

Neuronale Netze/ Soft Computing

Teil 3

**BiTS, Wintersemester 2004/2005
Dr. Stefan Kooths**

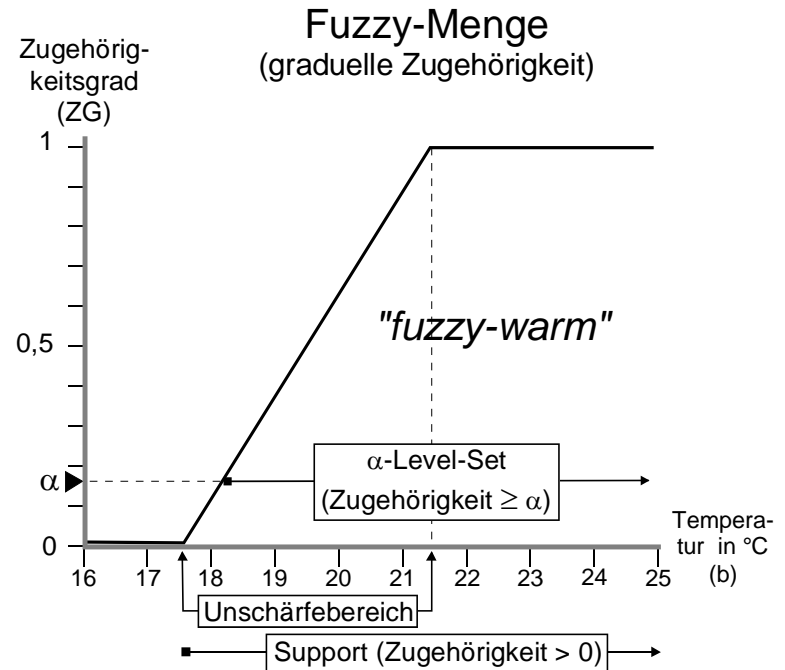
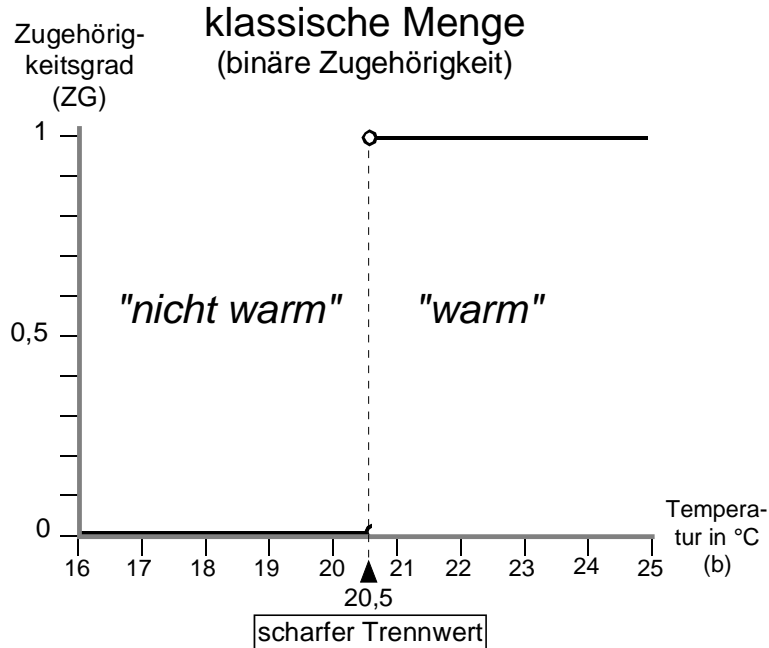
Gliederung

1. Einführung und Einordnung
2. Neuronale Netze 1: Grundlagen
3. Neuronale Netze 2: Konzeption und Anwendung
- 4. Neuro-Fuzzy-Systeme**
5. Genetische Algorithmen
6. Zusammenfassung und Ausblick

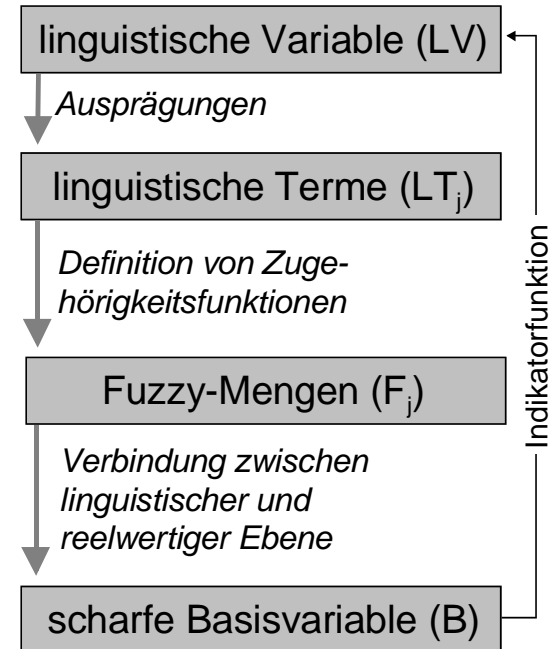
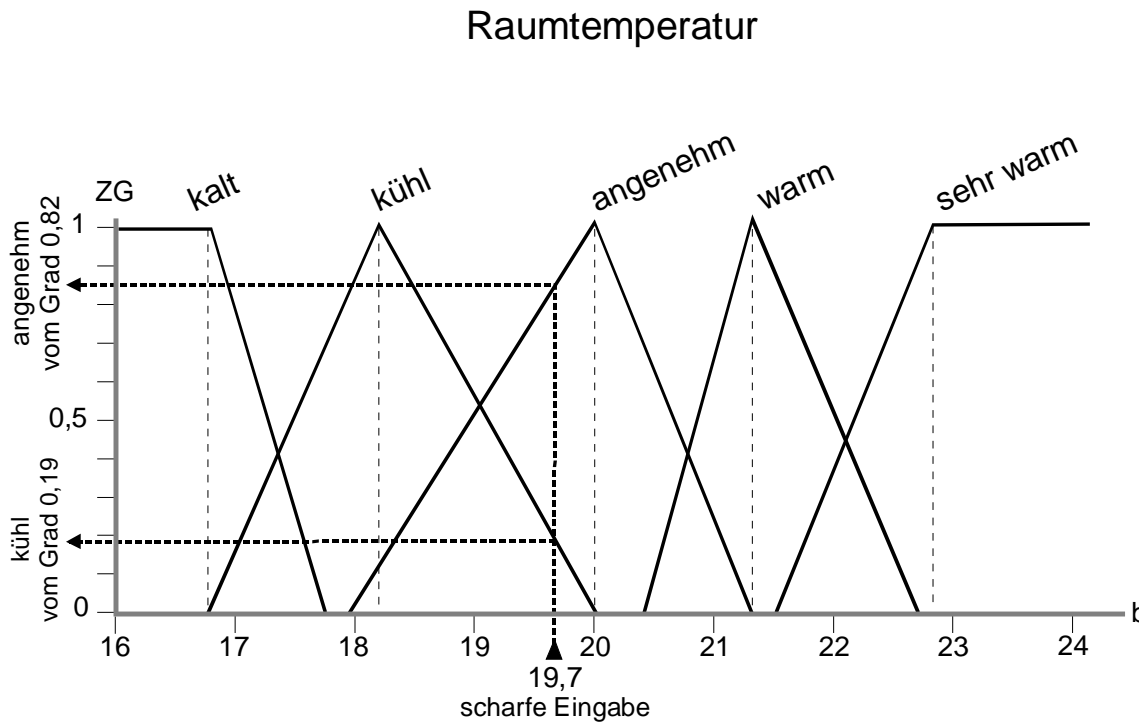
Vorgehensweise

- Grundlagen der Fuzzy Logik
- Transformation eines Fuzzy Systems in ein BP-Netz
- Modifizierter BP-Algorithmus

