

MINE 432
Industrial Automation and Robotics in Mining

Lecture 10
Introduction to Artificial Intelligence

Background

Artificial Intelligence has experienced rapid growth over the past two decades. Numerous applications and development tools entered the market in the late 1980s and early 1990s and together with the continued growth of microcomputers, many processing plants now apply these methods routinely to solve real-world problems. As technical personnel have become more familiar with these methods, it is apparent there are limitations in the ease by which applications are developed. To ensure successful implementation, considerable skills in human psychology, computer programming, knowledge acquisition, knowledge representation, etc. are required. Without proper training, it is not simple to build successful and useful systems - ones with big payback!

Part of the problem relates to combinatorial complexity. As the number of system inputs and outputs increase, the possible combinations of data rise exponentially. Symbolic methods, although useful at providing explanations and access to the inner workings of a system, can be clumsy and onerous to develop and maintain. Researchers in AI are beginning to realize that more efficient techniques are needed to develop a support foundation for the symbolic representation of knowledge. A set of paradigms has now evolved into a unified field called Computational Intelligence which combines expertise from three main areas:

- Fuzzy Systems,
- Artificial Neural Networks and
- Evolutionary Computing.

These methods are being applied to support the high overhead associated with symbolic processing on today's microcomputers and workstations. Without knowledge of such methods, AI applications will remain trivial, slow to develop, and ineffective to maintain.

Fuzzy Systems

Fuzzy Systems are Expert Systems that use Fuzzy Sets and Fuzzy Logic to perform qualitative modeling or to control processes for which no model has yet been identified or developed from first principles. At one time, if an Expert System contained less than a hundred rules, it was considered small and trivial. Today, if a system has hundreds of rules to perform a similar task, then it probably doesn't use Fuzzy Logic, or it truly is a large domain. Fuzzy Logic allows rules to be generalized across a space-map by interpolation from rule node to rule node. While the answer provided by such a system is an approximation of the "correct" one, it is usually close-enough. The gains in search and storage efficiencies are remarkable.

Fuzzy Systems are adaptable, in that the mapping of fuzzy concepts across a Universe of Discourse can change in response to external forces. The context of a Fuzzy I/O Map (often called a Fuzzy Associative Memory) can be taken into account so that the term "tall", for example, means one thing to a pygmy and another to a giant. Such adaptability is often considered as "learning" since the system seems to change its frame of reference in response to external inputs. However, this is a rather limiting definition since it doesn't account for adding new concepts or for automated changes in the discretization of the Universe of Discourse.

Artificial Neural Networks

The area of Computational Intelligence that offers true learning capability is that of Artificial Neural Networks. The paradigms in this field directly model the human neuronal system and hence, researchers in this field are often referred to as "Connectionists".

Basically an input/output space map is modeled as a series of interacting nodes in a network. When a node fires (taking on a signal value > 0), the signal is amplified or suppressed by multiplying by a "weight" assigned to each connection between this node and another. This weight can take on a value between $-\infty$ and $+\infty$. At the receiving node, all incoming signals are added together giving a force that acts on the neuron. This sum is passed through a function to generate an output signal value between 0 and 1 for the neuron. This output is then passed to other nodes to which this node is connected. The process continues until all nodes are exhausted and the signal strength of all output nodes has been calculated.

The power of this method will be fully-exploited when parallel-processing hardware becomes widely available so that all neurons in the network fire simultaneously. Nevertheless, there are many examples of successful implementation of these systems with large numbers of connections (100 - 1000s) that perform well on today's high-speed microcomputers.

The main advantage of a neural network is its ability to learn from experience or adapt to changing circumstances. Learning begins by selecting a series of random weights and presenting the network with one set of known input/output values from a large collection of training examples. The network is fired and predicted outputs are calculated. These are compared with the desired outputs and the errors are passed back through the network using a regression analysis or gradient-descent technique. New weight values are derived and the network is now presented with a new set of I/O values from the training data. This process continues until the overall error lies within a predefined tolerance level or until a selected number of iterations have been executed.

The trained network is then placed in service to deal with the environment it was designed to handle. Its performance is monitored on a regular basis by examining the accuracy of its predictions. Should it drift away from acceptable output due to changing

external forces, the network can be removed temporarily from service to be retrained using more up-to-date data.

There are a number of ways in which a network can be configured - the Perceptron model, the 3-layer Back-Propagation model, a Kohonen network with unsupervised learning, Cerebellum Model Articulation Control (CMAC), Holographic networks, etc. Each has its own unique set of advantages and disadvantages. For instance the Perceptron model cannot deal with problems that have non-separable mappings in space. Kohonen networks are used to classify input patterns rather than perform process control tasks. Back-Propagation methods take considerable time to learn and can be trapped in local minima regions.

Holographic networks were developed by AND Corporation, Toronto, Canada. Unlike other approaches this model is not a connectionist approach -- rather it attempts to model single neurons responding to frequency-stimulated inputs. All training is done by examining all data at one time rather than one set of data at a time. The matrix used to enfold the data onto the synapse of the cell is derived by a process akin to vector arithmetic. The speed of learning is orders of magnitude above that of Back-Propagation. Higher accuracy is also claimed.

