Expert Systems and Adaptive Control

Robert Stengel
Robotics and Intelligent Systems MAE 345, Princeton University, 2013

- Expert systems
- Gain scheduling
- Adaptive critic
- Cerebellar model articulation controller
- Reinforcement learning
- Failure-tolerant control

Copyright 2013 by Robert Stengel. All rights reserved. For educational use only. http://www.princeton.edu/~stengel/MAE345.html

Expert Systems: Using Signals to Make Decisions

- Programs that exhibit intelligent behavior
- Program that uses rules to evaluate information
- Program meant to emulate an expert or group of experts making decisions in a specific domain of knowledge (or universe of discourse)
- Program that chains algorithms to derive conclusions from evidence
Functions of Expert Systems

- **Design**
  - Conceive the form and substance of a new device, object, system, or procedure
- **Diagnosis**
  - Determine the nature or cause of an observed condition
- **Instruction**
  - Impart knowledge or skill
- **Interpretation**
  - Explain or analyze observations
- **Monitoring**
  - Observe a process, compare actual with expected observations, and indicate system status
- **Negotiation**
  - Propose, assess, and prioritize agreements between parties
- **Planning**
  - Devise actions to achieve goals
- **Prediction**
  - Reason about time, forecast the future
- **Reconfiguration**
  - Alter system structure to maintain or improve performance
- **Regulation**
  - Respond to commands and adjust control parameters to maintain stability and performance

Principal Elements of a Rule-Based Expert System

```
DORMANT RULE BASE

KNOWLEDGE ACQUISITION
(Induction)

KNOWLEDGE
(Rule) BASE

INFORMATION ENGINE
(Deduction)

COMMANDS, MEASUREMENTS

DATA BASE

USER INTERFACE

EXPLANATION
FACILITY

SIDE EFFECTS

REGULATED SYSTEM
```
Critical Issues for Expert System Development

• System architecture
• Inference or reasoning method \textit{(Deduction)}
• Knowledge acquisition \textit{(Induction)}
• Explanation \textit{(Abduction)}
• User interface

Representation of Knowledge for Inference

• Logic
  – Predicate calculus, 1\textsuperscript{st}-order logic
  – Fuzzy logic, Bayesian belief network, …
• Search
  – Given one state, examine all possible alternative states
• Procedures
  – Function-specific routines executed within a rigid structure (e.g., flow chart)
• Semantic (propositional) networks
  – Model of associative memory
  – Tree or graph structure
  – \textbf{Nodes}: objects, concepts, and events
  – \textbf{Links}: interrelations between nodes
• Production (rule-based) systems
  – Rules
  – Data
  – Inference engine
Basic Rule Structure

- Rule sets values of action parameters
- Rule tests values of premise parameters
- Forward chaining
  - Reasoning from premises to actions
  - Data-driven: facts to conclusions
- Backward chaining
  - Reasoning from actions to premises
  - Goal-driven: find facts that support a hypothesis
  - Analogous to numerical inversion

Elements of a Parameter

- Type
- Name
- Current value
- Rules that test the parameter
- Rules that set the parameter
- Allowable values of the parameter
- Description of parameter (for explanation)
Elements of a Rule

- **Type**
- **Name**
- **Status**
  - 0: Has not been tested
  - 1: Being tested
  - T: Premise is true
  - F: Premise is false
  - U: Premise is unknown
- **Parameters tested by rule**
- **Parameters set by rule**
- **Premise**: Logical statement of proposition or predicates
- **Action**: Logical consequence of premise being true
- **Description of premise and action**
  (for explanation)

The Basic Rule: **IF-THEN-ELSE**

- If A = TRUE, then B, else C
- Material equivalence of propositional calculus, extended to predicate calculus and 1st-order logic, i.e., applied to logical statements
- Methods of inference lead to plans of action
- **Compound rule**: Logic embedded in The Basic Rule, e.g.,
  - Rule 1: If (A = B and C = D), then perform action E, else ....
  - Rule 2: If (A ≠ B or C = D), then E = F, else ....
- **Nested (pre-formed compound) rule**: Rule embedded in The Basic Rule, e.g.,
  - Rule 3: If (A = B), then [If (C = D), then E = F, else ...], else ....
Finding Decision Rules in Data

- Identification of key attributes and outcomes
- Taxonomies developed by experts
- First principles of science and mathematics
- Trial and error
- Probability theory and fuzzy logic
- Simulation and empirical results