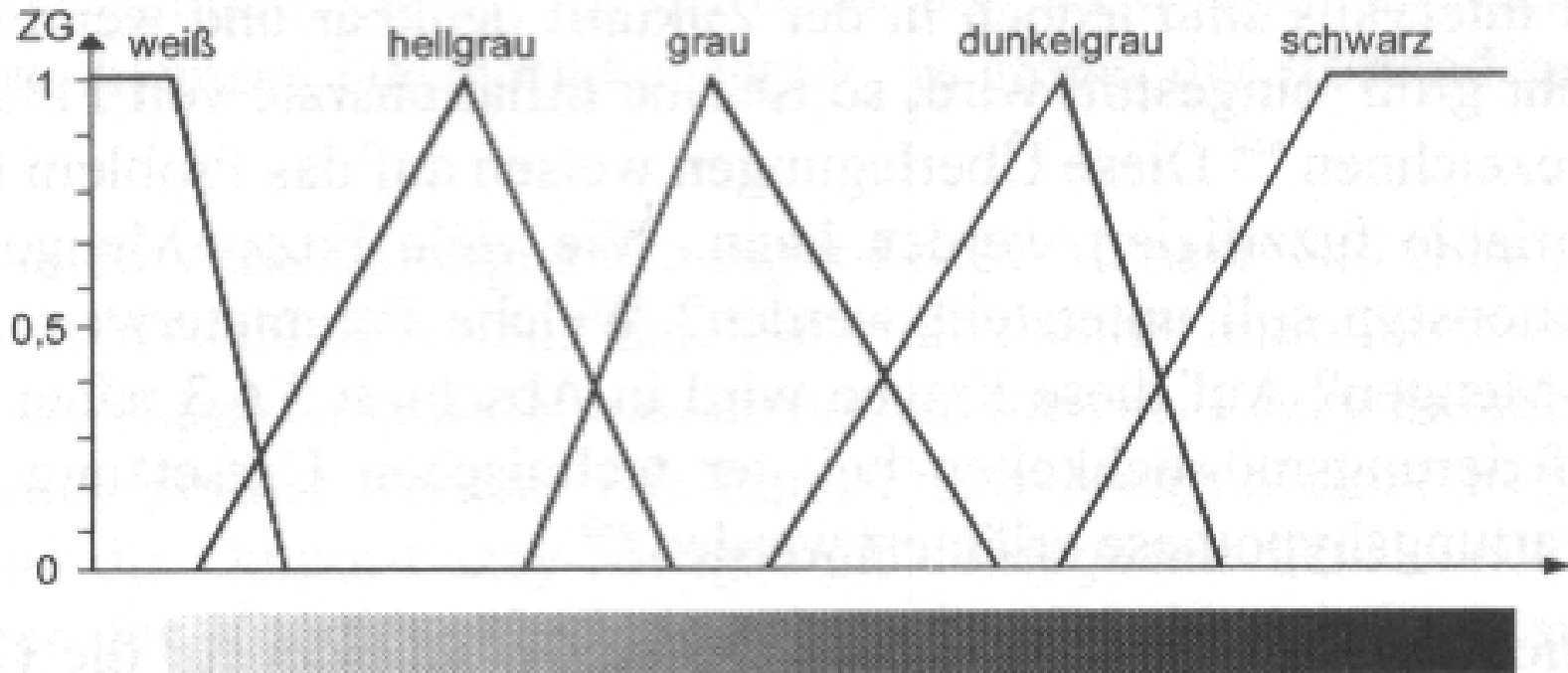
Klassische Mengen und Fuzzy Mengen



Fuzzifizierung

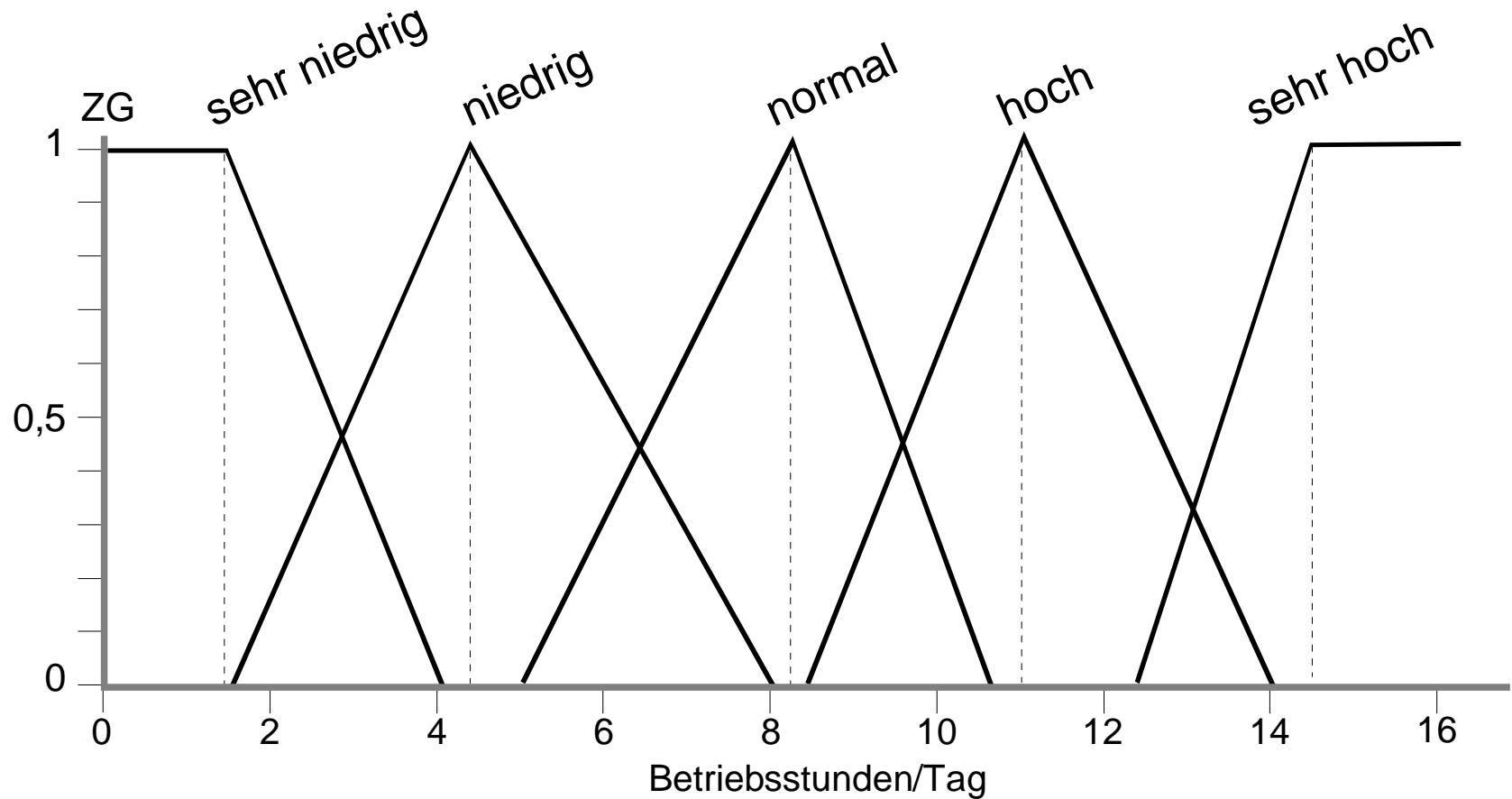


Fuzzifizierung: Schwarz-Weiß-Grau-Beispiel



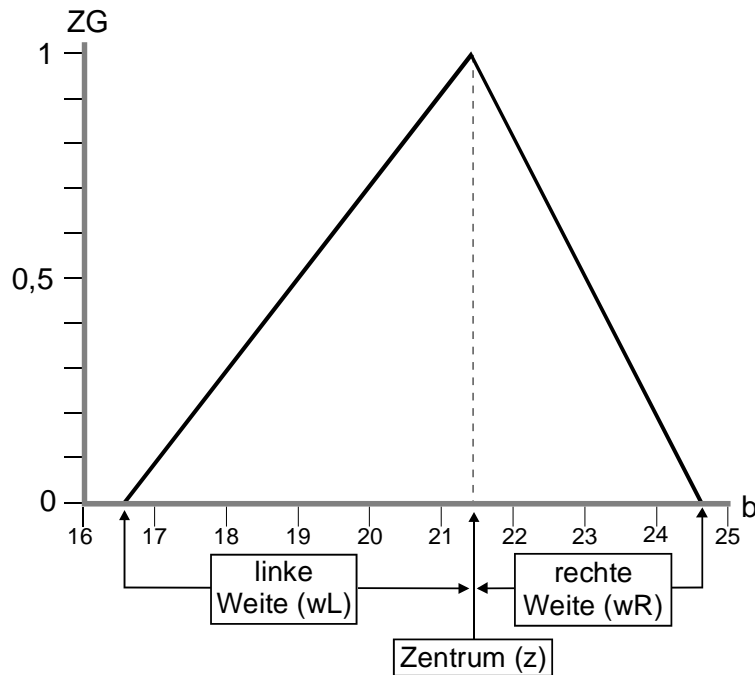
Fuzzifizierung: Ökonomisches Beispiel

Auslastungsgrad einer Maschine

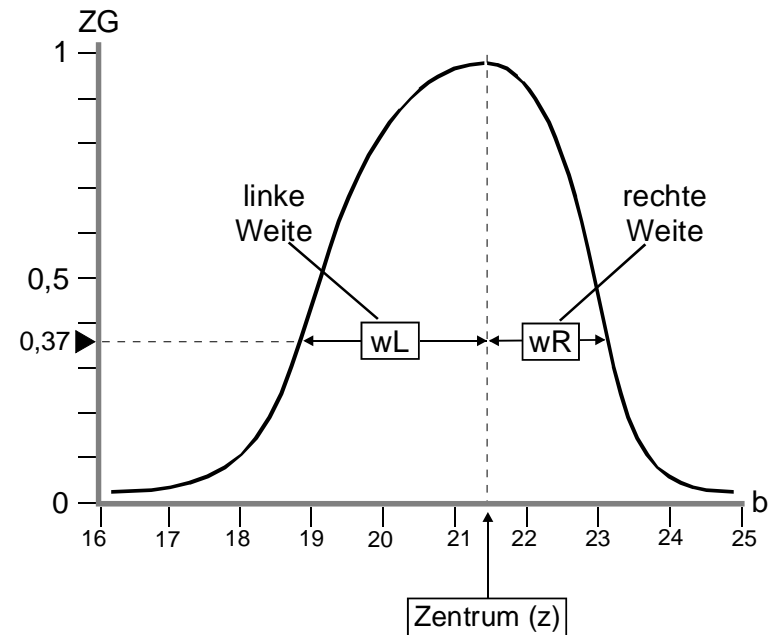


Fuzzy-Mengen: Grundformen

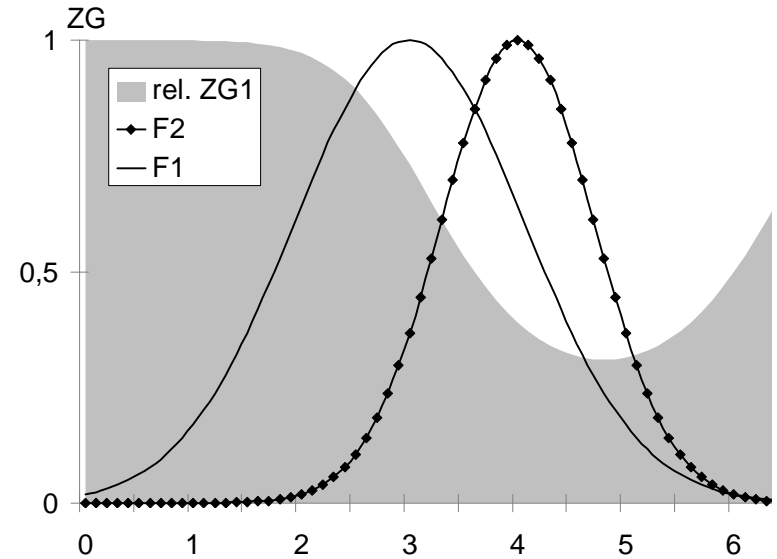
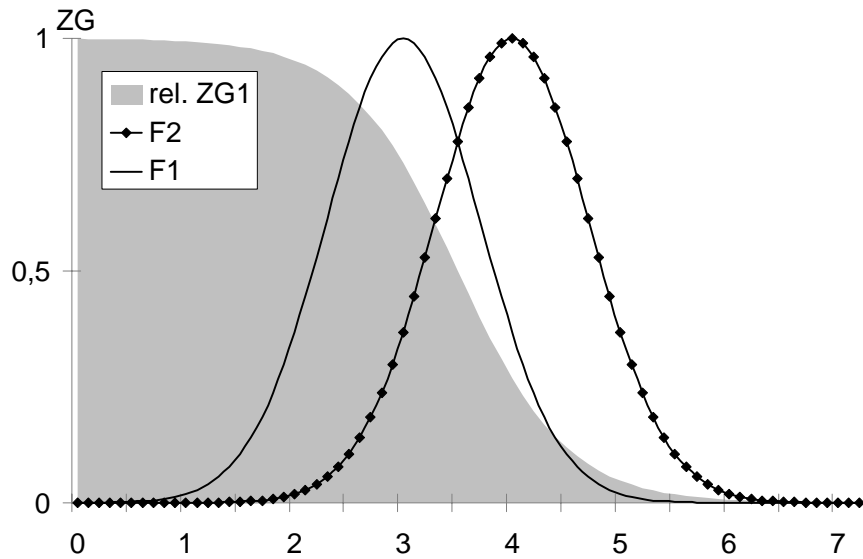
trianguläre Form



