Intelligent Control Systems Methods
– Introduction

Prof Ka C Cheok
Dept of Electrical and Systems Engineering
Oakland University
Rochester MI 48309

Vetronics Institute Winter Workshop Series – 2003
US Army TACOM, Warren, Michigan
Dec 3rd, 2002
Human Intelligence is complicated and multifaceted. The interpretation of intelligence is subjective and relative. Depends on who you talk to, there is not a single definition that fits all.

That’s why we have many words that describe human intelligence based on situations. For example,

**General:** Clever, Bright, Brilliant, Wise, Sharp, Smart,
**Business:** Shrewd, Perceptive, Insightful,
**Arts:** Gifted, Talented, Creative
**Trades:** Resourceful, Ingenious, Inventive, Skillful
**Reasoning:** Rational, Logical, Reasonable, Sound, Sensible, Quick witted,

* A smart person learns from his/her mistakes; but a wise person learns from others’ mistakes

**Definition of Intelligent Quotient:**

\[
IQ = \frac{\text{Person's Intelligence}}{\text{Average intelligence of persons of same age}}
\]

**Definition of Emotional Quotient:**

\[
EQ = \frac{\text{Person's emotional behavior}}{\text{Average emotional behavior of persons of same age}}
\]
Q: If we are looking for “intelligence” in a strange place (on Earth or another planet) or at a thing, what do you look for?

A: We may look for one or more of these features:

**Essence of Life** - Survivability (plants, animals, humans) Reproduction (children) Growth

**Instinct** - Preprogrammed response to stimuli, must find energy (food) and replenish its energy supply

**Memory** – recall, information retrieval

**Computation** - The change for $2.50 out of $10.00 is $8.50 (duh?). (This is a main key element!)

**Consciousness** - Recognition (who, what, when, where, why, how),

**Cognition** (awareness of present conditions and need to adapt)

**Learning** - Question-Think-Learn-Discover, Rate of learning - Quick study, Self-taught

**Understanding** - Comprehension

**Communication** - Input & Output, Spoken language, Written language, Signs and drawing,

**Problem Solving** - How to get to what it wants.

**Abstract Reasoning** - Prediction and rules

**Sensory and Motor Ability** - Tara Lipinski

**Emotion** - Feeling

**Social Competence** - Rain man
Intelligence is relative (to individuals)

You traveled to a distant planet and discovered that ‘lowly’ worms are the main citizens of the planet. Did you discover intelligence?

Advanced inter-galactic super aliens from another distant planet came to Earth and discovered the ‘lowly’ human. Did they discover intelligence?

A four year-old kid plays “Mary had a little lamb” on a piano versus an adult playing it. (Age)

A person takes only a minute to understand a complex concept while another takes months. (Rate)

Oprah Winfield made $1 million a week at one time, so did Paul McCartney. (Skill specific)

A man looked up and spoke to God, “Hey, God, a million years, that must be like a second or so to you, right?” And God said, “Yeah, that’s right.” The man said, “Hey, God, a million dollars, that must be like a penny, right?” And God said, “Yeah, that’s right.” The man said, “Hey, God, in that case, will you give me a million dollars?” And God said, “Sure, I’ll give you a penny in a second.” (Standardization)
Who rules the Earth?

You say the Human? Well,… may be it’s the Microbes!!!

In H.G. Well’s “War of the Worlds,” the Martians were defeated not by humans, but by microbes (virus).

In the movie, *Independence Day*, Hollywood changed that to a computer virus!

Who do you think is more intelligent? Humans or microbes?

If humans are intelligent, why can’t we beat cancer? One day we will. You see, our intelligence is still developing. Discovered DNA and bioengineering…
Q: What is essential elements of Life?
A: Two words: Reproduction and Survival, for all plants and animals.
   In the case of humans, we have to have an additional elements called Fun.
   In a movie, an alien said to an earthling, “Your species reproduce and consume the world resources.”

   Dead things cannot be intelligent, can it?
   Mechanical machines are clever but not intelligent!
   So we add computers to control machines!

This leads to the areas of Intelligent Systems/Machines:

- Limited Task Specific Intelligence
- Artificial Intelligence
- Machine Intelligence
- Robot Intelligence
- Intelligent Control
- Computational Intelligence (a key development)

COMPUTERS AND BRAINS: The brain is the hardware. The mind is the software.
The thought is the processing of intelligence.
In a broad sense, a smart machine must necessarily possess five essential elements to qualify as an "intelligent agent":
(1) Input (sensing);
(2) Output (actuation);
(3) Memory (database);
(4) Rules (interpretation); and
(5) Ability to modify or augment one or more of the above components as necessary (adaptation).

In general, **Unmanned Vehicle Systems (UVS)** operations comprise one or more of the following modes:

(1) **Manual**: The system requires a human operator to control the vehicle.

(2) **Semi-autonomous or supervised**: The system temporarily controls the operation of the vehicle and seeks supervision of the operator at planned or unplanned schedule or event.

(3) **Autonomous or automatic**: The system controls the vehicle operation for an extended period.
### A Timeline for Progress in Computational Intelligence

<table>
<thead>
<tr>
<th>Year</th>
<th>CONVENTIONAL AI</th>
<th>NEURAL NETWORKS</th>
<th>FUZZY SYSTEMS</th>
<th>GENETIC METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1940s</td>
<td>1947 Cybernetics</td>
<td>1943 Neuron model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1950s</td>
<td>1956 Artificial Intelligence</td>
<td>1957 Perceptron</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1960s</td>
<td>1960 LISP language</td>
<td>'60s Adaline, Madeline</td>
<td>'65 Fuzzy sets</td>
<td></td>
</tr>
<tr>
<td>1970s</td>
<td>'75 Knowledge Engineering/Expert Systems</td>
<td>'74 Backpropagation algo</td>
<td>'74 Fuzzy controllers</td>
<td>'70s Genetic algorithm</td>
</tr>
<tr>
<td>1980s</td>
<td>'80s Heuristics searches</td>
<td>'80 Self-organizing map</td>
<td>'85 Fuzzy modeling</td>
<td>'80s Artificial Life Immune modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'82 Hopfield net</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>'83 Boltzmann machine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>'86 Backpropagation algo boom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990s</td>
<td></td>
<td>Applications</td>
<td>'90s Neuro-fuzzy models</td>
<td>'90 Genetic Programming</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'91 ANFIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'94 CANFIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fuzzy Clustering</td>
<td></td>
</tr>
<tr>
<td>2000s</td>
<td></td>
<td>Applications</td>
<td>Applications</td>
<td></td>
</tr>
</tbody>
</table>
References


R.E. King, *Computational Intelligence in Control Engineering* (lighter reading)


Mathworks Product, www.mathworks.com
- Matlab Neural Networks Toolbox,
- Matlab/Simulink Neural Networks Blockset
- Matlab Fuzzy Logic Toolbox,
- Matlab/Simulink Fuzzy Logic Blockset

GAOT – Genetic Algorithms Optimization ToolBox for Matlab

*IEEE Transactions on Fuzzy Logic*

*IEEE Transactions on Neural Networks*
Historical notes on Fuzzy Logic

1920-30. Heisenberg & Max Black (mathematician) introduced principle of uncertainty

1930’s Probability theory

1965 Lofti Zadeh introduced fuzzy set theory & fuzzy logic.

Many claimed that fuzzy logic theory which represents “possibility theory” resembled “probability theory” even though they are mathematically and conceptually different.

1970’s. Several successful commercial application of FL in Japan made the world aware of its potential. Accelerating & decelerating a subway train; camera adjustments, consumer appliances, etc.