Example of On-Line Code Modification

- Execute a decision tree
  - Get wrong answer
- Add logic to distinguish between right and wrong cases
  - If Comfort Zone = Water,
    - then Animal = Hippo,
    - else Animal = Rhino
  - True, but Animal is Dinosaur, not Hippo
  - Ask user for right answer
  - Ask user for a rule that distinguishes between right and wrong answer: If Animal is extinct, …
Decision Rules

Representation of Data

- **Set**
  - Crisp sets
  - Fuzzy sets

- **Schema**
  - Diagrammatic representation
  - A pattern that represents elements (or objects), their attributes (or properties), and relationships between different elements

- **Frame**
  - Hierarchical data structure, with inheritance
  - Slots: Function-specific cells for data
  - Scripts: frame-like structures that represent a sequence of events

- **Database**
  - Spreadsheets/tables/graphs
  - Linked spreadsheets
Structure of a Frame

- Structure array in MATLAB
- Structure or property list in LISP
- Object in C++
- Ordered set of computer words that characterize a parameter or rule
- An archetype or prototype
- Object-oriented programming: Express Rules and Parameters as Frames

Example, Fillers, and Instance of a Frame

Application-Specific Frame
- Name
- Maker
- Year
- Model
- Color

Generic Fillers
- Object Type
  - Identifier
  - Manufacturer
  - When Built
  - Body Style
  - Color

Instantiation
- Specific Vehicle
  - VIN Number
  - BMW
  - 1994
  - 4-Door Sedan
  - White
Inheritance and Hierarchy of Frame Attributes

- Legal fillers: Can be specified by
  - Data type
  - Function
  - Range

- Inheritance property
  - All instances of a specific frame may share certain properties or classes of properties

- Hierarchical property
  - Frames of frames may be legal

- Inference engine
  - Decodes frames
  - Establishes inheritance and hierarchy
  - Executes logical statements

Animal Decision Tree: Forward Chaining

- What animal is it?

  Premise Parameter: Size
  Rule 1: If ‘Small’, test ‘Sound’
          Else, test ‘Neck’
  Action Parameter: None

  Premise Parameter: Sound
  Rule 2: If ‘Squeak’, Animal = Mouse
          Else, Animal = Squirrel [END]
  Action Parameter: Animal

  Premise Parameter: Neck
  Rule 3: If ‘Long’, Animal = Giraffe
          Else, test ‘Trunk’
  Action Parameter: Animal

  Premise Parameter: Trunk
  Rule 4: If ‘True’, Animal = Elephant
          Else, test ‘Comfort Zone’
  Action Parameter: Animal

  Premise Parameter: Comfort Zone
  Rule 5: If ‘Water’, Animal = Hippo
          Else, Animal = Rhino [END]
  Action Parameter: Animal
Animal Decision Tree:
Backward Chaining

• What are an animal’s attributes?

Animal = Hippo

From Rule 5, Comfort Zone = Water
From Rule 4, Trunk = False
From Rule 3, Neck = Short
From Rule 1, Size = Large

Animal Decision Tree:
Parameters

Type: Object Attribute
Name: Animal
Current Value: Variable
Rules that Test: None
Rules that Set: 2, 3, 4, 5
Allowable Values: Mouse, Squirrel, Giraffe, Elephant, Hippo, Rhino
Description: Type of Animal

Type: Object Attribute
Name: Neck
Current Value: Variable
Rules that Test: 3
Rules that Set: None
Allowable Values: Long, Short
Description: Neck of Animal

Type: Object Attribute
Name: Size
Current Value: Variable
Rules that Test: 1
Rules that Set: None
Allowable Values: Large, Small
Description: Size of Animal

Type: Object Attribute
Name: Sound
Current Value: Variable
Rules that Test: 2
Rules that Set: None
Allowable Values: Squeak, No Squeak
Description: Sound made by Animal

Type: Object Attribute
Name: Comfort Zone
Current Value: Variable
Rules that Test: 5
Rules that Set: None
Allowable Values: Water, Dry Land
Description: Habitat of Animal
Animal Decision Tree: Rules

Type: If-Then-Else
Name: Rule 1
Status: Variable (e.g., untested, being tested, tested and premise = T/F/unknown)
Parameters Tested: Size
Parameters Set: None
Premise: Size = Large or Small
Action: Test 'Sound' OR Test 'Neck'
Description: Depending on value of 'Size', test 'Sound' or 'Neck'

Type: If-Then-Else
Name: Rule 2
Status: Variable
Parameters Tested: Sound
Parameters Set: Animal
Premise: Size = Large or Small
Action: Set value of 'Animal' AND END
Description: Depending on value of 'Sound', identify 'Animal' as 'Mouse' or 'Squirrel'

Type: If-Then-Else
Name: Rule 3
Status: Variable
Parameters Tested: Neck
Parameters Set: Animal
Premise: Neck = Long or Short
Action: Set value of 'Animal' AND END
Description: Depending on value of 'Neck', identify 'Animal' as 'Giraffe' or test 'Comfort Zone'

Type: If-Then-Else
Name: Rule 4
Status: Variable
Parameters Tested: Trunk
Parameters Set: Animal
Premise: Trunk = True or False
Action: Set value of 'Animal' AND END
Description: Depending on value of 'Trunk', identify 'Animal' as 'Elephant' or test 'Comfort Zone'

Type: If-Then-Else
Name: Rule 5
Status: Variable
Parameters Tested: Comfort Zone
Parameters Set: Animal
Premise: Comfort Zone = Water or Dry Land
Action: Set value of 'Animal' AND END
Description: Depending on value of 'Comfort Zone', identify 'Animal' as 'Hippo' or 'Rhino'

Animal Decision Tree: Programs

Procedural Sequence of Rules
Rule1(Size, Rule2, Rule3)
Rule2(Sound, Animal, Animal)
Rule3(Neck, Animal, Rule4)
Rule4(Trunk, Animal, Rule5)
Rule5(Comfort Zone, Animal, Animal)

Declarative Sequence of Rules
BasicRule(Size, Sound, Neck)
BasicRule(Sound, Animal, Animal)
BasicRule(Neck, Animal, Trunk)
BasicRule(Trunk, Animal, Comfort Zone)
BasicRule(Comfort Zone, Animal, Animal)
Animal Decision Tree:
Rule-Based Approach

- Well suited to simple graphical user interface (GUI)

<table>
<thead>
<tr>
<th>Frame Type</th>
<th>Parameter</th>
<th>Frame Type</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Current Value</td>
<td></td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Rules That Test</td>
<td></td>
<td>Parameters Tested</td>
<td></td>
</tr>
<tr>
<td>Rules That Set</td>
<td></td>
<td>Parameters Set</td>
<td></td>
</tr>
<tr>
<td>Allowable Values</td>
<td></td>
<td>Premise</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td>Action</td>
<td></td>
</tr>
</tbody>
</table>

Rule-Based Approach:
Training and Chaining

- GUIs for training and operations

<table>
<thead>
<tr>
<th>Training</th>
<th></th>
<th>Chaining</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decisions</td>
<td></td>
<td>Input Parameter</td>
<td></td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>Output Parameter</td>
<td></td>
</tr>
<tr>
<td>Training Trials</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Animal Decision Tree: Procedural Logic

- Simple exposition of logic
- Rigid description of solution

If Size = Big
    Then If Sound = Squeak
        Then Animal = Mouse
        Else Animal = Squirrel
        EndIf
    Else If Neck = Long
        Then Animal = Giraffe
        Else If Trunk = True
            Then Animal = Elephant
            Else If Comfort Zone = Water
                Then Animal = Hippo
                Else Animal = Rhino
                EndIf
            EndIf
        EndIf
    EndIf
EndIf

Adaptive Control
Adaptive Control System Design

- Control logic changes to accommodate changes or unknown parameters of the plant
  - System identification to improve state estimate
  - Gain scheduling to account for environmental change
  - Learning systems that track performance metrics (e.g., CMAC)
  - Reinforcement learning

- Control law is nonlinear

\[ u(t) = c[z(t), a, y^*(t)] \]

Operating Points Within a Flight Envelope

Dynamic model is a function of altitude and airspeed
Design LTI controllers throughout the flight envelope
Gain Scheduling

Proportional-integral controller with scheduled gains

\[ u(t) = C_F(a)y^* + C_B(a)\Delta x + C_I(a)\int \Delta y(t) dt \approx c\begin{bmatrix} x(t), a, y^*(t) \end{bmatrix} \]

Scheduling variables, \( a \), e.g., altitude, speed, properties of chemical process, ...

Adaptive Critic Neural Network Controller

- On-line adaptive critic controller
  - Replace gain matrices by neural networks (see Lecture 19)
  - Nonlinear control law implemented as “action network”
  - Performance and control usage evaluated via “critic network”
  - Control gains adapted to improve performance
  - Cost model adapted to improve critique
Action Network On-line Training

Train action network, at time $t$, holding the critic parameters fixed

Critic Network On-line Training

Train critic network, at time $t$, holding the action parameters fixed
Cerebellar Model Articulation Controller (CMAC)

- Inspired by models of human cerebellum
- CMAC: Two-stage mapping of a vector input to a scalar output
  - First mapping: Input space to association space
    - $s$ is fixed
    - $a$ is binary
  - Second mapping: Association space to output space
    - $g$ contains learned weights

