هوش مصنوعی پیشرفته

سیستم‌های عصبی ـ فازی

Neuro-Fuzzy Systems

کاظم فولادی
دانشکده مهندسی برق و کامپیوتر
دانشگاه تهران

http://courses.fouladi.ir/aai

هوش مصنوعی پیشرفته

سیستم‌های عصبی ـ فازی

۱

مقدمه

سیستم‌های عصبی–فازی

NEURO-FUZZY SYSTEMS

شبکه‌های عصبی مصنوعی
Artificial Neural Networks

ساختارهای محاسباتی سطح پایین؛
به‌خوبی با داده‌های خام کار می‌کنند.

شبکه‌های عصبی توانایی یادگیری دارند،
اما عملکرد آنها برای کاربر شفاف نیست.

منطق فازی
Fuzzy Logic

استدلال در سطح بالاتر؛
استفاده از اطلاعات زبانی کسب شده از خبره‌های دامنه.

سیستم‌های فازی معمولی توانایی یادگیری ندارند،
و نمی‌توانند خوشان را با یک محیط جدید تنظیم کنند.

سیستم‌های عصبی–فازی
Neuro-Fuzzy Systems

ترکیب توانایی‌های
* **شبکه‌های عصبی** (محاسبه‌ی موازی، توانایی یادگیری) + عملکرد شفاف‌تر
* **سیستم‌های فازی** (بازنمایی دانایی شبیه انسان، توانایی توضیح) + قابلیت یادگیری

هوش مصنوعی پیشرفته

سیستم‌های عصبی ـ فازی

۲

سیستم عصبی ـ فازی

سیستم‌های عصبی–فازی

NEURO-FUZZY SYSTEMS

سیستم عصبی–فازی
Neuro-Fuzzy System

یک شبکه‌ی عصبی که کارکرد آن معادل با یک سیستم استنتاج فازی است.

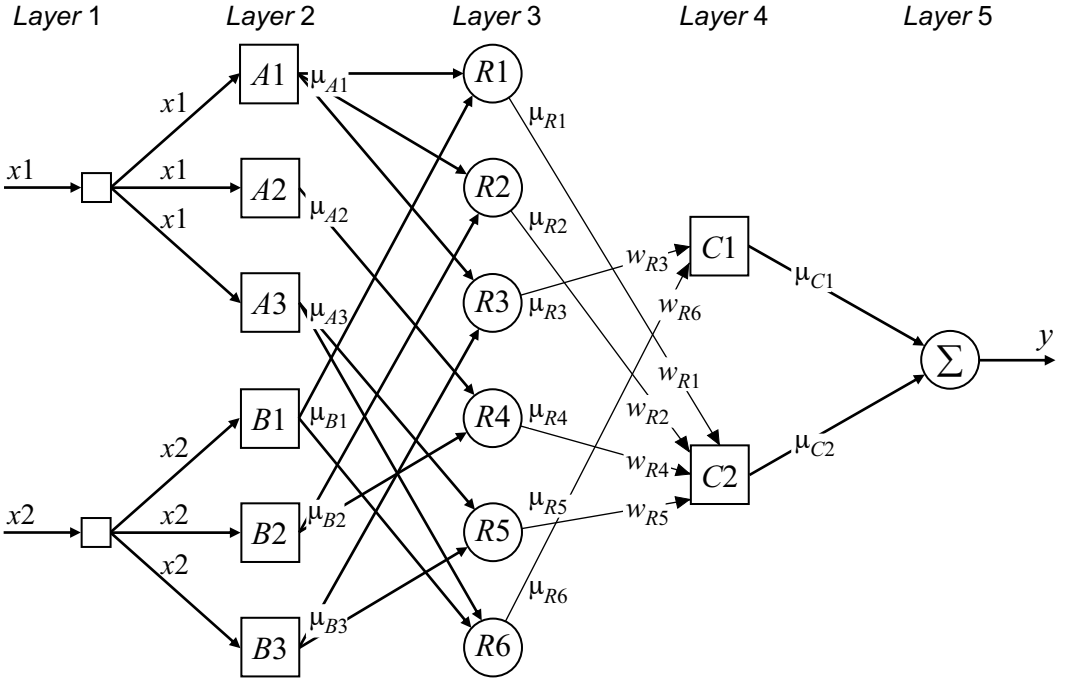
سیستم عصبی–فازی می‌تواند آموزش داده شود تا:
- قواعد فازی IF-THEN را توسعه دهد.
- توابع عضویت برای متغیرهای ورودی و خروجی سیستم را تعیین کند.

دانایی خبره می‌تواند در ساختار سیستم عصبی–فازی وارد شود.
در عین حال ساختار اتصال‌گرا (شبکه‌ی عصبی) کاری با استنتاج فازی ندارد.

دانایی خبره می‌تواند در ساختار سیستم عصبی–فازی وارد شود.
در عین حال ساختار اتصال‌گرا (شبکه‌ی عصبی) کاری با استنتاج فازی ندارد.

ساختار یک سیستم عصبی–فازی

STRUCTURE OF NEURO-FUZZY SYSTEM

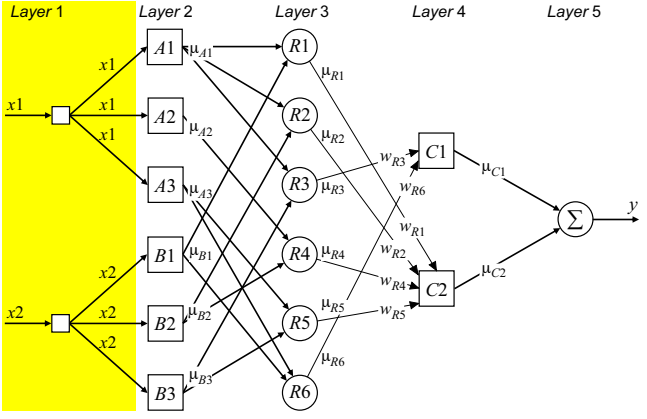


ساختار یک سیستم عصبی–فازی مشابه یک شبکه‌ی عصبی چندلایه است:
لایه‌های: ورودی، خروجی، و سه لایه‌ی پنهان (برای بازنمایی توابع عضویت و قواعد فازی)

ساختار یک سیستم عصبی–فازی

لایه‌ی ورودی

STRUCTURE OF NEURO-FUZZY SYSTEM



لایه‌ی ورودی
Input Layer

لایه‌ی ۱

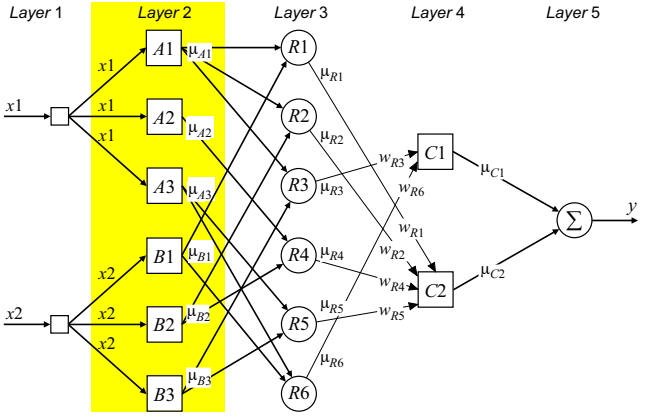
هر نرون در این لایه،
سیگنال‌های کریسپ خارجی را مستقیماً به لایه‌ی بعدی می‌فرستد:

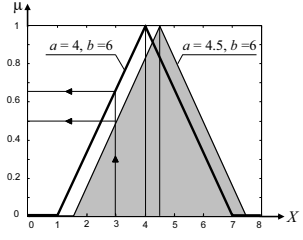
$$y_i^{(1)} = x_i^{(1)}$$

ساختار یک سیستم عصبی–فازی

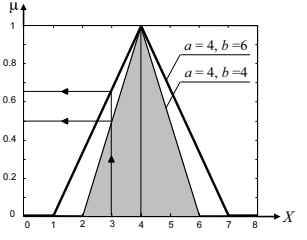
لایه‌ی فازی‌سازی

STRUCTURE OF NEURO-FUZZY SYSTEM





(*a*) Effect of parameter *a*.



(*b*) Effect of parameter *b*.

لایه‌ی ۲

لایه‌ی فازی‌سازی
Fuzzification Layer

هر نرون در این لایه،
مجموعه‌ی فازی در بخش مقدم قاعده‌های فازی را بازنمایی می‌کند.

یک نرون فازی‌سازی، یک سیگنال ورودی کریسپ را دریافت می‌کند و درجه‌ی تعلق آن ورودی به مجموعه‌ی فازی آن نرون را تعیین می‌کند.

