ARTIFICAL INTELIGENCE

University of Washington – CSEP 590A Alternate Computing Paradigms - Spring 2008

Hammad Hashmi, Ovidiu Elenes, Shahid Razzaq, Mir Tariq, Troy Schneringer, Leonardo Viana, Fabricio Voznika

INTRODUCTION

In 1950 Alan Turing first proposed the question “can machines think?” Using a simple game of imitation, Turing presented a simple test of “machine intelligence” whereby one could answer his question. While the game is somewhat simplistic in nature, his musings gave rise to the birth of the field of Artificial Intelligence and soon found a community of researchers dedicated to the pursuit of creating intelligent machines (1). Dartmouth College fed the research frenzy through the early years of AI through the aid of John McCarthy et al and the summer institutes of AI research started in 1955 (2).

Soon however, the reality of Moravec's Paradox set in declaring that which is difficult for humans is easy for computers, and that which is easy for humans is difficult for computers (3). Machines perform well on the most difficult of intelligence based problems such as differential calculus, while they fail to thrive at the simple tasks of human life such as facial recognition and mobility. Couple this with the explosion of the computational complexity and the hardware requirements needed to perform those computations, studies in artificial intelligence has proven to be most challenging (4).

While still a blossoming field of study within computer science, artificial intelligence has experienced a roller coaster of popularity and is currently enjoying a renewed focus as modern technologies are opening the doors to new research. In this paper we will look at some of the popular and emerging fields of current AI research that are bringing intelligent computing to life both in hardware and software solutions, including statistical learning, evolutionary computing, and AI hardware.

STATISTICAL LEARNING

Intelligence can be classified as a series of decisions that are made based on a given set of inputs to achieve a desired output. At base, each decision is chosen based on the likelihood of achieving success for the problem at hand. To this end the broad category of statistical learning encompasses agents that attempt to mimic human intelligence by analyzing inputs, assigning probabilities and making intelligent decisions to reach success. At a surface level, such problems may seem simplistic in nature but consider the following complications: a typical agent has only partial information about factors that affect the consequences of the decision; the measurement of these factors may be prone to error; and there will likely be additional unknown factors that affect the outcome of a given decision. Statistical learning addresses such problems and derives conclusions about complex environments which can only be partially observed by using principles of probability and statistical theory.

The following discussion will provide a brief survey of three popular statistical learning approaches: Bayesian Networks, Markov Networks and Kernel Machines.

BAYESIAN NETWORKS

An 18th century English cleric, Thomas Bayes, derived a seminal rule for determining conditional probability that was later extended to a broader set of problems whereby the degree of confidence is more important in decision making than observing the actual occurrence of events (5). Bayesian networks, as they have become known, address statistical learning problems by modeling the semantics of an environment, capturing the subjective nature of the input information, and making intelligent decisions based on uncertain or difficult dependencies.

Bayesian Networks model the interrelationship between variables by means of a Directed Acyclic Graph (DAG), where the nodes are the events of interest, and the conditional probabilities associated with edges are adjusted over time. The DAG in turn provides an efficient structure for calculating probabilities of choosing one node given another (6). In essence, Bayesian networks are similar to human's empirical study of laws of physical sciences (e.g. the correlation between smoking and cancer) before a theoretical causal relationship was established.

The full specification is as follows (7):

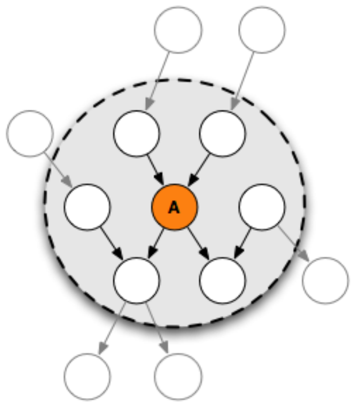
- A set of random variables make up the nodes of the network
- A set of directed links connect node X and Y if, and only if X is said to be a parent of Y
- Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node
- The graph as no directed cycles

MARKOV NETWORKS

A Markov network, named after Andrey Markov, is a stochastic process with the property that given the current state, all future states are independent of the past states (6). In other words, the description of the present state fully captures all the information that could influence the future evolution of the process. Future states will be reached through a probabilistic process instead of a deterministic one. Markov networks are classified on their Markov chain properties:

- Reducibility: reachibility from one state to another
- Periodicity: regular occurrence of states
- Recurrence: repetition of states
- Ergodicity: finite states with recurrence in no particular order

Markov networks are an extension of the Bayesian networks in their representation of dependencies (7). While Markov networks can represent dependencies that a Bayesian network cannot, such as cyclic dependencies, they fail in representing other dependencies, such as induced dependencies.