1980’s Fuzzy logic that learns to program itself.
Intelligent Control Systems Methods – Neural Networks

Prof Ka C Cheok
Dept of Electrical and Systems Engineering
Oakland University
Rochester MI 48309

Vetronics Institute Winter Workshop Series – 2003
US Army TACOM, Warren, Michigan
Dec 3rd, 2002
Biological Neural Network

a) An isolated neuron under a microscope with a magnification of about $10^6$

b) Looking at slices of live neurons under microscope, one can observe chemical causing electrical activities among the neurons.

c) A human brain has about a massive network of $10^{11}$ to $10^{12}$ neurons, connected in a random-like parallel pattern

Note of interest: Albert Einstein had $10^{13} - 10^{14}$ neurons. How do we know that? He donated his brain to Princeton Univ after his death so it could be preserved and studied. Somebody did.
Biologically Inspired Neural Network Model

Massively parallel connections
Artificial Neural Networks (ANN)

ANN is a branch of science, engineering and technology involving

• Observation of biological neural activities
• Observation of human behavior and decision activities
• Mathematical modeling of observation
• Training of model in learning to emulate examples
• Self-organization and adaptation of model to optimize performance
• Computer hardware and software aspects
• Applications to various fields

Mathematics is a branch of Science.

| THEORY  | Endeavor to conceptualize NATURE |
| SCIENCE | Endeavor to understand NATURE    |
| ENGINEERING | Endeavor to extend NATURE |
| TECHNOLOGY | Successes of these endeavors KaCC |
Activation threshold and squashing functions

- pure lin
- pos lin
- satlin
- hardlim
- logs ig
- tribas, radbas
- hardlims
- tans ig
- satlins

12/2/2002 VI Winter Workshop Series - 2003
### Math expression
\[ y = f(s) \]

### Matlab function
\[
\begin{align*}
\text{purelin} & : y = f(s) = s \\
\text{poslin} & : y = \begin{cases} 
  s & \text{if } s > 0 \\
  0 & \text{if } s \leq 0 \\
  0 & \text{if } s < 0
\end{cases} \\
\text{satlin} & : y = \begin{cases} 
  s & \text{if } 0 \leq s \leq s_i \\
  1 & \text{if } s_i \leq s
\end{cases} \\
\text{hardlim} & : y = \begin{cases} 
  1 & \text{if } s > 0 \\
  0 & \text{if } s \leq 0
\end{cases} \\
\text{logsig} & : y = \frac{1}{1 + e^{-s}} \\
\text{hardlims} & : y = \begin{cases} 
  1 & \text{if } s > 0 \\
  -1 & \text{if } s \leq 0
\end{cases} \\
\text{tansig} & : y = \frac{1 - e^{-s}}{1 + e^{-s}} \\
\text{satlims} & : y = \begin{cases} 
  s & \text{if } -s_i \leq s \leq s_i \\
  1 & \text{if } s_i \leq s \\
  0 & \text{if } s < -1 \text{ or } s > 1
\end{cases} \\
\text{tribas} & : y = \begin{cases} 
  s + 1 & \text{if } -1 \leq s < 0 \\
  -s + 1 & \text{if } 0 \leq s \leq 1
\end{cases} \\
\text{radbas} & : y = e^{-s^2}
\end{align*}
\]

### Transfer function
- pure linear function
- positive linear function
- saturation linear function
- hard limits
- logarithmic sigmoid
- hard limits symmetric
- hyperbolic tangent sigmoid
- saturation limits symmetric
- triangular basis
- radial basis
Feedforward Neural Network (FNN) Model

1-input 1-output single feedforward neuron model

Nonlinear activation function

Math model

\[ s = wu + b \]
\[ y = f(s) \]

Examples of I/O mapping

\[ s = wu + b; \quad y = \text{logsig}(s) \]

\[ s = wu + b; \quad y = \text{radbas}(s) \]
Math model

\[ s = w u + b \]

\[ y = f(s) \]
2-input 1-output single feedforward neuron model

Examples of I/O mapping

\[ y = \text{radbas}(s), \quad s = w_1 u_1 + w_2 u_2 + b, \quad w_1 = 1, \quad w_2 = 1; \quad b = 1 \]

\[ y = \text{radbas}(s), \quad s = w_1 u_1 + w_2 u_2 + b, \quad w_1 = 1, \quad w_2 = 1; \quad b = .1 \]

\[ s_1 = w_1 u_1 + w_2 u_2 + b_1 \]

\[ y_1 = f(s_1) \]
2-2-1 FNN  (2 inputs, 2 hidden neurons, 1 output neuron)

Examples of I/O mapping

Note: Feedforward neural networks (FFNN) are capable of mapping various input-output patterns!
Matrix-Vector Models for FNN

Individual Variable Form

\[ s = w_1 u_1 + w_2 u_2 + b \]
\[ y = f(s) \]

Matrix-Vector Form

\[ u \xrightarrow{W} s \xrightarrow{f} y \]
\[ s = [w_1 \ w_2] [u_1 \ u_2] + b \]
\[ y = f(s) \]
Individual Variable Form

\[ s_1 = w_{11} u_1 + \cdots + w_{1r} u_r + b_1 \]
\[ y_1 = f(s_1) \]

\[ s_2 = w_{21} u_1 + \cdots + w_{2r} u_r + b_2 \]
\[ y_2 = f(s_2) \]

Matrix-Vector Form

\[ s = W u + b \]
\[ y = f(s) \]
**r-input m-output FNN**

**Individual Variable Form**

\[ u_1 \rightarrow w_{11} \rightarrow s_1 \rightarrow f \rightarrow y_1 \]
\[ u_r \rightarrow w_{1r} \rightarrow b_1 \rightarrow \sum \rightarrow f \rightarrow y_1 \]
\[ 1 \rightarrow w_m \rightarrow s_m \rightarrow f \rightarrow y_m \]

\[ s_1 = w_{11}u_1 + \cdots + w_{1r}u_r + b_1 \]
\[ y_1 = f(s_1) \]
\[ \vdots \]
\[ s_m = w_{m1}u_1 + \cdots + w_{mr}u_r + b_m \]
\[ y_m = f(s_m) \]

**Matrix-Vector Form**

\[ u \rightarrow W \rightarrow s \rightarrow f \rightarrow y \]
\[ 1 \rightarrow b \]
\[ s = Wu + b \]
\[ y = f(s) \]

\[ \begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1r} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mr} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_r \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \]

\[ \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = f(s) \]

\[ \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} f_1(s_1) \\ \vdots \\ f_m(s_m) \end{bmatrix} \]
\textbf{r-n-m FNN} \hspace{1em} (r-inputs, n-nodes, m-outputs)

\textbf{Matrix-Vector Form}

\[ u \xrightarrow{W_1} s_1 \xrightarrow{f_1} y_1 \]
\[ b_1 \]
\[ 1 \]
\[ s_1 = W_1 u + b_1 \]
\[ y_1 = f(s_1) \]

\[ y_1 = f(s_1) \]
\[ y_2 = f(s_2) \]

\[ s_1 = W_1 u + b_1 \]
\[ s_2 = W_2 y_1 + b_2 \]

\[ W_2 \]
\[ \sum \]
\[ s_2 \]
\[ b_2 \]
\[ 1 \]

\[ y_2 = f(s_2) \]

\textbf{Individual Variable Form}

\[ s_{11} = w_{111} u_1 + \cdots + w_{11r} u_r + b_{11} \]
\[ y_{11} = f(s_{11}) \]
\[ s_{1n} = w_{1n1} u_1 + \cdots + w_{1nr} u_r + b_{1n} \]
\[ y_{1n} = f(s_{1n}) \]
\[ s_{1} = W_{1} u + b_{1} \]
\[ y_{1} = f(s_{1}) \]

\[ s_{21} = w_{211} y_{11} + \cdots + w_{21n} y_{1n} + b_{21} \]
\[ y_{21} = f(s_{21}) \]
\[ s_{2m} = w_{2m1} y_{11} + \cdots + w_{2mn} y_{1n} + b_{2m} \]
\[ y_{2m} = f(s_{2m}) \]

\[ s_{2} = W_{2} y_{1} + b_{2} \]
\[ y_{2} = f(s_{2}) \]

12/2/2002 VI Winter Workshop Series - 2003
**r-n₁–n₂–m FNN**  
*(r-inputs, n₁-nodes, n₂-nodes m-outputs) (3-layers)*

**Matrix-Vector Form**

- \( s_1 = W_1 u + b_1 \)
- \( y_1 = f(s_1) \)
- \( s_2 = W_2 y_1 + b_2 \)
- \( y_2 = f(s_2) \)
- \( s_3 = W_3 y_2 + b_3 \)
- \( y_3 = f(s_3) \)

\[
\begin{align*}
\mathbf{s}_1 &= \begin{bmatrix} w_{11} & \cdots & w_{1r} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nr} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_r \end{bmatrix} + \begin{bmatrix} b_{11} \\ \vdots \\ b_{1r} \end{bmatrix} \\
\mathbf{y}_1 &= \begin{bmatrix} f_1(s_{11}) \\ \vdots \\ f_1(s_{n1}) \end{bmatrix} \\
\mathbf{s}_2 &= \begin{bmatrix} w_{211} & \cdots & w_{21n} \\ \vdots & \ddots & \vdots \\ w_{2n1} & \cdots & w_{2nn} \end{bmatrix} \begin{bmatrix} y_{11} \\ \vdots \\ y_{1n} \end{bmatrix} + \begin{bmatrix} b_{21} \\ \vdots \\ b_{2n} \end{bmatrix} \\
\mathbf{y}_2 &= \begin{bmatrix} f_2(s_{21}) \\ \vdots \\ f_2(s_{2n}) \end{bmatrix} \\
\mathbf{s}_3 &= \begin{bmatrix} w_{311} & \cdots & w_{31n} \\ \vdots & \ddots & \vdots \\ w_{3n1} & \cdots & w_{3nn} \end{bmatrix} \begin{bmatrix} y_{21} \\ \vdots \\ y_{2n} \end{bmatrix} + \begin{bmatrix} b_{31} \\ \vdots \\ b_{3n} \end{bmatrix} \\
\mathbf{y}_3 &= \begin{bmatrix} f_3(s_{31}) \\ \vdots \\ f_3(s_{3n}) \end{bmatrix}
\end{align*}
\]
Human learns to imitate actions of others.