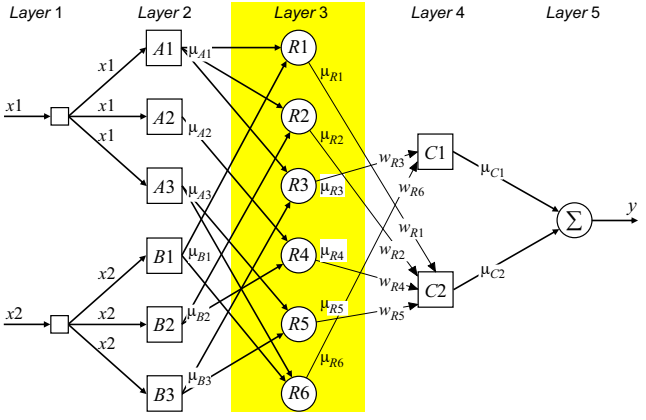
تابع فعال‌سازی یک نرون عضویت:
تابعی که مجموعه‌ی فازی آن نرون را مشخص می‌کند.
(مثل تابع عضویت مثلثی شکل)

$$y_i^{(2)} = \begin{cases} 0, & \text{if } x_i^{(2)} \le a - \dfrac{b}{2} \\ 1 - \dfrac{2\left|x_i^{(2)} - a\right|}{b}, & \text{if } a - \dfrac{b}{2} < x_i^{(2)} < a + \dfrac{b}{2} \\ 0, & \text{if } x_i^{(2)} \ge a + \dfrac{b}{2} \end{cases}$$

ساختار یک سیستم عصبی–فازی

لایه‌ی قاعده‌ی فازی

STRUCTURE OF NEURO-FUZZY SYSTEM



$$y_{R1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_{R1}$$

لایه‌ی ۳
لایه‌ی قاعده‌ی فازی
Fuzzy Rule Layer

هر نرون در این لایه،
متناظر با یک قاعده‌ی فازی واحد است.

یک نرون قاعده‌ی فازی، ورودی‌ها را از نرون‌های فازی‌سازی می‌گیرد (نرون‌های بازنمایی‌کننده‌ی مجموعه‌های فازی در قسمت مقدم قاعده) و یک درجه‌ی عضویت برمی‌گرداند.

مثلاً: نرون $R1$ متناظر با Rule 1
ورودی‌هایش را از نرون‌های $A1$ و $B1$ می‌گیرد.

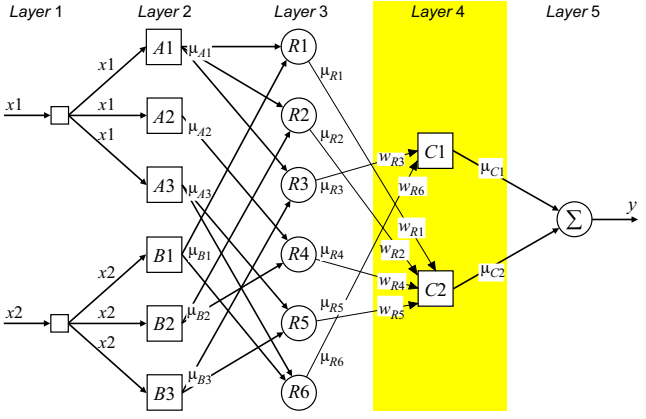
در سیستم‌های عصبی–فازی، اشتراک می‌تواند با **عملگر ضرب** پیاده‌سازی شود، پس خروجی لایه ۳ می‌شود:

$$y_i^{(3)} = x_{1i}^{(3)} \times x_{2i}^{(3)} \times \ldots \times x_{ki}^{(3)}$$

ساختار یک سیستم عصبی–فازی

لایه‌ی عضویت خروجی

STRUCTURE OF NEURO-FUZZY SYSTEM



لایه ۴

لایه‌ی عضویت خروجی
Output Membership Layer

نرون‌های این لایه، مجموعه‌های فازی مورد استفاده در تالی قواعد فازی را بازنمایی می‌کند.

یک نرون عضویت خروجی، همه‌ی ورودی‌هایش را با استفاده از عملگر فازی **اجتماع** ترکیب می‌کند.
(این عمل می‌تواند با **probabilistic OR** پیاده‌سازی شود)

خروجی این نرون: قوت فایر کردن جمع‌بندی شده‌ی نرون‌های قاعده‌ی فازی ورودی است.

$$y_{C1}^{(4)} = \mu_{R3} \oplus \mu_{R6} = \mu_{C1}$$

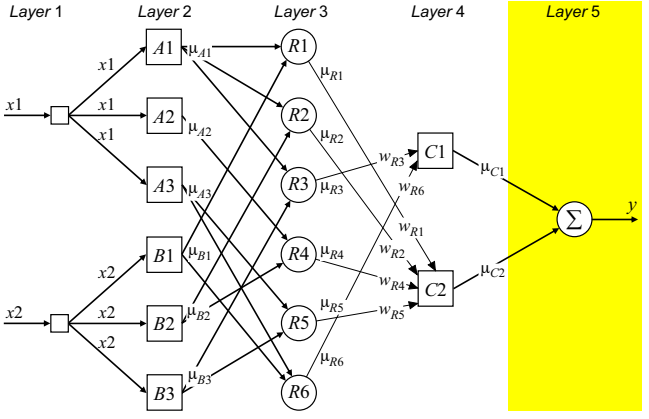
$$y_i^{(4)} = x_{1i}^{(4)} \oplus x_{2i}^{(4)} \oplus \ldots \oplus x_{li}^{(4)}$$

$$u \oplus v = u + v - u \cdot v$$

ساختار یک سیستم عصبی–فازی

لایه‌ی غیرفازی‌سازی

STRUCTURE OF NEURO-FUZZY SYSTEM



$$y = \frac{\mu_{C1} \times a_{C1} \times b_{C1} + \mu_{C2} \times a_{C2} \times b_{C2}}{\mu_{C1} \times b_{C1} + \mu_{C2} \times b_{C2}}$$

مرکز عرض

لایه‌ی ۵

لایه‌ی غیرفازی‌سازی
Defuzzification Layer

هر نرون در این لایه،
یک خروجی واحد از سیستم عصبی–فازی را بازنمایی می‌کند.

یک نرون غیرفازی‌کننده، مجموعه‌های فازی خروجی بریده شده با قوت‌های فایرکردن جمع‌بندی شده را دریافت می‌کند و آنها در یک مجموعه‌ی فازی واحد ترکیب می‌کند.

می‌توان از روش‌های مختلف غیرفازی‌سازی استفاده کرد (مثل مرکز جرم) در اینجا از ترکیب **sum-product** استفاده می‌کنیم.

ترکیب **sum-product**
خروجی کریسپ را به صورت متوسط وزن‌دار مرکزهای همه‌ی توابع عضویت خروجی محاسبه می‌کند.

یادگیری در یک سیستم عصبی–فازی

LEARNING IN A NEURO-FUZZY SYSTEM

یک سیستم عصبی–فازی اساساً یک شبکه‌ی عصبی چندلایه است

⇓

می‌توان الگوریتم‌های یادگیری استاندارد شبکه‌های عصبی را به‌کار گرفت:

مانند **الگوریتم پس‌انتشار خطا**

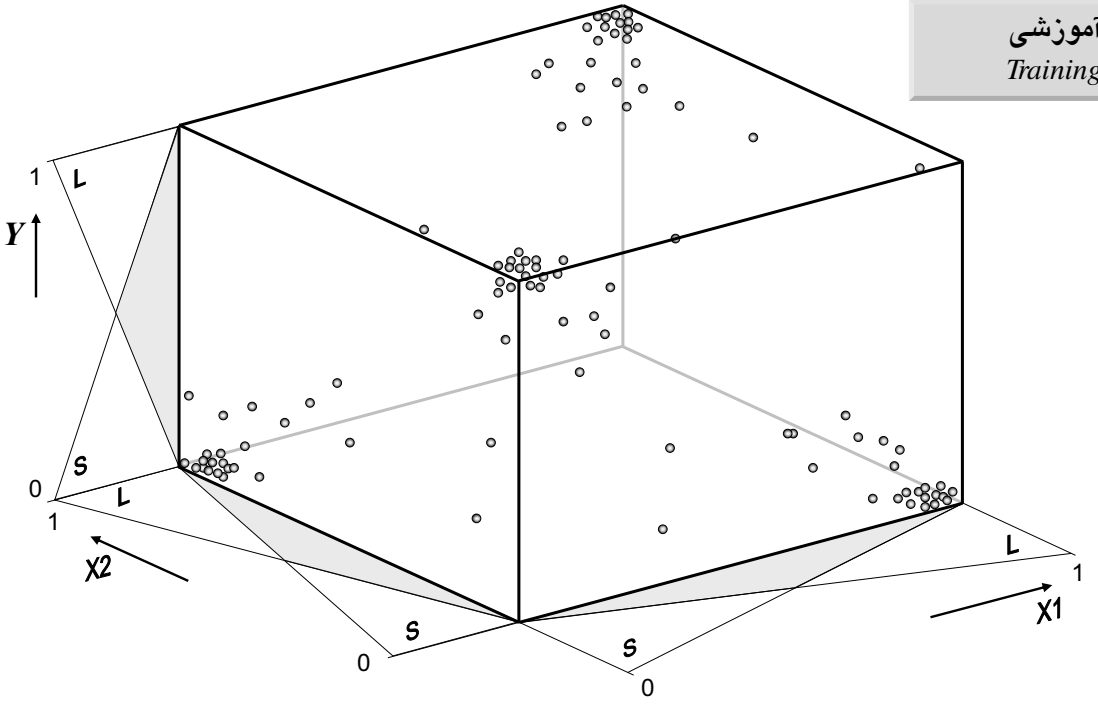
وقتی یک مثال ورودی–خروجی آموزشی به سیستم نمایش داده می‌شود،
الگوریتم پس‌انتشار خروجی سیستم را محاسبه می‌کند و
با خروجی مطلوب آن مثال آموزشی مقایسه می‌کند.
خطا در جهت پس‌رو در سراسر شبکه از لایه‌ی خروجی به لایه‌ی ورودی منتشر می‌شود.
توابع فعال‌سازی نرون‌ها با منتشر شدن خطا تغییر پیدا می‌کنند.
(برای تعیین تغییرات لازم، الگوریتم پس‌انتشار از توابع فعال‌سازی نرون‌ها مشتق‌گیری می‌کند.)