Sample Markov Networks. A node is independent of all other nodes given a special set of nodes, called its Markov Blanket

A Markov network consists of an undirected graph where vertices represent variables and edge represent dependencies; a set of potential functions map possible joint assignments to non-negative real numbers. Similar to Bayesian networks, conditional distribution can be calculated by summing all possible assignments whose values are not given (8).

Bayesian and Markov networks have broad applications from 3D scene creation to speech recognition. Recently, Domingos and Pazzani (1997) provided an explanation for the surprising success of simplified Bayesian reasoning even in domains where the independence assumptions are violated and thus offering a general framework for great simplification of complex environments.

KERNEL MACHINES

In the field of machine learning, the use of linear machines is usually the first approach used when facing pattern analysis problems. The simple mathematical form of linear methods allows for the development of simple training algorithms and the detailed study of individual properties. The extension of these linear machines to non-linear decision rules using the *Kernel Trick* is referred to as Kernel Machines. This trick only works for algorithms where all training sample vectors are in the form of Euclidean dot-products. The most popular algorithms which work with kernel methods are Support Vector Machines, Principal Component Analysis, and Fisher's Linear Discriminate Analysis (9).

KERNEL TRICK

The *Kernel Trick* is a method which enables the use of a linear classifier algorithm to solve a non-linear problem by mapping the original non-linear observations into a higher-dimensional space. The linear classifier can then be used to make an equivalent linear classification in the new higher dimension space. All the dot products in the form *x.y* used in the linear algorithm are replaced by a Kernel Function *K(x,y)* if it satisfies the conditions specified in Mercer's Theorem (10).

KERNEL FUNCTION

A Kernel Function is the function of distance that is used to determine the weight of each training example with respect to the data which needs to be classified (11). In other words the kernel function is the function K such that, $w_i = K(d(x_i, x_q))$.

SUPPORT VECTOR MACHINES

Many applications of machine learning address problems where both the size of sample data and the number of attributes for each example is large such as text classification. Linear Support Vector Machines are among the most popular machine learning techniques for such high-dimensional and sparse data (12).

Support Vector Machines (SVN) view every sample in the training dataset as an n-dimensional vector and try to separate them in different classes using an n-1 dimensional hyper-plane. Since there may be many hyper-planes that classify the sample data correctly, SVMs try and maximize the separation between classes. This means picking the hyper-plane which maximizes the distance from the hyper-plane to the closest example data point, which also results in the simultaneous minimization of the classification error. In the figure H1 doesn't separate the 2 classes. H2 and H3 both separate the 2 classes; however H3 does it with the maximum margin (13). Support Vector

Machines are being used to solve problems such as drug design, word-sense disambiguation and Web Searching.

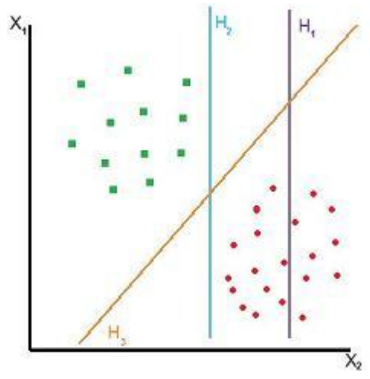


Figure: H1 doesn't separate the 2 classes. H2 and H3 both separate the 2 classes, however H3 does it with the maximum margin.

EVOLUTIONARY COMPUTING

The natural world is replete with pictures of intelligence that emerges over time through the evolutionary process. Natural scientists study nature to determine how species change and grow over time, and have inspired computer scientists to attempt to model the evolutionary behaviors through software in an attempt to derive more intelligent solutions.

In this section we will discuss two such modeling attempts: ant colony algorithms and genetic algorithms.

ANT COLONY ALGORITHMS

Ant Colony algorithms were first introduced by Marco Dorigo in 1992. Inspired by the scavenging behavior of ants which lead the colony to food, they provide a way of deriving optimal paths within a graph.

When traveling from the colony to food sources, ants deposit chemicals called pheromones on the trails. The trail is used as a bread crumb trail of sorts that allow the ants to find the path back to the colony. As other ants find this marked path they are likely to follow it. This will cause more pheromones to be deposited on the trail, which in effect reinforces the strength of the trail. Over time, however, the pheromone starts to evaporate, and this reduces its attractiveness to other ants. Short paths have the advantage of being marched over faster and more often, therefore, the pheromone density remains high. Consequently, a short path will have more ants traveling on it, increasing the pheromone density, and eventually the majority of ants will follow it.