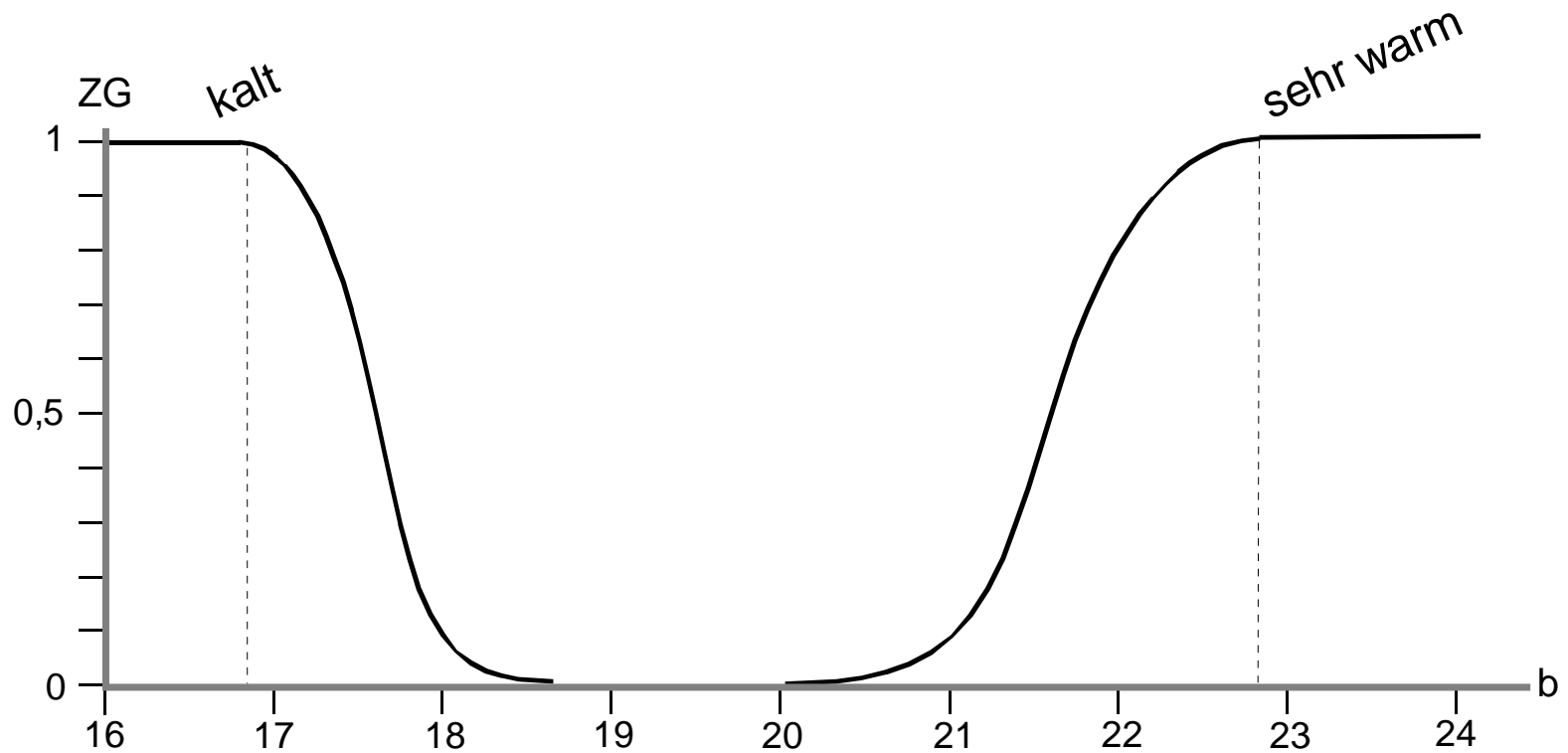
Gauß-Form



Gaußproblem: Dominanzeffekt der großen Weite

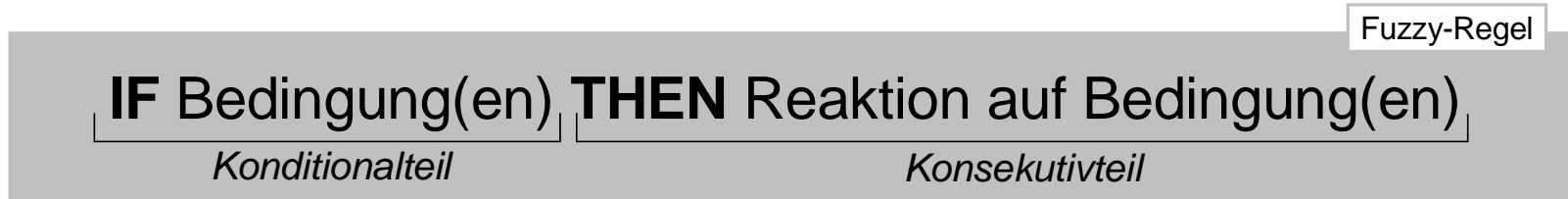


Randsättigung als Lösung des Gaußproblems

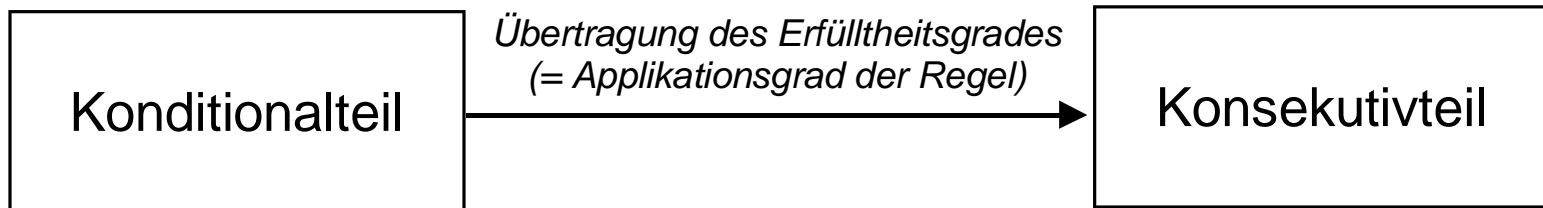


Fuzzy-Regeln: Aufbau und Inferenzprinzip

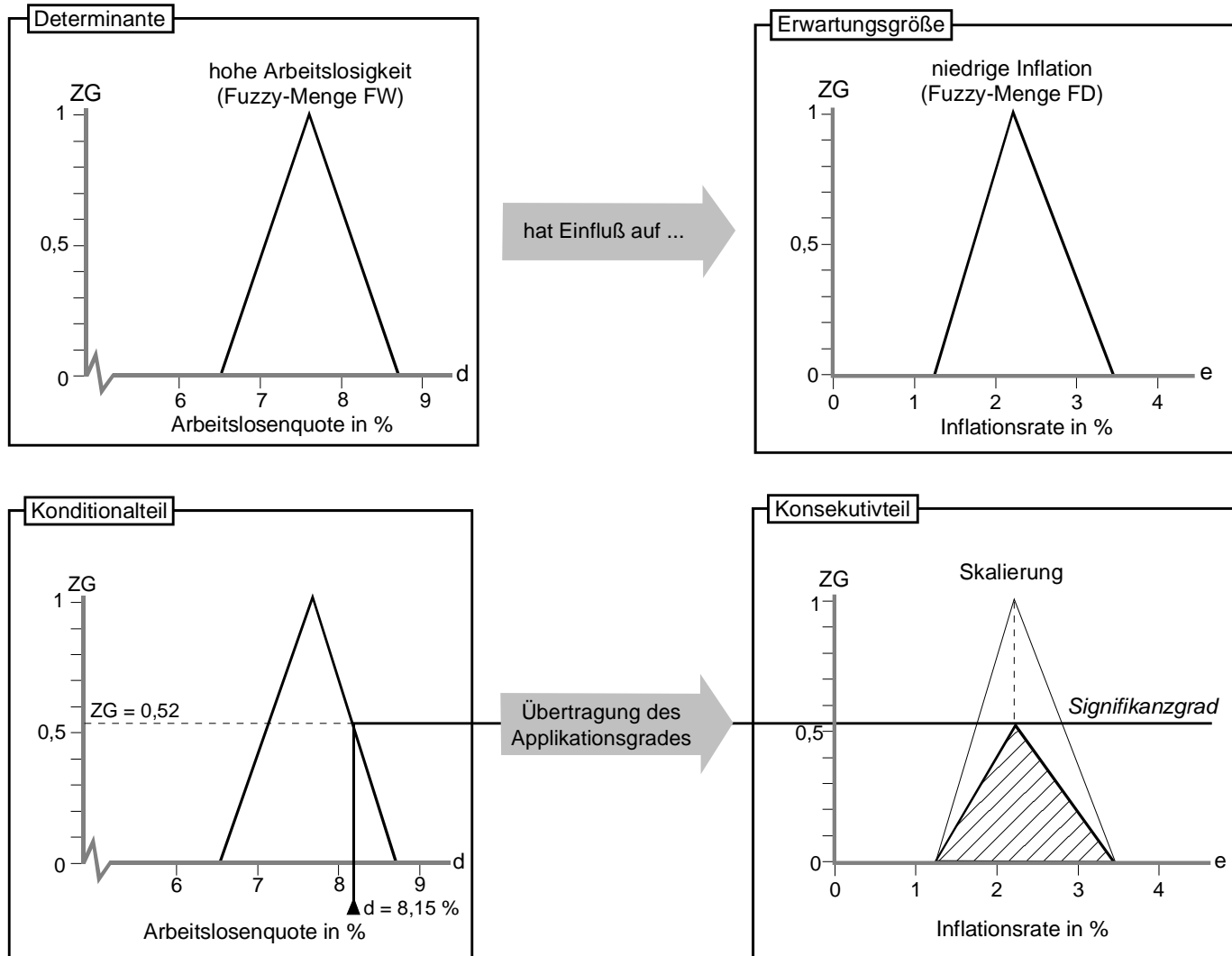
- Stilisierte Form einer Fuzzy-Regel



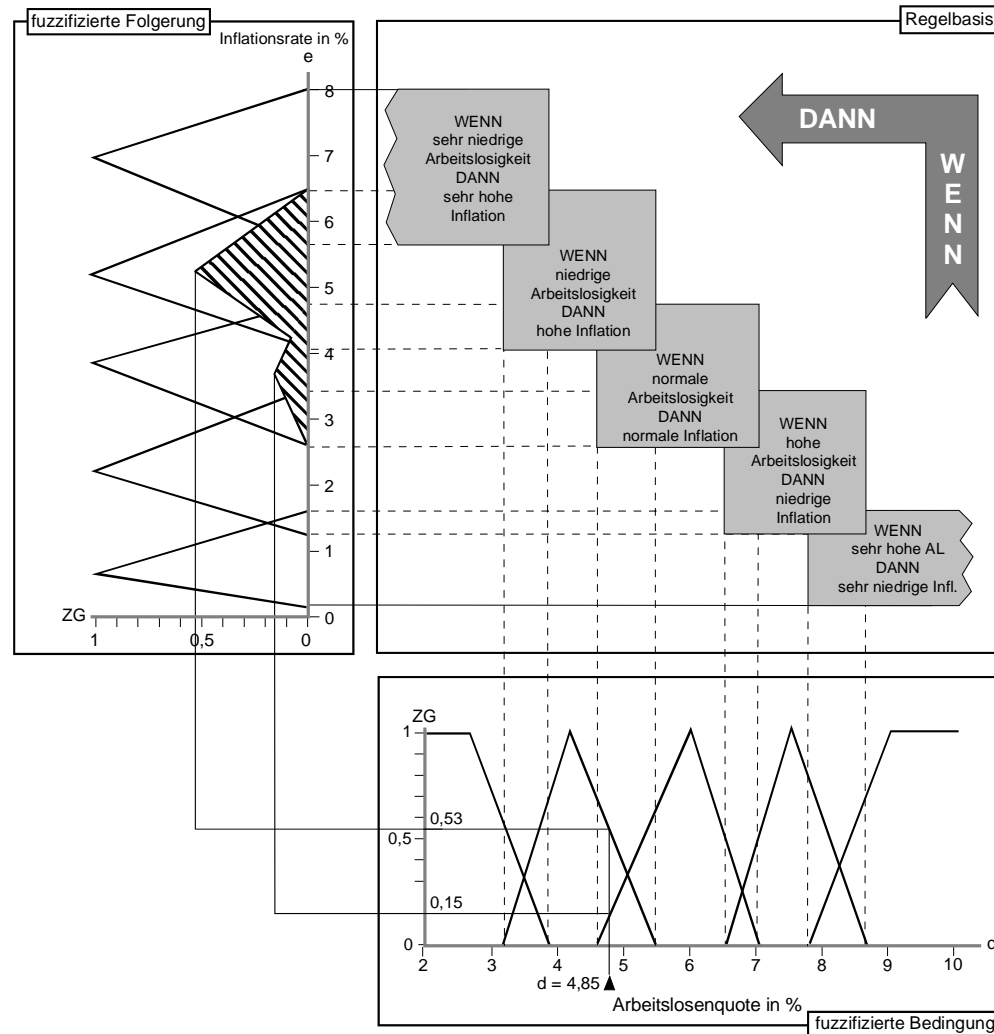
- Inferenzprinzip



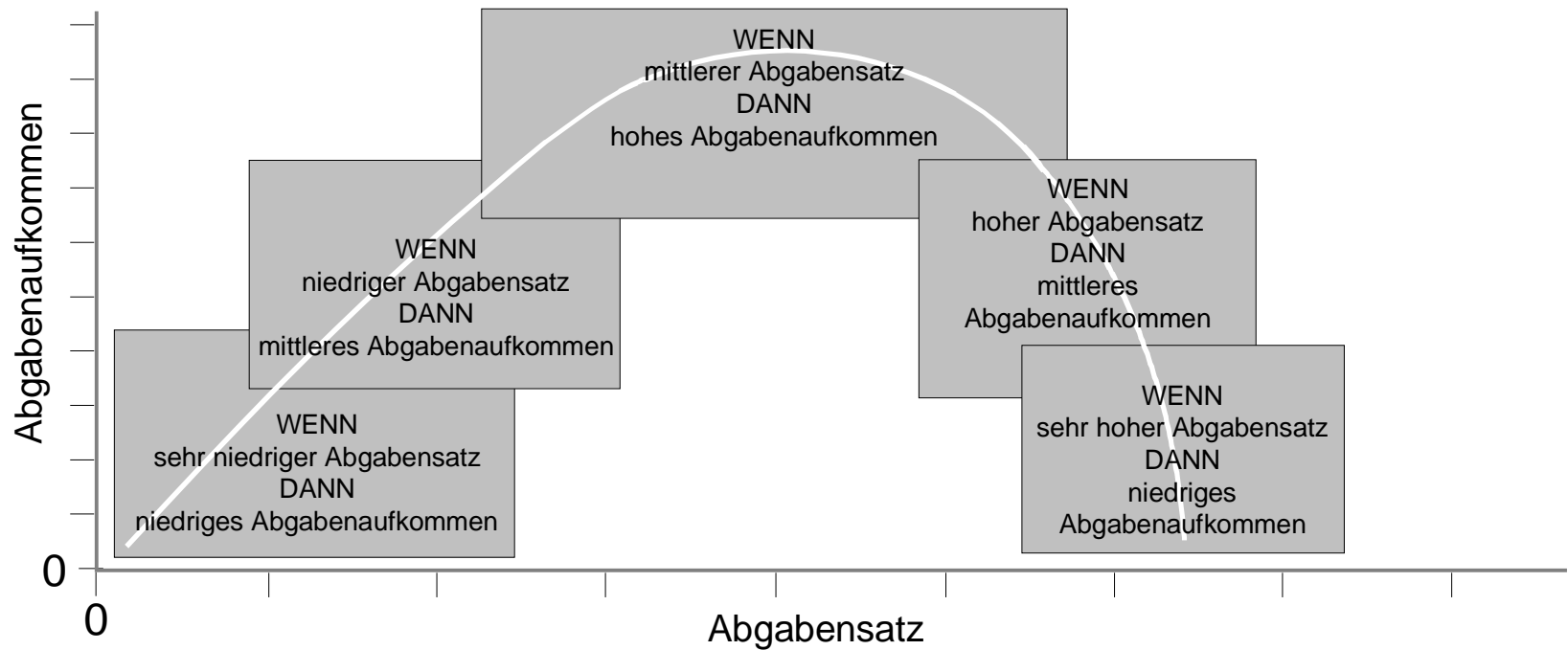
Fuzzy-Inferenz: Beispiel Inflationsprognosen



Fuzzy-Regelbasis: Inflationsprognose



Fuzzy-Regelbasis: Laffer-Kurve (Abschätzung des Abgabenaufkommens)



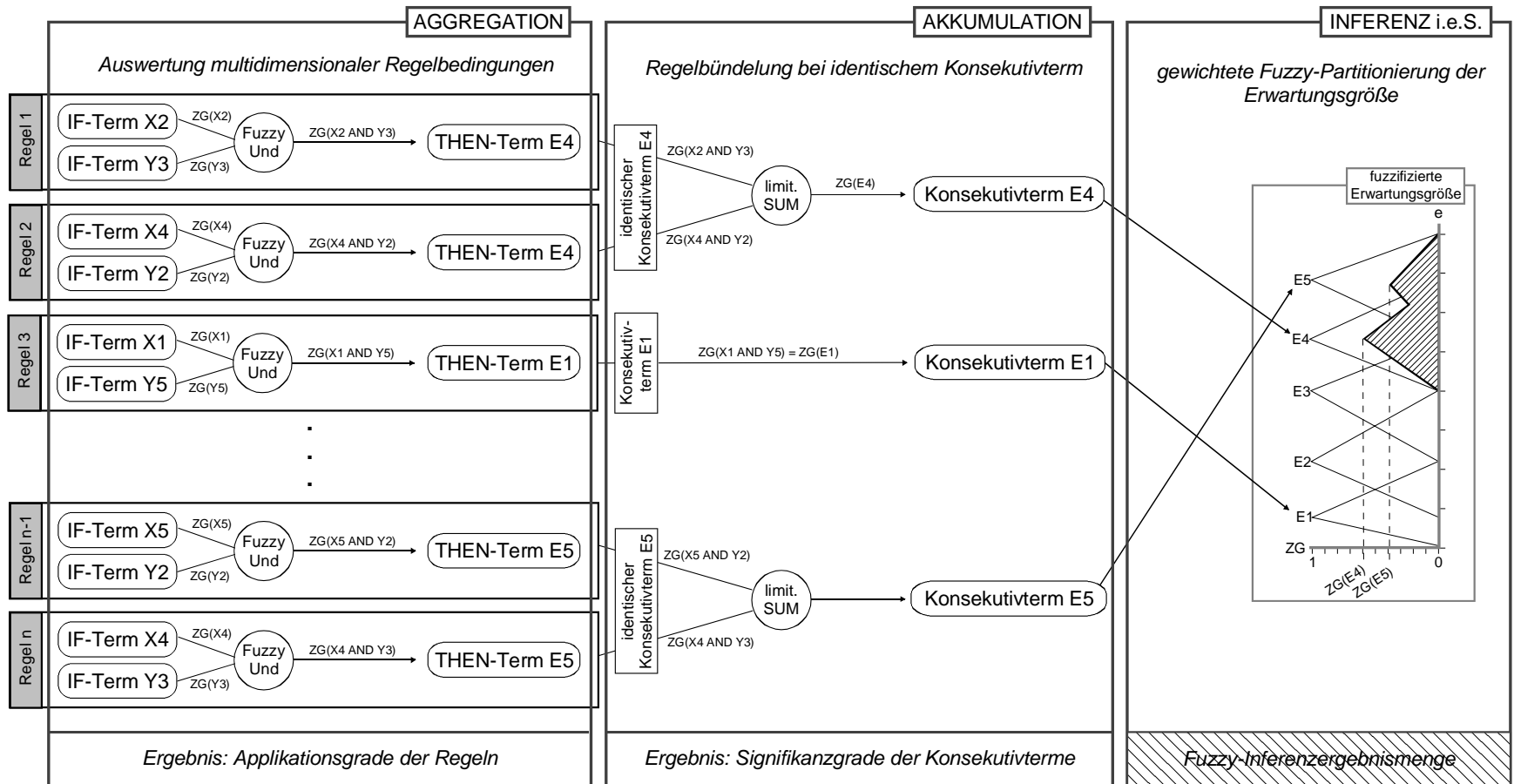
Fuzzy-Inferenz: Akkumulation

- verschiedene Regeln mit identischen Konsektivteil
- Signifikanzgrad = $\min(1, \text{Summe der Applikationsgrade})$