یادگیری در یک سیستم عصبی–فازی

مثال (۱ از ۲)

LEARNING IN A NEURO-FUZZY SYSTEM

الگوهای آموزشی
Training Patterns



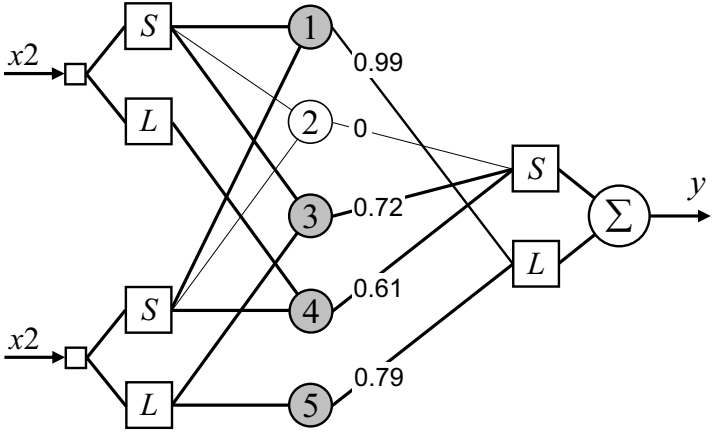
از این مجموعه داده برای آموزش یک سیستم عصبی–فازی با ۵ قاعده استفاده می‌شود.

یادگیری در یک سیستم عصبی–فازی

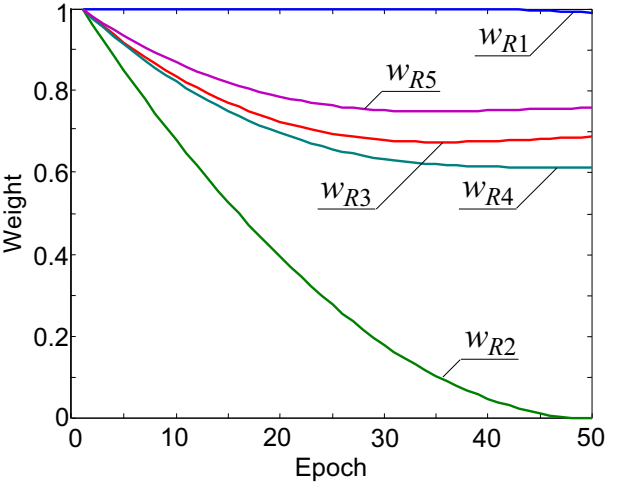
مثال (۲ از ۲)

LEARNING IN A NEURO-FUZZY SYSTEM

سیستم عصبی–فازی با ۵ قاعده
5-Rule Neuro-Fuzzy System



(*a*) Five-rule system.

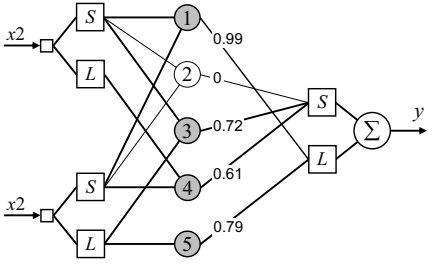


(*b*) Training for 50 epochs.

یادگیری در یک سیستم عصبی–فازی

نکات

LEARNING IN A NEURO-FUZZY SYSTEM



(*a*) Five-rule system.

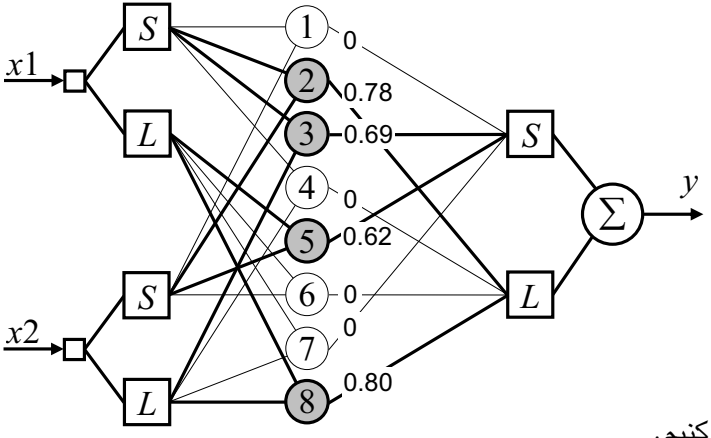
- فرض می‌کنیم قواعد IF-THEN فازی وارد شده به ساختار سیستم، توسط یک خبره‌ی دامنه وارد شده باشند.
- دانایی پیشین یا دانایی موجود، می‌تواند آموزش سیستم را تا حد زیادی شتاب ببخشد.
- به‌علاوه، اگر کیفیت داده‌های آموزشی ضعیف باشد، دانایی خبره ممکن است تنها راه رسیدن به راه‌حل باشد.
- با این وجود، خبره‌ها گه‌گاهی مرتکب اشتباه می‌شوند و بنابراین برخی قواعد موجود در سیستم عصبی–فازی ممکن است نادرست یا افزونه باشند.
- ⇐ سیستم عصبی–فازی باید قادر به شناسایی **قواعد بد** باشد.
- با داشتن مقادیر زبانی ورودی و خروجی، یک سیستم عصبی–فازی می‌تواند یک مجموعه قواعد فازی IF-THEN را به‌طور خودکار تولید کند.

یادگیری در یک سیستم عصبی–فازی

مثال: یادگیری یای انحصاری (۱ از ۲)

LEARNING IN A NEURO-FUZZY SYSTEM

سیستم عصبی–فازی با ۸ قاعده
8-Rule Neuro-Fuzzy System



(*a*) Eight-rule system.

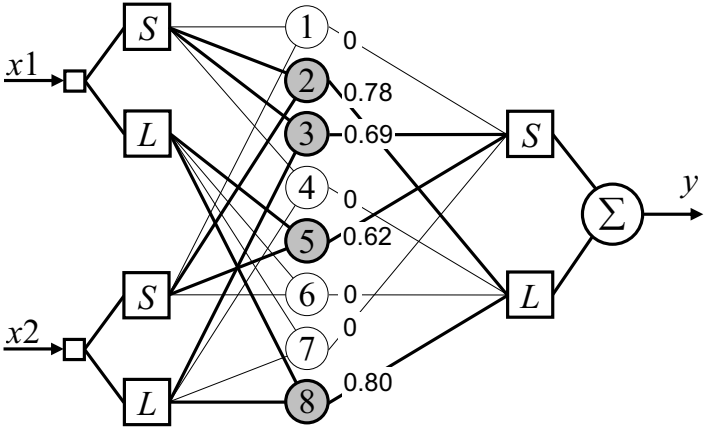
- برای ایجاد یک سیستم برای مثال XOR:
  - این سیستم $2^2 \times 2 = 8$ قاعده دارد.
  - چون دانایی خبره هم‌اکنون در این سیستم جای داده نشده است، همه‌ی وزن‌های لایه‌ی ۳ و لایه‌ی ۴ را مساوی با 0.5 قرار می‌دهیم.
  - پس از آموزش، می‌توانیم همه‌ی قواعدی که فاکتورهای اطمینان آنها کمتر از یک عدد به‌اندازه‌ی کافی کوچک (مثلاً 0.1) است را حذف کنیم.
  - در نتیجه، همان مجموعه از چهار قاعده‌ی فازی IF-THEN بازنمایی‌کننده‌ی عمل XOR را به دست می‌آوریم.

یادگیری در یک سیستم عصبی–فازی

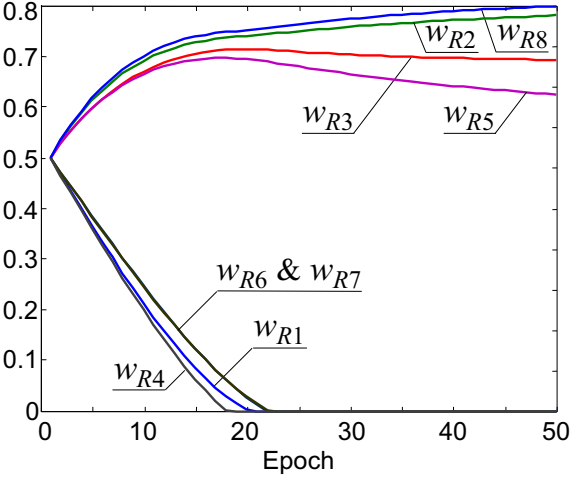
مثال: یادگیری یای انحصاری (۲ از ۲)

LEARNING IN A NEURO-FUZZY SYSTEM

سیستم عصبی–فازی با ۸ قاعده
8-Rule Neuro-Fuzzy System



(*a*) Eight-rule system.



(*b*) Training for 50 epochs.

سیستم‌های عصبی–فازی

خلاصه

NEURO-FUZZY SYSTEMS

وقتی یک مجموعه از مثال‌ها نماینده موجود باشد، سیستم عصبی–فازی می‌تواند آن به طور خودکار به یک مجموعه‌ی مقاوم از قواعد فازی IF-THEN تبدیل کند.

دانایی دامنه می‌تواند توسط خبره‌های انسانی در قالب **متغیرهای زبانی** در یک سیستم عصبی–فازی قرار گیرد.

سیستم‌های عصبی–فازی

Neuro-Fuzzy Systems

ترکیب منطق فازی و شبکه‌های عصبی، یک ابزار قدرت‌مند برای طراحی سیستم‌های هوشمند فراهم می‌کند.

