Dynamic Fuzzy Neural Networks: Architectures, Algorithms, and Applications

Meng Joo Er
School of Electrical and Electronic Engineering
Nanyang Technological University
Email: emjer@ntu.edu.sg

Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions
Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions

Introduction

Artificial Intelligence

Face Recognition

Self-Driving Car

IBM Watson

AlphaGo
Introduction

Review of Intelligent System Theory

- Fuzzy Systems
- Neural Networks
- Evolutionary Computation
- Expert System
- Data Mining
- ... ...

Fuzzy system + Neural network

Fuzzy Neural Network

Introduction

Fuzzy System

- **Rule set**: contains a number of fuzzy IF-THEN rules.
- **Label set**: defines the membership functions of the fuzzy sets used in the fuzzy rules.
- **Inference mechanism**: performs the inference operations on the rules.
- **Fuzzifier**: transforms the crisp inputs into degrees of match with linguistic values.
- **Defuzzifier**: transforms the fuzzy results of the inference into a crisp output.
Fuzzy Set

Membership function (MF) is used to describe the degree of belonging of an object

- **Triangular membership:**
  \[ \mu(x) = \begin{cases} 1 \frac{|x-m|}{\sigma} & \text{if } |x-m| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \]

- **Gaussian membership:**
  \[ \mu(x) = \exp\left[-\frac{(x-c)^2}{\sigma^2}\right] \]

Fuzzy Rule

Fuzzy systems are essentially rule-based expert systems, which consist of a set of linguistic rules in the form of “IF-THEN”.

- **Fuzzy IF-THEN rules:**
  \[ R_i: \text{IF } x_1 \text{ is } F_{11} \text{ and } \ldots \text{and } x_r \text{ is } F_{1r}, \text{THEN } y_1 \text{ is } G_{11} \text{ and } \ldots \text{and } y_s \text{ is } G_{1s} \]

- **Takagi-Sugeno-Kang model:**
  \[ R_i: \text{IF } x_1 \text{ is } F_{1i} \text{ and } \ldots \text{and } x_r \text{ is } F_{ri}, \text{THEN } y^i = \alpha_0^i + \alpha_1^i x_1 + \ldots + \alpha_r^i x_r \]

- **Advantages of TSK model:**
  - Computational efficiency
  - Works well with linear techniques
  - Works well with optimization and adaptive techniques
  - Guaranteed continuity of the output surface
  - Better suited to mathematical analysis
**Introduction**

**Fuzzy Inference**

Produce a crisp output by inference operations upon fuzzy IF-THEN rules

**Type I - Tsukamoto Fuzzy Models:**

The overall output is the weighted average of each rule's crisp output induced by the rule's firing strength $w_i$:

$$y = \frac{\sum_{i=1}^{u} y_i \times w_i}{\sum_{i=1}^{u} w_i}$$

**Type II - Mamdani Fuzzy Models:**

The overall fuzzy output is derived by applying "maximum" operation to the qualified fuzzy output

$$y = \frac{\sum_{i=1}^{u} \mu_y(w_i) \times w_i}{\sum_{i=1}^{u} \mu_y(w_i)}$$

(y centroid of area)
Type III - TSK Fuzzy Models:
The output of each rule is a linear combination of input variables plus a constant term and the final crisp output is the weighted average of each rule’s output

\[ y_x = \sum \sum \alpha i \times x_i \]

\[ y_w = \sum \sum \alpha i \times x_i \]

\[ y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} \]

Neural Networks

A neural network consists of many neurons and can be classified into feedforward (or non-recurrent) and feedback (or recurrent) neural networks

A neuron:

Each neuron performs two functions: aggregation of its inputs from other neurons or the external environment and generation of an output from the aggregated inputs.
**Properties of Neural Networks**

- An efficient method to approximate any nonlinear mapping
- Learning ability -- be trained from sample data
- Ability of generalization -- can respond correctly to any inputs which are not learned in the sample data
- Can operate simultaneously on both quantitative and qualitative data
- Naturally process many inputs and outputs and are readily applicable to multivariable systems
- Highly parallel implementation

*One of the most salient features of neural networks lies in its learning capability from samples*

