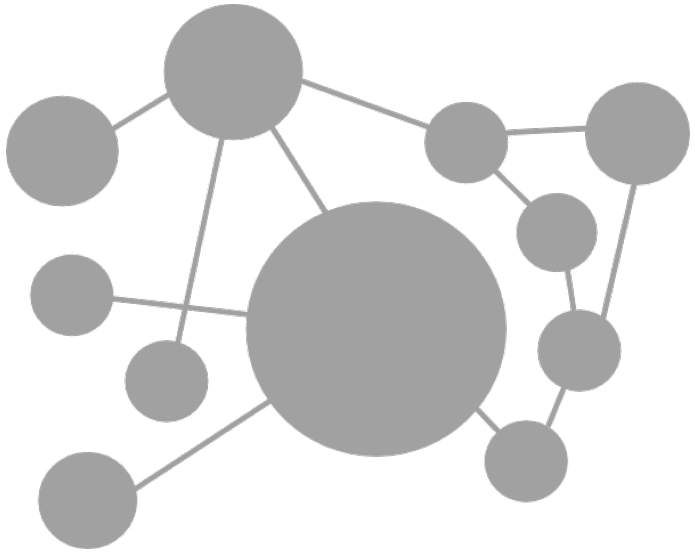
Vortrag zur Seminararbeit

Fach: Softcomputing
Dozent: Prof. Dr. Reinhard Eck

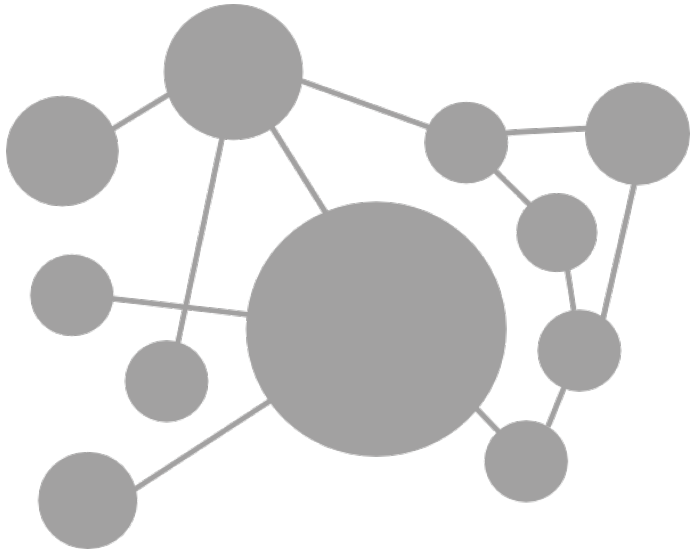
Vorgelegt von:
- Sebastian Schötteler
- Benedikt Hofrichter

GEORG-SIMON-OHM
HOCHSCHULE NÜRNBERG

Inhaltsverzeichnis



Inhaltsverzeichnis



I. Motivation

- Künstliche neuronale Netze als Hilfsmittel zur Prognose:
  - Therapieverläufen in der Medizin
  - Arbeitslosenzahlen auf dem Arbeitsmarkt
  - Börsenkursen

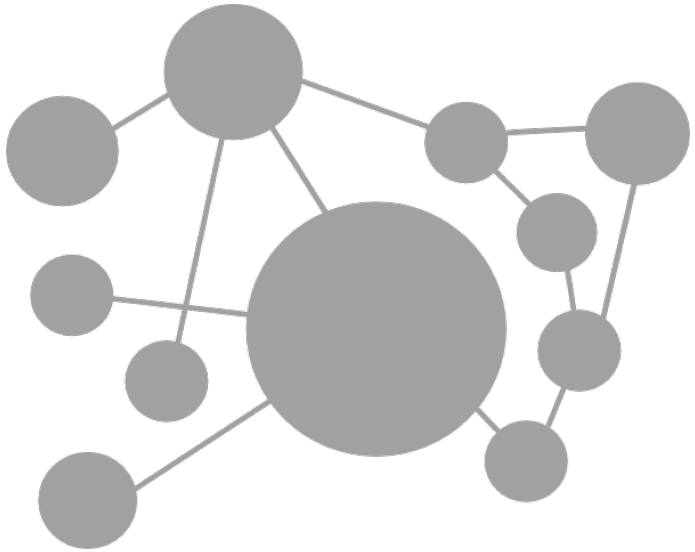
- Besonderheit:
  - Fähigkeit, nichtlineare Zusammenhänge zu erkennen.
  - Prognostiziert objektiv und vorurteilsfrei.

I. Motivation

- Zweck der Seminararbeit:
  - Erstellung einer Anwendung zur Prognose von Börsenkursen mittels KNN.
    - Fokus : Erlangen eines Grundverständnisses über Prognosen mittels KNN.
    - Präzision der Prognosen sollte jedoch nicht vernachlässigt werden.

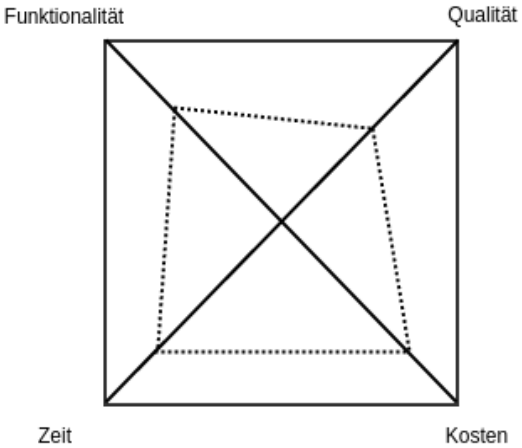
- Die Anwendung soll in der Lage sein...
  - …den zukünftigen Kurs verschiedener Börsen prognostizieren zu können.
  - …eine genaue statistische Analyse der Prognose liefern.

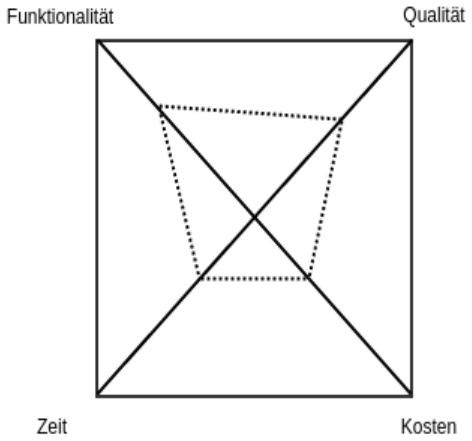
Inhaltsverzeichnis



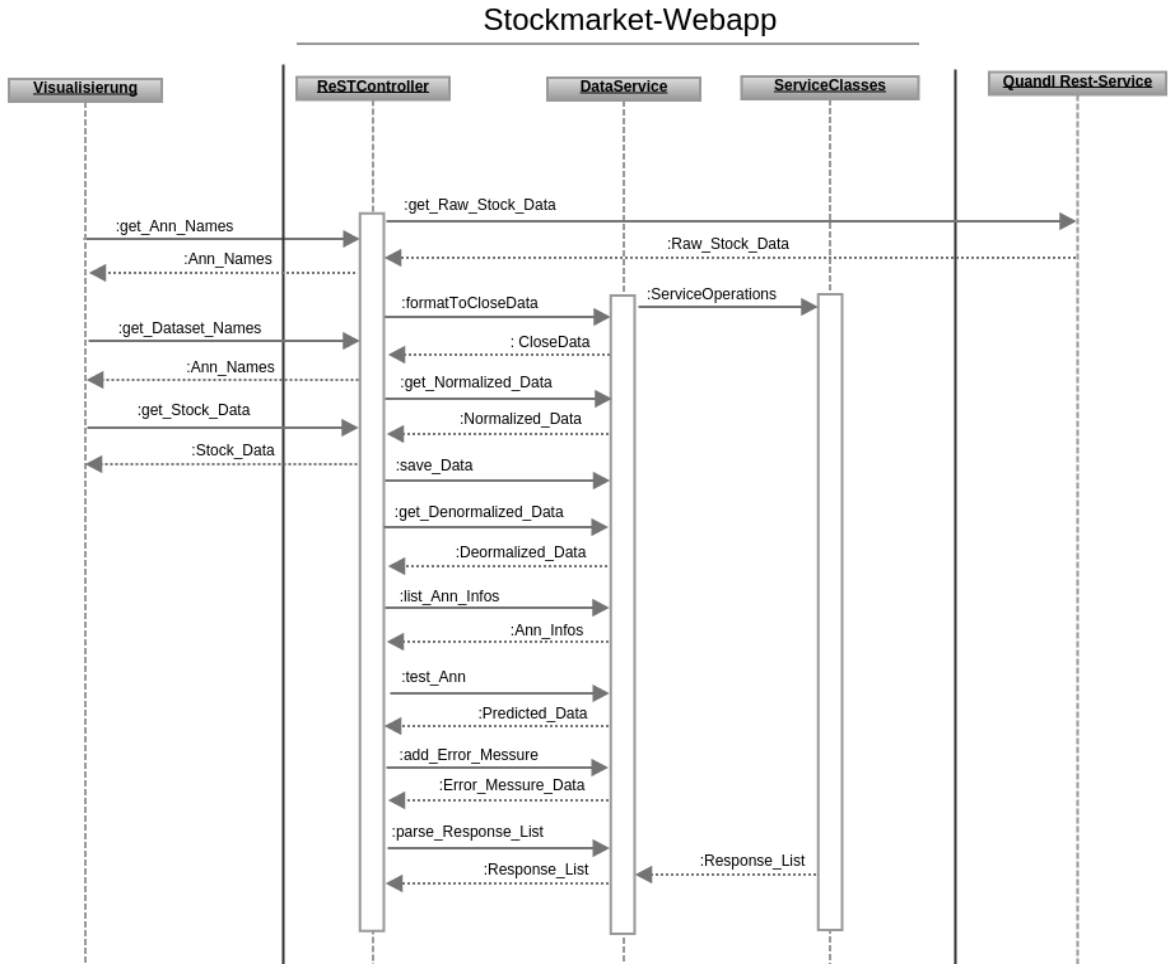
II. Konzeption der Anwendung

- Entscheidungsfaktoren
  - Integrationsfähigkeit
  - Abstraktionsfähigkeit
  - Skalierbarkeit
  - Kosten
  - Schnittstellen





II. Konzeption der Anwendung



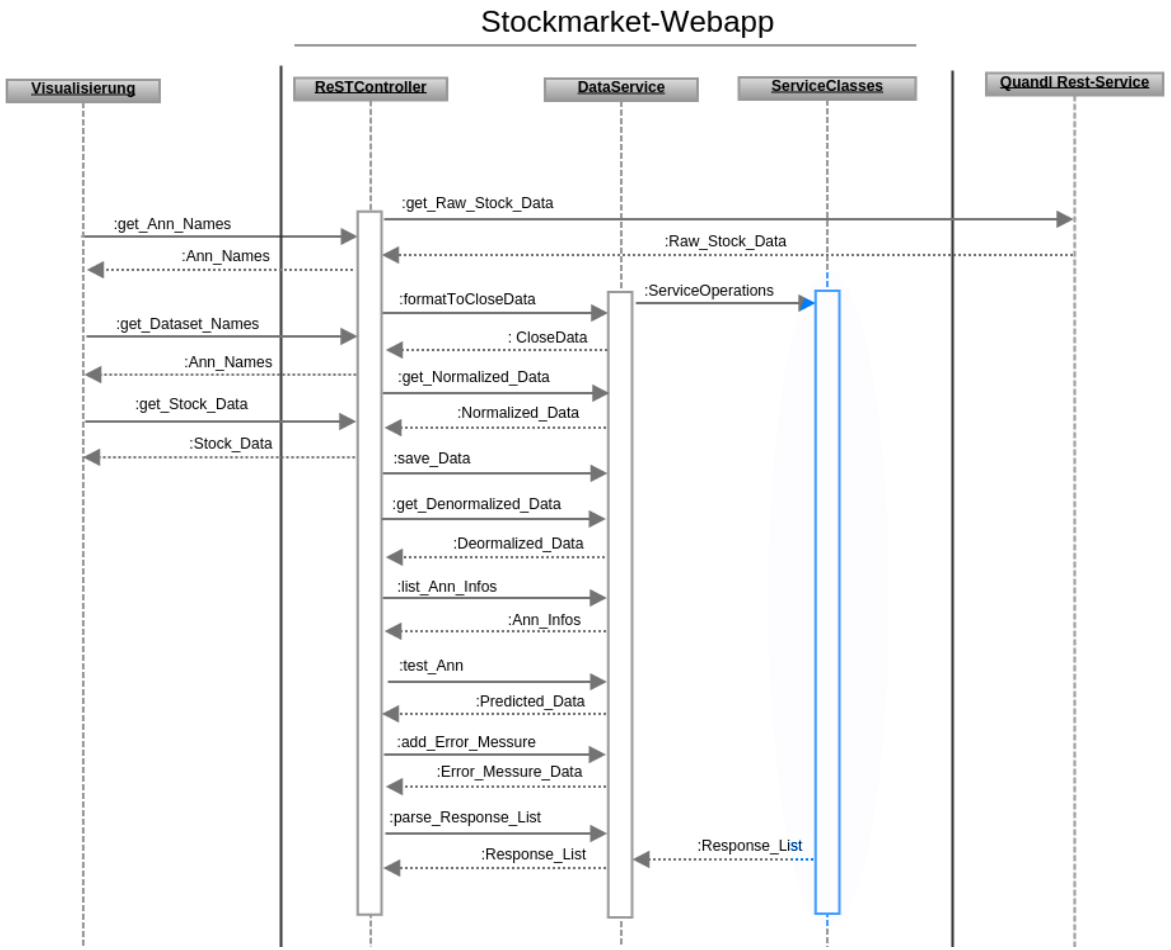
II. Konzeption der Anwendung

- Grundidee der Architektur
  - Anwendung besteht aus zwei Modulen
  - Anwendungslandschaft aus drei Modulen
  - Strikte Trennung zwischen Komponenten
  - Restful Kommunikation
  - Visualisierung (Client)
  - Stockmarket-Webapp (Client / Server)
  - Quandl-API (Server)