* کاهش وابستگی به دانایی خبره هنگام ساخت سیستم‌های هوشمند

هوش مصنوعی پیشرفته

سیستم‌های عصبی ـ فازی

۳

ANFIS

ANFIS

ANFIS: Adaptive Neuro-Fuzzy Inference System

ANFIS یک شبکه‌ی عصبی–فازی ۶ لایه بر اساس مدل فازی سوگنو مرتبه یک

مدل فازی سوگنو، برای تولید قواعد فازی از یک مجموعه داده‌ی ورودی–خروجی داده شده:

$$
\begin{aligned}
&\text{IF} \quad x_1 \text{ is } A_1 \\
&\text{AND} \quad x_2 \text{ is } A_2 \\
&\quad \ldots\ldots \\
&\text{AND} \quad x_m \text{ is } A_m \\
&\text{THEN} \quad y = f(x_1, x_2, \ldots, x_m)
\end{aligned}
$$

متغیرهای زبانی: $x_1, x_2, \ldots, x_m$

مجموعه‌های فازی: $A_1, A_2, \ldots, A_m$

مدل فازی سوگنو مرتبه صفر
Zero-Order Sugeno Fuzzy Model

وقتی y ثابت است:
تالی قاعده به‌صورت یک سینگلتون (عدد تکی) مشخص می‌شود.

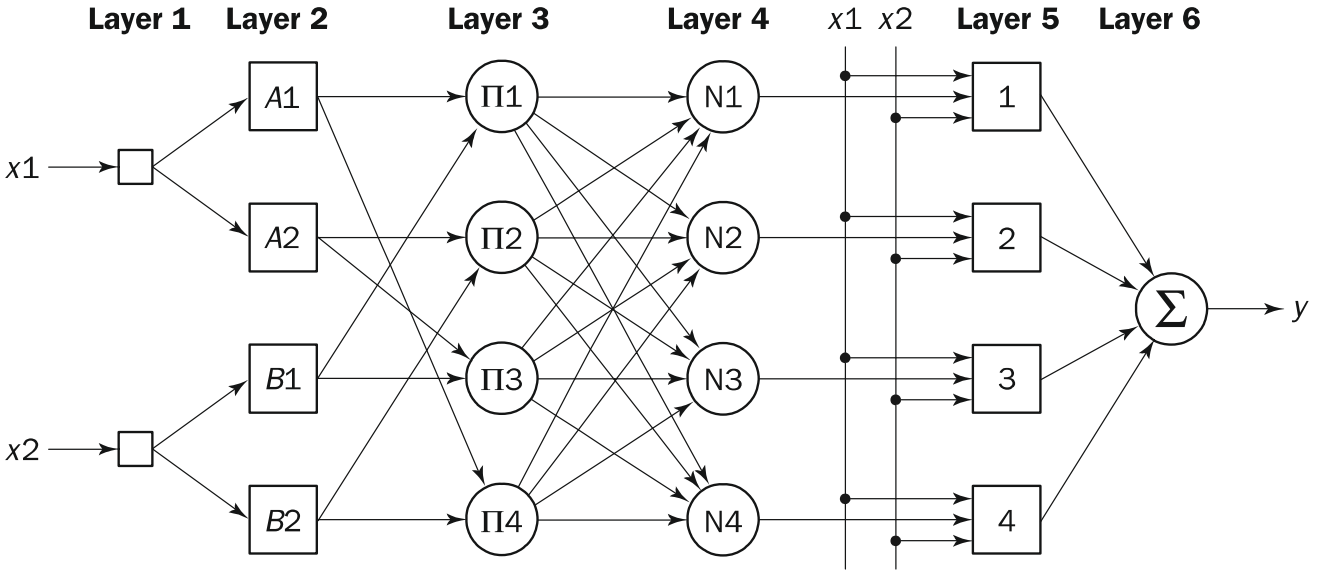
مدل فازی سوگنو مرتبه یک
First-Order Sugeno Fuzzy Model

وقتی y یک چندجمله‌ای مرتبه اول است:
تالی قاعده به‌صورت: $y = k_0 + k_1 x_1 + k_2 x_2 + \ldots + k_m x_m$

ANFIS

یک شبکه‌ی ۶ لایه

ANFIS: Adaptive Neuro-Fuzzy Inference System

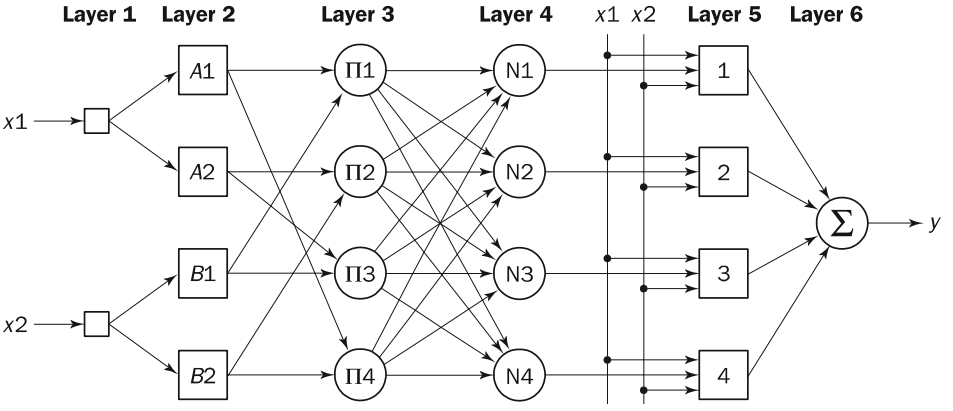


Adaptive Neuro-Fuzzy Inference System (ANFIS)

ANFIS

قواعد فازی مرتبط

ANFIS: Adaptive Neuro-Fuzzy Inference System



Rule 1:
IF $x1$ is $A1$
AND $x2$ is $B1$
THEN $y = f_1 = k_{10} + k_{11}x1 + k_{12}x2$

Rule 2:
IF $x1$ is $A2$
AND $x2$ is $B2$
THEN $y = f_2 = k_{20} + k_{21}x1 + k_{22}x2$

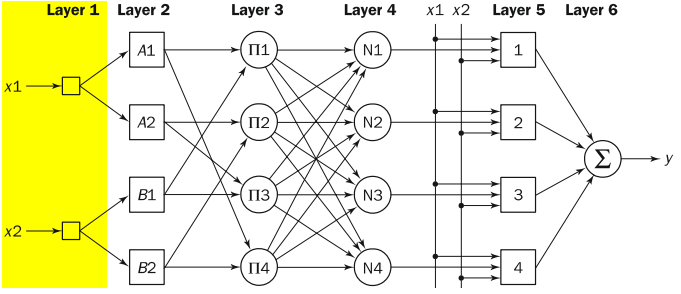
Rule 3:
IF $x1$ is $A2$
AND $x2$ is $B1$
THEN $y = f_3 = k_{30} + k_{31}x1 + k_{32}x2$

Rule 4:
IF $x1$ is $A1$
AND $x2$ is $B2$
THEN $y = f_4 = k_{40} + k_{41}x1 + k_{42}x2$

ساختار ANFIS

لایه‌ی ورودی

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۱

لایه‌ی ورودی
Input Layer

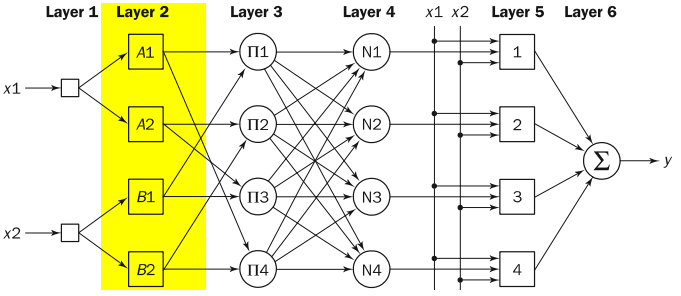
هر نرون در این لایه،
سیگنال‌های کریسپ خارجی را مستقیماً به لایه‌ی ۲ می‌فرستد:

$$y_i^{(1)} = x_i^{(1)}$$

ساختار ANFIS

لایه‌ی فازی‌سازی

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۲

لایه‌ی فازی‌سازی
Fuzzification Layer

هر نرون در این لایه،
فازی‌سازی را انجام می‌دهد.

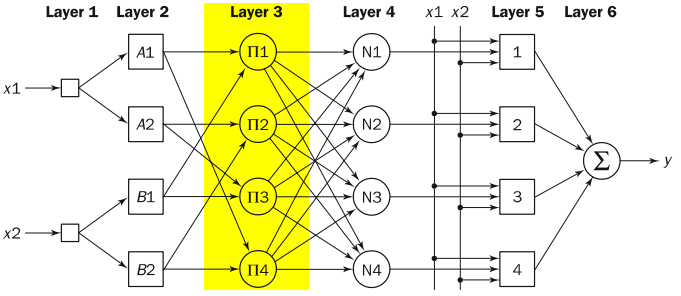
در مدل Jang نرون‌های فازی‌سازی
تابع فعال‌سازی زنگوله‌ای دارند.

$$y_i^{(2)} = \frac{1}{1 + \left(\frac{x_i^{(2)} - a_i}{c_i} \right)^{2b_i}}$$

ساختار ANFIS

لایه‌ی قواعد

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۳

لایه‌ی قواعد
Rule Layer

هر نرون در این لایه،
متناظر با یک قاعده‌ی فازی واحد از نوع سوگنو است.