**Learning Algorithms**

- **Supervised learning**: Data and corresponding labels are given
- **Unsupervised learning**: Only data are given, no labels provided
- **Semi-supervised learning**: Some (if not all) labels are provided
- **Reinforcement learning**: Only provides a scalar performance measure without indicating the direction in which the system could be improved
RBF Neural Networks

Neural network formula:

\[ y(X) = w_0 + \sum_{i=1}^{u} w_i R(||X - C_i||) \]

- Thin-plate-spline function: \( R(x) = x^2 \log(x) \)
- Multiquadric function: \( R(x) = (x^2 + \sigma^2)^{1/2} \)
- Inverse multiquadric function: \( R(x) = (x^2 + \sigma^2)^{-1/2} \)
- Gaussian function: \( R(x) = \exp\left(-\frac{x^2}{\sigma^2}\right) \)

Comparison

<table>
<thead>
<tr>
<th>Skills</th>
<th>Fuzzy System</th>
<th>Neural Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge acquisition</td>
<td>human expert</td>
<td>sample sets</td>
</tr>
<tr>
<td></td>
<td>interaction algorithm</td>
<td></td>
</tr>
<tr>
<td>Uncertainty</td>
<td>quantitative and qualitative</td>
<td>quantitative</td>
</tr>
<tr>
<td></td>
<td>decision making</td>
<td>perception</td>
</tr>
<tr>
<td>Reasoning</td>
<td>heuristic search</td>
<td>parallel computation</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Speed</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Fault-tolerance</td>
<td>low</td>
<td>very high</td>
</tr>
<tr>
<td>Learning</td>
<td>induction</td>
<td>adjusting weights</td>
</tr>
<tr>
<td>Natural language</td>
<td>explicit</td>
<td>implicit</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>
Introduction

**Fuzzy Neural Networks**

**Fuzzy System**
- Dependent on the specification of good rules by human experts
- No formal framework for the choice of various parameters

**Neural Networks**
- No easy way to predict the output
- Connecting weights invariably shed no light on what the network will do with the data

**Fuzzy Neural Networks**
Capture capabilities of both types of systems and overcome drawbacks of each system
FNNs were used to:
- Tune fuzzy systems
- Extract fuzzy rules from given numerical examples
- Develop hybrid systems combining neural networks and fuzzy systems

**Contents**

- Introduction
- **Dynamic Fuzzy Neural Networks**
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions
Dynamic Fuzzy Neural Network

Motivation

- Lack of systematic design for membership functions
- Lack of adaptability for reasoning environment changes
- Structure identification is time-consuming

- Dynamic Fuzzy Neural Network (D-FNN)
  - Pruning technology
  - Hierarchical learning
  - Extended RBF NN
  - Hierarchical on-line self-organizing learning
  - Adaptive structure identification
  - Fast learning speed

Architecture of D-FNN
Architecture of D-FNN

Layer 1: Each node in layer 1 represents an input linguistic variable.

Layer 2: Each node in layer 2 represents a membership function (MF) which is a Gaussian function given by:

\[ \mu_i(x) = \exp\left[-\frac{(x - c_i)^2}{\sigma_i^2}\right] \quad i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, u \]
**Architecture of D-FNN**

Layer 3: Each node in layer 3 represents a possible IF-part of fuzzy rules.

Output of \( j \)th RBF unit: 
\[
\phi_j = \exp \left( -\frac{\|X - C_j\|^2}{\sigma_j^2} \right) \quad j = 1, 2, \cdots, u
\]

Layer 4: We refer to these nodes as N (normalized) nodes
\[
\psi_j = \frac{\phi_j}{\sum_{k=1}^{u} \phi_k} \quad j = 1, 2, \cdots, u
\]
Architecture of D-FNN

Layer 5: Each node in this layer represents an output variable which is the summation of incoming signals

$$y(X) = \sum_{k=1}^{u} w_k \cdot \psi_k$$

TSK model: $$w_k = \alpha_{k0} + \alpha_{k1} x_1 + \cdots + \alpha_{kr} x_r$$

Learning Algorithm of the D-FNN

- Rule Generation
- Allocation of RBF Unit Parameters
- Weight Adjustment
- Rule Pruning
### Learning Algorithm of the D-FNN

#### Error criterion:
For observation \((X_i, t_i)\), define
\[
\|e_i\| = \|t_i - y_i\| > k_e?
\]
An RBF unit should be considered.

