ADAPTIVE NEURO-FUZZY INFERENCE SYSTEMS

RBFN and TS systems

Equivalent if the following hold:

- Both RBFN and TS use same aggregation method for output (weighted sum or weighted average)
- Number of basis functions in RBFN equals number of rules in TS
- TS uses Gaussian membership functions with same $\sigma$ as basis functions and rule firing is determined by multiplication
- RBFN response function ($c_i$) and TS rule consequents are equal


Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

- Takagi-Sugeno fuzzy system mapped onto a neural network structure.
- Different representations are possible, but one with 5 layers is the most common.
- Network nodes in different layers have different structures.

Consider a first-order Sugeno fuzzy model, with two inputs, x and y, and one output, z.

**Rule set**

- Rule 1: If x is $A_1$ and y is $B_1$, then $f_1 = p_1x + q_1y + r_1$
- Rule 2: If x is $A_2$ and y is $B_2$, then $f_2 = p_2x + q_2y + r_2$

Weighted fuzzy-mean:

$$ f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \bar{w}_1 f_1 + \bar{w}_2 f_2 $$
ANFIS architecture

- Corresponding equivalent ANFIS architecture:

ANFIS layers

- **Layer 1**: every node is an adaptive node with node function:
  \[ O_{1,j} = \mu_j(x_j) \]
  - Parameters in this layer are called *premise parameters*.
- **Layer 2**: every node is fixed whose output (representing firing strength) is the product of the inputs:
  \[ O_{2,j} = w_i = \prod_j \mu_j \]
- **Layer 3**: every node is fixed (normalization):
  \[ O_{3,j} = \bar{w}_j = \frac{w_j}{\sum_i w_j} \]
ANFIS layers

- **Layer 4**: every node is adaptive (consequent parameters):
  \[ O_{4,i} = O_{3,i}f_i = \bar{w}_i(p_0 + p_1x_1 + \ldots + p_nx_n) \]

- **Layer 5**: single node, sums up inputs:
  \[ O_{5,i} = \sum_i \bar{w}_if_i = \frac{\sum_i w_if_i}{\sum_i w_i} \]

> Adaptive network is functionally equivalent to a Sugeno fuzzy model!

---

ANFIS with multiple rules

![ANFIS with multiple rules diagram](image-url)
Consider the two rules ANFIS with two inputs $x$ and $y$ and one output $z$;

Let the premise parameters be fixed;

ANFIS output is given by linear combination of consequent parameters $p$, $q$ and $r$:

$$z = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2$$

$$= w_1 (p_1 x + q_1 y + r_1) + w_2 (p_2 x + q_2 y + r_2)$$

$$= (w_1 x) p_1 + (w_1 y) q_1 + (w_1) r_1 + (w_2 x) p_2 + (w_2 y) q_2 + (w_2) r_2$$

$$= A\theta$$

Partition total parameters set $S$ as:

- $S_1$: set of premise (nonlinear) parameters
- $S_2$: set of consequent (linear) parameters

$q$: unknown vector which elements are parameters in $S_2$

$z = Aq$ standard linear least-squares problem

**Best solution** for $q$ that minimizes $\|Aq - z\|^2$ is the least-squares estimator $q^*$:

$$q^* = (A^T A)^{-1} A^T z$$
Hybrid learning for ANFIS

- What if premise parameters are not optimal?
- Combine *steepest descent* and *least-squares estimator* to update parameters in adaptive network.
- Each *epoch* is composed of:
  1. **Forward pass**: node outputs go forward until Layer 4 and consequent parameters are identified by *least-squares estimator*;
  2. **Backward pass**: error signals propagate backward and the premise parameters are updated by *gradient descent*.

Hybrid learning for ANFIS

- Error signals: derivative of error measure with respect to each node output.

<table>
<thead>
<tr>
<th></th>
<th>Forward pass</th>
<th>Backward pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premise parameters</td>
<td>Fixed</td>
<td>Gradient descent</td>
</tr>
<tr>
<td>Consequent parameters</td>
<td>Least-squares estimator</td>
<td>Fixed</td>
</tr>
<tr>
<td>Signals</td>
<td>Node outputs</td>
<td>Error signals</td>
</tr>
</tbody>
</table>

- Hybrid approach converges much faster by reducing the search space of pure backpropagation method.
Stone-Weierstrass theorem

Let $D$ be a compact space of $N$ dimensions and let $\mathcal{F}$ be a set of continuous real-valued functions on $D$ satisfying:

1. **Identity function**: the constant $f(x) = 1$ is in $\mathcal{F}$.
2. **Separability**: for any two points $x_1 \neq x_2$ in $D$, there is an $f$ in $\mathcal{F}$ such that $f(x_1) \neq f(x_2)$.
3. **Algebraic closure**: if $f$ and $g$ are two functions in $\mathcal{F}$, then $fg$ and $af + bg$ are also in $\mathcal{F}$ for any reals $a$ and $b$.