یک نرون قاعده، ورودی‌ها را از نرون‌های فازی‌سازی دریافت می‌کند و قوت فایرکردن قاعده‌ی بازنمایی‌شده توسط آن را محاسبه می‌کند.

در ANFIS عطف مقدم‌های قواعد
با عملگر **ضرب** ارزیابی می‌شود.

$$y_{\Pi 1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_1,$$

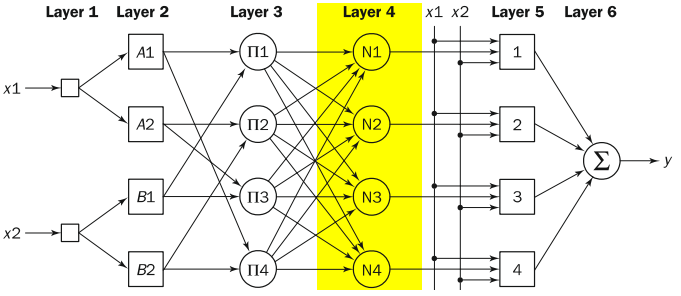
قوت فایر کردن قاعده
= مقدار درستی قاعده R1

$$y_i^{(3)} = \prod_{j=1}^{k} x_{ji}^{(3)}$$

ساختار ANFIS

لایه‌ی نرمال‌سازی

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۴

لایه‌ی نرمال‌سازی
Normalization Layer

هر نرون در این لایه،
ورودی‌هایش را از همه‌ی نرون‌ها در لایه‌ی قاعده می‌گیرد و
قوت فایر کردن نرمال‌شده‌ی آن قاعده را محاسبه می‌کند.

قوت فایرکردن نرمال‌شده یک قاعده:
نسبت قوت فایر کردن آن قاعده بر
مجموع قوت فایر کردن همه‌ی قاعده‌هاست.
(بیانگر نقش قاعده‌ی داده‌شده در نتیجه‌ی نهایی)

$$y_{N1}^{(4)} = \frac{\mu_1}{\mu_1+\mu_2+\mu_3+\mu_4} = \bar{\mu}_1$$

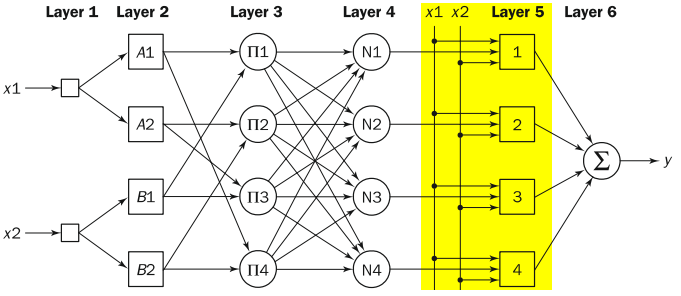
قوت فایر کردن نرمال شده‌ی قاعده

$$y_i^{(4)} = \frac{x_{ii}^{(4)}}{\sum_{j=1}^{n} x_{ji}^{(4)}} = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j} = \bar{\mu}_i$$

ساختار ANFIS

لایه‌ی غیرفازی‌سازی

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۵

لایه‌ی غیرفازی‌سازی
Defuzzification Layer

هر نرون در این لایه،
به نرون نرمال‌سازی متناظر متصل است
و ورودی‌های آغازین x_1 و x_2 را هم می‌گیرد.

نرون غیرفازی‌کننده،
مقادیر تالی وزن‌دار یک قاعده‌ی داده شده را محاسبه می‌کند.

مجموعه‌ی پارامترهای تالی قاعده‌ی i

$$y_i^{(5)} = x_i^{(5)} \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2\right] = \overline{\mu}_i \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2\right]$$

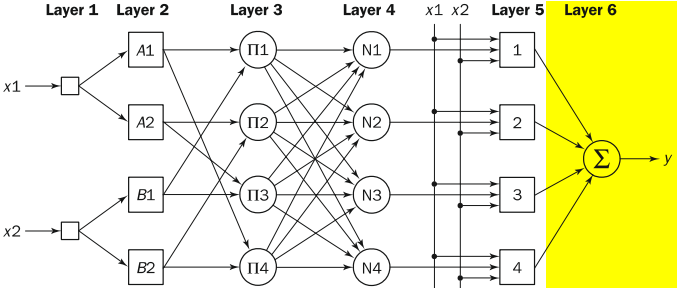
ورودی نرون غیرفازی‌کننده i در لایه ۵

خروجی نرون غیرفازی‌کننده i در لایه ۵

ساختار ANFIS

لایه‌ی خروجی

STRUCTURE OF ANFIS: ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM



لایه‌ی ۶

لایه‌ی خروجی
Output Layer

با یک نرون جمع‌کننده‌ی واحد بازنمایی می‌شود.

نرون جمع‌کننده،
مجموع خروجی‌های همه‌ی نرون‌های غیرفازی‌کننده را محاسبه می‌کند و خروجی نهایی ANFIS یعنی y را تولید می‌کند.

$$y = \sum_{i=1}^{n} x_i^{(6)} = \sum_{i=1}^{n} \bar{\mu}_i \left[k_{i0} + k_{i1}\, x1 + k_{i2}\, x2\right]$$

ANFIS

برخورد با فقدان دانایی پیشین

آیا ANFIS می‌تواند با مسائلی که در آنها دانایی پیشین از پارامترهای تالی قاعده وجود ندارد، کار کند؟

بله:
لازم نیست هیچ دانایی پیشینی از پارامترهای تالی قاعده داشته باشیم.
ANFIS این پارامترها را یاد می‌گیرد و توابع عضویت را تنظیم می‌کند.

یادگیری در مدل ANFIS

LEARNING IN THE ANFIS MODEL

یک ANFIS از یک الگوریتم یادگیری هیبرید استفاده می‌کند:
ترکیب تخمین‌زن حداقل مربعات + روش کاهش گرادیان

در الگوریتم آموزش ANFIS هر اپُک از دو گذر تشکیل می‌شود:

۱) گذر پیش‌رو (forward pass):

یک مجموعه‌ی آموزشی از الگوهای ورودی (بردار ورودی) به ANFIS نشان داده می‌شود،
خروجی نرون‌ها لایه به لایه محاسبه می‌شود،
و پارامترهای تالی قاعده‌ها شناسایی می‌شوند (با روش تخمین حداقل مربعات): $\boldsymbol{k}$.
سپس، بردار خروجی واقعی شبکه محاسبه می‌شود: $\boldsymbol{y}$،
آن‌گاه، بردار خطا مشخص می‌شود.

$$\boldsymbol{e} = \boldsymbol{y}_d - \boldsymbol{y}$$

۲) گذر پس‌رو (backward pass):

الگوریتم پس‌انتشار خطا استفاده می‌شود:
سیگنال‌های خطا به سمت عقب منتشر می‌شود
و پارامترهای مقدم قاعده‌ها با توجه به قاعده‌ی زنجیره‌ای به‌هنگام‌سازی می‌شوند.

یادگیری در مدل ANFIS

شناسایی پارامترهای تالی قواعد با تخمین‌زن حداقل مربعات

RULE CONSEQUENT PARAMETERS IDENTIFICATION BY THE LEAST-SQUARES ESTIMATOR

پارامترهای تالی قاعده‌ها با روش تخمین حداقل مربعات شناسایی می‌شوند:

در استنتاج فازی به سبک سوگنو، خروجی y تابعی خطی است.
پس با داشتن مقادیر پارامترهای عضویت و یک مجموعه‌ی آموزشی از P الگوهای ورودی–خروجی
می‌توانیم یک دستگاه معادلات از P معادله‌ی خطی بر حسب پارامترهای تالی تشکیل بدهیم:

$$\begin{cases} y_d(1) = \bar{\mu}_1(1)f_1(1) + \bar{\mu}_2(1)f_2(1) + \ldots + \bar{\mu}_n(1)f_n(1) \\ y_d(2) = \bar{\mu}_1(2)f_1(2) + \bar{\mu}_2(2)f_2(2) + \ldots + \bar{\mu}_n(2)f_n(2) \\ \vdots \\ y_d(p) = \bar{\mu}_1(p)f_1(p) + \bar{\mu}_2(p)f_2(p) + \ldots + \bar{\mu}_n(p)f_n(p) \\ \vdots \\ y_d(P) = \bar{\mu}_1(P)f_1(P) + \bar{\mu}_2(P)f_2(P) + \ldots + \bar{\mu}_n(P)f_n(P) \end{cases} \Rightarrow$$

$$\begin{cases} y_d(1) = \bar{\mu}_1(1)[k_{10} + k_{11}x_1(1) + k_{12}x_2(1) + \ldots + k_{1m}x_m(1)] \\ \quad + \bar{\mu}_2(1)[k_{20} + k_{21}x_1(1) + k_{22}x_2(1) + \ldots + k_{2m}x_m(1)] + \ldots \\ \quad + \bar{\mu}_n(1)[k_{n0} + k_{n1}x_1(1) + k_{n2}x_2(1) + \ldots + k_{nm}x_m(1)] \\ y_d(2) = \bar{\mu}_1(2)[k_{10} + k_{11}x_1(2) + k_{12}x_2(2) + \ldots + k_{1m}x_m(2)] \\ \quad + \bar{\mu}_2(2)[k_{20} + k_{21}x_1(2) + k_{22}x_2(2) + \ldots + k_{2m}x_m(2)] + \ldots \\ \quad + \bar{\mu}_n(2)[k_{n0} + k_{n1}x_1(2) + k_{n2}x_2(2) + \ldots + k_{nm}x_m(2)] \\ \vdots \\ y_d(p) = \bar{\mu}_1(p)[k_{10} + k_{11}x_1(p) + k_{12}x_2(p) + \ldots + k_{1m}x_m(p)] \\ \quad + \bar{\mu}_2(p)[k_{20} + k_{21}x_1(p) + k_{22}x_2(p) + \ldots + k_{2m}x_m(p)] + \ldots \\ \quad + \bar{\mu}_n(p)[k_{n0} + k_{n1}x_1(p) + k_{n2}x_2(p) + \ldots + k_{nm}x_m(p)] \\ \vdots \\ y_d(P) = \bar{\mu}_1(P)[k_{10} + k_{11}x_1(P) + k_{12}x_2(P) + \ldots + k_{1m}x_m(P)] \\ \quad + \bar{\mu}_2(P)[k_{20} + k_{21}x_1(P) + k_{22}x_2(P) + \ldots + k_{2m}x_m(P)] + \ldots \\ \quad + \bar{\mu}_n(P)[k_{n0} + k_{n1}x_1(P) + k_{n2}x_2(P) + \ldots + k_{nm}x_m(P)] \end{cases}$$