#### Accommodation criterion:
\[
d_i(j) = \|X_i - C_j\|, \quad j = 1, 2, \ldots, u
\]
If \(d_{min} = \arg \min(d_i(j)) > k_d\), an RBF unit should be considered.

#### Hierarchical learning:
Accommodation boundary is adjusted dynamically
\[
k_e = \max\{e_{max} \times \beta, e_{min}\}
k_d = \max\{d_{max} \times \gamma, d_{min}\}
\]
Coarse learning to fine learning

#### Allocation of premise parameter:
\[
C_i = X_i, \quad \sigma_i = k \times d_{min}
\]

| Case 1: \(\|e_i\| > k_e, \quad d_{min} > k_d\) | Rule generation |
| Case 2: \(\|e_i\| \leq k_e, \quad d_{min} \leq k_d\) | Do nothing or only update consequent parameters |
| Case 3: \(\|e_i\| \leq k_e, \quad d_{min} > k_d\) | Only adjust consequent parameters |
| Case 4: \(\|e_i\| > k_e, \quad d_{min} \leq k_d\) | Update the width of the nearest RBF node and all the weights |

For the \(k\)th nearest RBF unit:
\[
\sigma_k^i = k_w \times \sigma_k^{i-1}
\]
Learning Algorithm of the D-FNN

**Rule Generation**

**Allocation of RBF Unit Parameters**

**Weight Adjustment**

**Rule Pruning**

Assume $n$th training pattern enters the D-FNN and all these $n$ data are memorized by the D-FNN in the input matrix $P$ and the output matrix $T$, respectively, upon which the weights are determined.

Give the output in the compact form as follows:

$$ Y = W \times \Psi $$

where $W$ is the weight matrix, $\Psi$ is the output matrix of $N$ nodes.

The objective is to find an optimal $W^*$ such that $E^*E$ is minimized.

This problem can be solved by the well-known Linear Least Square (LLS) method by approximating:

$$ W^* = T(\Psi^T\Psi)^{-1}\Psi^T $$

**Error Reduction Ratio (ERR) Method:**

$$ D = H\theta + E $$

where $D = T^T$, $H = \Psi^T$, and $\theta = W^T$.

**QR decomposition:**

$$ H = QA $$

where $Q = (q_1, q_2, \cdots, q_v) \in \mathbb{R}^{v \times v}$

**ERR:**

$$ err_i = \frac{(q_i^T D)^2}{q_i^T q_j D^T D} \quad i \leq v $$

The ERR offers a simple and effective means of seeking a subset of significant regressors.
Learning Algorithm of the D-FNN

Determination of RBF Units:

Define the ERR matrix as

$$\Delta = (\delta_1, \delta_2, \ldots, \delta_u) \in \mathbb{R}^{(r+1) \times u}$$

Significance of the $i$th rule is defined as

$$\eta_i = \sqrt{\frac{\delta_i^T \delta_i}{r+1}}$$

If

$$\eta_i < k_{err}$$

then the $i$th rule is deleted.

---

Flowchart of D-FNN Algorithm

1. **Initialization**
   - Generate first rule
   - Compute distance and find $d_{min}$
   - Compute actual output error $e_i$
   - If $d_{min} > k_d$?
   - If $e_i > k_e$?

2. **Generate a new rule**
   - Compute ERR
   - If $\eta_i > k_{err}$?

3. **Delete $i$th rule**
   - Adjust widths
   - Adjust consequents parameters

4. **Observations completed?**
   - If $e_i \leq k_e$?

5. **End**
Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions

Motivation

Adaptive noise cancellation (ANC) using an adaptive filter:

\[ y(k) = x(k) + d(k) \]

\[ \hat{d}(k) \]

\[ \hat{x}(k) = x(k) + d(k) - \hat{d}(k) \]

D-FNN cannot be directly applied to ANC, because the neuron generation depends on the systems error, which cannot be defined for ANC.
Motivation

- The method based on accommodation boundary and system error could hardly be evaluated online.
- Linear Least Square (LLS) method is not very effective in processing signals online in a very noisy environment.

