machine intelligence to date.

Seasons of hope and despair

In the summer of 1956 at Dartmouth College, ten scientists sharing an interest in neural nets, automata theory, and the study of intelligence convened for a six-week workshop. This Dartmouth Summer Project is often regarded as the cockcrow of artificial intelligence as a field of research. Many of the participants would later be recognized as founding figures. The optimistic outlook among the delegates is reflected in the proposal submitted to the Rockefeller Foundation, which provided funding for the event:

> We propose that a 2 month, 10 man study of artificial intelligence be carried out.… The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines that use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

In the six decades since this brash beginning, the field of artificial intelligence has been through periods of hype and high expectations alternating with periods of setback and disappointment.

The first period of excitement, which began with the Dartmouth meeting, was later described by John McCarthy (the event's main organizer) as the "Look, Ma, no hands!" era. During these early days, researchers built systems designed to refute claims of the form "No machine could ever do *X*!" Such skeptical claims were common at the time. To counter them, the AI researchers created small systems that achieved *X* in a "microworld" (a well-defined, limited domain that enabled a pared-down version of the performance to be demonstrated), thus providing a proof of concept and showing that *X* could, in principle, be done by machine. One such early system, the Logic Theorist, was able to prove most of the theorems in the second chapter of Whitehead and Russell's *Principia Mathematica*, and even came up with one proof that was much more elegant than the original, thereby debunking the notion that machines could "only think numerically" and showing that machines were also able to do deduction and to invent logical proofs.[13] A follow-up program, the General Problem Solver, could in principle solve a wide range of formally specified problems.[14] Programs that could solve calculus problems typical of first-year college courses, visual analogy problems of the type that appear in some IQ tests, and simple verbal algebra problems were also written.[15] The Shakey robot (so named because of its tendency to tremble during operation) demonstrated how logical reasoning could be integrated with perception and used to plan and control physical activity.[16] The ELIZA program showed how a computer could impersonate a Rogerian psychotherapist.[17] In the mid-seventies, the program SHRDLU showed how a simulated robotic arm in a simulated world of geometric blocks could follow instructions and answer questions in English that were typed in by a user.[18] In later decades, systems would be created that demonstrated that machines could compose music in the style of various classical composers, outperform junior doctors in certain clinical diagnostic tasks, drive cars autonomously, and make patentable inventions.[19] There has even been an AI that cracked original jokes.[20] (Not that its level of humor was high—"What do you get when you cross an *optic* with a *mental object*? An *eye*-dea"—but children reportedly found its puns consistently entertaining.)

The methods that produced successes in the early demonstration systems often proved difficult to extend to a wider variety of problems or to harder problem instances. One reason for this is the "combinatorial explosion" of possibilities that must be explored by methods that rely on something like exhaustive search. Such methods work well for simple instances of a problem, but fail when things get a bit more complicated. For instance, to prove a theorem that has a 5-line long proof in a deduction system with one inference rule and 5 axioms, one could simply enumerate the 3,125 possible combinations and check each one to see if it delivers the intended conclusion. Exhaustive search would also work for 6- and 7-line proofs. But as the task becomes more difficult, the method of exhaustive search soon runs into trouble. Proving a theorem with a 50-line proof does not take ten times longer than proving a theorem that has a 5-line proof: rather, if one uses exhaustive search, it requires combing through $5^{50} \approx 8.9 \times 10^{34}$ possible sequences—which is computationally infeasible even with the fastest supercomputers.

To overcome the combinatorial explosion, one needs algorithms that exploit structure in the target domain and take advantage of prior knowledge by using heuristic search, planning, and flexible abstract representations—capabilities that were poorly developed in the early AI systems. The performance of these early systems also suffered because of poor methods for handling uncertainty, reliance on brittle and ungrounded symbolic representations, data scarcity, and severe hardware limitations on memory capacity and processor speed. By the mid-1970s, there was a growing awareness of these problems. The realization that many AI projects could never make good on their initial promises led to the onset of the first "AI winter": a period of retrenchment, during which funding decreased and skepticism increased, and AI fell out of fashion.