یادگیری در مدل ANFIS

شناسایی پارامترهای تالی قواعد با تخمین‌زن حداقل مربعات: نمایش ماتریسی

RULE CONSEQUENT PARAMETERS IDENTIFICATION BY THE LEAST-SQUARES ESTIMATOR

پارامترهای تالی قاعده‌ها با روش تخمین حداقل مربعات شناسایی می‌شوند:

در استنتاج فازی به سبک سوگنو، خروجی y تابعی خطی است.
پس با داشتن مقادیر پارامترهای عضویت و یک مجموعه‌ی آموزشی از P الگوهای ورودی–خروجی
می‌توانیم یک دستگاه معادلات از P معادله‌ی خطی بر حسب پارامترهای تالی تشکیل بدهیم:

$$\boldsymbol{y}_d = \mathbf{A}\,\boldsymbol{k}$$

$$\boldsymbol{y}_d = \begin{bmatrix} y_d(1) \\ y_d(2) \\ \vdots \\ y_d(p) \\ \vdots \\ y_d(P) \end{bmatrix}_{P\times 1} \qquad \mathbf{A} = \begin{bmatrix} \bar{\mu}_1(1) & \bar{\mu}_1(1)x_1(1) & \cdots & \bar{\mu}_1(1)x_m(1) & \cdots & \bar{\mu}_n(1) & \bar{\mu}_n(1)x_1(1) & \cdots & \bar{\mu}_n(1)x_m(1) \\ \bar{\mu}_1(2) & \bar{\mu}_1(2)x_1(2) & \cdots & \bar{\mu}_1(2)x_m(2) & \cdots & \bar{\mu}_n(2) & \bar{\mu}_n(2)x_1(2) & \cdots & \bar{\mu}_n(2)x_m(2) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \bar{\mu}_1(p) & \bar{\mu}_1(p)x_1(p) & \cdots & \bar{\mu}_1(p)x_m(p) & \cdots & \bar{\mu}_n(p) & \bar{\mu}_n(p)x_1(p) & \cdots & \bar{\mu}_n(p)x_m(p) \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \bar{\mu}_1(P) & \bar{\mu}_1(P)x_1(P) & \cdots & \bar{\mu}_1(P)x_m(P) & \cdots & \bar{\mu}_n(P) & \bar{\mu}_n(P)x_1(P) & \cdots & \bar{\mu}_n(P)x_m(P) \end{bmatrix}_{P\times n(1+m)}$$

$\boldsymbol{k}$ is an $n(1+m)\times 1$ vector of unknown consequent parameters,

$$\boldsymbol{k} = [k_{10}\ k_{11}\ k_{12}\ldots k_{1m}\ k_{20}\ k_{21}\ k_{22}\ldots k_{2m}\ldots k_{n0}\ k_{n1}\ k_{n2}\ldots k_{nm}]^T$$

راه‌حل حداقل مربعات $\boldsymbol{k}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{y}_d$

$(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the pseudoinverse of $\mathbf{A}$

یادگیری در مدل ANFIS

بهینه‌سازی پارامترها در الگوریتم آموزش ANFIS

در الگوریتم آموزش ANFIS ارائه شده توسط Jang
هم پارامترهای مقدم قاعده و هم پارامترهای تالی قاعده بهینه‌سازی می‌شوند.

۱) در گذر پیش‌رو (forward pass):
پارامترهای تالی تنظیم می‌شوند در حالی که پارامترهای مقدم ثابت گرفته می‌شوند.

۲) در گذر پس‌رو (backward pass):
پارامترهای مقدم تنظیم می‌شوند در حالی که پارامترهای تالی ثابت گرفته می‌شوند.

ANFIS

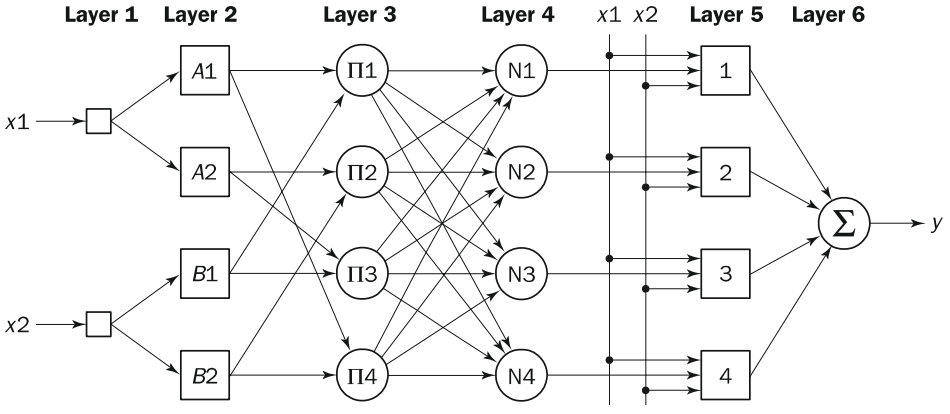
مثال: تقریب تابع با استفاده از مدل ANFIS (۱ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

در این مثال یک ANFIS برای دنبال کردن یک تراجکتوری از تابع غیرخطی زیر استفاده می‌شود:

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$

- ابتدا یک معماری مناسب برای ANFIS انتخاب می‌کنیم.
- این ANFIS باید دو ورودی x_1, x_2 و یک خروجی y داشته باشد.
- بنابراین، در این مثال ANFIS با ۴ قاعده با ساختار زیر تعریف می‌شود:



ANFIS

مثال: تقریب تابع با استفاده از مدل ANFIS (۲ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

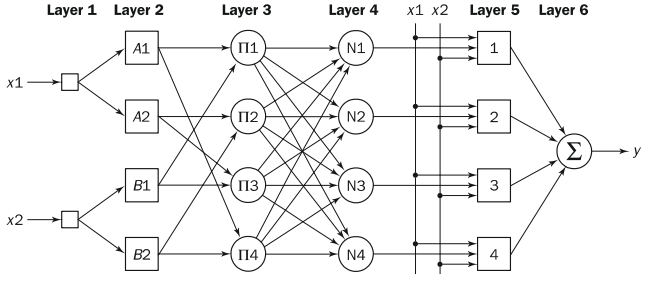
- داده‌های آموزشی این ANFIS حاوی 101 نمونه است که در قالب یک ماتریس 101×3 به صورت

$$[\boldsymbol{x1} \quad \boldsymbol{x2} \quad \boldsymbol{y}_d]$$

بازنمایی می‌شود.

- بردار ورودی اول $\boldsymbol{x1}$ از صفر شروع می‌شود، با گام‌های 0.1 افزایش می‌یابد و در 10 خاتمه می‌یابد.
- بردار ورودی دوم $\boldsymbol{x2}$ با گرفتن سینوس از هر عنصر بردار $\boldsymbol{x1}$ ساخته می‌شود.
- عناصر بردار خروجی مطلوب $\boldsymbol{y}_d$ با معادله‌ی تابع مشخص می‌شود.

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$



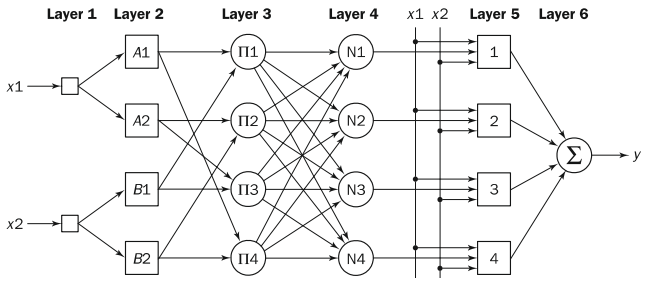
ANFIS

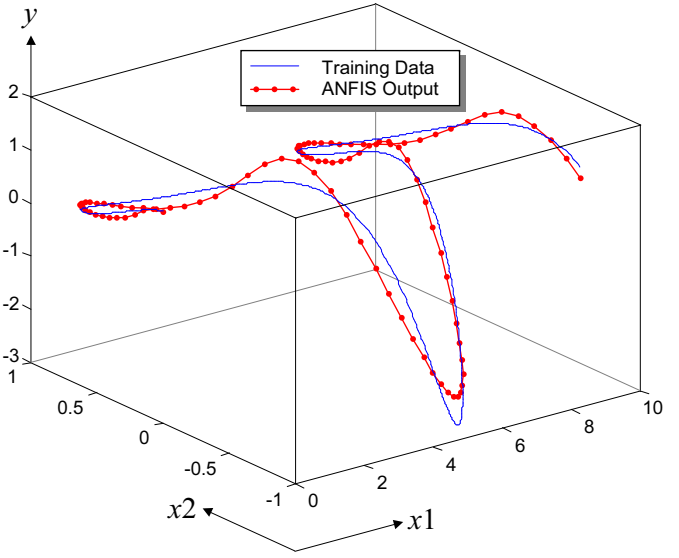
مثال: تقریب تابع با استفاده از مدل ANFIS (۳ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

