INTRODUCTION

Artificial intelligence (AI) deals with efforts to make computers to think and do things intelligently. According to Barr and Feigenbaum (1981), artificial intelligence is a part of computer science that deals with designing intelligent computer systems, i.e., system that exhibit the characteristics we associate with intelligence in human behaviour. Knowledge based expert system is a branch of AI. According to Feigenbaum, expert system (ES) is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. An expert system is a computer system that emulates the decision making of a human expert. The expert knowledge is stored in the computer in an organized manner. This so called knowledge base is used to provide advice. ES does the same reasoning process that a human decision maker would go through to arrive at a decision. According to Simonovic (1991), ES in water resources is a computer application that assists in solving complicated water resources problems by incorporating engineering knowledge, principle of system analysis and experience, to provide aid in making engineering judgments and including intuition in the solution procedure.

Expert system is a branch of Artificial Intelligence but it differs from others in that:

- It deals with subject matter of realistic complexity
- It must exhibit high performance
- It must be plausible

Expert Systems, Knowledge Based Systems and Knowledge Based Expert Systems are often used synonymously.

KNOWLEDGE ENGINEERING

The process of building an expert system is called knowledge engineering. Knowledge Engineers acquire the knowledge from a human expert or other source and code in the expert system. The problem of transferring human knowledge into an expert system is so major that it is called the knowledge acquisition bottleneck. Major bottlenecks are due to: Cognitive barrier, linguistic barrier, representation barrier and the problem of creating model.
CONVENTIONAL PROGRAMS vs. ES

ESs differ from the conventional computer programs in the following aspects:

(i) ESs are knowledge intensive programs
(ii) ESs are highly interactive
(iii) ESs mimic human experts in decision making and reasoning process
(iv) ESs divides expert knowledge into number of separate rules
(v) ESs are user friendly and intelligent.

ES DEVELOPMENT

Hayes – Roth and Lenat (1983) has recommended five stages in the development of ES

(i) Identification – determining characteristics of the problem
(ii) Conceptualisation – finding concepts to represent the knowledge
(iii) Formalisation – designing structures to organize knowledge
(iv) Implementation – formulating rules embodying the knowledge
(v) Testing – validating the rules

A good coordination between the knowledge engineer and the expert is necessary.

ES TOOLS

Language: A translator of commands written in a specific syntax. An expert system language will also provide an inference engine to execute the statement of the language. (Eg. LISP is not a language but PROLOG is a Language)

Shells: A special purpose tool designed for certain types of applications in which the user must only supply the knowledge base. (Eg. EMYCIN)

Tools: A language + utility programs to facilitate the development debugging, and delivery of application programs

ES ARCHITECTURE

A ES is specific to one problem domain. However, it is not for domain modeling but for problem solving. The expert system consists of (i) a knowledge base, (ii) a working memory, (iii) an inference engine, (iv) system analysis, graphic and other softwares and (v) user interface. The architecture of a typical ES is shown in figure 1.
Knowledge base consists of declarative knowledge that are facts about the domain and procedural knowledge that are heuristic rules from the domain. The working memory is the active set of knowledge base. Inference engine is the problem solving module. It also gives justification (explanation) for the advice from the ES. Communication module helps in interaction between other modules and also provide user – developer interfaces.

**KNOWLEDGE BASE**

Knowledge base module contains domain specific knowledge. Knowledge can be either

(i) Priori Knowledge which comes before and is independent of knowledge from the senses. It is considered to be universally true and cannot be denied without contradiction. Ex: All triangles in the plane have 180 degrees

(ii) Posteriori Knowledge that is derived from the senses. It can be denied on the basis of new knowledge without the necessity of contradictions. Ex: The light is green.

Knowledge can be represented in various forms as:

i. Rules

ii. Semantic Nets

iii. Frames

iv. Scripts

v. Object Oriented

vi. Others- KL-1, KRYPTON, Conceptual Graph and so on
**Rules:** The most popular format of rules are the IF – condition – THEN – action statements. This is useful when the knowledge is in the form of condition action.

\[ P_1, ..., P_m \Rightarrow Q_1, ..., Q_n \]

\( P_1, ..., P_m \Rightarrow Q_1, ..., Q_n \) means if premises \( P_1 \) and \( ..., \) and \( P_m \) are true then perform actions \( Q_1 \) and \( ..., \) and \( Q_n \). An example of a rule is

\[
\text{IF} \quad \text{Inflow} \quad < \quad 0.7 \times \text{Average} \\
\text{AND} \quad \text{Storage} \quad < \quad \text{Capacity} / 2 \\
\text{THEN} \quad \text{irrigation release} = 0.6 \times \text{Demand}.
\]

**Semantic nets:** This representation is used when knowledge is a subset of some other bigger set. A semantic network consists of nodes connected by links that describe the relation between nodes. It is possible to represent hierarchical information.