Biological neural networks are responsible for decision making capability of a person. Artificial neural networks can be programmed to imitate the perception involved in the decision.

For example

**Note**: In general, we need not restrict an ANN to emulate only a BNN

**Artificial Neural Network (ANN) firmware**

Layer 1: \( s_1 = W_1u + b_1 \) \( y_1 = f_1(s_1) \)

Layer 2: \( s_2 = W_2y_1 + b_2 \) \( y_2 = f_2(s_2) \)

\vdots

Layer N-1: \( s_{N-1} = W_{N-1}y_{N-2} + b_{N-1} \) \( y_{N-1} = f_{N-1}(s_{N-1}) \)

Layer N: \( s_N = W_Ny_{N-1} + b_N \) \( y_N = f_N(s_N) \)

**Note**: In practice, we’d need to specify
- The number of layers of neural network
- The number of neurons in each layer
- The type of activation functions
- The weights and biases for these
Training FNN to Learn from Examples

Acquire several (if possible) sets of observation (input $u$) $\rightarrow$ decision (output) pattern representing the perception we would like to retain.

Specify the configuration for the FNN (E.g., $r$-$n_1$-$n_2$-$m$ layers and type of activation functions). Guess the weights and biases, and compare the FNN outputs to the Training Patterns, as shown in the figure below.

Tune FNN by adjusting the weights and biases so that its output $y_N$ matches that of the pattern $y$. The goal then is to make the error $e = y - y_N$, as small as possible when the FNN is presented with observation $u$. 

\[
\begin{align*}
\text{Observation} & \rightarrow \text{Decision Pattern} \\
\text{Artificial Neural Network (ANN) firmware} & \\
\text{Layer 1} & \quad s_1 = W_1u + b_1 \quad y_1 = f_1(s_1) \\
\text{Layer 2} & \quad s_2 = W_2y_1 + b_2 \quad y_2 = f_2(s_2) \\
& \vdots \\
\text{Layer N-1} & \quad s_{N-1} = W_{N-1}y_{N-2} + b_{N-1} \quad y_{N-1} = f_{N-1}(s_{N-1}) \\
\text{Layer N} & \quad s_N = W_Ny_{N-1} + b_N \quad y_N = f_N(s_N)
\end{align*}
\]

(+ $e = y - y_N$)
Optimization of Compared Outcome

The observation \( u \) and decision \( y \) in a multi-input multi-output pattern are vectors

\[
\begin{bmatrix}
\vdots \\
\end{bmatrix}
\begin{bmatrix}
\vdots \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
u_1 \\
\vdots \\
u_n \\
\end{bmatrix}
\begin{bmatrix}
y_1 \\
\vdots \\
y_n \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
u(1) & u(2) & \cdots & u(K) \\
\end{bmatrix}
\begin{bmatrix}
y(1) & y(2) & \cdots & y(K) \\
\end{bmatrix}
\]

Similarly, the ANN output can be expressed as

\[
Y_N = [y_N(1) \ y_N(2) \ \cdots \ y_N(K)]
\]

The error vectors (compared outcome) can then be represented as

\[
E = [y(1) - y_N(1) \ y(2) - y_N(2) \ \cdots \ y(K) - y_N(k)]
\]

A typical cost function for judging the goodness of fit in the emulation is given by

\[
J = \frac{1}{2} \left( e'(1)e(1) + e'(2)e(2) + \cdots + e'(K)e(K) \right)
\]

Find weights \( W_i \) and \( b_j \) such that the cost function is minimized.
Optimization Techniques

Given a cost function, there are several ways to find an optimum solution. Optimization techniques can be categorized onto the following approaches:

**Calculus Gradient techniques** which adjust a parameter \( p \) based on sensitivity (\( \Delta J/\Delta p \) = variation of cost function over variation of parameter). Examples: The Delta Rule, Hill Climbing, Back-Propagation, etc.

**AI Heuristics techniques** which expand a branch in tree search based on evaluation of the cost function. Examples: Breadth first, depth first, A* algorithm, etc.

**Evolution type techniques** which use a population of parameters to evolve into better and better generations of parameters. Examples: Evolution algorithm, Genetic Algorithm, Genetic Programming.
Delta Rule & Back-Propagation

The Delta Rule

Suppose we have a cost function $J(\theta)$ of that has a minimum as shown in the figure on the right. The Delta rule for searching for the minimum is to

- Step backward if the gradient is uphill
- Step forward if the gradient is downhill
- Stop if the gradient is close to being flat

We can mathematically describe the rule as

\[
\Delta \theta = -\gamma \frac{\partial J}{\partial \theta}
\]

Parameter update

\[
\theta_{\text{new}} = \theta_{\text{old}} + \Delta \theta
\]

$\gamma$ = factor that determines search step size

$\theta$ may be a vector

Although drawn here as a scalar

Back-propagation is a well-known gradient search technique for training FNN that is based on the Delta Rule
Training FNN using Back-Propagation

**Back-Propagation for a 1-1 Neuron**

Input-output pattern to be emulated:
\[ \{u(1), u(2), \ldots, u(K)\} \quad \{y(1), y(2), \ldots, y(K)\} \]

Math model of neuron:
\[ s_i = wu + b \quad y_i = f(s_i) \]

Input-output generated by neuron model:
\[ \{u(1), u(2), \ldots, u(K)\} \quad \{y_i(1), y_i(2), \ldots, y_i(K)\} \]

Cost function to be minimized:
\[ J = \frac{1}{2} \left[ (y(1) - y_i(1))^2 + (y(2) - y_i(2))^2 + \cdots + (y(K) - y_i(K))^2 \right] \]

Back-Propagation update is given by:
\[ w = w + \Delta w \quad b = b + \Delta b \]
\[ \Delta w = -\gamma_w \frac{\partial J}{\partial w} \quad \Delta b = -\gamma_b \frac{\partial J}{\partial b} \]

\[ \frac{\partial J}{\partial w} = \left( \frac{\partial J}{\partial s_i} \right) \left( \frac{\partial s_i}{\partial w} \right) = \left( \frac{\partial J}{\partial y_i} \right) \left( \frac{\partial y_i}{\partial s_i} \right) \left( \frac{\partial s_i}{\partial w} \right) = ((y - y_i)(-1))(g(s_i))(u) \]
\[ g(s_i) = \frac{\partial y_i}{\partial s_i} = \frac{\partial f(s_i)}{\partial s_i} \]

\[ \frac{\partial J}{\partial b} = \left( \frac{\partial J}{\partial s_i} \right) \left( \frac{\partial s_i}{\partial b} \right) = \left( \frac{\partial J}{\partial y_i} \right) \left( \frac{\partial y_i}{\partial s_i} \right) \left( \frac{\partial s_i}{\partial b} \right) = ((y - y_i)(-1))(g(s_i))(1) \]
Example: Back-Propagation Training for a 1-1 Neuron

A pattern or phenomenon that we can observe. In this example, it so happens that the pattern behaves like this. \((y \text{ is normalized})\)

A single neuron model with logsig activation

Back-propagation algorithm

\[
\begin{align*}
\Delta w &= -\gamma_1 \frac{\partial J}{\partial w} = (e)(-1)(1 - y_1)y_1u \\
\Delta b &= -\gamma_2 \frac{\partial J}{\partial b} = (e)(-1)(1 - y_1)y_1
\end{align*}
\]

\[
\begin{align*}
w_{\text{new}} &= w_{\text{old}} + \Delta w \\
b_{\text{new}} &= b_{\text{old}} + \Delta b
\end{align*}
\]
Back-Propagation Training for a 2-1 Neuron

A pattern or phenomenon that we can observe. In this example, it so happens that the pattern behaves like this. (y is normalized)

\[
\begin{align*}
    s &= W\mathbf{u} + b \\
    y &= f(s)
\end{align*}
\]

\[
\begin{align*}
    W &= \begin{bmatrix} w_1 & w_2 \end{bmatrix} \\
    \mathbf{u} &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
    w_{1\text{new}} &= w_{1\text{old}} + \Delta w_1 \quad \leftarrow \Delta w_1 = -\gamma_1 \frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial w_1} = (e)(-1)(1-y_1)y_1u_1 \\
    w_{2\text{new}} &= w_{2\text{old}} + \Delta w_2 \quad \leftarrow \Delta w_2 = -\gamma_1 \frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial w_2} = (e)(-1)(1-y_1)y_1u_2 \\
    b_{\text{new}} &= b_{\text{old}} + \Delta b \quad \leftarrow \Delta b = -\gamma_2 \frac{\partial J}{\partial b} = \frac{\partial J}{\partial b} = (e)(-1)(1-y_1)y_1
\end{align*}
\]