Fuzzy-Inferenz: Aggregation

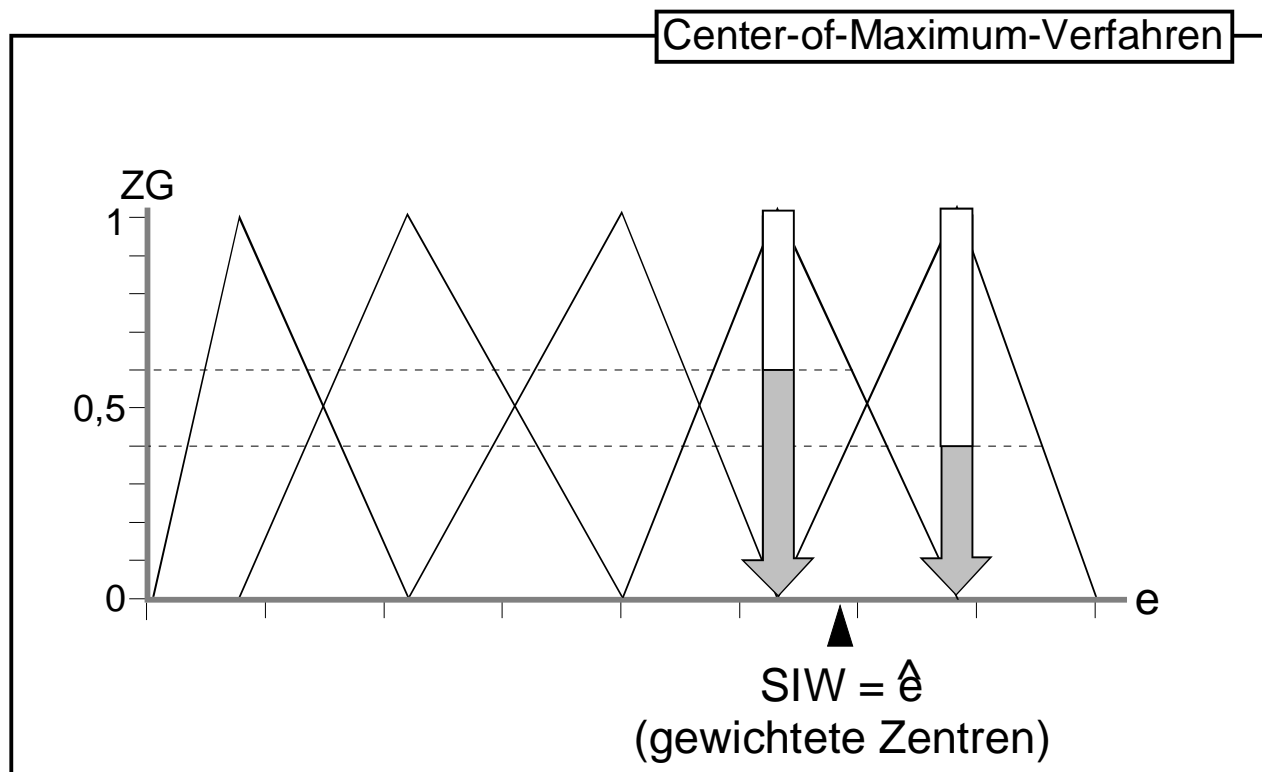
- Auswertung mehrdimensionaler Fuzzy-Regeln
- Fuzzy-Und
 - Minimum-Operator
 - Produkt-Operator

Fuzzy-Regelauswertung im Überblick



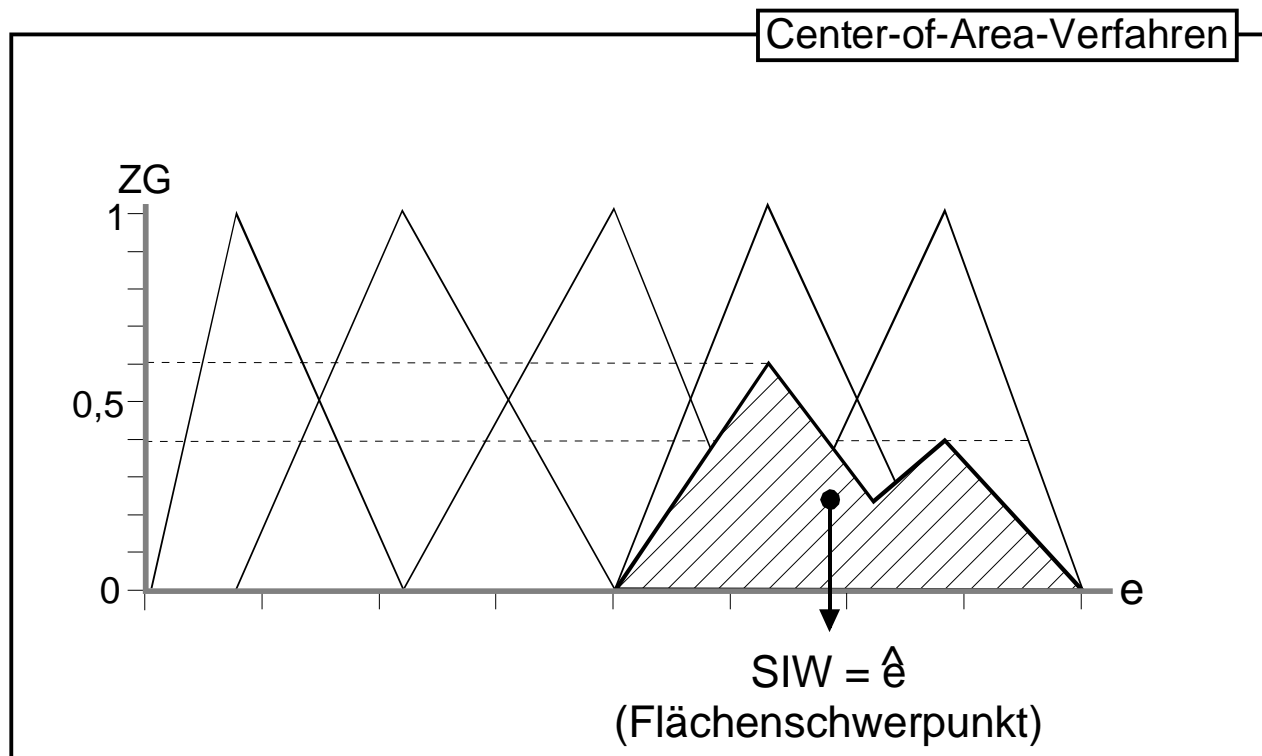
Defuzzifizierung

- Suche nach dem repräsentativen scharfen Wert der Outputbasisvariablen
- Center-of-Maximum-Verfahren

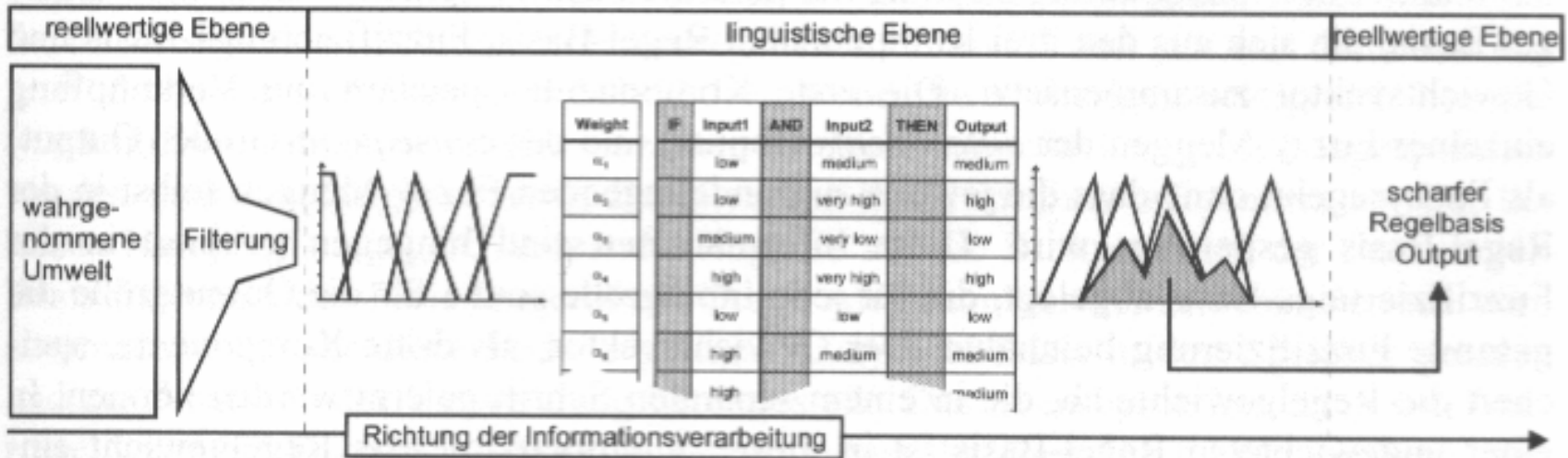


Defuzzifizierung

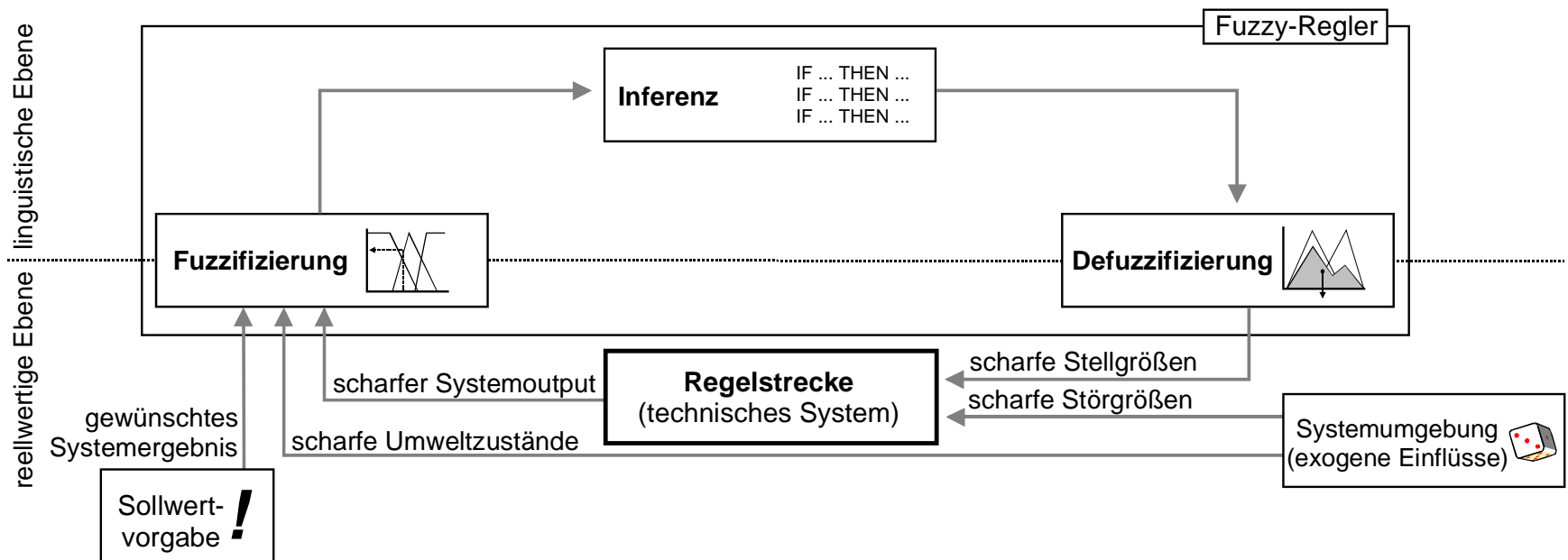
- Suche nach dem repräsentativem scharfen Wert der Outputbasisvariablen
- Center-of-Area-Verfahren



Fuzzy-Inferenzprozess



Fuzzy-Regelung (Fuzzy-Control)



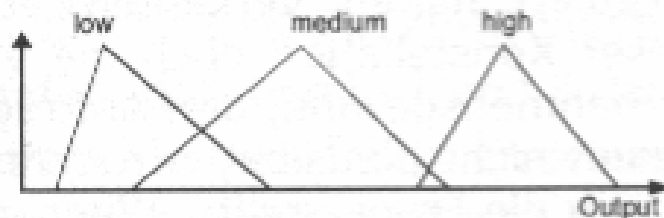
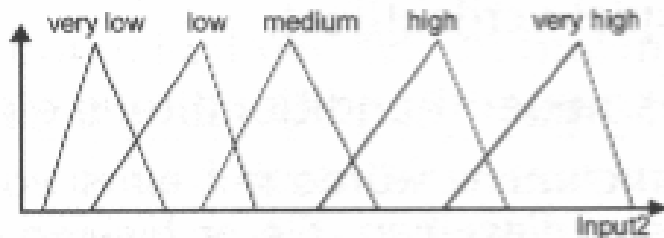
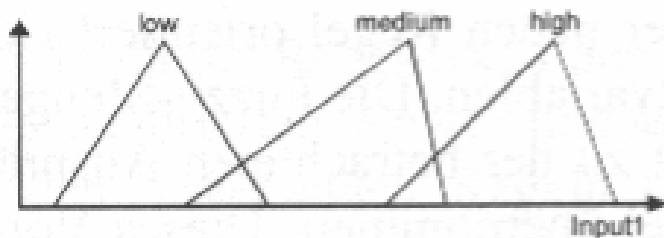
Wissensbasis eines Regelsystems

Wissensbasis = Fuzzregel-Basis (FRB)