Ant Colony algorithms mimic this behavior in order to find optimal paths within a graph. Initially, all paths have a random small amount of "pheromone" deposited on it. An "ant" departs from the starting node and starts the process of visiting the other nodes in the graph. At each node, the ant decides which node it should visit next. The attractiveness of the node is based on its derived pheromone level such that a nearby node with a large amount of pheromone will be have a higher probability of being chosen than a distant node with little pheromone (14).

Ant Colony algorithms excel at producing strong results relatively fast. They can be run continuously in order to adapt to changes in real-time, an advantage compared to genetic algorithms (15). For instance, an obstacle such as a traffic jam would quickly lead an ant colony algorithm to discover a different good path, whereby a genetic algorithm would rely on the evolutionary process over time to adjust.

Ant colony algorithms are of special interest in applications such as urban transportation and network routing

GENETIC ALGORITHMS

John Holland fathered the study of Genetic Algorithms (GA) in 1975 in the first paper published on the subject, "Adaptation in Natural and Artificial Systems." Conceptually genetic algorithms are simple, as they attempt to model the natural evolution. A population of individual solutions to a problem is either generated randomly or following certain rules to force a better initial sampling. A fitness function is then applied to each individual measuring the quality of the solution. Higher fitness values mean the individual is closer to the optimal solution. Next, one or more individuals are selected to mate and create new offspring. This process is called crossover, as it tries to simulate how DNA is broken and fused together to create new cells. By combining individuals, the hope is that the offspring will evolve and become better than their parents. Furthermore, individuals with higher fitness value are more likely to be chosen to mate. In addition to crossover, random mutations can also aid in introducing variance into the resultant population (16).

Typical implementations of genetic algorithms model individual solutions in ways that can be easily broken and rejoined, such as arrays (17). Algorithms then randomly select a place to break the individual into fragments and recombine with another individual that was also broken. More sophisticated algorithms exist and usually are tailored for a specific problem. For example, data mining GAs can decide to break a given rule where it generates most hits instead of purely random breaks (18).

In developing an algorithm that closely models the evolution of the natural world, several factors can play into the quality of the resulting solutions and the length of time it takes to derive the solution. Common parameters include number of generations, population size, rate of evolution, etc. Just as the factors that affect natural reproduction are numerous, usually these parameters are used in conjunction with one another.

Common application of genetic algorithms include: data mining, the traveling salesman problem, game theory, and protein folding (19).

AI HARDWARE

The shortcomings of hardware performance have often been blamed for the slow progress made in emulating human intelligence. In fact, Hans Moravec predicts that it won't be until 2003 when regular (consumer) hardware will catch up to raw power of the human brain (20). Consider the numbers that must be closed between brain power and computing power. The brain contains 100 billion neurons with 1,015 synapses each on average, while the most powerful supercomputers operate at 1,015 teraflops with 1,015 Petabytes of memory. There exists a virtual chasm of difference between the two and scientists are researching new hardware implementations that attempt to bridge this gap.

In this section we will discuss two current topics of intelligent hardware research that are making strides toward realizing faster, more intelligent hardware: Neuromorphic Engineering and Artificial Life.

NEUROMORPHIC ENGINEERING

Currently, the successful computing machines that we know of are the brain and digital computers (21). However, unlike man made machines, the brain has significant advantages over its silicon counterpart, such as parallel processing, reliability (despite unreliable components) and self configurability. Moreover, the brain is event-driven, stochastic and parallel while the computers are result-driven, deterministic and serial. The differences as the stand currently are hard to fathom, from the molecular level to the more abstract conceptual level, but the nascent field of Neuromorphic Engineering is very promising and can take Artificial Intelligence in new domains not yet explored through previous technologies and concepts.

The term neuromorphic was coined by Carver Mead, in the late 1980s to describe Very Large Scale Integration (VLSI) systems containing electronic analog circuits that mimic neuro-biological architectures present in the nervous system (22). Neuromorphic Engineering is the emulation of the brain in hardware, along with its neuro-biological architecture of the nervous system (23). It differs from biomorphic engineering in that it only attempts to model the control and sensory systems rather than an entire biological system. It attempts to mimic the basic interactions of individual neurons and proper implementations must have the following properties:

- Carrying out Computations: Using sensory data input to come up with the appropriate series of actions.
- Information Representation: Creating a model of the environment.
- Robustness to Damage: Working around breakdown of certain cells or paths so that the system as a whole continues to function.
- Learning and Development: Interpreting information temporally to yield a better understanding of the environment.
- Evolutionary Change: To alter the interconnection of components to evolve current behavior and come up with new solutions.