Online self-organizing mapping (SOM)  
Recursive Linear Least Square (RLLS) Estimator or Kalman Filter method

ED-FNN Learning Algorithms

The ultimate objective is to implement the D-FNN learning algorithm in real time.

Input Space Partitioning: Online SOM is performed for every training pattern $(X(k), y(k))$.

Best Matching Neuron Center:

$$||X(k) - C_j(k)|| = \min_j ||X(k) - C_j(k)||$$

Update rule:

$$C_j(k+1) = C_j(k) + \alpha(k)h_{bi}(k)[X(k) - C_j(k)]$$

where $k$ is the $k$th input pattern, $\alpha(k)$ is the adaptation coefficient, and $h_{bi}(k)$, the neighborhood kernel centered on the winner unit, is given by

$$h_{bi}(k) = \exp\left(-\frac{||C_i - C_j||^2}{2\sigma^2(k)}\right)$$
**ED-FNN Learning Algorithms**

**Adaptation of the Output Linear Weights:** Recursive Linear Least Square (RLLS) estimator is used to adjust the weights.

\[
S_0 = (\Psi_{i-1}^T \Psi_{i-1})^{-1}
\]

\[
S_i = S_{i-1} \Psi_i^T \Psi_i S_{i-1}^{-1} + S_{i-1}^{-1} - S_{i-1}^{-1} \Psi_i^T \Psi_i S_{i-1}^{-1} S_{i-1}^{-1} \Psi_i^T \Psi_i S_{i-1}^{-1} S_{i-1}^{-1} \Psi_i^T \Psi_i S_{i-1}^{-1}
\]

\[
W_i = W_{i-1} + S_{i-1}^{-1} \Psi_i^T (T_i - \Psi_i W_{i-1})
\]

\( i = 1, 2, \ldots, n \)

where \( S_i \) is the error covariance matrix for the \( i \)th observation.

**Flowchart of ED-FNN Algorithm**

1. **Start**
2. Initialization
3. Generate first rule
4. SOM to update center clustering
5. Compute distance and find \( d_{\text{min}} \)
   - If \( d_{\text{min}} > k_d \) then go to \( N \), else go to \( Y \)
6. Determine parameters of new rule
   - LLS to adjust weights
7. First observation completed?
   - If \( N \) then go to \( N \), else go to \( Y \)
   - If \( Y \) then go to \( Y \)
   - LLS to adjust weights
8. Observations completed?
   - If \( N \) then go to \( N \), else go to \( Y \)
   - If \( Y \) then go to End

End
Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions

Motivation

- Widths of Gaussian MFs are the same
  - Do not coincide with the reality

- The number of MFs is irrespective of MF distribution
  - Result in significant overlapping and be opaque for users to understand

- The width of a new Gaussian MF is only determined by $d_{\text{min}}$ and $k$
  - If $k$ and $d_{\text{min}}$ is too large, the widths will be very large.

- Several prespecified parameters are selected randomly
  - Not easy for users to implement
Structure and parameters identification are performed automatically and simultaneously without partitioning input space and selecting initial parameters a priori.

Fuzzy rules can be recruited or deleted dynamically.

Fuzzy rules can be generated quickly without resorting to the BP iteration learning.

**Architecture of GD-FNN**
Corollary 1: The proposed GD-FNN is functionally equivalent to the TSK fuzzy system if the following conditions are true:

- The number of receptive field units of the GD-FNN is equal to the number of fuzzy IF-THEN rules.
- The output of each fuzzy IF-THEN rule is a linear combination of input variables plus a constant term.
- The membership functions within each rule are chosen as Gaussian functions.
- The T-norm operator used to compute each rule's firing strength is multiplication.
- Both the GD-FNN and the TSK fuzzy inference system under consideration use the same method, i.e., either weighted average or weighted sum to derive their overall outputs.