یادگیری در یک ANFIS با دو تابع عضویت منتسب به هر ورودی
(۱ اپک)

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$





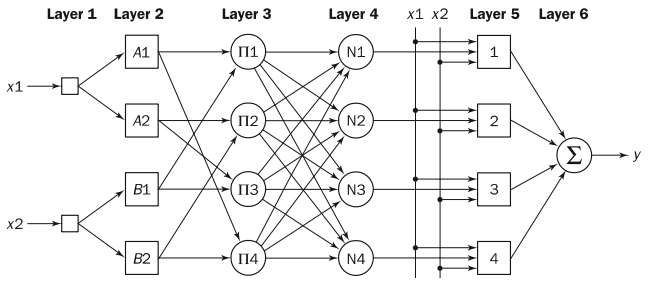
ANFIS

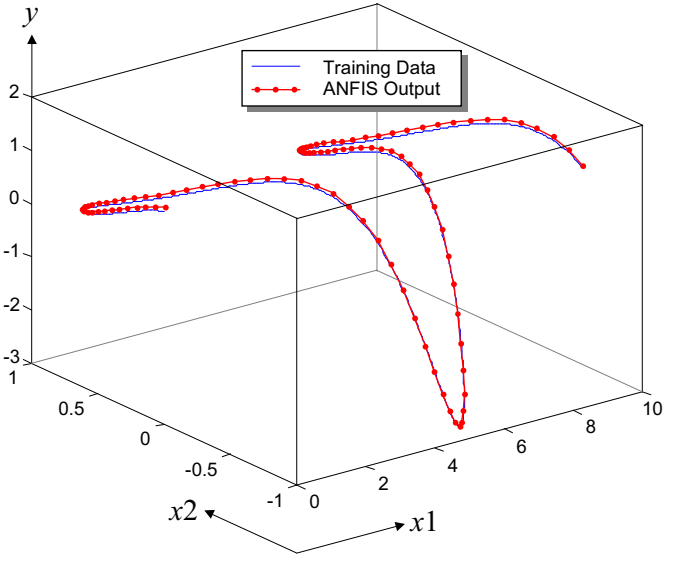
مثال: تقریب تابع با استفاده از مدل ANFIS (۴ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

یادگیری در یک ANFIS با دو تابع عضویت منتسب به هر ورودی
(۱۰۰ اپک)

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$



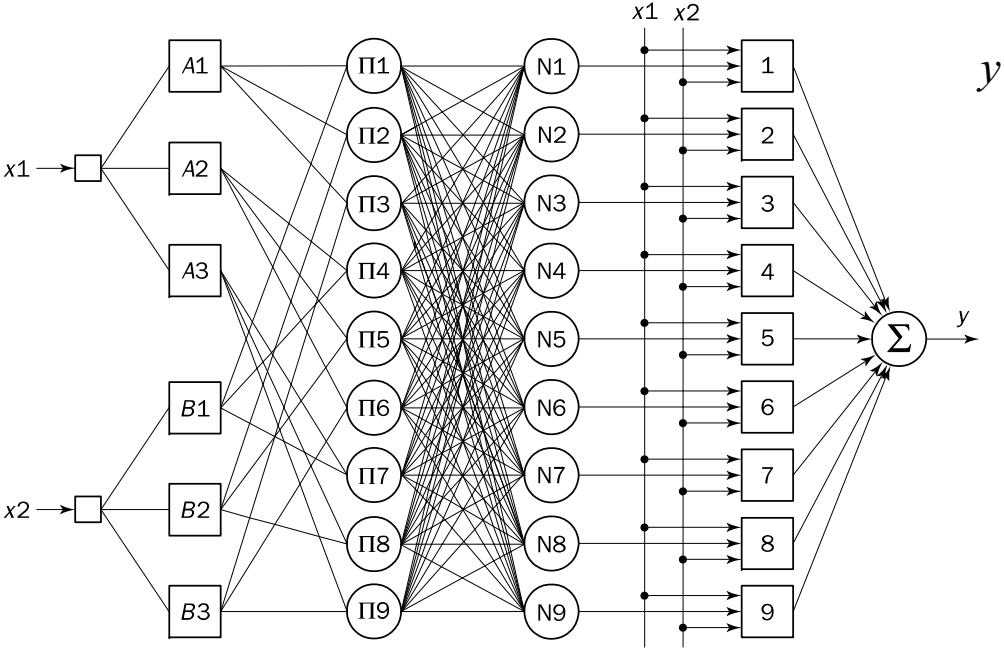


ANFIS

مثال: تقریب تابع با استفاده از مدل ANFIS (۵ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

می‌توان به بهبود بیشتر رسید، اما نتایج بهتر زمانی حاصل می‌شود که
سه تابع عضویت را به هر متغیر ورودی نسبت بدهیم. ⇐ یک ANFIS با ۹ قاعده



$$y = \frac{\cos(2\,x1)}{e^{x2}}$$

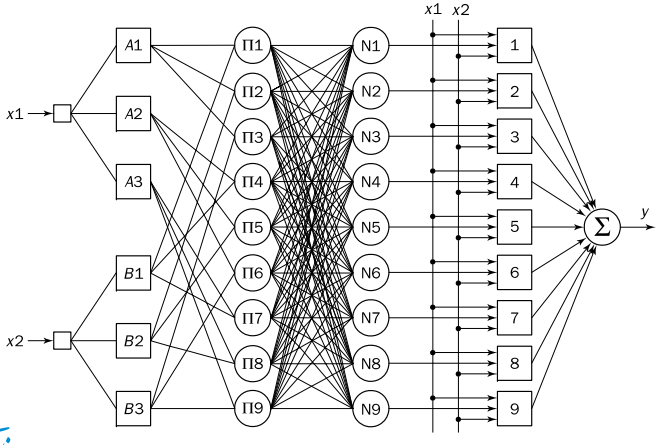
ANFIS

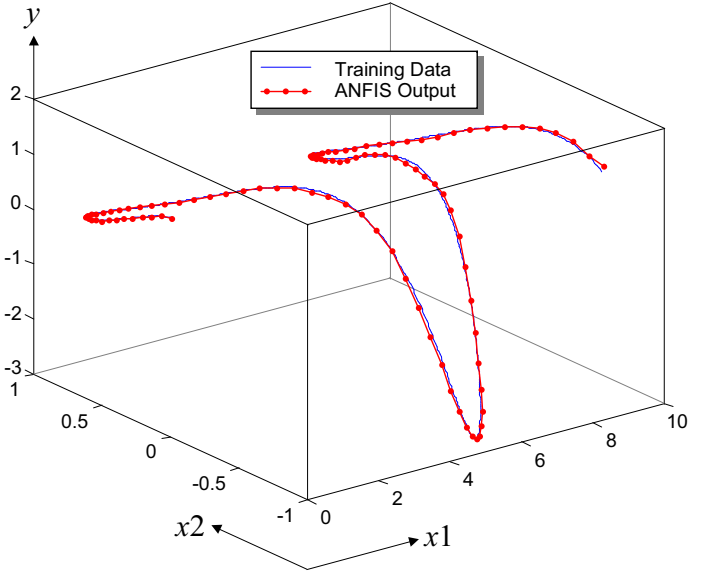
مثال: تقریب تابع با استفاده از مدل ANFIS (۶ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

یادگیری در یک ANFIS با سه تابع عضویت منتسب به هر ورودی
(۱ اپک)

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$





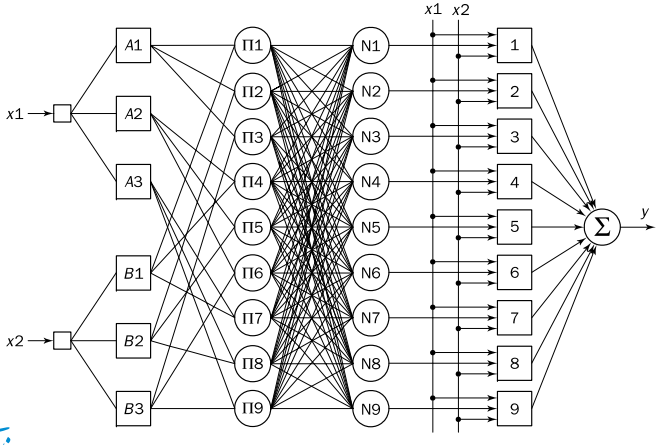
ANFIS

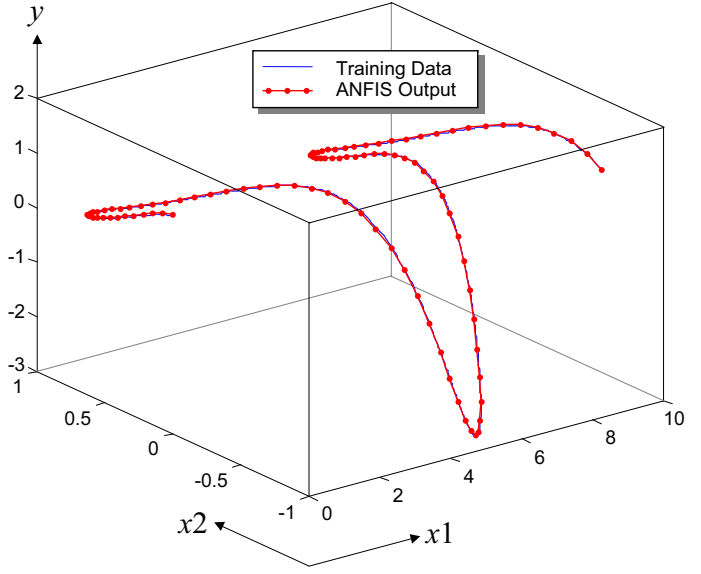
مثال: تقریب تابع با استفاده از مدل ANFIS (۷ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