Fuzzifizierungs-Basis ①

Gewichtsvektor ②

Regel-Basis ③



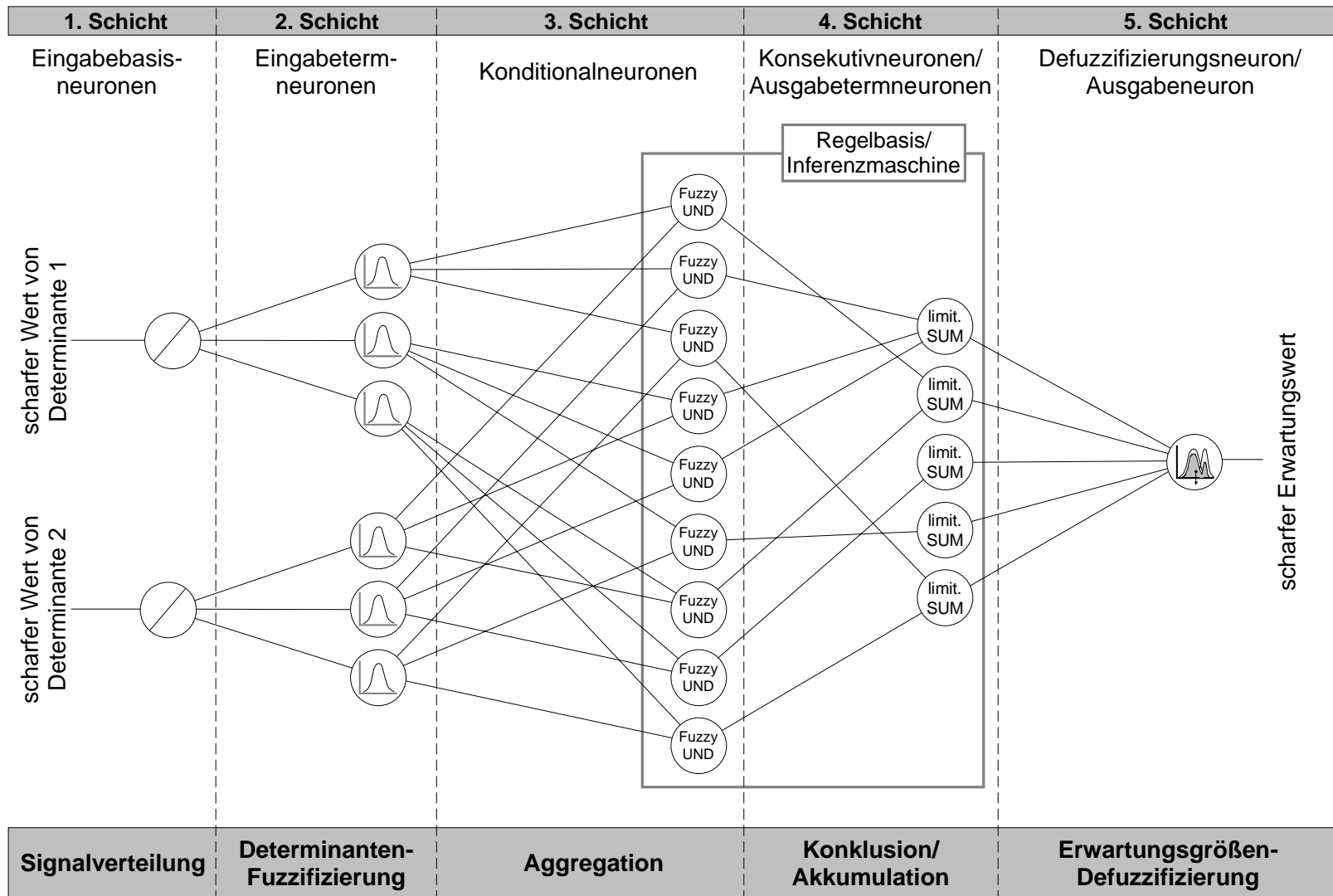
Weight
α_1
α_2
α_3
α_4
α_5
α_6
α_7
α_8
α_9
α_{10}

IF	Input1	AND	Input2	THEN	Output
	low		medium		medium
	low		very high		high
	medium		very low		low
	high		very high		high
	low		low		low
	high		medium		medium
	high		very low		medium
	medium		low		low
	medium		medium		medium
	high		medium		medium

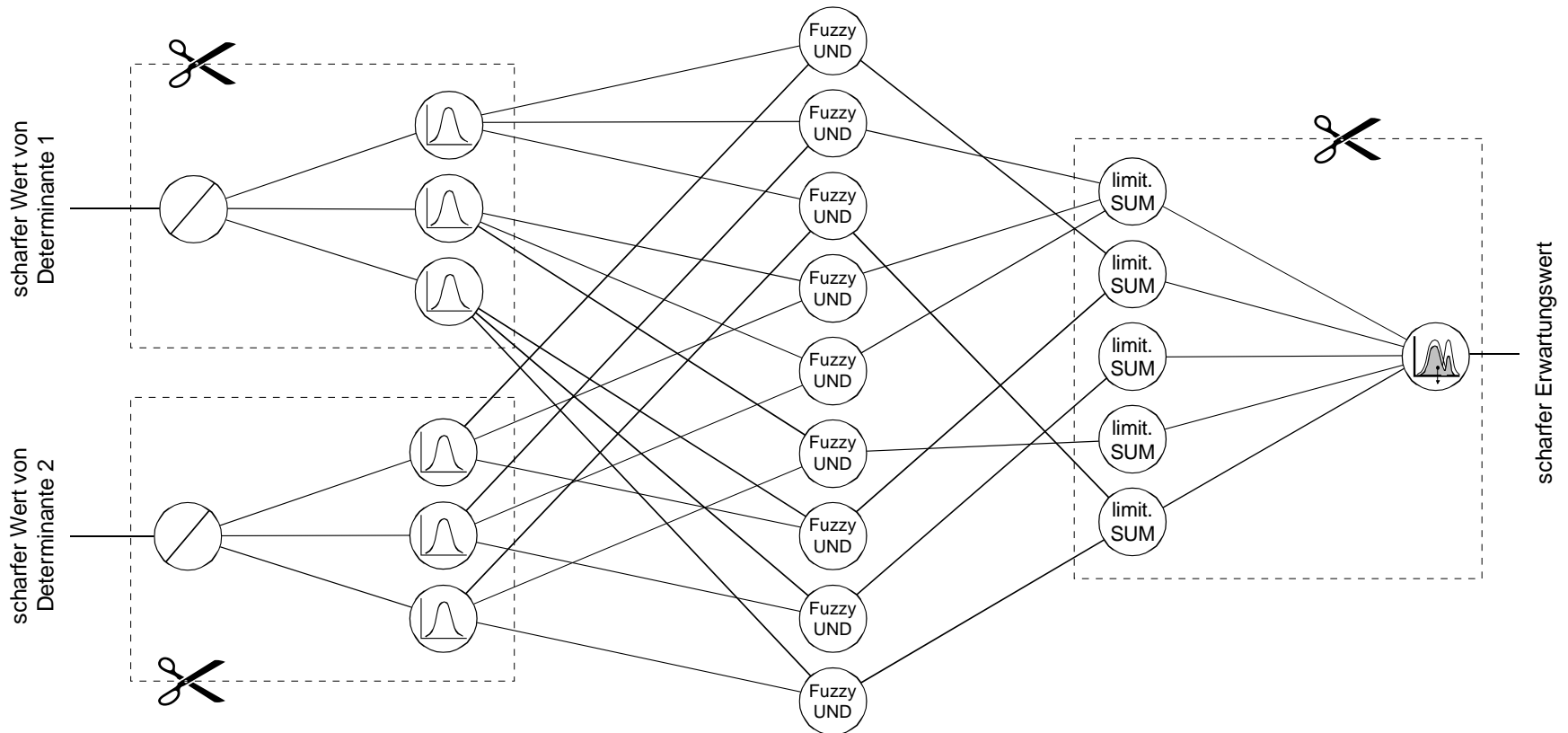
Motivation für Neuro-Fuzzy-Systeme

Fuzzy-Systeme	Neuronale Netze
<p><i>Nachteile:</i></p> <ul style="list-style-type: none">• nicht lernfähig• A-priori-Wissen erforderlich <p><i>Vorteile:</i></p> <ul style="list-style-type: none">• Umgang mit Unschärfe in den Aussagen• explizite Wissensrepräsentation (Regeln)• A-priori-Wissen nutzbar• Architektur vorgegeben, einfaches Design	<p><i>Vorteile:</i></p> <ul style="list-style-type: none">• Umgang mit Unschärfe in den Daten• verschiedene Lernalgorithmen• kein A-priori-Wissen erforderlich <p><i>Nachteile:</i></p> <ul style="list-style-type: none">• Black-Box-Charakter• kein A-priori-Wissen nutzbar• heuristische Architektur• keine Garantie für Konvergenz des Lernvorgangs

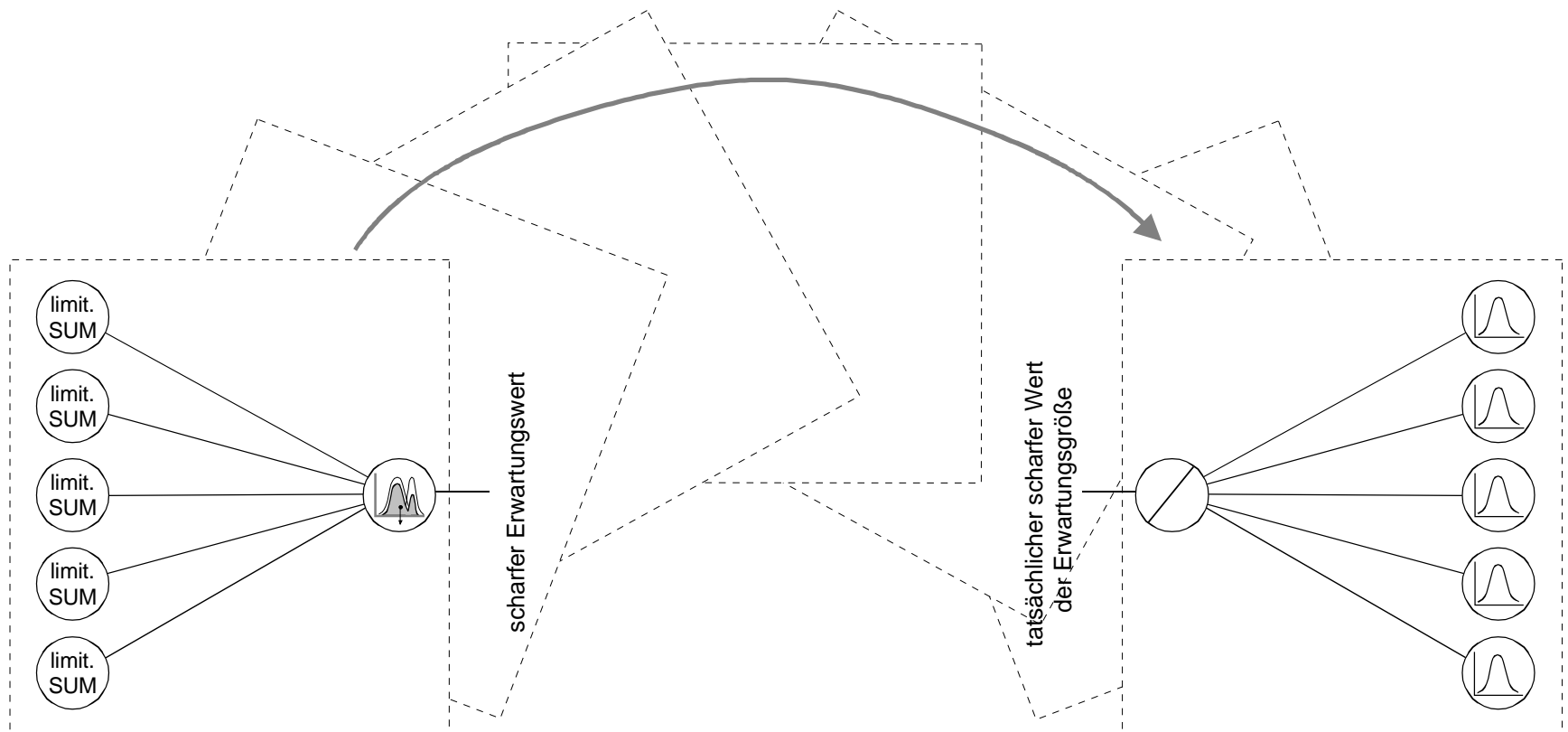
Hybrides Neuro-Fuzzy-System



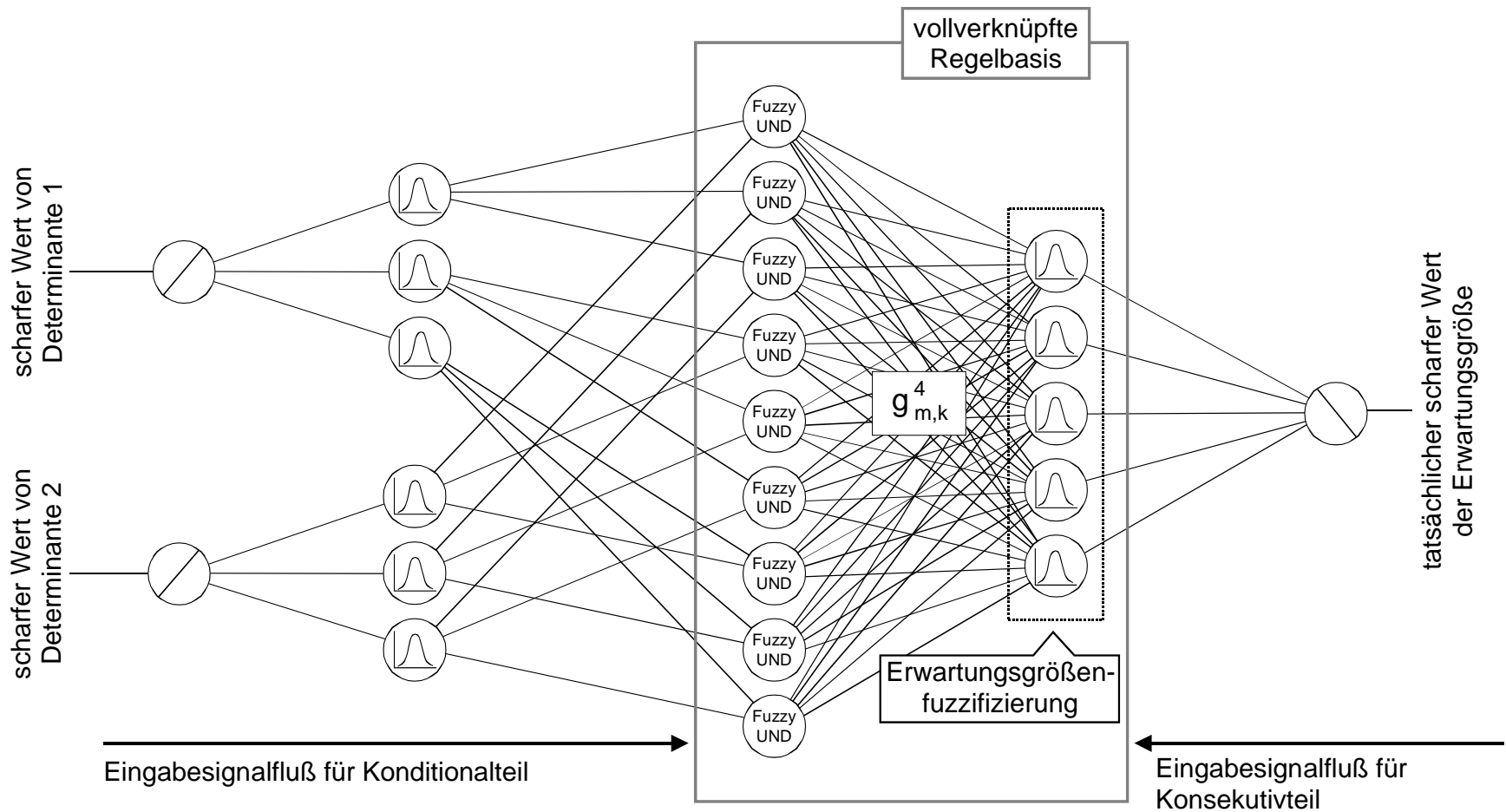
Variablenfuzzifizierung: Subnetz-Extraktion