Then, $\mathcal{F}$ is dense in the closure $C(D)$ of $D$, i.e.:

"$\epsilon > 0$, $g \in C(D)$, $\exists f \in \mathcal{F}: |g(x) - f(x)| < \epsilon$ $\forall x \in D$."

Universal approximator ANFIS

- According to Stone-Weierstrass theorem, an ANFIS has *unlimited approximation power* for matching any continuous nonlinear function arbitrarily well.
- **Identity**: obtained by having a constant consequent.
- **Separability**: obtained by selecting different parameters in the network.
Algebraic closure

- Consider two systems with two rules and final outputs:
  \[ z = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} \quad \text{and} \quad \hat{z} = \frac{\hat{w}_1 \hat{f}_1 + \hat{w}_2 \hat{f}_2}{\hat{w}_1 + \hat{w}_2} \]

- Additive:
  \[ az + b\hat{z} = a \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} + b \frac{\hat{w}_1 \hat{f}_1 + \hat{w}_2 \hat{f}_2}{\hat{w}_1 + \hat{w}_2} = \frac{w_1 \hat{w}_1 (af_1 + bf_1) + w_1 \hat{w}_2 (af_1 + bf_1) + w_2 \hat{w}_1 (af_2 + bf_2) + w_2 \hat{w}_2 (af_2 + bf_2)}{w_1 \hat{w}_1 + w_1 \hat{w}_2 + w_2 \hat{w}_1 + w_2 \hat{w}_2} \]

- Construct 4 rule inference system that computes:
  \[ az + b\hat{z} \]

Algebraic closure

- Multiplicative:
  \[ \hat{z} \hat{z} = \left( \frac{w_1 \hat{f}_1 + w_2 \hat{f}_2}{w_1 + w_2} \right) \left( \frac{\hat{w}_1 \hat{f}_1 + \hat{w}_2 \hat{f}_2}{\hat{w}_1 + \hat{w}_2} \right) = \frac{w_1 \hat{w}_1 \hat{f}_1 \hat{f}_1 + w_1 \hat{w}_2 \hat{f}_1 \hat{f}_2 + w_2 \hat{w}_1 \hat{f}_1 \hat{f}_2 + w_2 \hat{w}_2 \hat{f}_2 \hat{f}_2}{w_1 \hat{w}_1 + w_1 \hat{w}_2 + w_2 \hat{w}_1 + w_2 \hat{w}_2} \]

- Construct 4 rule inference system that computes:
  \[ \hat{z} \hat{z} \]
Model building guidelines

- Select number of fuzzy sets per variable:
  - empirically by examining data or trial and error
  - using clustering techniques
  - using regression trees (CART)
- Initially, distribute bell-shaped membership functions evenly:

Using an adaptive step size can speed up training.

How to design ANFIS?

- Initialization
  - Define number and type of inputs
  - Define number and type of outputs
  - Define number of rules and type of consequents
  - Define objective function and stop conditions
- Collect data
- Normalize inputs
- Determine initial rules
- Initialize network
TRAIN
Ex. 1: Two-input sinc function

\[ z = \text{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy} \]

- Input range: \([-10,10] \times [-10,10]\), 121 training data pairs.
- Multi-Layer Perceptron vs. ANFIS:
  - **MLP**: 18 neurons in hidden layer, 73 parameters, quick propagation (best learning algorithm for backpropagation MLP).
  - **ANFIS**: 16 rules, 4 membership functions per variable, 72 fitting parameters (48 linear, 24 nonlinear), hybrid learning rule.

MLP vs. ANFIS results

**Average of 10 runs:**
- **MLP**: different sets of initial random weights;
- **ANFIS**: 10 step sizes between 0.01 and 0.10.

