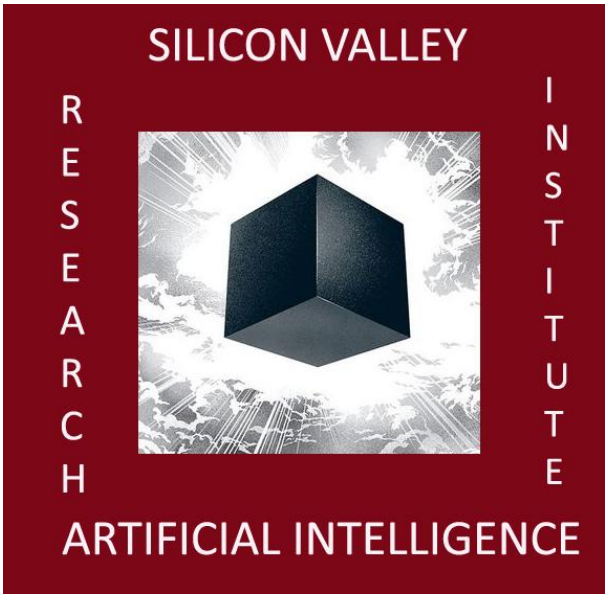


# Reinforcement Learning - Part 1



In 1946 Adriaan de Groot, a Dutch psychologist who was also a chess master, had showed in his analysis "Thought and Choice in Chess" that expert chess players do not search any more alternative moves than novice players: the experts are good at "guessing" the best next move, not necessarily at calculating it. Nevertheless, programs designed to play games kept using "deep" searches. Goro Hasegawa's board game Othello (a sensation in Japan since its introduction in 1973) was a particular favorite of the A.I. community. In 1980 Peter Frey at Northwestern University organized the first "human versus machine" Othello tournament that featured the then world-champion Hiroshi Inouie. Paul Rosenbloom's Iago (1982) at Carnegie Mellon University, which was later (1986) evolved by Sanjoy Mahajan and Kai-fu Lee into Bill, a program that beat the top US master, as well as Michael Buro's Logistello (1997) at NEC Research Institute, that defeated the then world-champion Takeshi Murakami, played Othello like masters by employing "deep" searches into the space of all possible moves. Trivia: Kai-fu Lee's 1988 dissertation would be Sphinx, the first speaker-independent speech-recognition system, and Kai-fu Lee would later become a famous Chinese venture capitalist.

There was another way for algorithms to learn how to play games.

In 2016 reinforcement learning became popular thanks to Google DeepMind and their AlphaGo go-playing program, developed by Aja Huang's team (a former student of Coulom). Reinforcement learning is a very old idea. We've always known that it works. Unfortunately it works for learning only one thing (in AlphaGo's case playing weichi). Once the machine has learned to do that one thing it is terribly difficult to train it for doing something else. It is also a bit silly because the time required for the machine to learn is so colossal that reinforcement learning has rarely been applied to mission-critical applications. Playing a game is the kind of application in which reinforcement learning makes sense: let the machine play against itself and

after a few months it will become a champion. While you can use it wherever you like, reinforcement learning really makes sense only in cases in which the rules will never change and you can wait a few months. Learning how to cross a street, for example, may not be a suitable application because the conditions change all the times and the learning machine is playing against cars. It would have to die thousands of times before it learns to cross the street. It is possible but one wonders if it's not easier to simply write a procedure on how to cross a street: the learning time is a few milliseconds and the machine doesn't have to die even once.

Neural networks can be supervised or unsupervised. Supervised networks learn from a dataset of positive instances, and the typical application has been object recognition in computer vision. Unsupervised networks cluster objects together by similarity, simulating the formation of concepts. Reinforcement learning is neither: it improves performance by direct interaction with the environment. Reinforcement learning tasks can be described mathematically as Markov decision processes. Reinforcement learning is about maximizing a value function and/or a policy.

Reinforcement learning works with a combination of positive and negative rewards, i.e. reward and punishment, just like you teach children ("trial-and-error"). The "value function" assigns rewards and punishments to various states of the system. The "policy" function determines which "next move" is more likely to get the maximum reward from the value function. The "Q-function" is the combination of the value and policy functions. For mathematicians, "reinforcement" denotes an objective function that has to be maximized. Some algorithms modify the policy directly and are called "actor-only", whereas other algorithms work on the value function and are called "critic-only". The role of the neural network, in general, is to approximate the policy function when it is too complex.