Evolutionary Computing

There has been much excitement in recent years in the field of Genetic Algorithms -- a concept based on Darwin's criterion of "Survival of the Fittest". Genetic Algorithms and/or Genetic Computing are methodologies that automate the search for a better solution to a problem or set of problems. The process involves formulating possible solutions to a problem as a string of numbers much like the representation used to represent DNA (or a chromosome). These solutions are then "mated" two at a time to create "child" solutions with characteristics different from either parent but partially derived from each. Children that are more "fit" are allowed to procreate among themselves while those that are weaker are forced to resign or die-off.

The method is an adaptive parallel-search strategy that locates the true global minimum or maximum position of an output function without becoming trapped within local minima or maxima positions. The algorithm uses the following steps:

- Transformation of data into a chromosomal representation;
- Selection of the initial population of solutions;
- Selection of a suitable objective function to rate the fitness of the solutions;
- Application of genetic operators to alter solution composition during reproduction.

These operators include: reproduction, cross-over and mutation. Reproduction selects the number and position of alleles in the chromosome to be placed in the child solution. Cross-over causes certain selected genes to be transposed in the child. Mutation causes certain inherited genes to be altered randomly. Mutation does not provide better solutions

on its own but can serve to give a periodic push to the solution population to move out of a local minima position. This process continues until all solutions converge to a stable position. It is claimed that GA is faster than any other optimization method and avoids local minima traps.

There is considerable interest (perhaps at an academic level only) in applying this technique to a field called "Artificial Life". There are many popular books available today which discuss the creation of artificial populations of "bytes" based on functions that are optimized using a Genetic Algorithm. Graphical representation of these populations has given rise to the suggestion of new forms of life. A recent television documentary actually discussed the procreation of desk lamps using a computer program. We will not deal with such "games" but one can imagine such ideas eventually generating innovative novel concepts in the future.

There is a company in Florida using GA to project stock purchases and sales. It apparently has produced suggestions that work for unknown reasons. Perhaps these techniques are able to discern patterns in data that humans cannot fathom. The ability to explain or justify the solutions generated will certainly be necessary if these approaches are to be widely accepted by industry.

Development of Computationally-Intelligent Real-Time Systems

A real-time system must function in an environment where resources are shared in an efficient and effective manner. A multi-tasking operating system is a prerequisite for correct operation in real-time. Decisions must occur at speeds able to address process requirements and not delay other modules from completing their tasks on time. To properly organize such a system, it is necessary to place the important processing functions into separate modules.

Each module performs a major task such as: data-storage, message-transfer, alarm-sensing, event-scheduling, process-communication, user-communication, knowledge-processing, knowledge-explaining, etc. These modules interact with one other, with the user, and with the process itself. As the system grows in size and complexity, availability and turn-around of resources decline until eventually, more hardware (more memory or more CPU) is needed.

To delay the onset of such problems for an expanding system or to design for maximum efficiency with currently available resources, intelligent computation methods must be considered at the initial stages of development. Traditionally, Intelligent SCADA systems have been applied for supervisory control service only. Typical turn-around times (cycle times) for knowledge base-processing are measured in seconds depending upon the size of the problem. Typical data update times are 1 second to ensure efficient utilization of system resources. For some applications, this delay is too large -- certainly, for direct regulatory control, this sampling interval is inappropriate. Also for rapidly-changing variables that indicate important process transformations, much faster measurements are necessary.

To apply a knowledge base system to sampled, batch data such as that derived from an on-stream XRF system or to temperature profiles on steel billet surfaces, or to the thickness profile of paper on a roll from a paper mill, it is necessary to sample data at much accelerated rates. Suppose one wants to record temperatures on a steel billet as it passes in front of an optical pyrometer. The duration of measurement is relatively short (~2 seconds) but the intensity of data output is substantial (perhaps as high as 2000 points). The sampling time interval must decrease to a millisecond to accomplish such a task.

Storage of this data (much of which is redundant) within the conventional database of a SCADA system is inappropriate. RAM requirements are exorbitant and update delays are obvious. A means to acquire this data and apply pre-filtering are necessary to integrate these tasks into an existing SCADA system. A special data acquisition board can be used with on-board RAM and, in some cases, an on-board CPU. These boards range in price from \$2000 to \$5000. Up to 10,000 points per channel with up to 16 channels can be handled on a board with 4Mb of RAM. The data are pre-filtered using an intelligent communication driver.

These drivers are C-code programs to establish communication between the on-board data and the RAM-resident datafile used by the SCADA system. Functions to interpret shapes and features in the data act as filters. All functions and channels can be configured using a simple ASCII initialization file. A heuristic approach is used to ensure rapid computation of the required features or filtered data. These modules can detect well-understood changes in data and supply the AI supervisory module with symbolic knowledge about such occurrences. Without the use of such modules, AI methods are not fast enough to interpret these changes.