A new springtime arrived in the early 1980s, when Japan launched its Fifth-Generation Computer Systems Project, a well-funded public–private partnership that aimed to leapfrog the state of the art by developing a massively parallel computing architecture that would serve as a platform for artificial intelligence. This occurred at peak fascination with the Japanese "post-war economic miracle," a period when Western government and business leaders anxiously sought to divine the formula behind Japan's economic success in hope of replicating the magic at home. When Japan decided to invest big in AI, several other countries followed suit.

The ensuing years saw a great proliferation of *expert systems*. Designed as support tools for decision makers, expert systems were rule-based programs that made simple inferences from a knowledge base of facts, which had been elicited from human domain experts and painstakingly hand-coded in a formal language. Hundreds of these expert systems were built. However, the smaller systems provided little benefit, and the larger ones proved expensive to develop, validate, and keep updated, and were generally cumbersome to use. It was impractical to acquire a standalone computer just for the sake of running one program. By the late 1980s, this growth season, too, had run its course.

The Fifth-Generation Project failed to meet its objectives, as did its counterparts in the United States and Europe. A second AI winter descended. At this point, a critic could justifiably bemoan "the history of artificial intelligence research to date, consisting always of very limited success in particular areas, followed immediately by failure to reach the broader goals at which these initial successes seem at first to hint."[21] Private investors began to shun any venture carrying the brand of "artificial intelligence." Even among academics and their funders, "AI" became an unwanted epithet.[22]

Technical work continued apace, however, and by the 1990s, the second AI winter gradually thawed. Optimism was rekindled by the introduction of new techniques, which seemed to offer alternatives to the traditional logicist paradigm (often referred to as "Good Old-Fashioned Artificial Intelligence," or "GOFAI" for short), which had focused on high-level symbol manipulation and which had reached its apogee in the expert systems of the 1980s. The newly popular techniques, which included neural networks and genetic algorithms, promised to overcome some of the shortcomings of the GOFAI approach, in particular the "brittleness" that characterized classical AI programs (which typically produced complete nonsense if the programmers made even a single slightly erroneous assumption). The new techniques boasted a more organic performance. For example, neural networks exhibited the property of "graceful degradation": a small amount of damage to a neural network typically resulted in a small degradation of its performance, rather than a total crash. Even more importantly, neural networks could learn from experience, finding natural ways of generalizing from examples and finding hidden statistical patterns in their input.[23] This made the nets good at pattern recognition and classification problems. For example, by training a neural network on a data set of sonar signals, it could be taught to distinguish the acoustic profiles of submarines, mines, and sea life with better accuracy than human experts—and this could be done without anybody first having to figure out in advance exactly how the categories were to be defined or how different features were to be weighted.

While simple neural network models had been known since the late 1950s, the field enjoyed a renaissance after the introduction of the backpropagation algorithm, which made it possible to train multi-layered neural networks.[24] Such multilayered networks, which have one or more intermediary ("hidden") layers of neurons between the input and output layers, can learn a much wider range of functions than their simpler predecessors.[25] Combined with the increasingly powerful computers that were becoming available, these algorithmic improvements enabled engineers to build neural networks that were good enough to be practically useful in many applications.

The brain-like qualities of neural networks contrasted favorably with the rigidly logic-chopping but brittle performance of traditional rule-based GOFAI systems—enough so to inspire a new "-ism," *connectionism*, which emphasized the importance of massively parallel sub-symbolic processing. More than 150,000 academic papers have since been published on artificial neural networks, and they continue to be an important approach in machine learning.

Evolution-based methods, such as genetic algorithms and genetic programming, constitute another approach whose emergence helped end the second AI winter. It made perhaps a smaller academic impact than neural nets but was widely popularized. In evolutionary models, a population of candidate solutions (which can be data structures or programs) is maintained, and new candidate solutions are generated randomly by mutating or recombining variants in the existing population. Periodically, the population is pruned by applying a selection criterion (a fitness function) that allows only the better candidates to survive into the next generation. Iterated over thousands of generations, the average quality of the solutions in the candidate pool gradually increases. When it works, this kind of algorithm can produce efficient solutions to a very wide range of problems—solutions that may be strikingly novel and unintuitive, often looking more like natural structures than anything that a human engineer would design. And in principle, this can happen without much need for human input beyond the initial specification of the fitness function, which is often very simple. In practice, however, getting evolutionary methods to work well requires skill and ingenuity, particularly in devising a good representational format. Without an efficient way to encode candidate solutions (a genetic language that matches latent structure in the target domain), evolutionary search tends to meander endlessly in a vast search space or get stuck at a local optimum. Even if a good representational format is found, evolution is computationally demanding and is often defeated by the combinatorial explosion.

Neural networks and genetic algorithms are examples of methods that stimulated excitement in the 1990s by appearing to offer alternatives to the stagnating GOFAI paradigm. But the intention here is not to sing the praises of these two methods or to elevate them above the many other techniques in machine learning. In fact, one of the major theoretical developments of the past twenty years has been a clearer realization of how superficially disparate techniques can be understood as special cases within a common mathematical framework. For example, many types of artificial neural network can be viewed as classifiers that perform a particular kind of statistical calculation (maximum likelihood estimation).[26] This perspective allows neural nets to be compared with a larger class of algorithms for learning classifiers from examples—"decision trees," "logistic regression models," "support vector machines," "naive Bayes," "*k*-nearest-neighbors regression," among others.[27] In a similar manner, genetic algorithms can be viewed as performing stochastic hill-climbing, which is again a subset of a wider class of algorithms for optimization. Each of these algorithms for building classifiers or for searching a solution space has its own profile of strengths and weaknesses which can be studied mathematically. Algorithms differ in their processor time and memory space requirements, which inductive biases they presuppose, the ease with which externally produced content can be incorporated, and how transparent their inner workings are to a human analyst.

Behind the razzle-dazzle of machine learning and creative problem-solving thus lies a set of mathematically well-specified tradeoffs. The ideal is that of the perfect Bayesian agent, one that makes probabilistically optimal use of available information. This ideal is unattainable because it is too computationally demanding to be implemented in any physical computer (see Box 1). Accordingly, one can view artificial intelligence as a quest to find shortcuts: ways of tractably approximating the Bayesian ideal by sacrificing some optimality or generality while preserving enough to get high performance in the actual domains of interest.

A reflection of this picture can be seen in the work done over the past couple of decades on probabilistic graphical models, such as Bayesian networks. Bayesian networks provide a concise way of representing probabilistic and conditional independence relations that hold in some particular domain. (Exploiting such independence relations is essential for overcoming the combinatorial explosion, which is as much of a problem for probabilistic inference as it is for logical deduction.) They also provide important insight into the concept of causality.[28]

One advantage of relating learning problems from specific domains to the general problem of Bayesian inference is that new algorithms that make Bayesian inference more efficient will then yield immediate improvements across many different areas. Advances in Monte Carlo approximation techniques, for example, are directly applied in computer vision, robotics, and computational genetics. Another advantage is that it lets researchers from different disciplines more easily pool their findings. Graphical models and Bayesian statistics have become a shared focus of research in many fields, including machine learning, statistical physics, bioinformatics, combinatorial optimization, and communication theory.[35] A fair amount of the recent progress in machine learning has resulted from incorporating formal results originally derived in other academic fields. (Machine learning applications have also benefitted enormously from faster computers and greater availability of large data sets.)

Box 1 An optimal Bayesian agent

An ideal Bayesian agent starts out with a “prior probability distribution,” a function that assigns probabilities to each “possible world” (i.e. to each maximally specific way the world could turn out to be).[29] This prior incorporates an inductive bias such that simpler possible worlds are assigned higher probabilities. (One way to formally define the simplicity of a possible world is in terms of its “Kolmogorov complexity,” a measure based on the length of the shortest computer program that generates a complete description of the world.[30]) The prior also incorporates any background knowledge that the programmers wish to give to the agent.