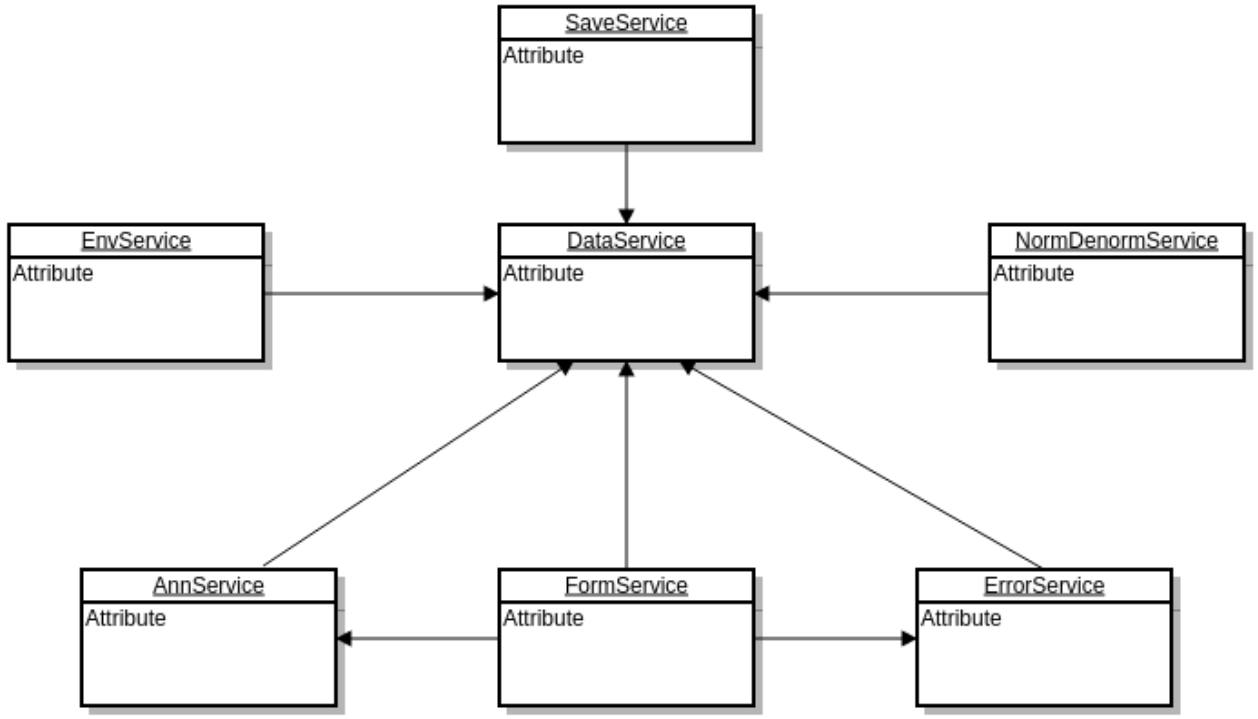
II. Konzeption der Anwendung

- Rest-Kommunikation
  - Adressierbarkeit
  - Variierende Repräsentationen
  - Zustandslosigkeit
  - Zustandsloses Protokoll

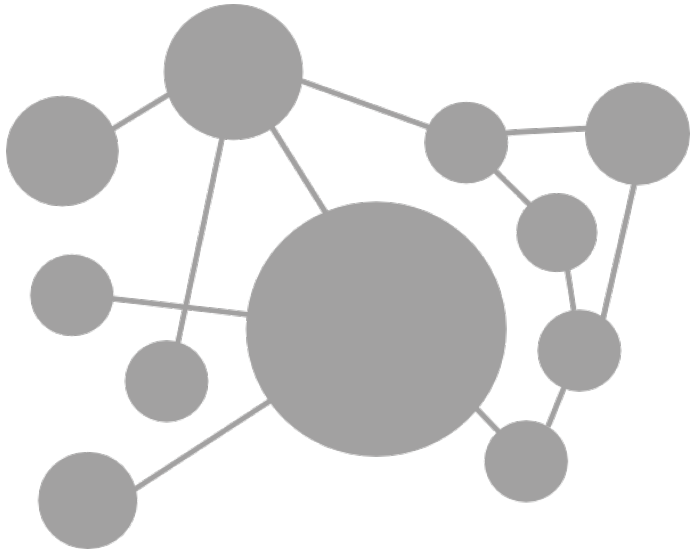
II. Konzeption der Anwendung



II. Konzeption der Anwendung



Inhaltsverzeichnis



III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Heteroassoziative Netze:
    - $\overrightarrow{V_i}\,(1,\dots,n) \rightarrow \overrightarrow{V_o}\,(1,\dots,k); k \leq n$
  - Autoassoziative Netze:
    - $\overrightarrow{V_i}\,(1,\dots,n) \rightarrow \overrightarrow{V_o}\,(1,\dots,n)$

Heteroassoziative Netze	Autoassoziative Netze
Adaline	Hopfield-Netze
Madaline	Boltzmann-Maschine
Perzeptron	
Multilayerperzeptron	

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Wir bilden einen Eingabevektor auf einen skalaren Wert ab.

Heteroassoziative Netze	Autoassoziative Netze
Adaline	Hopfield-Netze
Madaline	Boltzmann-Maschine
Perzeptron	
Multilayerperzeptron	

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Definition & Theorem zur weiteren Bestimmung des Netztyps
    - Definition der linearen Separierbarkeit.
    - Beweis der eingeschränkten Fähigkeit von einschichtigen neuronalen Netzen.
    - Konvergenz-Theorem von Rosenblatt & Theorem der universellen Approximation.

?

Heteroassoziative Netze	Autoassoziative Netze
Adaline	Hopfield-Netze
Madaline	Boltzmann-Maschine
Perzeptron	
Multilayerperzeptron	

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Definition der linearen Separierbarkeit:

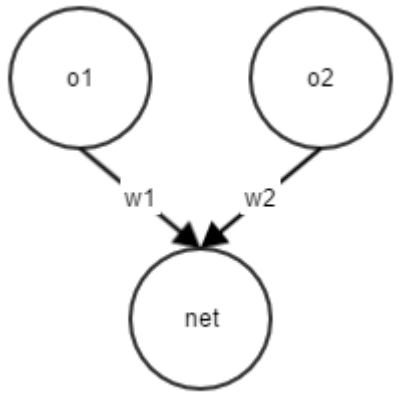
Seien X_1 und X_2 zwei Wertemengen im n-dimensionalen euklidischen Raum. Diese sind genau dann linear sepairerbar, wenn $n + 1$ reelle Zahlen $w_1 \ldots w_n, k$ existieren, sodass für alle $\mathrm{x} \in X_1, y \in X_2$ die folgende Ungleichung erfüllt ist:

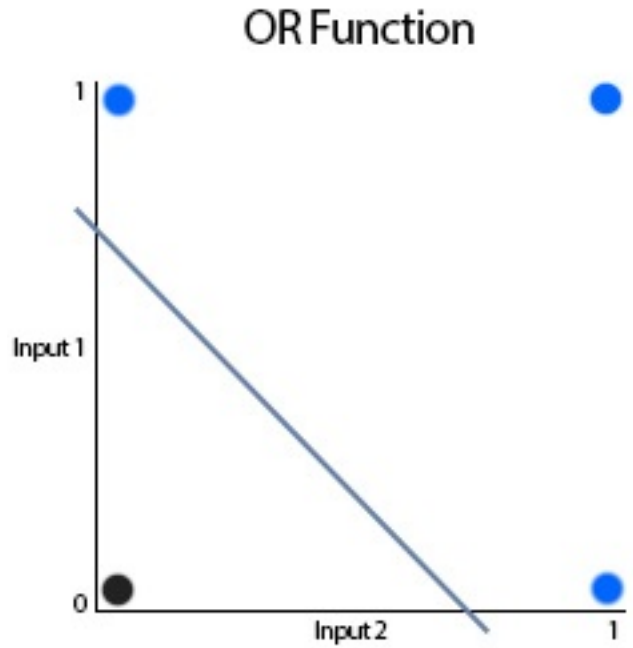
$$\sum_{i=0}^{n} w_i x_i \leq k < \sum_{i=0}^{n} w_i y_i$$

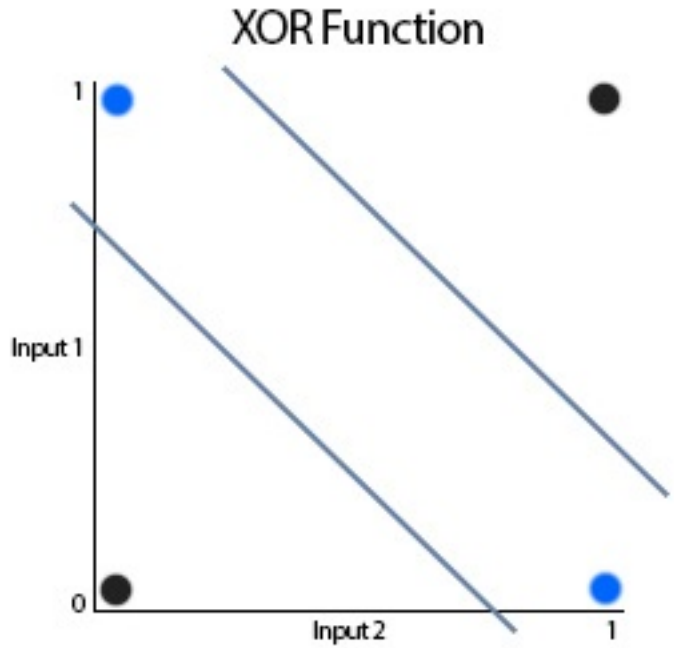
  - 2 Klassen sind linear separierbar, wenn ihre konvexen Hüllen disjunkt sind.
  - 2 Klassen sind linear separierbar, wenn sie durch eine Gerade geteilt werden können.

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Einschichtige neuronale Netze können nur linear separierbare Funktionen klassifizieren

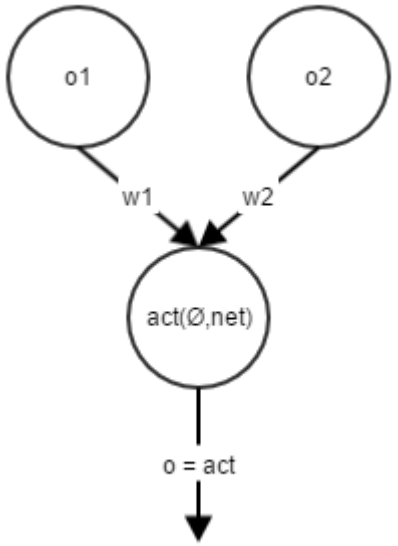






III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Kontradiktionsbeweis der eingeschränkten Fähigkeit von einschichtigen neuronalen Netzen beim XOR-Problem nach Minski / Papert:



Gegeben:

$net = o_1 * w_1 + o_2 + w_2$

$f_{out}(act) = Id \rightarrow o = act$

a) $0 * w_1 + 0 * w_2 < Ø$ Inputvektor (0,0) liefert den Output 0.

b) $0 * w_1 + 1 * w_2 \geq Ø$ Inputvektor (0,1) liefert den Output 1.

c) $1 * w_1 + 0 * w_2 \geq Ø$ Inputvektor (1,0) liefert den Output 1.

d) $1 * w_1 + 1 * w_2 < Ø$ Inputvektor (1,1) liefert den Output 0.

→ Widerspruch: $(b + c): w_1 + w_2 \geq Ø \wedge (d): w_1 + w_2 < Ø$

➡ Beweis auf andere nicht linear separierbare Funktionen anwendbar.

III. Konzeption des künstlichen neuronalen Netzes

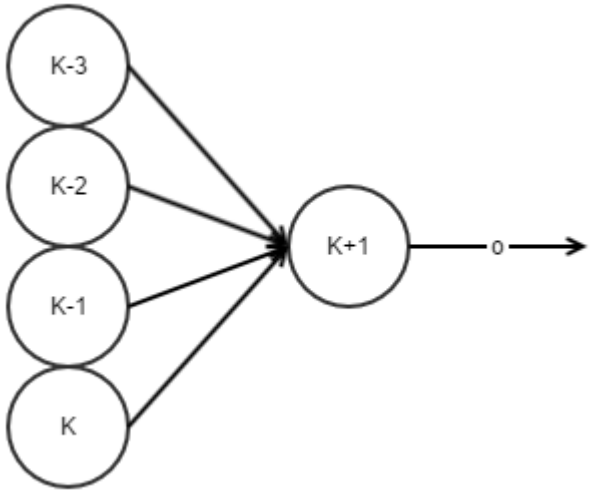
- Netztyp
  - Einschichtige neuronale Netze können nur linear separierbare Funktionen klassifizieren
  - Börsenkurs linear separabel?
  - Konvergenz –Theorem:

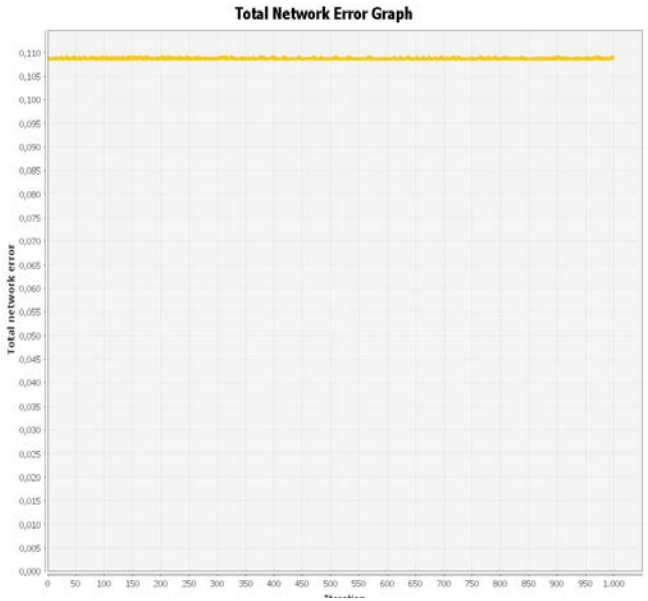
    „Der Lernalgorithmus des Perzeptrons konvergiert in endlicher Zeit, d.h. das Perzeptron kann in endlicher Zeit alles lernen, was es repräsentieren kann.“

    Perzeptron konvergiert ↔ Funktion linear separabel

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Test auf linearer Separierbarkeit:





  - Perzeptron konvergiert nicht → Börsenkurs nicht linear separabel → einlagige neuronale Netze nicht zur Prognose des Börsenkurses geeignet.

