Examining the Mechanisms of the Human Brain with Computer Science

Arvind Ravichandran

Computer System Research

2005-2006

Abstract

Artificial Intelligence is a growing field today. Many of the amenities we have become accustomed too (such as spellcheck) rely, in some part, on artificial intelligence. Yet the holy grail of artificial intelligence is the simulation of the human brain: no one has ever approached this pinnacle of science and many doubt its very feasibility. While the author of this paper makes no such attempt, he attempts to harness Artificial Intelligence for a different purpose – to study the mechanisms of the human brain. One popular theory for the mechanism of learning in the human brain is pattern recognition – the eye sees the mouth move so many times that it can recognize and eventually mimic, thus speech is formed. To do this study, the nature and field of artificial intelligence was first studied, then a program mimicking possible analogous pattern recognition in the brain was constructed, and finally it was tested in the learning process in the game of tic-tac-toe against a perfect computer.

Introduction

How does the Human mind work? How is something so small powerful enough to completely separate humans from “animals?”

Current biological and neurological studies have isolated the apparent underlying mechanism in the brain – neurons. These cells essentially operate and communicate as on-and-off switches; there are millions in the body and they somehow integrate before they even reach the brain to be processed.

Obviously, simulating the neuron from an entry-level level would be impossible, so this project divided into 3 stages: studying artificial intelligence today, formulating a program, and testing that program in a game of tic-tac-toe.

Ultimately, the goal was to use the analysis from the tic-tac-toe learning simulation to assess the validity of a prevalent theory of learning – pattern recognition. The brain “sees” in image multiple times until it is eventually able to mimic it: a possible explanation for the acquisition of speech.

Background

Artificial Intelligence Today

Artificial Intelligence is an integral part of our world today. For example, typing a few letters of this month into Microsoft Word 2003 and Word offers to fill in the date. How does it know that the user wants to type the date? How does it even know what the date is? This is artificial intelligence in a very pure, basic form. The artificial intelligence in Word is mostly hard coded: type in a little of the month and it will fill it in, type in a little of the user name and again Word will fill it in.

However, Word is not the only application with artificial intelligence. Artificial Intelligence is commonly found in Video games, and today the complexity is astounding. For example in Battlefield 2, Artificial Teammates actually work together to accomplish specific objectives while individually dashing in and out of cover to avoid tactical fire. Similarly, artificial teammates in Real Time Strategy games show amazing human-like capabilities. For example, in Rise of Nations, maps (playing fields) are randomly generated. Thus resources, the key to development, could be anywhere near the starting area, however Artificial Opponents show surprising capabilities to find and exploit these resources. They, too, work with teammates to coordinate attacks and defense.

Yet, Artificial Intelligence has a massive flaw. It is largely a hard-coded field. That is, the Artificial Intelligence code in Microsoft Word can not be used in Battlefield 2 or in Rise of Nations. Why? Artificial Intelligence, at the moment, must be hard-coded and specific. A number of algorithms in the above mentioned are accessible to the Artificial

Intelligence that help it determine the best course of action. For example, in Battlefield 2, the artificial intelligence has access to object collision, objective position, and enemy (human) movement in order to formulate its decision. In Word, the algorithm is hard-coded so that a series of characters immediately triggers a fill-in the blank response. Similarly, in Rise of Nations, the Artificial Intelligence has access to key algorithms that determine the nature of the map, allowing it to determine key battle points and objectives.

The methods, especially in Battlefield 2 and Rise of Nations, to determine the course of action are largely varied, including heuristic values and limited foresight planning. Again, however, they are completely specific. There is no way to translate artificial intelligence from Rise of Nations to Battlefield 2. So Artificial Intelligence, according to many, has stagnated since it is completely specific and the specific methods used have mostly been discovered.

Current Methods

There are a number of current methods used today to address various Artificial Intelligence needs. Depending on what needs to be addressed, computer scientists can model a program to perform certain functions. There are three current methods used in Computer Science extensively to address Artificial Intelligence needs. They are hard-coding, statistical analysis, and neural networks.

Hard-coding is simply programming in exactly what a response should be. For example, flying a rocket to Pluto (recently launched) is completely hard-coded; the instrument knows exactly when to shut off its accelerator, how much to turn it on and when to do it. If unforeseen circumstances arise, over ten billion dollars worth of research, time,

and equipment has no way of responding and will probably be scratched. On the other hand, hard-coding is very useful for simple circumstances: it is easy, from a testing standpoint, and the results are always known. Hard-coding is very useful for certain applications, but its overall use is extremely limited.