As the agent receives new information from its sensors, it updates its probability distribution by conditionalizing the distribution on the new information according to Bayes’ theorem.[31] Conditionalization is the mathematical operation that sets the new probability of those worlds that are inconsistent with the information received to zero and renormalizes the probability distribution over the remaining possible worlds. The result is a “posterior probability distribution” (which the agent may use as its new prior in the next time step). As the agent makes observations, its probability mass thus gets concentrated on the shrinking set of possible worlds that remain consistent with the evidence; and among these possible worlds, simpler ones always have more probability.

Metaphorically, we can think of a probability as sand on a large sheet of paper. The paper is partitioned into areas of various sizes, each area corresponding to one possible world, with larger areas corresponding to simpler possible worlds. Imagine also a layer of sand of even thickness spread across the entire sheet: this is our prior probability distribution. Whenever an observation is made that rules out some possible worlds, we remove the sand from the corresponding areas of the paper and redistribute it evenly over the areas that remain in play. Thus, the total amount of sand on the sheet never changes, it just gets concentrated into fewer areas as observational evidence accumulates. This is a picture of learning in its purest form. (To calculate the probability of a *hypothesis*, we simply measure the amount of sand in all the areas that correspond to the possible worlds in which the hypothesis is true.)

So far, we have defined a learning rule. To get an agent, we also need a decision rule. To this end, we endow the agent with a “utility function” which assigns a number to each possible world. The number represents the desirability of that world according to the agent’s basic preferences. Now, at each time step, the agent selects the action with the highest expected utility.[32] (To find the action with the highest expected utility, the agent could list all possible actions. It could then compute the conditional probability distribution given the action—the probability distribution that would result from conditionalizing its current probability distribution on the observation that the action had just been taken. Finally, it could calculate the expected value of the action as the sum of the value of each possible world multiplied by the conditional probability of that world given the action.[33])

The learning rule and the decision rule together define an “optimality notion” for an agent. (Essentially the same optimality notion has been broadly used in artificial intelligence, epistemology, philosophy of science, economics, and statistics.[34]) In reality, it is impossible to build such an agent because it is computationally intractable to perform the requisite calculations. Any attempt to do so succumbs to a combinatorial explosion just like the one described in our discussion of GOFAI. To see why this is so, consider one tiny subset of all possible worlds: those that consist of a single computer monitor floating in an endless vacuum. The monitor has 1, 000 × 1, 000 pixels, each of which is perpetually either on or off. Even this subset of possible worlds is enormously large: the $2^{(1,000 \times 1,000)}$ possible monitor states outnumber all the computations expected ever to take place in the observable universe. Thus, we could not even enumerate all the possible worlds in this tiny subset of all possible worlds, let alone perform more elaborate computations on each of them individually.

Optimality notions can be of theoretical interest even if they are physically unrealizable. They give us a standard by which to judge heuristic approximations, and sometimes we can reason about what an optimal agent would do in some special case. We will encounter some alternative optimality notions for artificial agents in Chapter 12.

State of the art

Artificial intelligence already outperforms human intelligence in many domains. Table 1 surveys the state of game-playing computers, showing that AIs now beat human champions in a wide range of games.[36]

These achievements might not seem impressive today. But this is because our standards for what is impressive keep adapting to the advances being made. Expert chess playing, for example, was once thought to epitomize human intellection. In the view of several experts in the late fifties: "If one could devise a successful chess machine, one would seem to have penetrated to the core of human intellectual endeavor."[55] This no longer seems so. One sympathizes with John McCarthy, who lamented: "As soon as it works, no one calls it AI anymore."[56]