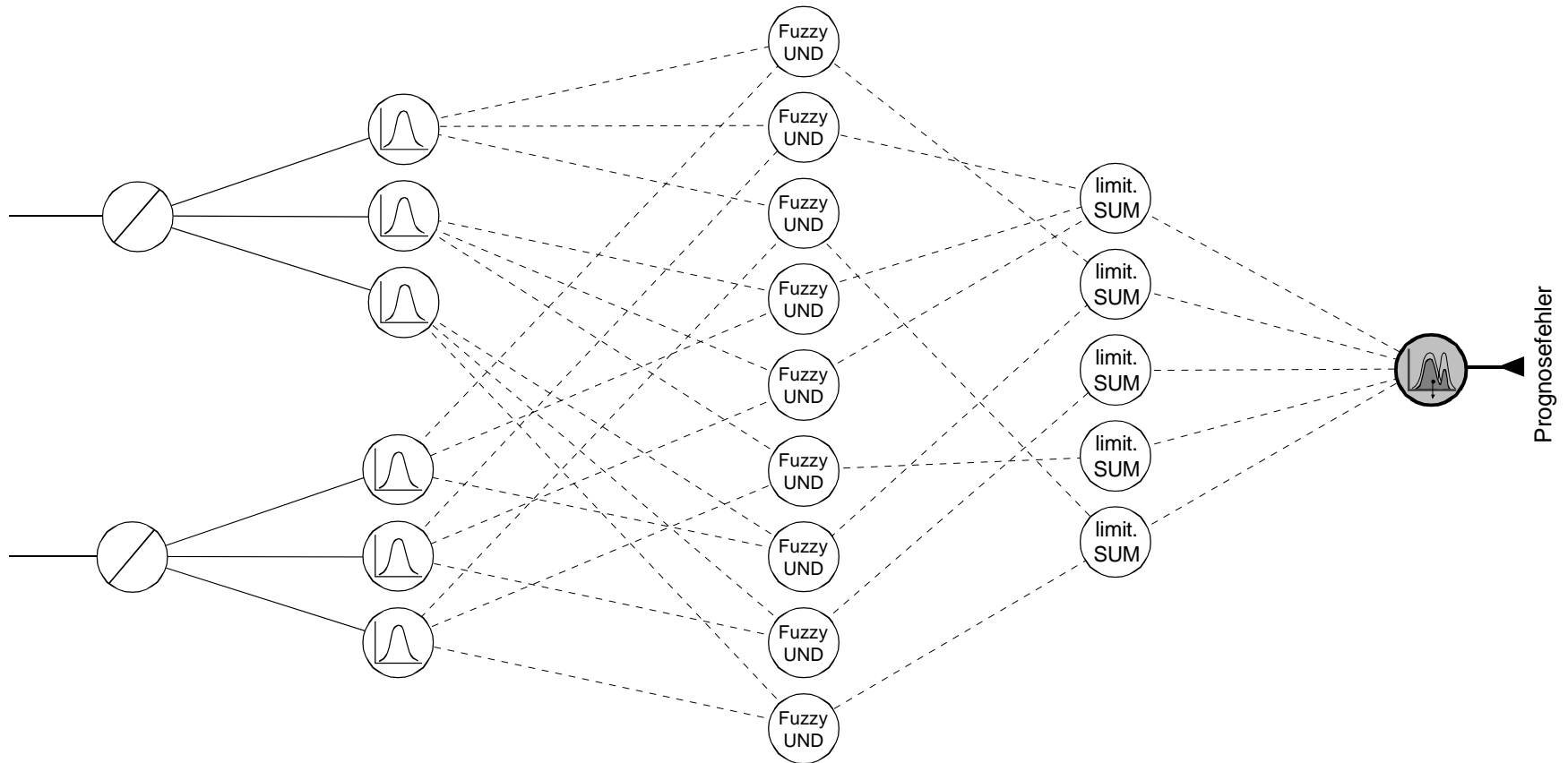
Umbau des Subnetzes für die Outputfuzzifizierung



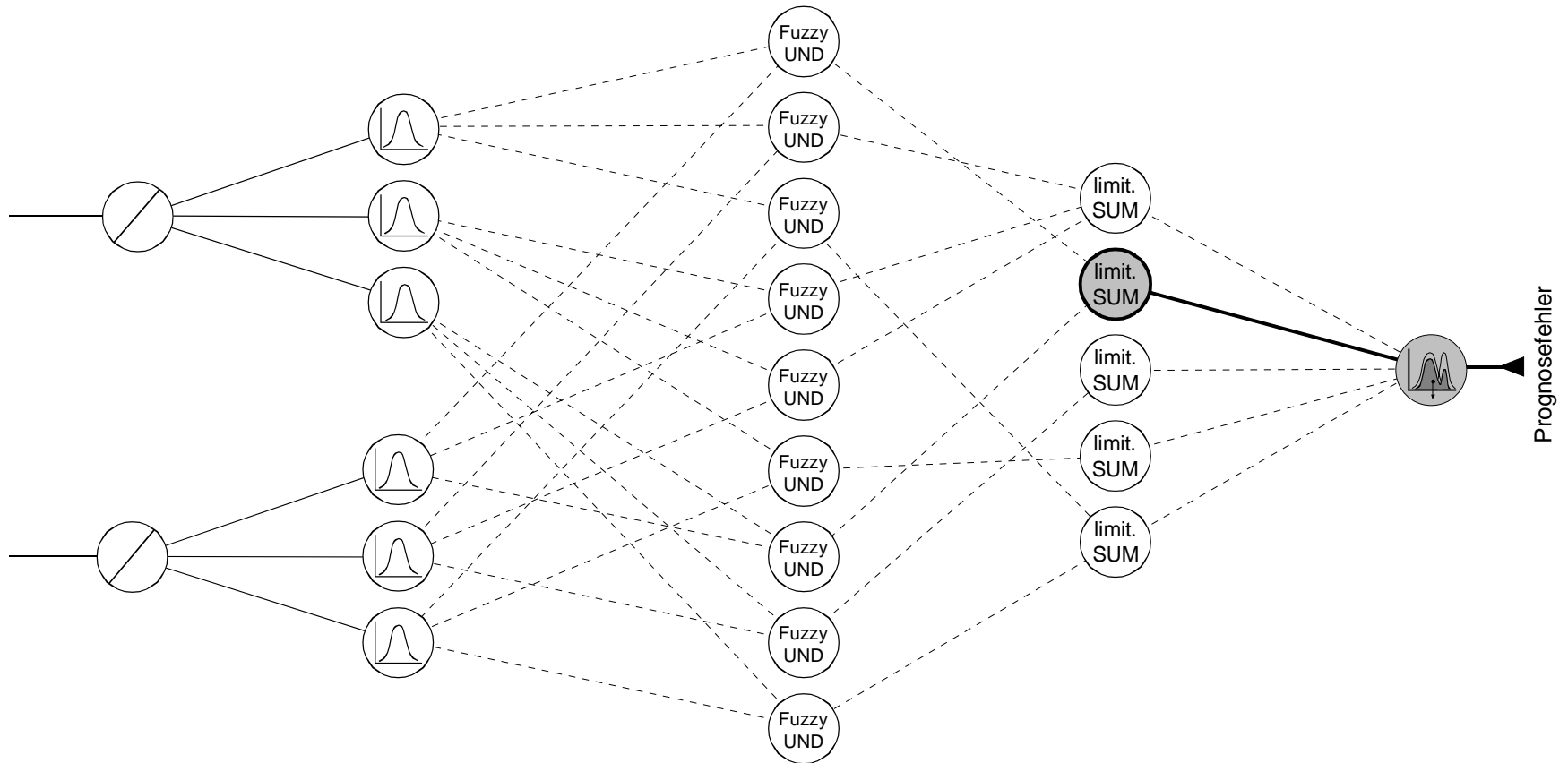
Wettbewerbliches Regeltraining



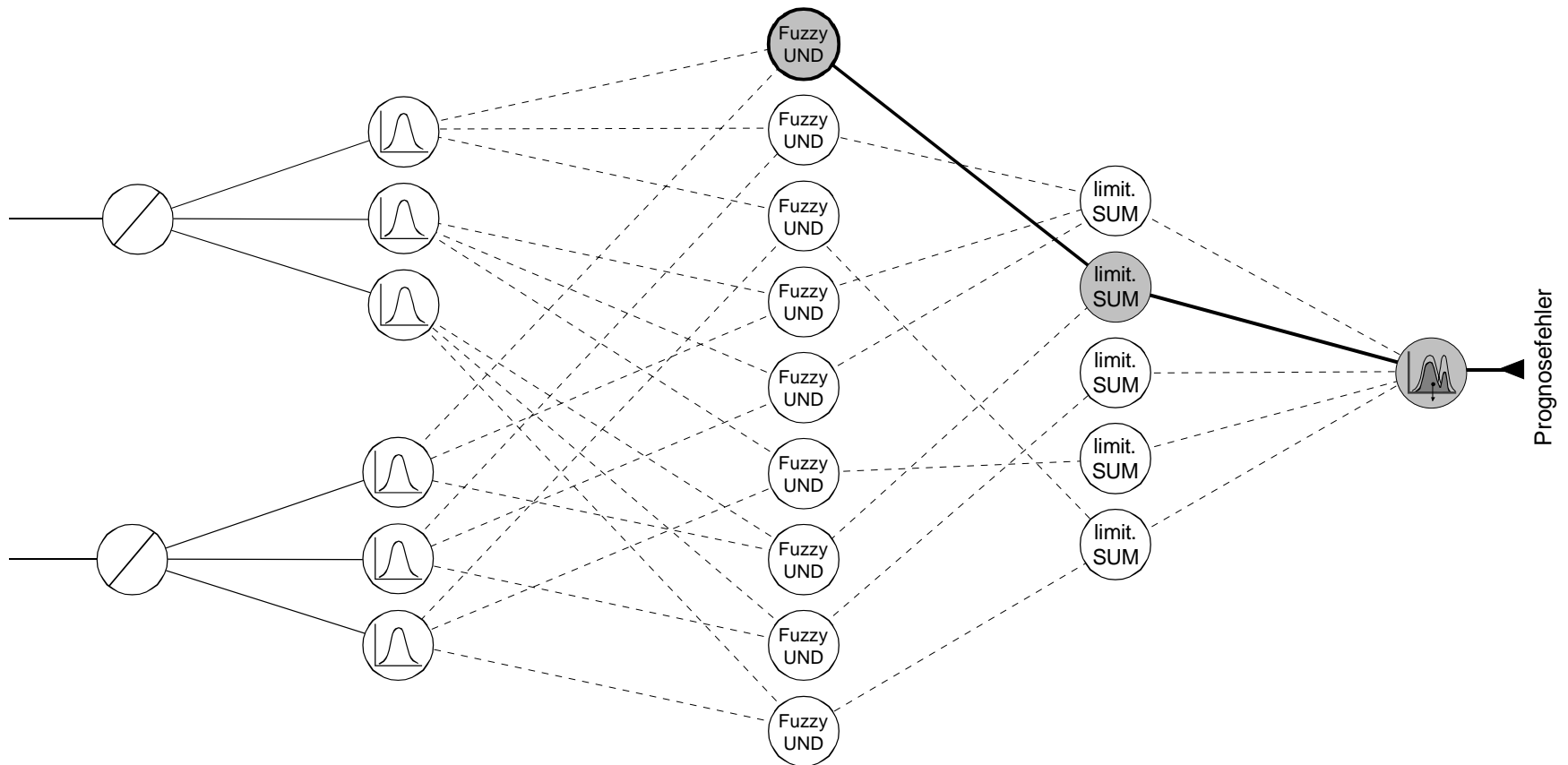
BP-Training des Gesamtsystems 1 (Rückwalzung des Fehlersignals)