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp

Heteroassoziative Netze
Adaline
Madaline
Perzeptron
Multilayerperzeptron

  - Ist ein Multylayerperzeptron zur Vorhersage von Börsenprognosen geeignet?
    - Theorem der universellen Approximation

III. Konzeption des künstlichen neuronalen Netzes

- Netztyp
  - Ist ein Multilayerperzeptron zur Vorhersage von Börsenprognosen geeignet?
    - Theorem der universellen Approximation:

      „Mit Hilfe eines dreischichtigen neuronalen Netzes lassen sich Funktionen beliebig genau approximieren.“
    - Ein Multilayerperzeptron ist also ein universeller Approximator.

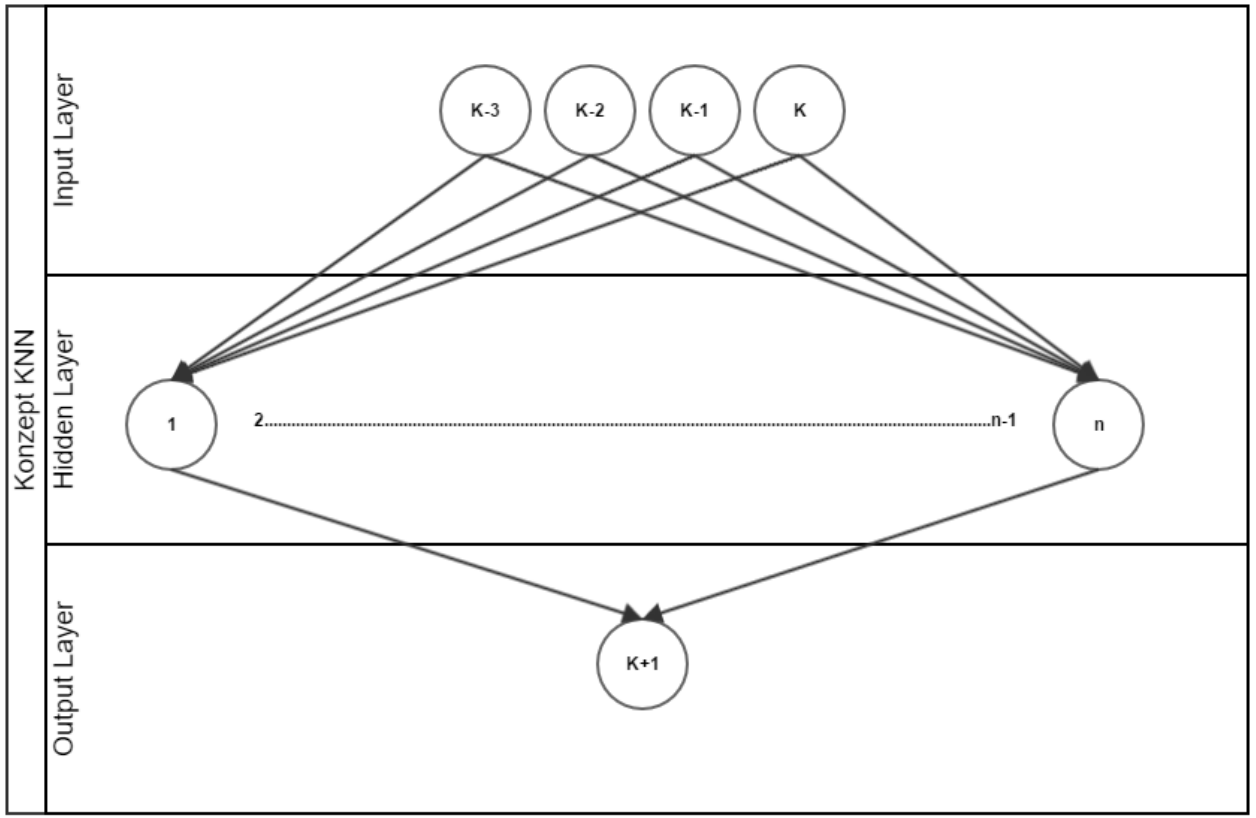
- Fazit: Multilayerperzeptron geeignet.

III. Konzeption des künstlichen neuronalen Netzes

- Topologie
  - K_i = Börsenkurs am Tag i.
  - Ein Vektor $V_i = (K_{i-3}, K_{i-2}, K_{i-1}, K_i)$ der Länge 4 als Input.
  - Ein Skalarwert K_{i+1} als Output.

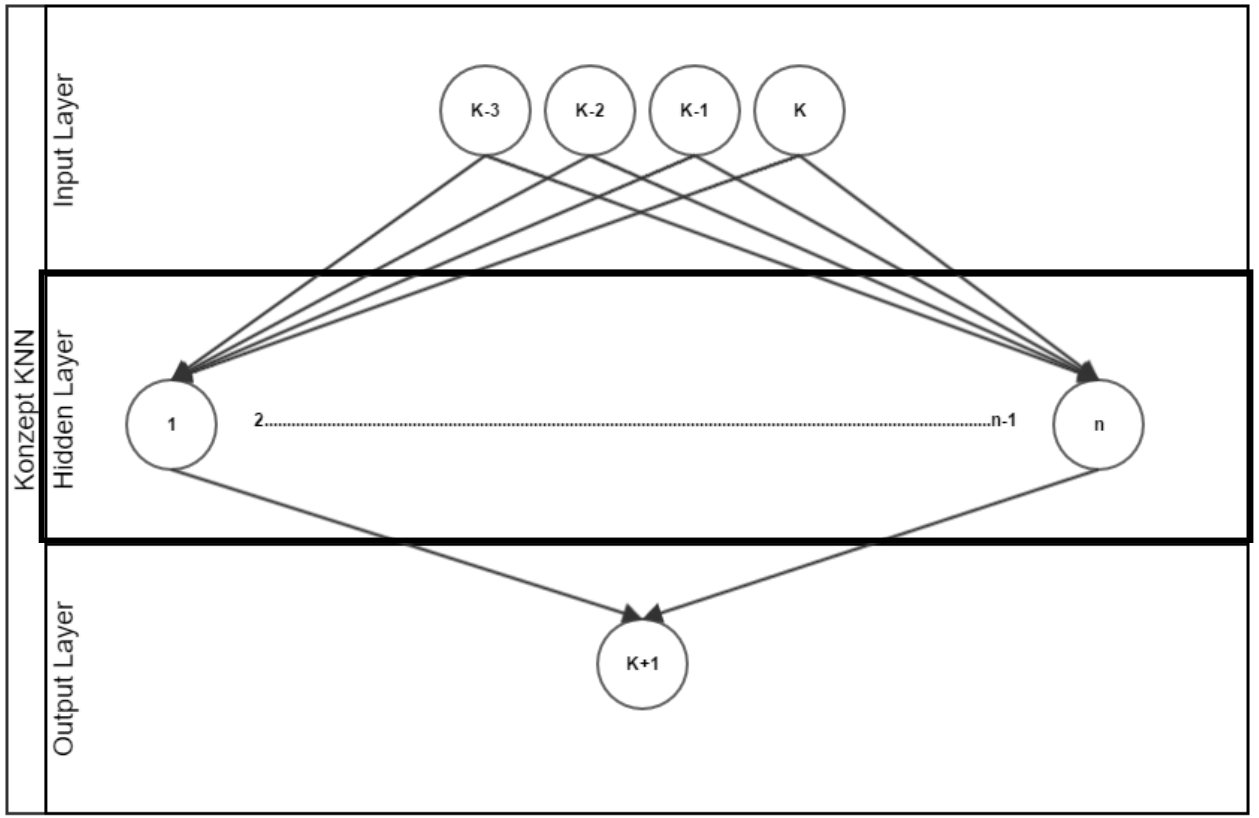
III. Konzeption des künstlichen neuronalen Netzes

- Topologie



III. Konzeption des künstlichen neuronalen Netzes

- Topologie



III. Konzeption des künstlichen neuronalen Netzes

- Topologie
  - Richtlinien zur Dimensionierung der Zwischenschicht:
    - Nicht zu viele Neuronen → Overfitting vermeiden → mangelnde Gen-F.
    - Nicht zu wenig Neuronen → Regelsatz kann nicht abgespeichert werden.
    - Faustregel zur Ermittlung einer Obergrenze:

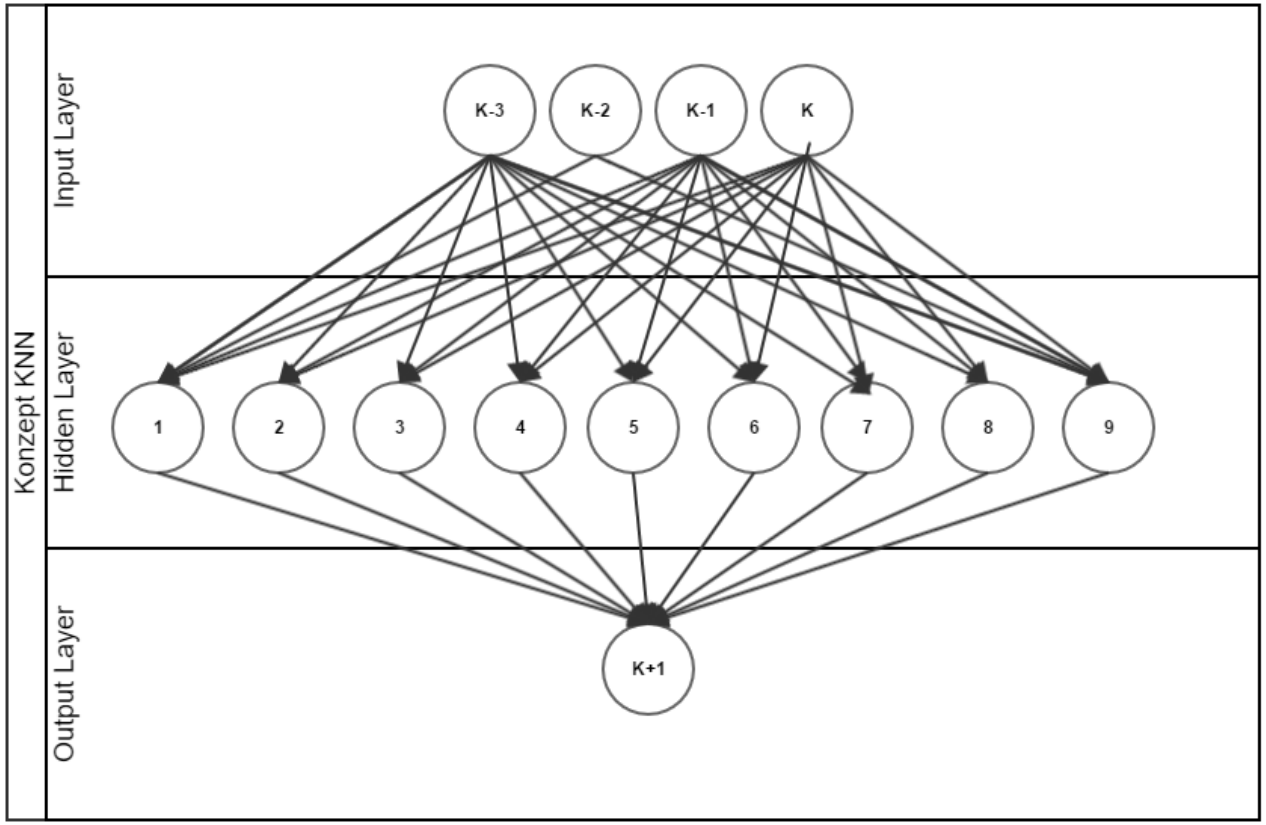
$$h = \frac{Anzahl\ Trainingsdaten}{10 * (m + n)} = \frac{450}{10 * (4 + 1)} = \frac{450}{50} = 9$$

    - $h =$ Obergrenze für die Anzahl der Neuronen in der versteckten Schicht.
    - Es werden 450 Trainingsdaten und 150 Testdaten verwendet.
    - $m =$ Anzahl Inputneuronen; $\mathrm{n} =$ Anzahl Outpurneuronen.

