The Intellectual Standard

Volume 1 | Issue 1 Article 5

1-2012

The Learning Process: Inspiration for Computer Scientists

Nick A. Nichols
Illinois Wesleyan University, nnichols@iwu.edu

October 2011

The Learning Process: Inspiration for Computer Scientists

The Learning Process: Inspiration for Computer Scientists

Nick Nichols

The human brain is capable of tasks with which the most complex computers have trouble. Comprehending language, a task that children can begin to develop by their second or third birthday, has only just reached a consistently usable level. Even with years of research, IBM's Watson, which competed against two all-time *Jeopardy!* contestants, was unable to detect many of the nuances of human communication. Puns, context clues, and more are critical parts of communication, but given several of Watson's "confused" answers, they were clearly misinterpreted.[1] From this, an important question becomes apparent, "What makes a human able to understand this type of question (or problem), and how do we make computers capable of doing the same?" This question is not only applicable to language recognition, but also chess, packing a bag, finding an efficient route, and many other daily problems. These are all skills that we learn, practice, and than improve upon. This general concept can be extended to computers for computationally difficult and expensive problems like these. Teaching computers to learn and to apply previous results, like humans do, can be an effective model for one of their main functions, problem solving. This is the foundation of the field now known as Machine learning.

To begin, we will need a formal definition of learning. Let learning be any process by which events that are observed modify related future behaviors or decisions in some measurable and positive way. For example, we could say that we learned to juggle if we are able to, at the end of a juggling class, juggle a longer period of time than we could at the start. This may not be the only way to learn how to juggle either. Practicing may also have the same effect, as there are many ways in which humans can learn.

Observational learning is one method of which humans are capable. Albert Bandura, a famous psychologist who currently teaches at Stanford, gave four critical conditions for learning, or modeling, behavior. They are as follows: The behavior must be visible, the learner must be capable of remembering the behavior, the learner must have the ability to recreate the

1 It should be noted that Watson was victorious in this match, and a match against several Congressmen.

behavior, and the learner must have an opportunity and desire to recreate the behavior. These conditions are mostly trivial in the case of computers. The problems that we are interested in can be encoded into a computer-readable form, computers are clearly able to store data, computers can return their solutions to us, and motivation is irrelevant since programmers, operating systems, and end-users determine when and if an application is run.[2]

Observational behavior is simple: one agent performs an action, and based upon the outcome the observer chooses to replicate the action. For humans, we can consider a child watching his parent use one of their toys. If a plush cow pillow gets squeezed, it might make a moo. If the child finds that funny, he will squeeze the cow pillow after observing his parent. Likewise, in teaching a computer to play blackjack, it could "watch" a few hands. Based upon the outcome of the hand, it could choose to emulate or avoid similar actions and behaviors. By presenting the computer with both "good" and "bad"[3] decisions, it will develop a strategy that accounts for both.

Given a large sample of strategies, the computer has a high probability of generating an effectinve one; however, there can be several setbacks with observational learning. If our blackjack program has never seen a split before, or none that have been used successfully, it will probably choose to hold with a pair of eights due to the probability of selecting a card that will cause the hand to bust. If the dealer is holding a pair of nines, we will lose this hand. In this case, splitting would have probably been a favorable choice, but our examples never showed this to be a good decision. When we begin to consider the number of moves that a blackjack player may find useful, it becomes clear that our sample space must be very large to be useful. The downside to this approach, then, is that generating cases can be very time-consuming. Thus a look at possible alternative and complimentary models is warranted.

Another core learning process that humans engage in is operant

2 Several philosophical questions may arise from the statement when Artificial Intelligences (AI) are considered; however, since no known AI is fully self-aware, this question may be delayed.

3 "Good" and "Bad" are used throughout with regards to positive and negative outcomes, i.e. a "good" strategy is one that leads positive outcomes more fre quently than a "bad" strategy.

conditioning. B.F. Skinner related various actions to rewards, punishments, or nothing. A reward, and/or the removal of a negative factor, would be associated with behavior that should be repeated. Likewise, a punishment, and/or the removal of a positive factor, would accompany behavior that was being conditioned against. We can easily see this behavior throughout the entire lifespan of a human. From spankings to jail sentences, and lollipops to bonuses, these are powerful tools in the shaping of human behavior. This insight gives us another tool for programming solutions in this manner.

Where our blackjack program observed actions and behaviors before, we now actively try solutions. To continue our example, we simply let our untrained program start playing. It keeps track of its moves for each hand, and based upon the outcome we can mark the moves as more or less favorable. Once it has achieved a certain success rate, or confidence, with a strategy, it is free to explore other scenarios. Thus, the program will attempt to learn more about the weaker aspects of its game-play rather than learning only what it can from an available sample set.

Like observational learning, this solution is not perfect. It is very possible that a program's cases don't cover many "real" scenarios. After finding a strategy that is successful eighty percent of the time, it may explore other options; however, this strategy may only be useful in very narrow circumstances, or in a game like chess it may be devastatingly countered. Thus a somewhat useful strategy may be favored to a vastly superior one, because the program hasn't created a scenario in which its solution is poor.

Since humans don't strictly learn via one method or another, we can extend this philosophy to our program. Let it explore solutions and show it various examples of good and bad play. While formulating a winning strategy, the observed behavior can deter our program from poorer solutions by demonstrating strategies and moves that would effectively counter its current solution. This gives us a much more powerful means of exploring many different strategies efficiently. Naturally, this method, while effective, is also imperfect. The total number of moves that are legal in a game of chess, for example, is massive, and even modern computers cannot account for all of them. In language, it is possible for a single sentence to have several meanings. Inside jokes, sarcasm, and words with

multiple definitions change the meaning of a sentence subtly but all of the options may appear equally likely to both man and machine.

We can compound more of the learning and problem-solving strategies that human's employ, but that will only boost our confidence in an answer. In many cases, the absolute best solution may be undefined, taking years to compute, with the potential to change with time. While many strategies and solutions exist for problems of this nature, machine learning provides powerful tools to the programmers working with them.