Statical analysis is a more mathematical-based approach. It attempts to follow statistical pattern and laws to derive consequences of significance. This method is extremely useful in the field of biology for DNA analysis. However, it can be applied to any field that has laws and follows some rules of math. Unfortunately, its overall use is limited due to lack of general application. This AI can not derive new methods of doing anything, because it must follow pre-programmed laws. Its best use is for simple data collection and analysis.

Neural networks is a relatively new method attempting to model the biological aspect of the way the mind works. In this method, classes and objects simulate neurons, the building blocks of the brain. The goal is to achieve cognitive emergence (development of complexity from simplicity) from these neurons. The success is ultimately mixed. Two problems are that humans do not fully understand how the human mind and neurons work, so the current rules on which these systems are based may be flawed, and that millions of neurons fire every second in the brain, this requires an enormous amount of computational power to simulate.

Ultimately, advanced AIs make use of multiple methods to achieve desired results. However, the AI is rarely capable of being applied to other situations (a rocket to Pluto's AI can not be used for a rocket to Jupiter), so the field seems largely redundant. Overall, current methods are effective but limited.

The Human Brain

Ultimately, all artificial intelligence seeks to achieve a level of intellect on par with that of the Human Brain. To date, no more powerful "processor" has been seen on the planet. The capabilities of the human brain are absolutely enormous; however the nature of their existence is hard to understand. From a biological perspective, the brain is a set of interacting neurons; there is no central processing unit. The entire brain seems to operate on simple neurons. Neurons themselves function as switches, either on or off. So how is it that this analog data becomes so complex? Ultimately, many theories have been proposed to answer this, and none have yet sufficed.

From where does thought emerge? Consciousness? It seems counter-intuitive for such simple sets of interactions to form such behavior. However, this is the current approach: "emergence." The idea that a number of a simple, small interactions combine to provide states and interactions previously unavailable. Ultimately, the Brain remains a mystery, although this project hopes to shed some light.

Theory

The ultimate heart of the theory in this project was simulating the Human Brain. However, no one has of yet to create anything even remotely close to the Human Brain, so I had to focus my efforts. The first identified problem was the actual act of learning. While most scientists agree that learning and processing new information is an in-born trait (you don't learn how to learn), the method is unclear so I had to decide which method to go with. I chose pattern recognition: identifying similarities in data.

The second issue was inputs. The human brain ultimately receives inputs from each neuron on the body's skin along with the millions of cells in the eyes, ears, mouth, and nose. All of these inputs contribute significantly to form the overall impression the brain perceives, however, simulating this many inputs in one second would require multiple supercomputers. Thus I chose to neglect actual inputs and simply simulate data that was specifically pertinent.

Finally, the amount of emergence I would allow for was difficult. Simulating neurons was unlikely, because it would require too many calls and force me to approach a cellular level for effective simulation. Thus I chose to simulate packages of neurons. That is rather than simulating each neuron's interpretation of a visual scene, I chose to simulate the data of the scene itself and package that.

Design Criteria

The ultimate result of my exploration into the options presented me with a dilemma. I could not realistically hope to simulate everything that I wanted to. However, I wanted to make sure the program could be expanded as needed. So I decided to have layers of abstraction. With each progressive "layer" in the program, the data would become more and more abstract. Thus the learning-center, or the brain, was working with purely numerical data that could be simulated anywhere.

Along with the layer of abstraction, I chose to use a "plug-and-play" module approach. A person could simulate data in a module, send it to the appropriate source and be able to use the program without dealing with other non-sense. In order to effectively use this approach, I needed to compartmentalize my program, separating the brain from the layers of abstraction.

However, this presented the problem of data transmission. How do I move data around, if layers don’t know they exist. The answer was a set of messages and a message-processing unit. Each message contained relevant data, and the message-processing unit interpreted where to send the data, whether the message explicitly stated it or not.

Finally, since I did not have time to simulate the learning of the algorithm, I had to simulate one in the brain. This process would have to be the hardest, because I had to simulate abstract learning. The brain could only work with numbers, so I had to simulate processing and return of information purely abstractly. I also had to simulate memory in the brain, which I did with text files.

Results

The testing was divided into two phases. The first being the testing of the core of the program and the second was the testing of the learning of tic-tac-toe.

The first test of the core program was the message processing unit was the passing of arbitrary data. In this case, the program included 3 elements: a simplified "brain" - it only had the ability to store data that it received, a message processing unit (the focus of the testing), and a console element for the input of data. User-end input was a set of 4 digit numbers. The goal was for this data to be read by the console, passed on by message units to the message processing unit, and finally to make their way to the brain. The results were very favorable (note that the specific numbers can be found in table R1). The program was able to pass the data, uncorrupted, through the program from the user-end input to an output text file.