Bieten nur einen Anhaltspunkt

III. Konzeption des künstlichen neuronalen Netzes

- Topologie



III. Konzeption des künstlichen neuronalen Netzes

- Lernverfahren
  - Überwachtes Lernen
    - Eingabewerte bekannt
    - Erwartete Ausgabewerte bekannt
    - Tatsächlicher Wert wird mit erwarteten Ausgabewert verglichen.
    - Differenz wird gebildet und zum „trainieren“ des Netzes genutzt.
  - Bestärkendes Lernen
    - Ähnlich wie überwachtes Lernen.
    - Anwendbar, wenn keine Ausgabewerte zur Verfügung stehen.
    - Netz erhält nur Information ob richtig oder falsch und muss damit trainiert werden.
  - Nicht überwachtes Lernen
    - Sehr nah am biologischen Vorbild.
    - Das Neuronale Netz verändert sich entsprechend den Eingabemustern von selbst.

III. Konzeption des künstlichen neuronalen Netzes

- Lernverfahren
  - Überwachtes Lernen
    - Eingabewerte bekannt
    - Erwartete Ausgabewerte bekannt
    - Tatsächlicher Wert wird mit erwarteten Ausgabewert verglichen.
    - Differenz wird gebildet und zum „trainieren“ des Netzes genutzt.
  - Bestärkendes Lernen
    - Ähnlich wie überwachtes Lernen.
    - Anwendbar, wenn keine Ausgabewerte zur Verfügung stehen.
    - Netz erhält nur Information ob richtig oder falsch und muss damit trainiert werden.
  - Nicht überwachtes Lernen
    - Sehr nah am biologischen Vorbild.
    - Das Neuronale Netz verändert sich entsprechend den Eingabemustern von selbst.

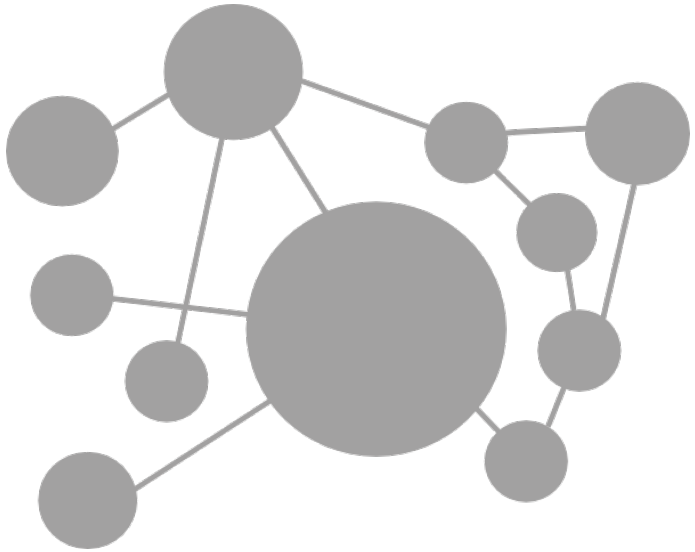
III. Konzeption des künstlichen neuronalen Netzes

- Lernverfahren
  - Überwachtes Lernen
    - Eingabewerte bekannt
    - Erwartete Ausgabewerte bekannt
    - Tatsächlicher Wert wird mit erwarteten Ausgabewert verglichen.
    - Differenz wird gebildet und zum „trainieren“ des Netzes genutzt. → MSE
  - MSE - Funktion:

$$MSE = \frac{1}{2}\sum_{j}^{n}((k_{i+1})_j - (k_{i+1}{}')_j)^2$$

  - k_{i+1}, = Prognostizierter Kurs des KNN zum Tag $i+1$.
  - $k_{i+1}{}'$ = Echter Kurs zum Tag $i+1$.

Inhaltsverzeichnis



IV. Umsetzung der Anwendung

Entwicklung	Laufzeit
Java 8 – JDK 1.8	Java 8 – JRE 1.8
Tomcat 7.0.64	Tomcat 7.0.64
Intellij IDE	
Linux Mint 17.1	
Neuroph-Studio-2.92	

- Vorteile:
- Betriebssystemunabhängigkeit
- Kontextbasierte Entwicklung
- Neuroph ist eine Java-Anwendung

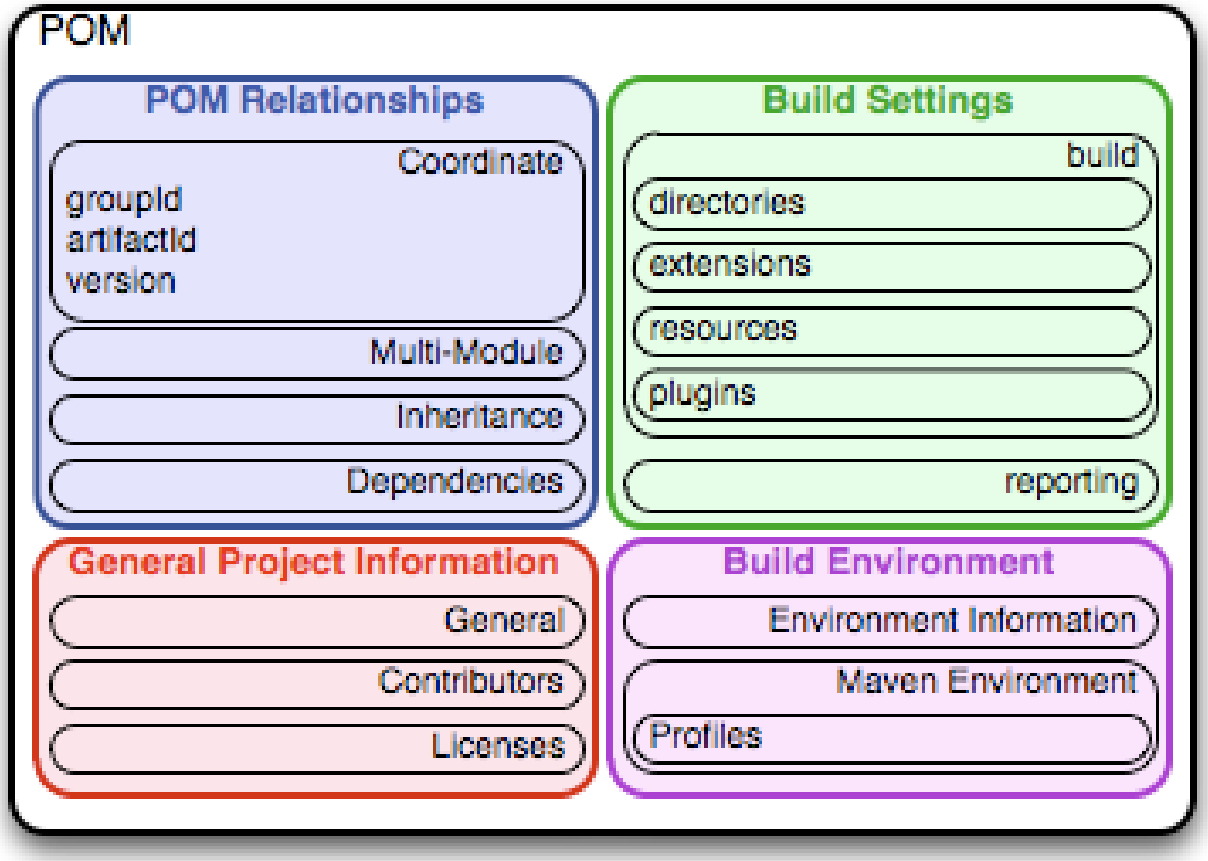
IV. Umsetzung der Anwendung

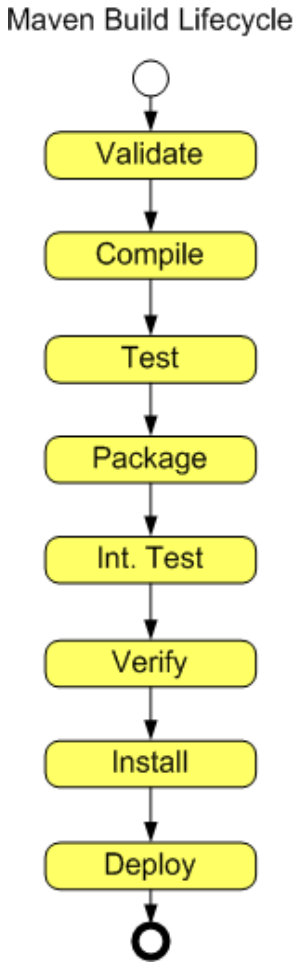
- Frameworks
  - Apache Maven
  - Spring Boot
  - C3js
  - Bootstrap

IV. Umsetzung der Anwendung

- Apache Maven
  - Build-Werkzeug
  - Abhängigkeiten-Management (Dependency Management)
  - Local und Remote Repositories
  - Maven-Build-Lifecycle
  - Anwendungspezifisches Management definiert das Project Object Model (POM.xml)
  - Alternativen sind Apache Ant und Gradle
  - Vorteile: Skalierbarkeit, Komplexitätsverringerung ...

IV. Umsetzung der Anwendung





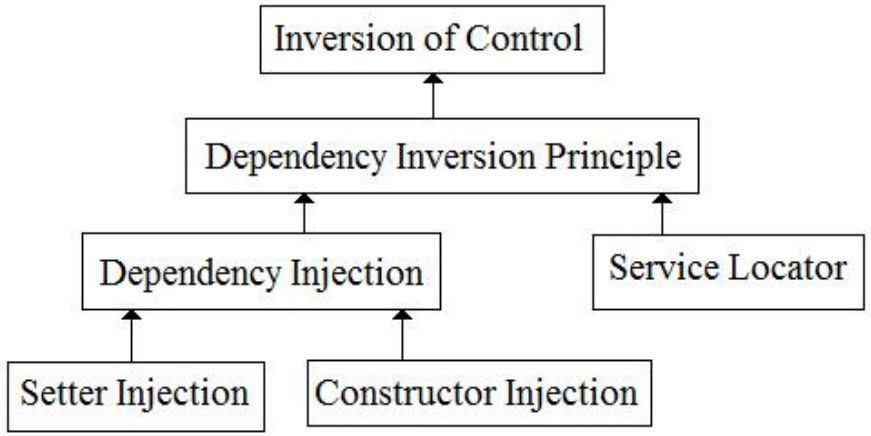
IV. Umsetzung der Anwendung

- Spring Boot
  - Umsetzung des Spring Frameworks
  - Besseres Code Management
  - Vermeidung von Boiler Plate Code
  - Elementarer Bestandteile
  - Abhängigkeitsinjizierung (Dependency Injection)
  - Annotationen

IV. Umsetzung der Anwendung

- Abhängigkeit-Injizierung
  - Entwurfsmuster (Software Pattern)
  - Ziel: Abhängigkeitsminimierung zwischen Java-Klassen
  - Abhängigkeiten werden beim Aufruf übergeben.
  - Arten:
  - Inversion-of-Control (Spring IoC-Container)
  - Konstruktor Injektion
  - Setter Injektion

IV. Umsetzung der Anwendung



IV. Umsetzung der Anwendung

- Annotationen
  - Implementierung von Interface
  - Typen:
  - Retention.SOURCE – Typen
  - Retention.RUNTIME – Typen

```
@Component
public class DataService {

    @Autowired
    private FormatService formatService;
```

IV. Umsetzung der Anwendung

- C3js – Power für die Börsencharts
  - Basiert auf D3js
  - Relativ schlanker Ansatz
  - Informationsgehalt kann angemessen dargestellt werden
  - Visualisierung aller Diagramme
  - Alternativen: D3js, NVD3, CanvasJS, Crossfilter, …

IV. Umsetzung der Anwendung

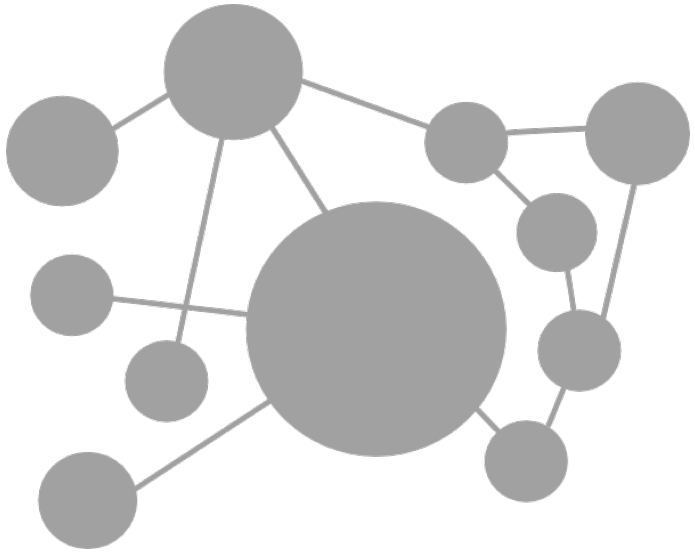
- Bootstrap – CSS Bibliothek
  - Entwickelt in CSS, LESS, JavaScript
  - Ermöglicht Responsive Design
  - Hauptaufgabe: Formatierung des Layouts

IV. Umsetzung der Anwendung

- Technologie - Fazit
  - Integrationsfähigkeit durch modulare Entwicklung und einheitliche Schnittstellen
  - Abstraktionsfähigkeit durch intelligente Frameworks und Entwurfsmuster
  - Skalierbarkeit durch Rest-Kommunikation (Abhängigkeit-Injizierung)