1 layer NN may not fit all data points
Back-Propagation Training for a Two-Layer r-n-m Neural Network

A pattern or phenomenon that we can observe. In this example, it so happens that the pattern behaves like this. (y is normalized)

\[ s_1 = W_1 u + b_1 \]
\[ s_2 = W_2 y_1 + b_2 \]
\[ y_1 = f(s_1) \]
\[ y_2 = f(s_2) \]

\[ \frac{\partial J}{\partial s_1} = \left[ \frac{\partial J}{\partial s_{11}} \right] \]
\[ \frac{\partial J}{\partial s_2} = \left[ \frac{\partial J}{\partial s_{21}} \right] \]

\[ W_i = W_i + \Delta W_i, \quad \Delta W_i = -\gamma W_i \left[ \frac{\partial J}{\partial W_i} \right] \]
\[ b_i = b_i + \Delta b_i, \quad \Delta b_i = -\gamma b_i \left[ \frac{\partial J}{\partial b_i} \right] \]

2 layer NN fits data points better

12/2/2002
Use an FNN to mimic an operator eye-hand coordination

- **Human-in-the-loop**
  - Eyes
  - Hand

- **Output Device**
  - Monitor

- **Input Device**
  - Joystick

- **Visual Animation**
  - VRML View of Driving Scenery

- **Dynamics System Simulation**
  - Motorized Kinematics Automobile Model

**PC-based Simulation/Animation**

**Throttle & Steer**
- (Hand action)

**Vehicle**

**Preview deviations**
- (Eye observation)

- **α β γ δ**

**You’ve been replaced!**

- **Artificial Neural Network**
  - Emulation of Driving Skill

- **Visual Animation**
  - VRML View of Driving Scenery

- **Dynamics System Simulation**
  - Motorized Kinematics Automobile Model
Virtual Simulation of Lane Keeping

Human outputs (Throttle, Steer)

VRML Driving Simulation

Lane Offset

Display

RUN1.mat

ToFile

Trained FNN outputs (NNThrottle, NNSteer) replaces human operator
Comparison of Trained FNN outputs (NNThrottle, NNSteer) to Human outputs (Throttle, Steer)

Neural networks learn and mimic I/O behavior of a pattern.
Matlab Neural Networks Toolbox

This example shows how Matlab NN Toolbox was used to train the Lane Keeping FNN

```matlab
load Run1
figure(1);
subplot(2,1,1); plot(LD4XY(1,:),LD4XY(2:5,:),'linewidth',2); legend('LD','LD3','LD6','LD12')
subplot(2,1,2); plot(LD4XY(1,:),LD4XY(6:7,:),'linewidth',2); legend('Throttle','Steer');
P = LD4XY(2:5,:); T = LD4XY(6:7,:);

% Training and Simulation of a 4-1-2 FFNN
% We'd like to train the following FFNN
%   s11 = [w111 w112 w113 w114][p1 p2 p3 p4]' + [b11]
%   s12 = [w121 w122 w123 w124][p1 p2 p3 p4]' + [b12]
%   s13 = [w131 w132 w133 w134][p1 p2 p3 p4]' + [b13]
%   s14 = [w141 w142 w143 w144][p1 p2 p3 p4]' + [b14]
%   s15 = [w151 w152 w153 w154][p1 p2 p3 p4]' + [b15]
%   y11 = tansig(s11)
%   y12 = tansig(s12)
%   y13 = tansig(s13)
%   y14 = tansig(s14)
%   y15 = tansig(s15)
%   y21 = [w211 w212 w213 w214 w215][y11 y12 y13 y14 y15]' + [b21]
%   y22 = [w221 w222 w223 w224 w225][y11 y12 y13 y14 y15]' + [b22]
% so that y2 duplicates T.

net = newff([-5 5; -5 5; -5 5; -5 5],[5 2],{'tansig' 'purelin'}); % 4-5-2 FNN with input ranges
net.trainParam.epochs = 100; % Train for this no. of epochs
net = train(net,P,T);
YTrained = sim(net,P);
figure(1); subplot(2,1,2); plot(LD4XY(1,:),[LD4XY(6:7,:); YTrained],'linewidth',2); legend('Throttle','Steer','NNThrottle','NNSteer');

W1 = net.IW{1,1}; b1 = net.b{1,1}; W2 = net.LW{2,1}; b2 = net.b{2,1}
```

12/2/2002
Simulink Neural Networks Blockset

Matlab Simulink provides NN Blocksets for simulating NN using block diagrams
Recursive Neural Networks (RNN)

\[ y_{1d} = \text{delayed } y_1 \]
\[ y_{2d} = \text{delayed } y_2 \]

\[ s_1 = W_1 u + b_1 + V_1 y_{1d} \]
\[ s_2 = W_2 y_1 + b_2 + V_2 y_{2d} \]
\[ y_1 = f(s_1) \]
\[ y_2 = f(s_2) \]

Neural networks learn and mimic I/O behavior of a pattern which depends on output past values.
References - Neural Networks


R.E. King, *Computational Intelligence in Control Engineering* (lighter reading)

*IEEE Transactions on Neural Networks*

*Matlab Neural Networks Toolbox, The Mathworks, www.mathworks.com*

*Matlab/Simulink Neural Networks Blockset, The Mathworks, www.mathworks.com*
Intelligent Control Systems Methods – Fuzzy Logic

Prof Ka C Cheok
Dept of Electrical and Systems Engineering
Oakland University
Rochester MI 48309

Vetronics Institute Winter Workshop Series – 2003
US Army TACOM, Warren, Michigan
Dec 3rd, 2002
**Precision & Non-Precision**

**Precise Statement**
Mathematics is a precise language for describing scientific, engineering and business principles.

E.g. You will earn 3.4% A.P.R as interest for your saving account. The interest dividend will be computed and distributed at the end of each month. Bank statements?

E.g. Force-Acceleration-Speed-Displacement calculus

\[
ma = -bv - kd + f
\]

\[
v(t) = \int_0^t a(\tau) d\tau + v(0)
\]

\[
d(t) = \int_0^t v(\tau) d\tau + d(0)
\]

**Non-Precise Statement**
Adjectives and adverbs are non-precise words for describing fuzzy meaning ideas.

“You look nice.”
Better still, “You look like shit. What’s your secret?” (joke)
“Did you like it?” “Er, interesting.”
Few, several, many, plentiful, lots

If you are hot, then turn the thermostat down.

Q. How can a computer interpret this statement?
A. Use Fuzzy logic
Fuzzy Variables, Values and Membership

Rating a movie from 0 to 10… *Spiderman* is a *nine* out of 10. Or simply *marvelous*.

“He’s *twenty five* years old,” means he could be between 25.000 and 25.999 years of age.

Human are non-precise in thinking and speaking:
E.g., I would like to spend *fifty grand* dollars on a car.
$50,000 is really close to the verbal *fifty grand*. (100% membership)
$55,000 is on the high side of *fifty grand*. (Say 50% membership of *fifty grand*)
$42,000 is on the low side of *fifty grand*. (Say 25% membership)

Precise Variables and Values

A computer, on the other hand, can operate with high precision
E.g. a double precision floating number can deal with any number in the range
Variable = +/-y.yyyyyyyyyyyyyyy x 10^+/-308
E.g., \( \pi = 3.141592653589793 \ldots \) is a magical number

Numbering system would be simpler and more natural if we have only 2 or 4 or 8 or 2^n fingers.
*Octopi* got the number systems right.
Boolean Sets versus Fuzzy Sets

Classical Boolean Set
(crisp membership value, either 0 or 1)

Fuzzy Set
(soft membership values varying between 0 or 1)

characteristic function
\[
f_A(x) = \begin{cases} 
1 & \text{if } x \in A \\
0 & \text{if } x \not\in A 
\end{cases}
\]

membership function
\[
\mu_A(x) : X \rightarrow [0, 1]
\]
Shapes of Membership Functions

Triangular
$$\mu = \max \left( \min \left( \frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right)$$

Trapezoidal
$$\mu = \max \left( \min \left( \frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right)$$

Gaussian standard bell shape
$$\mu = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

Gaussian with two different sides

Generalized Gaussian bell shape
$$\mu = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2h}}$$

Sigmoid
$$\mu = \frac{1}{1 + e^{-a(x-c)}}$$

Difference of 2 sigmoids
$$\mu = \frac{1}{1 + e^{-a_1(x-c_1)}} - \frac{1}{1 + e^{-a_2(x-c_2)}}$$

Product of 2 sigmoids
$$\mu = \frac{1}{1 + e^{-a_1(x-c_1)}} \times \frac{1}{1 + e^{-a_2(x-c_2)}}$$

Z-shape spline

$\pi$-shape spline

S-shape spline

ANY OTHER CONVEX TYPE SHAPE WILL ALSO DO!!!
“If-Then” rules are the most commonly used fuzzy logic statements. They can be used to represent knowledge.

A single antecedent-single consequence “If-Then” rules has the form:

Rule1. \( \text{If } x \text{ is } A1 \text{ then } y \text{ is } B1 \)
Rule2. \( \text{If } x \text{ is } A2 \text{ then } y \text{ is } B2 \)
Rule3. \( \text{If } x \text{ is } A3 \text{ then } y \text{ is } B3 \)

Compounded-antecedent-single consequence rules take the form:

Rule1. \( \text{If } x \text{ is } A1 \text{ and/or } y \text{ is } B1 \text{ then } z \text{ is } C1 \)
Rule2. \( \text{If } x \text{ is } A2 \text{ and/or } y \text{ is } B2 \text{ then } z \text{ is } C2 \)
Rule3. \( \text{If } x \text{ is } A3 \text{ and/or } y \text{ is } B3 \text{ then } z \text{ is } C3 \)

If \( \text{Road is Left} \) then \( \text{Steer to Left} \)
If \( \text{Road is Middle} \) then \( \text{Steer to Middle} \)
If \( \text{Road is Right} \) then \( \text{Steer to Right} \)

If \( \text{Road is Left or Obstacle is Right} \) then \( \text{Steer to Left} \)
If \( \text{Road is Left and Obstacle is Left} \) then \( \text{Steer to Middle} \)
And so on…
Fuzzy Inference Systems (FIS)

1) Fuzzification
Convert crisp value into fuzzy association

2) Fuzzy Arithmetic
Interpret And, Or, Not operations in Antecedents

3) Implication
Produce the consequence for each rule

4) Aggregation
Produce the total consequence for all rules

5) Defuzzification
Convert fuzzy set into a crisp value

Knowledge & Data Base
Rule1. If x is A1 and/or y is B1 then z is C1
Rule2. If x is A2 and/or y is B2 then z is C2
Rule3. If x is A3 and/or y is B3 then z is C3

Crisp numerical input value

Sensors

Mechanical System

Actuators

Crisp numerical output value

Computer

12/2/2002
VI Winter Workshop Series - 2003
1) FUZZIFICATION

Associate a crisp numerical input value to fuzzy values with degree of membership.

Example: Sometimes we rate a movie on the scale of 0 to 10. Let’s say we just want to use a fuzzy labels like 'Horrible', 'Bad', 'OK', 'Good', 'Excellent', to describe the movie. Suppose that the membership function for these fuzzy labels/values are as shown below:

![Membership Function Diagram]

We associate a crisp value 6.50 to

- 0% Horrible
- 22% Bad
- 75% OK
- 97% Good
- 27% Excellent

A crisp numerical value

Fuzzy values with degree of membership
5) DEFUZZIFICATION

Convert fuzzy sets into a crisp numerical output value.

Example: Suppose we have the fuzzy set shown below:

There are many ways to do this:

Centroid.

\[ y_{\text{Centroid}} = \frac{\int_0^1 y \times y_{\text{Aggregate}(y)}\,dy}{\int_0^1 y_{\text{Aggregate}(y)}\,dy} \]

Bisector/Median.

\[ y_{\text{Bisector}} = \text{median}\left(y \times y_{\text{Aggregate}(y)}\right) \]

Maximum

\[ y_{\text{Max}} = \{y_{\text{Aggregate}(y)}\} \]

\[ y_{\text{Max}} = y_{i_{\text{Max}}} \]
2) FUZZY ARITHMETIC - LOGICAL OPERATIONS

An AND operation can be treated as equivalent to a MIN or PRODUCT operation

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A and B</th>
<th>min(A, B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Boolean AND operation

Fuzzy AND operation

Note: 0.5 is the largest value

Note: 0.25 is the largest value

12/2/2002
An OR operation can be treated as equivalent to a MAX or PROBOR operation

**Max operation**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A or B</th>
<th>max(A, B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Probabilistic OR**

**Probor operation**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A or B</th>
<th>A+B-A*B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Boolean OR**

operation

**Fuzzy OR**

operation

Note: 0.5 is the smallest value

Note: 0.75 is the smallest value
3) IMPLICATION

Contribution from each rule toward the outcome

Example Rule 1: If \( x \) is \( A_1 \) and \( y \) is \( B_1 \), then \( z \) is \( C_1 \)

\[
\mu_{A_1}(x) \quad \mu_{A_1}(x_0) \quad \mu_{B_1}(y) \quad \mu_{B_1}(y_0) \quad \mu_{C_1}(z)
\]

At \( x = x_0 \) and \( y = y_0 \), we can compute the **Strength** \( s_1 \) of Antecedent of Rule 1 as

\[
s_1 = \min(\mu_{A_1}(x_0), \mu_{B_1}(y_0)) \quad \text{or} \quad s_1 = \mu_{A_1}(x_0) \ast \mu_{B_1}(y_0).
\]

The implication or contribution from the rule consequence is

\[
z_1(z) = \min(s_1, \mu_{C_1}(z)) \quad \text{or} \quad z_1(z) = s_1 \ast \mu_{C_1}(z)
\]
Example Rule 2: If $x$ is $A_1$ or $y$ is $B_1$, then $z$ is $C_1$

At $x = x_0$ and $y = y_0$, we can compute the Strength $s_i$ of Antecedent of Rule 1 as

$$s_i = \max(\mu_{A_i}(x_0), \mu_{B_i}(y_0)) \quad \text{or} \quad s_i = \mu_{A_i}(x_0) + \mu_{B_i}(y_0) - \mu_{A_i}(x_0)\mu_{B_i}(y_0).$$

The implication or contribution from the rule consequence is

$$z_1(z) = \min(s_i, \mu_{C_i}(z)) \quad \text{or} \quad z_1(z) = s_i \mu_{C_i}(z).$$
4) **AGGREGATION**

Example Rule 1

Example Rule 2

Aggregation = contribution from all rules

\[
\begin{align*}
\mu_{A_1}(x) &\quad \mu_{A_2}(x) \\
\mu_{B_1}(y) &\quad \mu_{B_2}(y) \\
\mu_{C_1}(z) &\quad \mu_{C_2}(z) \\
\end{align*}
\]

\[
\begin{align*}
c_1(z) &= \min(s_1, \mu_{C_1}(z)) \\
c_2(z) &= s_2 \cdot \mu_{C_2}(z) \\
\max(c_1(z), c_2(z)) &\quad z_1(z) + z_2(z) \\
\end{align*}
\]
**Mamdani Style FUZZY LOGIC**

**Example**

1. **Fuzzification**

   - For $x_0$, $A_1$:
     - $\mu_{A_1}(x_0)$
   - For $y_0$, $B_1$:
     - $\mu_{B_1}(y_0)$

2. **Fuzzy operation**

   - $s_1 = \min(\mu_{A_1}(x_0), \mu_{B_1}(y_0))$
   - $s_2 = \max(\mu_{A_2}(x_0), \mu_{B_2}(y_0))$

3. **Implication**

   - $s_1 = \mu_{C_1}(z)$ (Singleton)
   - $s_2 = \mu_{C_2}(z)$ (Singleton)

4. **Aggregation**

   - $\mu_{C_{\text{aggregate}}}(z) = s_1 \circ z_1 + s_2 \circ z_2$

5. **Defuzzification**

   - $z_0 = \frac{s_1 z_1 + s_2 z_2}{s_1 + s_2}$

**Rule1.** If $x$ is $A_1$ and $y$ is $B_1$ then $z$ is $C_1$.

**Rule2.** If $x$ is $A_2$ or $y$ is $B_2$ then $z$ is $C_2$.

**Surprisingly simple!!!**

12/2/2002  
VI Winter Workshop Series - 2003
Sugeno Style FUZZY LOGIC

Example

Rule 1. If x is A1 and y is B1 then u is

\[ u_1 = m_1 x + n_1 y + c_1 \]

Rule 2. If x is A2 or y is B2 then u is

\[ u_2 = m_2 x + n_2 y + c_2 \]

Rule 3. If x is A3 and y is B3 then u is

\[ u_3 = m_3 x + n_3 y + c_3 \]

1. Fuzzification

2. Fuzzy operation

3. Defuzzification

The Math for the Fuzzy Logic

\[ s_1 = \mu_A(x)\mu_B(y) \]

\[ s_2 = \mu_{A_2}(x) + \mu_{B_2}(y) - \mu_{A_2}(x)\mu_{B_2}(y) \]

\[ s_3 = \mu_A(x)\mu_B(y) \]

\[ u(x, y) = \frac{s_1u_1 + s_2u_2 + s_3u_3}{s_1 + s_2 + s_3} \]

Surprisingly simple!!!
ANFIS (Adaptive Network Fuzzy Inference System)

ANFIS is a Sugeno-style Fuzzy Logic that learns to emulate an I/O pattern (similar to a neural network).