**MLP’s approximation power decrease due to**: learning processes trapped in local minima or some neurons can be pushed into saturation during training.
ANFIS output

Training data

ANFIS Output

error curve

step size curve

ANFIS model

Initial MFs on X

Initial MFs on Y

Final MFs on X

Final MFs on Y
Ex. 2: 3-input nonlinear function

\[
\text{output} = \left(1 + x^{0.5} + y^{-1} + z^{-1.5}\right)^2
\]

- Two membership functions per variable, 8 rules
- Input ranges: \([1,6] \times [1,6] \times [1,6]\)
- 216 training data, 125 validation data

![Error curves](image1)

### ANFIS model

#### Initial MFs on X, Y and Z

#### Final MFs on X

#### Final MFs on Y

#### Final MFs on Z
Results comparison

APE = Average Percentage Error = \( \frac{1}{P} \sum_{i=1}^{P} \frac{|T(i) - O(i)|}{|T(i)|} \times 100\% \)

<table>
<thead>
<tr>
<th>Model</th>
<th>Training error</th>
<th>Checking error</th>
<th># Param.</th>
<th>Training data size</th>
<th>Checking data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS</td>
<td>0.043%</td>
<td>1.066%</td>
<td>50</td>
<td>216</td>
<td>125</td>
</tr>
<tr>
<td>GMDH model [1]</td>
<td>4.7%</td>
<td>5.7%</td>
<td>-</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Fuzzy model 1 [2]</td>
<td>1.5%</td>
<td>2.1%</td>
<td>22</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Fuzzy model 2 [2]</td>
<td>0.59%</td>
<td>3.4%</td>
<td>32</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>


Ex. 3: Modeling dynamic system

- Plant equation
  \( y(k+1) = 0.3y(k) + 0.6y(k-1) + f(u(k)) \)
- \( f(.) \) has the following form
  \( f(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u) \)
- Estimate nonlinear function \( F \) with ANFIS
  \( \hat{y}(k+1) = 0.3\hat{y}(k) + 0.6\hat{y}(k-1) + F(u(k)) \)
- Plant input: \( u(k) = \sin(2\pi k / 250) \)
- ANFIS parameters updated at each step (on-line)
- Learning rate: \( \eta = 0.1 \); forgetting factor: \( \lambda = 0.99 \)
- ANFIS can adapt even after the input changes
- Question: was the input signal rich enough?
Plant and model outputs

Effect of number of MFs

5 membership functions

f(u) and ANFIS Outputs

Each Rule's Outputs
Effect of number of MFs

4 membership functions

Initial MFs

Final MFs

f(u) and ANFIS Outputs

Each Rule’s Outputs

Effect of number of MFs

3 membership functions

Initial MFs

Final MFs

f(u) and ANFIS Outputs

Each Rule’s Outputs
Ex. 4: Chaotic time series

- Consider a chaotic time series generated by

\[ \dot{x}(t) = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t) \]

- Task: predict system output at some future instance \( t+P \) by using past outputs
- 500 training data, 500 validation data
- ANFIS input: \([x(t - 18), x(t - 12), x(t - 6), x(t)]\)
- ANFIS output: \(x(t + 6)\)
- Two MFs per variable, 16 rules
- 104 parameters (24 premise, 80 consequent)
- Data generated from \( t =118 \) to \( t =1117 \)

ANFIS model

- Final MFs on Input 1, \(x(t - 18)\)
- Final MFs on Input 2, \(x(t - 12)\)
- Final MFs on Input 3, \(x(t - 6)\)
- Final MFs on Input 4, \(x(t)\)
Model output

Error Curves
- Training Error
- Checking Error

Step Sizes

Potential and ANFIS Outputs

Prediction Errors

103rd order AR model

(a) Desired (Solid Line) and Predicted (Dashed Line) MG Time Series
(b) Prediction Errors
**Order selection**

\[ y^{(n)}(t) + y^{(n-1)}(t) + \cdots + y^{(1)}(t) + y(t) = u(t) \]

- Select optimal order of AR model in order to prevent overfitting
- Select the order that minimizes the error on a test set

---

**44th order AR model**

(a) Desired (Solid Line) and Predicted (Dashed Line) MG Time Series

(b) Prediction Errors
ANFIS output for P = 84

(a) Desired (Solid) and Predicted (Dashed) Time Series of ANFIS When P=84

(b) Prediction Errors

ANFIS extensions

- Different types of membership functions in layer 1
- Parameterized t-norms in layer 2
- Interpretability
  - constrained gradient descent optimization
  - bounds on fuzziness
  \[ E' = E + \beta \sum_{i=1}^{N_t} \overline{w}_i \ln(\overline{w}_i) \]
- parameterize to reflect constraints
- Structure identification