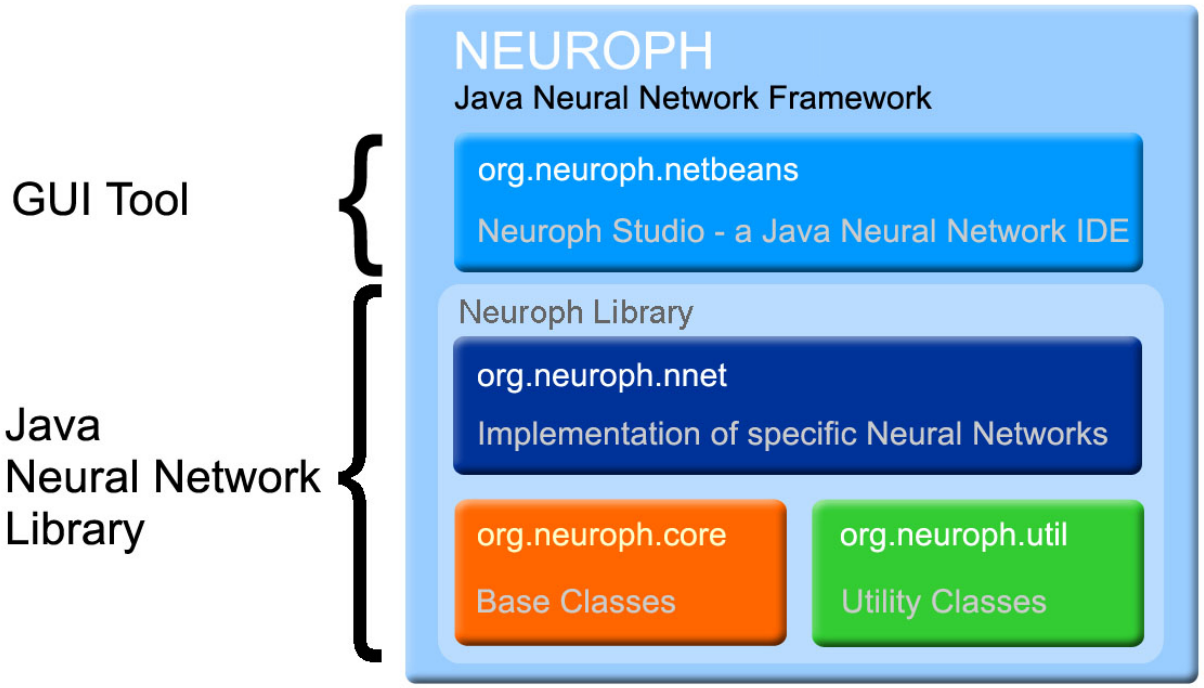
  → Nachhaltige Entwicklung

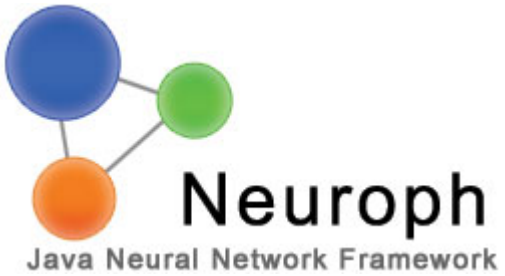
Inhaltsverzeichnis



V. Umsetzung des künstlichen neuronalen Netzes

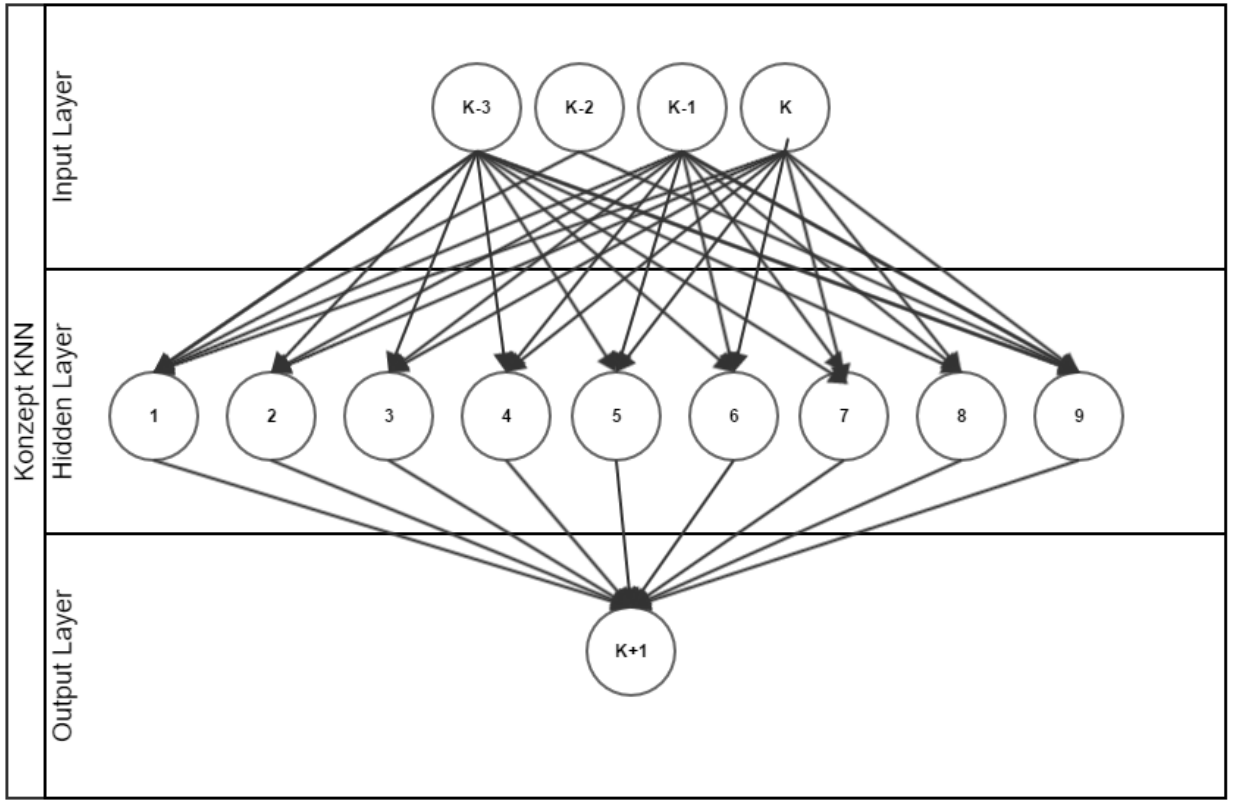
- Umsetzung mit Neuroph Studio





V. Umsetzung des künstlichen neuronalen Netzes

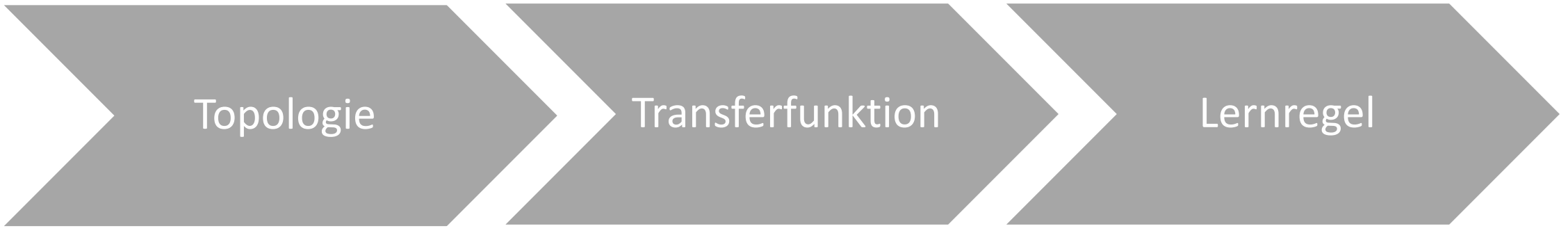
- Startnetz aus der Konzeption



Topologie	4-9-1
Transferfunktion	Sigmoid
Lernregel	Backpropagation
Lernrate	0,7

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierungsprozess



- …
- 4-5-1 (B)
- **4-9-1** (B)
- 4-13-1 (B)
- …

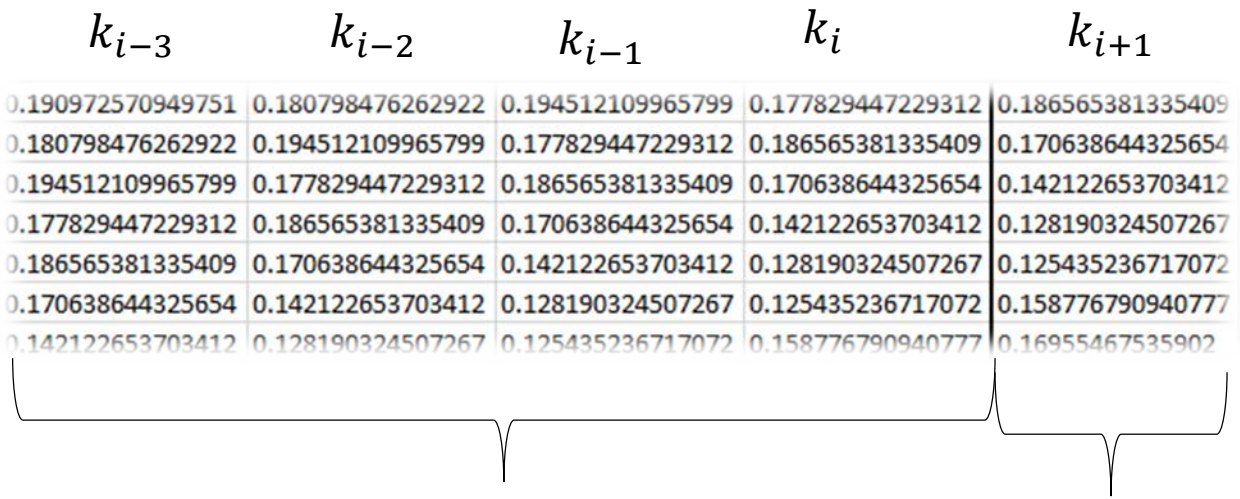
- **Sigmoid**
- Tanh

- **Backpropagation**
- M-Backpropagation
- R-BP

Jeweils:
5 mal mit je
10.000 Zyklen

V. Umsetzung des künstlichen neuronalen Netzes

- Datensätze
  - Trainingsdatensatz: 600 Daten
  - Testdatensatz: 200 Daten



Normalisierungsformel:

$$Norm = \frac{A - \min(A)}{\max(A) - \min(A)} * 0{,}8 + 0{,}1$$

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Topologie

Topologie	MSE		MSE-BIAS	
	Training	Test	Training	Test
4-3-1 (B)	0,0011562	0,002569	$9{,}449 \cdot 10^{-4}$	0,001788
4-5-1 (B)	0,001062	0,002879	$9{,}598 \cdot 10^{-4}$	0,001799
4-7-1 (B)	0,001090	0,001784	$9{,}407 \cdot 10^{-4}$	0,001781
4-9-1 (B)	0,001048	0,002134	$9{,}488 \cdot 10^{-4}$	0,0024436
4-11-1 (B)	0,001022	0,001785	$9{,}760 \cdot 10^{-4}$	0,0033215
4-13-1 (B)	0,001002	0,001787	$9{,}906 \cdot 10^{-4}$	0,004067

- B steht hierbei für Bias-Neuron → Schnellere Konvergenz.

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Topologie

Topologie	MSE		MSE-BIAS	
	Training	Test	Training	Test
4-3-1 (B)	0,0011562	0,002569	$9{,}449 \cdot 10^{-4}$	0,001788
4-5-1 (B)	0,001062	0,002879	$9{,}598 \cdot 10^{-4}$	0,001799
4-7-1 (B)	0,001090	0,001784	$9{,}407 \cdot 10^{-4}$	0,001781
4-9-1 (B)	0,001048	0,002134	$9{,}488 \cdot 10^{-4}$	0,0024436
4-11-1 (B)	0,001022	0,001785	$9{,}760 \cdot 10^{-4}$	0,0033215
4-13-1 (B)	0,001002	0,001787	$9{,}906 \cdot 10^{-4}$	0,004067

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Transferfunktion

Transferfuntion	MSE	
	Training	Test
Sigmoid	$9{,}406 \cdot 10^{-4}$	0,001767
Tanh	0,010333	0,044330

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Transferfunktion

Transferfuntion	MSE	
	Training	Test
Sigmoid	$9{,}406 \cdot 10^{-4}$	0,001767
Tanh	0,010333	0,044330

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Lernregel

Lernregel	MSE	
	Training	Test
Backpropagation	$9{,}325 \cdot 10^{-4}$	0,001636
Momentum Backpropagation	$9{,}109 \cdot 10^{-4}$	0,001608
Resilient Backpropagation	$8{,}89 \cdot 10^{-4}$	$9{,}406 \cdot 10^{-4}$

V. Umsetzung des künstlichen neuronalen Netzes

- Optimierung der Lernregel

Lernregel	MSE	
	Training	Test
Backpropagation	$9{,}325 \cdot 10^{-4}$	0,001636
Momentum Backpropagation	$9{,}109 \cdot 10^{-4}$	0,001608
Resilient Backpropagation	$8{,}89 \cdot 10^{-4}$	$9{,}406 \cdot 10^{-4}$

V. Umsetzung des künstlichen neuronalen Netzes

- RPROP →„Federndes“ Backpropagation:
  - Verfahren ändert Gewichte nur durch Vorzeichen des Gradienten.
    - Dazu wird der Kurvenanstieg von t und herangezogen (namens $S(t)$)