Example of Single-Input CMAC Association Space

- $x$ is in $(x_{\min}, x_{\max})$
- Selector vector is binary and has $N$ elements
- Receptive regions of association space map $x$ to $a$
  - Analogous to neurons that “fire” in response to stimulus
- $N_A = \text{Number of receptive regions} = N + C - 1 = \text{dim}(a)$
- $C = \text{Generalization parameter} = \# \text{ of overlapping regions}$
- Input quantization $= (x_{\max} - x_{\min}) / N$

$S : x \rightarrow a$

Input $\rightarrow$ Selector vector

$g : a \rightarrow y$

Selector $\rightarrow$ Output
CMAC Output and Training

- CMAC output (i.e., control command) from activated cells of $c$ Associative Memory layers
  \[
  y_{\text{CMAC}} = w^T a = \sum_{i=j}^{j+C-1} w_{i,\text{activated}}
  \]
  \(j=\) index of first activated region

- Least-squares training of CMAC weights, $w$
  - Analogous to synapses between neurons

\[
  w_{\text{new},j} = w_{\text{old},j} + \frac{\beta}{c} \left( y_{\text{desired}} - \sum_{i=1}^{c} w_{\text{old},i} \right)
\]

$\beta$ is the learning rate and $w_{j}$ is an activated cell weight

- Localized generalization and training

CMAC Output and Training

- In higher dimensions, association space is $\text{dim}(x)$, a plane, cube, or hypercube
- Potentially large memory requirements
- Granularity (quantization) of output
- Variable generalization and granularity

2-dimensional association space
CMAC Control of a Fuel-Cell Pre-Processor
(Iwan and Stengel)

Fuel cell produces electricity for electric motor
Comparison of PrOx Controllers on Federal Urban Driving Cycle

<table>
<thead>
<tr>
<th></th>
<th>mean $H_2$ error</th>
<th>maximum $H_2$ error</th>
<th>mean CO out</th>
<th>max. CO out</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%</td>
<td>%</td>
<td>ppm</td>
<td>ppm</td>
</tr>
<tr>
<td>Fixed-Air</td>
<td>0.68</td>
<td>0.87</td>
<td>6.3</td>
<td>28</td>
</tr>
<tr>
<td>Table Look-up</td>
<td>0.13</td>
<td>1.43</td>
<td>6.5</td>
<td>26</td>
</tr>
<tr>
<td>PID</td>
<td>0.05</td>
<td>0.51</td>
<td>7.7</td>
<td>30</td>
</tr>
<tr>
<td>CMAC/PID</td>
<td>0.02</td>
<td>0.16</td>
<td>7.3</td>
<td>26</td>
</tr>
</tbody>
</table>

| net $H_2$ output |

Reinforcement Learning

- Learn from success and failure
- Repetitive trials
  - Reward correct behavior
  - Penalize incorrect behavior
- Learn to control from a human operator

http://en.wikipedia.org/wiki/Reinforcement_learning
Reinforcement ("Q") Learning
Control of a Markov Process

- Q: Quality of a state-action function
- Heuristic value function
- One-step philosophy for heuristic optimization

\[ Q[x(t_{k+1}), u(t_{k+1})] = Q[x(t_k), u(t_k)] + \alpha(t_k) \left[ L_{u(t_k)} x(t_k) + \gamma(t_k) \max_u Q[x(t_{k+1}), u] - Q[x(t_k), u(t_k)] \right] \]
\( \alpha(t_k) \): learning rate, 0 < \alpha(t_k) < 1

- Various algorithms for computing best control value

\[ u_{\text{best}}(t_k) = \arg \max_u Q[x(t_k), u] \]

Q-Learning Snail  Q-Learning, Ball on Plate

Q Learning Control of a Markov Process is Analogous to LQG Control in the LTI Case

\[ Q[x(t_{k+1}), u(t_{k+1})] = Q[x(t_k), u(t_k)] + \alpha(t_k) \left[ L_{u(t_k)} x(t_k) + \gamma(t_k) \max_u Q[x(t_{k+1}), u] - Q[x(t_k), u(t_k)] \right] \]
\( \alpha(t_k) \): learning rate, 0 < \alpha(t_k) < 1

Controller

\[ x_{k+1} = \Phi x_k + \Gamma C (\hat{x}_k - x_k^*) \]

Estimator

\[ \hat{x}_k = \Phi \hat{x}_{k-1} - \Gamma C (\hat{x}_{k-1} - x_{k-1}^*) + K \left[ z_k - H_x \Phi \hat{x}_{k-1} - \Gamma C (\hat{x}_{k-1} - x_{k-1}^*) \right] \]
Real-Time
Implementation of Rule-Based Control System