Reinforcement learning was invented even before the field was called "Artificial Intelligence". Its roots harken back to the beginning of neuroscience and to the era of behaviorism in psychology. It was already described in 1911 by Edward Thorndike of Columbia University: "The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond." Conditioning was described in 1926 by Ivan Pavlov in Russia: “the magnitude and timing of the conditioned response changes as a result of the contingency between the conditioned stimulus and the unconditioned stimulus”. In 1938 Burrhus Skinner, while at the University of Minnesota, coined the expression "operant conditioning" for the learning process in both humans and animals, both acting in order to obtain rewards and avoid punishments. Donal Hebb in his book "The Organization of Behavior" (1949), written while he was at the Yerkes National Primate Research Center in Atlanta (1942-46), formulated a simple rule (now known as "Hebbian learning") for what happens between neurons in the brain: the bond between neurons that fire together gets stronger, the bond between neurons that don't fire together gets weaker; i.e. simultaneous firing of afferent and efferent elements. (Hebb, influenced by Pavlovian conditioning, had already stated a similar learning rule in his master thesis "Conditioned and Unconditioned Reflexes and Inhibition", submitted to McGill University in Canada in 1932). In 1949 Claude Shannon suggested to use an evaluation function to help a computer learn how to play chess ("Programming a Computer for Playing Chess", 1949). Reinforcement learning was the topic of Marvin Minsky's PhD dissertation in 1954 at Princeton University ("Neural Nets and the Brain Model Problem"), a student of mathematician Albert Tucker of the "prisoner's dilemma" fame. Reinforcement learning was first used in 1959 by Samuel's checkers-playing

program. The program learned quite well from excellent players, but playing against bad players caused its performance to decline, so eventually Samuel allowed it to play only against champions. In 1961 British wartime code-breaker, Alan Turing cohort and molecular biologist Donald Michie at the University of Edinburgh built a device (made of matchboxes!) to play Tic-Tac-Toe called MENACE (Matchbox Educable Noughts and Crosses Engine) that learned how to improve its performance. In 1976 John Holland at the University of Michigan introduced classifier systems, which are reinforcement-learning systems with a credit-assignment algorithm inspired by Samuel's checkers player ("Adaptation", 1976).

Bernard Widrow modified his (supervised) Adaline algorithm to produce a reinforcement learning rule ("Punish/Reward - Learning with a Critic in Adaptive Threshold Systems", 1973), while Ian Witten at the University of Essex in Britain was working on a reinforcement-learning system capable of long-term strategizing ("Human Operators and Automatic Adaptive Controllers", 1973). In 1977 Witten implemented the first actor-critic method ("An Adaptive Optimal Controller for Discrete-time Markov Environments", 1977), although the term was introduced only later by Andrew Barto ("Neuronlike Elements That Can Solve Difficult Learning Control Problems", 1983). This was an important step in reinforcement learning because the same model emerged from experiments on animal learning and from the study of the brain (specifically, the basal ganglia).

Reinforcement learning was resurrected in 1978 by Andrew Barto's student Richard Sutton at the University of Massachusetts ("Single Channel Theory", 1978). They applied ideas published by the mathematician Harry Klopf at the Air Force research laboratories in Boston in his 40-page report "Brain Function and Adaptive Systems" (1972): the neuron is a goal-directed agent and an hedonist one; neurons actively seek "excitatory" signal and avoid "inhibitory" signals. Sutton expanded Klopf's idea into the modern "temporal-difference method" ("Toward a Modern Theory of Adaptive Networks", 1981). Temporal-difference methods had already been used in Arthur Samuel's checker player of 1959 and in Ian Witten's actor-critic architecture of 1977, and would be reenacted in John Holland's "bucket brigade" algorithm for his new classifier systems ("Properties of the Bucket Brigade", 1985). These algorithms assign credit based on the difference between temporally successive predictions.

Neuroscience was reaching similar conclusions as the temporal-difference method, for example Eric Kandel's group at Columbia University, notably Robert Hawkins, was studying the cellular mechanisms for habituation in higher invertebrates ("Is There a Cell-Biological Alphabet for Simple Forms of Learning?", 1984). Trivia: an Austrian influenced by Sigmund Freud's psychoanalysis, Kandel spent his life researching how memory is implemented in the cortex of the brain, and, in particular, he was the one who proved experimentally Hebbian learning when studying sea slugs ("Behavioral Biology of Aplysia", 1979). He also published one of the earliest textbooks of neuroscience, "Principles of Neural Science" (1981).

Sutton already employed differential Hebbian learning rules in 1981, but these rules were best outlined by Bart Kosko at Verac in San Diego ("Differential Hebbian Learning", 1986) and by Klopf himself ("A Drive-reinforcement Model of Single Neuron Function", 1986).

Ronald Williams was Rumelhart's and Hinton's coauthor in the seminal paper on backpropagation and at the time was already working on reinforcement learning ("Reinforcement Learning in Connectionist Networks", 1986). After moving at Northeastern University, he developed the REINFORCE algorithms (which, believe it or not, stands for "REward Increment = Non-negative Factor times Offset Reinforcement times Characteristic Eligibility"), a class of reinforcement learning algorithms for neural networks with stochastic neurons whose main virtue is that they are simple to implement ("Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning", 1992).