**Frames:** Schema is used to describe a more complex knowledge structure (than semantic nets) and the frame is one type of schema. Frame is data structure for representing stereotyped situation (Minsky, 1975). Frames represent objects as sets of slot/filler pairs. Objects can contain programs as well as data (if-needed, if-added, if-removed). The utility of frames lies in hierarchical frame system and inheritance. This makes it easy to construct and manipulate a complex knowledge base. The main disadvantages in this representation are unrestrained alteration or cancellation of slots and ad hoc inference.

**INFEERENCE ENGINE**

This module examines the knowledge base and answers the questions (how and why) from the user. It is the most crucial component of ES. It derives the knowledge i.e., guides the selection of a proper response to a specific situation which is called pruning. Three formal approaches used in this case are: production rules, structured objects and predicate logic. Production rules consist of a rule set, a rule interpreter which specifies when and how to apply the rules and a working memory which holds the data, goals and intermediate results. Structured objects use vector representation of essential and accidental properties. Predicate logic uses propositional and predicate calculi. The inference engine can work in the following ways:

1. Forward Chaining
2. Backward Chaining
3. Abduction
4. Reasoning under Uncertainty
1. Forward Chaining (Bottom – up reasoning)

It starts from the known initial state and proceeds in the forward direction to achieve the goal. The inference engine searches the knowledge base with the given information for rules whose precedence matches the given current state. The basic steps are:

(i) The system is given one or more conditions

(ii) The system searches the rules in the knowledge base for each condition. Those rules that correspond to the condition in IF part are selected.

(iii) Each rule can generate new conditions from the conclusions of the invoked THEN part, which in turn are again added to the existing ones

(iv) The added conditions, if any will be processed again (step ii). The session ends if there are no new conditions.

2. Backward Chaining (Top-down reasoning)

Reasoning is done in the backward direction. The system selects a goal state and reasons in the backward direction. The initial state condition is established for the goal to be true. If the given initial state conditions matches with the established ones, then the goal is the solution. Otherwise, the system selects another goal and the process is repeated. The basic steps are:
(i) Select a goal state and rules whose THEN portion has the goal state as conclusion
(ii) Establish sub goals to be satisfied for the goal state to be true, from the IF portion of the selected rules.
(iii) Establish initial conditions necessary to satisfy all the sub goals.
(iv) Check whether the given initial state matches with the established ones. If so, then the goal is one solution. If not, select another goal state.

3. Abduction
Reasoning from observed facts to the best explanation.

\[ p \rightarrow q, \ q \text{ proves } p \]

Abduction is related to the analysis of backward chaining and implication. Abduction is a mathematically justifiable, practical, and reasonable way to generate hypotheses. Abduction is another name for a fallacious argument. It is not guaranteed to work.
Summary of the Purpose of Forward chaining, Backward chaining, and Abduction

<table>
<thead>
<tr>
<th>Inference</th>
<th>Start</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward chaining</td>
<td>Facts</td>
<td>Conclusions that must follow</td>
</tr>
<tr>
<td>Backward chaining</td>
<td>Uncertain conclusion</td>
<td>Facts to support the conclusions</td>
</tr>
<tr>
<td>Abduction</td>
<td>True conclusion</td>
<td>Facts which may follow</td>
</tr>
</tbody>
</table>

4. Reasoning under Uncertainty

When knowledge is certain, the conclusions are also certain. We can use the normal rules of logic to deduce conclusions.

Fig. 4. Reasoning under Certainty

Often, experts can't give definite answers. It may require an inference mechanism that derives conclusions by combining.

Fig. 5. Reasoning under Uncertainty
**Inference engine – Explanations**

An expert system seeks to make problem solving knowledge explicit. The knowledge applied to a problem must be available to the user. The system must be able to explain *how* it arrived at a conclusion and *why* it is performing some computation. It may also be required to answer *what if* questions.

**Answering HOW?**

To answer *how* a conclusion was reached, work back through the inference chain. *Decision 4* was made as a result of making *decision 1* and *decision 2*. *Decision 1* was made because *Facts 1, 2 & 3* are true, etc.

![Decision Diagram](image)

**Answering WHY?**

To answer *why* a computation is being performed, the system must state its current goal. The system may ask the user if *fact 3* is true because it is trying to determine if *decision 1* should be made.