Control system knowledge-base contents

<table>
<thead>
<tr>
<th>Task</th>
<th>Parameters</th>
<th>Rules</th>
<th>Major subtasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive control</td>
<td>18</td>
<td>23</td>
<td>Kalman filter and linear-quadratic regulator</td>
</tr>
<tr>
<td>Failure detection</td>
<td>9</td>
<td>15</td>
<td>Normalized innovations monitor</td>
</tr>
<tr>
<td>Failure diagnosis</td>
<td>135</td>
<td>147</td>
<td>Signal dependency search</td>
</tr>
<tr>
<td>Failure model estimation</td>
<td>15</td>
<td>23</td>
<td>Multiple-model algorithm</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>32</td>
<td>39</td>
<td>Weighted left pseudo-inverse</td>
</tr>
</tbody>
</table>

Rule-Based Control System
(Handelman and Stengel, 1989)

- Application: Failure-tolerant flight control for CH-47 Chinook helicopter
- Control is a side effect from expert system perspective
- Search until root node is solved
  - Initiates lower-level functions to declare leaf node is TRUE

---

**Example of a Failure-Diagnosis Rule**

**Rule-141:**

IF control failure candidates are determined
AND forward collective pitch control is a candidate
AND the largest element of the normalized innovations rms is pitch rate
AND the ratio of pitch rate (rad/s) to vertical velocity (m/s)
normalized innovations rms is within 10% of 6.01

THEN hypothesize forward collective pitch control stuck at

\[ \text{sign} \times \left( \frac{35.8 \times \text{pitch rate innovations rms}}{1.48} \right) \times \left( -2.85 \times \text{pitch rate innovations rms} \right) \times 0.890 \text{ s prior to failure detection} \]
Failure Response

Response to Stuck Pitch-Rate Sensor

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Time</td>
<td>1.0</td>
</tr>
<tr>
<td>Failure Detected Time</td>
<td>1.8</td>
</tr>
<tr>
<td>Failure Deteriorated Time</td>
<td>2.4</td>
</tr>
<tr>
<td>Failure Model Estimated Time</td>
<td>6.0</td>
</tr>
<tr>
<td>Reconfiguration Implemented Time</td>
<td>6.2</td>
</tr>
</tbody>
</table>

Response to Stuck Forward-Collective Pitch Actuator

Next Time: Task Planning and Multi-Agent Systems
More on Rules

• Example of a pre-formed compound rule

• Side effects: Actions triggered by inference
  – If A = TRUE, ... but what is A?
  – Execute a function to find out, and return to the rule
  – ... then B = C, ... but what is C?
  – Execute a function ...
Intelligent Aircraft/Airspace System

Graphical Representation of Knowledge: Principled Negotiation in Air Traffic Management
Real-Time Implementation of Rule-Based Control System

- Original code written in **LISP**
- Automatic procedural code generation (**LISP** to **Pascal**)
- **Real-time execution** on three i386 processors in **Multibus™ architecture**
- External PC used for code development, testing, and helicopter simulation

![Diagram of Rule-Based Controller](image)

**Preferential Oxidizer (PrOx)**

- **Proton-Exchange Membrane Fuel Cell** converts hydrogen and oxygen to water and electrical power
- **Steam Reformer/Partial Oxidizer-Shift Reactor** converts fuel (e.g., alcohol or gasoline) to $\text{H}_2$, $\text{CO}_2$, $\text{H}_2\text{O}$, and $\text{CO}$. Fuel flow rate is proportional to power demand

  - CO “poisons” the fuel cell and must be removed from the reformate
  - **Catalyst** promotes oxidation of CO to $\text{CO}_2$ over oxidation of $\text{H}_2$ in a Preferential Oxidizer (PrOx)
  - **PrOx reactions** are nonlinear functions of catalyst, reformate composition, temperature, and air flow
Summary of CMAC Characteristics

- Inputs and Number of Divisions:
  - PrOx inlet reformate flow rate (95)
  - PrOx inlet cooling temperature (80)
  - PrOx inlet CO concentration (100)
- Output: PrOx air injection rate
- Associative Layers, $C$: 24
- Number of Associative Memory Cells/Weights and Layer Offsets: 1,276 and [1,5,7]
- Learning Rate, $\eta$: $\sim$0.01
- Sampling Interval: 100 ms

Flow Rate and Hydrogen Conversion of CMAC/PID Controller

- $H_2$ conversion command (across PrOx only): 1.5%
- Novel data, with (---) and without pre-training (—)
- Federal Urban Driving Cycle (= FUDS)