Table R1:

User end Input	Output Text File
1001	1001
0000	0000
5467	5467
9999	9999

The second test of the core program was numeric-translating capability. In order to test this, the goal was for user-end inputted alphabet to be appropriately converted to numeric data at the brain end. For this test, a simple brain that could only store data, a message processing unit, and a console plug-in was used. The user inputted each letter in the alphabet in alphabetical order individually. The data was then examined from the output text file. The numeric results reflected that the computer had indeed successfully translated the data (note that specific numbers can be found in table R2).

Table R2:

Alphabetical Character	Converted Numeral
a	1001
b	1002
c	1003
d	1004
e	1005
F	1006
g	1007
h	1008
i	1009
j	1010
k	1011
l	1012
m	1013

n	1014
o	1015
p	1016
q	1017
r	1018
s	1019
t	1020
u	1021
v	1022
w	1023
x	1024
y	1025
z	1026

A third test had to be performed on the pattern-recognition aspect of the brain. The goal was the for the computer to correctly identify the pattern of "01"s in the data. To perform this test, a brain that could store and read data, along with examine it, (this is the most complicated brain that the program ever used), a simple message processing unit, and an input text file was used. The brain outputted to the console the number of it recognized repeated two-digit patterns (note: only 01 patterns were present). The results were moderately favorable and cane be found in table R3.

Table R3

Numerical Pattern	Correct Number	Computer Report
01010101	1	1
0102020101	2	3
010101020203030101	2	4
010201	2	1
020202020101	1	2

The final test was the tic-tac-toe game. The game itself "plugged in" to the already developed AI. The goal for this test was to establish a computer that would never lose and take any open wins. In order to perform this test, I used the same brain as described above, a message processing unit, and a tic-tac-toe game that had an unbeatable AI (so the computer could play itself). Results were measured by reported tie percentage (since the perfect AI would take any win). The results were not favorable, as the computer never improved, the specific numbers can be found in table R4.

Table R4:

Trial #	Tie Percentage (out of 1000 attempts)
1	.1
2	.2
3	0
4	.1
5	0

6	0
7	0
8	.2
9	0

Discussion

The first and most important note I made was that the AI was not improving because it could not distinguish a win from a loss. This raised the fundamental question of psychology, how does one know what is good and what is bad? The AI needed significantly more work in order to be able to distinguish favorable results from unfavorable ones without significant hard-coding.

Other problems the computer had were problems of generalization and specification. The AI could find patterns easily; however, it found more patterns than there really were. For example, since a win and a loss are separated by only a few digits, the AI could find patterns between the win and the loss, and thus got easily confused. The computer had problems with specification because it couldn't. It could not understand what data was necessary and what was not.

The ultimate failure, I believe, with this program, was that it started at too high of an intellectual level. If I were able to simulate more of the brains neural capabilities, I would have started from a much earlier point. Possibly even neural data from the womb (which would be very limited).

However, it is important to recognize the successes of these efforts. I firmly believe that the above suggestion is completely possible within the confines of the program. Since one can always add more modules and incorporate more data. One major restriction would ultimately become CPU power, since with enough modules and enough data flying around, the CPU would bog down.

From a programming level, the modular approach and layers of abstraction were great successes. The modular approach allowed me to constantly add or delete parts of the program (besides the core) that were not necessary with ease. Furthermore, adding algorithmic specificity to the brain would apply to all modules. The program, thanks to the modules, has plenty of room to expand. The layers of abstraction go hand in hand with the modules. Because each progressive layer in the program lost specificity, adding new layers, new data, or new modules presented virtually no problems. However, I do not believe this approach is useful for non-scientific purposes, because each layer of abstraction and each module consumes more and more system resources.

Conclusion

The overall result of my research showed the initial strengths and weaknesses of my intended target area. It seems apparent to me that the computer science is ready to handle Artificial Intelligence, but human beings have not gained a deep enough understand of what truly defines intelligence to use computer science in such a manner.

I found, however, that using computer science to investigate and simulate otherwise unknown theories is a viable option. Using my program, I was able to see that pattern recognition independently would not be sufficient for complete and complex learning. I believe that computer science can be used in more cases to create possible models. However, admissibility may become an issue. The merits and accuracy of such models is debatable. Had my program been successful, the simulated model would not be admissible evidence for scientific inquiry in a scientific community. Thus it would seem that computer science may be used to simulate and test theories but not prove them.