![Decision Diagram](image)

**LEARNING BY INDUCTION**

Inductive learning is the process of acquiring generalized knowledge from examples or instances of some class. This form of learning is accomplished through inductive inference, the process of learning from a part to a whole, from particular instances to generalizations or from the individual to the universal. It is a powerful form of learning which we humans do...
almost effortlessly. Even though it is not a valid form of inference, it appears to work well much of the time.

Examples

- We conclude that “weather in South India is always pleasant in winter”, by observing a few seasons
- “All swans are white”: After seeing only a small number of white swans
- “All North Indians speak Hindi”: After talking to a few people in North India.
- The inductive process can be described symbolically through the use of predicates P and Q. If we observe repeated occurrence of events $P(a_1), P(a_2), \ldots, P(a_k)$, we generalize by inductively concluding that for all $x$, $P(x), Q(y)$ will happen (ex. Paddy – Green)

Different Approaches

- Learning by Observation
- Learning by Discovery
- Supervised learning
- Learning from examples
- Unsupervised learning

Generalization tree for the hierarchy of All Things

![Generalization tree for the hierarchy of All Things](image)

Fig. 8
Tree Representation for Object Descriptions

![Tree Diagram]

- s – Small; l – Large

Fig. 9

ID3 – Example

![ID3 Decision Tree]

- br = brown, bk = black, w = white, g = gray; y = yes, n = no,
- h = heavy, m = medium, l = light; t = tall, and s = short.

Fig. 10

CLIPS (C Language Integrated Production System)

CLIPS is a tool for Building Expert Systems [http://www.ghg.net/clips/CLIPS.html](http://www.ghg.net/clips/CLIPS.html). CLIPS is a multi-paradigm programming language that provides support for rule-based object-oriented, and procedural programming. It was designed at NASA/Johnson Space Center. It was
designed with the specific purpose of providing high portability, low cost, and easy integration with external systems. It has been installed on a wide variety of computers ranging from PCs to CRAY supercomputers. The main characteristics are:

- Expert system shell
- It has an excellent external language integration
- Uses forward chaining based on Rete’s algorithm
- Allows both rule-based and procedural programming paradigms

**RADEX: APPLICATION OF EXPERT SYSTEM**

Developed by Raju and Subramanian (2002). RADEX is developed for prediction of Evapo-transpiration using Christiansen method. RADEX uses both rules and computations simultaneously.

**Christiansen Method**

This method calculates $ET$ using data such as latitude, month, extra-terrestrial radiation, wind velocity, possible sunshine hours, humidity, elevation of the place. Christiansen method employs the equation:

$$ET = 0.473R_a C_T C_H C_U C_S C_M C_E$$

where $R_a$ is extra-terrestrial radiation and $C_T$, $C_H$, $C_U$, $C_S$, $C_M$ and $C_E$ are coefficients of temperature, humidity, wind velocity, bright sunshine hours, elevation and consumptive use.

**Application of Christiansen Method**

Climatic data for Mount Abu has been used and the results have been presented. Mount Abu is the place at the highest altitude in Rajasthan state. Mount Abu is located at an elevation of 1195 m and 24.36° N latitude.

Solar radiation $R_s$ is calculated from extraterrestrial radiation $R_a$ using the equation

$$R_s = \left(0.25 + \frac{0.5n}{N}\right) R_a$$

where $n$ is the sunshine hours on a given day and $N$ is the maximum possible sunshine hours. Typical inputs to the model are shown in Table 1 and outputs are shown in Table 2.
### Table 1. Typical Inputs