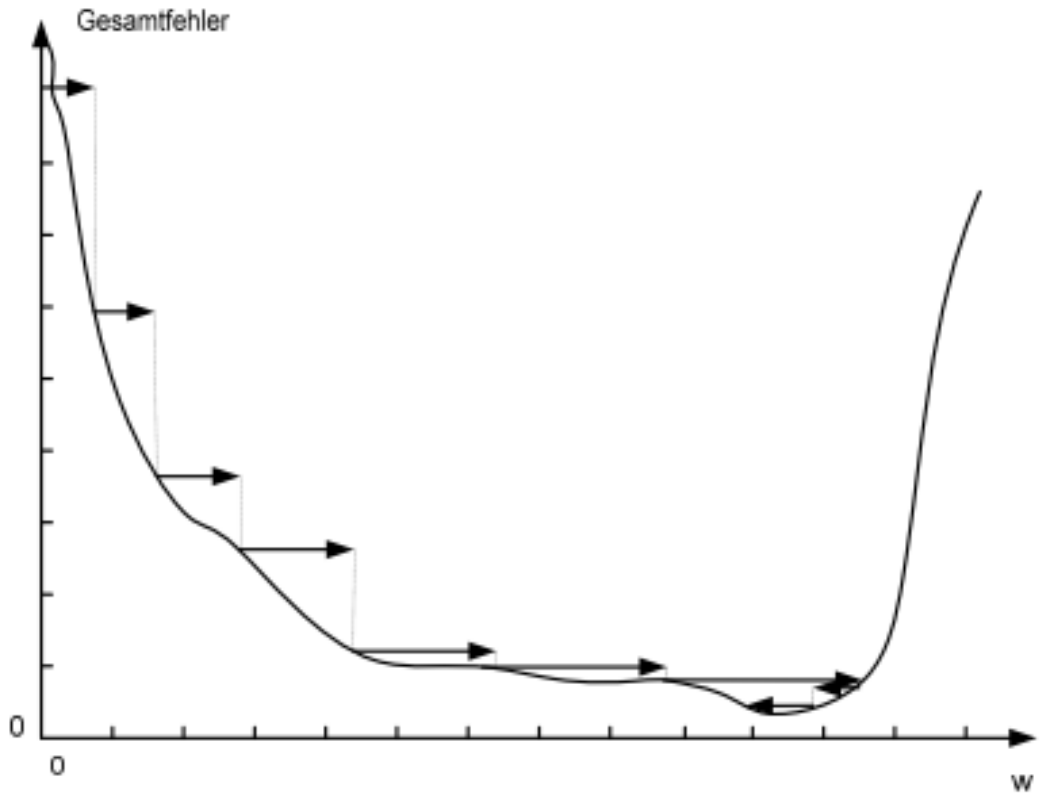
$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t) \; falls \; S(t) > 0 \\ +\Delta_{ij}(t) \; falls \; S(t) < 0 \\ 0 \; sonst \\ - \end{cases}$$

V. Umsetzung des künstlichen neuronalen Netzes

- RPROP →„Federndes“ Backpropagation:
  - Betrag der Gewichtsveränderung Δ_{ij} wird getrennt bestimmt:
    - Zwei konstante Parameter n_+ und n_- mit $0 < n_- < 1 < n_+$

$$\Delta_{ij}(t) = \begin{cases} \Delta_{ij}(t-1) \cdot n_+ \; falls \; S(t-1) \cdot S(t) > 0 \\ \Delta_{ij}(t-1) \cdot n_- \; falls \; S(t-1) \cdot S(t) < 0 \\ \Delta_{ij}(t-1) \; sonst \end{cases}$$

V. Umsetzung des künstlichen neuronalen Netzes

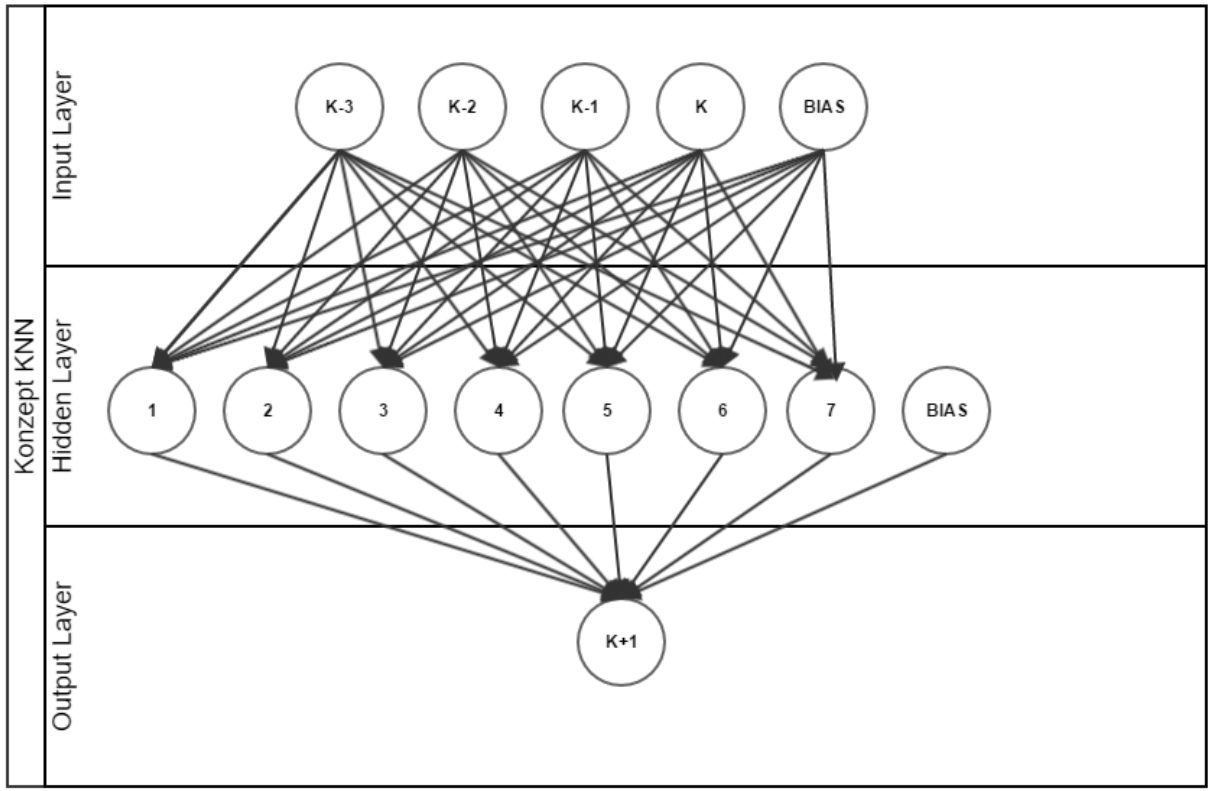


V. Umsetzung des künstlichen neuronalen Netzes

- Resilient Backpropagation:
  - Sehr effizienter Backpropapagation-Algorithmus.
  - Verfügt über keine Lernrate.
  - Benötigt keinen Momentum-Faktor.
  - Ist in der Praxis meistens anderen Lernregeln überlegen.

V. Umsetzung des künstlichen neuronalen Netzes

- Endgültiges Netz – DAX



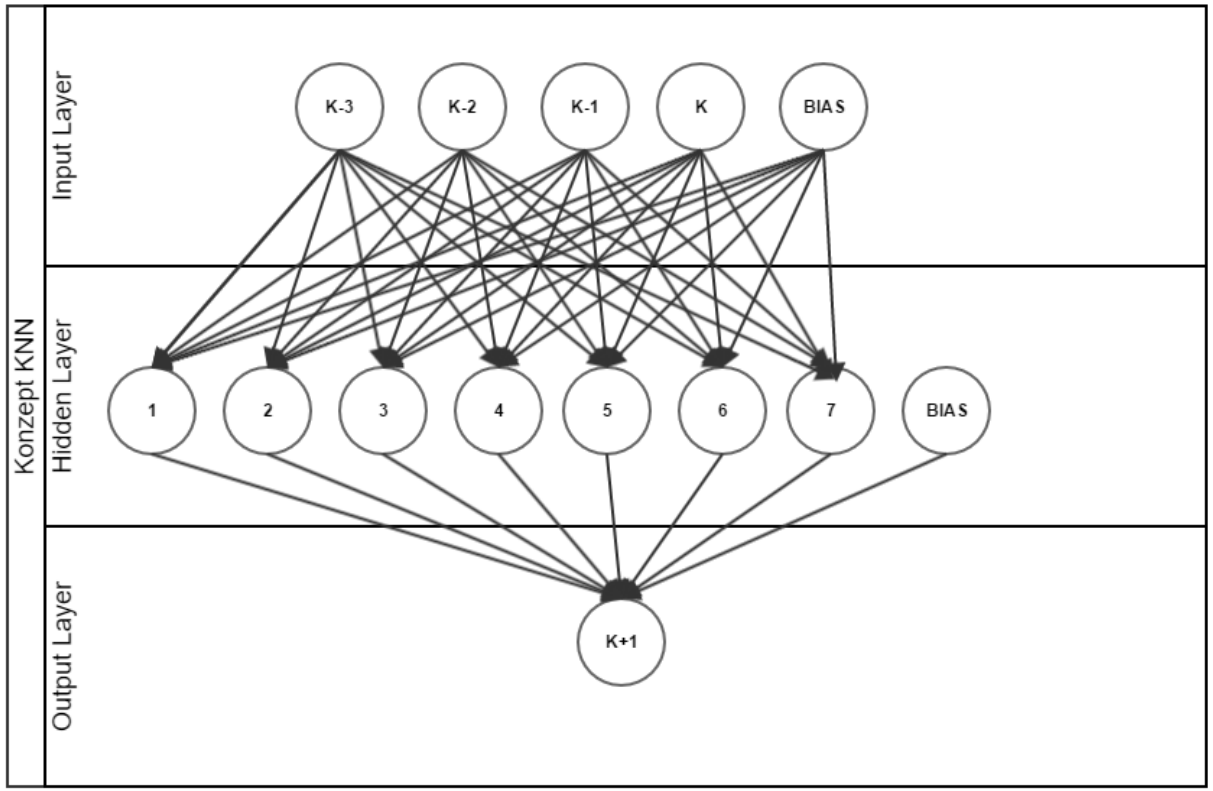
Topologie	4-7-1 mit BIAS
Transferfunktion	Sigmoid
Lernregel	R-Backpropagation

Endgültiges Netz nochmals mit 200.000 Zyklen trainiert und getestet:

- MSE-Training: $4{,}252 \cdot 10^{-5}$
- MSE-Test: $4{,}820 \cdot 10^{-5}$

V. Umsetzung des künstlichen neuronalen Netzes

- Analog – Nikkei 225



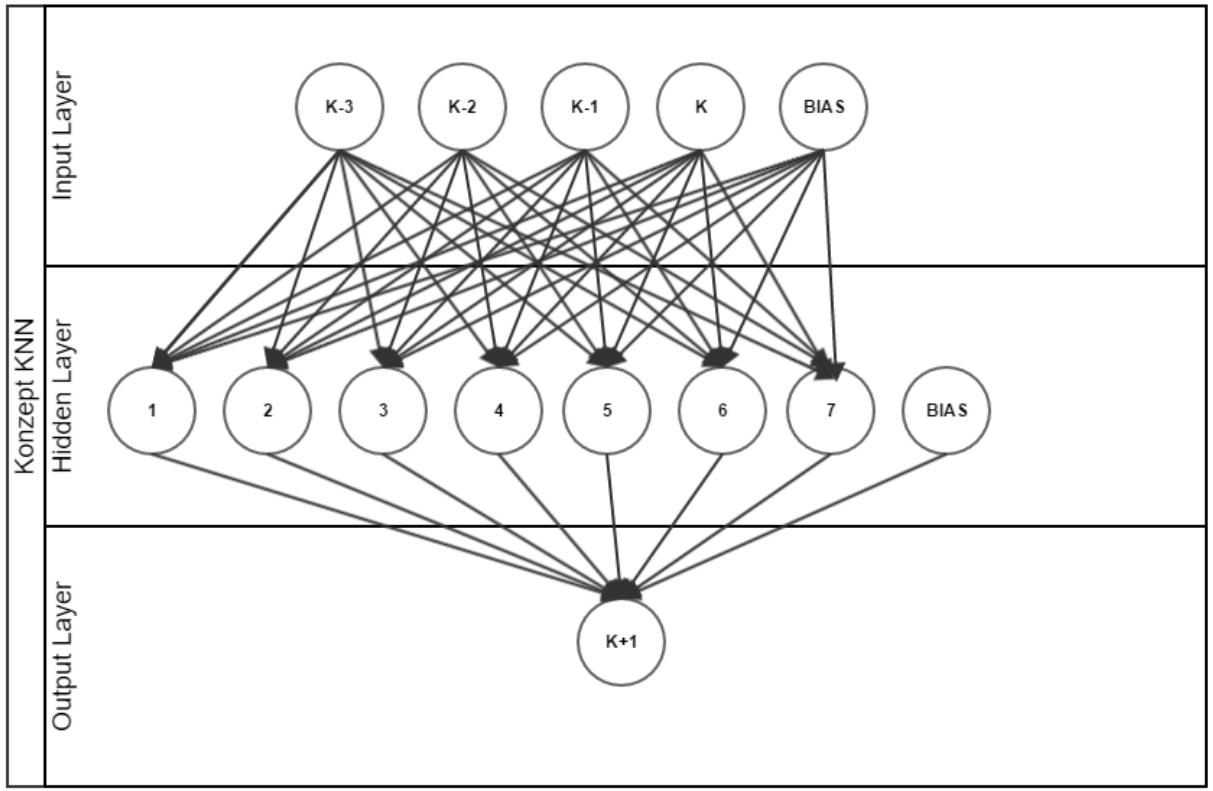
Topologie	4-7-1 mit BIAS
Transferfunktion	Sigmoid
Lernregel	R-Backpropagation

Endgültiges Netz nochmals mit 200.000 Zyklen trainiert und getestet:

- MSE-Training: $1.350 \cdot 10^{-5}$
- MSE-Test: $4{,}520 \cdot 10^{-5}$

V. Umsetzung des künstlichen neuronalen Netzes

- Analog – Dow Jones

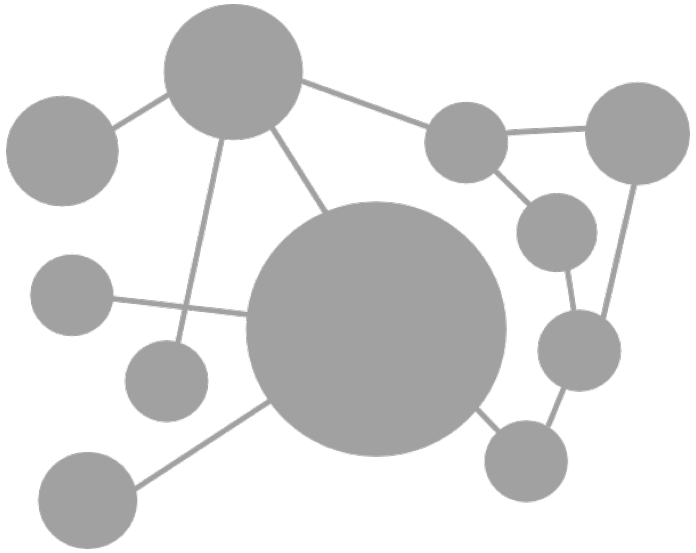


Topologie	4-7-1 mit BIAS
Transferfunktion	Sigmoid
Lernregel	R-Backpropagation

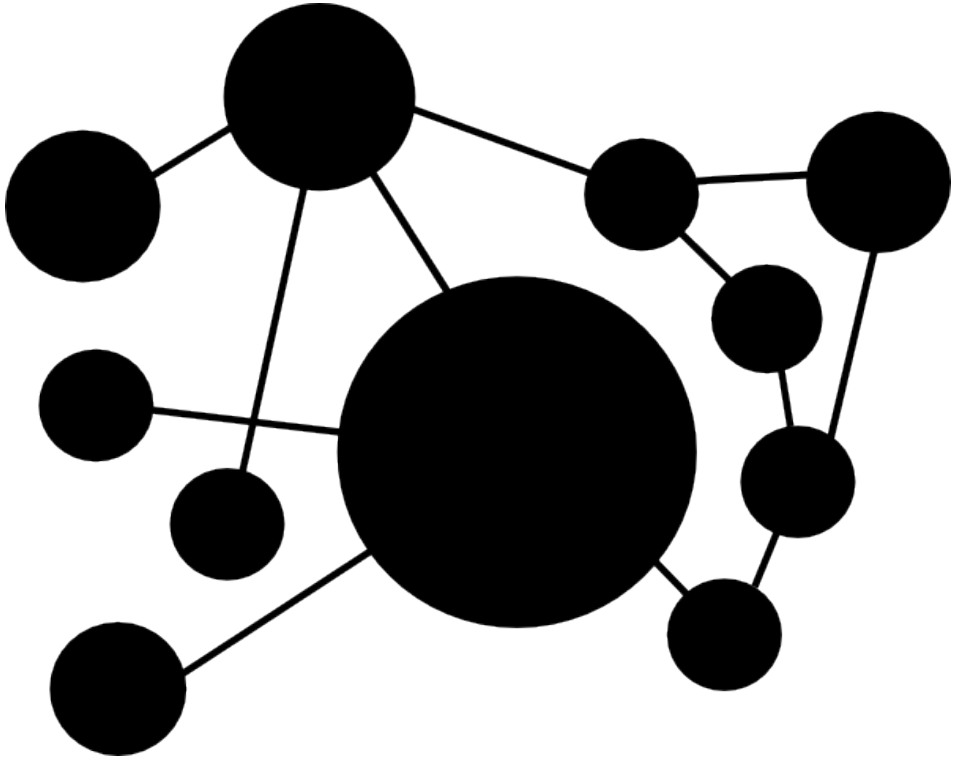
Endgültiges Netz nochmals mit 200.000 Zyklen trainiert und getestet:

- MSE-Training: $6{,}672 \cdot 10^{-5}$
- MSE-Test: $2{,}820 \cdot 10^{-4}$

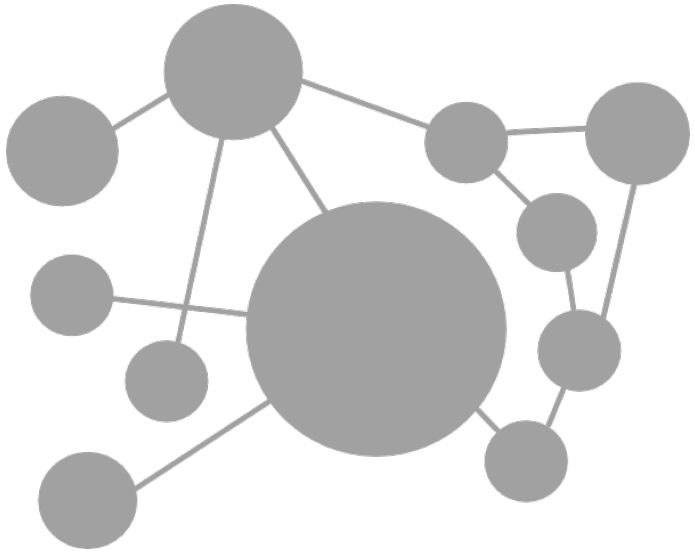
Inhaltsverzeichnis



VII. Livedemonstration der Anwendung



Inhaltsverzeichnis

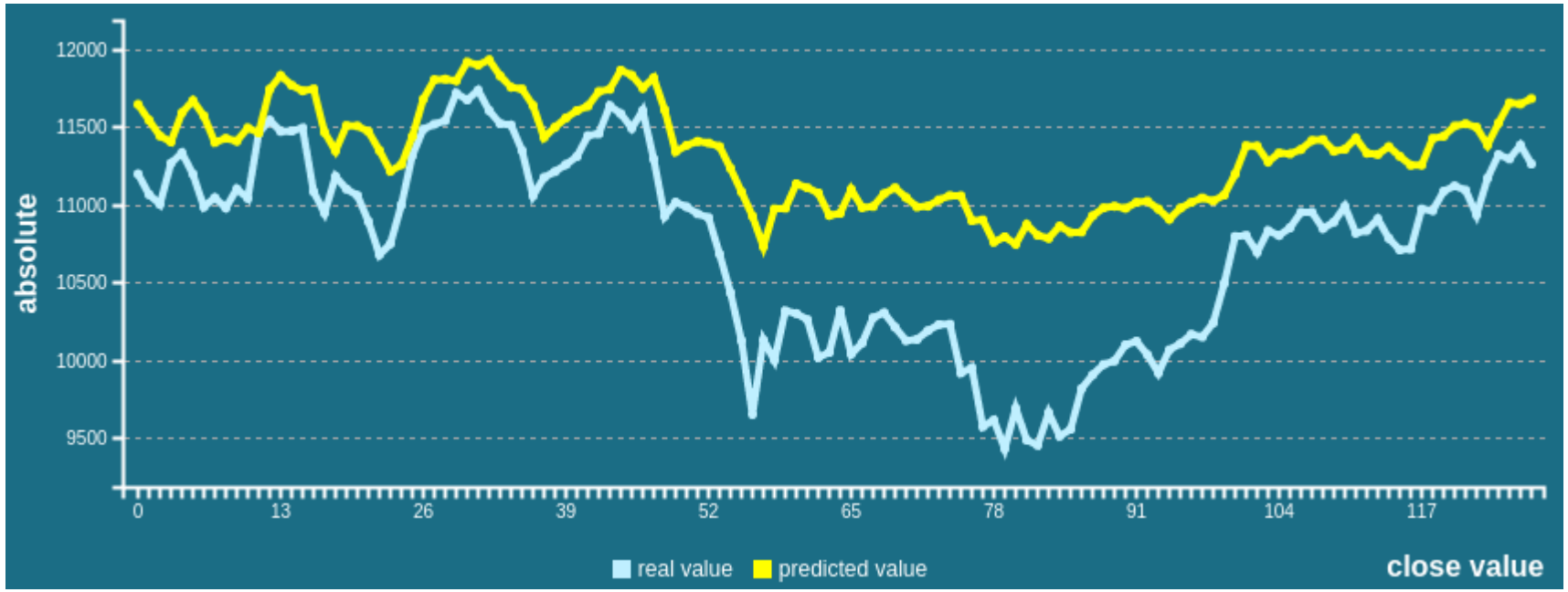


VIII. Analyse

- Börsencrash 2008
  - Zeitraum vom 01.06.2015 bis zum 01.12.2015
- Fukushima 2011 (Nikkei)
  - Zeitraum vom 01.01.2015 bis zum 01.06.2015
- Die letzte Jahreshälfte
  - Zeitraum vom 01.06.2016 bis zum 01.12.2015