Sugeno style Fuzzy Logic

If $x$ is $A_1$ and $y$ is $B_1$, then $z$ is $g_1 = l_1 x + m_1 y + n_1$.
If $x$ is $A_2$ and $y$ is $B_2$, then $z$ is $g_2 = l_2 x + m_2 y + n_2$.
If $x$ is $A_3$ and $y$ is $B_3$, then $z$ is $g_3 = l_3 x + m_3 y + n_3$.

\[
\mu_A(x) = \frac{1}{1 + e^{-\alpha_1(x-c_1)}} = \frac{1}{1 + e^{-h}} \\
\mu_B(y) = \frac{1}{1 + e^{-\alpha_2(y-c_2)}} = \frac{1}{1 + e^{-h}} \\
\mu_A(x) = e^{\frac{(x-c_1)}{\sigma_2}} = e^{x_2} \\
\mu_B(y) = e^{\frac{(y-c_2)}{\sigma_2}} = e^{y_2} \\
\mu_A(x) = \frac{1}{1 + e^{\alpha_3(x-c_3)}} = \frac{1}{1 + e^{-h}} \\
\mu_B(y) = \frac{1}{1 + e^{\alpha_4(y-c_4)}} = \frac{1}{1 + e^{-h}}
\]

ANFIS will automatically tune the parameters $a_1, \sigma_2, a_3, \sigma_3, a_4, \sigma_4, a_5, c_1, c_2, c_3, c_4, c_5$ and $c_6$, and $l_1, m_1, n_1, l_2, m_2, n_2, l_3, m_3, n_3$ so that the fuzzy logic output $z$ matches up with the observed $z_{obs}$.

12/2/2002
VI Winter Workshop Series - 2003
**Sugeno fuzzy logic**

\[ s_1 = a_1(x - c_1) \quad s_4 = a_4(y - c_4) \]

\[ s_2 = \frac{1}{2} \left( \frac{x - c_2}{\sigma_2} \right)^2 \quad s_5 = \frac{1}{2} \left( \frac{y - c_5}{\sigma_5} \right)^2 \]

\[ s_3 = a_3(x - c_3) \quad s_6 = a_6(y - c_6) \]

\[ g_1 = g_1(x, y) = l_1x + m_1y + n_1 \]

\[ g_2 = g_2(x, y) = l_2x + m_2y + n_2 \]

\[ g_3 = g_3(x, y) = l_3x + m_3y + n_3 \]

\[ z = \frac{w_1g_1 + w_2g_2 + w_3g_3}{w_1 + w_2 + w_3} \]

**ANFIS scheme**

Cost function to be minimized \[ J = \frac{1}{2} \sum_{k=1}^{N} (R_k - Z_k)^2 \]

1. Initialize the coefficients \( a_1, a_2, a_3, a_4, s_5, a_6, c_1, c_2, c_3, c_4, c_5 \) and \( c_6 \), to some estimated values.

2. Calculate \( w_1, w_2 \) & \( w_3 \) for each set of input data, \( X_i \) and \( Y_i \).

3. Apply least square estimation technique to calculate \( l_1, m_1, n_1, l_2, m_2, n_2, l_3, m_3, n_3 \)

4. Apply gradient search technique to update \( a_1, s_2, a_3, a_4, s_5, a_6, c_1, c_2, c_3, c_4, c_5 \) and \( c_6 \)

5. Repeat from Step 2, until satisfactory
Matlab Fuzzy Logic Toolbox

Main diagram

Rule editor: E.g., IF Service is Excellent or Food is Delicious, THEN Tip is Generous

I/O membership functions

Rule viewer

I/O surface viewer
Simulink Fuzzy Logic Toolset
A Fuzzy Logic Example for Tipping
1. If (Service is Excellent) or (Food is Delicious) then (Tip is Generous) [1]
2. If (Service is Poor) or (Food is Rancid) then (Tip is Poor) [1]
3. If (Service is Nothing) or (Food is Delicious) then (Tip is RealGenerous) [1]
4. If (Waitress is Friendly) then (Tip is RealGenerous) [1]
**Example Application of ANFIS**

*Use an ANFIS to mimic an operator eye-hand coordination*

- **Human-in-the-loop**: Eyes, Hand
- **Output Device**: Monitor
- **Input Device**: Joystick
- **Visual Animation**: VRML View of Driving Scenery
- **Dynamics System Simulation**: Motorized Kinematics Automobile Model
- **You've been replaced!**
  - **Adaptive Network Fuzzy Inference System**: Emulation of Driving Skill
  - **Visual Animation**: VRML View of Driving Scenery
  - **Dynamics System Simulation**: Motorized Kinematics Automobile Model

**PC-based Simulation/Animation**

- **Throttle & Steer (Hand action)**
  - **Vehicle**
  - Preview deviations (Eye observation)

Your neurons and you
Virtual Simulation of Lane Keeping

Human outputs (Throttle, Steer)

VRML Driving Simulation

Lane Offset

Display

RUN1.mat

To File

Virtual Simulation of Lane Keeping

Trained ANFIS outputs (FLThrottle, FLSteer) replaces human operator
A Matlab program for train ANFIS

```matlab
load Run1
figure(1);
subplot(2,1,1);
plot(LD4XY(1,:),LD4XY(2:5,:),'linewidth',2);
legend('LD','LD3','LD6','LD12')
subplot(2,1,2);
plot(LD4XY(1,:),LD4XY(6:7,:),'linewidth',2);
legend('Throttle','Steer')

P = LD4XY(2:5,:);
T = LD4XY(6:7,:);
x = sort(P(2,:))';
y = sort(T(2,:))';

trnData = [x y];
umMFs = 5;
mfType = 'gbellmf';
epoch_n = 20;
in_fismat = genfis1(trnData,numMFs,mfType);
out_fismat = anfis(trnData,in_fismat,20);
plot(x,y,x,evalfis(x,out_fismat));
legend('Training Data','ANFIS Output');
```
Comparison of Trained ANFIS outputs (FLSteer) to Human outputs (Steer)

-20 -10 0 10 20 30 40 50 0 5 10 15 20 25 30 35 40 45 50

LD  LD3  LD6  LD12

Steer versus FLSteer

-2 -1 0 1 2 3 4 5

-1 -0.5 0 0.5 1

Training data Steer  ANFIS FLSteer
Recurrent Fuzzy Logic

Rule 1. If \( x \) is \( A_1 \) and \( y \) is \( B_1 \) and \( z_{old} \) is \( C_1 \), then \( z \) is \( D_1 \).

Rule 2. If \( x \) is \( A_2 \) or \( y \) is \( B_2 \) and \( z_{old} \) is \( C_2 \), then \( z \) is \( D_2 \).

1) Fuzzification
   Convert crisp value into fuzzy association

2) Fuzzy Arithmetic
   Interpret \textit{And}, \textit{Or}, \textit{Not} operations in Antecedents

3) Implication
   Produce the consequence for each rule

4) Aggregation
   Produce the total consequence for all rules

5) Defuzzification
   Convert fuzzy set into a crisp value

6) Delayed Effect
   Feedback effect based on delayed output

Computer

Sensors

Mechanical System

Actuators

Crisp numerical input value

Crisp numerical output value
References - Fuzzy Logic


R.E. King, *Computational Intelligence in Control Engineering* (lighter reading)


*IEEE Transactions on Fuzzy Logic*


Intelligent Control Systems Methods – Genetic Algorithms & Programming

Prof Ka C Cheok
Dept of Electrical and Systems Engineering
Oakland University
Rochester MI 48309

Vetronics Institute Winter Workshop Series – 2003
US Army TACOM, Warren, Michigan
Dec 3rd, 2002
### From Quark to Universe