یادگیری در یک ANFIS با سه تابع عضویت منتسب به هر ورودی
(۱۰۰ اپک)

$$y = \frac{\cos(2\,x1)}{e^{x2}}$$

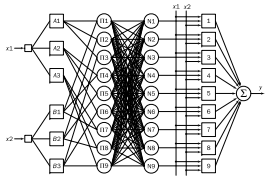




ANFIS

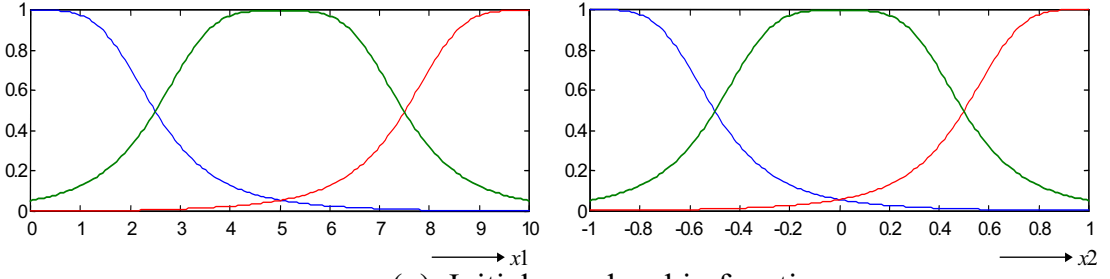
مثال: تقریب تابع با استفاده از مدل ANFIS (۸ از ۸)

FUNCTION APPROXIMATION USING THE ANFIS MODEL

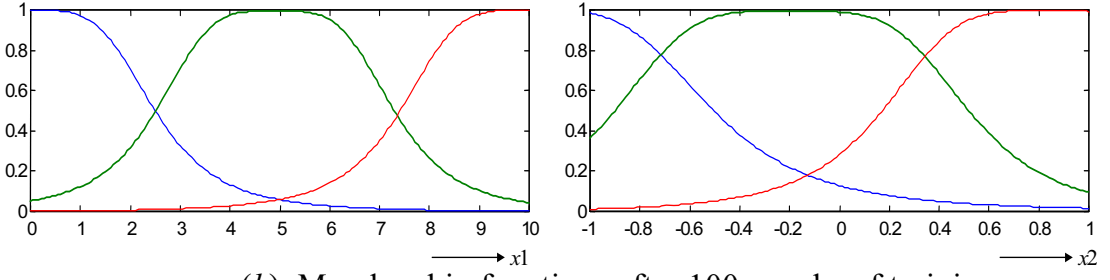


$$y = \frac{\cos(2\,x1)}{e^{x2}}$$

توابع عضویت آغازین و نهایی ANFIS



(*a*) Initial membership functions.



(*b*) Membership functions after 100 epochs of training.

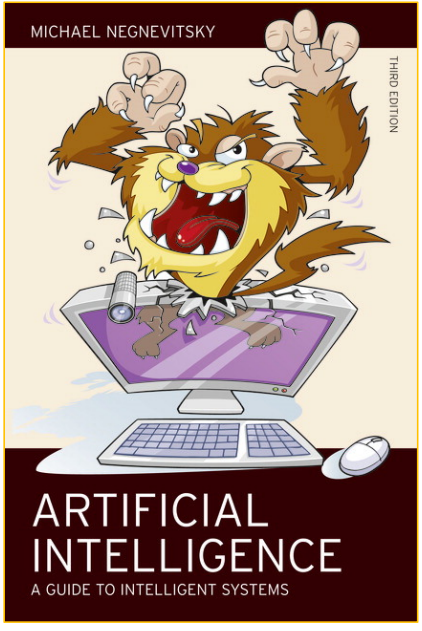
هوش مصنوعی

سیستم‌های عصبی–فازی

۴

منابع، مطالعه، تکلیف

منبع اصلی



Michael Negnevitsky,
Artificial Intelligence: A Guide to Intelligent Systems,
Pearson Education Canada, 2011.
Chapter 8 (8-3 : 8-4)

Hybrid intelligent systems 8

In which we consider the combination of expert systems, fuzzy logic, neural networks and evolutionary computation, and discuss the emergence of hybrid intelligent systems.

8.1 Introduction, or how to combine German mechanics with Italian love

In previous chapters, we considered several intelligent technologies, including probabilistic reasoning, fuzzy logic, neural networks and evolutionary computation. We discussed the strong and weak points of these technologies, and noticed that in many real-world applications we would need not only to acquire knowledge from various sources, but also to combine different intelligent technologies. The need for such a combination has led to the emergence of **hybrid intelligent systems**.

A hybrid intelligent system is one that combines at least two intelligent technologies. For example, combining a neural network with a fuzzy system results in a hybrid neuro-fuzzy system.

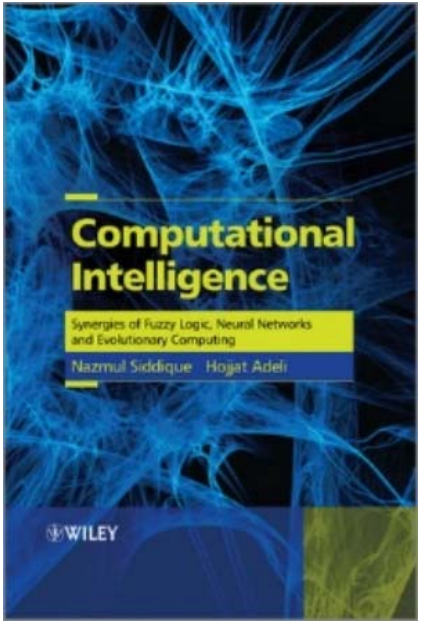
The combination of probabilistic reasoning, fuzzy logic, neural networks and evolutionary computation forms the core of **soft computing** (SC), an emerging approach to building hybrid intelligent systems capable of reasoning and learning in an uncertain and imprecise environment.

The potential of soft computing was first realised by Lotfi Zadeh, the 'father' of fuzzy logic. In March 1991, he established the Berkeley Initiative in Soft Computing. This group includes students, professors, employees of private and government organisations, and other individuals interested in soft computing. The rapid growth of the group suggests that the impact of soft computing on science and technology will be increasingly felt in coming years.

What do we mean by 'soft' computing?

While traditional or 'hard' computing uses **crisp values**, or numbers, soft computing deals with **soft values**, or fuzzy sets. Soft computing is capable of operating with uncertain, imprecise and incomplete information in a manner that reflects human thinking. In real life, humans normally use soft data

منبع دوم



Nazmul Siddique, Hojjat Adeli,
Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing,
John Wiley & Sons, 2013.
Chapter 10

10

Neural Fuzzy Systems

10.1 Introduction

In general, there are two main but apparently separate methodological developments relevant to computational intelligence: fuzzy logic systems and neural networks. Fuzzy logic systems try to emulate human-like reasoning using linguistic expression, whereas neural networks try to emulate the human brain-like learning and storing information on a purely experiential basis. Both the methodologies have been successfully applied in many complex and industrial processes though they experience a deficiency in knowledge acquisition.

The most important considerations in designing fuzzy systems are the construction of the membership functions (MF) and constructing the rule-base and have been a tiring process. The choice of MFs also plays a decisive role in the success of an application. But there is no automated way of constructing the MFs. They are mainly done by trial and error, or by human experts. As is well recognized, rule acquisition has been and continues to be regarded as a bottleneck for implementation of any kind of rule-based system. In most existing applications, fuzzy rules are generated by an expert in the area, especially for systems with only few inputs. With an increasing number of inputs, outputs and linguistic variables, the possible number of rules for the system increases exponentially, which makes it difficult for experts to define a complete set of rules and associated MFs for reasonable system performance. In Chapter 8, the construction of MFs, generation of rule base and tuning of scaling parameters using evolutionary algorithms were investigated. Evolutionary algorithms are the suitable choice where no *a priori* information about the MFs and the rule base is available. There have been many successful applications of evolutionary fuzzy systems reported in the literature, and Chapter 8 presents an overview of these techniques and their applications. As is well known, evolutionary algorithms are a slow process and the performance of an evolutionary algorithm depends inherently on the size of the population and the number of generations required for a solution to be robust for specific problems. Some designers may not like this. Then the problem is, if there is no expert knowledge available for constructing the MFs and the rule base, they must be constructed from environmental data, which may or may not be available. A second issue in fuzzy systems is processing of the rule base consisting of $R = n_1 \times n_2 \times \cdots \times n_N$ rules with n_i, $i = 1, 2, \ldots, N$, the number of MFs (primary fuzzy sets) for each of the N inputs. The processing of such a huge rule base is time-consuming. Consequently, computing

Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing, First Edition.
Nazmul Siddique and Hojjat Adeli.