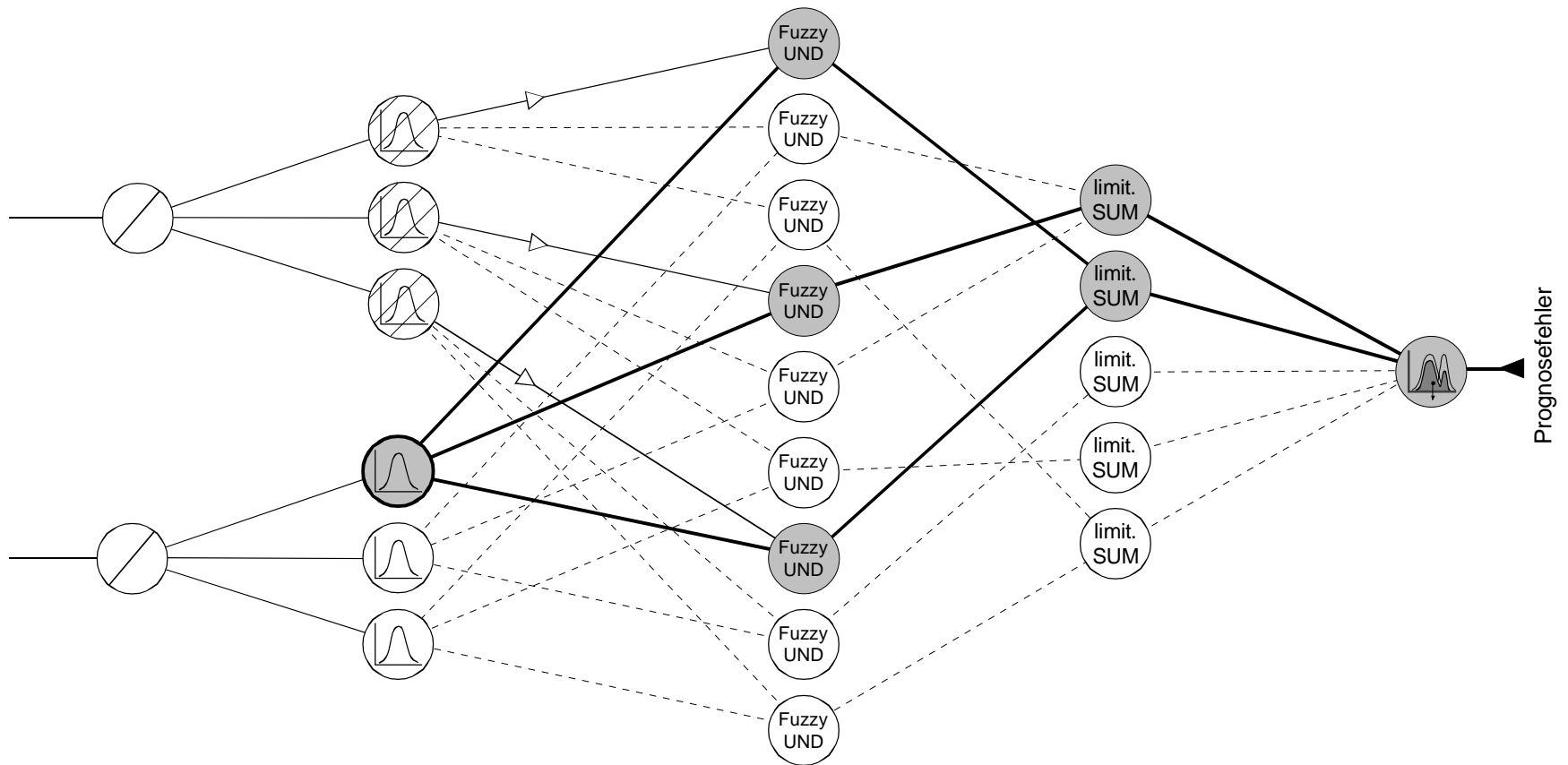
BP-Training des Gesamtsystems 2 (Rückwalzung des Fehlersignals)



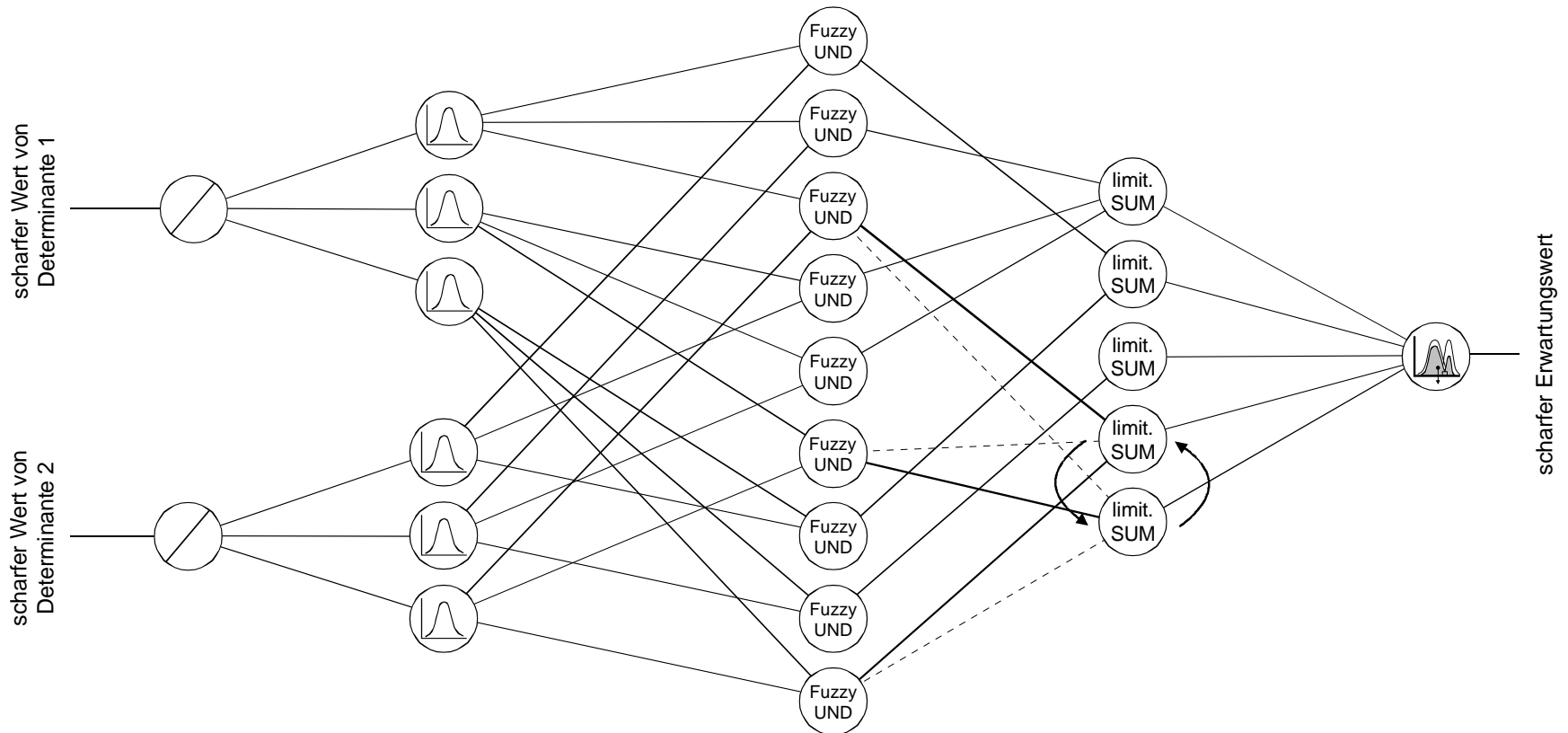
BP-Training des Gesamtsystems 3 (Rückwalzung des Fehlersignals)



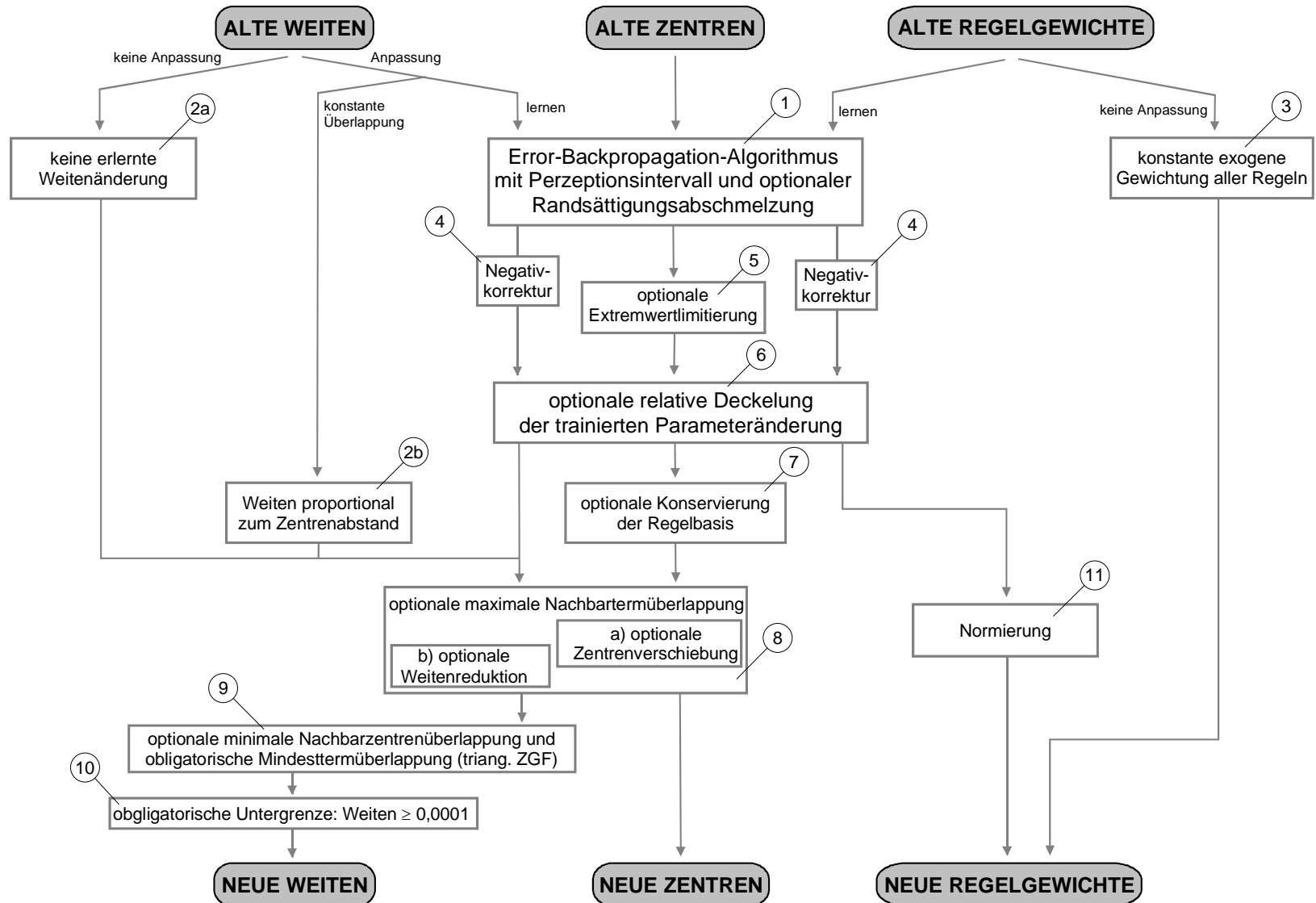
BP-Training des Gesamtsystems 4 (Rückwalzung des Fehlersignals)



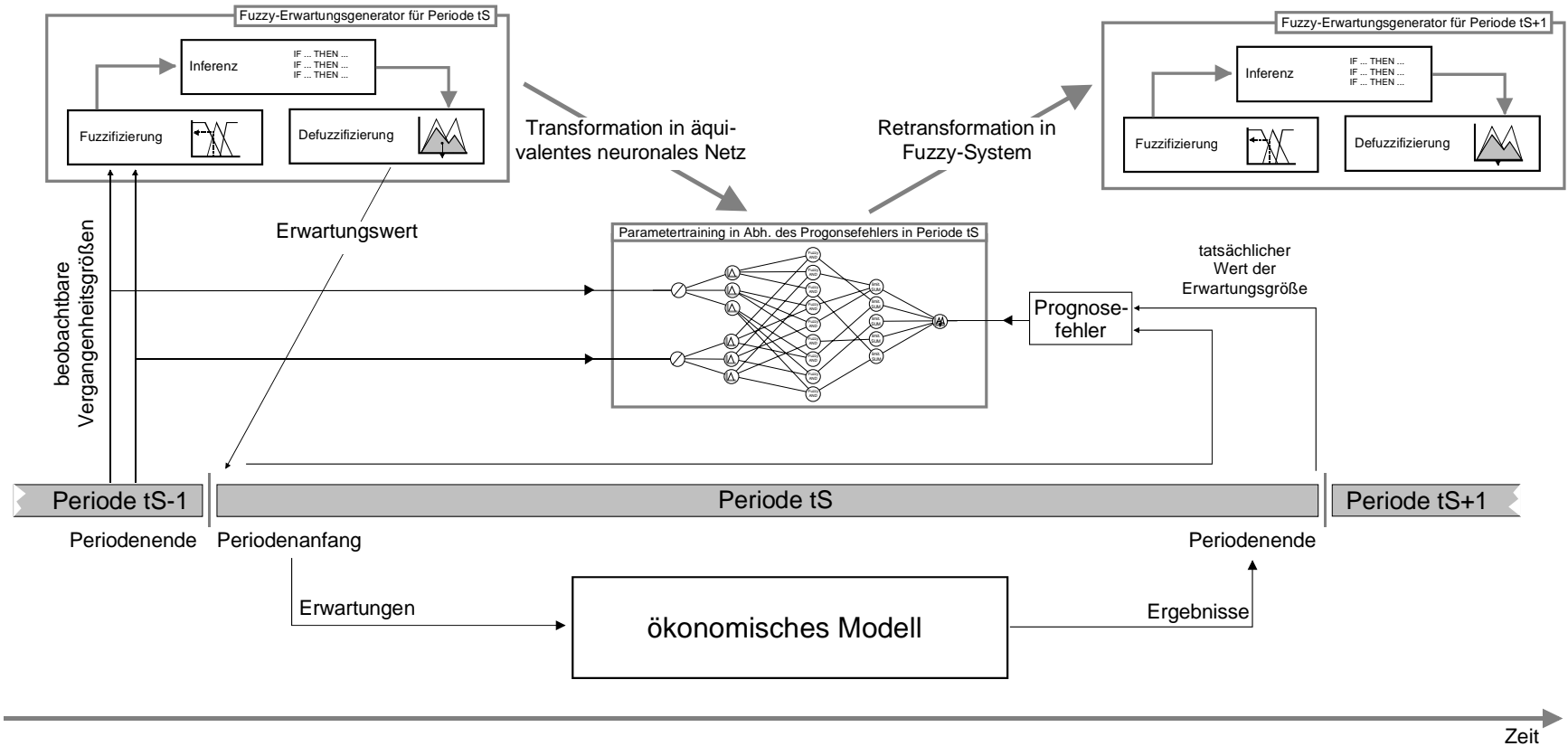
Neuverknotung bei Termzentrentausch



Modifizierter BP-Algorithmus



Einsatz als Prognosewerkzeug



Gliederung

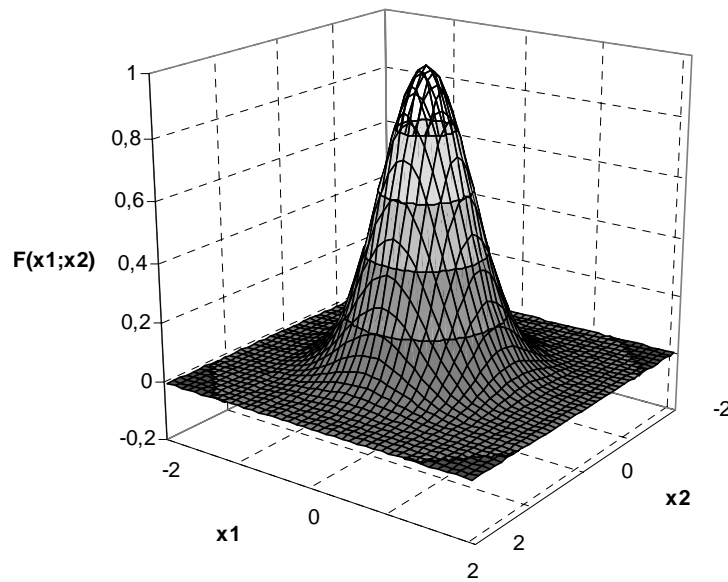
1. Einführung und Einordnung
2. Neuronale Netze 1: Grundlagen
3. Neuronale Netze 2: Konzeption und Anwendung
4. Neuro-Fuzzy-Systeme
5. Genetische Algorithmen (Überblick)
6. Zusammenfassung und Ausblick