In Neuromorphic Engineering, models of neural systems are implemented as analog, digital or mixed-mode analog/digital VLSI systems. Analog technology appears to be the most appropriate (both in terms of power and behavior) for implementing artificial neurons that behave in a biologically plausible way. Field programmable analog arrays (FPAAs) are another option that can enable rapid prototyping of chips, and make training and use a lot easier. There is an important distinction between two areas in Neuromorphic Engineering: Neuromorphic Computation and Neuromorphic Modeling. While the former is concerned with using a subset of neuronal properties to build neuron-like computing hardware (membrane ion channel, firing behavior, conductance), the latter is involved in the more abstract concept i.e. reproducing the neurophysiologic phenomenon to increase our understanding of nervous systems.

The term Neuromorphic hardware is used for full custom designed integrated circuits that are the product of neuromorphic engineering. Currently, most examples of this type of hardware are constructed using analog circuits in complementary metal-oxide-semiconductor technology. The correspondence between these circuits and neurons (or networks of neurons) can exist at a number of levels. At the lowest level, the correspondence is between field effect transistors and membrane ion channels. At higher levels, the correspondence can be between filters/amplifiers and whole conductances/firing behavior. Similarly, neuromorphic engineers can choose to design Hodgkin-Huxley model neurons, or reduced models, such as integrate-and-fire neurons. In addition to the choice of level, there is also choice within the design technique itself; for example, resistive and capacitive properties of the neuronal membrane can be constructed with extrinsic devices, or using the intrinsic properties of the materials from which the transistors themselves are composed.

This growing area covers topics such as sensory systems that can compete with human senses and pattern recognition systems that can run in real time.

Neuromorphic engineering commonly finds its usages in vision, hearing, sonar, speech processing, robotics and learning.

ARTIFICIAL LIFE

AI traditionally focused on machines that perform complex multi-function problems (chess playing, medical diagnosis, etc), while Artificial Life (Alife) is preoccupied by natural behaviors, evolution and dynamic environments. Alife is a field that explores natural life by attempting to artificially create biological phenomena from scratch within computers and other nonliving media. Alife has three branches, named for their approaches: soft (software); hard (hardware) and wet (biochemistry) (24).

ORIGINS OF ALIFE

While many have contributed to the study of Alife, Chris Longton is considered to have brought theoretical coherence in this field and was the first to organize a workshop for Alife in 1987 entitled "Interdisciplinary Workshop on the Synthesis and Simulation of Artificial Life". The workshop manifesto stated that the ultimate goal of artificial life is to mimic the logical form of living systems (25).

CARBON VS NON-CARBON BASED LIFE

While life on earth has carbon as its primary chemical element, the question is asked that if we have silicon, germanium or any other element as the base element, what would life look like? Scientists, like Charles Tayler (UCLA) and David Jefferson (LLNL), believe that Alife models could demonstrate how cellular organization begins, interacts between genotypes, and creates parasites etc. Speculation is limitless but Alife supporters believe that digital simulation of non-carbon life can present answers to that question (25).

WEAK VS STRONG ALIFE

There is a major difference in Alife community between "weak" and "strong" Alife. Weak Alife advocates claim that simulation of evolving systems may help them understand real biological life natural systems. This group of people, and especially experimental biologists, are reluctant to consider computer based experiments as valid approaches. At the opposite pole there are "strong" Alife supporters. They consider that replicating life using computer programs is as equally valuable as natural life experiments. Rob Knight (biologist, University of Colorado) believes strong Alife views are controversial considering biological life is complex and the similarities with biological evolution tend to be abstract (25).

EXAMPLES OF ALIFE PROJECTS

Projects that have gained traction within the field of Alife are:

- Golem (Genetically Organized Lifelike Electro Mechanics - Brandeis University): a good example of engineered virtual life involving robots that can design and build other robots. The computer is running an evolutionary algorithm producing a design based on natural selection (25).
- The Tierra Project (Tom Roy - University of Oklahoma): simulates in a virtual world how life started during Cambrian Era. The "organisms" were represented by assembly language programs that were competing for CPU time and memory (25).
- Avida Project (C. Adami - CalTech): similar to the Tierra project with the main difference being that particular types of mutations were inserted in program and observing what happened in following generations (25).