Table 1 ***Game-playing AI***

Checkers	Superhuman	Arthur Samuel's checkers program, originally written in 1952 and later improved (the 1955 version incorporating machine learning), becomes the first program to learn to play a game better than its creator.[37] In 1994, the program CHINOOK beats the reigning human champion, marking the first time a program wins an official world championship in a game of skill. In 2002, Jonathan Schaeffer and his team "solve" checkers, i.e. produce a program that always makes the best possible move (combining alpha-beta search with a database of 39 trillion endgame positions). Perfect play by both sides leads to a draw.[38]
Backgammon	Superhuman	1979: The backgammon program BKG by Hans Berliner defeats the world champion—the first computer program to defeat (in an exhibition match) a world champion in any game—though Berliner later attributes the win to luck with the dice rolls.[39]

		1992: The backgammon program TD-Gammon by Gerry Tesauro reaches championship-level ability, using temporal difference learning (a form of reinforcement learning) and repeated plays against itself to improve.[40] In the years since, backgammon programs have far surpassed the best human players.[41]
Traveller TCS	Superhuman in collaboration with human[42]	In both 1981 and 1982, Douglas Lenat's program Eurisko wins the US championship in Traveller TCS (a futuristic naval war game), prompting rule changes to block its unorthodox strategies.[43] Eurisko had heuristics for designing its fleet, and it also had heuristics for modifying its heuristics.
Othello	Superhuman	1997: The program Logistello wins every game in a six-game match against world champion Takeshi Murakami.[44]
Chess	Superhuman	1997: Deep Blue beats the world chess champion, Garry Kasparov. Kasparov claims to have seen glimpses of true intelligence and creativity in some of the computer's moves.[45] Since then, chess engines have continued to improve.[46]
Crosswords	Expert level	1999: The crossword-solving program Proverb outperforms the average crossword-solver.[47] 2012: The program Dr. Fill, created by Matt Ginsberg, scores in the top quartile among the otherwise human contestants in the American Crossword Puzzle Tournament. (Dr. Fill's performance is uneven. It completes perfectly the puzzle rated most difficult by humans, yet is stumped by a couple of nonstandard puzzles that involved spelling backwards or writing answers diagonally.)[48]
Scrabble	Superhuman	As of 2002, Scrabble-playing software surpasses the best human players.[49]
Bridge	Equal to the best	By 2005, contract bridge playing software reaches parity with the best human bridge players.[50]
Jeopardy!	Superhuman	2010: IBM's *Watson* defeats the two all-time-greatest human *Jeopardy!* champions, Ken Jennings and Brad Rutter.[51] *Jeopardy!* is a televised game show with trivia questions about history, literature, sports, geography, pop culture, science, and other topics. Questions are presented in the form of clues, and often involve wordplay.
Poker	Varied	Computer poker players remain slightly below the best humans for full-ring Texas hold 'em but perform at a superhuman level in some poker variants.[52]
FreeCell	Superhuman	Heuristics evolved using genetic algorithms produce a solver for the solitaire game FreeCell (which in its generalized form is NP-complete) that is able to beat high-ranking human players.[53]
		As of 2012, the Zen series of go-playing programs has reached rank 6

Go	Very strong amateur level	dan in fast games (the level of a very strong amateur player), using Monte Carlo tree search and machine learning techniques.[54] Go-playing programs have been improving at a rate of about 1 dan/year in recent years. If this rate of improvement continues, they might beat the human world champion in about a decade.