Grundidee

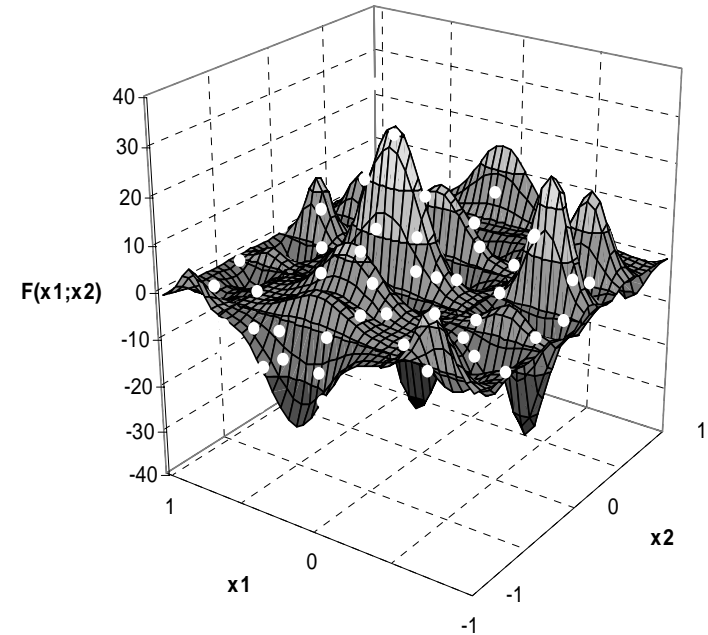
- Kodierung der Problemlösung in „Chromosomenform“ (meist 0-1-Strings)
- Erzeugung einer Population möglicher Lösungen
- Bewertung über Fitnessfunktion
- Weiterentwicklung der Population (Generationen)
 - Selektion/Reproduktion
 - Crossover/Recombination
 - Mutation
- Anwendung
 - kein abzählbarer Lösungsraum
 - Ziel: gute (nicht zwingend perfekte) Lösung
 - geeignet bei zerklüfteten Suchräumen

Neuronale Netze und Genetische Algorithmen

Neuronale
Netze (EBP)



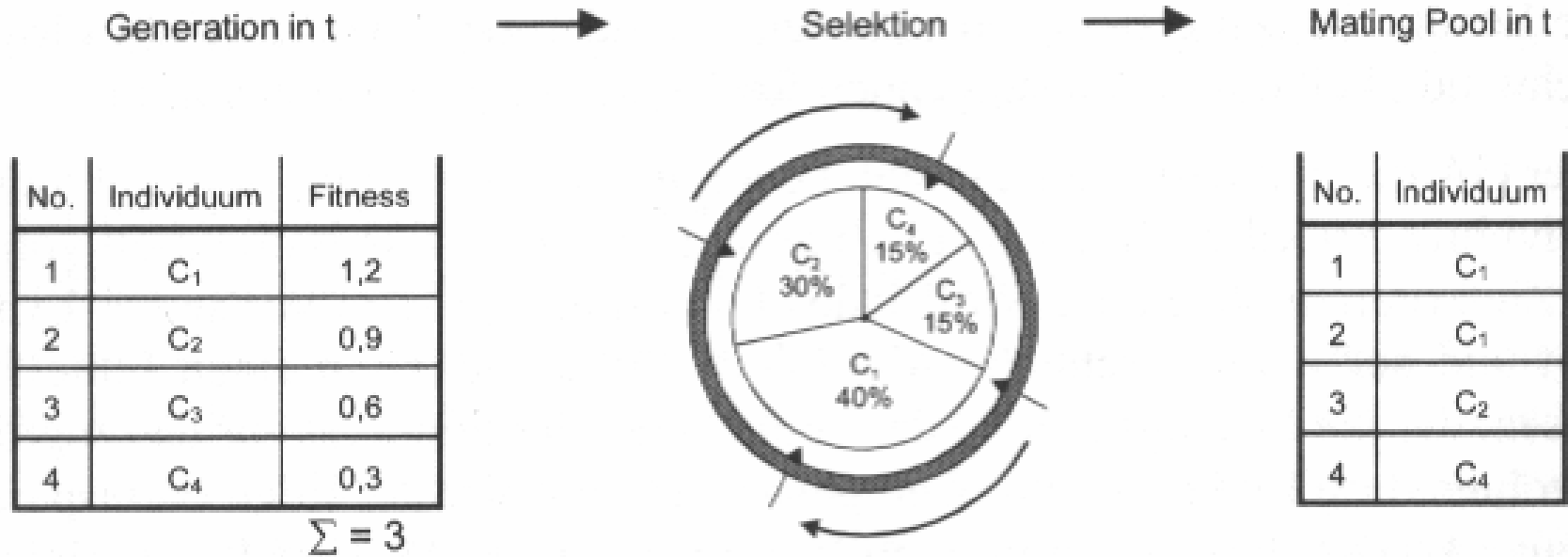
Genetische
Algorithmen



Selektion und Reproduktion

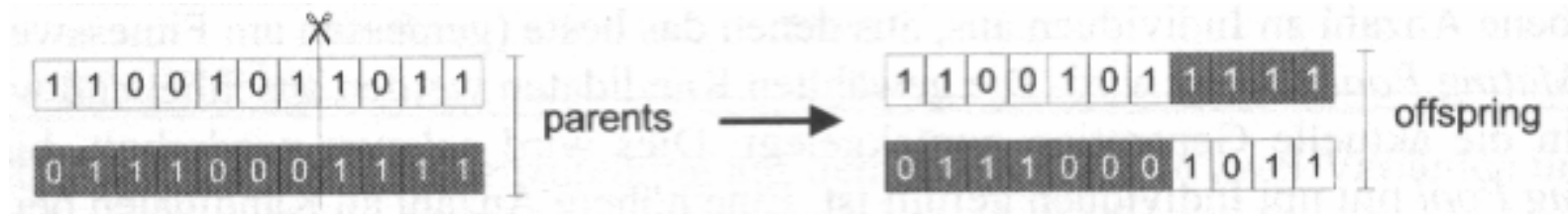
- „Survival of the fittest“
- Mating Pool
 - Individuenauswahl aus bestehender Population (Eltern) für die Bildung der nächsten Generation (offspring)
 - Ziehen mit Zurücklegen
 - Auswahlwahrscheinlichkeit steigt mit zunehmender Fitness
- Selektionsalgorithmen
 - Stochastic Universal Sampling
 - fitnessproportional
 - rangbasiert \longrightarrow $Fit(C_{pi}) = \frac{E_{max} - (E_{max} - E_{min}) \cdot \frac{Rang - 1}{npi - 1}}{npi}$ mit: $E_{max} \in [1,2] \wedge E_{min} = 2 - E_{max}$.
 - Tournament Selection
 - (Elite Selection)

Stochastic Universal Sampling



Crossover/Recombination

- Suchoperator zur Erzeugung neuer Lösungen aus dem Mating Pool
- zufälliger Austausch von Teilabschnitten der Zeichenkette der Eltern
- Crossover-Wahrscheinlichkeit: ca. 60 %
- Bsp.: Ein-Punkt-Crossover, binäre Zeichenkette:



Mutation

- zufällige Veränderung einzelner Elemente der Zeichenkette (background-operator)
- Abkürzung des Crossover-Verfahrens
- wirkt lokalen Optima entgegen
- Mutationswahrscheinlichkeit: ca. 1 %



Ökonomisches Beispiel: Monopolmenge

Beispiel 3.1: Genetischer Algorithmus und Monopolmenge

Angenommen ein Monopolist sieht sich folgender Preis-Absatz-Funktion ausgesetzt:

$$(B3.1) \quad p = 100 - x \quad \text{Preis-Absatz-Funktion}$$

Mit x seien die Mengeneinheiten des Monopolgutes bezeichnet. Der Monopolist möge sich ferner folgender Kostenfunktion gegenüber sehen und sei annahmegemäß Gewinnmaximierer:

$$(B3.2) \quad K = 50 + 10x \quad \text{Kostenfunktion}$$

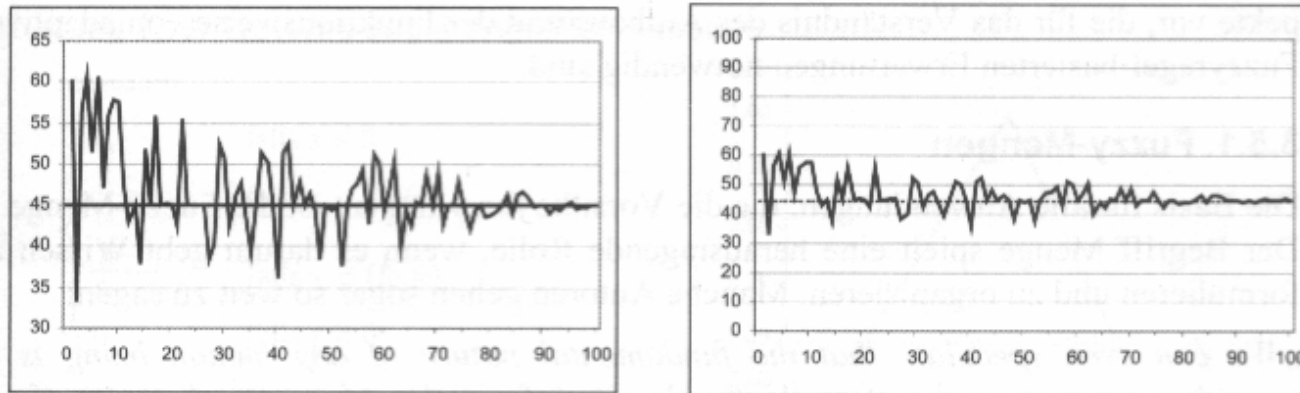
$$(B3.3) \quad G = (100 - x) \cdot x - 50 - 10x \quad \text{Gewinnfunktion}$$

Zudem möge seine Kapazitätsgrenze bei 100 ME liegen. Der klassische analytische Lösungsweg des (Gewinn)Maximierungsproblems führt über die Gewinnmaximierungsbedingungen erster und zweiter Ordnung zu der Lösung: $x = 45$ [ME] mit einem maximalen Gewinn von $G_{\max} = 1975$ [GE]. Diese Lösung setzt jedoch zwingend die Kenntnis des Monopolisten über die Preis-Absatz-Funktion voraus. Ist diese Funktion unbekannt, so wäre der Anbieter gezwungen, den Zusammenhang zwischen Angebotsmenge und Gewinn zu lernen. Er würde daher bspw. eine bestimmte Menge x' am Markt anbieten, den erzielbaren Preis beobachten und könnte damit seinen Gewinn berechnen. Den Lernprozess des Monopolisten, in dem er schrittweise die Menge erlernt, die zum Gewinnmaximum führt, wird nun über einen GA modelliert. Die Fitnessfunktion stellt die Gewinnfunktion dar und die Menge x ist der Parameter, der gelernt werden soll. Dieser Parameter wird daher binär kodiert und in einer Population der Größe 20 über 100 Generationen evolviert. Die Crossover-Wahrscheinlichkeit wird mit 0,8 angenommen, die

Ökonomisches Beispiel: Monopolmenge (Forts.)

der Mutation mit 0,05. Der Selektionprozess wird über das Tournament-Verfahren realisiert. Das Ergebnis zeigt die Abbildung B 3.1 auf der folgenden Seite.

Abbildung B 3.1: Funktionsoptimierung an Hand eines GA – ein einfaches Beispiel



Quelle: Eigene Berechnungen und Darstellung mit Hilfe von Genetik201.xls.

Das linke Diagramm zeigt die durchschnittliche Angebotsmenge der gesamten Population über die 100 Generationen im relevanten Intervall, während das rechte Diagramm den gleichen Inhalt im Intervall der möglichen Produktionsmengen zeigt. Die optische Inspektion macht deutlich, dass der GA in der Lage ist, das Niveau der analytisch optimalen Lösung zu lernen. Die Oszillationen um diesen Optimalwert werden jedoch nicht gänzlich verschwinden, da der Mutationsparameter immer wieder neue Lösungen in die Population bringt. Ökonomisch interpretiert könnte man sich einen solchen GA so vorstellen, dass eine Gruppe von 20 regionalen Monopolisten mit eigenen Strategien (hier der Angebotsmenge) existiert, deren Mitglieder über Imitation (Selektion), Kommunikation (Crossover) und Experimentieren (Mutation) versuchen, die optimale Menge x zu lernen.⁴⁵⁷

Pseudocode des Genetischen Algorithmus

```
01  begin
02      Create initial population randomly
03      Evaluate all individuals
04      Generation Counter = 1
05  repeat
06      Generation Counter = Generation Counter + 1
07      Create mating pool using the reproduction and selection operators
08      Choose parent individuals randomly
09      If  $\text{random}(0,1) \leq \text{prob}_{\text{Crossover}}$  then apply Crossover Operator
10      If  $\text{random}(0,1) \leq \text{prob}_{\text{Mutate}}$  then apply Mutate Operator
11      Copy offspring in next Generation and evaluate all individuals
12  until termination criterion is met
13  end;
```

Terminierungskriterien

- vorgegebene Generationenanzahl (Obergrenze)
- Gütekriterium für beste Lösung
- keine Verbesserung der Lösung

Anwendungsbeispiel: Kodierung einer Fuzzy-Regel