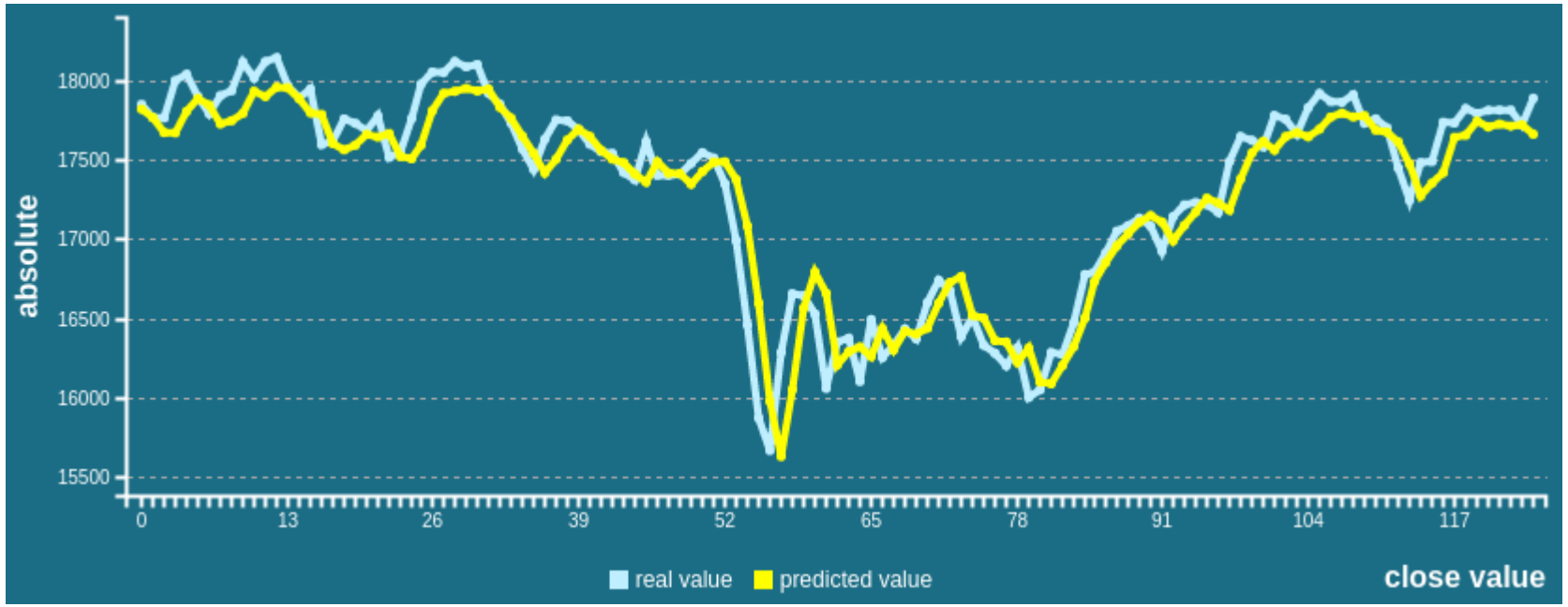
VIII. Analyse

- Letztes halbes Jahr - DAX



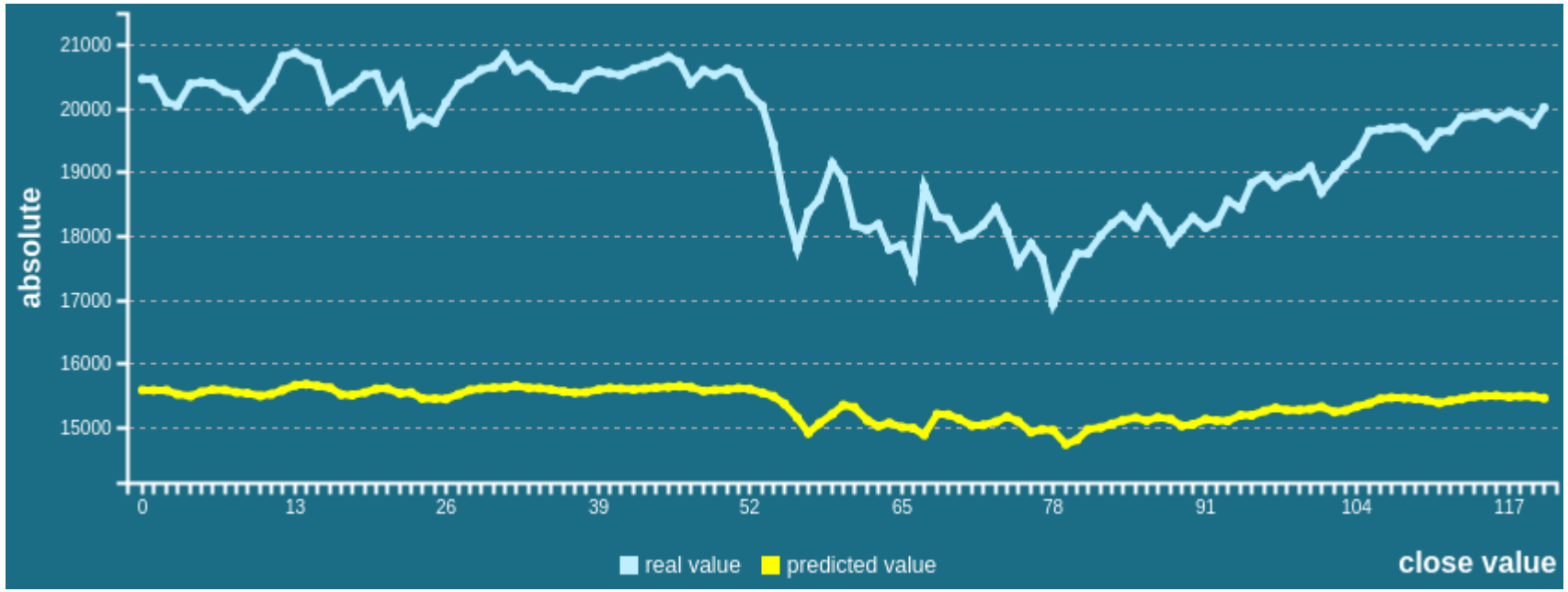
VIII. Analyse

- Letztes halbes Jahr – Dow Jones



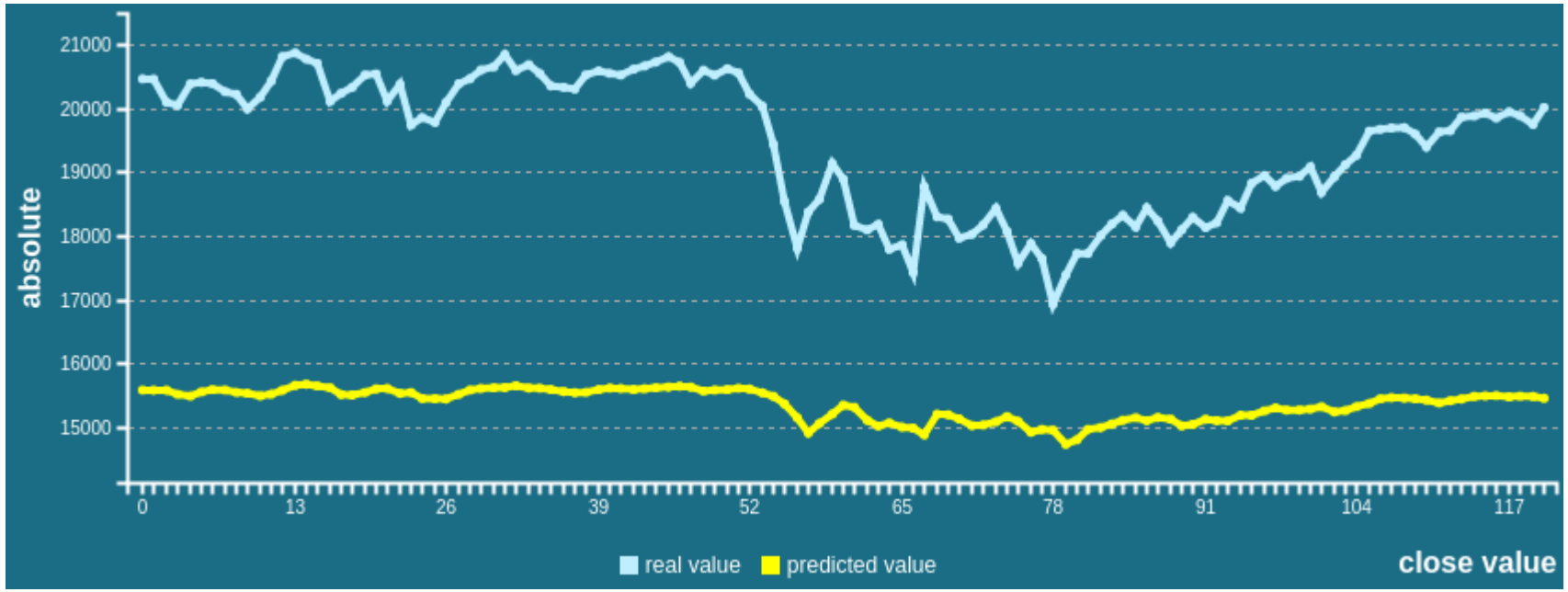
VIII. Analyse

- Letztes halbes Jahr – Nikkei 225



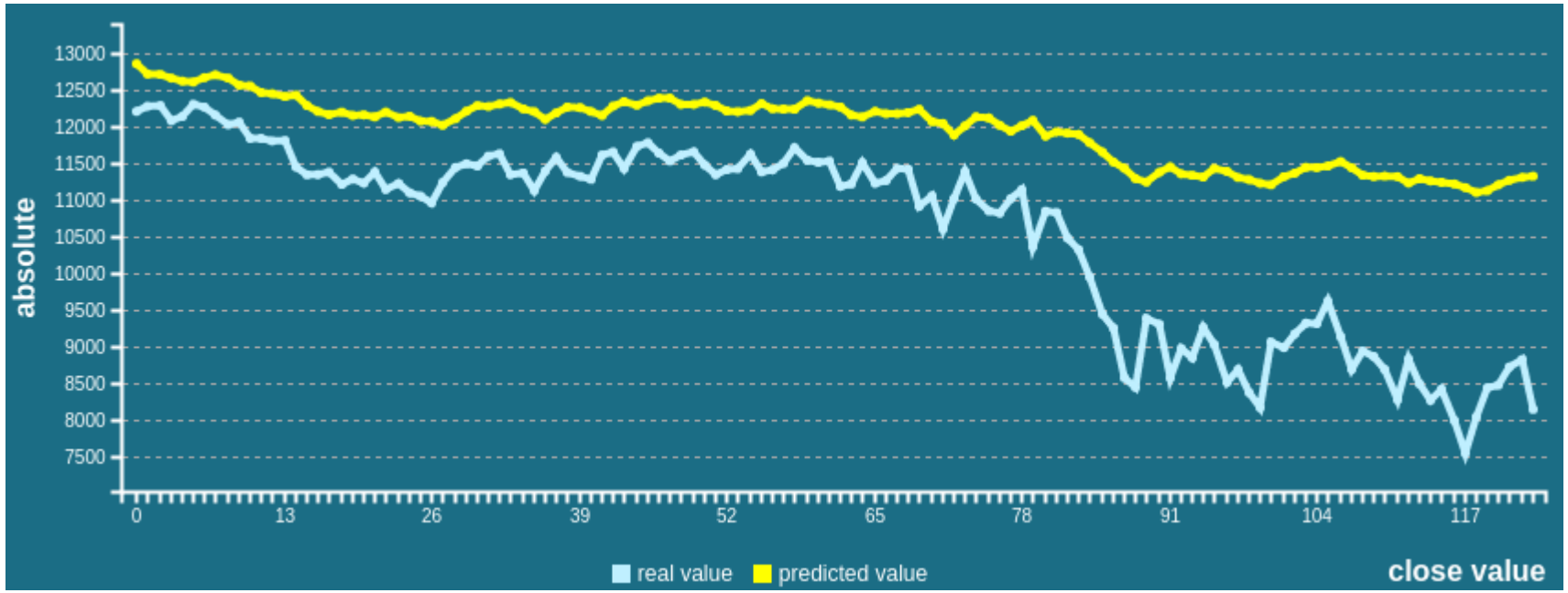
VIII. Analyse

- Börsencrash 2008 - DAX



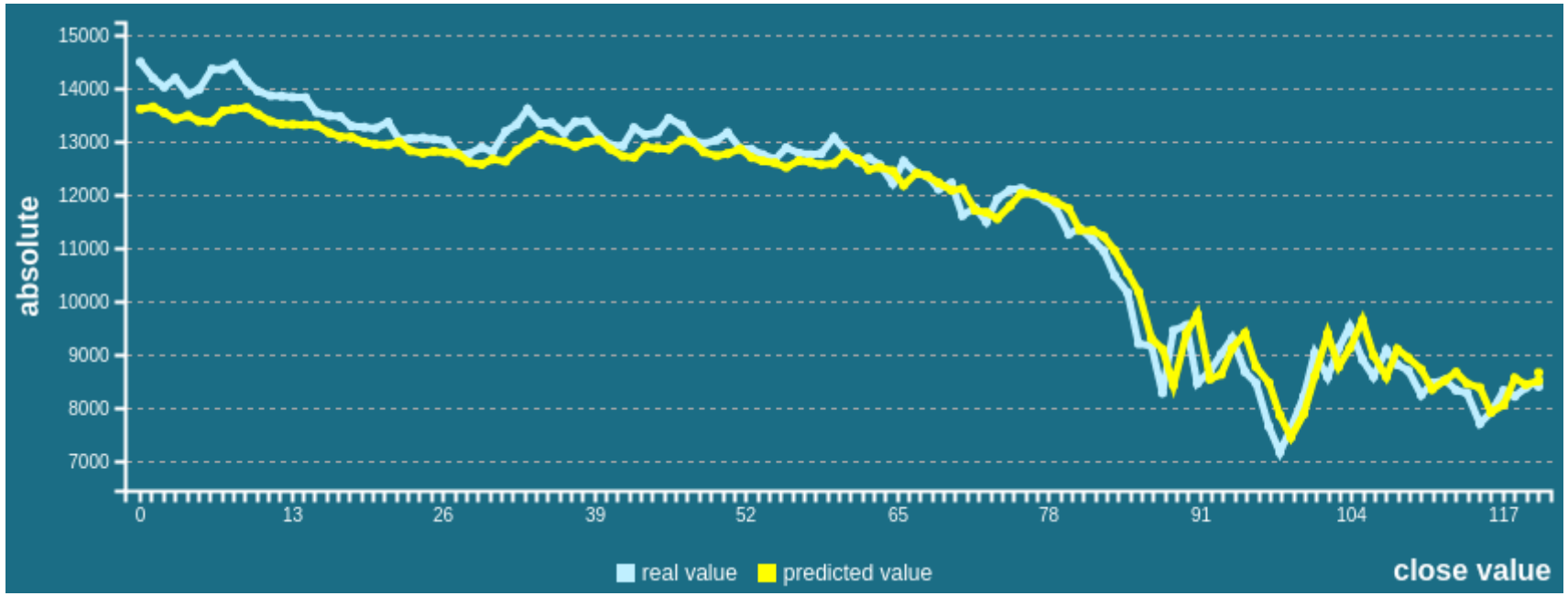
VIII. Analyse

- Börsencrash 2008 – Dow Jones



VIII. Analyse

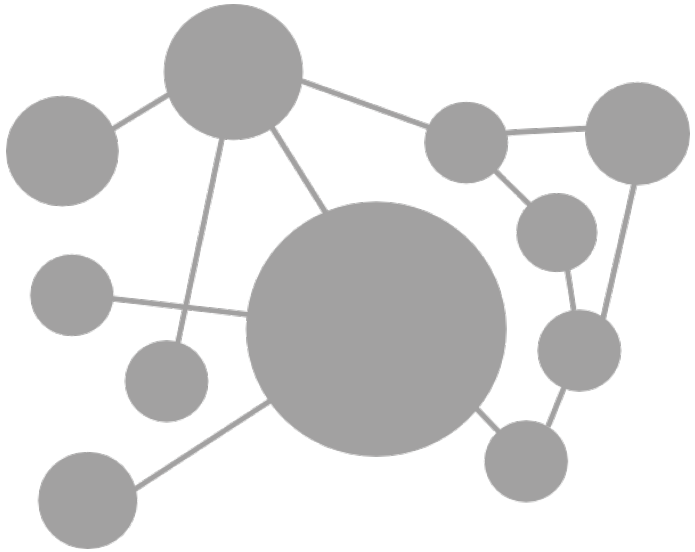
- Börsencrash 2008 – Nikkei 225



VIII. Analyse

- Fukushima 2011– Nikkei 225

Inhaltsverzeichnis



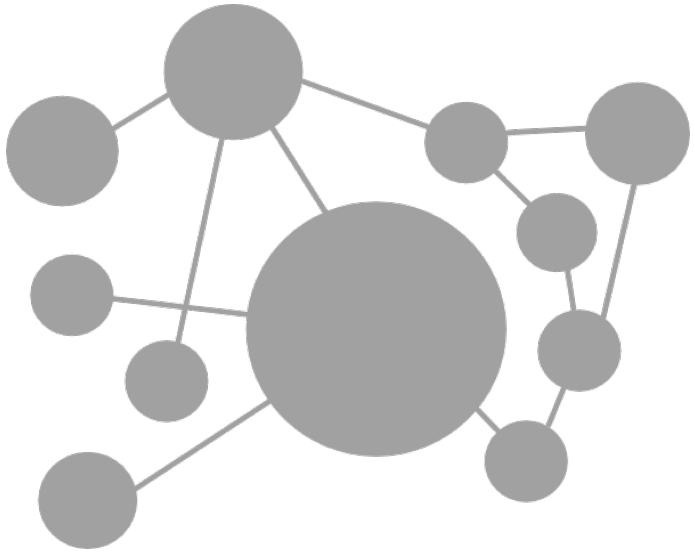
IX. Fazit

- Die Prognose von Börsenkursen ist prinzipiell möglich.
- Basismodell arbeitet nur mit linearen Zusammenhängen.
  - Abgeschottete Welt
  - Erweiterung durch nichtlineare Zusammenhänge möglich:
    - Leitzins
    - Weltereignisse
    - Kurse anderer Börsen
- Prognosen mit neuronalen Netzen sind umstritten:
  - Befürworter: nichtlineare Muster erkennen wertvoll.
  - Kritiker: KNN denkt wie ein Mensch ➡ macht die gleichen Fehler.

IX. Fazit

- KNN als Ergänzung sinnvoll, nicht als alleiniges Prognoseintrument.

- Anwendungen dieser Art bereits zahlreich auf dem Markt vorhanden:
  - Neuroshell Trader
  - Altredo
  - …



Präsentationsende…
…Fragen & Diskussion