There is an important sense, however, in which chess-playing AI turned out to be a lesser triumph than many imagined it would be. It was once supposed, perhaps not unreasonably, that in order for a computer to play chess at grandmaster level, it would have to be endowed with a high degree of *general* intelligence.[57] One might have thought, for example, that great chess playing requires being able to learn abstract concepts, think cleverly about strategy, compose flexible plans, make a wide range of ingenious logical deductions, and maybe even model one's opponent's thinking. Not so. It turned out to be possible to build a perfectly fine chess engine around a special-purpose algorithm.[58] When implemented on the fast processors that became available towards the end of the twentieth century, it produces very strong play. But an AI built like that is narrow. It plays chess; it can do no other.[59]

In other domains, solutions have turned out to be *more* complicated than initially expected, and progress slower. The computer scientist Donald Knuth was struck that "AI has by now succeeded in doing essentially everything that requires 'thinking' but has failed to do most of what people and animals do 'without thinking'—that, somehow, is much harder!"[60] Analyzing visual scenes, recognizing objects, or controlling a robot's behavior as it interacts with a natural environment has proved challenging. Nevertheless, a fair amount of progress has been made and continues to be made, aided by steady improvements in hardware.

Common sense and natural language understanding have also turned out to be difficult. It is now often thought that achieving a fully human-level performance on these tasks is an "AI-complete" problem, meaning that the difficulty of solving these problems is essentially equivalent to the difficulty of building generally human-level intelligent machines.[61] In other words, if somebody *were* to succeed in creating an AI that could understand natural language as well as a human adult, they would in all likelihood also either already have succeeded in creating an AI that could do everything else that human intelligence can do, or they would be but a very short step from such a general capability.[62]

Chess-playing expertise turned out to be achievable by means of a surprisingly simple algorithm. It is tempting to speculate that other capabilities—such as general reasoning ability, or some key ability involved in programming—might likewise be achievable through some surprisingly simple algorithm. The fact that the best performance at one time is attained through a complicated mechanism does not mean that no simple mechanism could do the job as well or better. It might simply be that nobody has yet found the simpler alternative. The Ptolemaic system (with the Earth in the center, orbited by the Sun, the Moon, planets, and stars) represented the state of the art in astronomy for over a thousand years, and its predictive accuracy was improved over the centuries by progressively complicating the model: adding epicycles upon epicycles to the postulated celestial motions. Then the entire system was overthrown by the heliocentric theory of Copernicus, which was simpler and—though only after further elaboration by Kepler—more predictively accurate.[63]

Artificial intelligence methods are now used in more areas than it would make sense to review here, but mentioning a sampling of them will give an idea of the breadth of applications. Aside from the game AIs listed in Table 1, there are hearing aids with algorithms that filter out ambient noise; route-finders that display maps and offer navigation advice to drivers; recommender systems that suggest books and music albums based on a user's previous purchases and ratings; and medical decision support systems that help doctors diagnose breast cancer, recommend treatment plans, and aid in the interpretation of electrocardiograms. There are robotic pets and cleaning robots, lawn-mowing robots, rescue robots, surgical robots, and over a million industrial robots.[64] The world population of robots exceeds 10 million.[65]

Modern speech recognition, based on statistical techniques such as hidden Markov models, has become sufficiently accurate for practical use (some fragments of this book were drafted with the help of a speech recognition program). Personal digital assistants, such as Apple's Siri, respond to spoken commands and can answer simple questions and execute commands. Optical character recognition of handwritten and typewritten text is routinely used in applications such as mail sorting and digitization of old documents.[66]

Machine translation remains imperfect but is good enough for many applications. Early systems used the GOFAI approach of hand-coded grammars that had to be developed by skilled linguists from the ground up for each language. Newer systems use statistical machine learning techniques that automatically build statistical models from observed usage patterns. The machine infers the parameters for these models by analyzing bilingual corpora. This approach dispenses with linguists: the programmers building these systems need not even speak the languages they are working with.[67]

Face recognition has improved sufficiently in recent years that it is now used at automated border crossings in Europe and Australia. The US Department of State operates a face recognition system with over 75 million photographs for visa processing. Surveillance systems employ increasingly sophisticated AI and data-mining technologies to analyze voice, video, or text, large quantities of which are trawled from the world's electronic communications media and stored in giant data centers.

Theorem-proving and equation-solving are by now so well established that they are hardly regarded as AI anymore. Equation solvers are included in scientific computing programs such as Mathematica. Formal verification methods, including automated theorem provers, are routinely used by chip manufacturers to verify the behavior of circuit designs prior to production.