<table>
<thead>
<tr>
<th></th>
<th>Radius (lightyears)</th>
<th>Approx Radius (km)</th>
<th>km in Log10 scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universe (&gt; 100 billion galaxies)</td>
<td>15 billion lightyears</td>
<td>141,912,000,000,000,000,000,000</td>
<td>1.4191E+23</td>
</tr>
<tr>
<td>Super Cluster (millions of galaxies)</td>
<td>75 million lightyears</td>
<td>709,560,000,000,000,000,000,000</td>
<td>7.0956E+20</td>
</tr>
<tr>
<td>Local Group Cluster (tens and hundreds of galaxies)</td>
<td>2 million lightyears</td>
<td>18,921,600,000,000,000,000,000</td>
<td>1.8922E+19</td>
</tr>
<tr>
<td>Milky Way Galaxy (tens of millions of stars)</td>
<td>250,000 lightyears</td>
<td>2,365,200,000,000,000,000,000,000</td>
<td>2.3652E+18</td>
</tr>
<tr>
<td>Solar Neighborhood (hundreds of stars)</td>
<td>20 lightyears</td>
<td>189,216,000,000,000,000,000</td>
<td>1.8922E+14</td>
</tr>
<tr>
<td>Solar System (1 star, 9 planets)</td>
<td>7 hours at speed of light</td>
<td>7,560,000,000,000,000,000,000</td>
<td>7.5600E+09</td>
</tr>
<tr>
<td>Earth</td>
<td>8917 miles</td>
<td>14,267</td>
<td>1.4267E+04</td>
</tr>
<tr>
<td>You and I</td>
<td>2m</td>
<td>0.001</td>
<td>1.0000E-03</td>
</tr>
<tr>
<td>Cell</td>
<td>1 micro m</td>
<td>0.0000000001</td>
<td>1.0000E-09</td>
</tr>
<tr>
<td>Atom</td>
<td>1 pico m</td>
<td>0.0000000000000001</td>
<td>1.0000E-15</td>
</tr>
<tr>
<td>Neutron, Positron, Electron, Quark &amp; …</td>
<td></td>
<td>0.0000000000000000001</td>
<td>1.0000E-18</td>
</tr>
</tbody>
</table>

1 lightyear = 9.4608E+12 Kilometers  
5.9130E+12 Miles  
5.9 trillion miles

4- byte representation of real numbers (single precision float)

1E-32 to 1E+32

8- byte representation of real numbers (double precision float)

1E-308 to 1E+308

We invented numbers and computers that help us thinking about quarks & universe.

Joke: There are 3 kinds of people in the world: Those who can count and those who can't. (Ha, ha, ha!)

Evolution... Life on Earth has been around for a billion years... Oldest human about 60 million years old.

We've come along way, baby... Recorded history ~ 3000 yrs... Computer/Information age ~ 50 years

12/2/2002  
VI Winter Workshop Series - 2003
Evolution. A gradual process in which something changes into a different and usually more complex or better form

Jean Baptise LaMarck (1801)
- Characteristics of animal/plants adapt to repeating behavior and environment.
- Acquired characteristic pass on to children.
- Example: Giraffes’ feeding habits explain the long necks.
- Proposed Theory of Evolution

Charles Darwin (1862)
- Proposed “Natural Selection” as the mechanism of evolution
- Species flourish or perish, based on how their genetic functions accommodate their environments.
- “Survival of the Fittest”

John Holland (1960s) University of Michigan
- Proposed the biological evolution inspired “Genetic Learning Algorithm”
- A method for optimization and “machine learning”.
- GA is a “learning mechanism” in the AI classification
Good genes, good stock, good breed…
Animals & plants …People, horses, dogs, etc.

In olden days, only the privileged have access to knowledge (books, instruction). So, good stock was very critical to society. Today, it still is in many cultures.

In modern days, we have education systems, mass publication and the Internet to educate the public. Widespread education help to shape genes for current and next generations.

Hence genetics and education go together well.
The two essential elements of life are: SURVIVAL & REPRODUCTION
GENETIC ALGORITHM

GA is an optimization technique that mimics biological evolution

- Population that keeps improving
- Reproduction & Mutation
- Evaluation & Survival of the fittest
Consider a simple $x$ versus $y$ relationship such as the one shown in the figure. The situation is this: Given a value of $x$, we can find out the value of $y$ via an $x$-$y$ look-up table or a very complicated math relationship between $x$ & $y$.

Apply GA procedure to find the maximum of the shape shown in figure.

We may, however, guess that $x$ lies in a certain region; in this case $x \in [0 \, 31]$. For simplicity of explanation, we may use a 5-bit binary number scheme to represent $x$ as integers. We may start with an initial population of 4 integers (guessed) as shown below.

**Initial Population**

<table>
<thead>
<tr>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0 1 1 0 1]</td>
</tr>
<tr>
<td>[1 1 0 0 1]</td>
</tr>
<tr>
<td>[0 0 1 0 0]</td>
</tr>
<tr>
<td>[1 0 0 1 1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x (value)</th>
<th>y (value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>121</td>
</tr>
<tr>
<td>19</td>
<td>16</td>
</tr>
</tbody>
</table>

**Arranged marriages**

<table>
<thead>
<tr>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0 0 1 0 0]</td>
</tr>
<tr>
<td>[1 1 0 0 1]</td>
</tr>
<tr>
<td>[1 0 0 1 1]</td>
</tr>
<tr>
<td>[0 1 1 0 1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x (value)</th>
<th>y (value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>121</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
</tbody>
</table>

Mate (best candidates)
Mate (so so, but the kid may be good...)
Reproduction 1

Parents
[0 1 0 1 0]
[1 1 0 0 1]
[1 0 0 1 1]
[0 1 1 0 1]

Children
[0 1 0 0 1]
[1 0 1 0 0]
[1 0 0 0 1]
[0 1 1 1 1]

Parents x (value) y (value)
[0 0 1 0 0] 4 121
[1 1 0 0 1] 25 100
[1 0 0 1 1] 19 16
[0 1 1 0 1] 13 4

Children
[0 1 0 0 1] 9 49
[1 0 1 0 0] 20 25
[1 0 0 0 1] 17 4
[0 1 1 1 1] 15 0

Sort using y
Keep only 4

Selection 1

Mate (best candidates)
Mate (so so, but the kid may be good…)

Parents x (value) y (value)
[0 1 0 1 0] 4 121
[1 1 0 0 1] 25 100
[0 1 0 0 1] 9 49
[1 0 1 0 0] 20 25

Arranged marriages
Reproduction 2

Parents
[0 0 1 0 0]
[1 1 0 0 1]
[0 1 0 0 0]
[1 0 1 0 0]

Children
[0 0 1 0 1]
[1 1 0 0 0]
[0 1 1 0 0]
[1 0 0 0 0]

Mutation

Parents
x (value) y (value)
[0 0 1 0 0] 4 121
[1 1 0 0 1] 25 100
[0 1 0 0 0] 8 49
[1 0 1 0 0] 20 25

Children
[0 0 1 0 1] 5 100
[1 1 1 0 0] 28 169
[0 1 1 0 0] 12 9
[1 0 0 1 0] 18 9

Sort using y
Keep only 4

Selection 2

Population
x (value) y (value)
[1 1 1 0 0] 28 169
[0 0 1 0 0] 4 121
[1 1 0 0 1] 25 100
[0 0 1 0 1] 5 100

Arranged marriages

Mate (best candidates)
Mate (so so, but the kid may be good…)

12/2/2002 VI Winter Workshop Series - 2003
### Reproduction 3

#### Parents

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1 1 1 0 0]</td>
<td>1 0 1 0 0</td>
<td>0 1 1 0 0</td>
</tr>
<tr>
<td>[0 0 1 0 0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1 1 0 0 1]</td>
<td>1 1 1 0 1</td>
<td>0 0 0 0 1</td>
</tr>
</tbody>
</table>

#### Children

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1 1 0 0 1]</td>
<td>0 1 1 0 0</td>
<td>0 0 0 0 1</td>
</tr>
</tbody>
</table>

#### Population

<table>
<thead>
<tr>
<th>Parents</th>
<th>x (value)</th>
<th>y (value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0 0 1 0 0]</td>
<td>4</td>
<td>121</td>
</tr>
<tr>
<td>[1 1 0 0 1]</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>[0 1 0 0 0]</td>
<td>8</td>
<td>49</td>
</tr>
<tr>
<td>[1 0 1 0 0]</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

#### Arranged marriages

- Mate (best candidates)
- Mate (so so, but the kid may be good…)

#### Selection 3

#### Mate (best candidates)

- [1 1 1 0 1] 29 196
- [0 0 0 0 1] 1 196

#### Mate (so so, but the kid may be good…)

- [1 1 1 0 0] 28 169
- [0 0 1 0 0] 4 121

Set using y

- Keep only 4
Reproduction 4  Selection 4
Reproduction 5  Selection 5
Reproduction 6  Mutation  Selection 6
Reproduction 7  Selection 7

And so on until an exit condition is satisfied
Formalization of GA Procedure

**Encoding Scheme:** A point (parameter, variable or potential solution) is represented by a binary bit string called a chromosome, which can be encoded to represent a numerical value, multiple values or some pattern. E.g., \( x_i = 000111101001 \) can represent the number 489 in decimal. It can also be encoded to represent the point \((1, 13, 9)_{10} = (0001, 1110, 1001)_2\) in a 3D parameter space. Or it can mean other things.

**Current Generation:** A collection of points \( \{ x_i, i = 1, \ldots, N \} = \{ x_1, x_2, \ldots, x_N \} \) is called a population (or gene pool).

**Fitness Evaluation:** Each point, \( x_i \), has a fitness evaluation index, \( f_i \). Hence, \( \{ f_i, i = 1, \ldots, N \} = \{ f_1, f_2, \ldots, f_N \} \).