Regardless, I gained significant insight into the fields of psychology and neurology. My research has made it apparent that human beings absolutely must be born with some knowledge of their surroundings. Recent psychological tests have indeed proved that certain signals (such as a smile indicated positive feelings) are universal. However, it is not clear as to how much and what kind of knowledge must be in born. Certainly, most knowledge is acquired, and it seems that pattern recognition is a key, but not single, component. Pattern recognition has merits: it is simple, repeatable, and generalized;

however, left undirected, pattern recognition does not provide results (as shown by the tic-tac-toe game). Also, neurologically, I doubt that data is stored in an analog format (as my program used). Early in the development, I began to realize the enormous amount of information the brain must process and it seems cumbersome and highly unnatural to process that much information in an analog format. Although the all-or-none mechanisms of neurons seems to dictate this approach, the processing in the brain must occur at digital levels in order to process so much information so rapidly.

Finally, my research indicated a strong future in more abstract programming. Rather than having many independent and un-related programs, as can be found today, the future is clearly a modular, abstracted approach. For example, modern companies must use different software for servers, desktops, word processing, spreadsheets, accounting, programming, record-keeping, employee-management etc. The amount of repeated and unnecessary code in each new piece of software is enormous. The number of man-hours needlessly consumed presents an opportunity for software companies to streamline software making. Rather than having so many individual applications, one overarching data-processor will be made. Beneath, a layer of abstraction would connect the processor to modules which would translate incoming data in and out of the necessary language. Thus repeated code would be minimized, productivity increased, and reusability increased. Furthermore, these programs would be “future-proofed” because the modules to translate data to and from a word-processor would not have to be changed or optimized, instead, with new technology and techniques, only the main processor would have to be optimized.

While I did not achieve my desired results, I believe I made significant progress in two important fields: neurology and computer science. I proved the usefulness of a modular, abstracted approach to programming along with proving that basic knowledge is necessary for proper human neural development.

Recommendations

For someone attempting to use a similar approach to the study of the human brain, I would recommend a number of things. First of all, to give the brain more time. In order to simulate effective learning, the computerized brain would have to run for at least one year. Secondly, in using the modular approach, choose not to quantify the data. Rather, use alpha-numerics. Third, start at a more basic level – begin with the senses rather than higher cognitive thinking. Simulating data from the senses would be an appropriate starting point for this kind of research. My greatest recommendation would be to not underestimate the task at hand. Understanding the human brain is a daunting challenge that many of the greatest minds of the world together have not accomplished. That does not mean it is not feasible, but rather, extremely difficult.

Bibliography

AIS Announces Markdown Optimization Software Aimed at Increasing Retailer Profitability. 16 Jan. 1996. 1 June 2006 <http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/01-16-2006/0004261084&EDATE=>.

Arai, Sachiyo, Katia Sycara, and Terry Payne. "Experience-based Reinforcement Learning to Acquire E_ective Behavior in a Multi-agent Domain ." Paper. 1 June 2006 <http://www.cs.cmu.edu/~softagents/papers/comPRICAI00.pdf>.

Artificial intelligence firm helps Air Force study enemies. 1 June 2006 <http://sanjose.bizjournals.com/sanjose/stories/2006/05/15/daily35.html>.

Boudreaux, Jill. Navy visits University for artificial intelligence gaming expertise. 26 May 2006. 1 June 2006 <http://www.unr.edu/nevadanews/detail.aspx?id=1685>.

Chang, Kenneth. "Intelligent Beings in Space!" New York Times 30 May 2006. 1 June 2006 <http://www.nytimes.com/2006/05/30/science/space/30rock.html?ex=1149307200&en=8e85c7dd4cc5b179&ei=5087%0A>.

Crouching Tiger, Hidden Robot. 14 Jan. 2006. 1 June 2006 <http://www.astrobio.net/news/modules.php?op=modload&name=News&file=article&sid=1834&mode=thread&order=0&thold=0>.

Miyashita, K, and K Sycara. "Learning Control Knowledge through Case-Based Acquisition of User Optimization Preferences." Knowledge Acquisition and Machine Learning: An Integrated Approach. 1 June 2006 <http://www.cs.cmu.edu/~softagents/learning.html>.

Poker Academy Texas Holdem Brings Artificial Intelligence Educational Tool to UK and Scandinavia. 1 June 2006 <http://www.poker777.com/20060112/poker-academy-texas-holdem-brings-artificial-intelligence-educational-tool-to-uk-and-scandinavia_3568_ofbhdr.php>.

Sycara, K. “Machine Learning for Intelligent Support of Conflict Resolution.” Decision Support Systems. 121-136. 1 June 2006 <http://www.cs.cmu.edu/~softagents/learning.html>.

Testing Artificial Intelligence on German Soccer Fields. 1 June 2006 <http://www.dw-world.de/dw/article/0,2144,2020365,00.html>.

Zeng, Dajun, and Katia Sycara. Benefits of Learning in Negotiation. Ms. 1 June 2006 <http://www.cs.cmu.edu/~softagents/papers/aaai97-final.pdf>.