The US military and intelligence establishments have been leading the way to the large-scale deployment of bomb-disposing robots, surveillance and attack drones, and other unmanned vehicles. These still depend mainly on remote control by human operators, but work is underway to extend their autonomous capabilities.

Intelligent scheduling is a major area of success. The DART tool for automated logistics planning and scheduling was used in Operation Desert Storm in 1991 to such effect that DARPA (the Defense Advanced Research Projects Agency in the United States) claims that this single application more than paid back their thirty-year investment in AI.[68] Airline reservation systems use sophisticated scheduling and pricing systems. Businesses make wide use of AI techniques in inventory control systems. They also use automatic telephone reservation systems and helplines connected to speech recognition software to usher their hapless customers through labyrinths of interlocking menu options.

AI technologies underlie many Internet services. Software polices the world's email traffic, and despite continual adaptation by spammers to circumvent the countermeasures being brought against them, Bayesian spam filters have largely managed to hold the spam tide at bay. Software using AI components is responsible for automatically approving or declining credit card transactions, and continuously monitors account activity for signs of fraudulent use. Information retrieval systems also make extensive use of machine learning. The Google search engine is, arguably, the greatest AI system that has yet been built.

Now, it must be stressed that the demarcation between artificial intelligence and software in general is not sharp. Some of the applications listed above might be viewed more as generic software applications rather than AI in particular—though this brings us back to McCarthy's dictum that when something works it is no longer called AI. A more relevant distinction for our purposes is that between systems that have a narrow range of cognitive capability (whether they be called "AI" or not) and systems that have more generally applicable problem-solving capacities. Essentially all the systems currently in use are of the former type: narrow. However, many of them contain components that might also play a role in future artificial general intelligence or be of service in its development —components such as classifiers, search algorithms, planners, solvers, and representational frameworks.

One high-stakes and extremely competitive environment in which AI systems operate today is the global financial market. Automated stock-trading systems are widely used by major investing houses. While some of these are simply ways of automating the execution of particular buy or sell orders issued by a human fund manager, others pursue complicated trading strategies that adapt to changing market conditions. Analytic systems use an assortment of data-mining techniques and time series analysis to scan for patterns and trends in securities markets or to correlate historical price movements with external variables such as keywords in news tickers. Financial news providers sell newsfeeds that are specially formatted for use by such AI programs. Other systems specialize in finding arbitrage opportunities within or between markets, or in high-frequency trading that seeks to profit from minute price movements that occur over the course of milliseconds (a timescale at which communication latencies even for speed-of-light signals in optical fiber cable become significant, making it advantageous to locate computers near the exchange). Algorithmic high-frequency traders account for more than half of equity shares traded on US markets.[69] Algorithmic trading has been implicated in the 2010 Flash Crash (see Box 2).

Box 2 The 2010 Flash Crash

By the afternoon of May, 6, 2010, US equity markets were already down 4% on worries about the European debt crisis. At 2:32 p.m., a large seller (a mutual fund complex) initiated a sell algorithm to dispose of a large number of the E-Mini S&P 500 futures contracts to be sold off at a sell rate linked to a measure of minute-to-minute liquidity on the exchange. These contracts were bought by algorithmic high-frequency traders, which were programmed to quickly eliminate their temporary long positions by selling the contracts on to other traders. With demand from fundamental buyers slacking, the algorithmic traders started to sell the E-Minis primarily to other algorithmic traders, which in turn passed them on to other algorithmic traders, creating a "hot potato" effect driving up trading volume—this being interpreted by the sell algorithm as an indicator of high liquidity, prompting it to increase the rate at which it was putting E-Mini contracts on the market, pushing the downward spiral. At some point, the high-frequency traders started withdrawing from the market, drying up liquidity while prices continued to fall. At 2:45 p.m., trading on the E-Mini was halted by an automatic circuit breaker, the exchange's stop logic functionality. When trading was restarted, a