THE FUTURE OF ALIFE

As one would expect, artificial life is very controversial. John M. Smith (biologist) stated that Alife is a "fact-free science". Lately, it seems that Alife techniques are becoming more accepted as method of studying evolution. At the same time scientists such as Francis Collins (head of Human Genome Project) think that chances of creating life from scratch in a lab are implausible; he believes that life is a product of intelligent design (26).

IN SUMMARY

As the explosion of complexity continues in modern computing it is apparent that new approaches must be employed to solve problems that are beyond human comprehension and reach the limits of our current computational power. Whether through intelligent software or life-like hardware, artificial intelligence provides an ideal alternative to modern approaches in that it empowers machines to adjust the approach for solving a problem. The ability to build intelligence into machines in such a way that they can analyze and optimize a solution expands our current computational limits beyond comprehension.

REFERENCES

1. *Computing Machinery And Intelligence.* **Turing, Alan.** 1950, Mind, pp. 433-460. http://loebner.net/Prizef/TuringArticle.html.

2. **McCarthy, J., et al.** A Proposal For The Dartmouth Summer Research Project On Artifical Intelligence. August 31, 1955.

3. **Moravec, Hans.** *Mind Children: The Future of Robot and Human Intelligence.* s.l. : Harvard University Press, 1988.

4. *Reducibility Among Combinatorial Problems.* **Karp, Richard M.** 1972, Complexity of Computer Computations, pp. 85-103.

5. **Wikipedia.** Thomas Bayes. [Online] http://en.wikipedia.org/wiki/Thomas_Bayes.

6. *Efficient Structure Learning of Markov Networks.* **Lee, Su-In, Ganapathi, Varun and Koller, Daphne.** s.l. : Stanford University.

7. **Russell, Stuart and Norvig, Peter.** *Artifical Intelligence, A Modern Approach.* s.l. : Prentice Hall, 1995.

8. **Wikipedia.** Markov Network. [Online] http://en.wikipedia.org/wiki/Markov_network.

9. Kernel-Machines.Org. [Online] http://www.kernel-machines.org/.

10. *Pattern Recognition with Support Vector Machines.* **Lee, Seong-Whan and Verri, Alessandro.** Niagara Falls, Canada : s.n., 2002. First International Workshop, SVM.

11. **Mitchell, Tom.** *Machine Learning.* s.l. : McGraw Hill, 1997.

12. *Training Linear SVMs in Linear Time.* **Joachims, Thorsten.** s.l. : Cornell University.

13. **Burges, Christopher J.C.** A Tutorial on Support Vector Machines for Pattern. [Online] http://research.microsoft.com/~cburges/papers/SVMTutorial.pdf.

14. **Colin, Andrew.** Dr. Dobb's journal. [Online] http://www.ddj.com/article/printableArticle.jhtml;jsessionid=I3POWR5NA02ICQSNDLPSKH0CJUNN2JVN?articleID=191800178&dept_url=/hpc-high-performance-computing.

15. **Wikipedia.** Ant Colony Optimization. [Online] http://en.wikipedia.org/wiki/Ant_colony_optimization.

16. **Koza, John R.** *Genetic Programming.* s.l. : MIT Press, 1998.

17. **Banzhaf, W., et al.** *Genetic Programming An Introduction.* s.l. : Morgan Kaufmann Publishers, 1998.

18. *Discovering fuzzy classification rules with genetic programming and co-evolution.* **Mendes, R.F., et al.** s.l. : Springer, 2001. Principles of Data Mining and Knowledge Discovery. pp. 314-325.

19. **Freitas, A.A.** *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* s.l. : Springer-Verlag, 2002.

20. *When will computer hardware match the human brain?* **Moravec, Hans.** 1997.

21. **Diorio, Chris.** Why Neuromorphic Engineering? [Online] http://www.cs.washington.edu/homes/diorio//Talks/InvitedTalks/Telluride99/sld001.htm.

22. **Wikipedia.** Neuromorphic. [Online] http://en.wikipedia.org/wiki/Neuromorphic.

23. Institue of Neuromorphic Engineering. [Online] http://www.ine-web.org/.

24. **Wikipedia.** Artificial Life. [Online] http://en.wikipedia.org/wiki/Artificial_life.

25. **Forbes, Nancy.** *Imitation of Life – How Biology Is Inspiring Computing.* s.l. : MIT Press, 2005.

26. **Krulwich, Robert.** Francis Collins Interview. [Online] http://www.pbs.org/wgbh/nova/sciencenow/3214/01-collins.html.