Learning Algorithm of the GD-FNN

- Rule Generation: Propose criteria of rule generation.
- Premise Parameter Estimation: Allocate parameters of new rules.
- Sensitivity Analysis: Analyze sensitivity of input variables and fuzzy rules and prune rules.
- Width Modification: Adjust width to obtain a better local approximation.
- Consequent Parameter Determination: Determine the optimal parameters $W^*$.
Generalized Dynamic Fuzzy Neural Networks

Rule Generation

- System Errors

\[ \| e_k \| = \| t_k - y_k \| \]

If \( \| e_k \| > k_e \), a new rule should be considered, where

\[ k_e = \begin{cases} 
  e_{\max} & 1 < k < n/3 \\
  \max \{ e_{\max} \times \beta^k, e_{\min} \} & n/3 \leq k \leq 2n/3 \\
  e_{\min} & 2n/3 < k \leq n 
\end{cases} \]

where \( e_{\min} \) is the desired accuracy of the output of the GD-FNN, \( e_{\max} \) is the maximum error chosen, \( k \) is the learning epoch, and \( \beta \in (0, 1) \), called convergence constant is given by

\[ \beta = \left( \frac{e_{\min}}{e_{\max}} \right)^{3/n} \]

- \( \varepsilon \)-Completeness of Fuzzy Rules

Definition (\( \varepsilon \)-Completeness of Fuzzy Rules): For any input in the operating range, there exists at least one fuzzy rule so that the match degree (or firing strength) is no less than \( \varepsilon \).

Calculate Mahalanobis distance (M-distance)

\[ m_d_k(j) = \sqrt{\left( X - C_j \right)^T \Sigma_j^{-1} \left( X - C_j \right)} \]

Find

\[ J = \arg \min_{1 \leq j \leq n} (md_k(j)) \]

If \( md_{k,\min} = md_k(J) > k_d \)

which implies that the existing system is not satisfied with \( \varepsilon \)-completeness and a new rule should be considered.

Premise Parameter Estimation

Definition (Semi-Closed Fuzzy Sets): Let $U = [a, b]$ be the universe of discourse of input $x$. If each fuzzy set $A_i = \{ (x, \mu_i(x)) | x \in U \}$ ($i = 1, 2, \ldots, m$) is represented as the Gaussian function
\[ \mu_i(x) = \exp \left( -\frac{(x-c_i)^2}{\sigma_i^2} \right) \]
for all $x \in U$, and satisfies the following boundary conditions simultaneously:
\[ \mu_i(x) = \exp \left( -\frac{(x-c_i)^2}{\sigma_i^2} \right) \text{ if } |c_i - a| \leq \delta \]
\[ \mu_i(x) = \exp \left( -\frac{(x-b)^2}{\sigma_i^2} \right) \text{ if } |c_i - b| \leq \delta \]
\[ \mu_i(x) = \exp \left( -\frac{(x-c_i)^2}{\sigma_i^2} \right) \text{ if } |c_i - a| > \delta \text{ and } |c_i - b| > \delta \]
where $\delta$ is a tolerable small value, and the widths of Gaussian functions are selected as
\[ \sigma_i(x) = \frac{\max(|c_i - c_{i-1}|, |c_i - c_{i+1}|)}{\sqrt{\ln(1/\varepsilon)}} \]
where $c_{i-1}$ and $c_{i+1}$ are the two centers of neighboring MFs of $i$th membership function. Then, we call $A_i$ semi-closed fuzzy sets.

Theorem: The semi-closed fuzzy set $A_i$ satisfy $\varepsilon$-completeness of fuzzy rules, i.e., for all $x \in U$, there exists $i \in \{1, 2, \ldots, m\}$, such that $\mu_i(x) \geq \varepsilon$.

Calculate E-distance:
\[ ed_i(j) = |x_i^k - \Phi_i(j)| \]
where $\Phi_i \in \{ x_{i_{\min}}, c_{i1}, c_{i2}, \ldots, c_{iu}, x_{i_{\max}} \}$ and find
\[ f_n = \arg \min_{j=1,2,\ldots,u+2} (ed_i(j)) \]
If $ed_i(f_n) \leq k_{mf}$, $x_i^k$ can be completely represented by the existing fuzzy set $A_{i,j_n}(c_{i_{f_n}}, \sigma_{i_{f_n}})$ without generating a new MF. Otherwise, a new Gaussian MF is allocated with its width and center respectively being determined by
\[ \sigma_i(x) = \frac{\max(|c_i - c_{i-1}|, |c_i - c_{i+1}|)}{\sqrt{\ln(1/\varepsilon)}} \]
where $c_{i+1} = x_i^k$.
## Sensitivity Analysis

### Error Reduction Ratio

Linear regression model:

$$D = H\theta + E$$

where $D = T^T$, $H = \Psi^T$, and $\theta = W^T$.

QR decomposition:

$$H = QA$$

where $Q = (q_1, q_2, \cdots, q_v) \in \mathbb{R}^{n \times v}$

ERR:

$$err_i = \frac{(q_i^T D)^2}{q_i^T q_i D^T D} \quad i \leq v$$

### Sensitivity of Fuzzy Rules

Define the ERR matrix

$$\Delta = (\delta_1, \delta_2, \cdots, \delta_u) \in \mathbb{R}^{(r+1) \times u}$$

Significance of the $i$th rule is defined as

$$\eta_i = \sqrt{\delta_i^2/\delta_i}$$

If

$$\eta_i < k_{err} \quad i = 1, 2, \cdots, u$$

then the $i$th rule is deleted.
**Sensitivity Analysis**

- **Sensitivity of Input Variables**

Define

\[ B_j = \sum_{k=2}^{r+1} \rho_j(k) \quad j = 1, 2, \ldots, u \]

\[ B_{ij} = \frac{err_{ij}}{B_j} \quad i = 1, 2, \ldots, r \]

- \( B_j \) -- total ERR related to input variables in \( j \)th rule
- \( err_{ij} \) -- ERR corresponding to \( i \)th input variable in \( j \)th rule
- \( B_{ij} \) -- The significance of \( i \)th input variable in \( j \)th rule

If \( B_{ij} \) is small, \( i \)th input variable is not sensitive to the system output, and consequently, width of the hyperellipsoidal region in \( i \)th input variable could be reduced without significant effect of system performance.

---

**Width Modification**

In the case \( \|e_k\| > k_e \) and \( md_{k,\min} \leq k_d \), which implies the significance of the rule is not good to accommodate all the patterns, and the ellipsoidal region should be reduced.

Width modification:

\[ \sigma_{ij}^{\text{new}} = \zeta \times \sigma_{ij}^{\text{old}} \]

where

\[ \zeta = \begin{cases} 1 & \text{if } B_j < 1/r \\ \frac{1}{1 + k_u (B_j - 1/r)^2} & \text{if } B_j \geq 1/r \end{cases} \]
Consequent Parameter Determination

Give the output in the compact form as follows:

\[ W\phi = Y \]

Assume that the desired output is:

\[ T = (t_1, t_2 \cdots t_n) \in \mathbb{R}^n \]

Minimizing

\[ \|W\phi - T\|_2 \]

The optimal \( W^* \) can be determined by

\[ W^* = T (\phi^T \phi)^{-1} \phi^T \]

Flowchart of GD-FNN Algorithm
Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions

Applications

Application 1: Mackey—Glass Time-Series Prediction

Time-series prediction is widely applied in economic and business planning, inventory and production control, weather forecasting, signal processing, control etc.

System model:

\[ x(t + 1) = (1 - a)x(t) + \frac{bx(t - \tau)}{1 + x^{10}(t - \tau)} \]

where \( a = 0.1, b = 0.2, \tau = 17 \).

Prediction model:

\[ x(t + p) = f[x(t), x(t - \Delta t), x(t - 2\Delta t), x(t - 3\Delta t)] \]

6000 exemplar samples were generated between \( t = 0 \) and \( t = 6000 \) with initial conditions \( x(t) = 0 \) for \( t < 0 \) and \( x(0) = 1.2 \).
Applications

Application 1: Mackey—Glass Time-Series Prediction

Mackey—Glass time-series prediction (from \( t = 118 \) to 617 and six-step-ahead prediction)

Generalization test of the D-FNN for Mackey—Glass time-series prediction (from \( t = 642 \) to 1141 and six-step ahead prediction)
Comparisons of structure and performance of different algorithms (Training case: n = 500, p = 6)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Number of rules</th>
<th>Number of parameters</th>
<th>RMSE of training</th>
<th>RMSE of testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-FNN</td>
<td>10</td>
<td>100</td>
<td>0.0082</td>
<td>0.0083</td>
</tr>
<tr>
<td>ANFIS</td>
<td>16</td>
<td>104</td>
<td>0.0016</td>
<td>0.0015</td>
</tr>
<tr>
<td>OLS</td>
<td>35</td>
<td>211</td>
<td>0.0087</td>
<td>0.0089</td>
</tr>
<tr>
<td>RBF-AFS</td>
<td>21</td>
<td>210</td>
<td>0.0107</td>
<td>0.0128</td>
</tr>
</tbody>
</table>

Remark: It is shown that the performance of the D-FNN is not so good as that of the ANFIS, even the D-FNN has more adjustable parameters. The reason is that the ANFIS is trained by iterative learning so that an overall optimal solution can be achieved, whereas the D-FNN only acquires a sub-optimal result.
### Application 1: Mackey—Glass Time-Series Prediction

Generalization comparison between D-FNN and RAN, RANEKF and M-RAN for $p=50$

<table>
<thead>
<tr>
<th>Technique</th>
<th>Pattern length</th>
<th>Number of rules</th>
<th>Number of parameters</th>
<th>NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-FNN</td>
<td>2000</td>
<td>25</td>
<td>250</td>
<td>0.0544</td>
</tr>
<tr>
<td>RAN</td>
<td>5000</td>
<td>50</td>
<td>301</td>
<td>0.071</td>
</tr>
<tr>
<td>RANEKF</td>
<td>5000</td>
<td>56</td>
<td>337</td>
<td>0.056</td>
</tr>
<tr>
<td>M-RAN</td>
<td>5000</td>
<td>28</td>
<td>169</td>
<td>—*</td>
</tr>
</tbody>
</table>

*The generalization error is measured by weighted prediction error (WPE)*

---

### Application 2: Prediction of Machine Health Condition

An online dynamic fuzzy neural network (OD-FNN) is developed for prediction of machine health condition.

**Experimental platform PRONOSTIA**

![Experimental platform PRONOSTIA](image-url)
Application 2: Prediction of Machine Health Condition

Prediction model:

\[
y_s(k + r) = f(y_s(k), y_s(k - r), y_s(k - 2r), \ldots, y_s(k - nr),
\]
\[
x_s(k), x_s(k - r), x_s(k - 2r), \ldots, x_s(k - nr))
\]

where \(x_s\) and \(y_s\) are the input and target signals respectively, \(r\) is the prediction step, and \(n + 1\) is maximum lag.

500 data are applied to initialize the OD-FNN and 500 data pairs are applied to test the OD-FNN performance for online prediction.

Applications

Application 2: Prediction of Machine Health Condition

Linear model online prediction results: (a) Prediction output with error curves. (b) Accuracies of online training and prediction.

\[
y_s(i) = 5x_s(i) + \text{rand}(t)
\]

White noise signal
Application 2: Prediction of Machine Health Condition

Bearing MHC online prediction results: (a) Prediction output with error curves. (b) Accuracies of online training and prediction.

Performance comparisons of all prediction methods

<table>
<thead>
<tr>
<th>Case</th>
<th>$t_{com}$</th>
<th>$\varphi_{train}$</th>
<th>$\varphi_{pred}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSBP-NN</td>
<td>8.2471</td>
<td>99.52</td>
<td>99.31</td>
</tr>
<tr>
<td>OSPELM-FNN</td>
<td>1.3841</td>
<td>99.23</td>
<td>98.28</td>
</tr>
<tr>
<td>RRLS-NN</td>
<td>5.2681</td>
<td>99.43</td>
<td><strong>99.34</strong></td>
</tr>
<tr>
<td>OD-FNN</td>
<td>1.6833</td>
<td>99.31</td>
<td>99.18</td>
</tr>
<tr>
<td>Bearing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSBP-NN</td>
<td>4.0210</td>
<td><strong>98.38</strong></td>
<td>96.01</td>
</tr>
<tr>
<td>OSPELM-FNN</td>
<td>0.6765</td>
<td>97.31</td>
<td>95.35</td>
</tr>
<tr>
<td>RRLS-NN</td>
<td>2.5951</td>
<td>97.34</td>
<td>95.89</td>
</tr>
<tr>
<td>OD-FNN</td>
<td>1.6833</td>
<td>97.94</td>
<td><strong>97.54</strong></td>
</tr>
</tbody>
</table>

$t_{com}$ — total computational time; $\varphi_{train}$ — online training accuracy; $\varphi_{pred}$ — online prediction accuracy.

Achieve a good trade-off between prediction accuracy and computational cost.

Best online prediction accuracy.
Articulated two-link robot manipulator

Manipulator dynamics:

\[ M(q) \ddot{q} + Q(q, \dot{q}) + r_d = r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \]

where

\[ r_1 = [(I_1 + m_l l_1^2) + I_2 + m_l l_2^2 + m_f l_f^2 + 2(m_f l_f \cos \delta_f) \cos q_2 + 2(m_f l_f \sin \delta_f) \sin q_1] q_1^{\ddagger} \]
\[ + [(I_2 + m_f l_f^2) + (m_f l_f \cos \delta_f) \cos q_2 + (m_f l_f \sin \delta_f) \sin q_1] q_2^{\ddagger} \]
\[ - [(m_f l_f \cos \delta_f) \sin q_2 - (m_f l_f \sin \delta_f) \cos q_1] q_1 q_2 \]
\[ + r_{a1} \]
\[ r_2 = [(I_2 + m_f l_f^2) + (m_f l_f \cos \delta_f) \cos q_2 + (m_f l_f \sin \delta_f) \sin q_1] q_1^{\ddagger} \dot{q}_1 + (I_2 + m_f l_f^2) q_2^{\ddagger} \]
\[ + [(m_f l_f \cos \delta_f) \sin q_2 - (m_f l_f \sin \delta_f) \cos q_1] q_1 q_2 \]
\[ + r_{a2} \]
Application 3: Online Manipulator Control

Online adaptive fuzzy neural control system:

Performance comparison of GD-FNN with PD and (adaptive fuzzy controller) AFC controllers:

<table>
<thead>
<tr>
<th></th>
<th>PD</th>
<th>AFC</th>
<th>GD-FNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>K_p</td>
<td>K_v</td>
<td>K_p</td>
<td>K_v</td>
</tr>
<tr>
<td></td>
<td>q_1</td>
<td>q_2</td>
<td>q_1</td>
</tr>
<tr>
<td></td>
<td>0.050</td>
<td>0.018</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>9.3</td>
<td>7.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.287</td>
<td>0.254</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>312.7</td>
<td>109.9</td>
<td>280.3</td>
</tr>
</tbody>
</table>

Steady State Error (rad) | Overshoot (%) | Rise Time (sec) | Maximum Control Torque (Nm)
Application 3: Online Manipulator Control

Trajectory tracking results of adaptive online control

Convergence of tracking error
Outline

- Introduction
- Dynamic Fuzzy Neural Networks
- Enhanced Dynamic Fuzzy Neural Networks
- Generalized Dynamic Fuzzy Neural Networks
- Applications
- Conclusions

Conclusions

- A D-FNN based on extended Radial Basis Function (RBF) neural network is introduced, which has the following properties: online learning, hierarchical learning, dynamic self-organizing structure, fast speed in learning, and generalization in learning.
- An Enhanced Dynamic Fuzzy Neural Network (ED-FNN) learning algorithm based on SOM and RLLS Estimator is developed, which is suitable for real-time applications and it outperforms other approaches.
Conclusions and Recommendations

Conclusions

- A GD-FNN based on Ellipsoidal Basis Function (EBF) is introduced, which can provide a simple and fast approach to configure a fuzzy system so that
  - some meaningful fuzzy rules could be acquired for knowledge engineering, and
  - the system could be used as a system modeling tool for control engineering, pattern recognition, etc..

Thank You!