REINFORCE's "likelihood-ratio method" started the vogue for "policy gradient methods" that greatly improved reinforcement learning: policy gradient methods optimize policies by gradient descent methods. They were employed, for example, in skill learning by Barto's student Vijaykumar Gullapalli at the University of Massachusetts ("Acquiring Robot Skills via Reinforcement Learning", 1994). Later Sham Kakade at University College London ("A Natural Policy Gradient", 2002) applied Shunichi Amari's natural policy gradient to reinforcement learning, an idea improved for high-dimensional movement by Jan Peters and Stefan Schaal at the University of Southern California ("Reinforcement Learning for Humanoid Robotics", 2003). Trivia: Amari was one of the pioneers of the stochastic gradient method ("Theory of Adaptive Pattern Classifiers", 1967) and founder of the field of information geometry ("Differential-geometrical Methods in Statistics", 1985).

All the studies on reinforcement learning since Michie's MENACE converged together in the Q-learning algorithm invented in 1989 at Cambridge University by Christopher Watkins, a temporal-difference method that simultaneously optimizes value function and policy. Watkins basically discovered the similarities between reinforcement learning and the theory of optimal control that had been popular in the 1950s thanks to the work of Lev Pontryagin in Russia (the "maximum principle" of 1956) and Richard Bellman at RAND Corporation (the "Bellman equation" of 1957). Trivia: Bellman is the one who coined the expression "the curse of dimensionality" that came to haunt the field of neural networks.

Unfortunately, the two main temporal-difference methods, i.e. Sutton's adaptive heuristic critic algorithm and Watkins' Q-learning algorithms, are both very slow. Then, a few years later, Long-ji Lin at Carnegie Mellon University proposed a way to speed up both by letting the learning agent remember and rehash its past experiences, specifically by replaying to the learning algorithm the sequence of past experiences backwards, a method that came to be known as “experience replay” ("Self-improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching", 1992).

The first success stories of Reinforcement Learning were in robotics because robots need to discover how to behave in the environment through trial-and-error interactions. For example: the Obelix robot built by Jonathan Connell and Sridhar Mahadevan at IBM in 1991; the Sarcos humanoid built in 1996 by Stefan Schaal when he was at ATR in Japan; and the autonomous helicopter built by Andrew Bagnell and Jeff Schneider at Carnegie Mellon University in 2001.

The first moderate success of reinforcement learning outside robotics came in 1992 when Gerald Tesauro at IBM unveiled a neural network, TD-Gammon, that learned to play better and

better at the board game of backgammon ("Programming Backgammon Using Self-teaching Neural Nets", 1992). Gerald Tesauro had already taught a neural network to play backgammon in 1987 when he was at the University of Illinois, in collaboration with Terry Sejnowski of John Hopkins University, but that used plain supervised learning (against a dataset of 3202 ranked board positions). In 1994 Sejnowski used TD-learning to train a network to play weichi/go and in 1995 Sebastian Thrun at the University of Bonn in Germany used it to train a network to play chess. In 1994 the program Chinook, designed by Jonathan Schaeffer at the University of Alberta in Canada, won the world championship of checkers (13 years later Schaeffer would prove mathematically that Chinook is impossible to beat), but it used old-fashioned heuristics (i.e., logic thinking). Trivia: the first program to beat a world champion of backgammon was Hans Berliner's BKG 9.8 in 1979, running on a PDP-10 at Carnegie Mellon University and connected by satellite to the robot Gammonoid in Monte Carlo.

For the record, a simpler version of Q-learning is State-Action-Reward-State-Action (SARSA), the subject of Gavin Rummery's dissertation at Cambridge University ("Online Q-learning Using Connectionist Systems", 1994).

Meanwhile in neuroscience "reinforcement" came to denote the effect of the neurotransmitter dopamine in the basal ganglia of the brain, notably in the work of Wolfram Schultz at the University of Fribourg in Switzerland ("Reward-related Signals Carried by Dopamine Neurons", 1995), of James Houk at Northwestern University ("A model of how the Basal Ganglia Generates and Uses Neural Signals that Predict Reinforcement", 1995), and of Kenji Doya at the Okinawa Institute of Science and Technology in Japan ("Temporal Difference Learning in Continuous Time and Space", 1996).

In 1998 Saso Dzeroski in Slovenia and Luc De Raedt in Belgium wed Q-learning with inductive logic programming and obtained relational reinforcement learning, which represents the Q-function of Watkins' Q-learning as a first-order regression tree ("Relational Reinforcement Learning", 1998)

At around this time history records the first major successes of evolutionary algorithms. For example, David Fogel and Kumar Chellapilla in San Diego developed Blondie24, an evolutionary algorithm to optimize a neural network: a population of programs that played each other in checkers and evolved keeping the best performing ones.

Despite all the progress, reinforcement learning suffers from serious computational problems. For example, the "credit assignment problem": a robot trying to achieve a goal has to make many moves whose reward is zero, and therefore it would not have the motivation to continue.