**Selection:** Each point, \( x_i \), is assigned a probability equal to \( \frac{f_i}{\sum_{k=1}^{N} f_k} = \frac{f_i}{f_1 + \ldots + f_N} \).

- For example, if \( N = 8 \), then the chances of each point to be selected are likened to a roulette wheel shown. Half of the population will be selected based on the probability; in this example, most likely the largest slices (of the pizza)
**Crossover:** About 99 to 99.99% (crossover rate) of the selected population is chosen for mating to produce possibly better offspring. From the chosen population, a pair of selected points is mated by a cut and paste crossover procedure. A **one-point crossover** (most common) may look like this, where the crossover cut is randomly picked:

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>000111</td>
<td>101001</td>
</tr>
<tr>
<td>111000</td>
<td>001010</td>
</tr>
</tbody>
</table>

A **two-point crossover** may look like this, where the two cut positions are again randomly picked:

<table>
<thead>
<tr>
<th>Parents</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01111</td>
</tr>
<tr>
<td>11</td>
<td>10000</td>
</tr>
</tbody>
</table>

**Mutation:** About 0.01 to 1% (mutation rate) of the selected population mutates to produce normal, superior or inferior offspring. A mutation may simply flip the value of a randomly picked bit in the chromosome as in

<table>
<thead>
<tr>
<th>Parent</th>
<th>Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>001111010101</td>
<td>000111101101</td>
</tr>
</tbody>
</table>

↑ mutated bit

**Next Generation:** The next generation consist of

- Selection of fittest population (SURVIVAL)
- Crossovers and mutants (the X-men) (REPRODUCTION)

**Life goes on:** Repeat the process of Selection, Fitness Evaluation, Crossover, Mutation to produce the future generations. Stop when an Exit Condition is satisfied.
It probably will take many hours (allow an engineering margin of 20dB (10x) to 40dB (100x) times, please) to program and debug our own Matlab code to perform the genetic algorithm.

Luckily, someone else has done this!

Search for “GAOT” on the the Web and download the free Genetic Algorithm Optimization Toolbox.
Example of a GA run using Matlab and GAOT

```matlab
clf; figure(gcf);  

% Let's consider the maximization of the function: f(x) = x + 10*sin(5*x)+7*cos(4*x) over the interval (0,9)
subplot(2,2,1); fplot('x + 10*sin(5*x)+7*cos(4*x)',[0 9]); hold on;
initPop = initializega([0 9], 'gademo1eval1');  

% Let's create a random starting population of size 10.
% Now let's run the ga for one generation.
x = ga([0 9], 'gademo1eval1', [], initPop, [1e-6 1 1], 'maxGenTerm', 1,...
   'normGeomSelect', [0.08], ['arithXover'], [2 0], 'nonUnifMutation', [2 1 3]);

% And plot the resulting population
plot(endPop(:,1), endPop(:,2), 'b*', 'linewidth', 2); xlabel('x'); ylabel('f(x)'); title('Initial Population'); hold off;

N = 10;  

% Now let's run the ga for N generations
[x endPop bpop trace] = ga([0 9], 'gademo1eval1', [], initPop, [1e-6 1 1], 'maxGenTerm', N,...
   'normGeomSelect', [0.08], ['arithXover'], [2], 'nonUnifMutation', [2 N 3]);

x = The best found
subplot(2,2,2); fplot('x + 10*sin(5*x)+7*cos(4*x)', [0 9]); hold on;
plot(endPop(:,1), endPop(:,2), 'g*', 'linewidth', 2); xlabel('x'); ylabel('f(x)'); title(strcat(num2str(N),'th Generation')); hold off;

N = 20;

% Now let's run the ga for N generations
[x endPop bpop trace] = ga([0 9], 'gademo1eval1', [], initPop, [1e-6 1 1], 'maxGenTerm', N,...
   'normGeomSelect', [0.08], ['arithXover'], [2], 'nonUnifMutation', [2 N 3]);

x = The best found
subplot(2,2,3); fplot('x + 10*sin(5*x)+7*cos(4*x)', [0 9]); hold on;
plot(endPop(:,1), endPop(:,2), 'r*', 'linewidth', 3); xlabel('x'); ylabel('f(x)'); title(strcat(num2str(N),'th Generation')); hold off;

% Let's take a look at the performance of the ga during the run
subplot(2,2,4); plot(trace(:,1), trace(:,3), 'b-'); hold on
plot(trace(:,1), trace(:,2), 'r-'); xlabel('Generation'); ylabel('Fitness'); hold off;

% The red line is a track of the best solution, the yellow is a track of the average of the population
```
This function is called by the preceding program

```
function [sol, val] = gaDemo1Eval(sol,options)
% Demonstration evaluation function used in gademo1.
% f(x)=x+10sin(5x)+7cos(4x)
%
% function [val,sol] = gaDemo1Eval(sol,options)
% val - the fitness of this individual
% sol - the individual, returned to allow for Lamarckian evolution
% options - [current_generation]

x=sol(1);
val = x + 10*sin(5*x)+7*cos(4*x);
```

Result of preceding Matlab program shows that GA find the maximum in the 20th generation or less.
GA Matlab program to search for a peak on a plateau

% MyGA_PlateauPeak.m
% First, we need to create MyGA_Eval.m to evaluate the function (the new function above)
% Second create MyGA_Initialize.m (exact copy of Initialize.m from GAOT)
% Third, run the GA.m

clear all; clear all; figure(1); subplot(2,2,1);
fplot('1+(x>5)*(x<5.2)*(x-4)',[0 9],100); % To look for the max of the function over interval (0,9):
initPop=MyGA_Initialize(10,[0 9],'MyGA_Eval_PlateauPeak'); % Random initial population of size 10.
hold on; plot (initPop(:,1),initPop(:,2),'bo', 'linewidth',3); hold off;
xlabel('x'); ylabel('y'); title('Initial Generation');

N = 2; % Now let's run the GA N time & plot the resulting population
[x endPop bpop trace] = ga([0 9],'MyGA_Eval_PlateauPeak',[],initPop,[1e-6 1 1],'maxGenTerm',N,...
 'normGeomSelect',[0.08],[2],['arithXover'],[2],['normGeomSelect'],[2 25 3]);
subplot(2,2,2); fplot('1+(x>5)*(x<5.2)*(x-4)',[0 9],100);
hold on; plot (endPop(:,1),endPop(:,2),'go', 'linewidth',3); hold off;
xlabel('x'); ylabel('y'); title(strcat(num2str(N),'th Generation'));

N = 10; % Run the GA N time starting from initPop
[x endPop bpop trace] = ga([0 9],'MyGA_Eval_PlateauPeak',[],initPop,[1e-6 1 1],'maxGenTerm',N,...
 'normGeomSelect',[0.08],[2],['arithXover'],[2],['normGeomSelect'],[2 25 3]);
subplot(2,2,3); fplot('1+(x>5)*(x<5.2)*(x-4)',[0 9],100);
hold on; plot (endPop(:,1),endPop(:,2),'r*', 'linewidth',3); hold off;
xlabel('x'); ylabel('y'); title(strcat(num2str(N),'th Generation'));

% Lets take a look at the performance of the ga during the run
subplot(2,2,4); plot(trace(:,1),trace(:,3),'g-', trace(:,1),trace(:,2),'r-')
legend('average fitness','best fitness'); xlabel('Generation'); ylabel('Fittness');

function [sol, val] = MyGA_Eval(sol,options)
x = sol(1);
val = 1+(x>5)*(x<5.2)*(x-4);
Results of preceding Matlab program shows that GA was able to find the peak in 10 generations or less. The flat plateau provides no glue whatsoever (no gradient).
GENETIC PROGRAMMING

Genetic Programming is an extension of Genetic Algorithm to a population of mathematical tree graph expressions, whose structures and functions are randomly initialized and then optimized.

For example, the tree graph expression below represents the expression

\[( + x ( * y (\sqrt{2z}) ) ) \]

(LISP S-expression) or

\[( x + ( y * \sqrt{2z} ) ) \]

(standard math expression)
Operators for the tree graph expression take on any math operations

\[ + - \times / \sqrt{ ( )^2 \sin \cos \tan \ etc} \]

Variables can change

Value of gains can change

Structure, operators, variables & gains for the tree graph expression can change
Population of tree expressions – Generation 1

Selected for parametric crossover

Selected for structural crossover

Population of tree expressions – Generation 2

Same structures = same function
New parameters

New structures = new functions
New parameters

Population of tree expressions – Generation 3 … repeat
References – Genetic Algorithms & Programming


R.E. King, *Computational Intelligence in Control Engineering*, Marcel Dekker, 1999 (lighter reading)


GAOT – *Genetic Algorithms Optimization ToolBox* for Matlab