<table>
<thead>
<tr>
<th>Month</th>
<th>Mean Temperature (°C)</th>
<th>Wind velocity (Km/day)</th>
<th>Relative humidity (%)</th>
<th>Sunshine hours on a given day (n)</th>
<th>Extraterrestrial radiation (mm/day)</th>
<th>Maximum possible sunshine hours (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>14.3</td>
<td>86</td>
<td>47</td>
<td>8.7</td>
<td>10.13</td>
<td>10.74</td>
</tr>
<tr>
<td>February</td>
<td>16.35</td>
<td>95</td>
<td>37</td>
<td>9.1</td>
<td>11.83</td>
<td>11.33</td>
</tr>
<tr>
<td>March</td>
<td>20.6</td>
<td>112</td>
<td>31</td>
<td>9.7</td>
<td>13.86</td>
<td>12.00</td>
</tr>
<tr>
<td>April</td>
<td>24.7</td>
<td>121</td>
<td>29</td>
<td>10.5</td>
<td>15.38</td>
<td>12.69</td>
</tr>
<tr>
<td>May</td>
<td>26.9</td>
<td>173</td>
<td>36</td>
<td>11.4</td>
<td>16.40</td>
<td>13.27</td>
</tr>
<tr>
<td>June</td>
<td>24.8</td>
<td>190</td>
<td>65</td>
<td>8.5</td>
<td>16.62</td>
<td>13.65</td>
</tr>
<tr>
<td>July</td>
<td>21.8</td>
<td>181</td>
<td>88</td>
<td>4.2</td>
<td>16.52</td>
<td>13.46</td>
</tr>
<tr>
<td>August</td>
<td>20.4</td>
<td>173</td>
<td>91</td>
<td>3.6</td>
<td>15.78</td>
<td>12.97</td>
</tr>
<tr>
<td>September</td>
<td>21.25</td>
<td>121</td>
<td>80</td>
<td>6.6</td>
<td>14.46</td>
<td>12.30</td>
</tr>
<tr>
<td>October</td>
<td>22</td>
<td>69</td>
<td>47</td>
<td>9.5</td>
<td>12.55</td>
<td>11.61</td>
</tr>
<tr>
<td>November</td>
<td>18.8</td>
<td>52</td>
<td>37</td>
<td>9.3</td>
<td>10.63</td>
<td>10.94</td>
</tr>
<tr>
<td>December</td>
<td>16.2</td>
<td>52</td>
<td>41</td>
<td>8.7</td>
<td>9.63</td>
<td>10.64</td>
</tr>
</tbody>
</table>

### Table 2. Typical outputs

<table>
<thead>
<tr>
<th>Month</th>
<th>$C_T$</th>
<th>$C_U$</th>
<th>$C_H$</th>
<th>$C_S$</th>
<th>$C_E$</th>
<th>ET (Assuming monthly consumptive coefficient of 1) (mm/day)</th>
<th>Solar radiation ($R_s$) (mm/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>0.82</td>
<td>0.97</td>
<td>0.97</td>
<td>0.61</td>
<td>1.09</td>
<td>2.42</td>
<td>6.64</td>
</tr>
<tr>
<td>February</td>
<td>0.88</td>
<td>1.00</td>
<td>1.01</td>
<td>0.61</td>
<td>1.09</td>
<td>3.30</td>
<td>7.71</td>
</tr>
<tr>
<td>March</td>
<td>1.02</td>
<td>1.04</td>
<td>1.04</td>
<td>0.61</td>
<td>1.09</td>
<td>4.85</td>
<td>9.07</td>
</tr>
<tr>
<td>April</td>
<td>1.16</td>
<td>1.06</td>
<td>1.05</td>
<td>0.62</td>
<td>1.09</td>
<td>6.34</td>
<td>10.21</td>
</tr>
<tr>
<td>May</td>
<td>1.23</td>
<td>1.18</td>
<td>1.02</td>
<td>0.62</td>
<td>1.09</td>
<td>7.81</td>
<td>11.14</td>
</tr>
<tr>
<td>June</td>
<td>1.16</td>
<td>1.22</td>
<td>0.85</td>
<td>0.60</td>
<td>1.09</td>
<td>6.20</td>
<td>9.33</td>
</tr>
<tr>
<td>July</td>
<td>1.06</td>
<td>1.20</td>
<td>0.56</td>
<td>0.57</td>
<td>1.09</td>
<td>3.44</td>
<td>6.71</td>
</tr>
<tr>
<td>August</td>
<td>1.01</td>
<td>1.18</td>
<td>0.50</td>
<td>0.57</td>
<td>1.09</td>
<td>2.75</td>
<td>6.14</td>
</tr>
<tr>
<td>September</td>
<td>1.04</td>
<td>1.06</td>
<td>0.69</td>
<td>0.59</td>
<td>1.09</td>
<td>3.34</td>
<td>7.50</td>
</tr>
<tr>
<td>October</td>
<td>1.07</td>
<td>0.92</td>
<td>0.97</td>
<td>0.61</td>
<td>1.09</td>
<td>3.75</td>
<td>8.27</td>
</tr>
<tr>
<td>November</td>
<td>0.96</td>
<td>0.87</td>
<td>1.01</td>
<td>0.61</td>
<td>1.09</td>
<td>2.84</td>
<td>7.18</td>
</tr>
<tr>
<td>December</td>
<td>0.88</td>
<td>0.87</td>
<td>1.00</td>
<td>0.61</td>
<td>1.09</td>
<td>2.29</td>
<td>6.35</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY / FURTHER READING: