Machine Learning

Erik Forseth[1], Ed Tricker[2]

Abstract
Machine learning is more fashionable than ever for problems in data science, predictive modeling, and quantitative asset management. Developments in the field have revolutionized many aspects of modern life. Nevertheless, it is sometimes difficult to understand where real value ends and speculative hype begins. Here we attempt to demystify the topic. We provide a historical perspective on artificial intelligence and give a light, semi-technical overview of prevailing tools and techniques. We conclude with a brief discussion of the implications for investment management.

Keywords
Machine learning, AI

[1] *Quantitative Research Analyst*
[2] *Head of Research and Development*

1. Introduction

Machine Learning is the study of computer algorithms that improve automatically through experience.

– Tom Mitchell (1997)

Machine Learning (ML) has emerged as one of the most exciting areas of research across nearly every dimension of contemporary digital life. From medical diagnostics to recommendation systems—such as those employed by Amazon or Netflix—adaptive computer algorithms have radically altered the way that humans interact with data. Computer vision systems can recognize our faces, organize our photos, diagnose diseases, and even drive our cars. Natural language processing (NLP) has matured to the point of doing real-time translation of speech to nearly any major language. More broadly, these methods have revolutionized how researchers in academia, industry, and finance understand their domains and make decisions about the future.

Many experts believe that we are in the midst of an artificial intelligence (AI) renaissance. And yet, such claims have been made in the past. The field has seen multiple "hype cycles:" periods of rapid progress and frenzied excitement, followed by disappointing plateaus and cuts to funding (so-called "AI winters").

Nevertheless, with the unprecedented convergence of data, computing power, methods, and software, there is a prevailing sense that the present era is *different*. Standard modeling benchmarks are being overturned with astonishing frequency, while fantastic headway is being made on the theoretical underpinnings of the subject. Many of the developments that we now regard as being extraordinary will no doubt become deeply embedded in computer systems of all kinds, eventually being taken for granted as common and essential tools.

Despite their compelling successes, machine learning models are still at this stage just computer programs—driven by the data they are fed—which attempt to find a solution to some mathematical optimization problem. There remains a vast gulf between the space of tasks which can be formulated in this way, and the space of tasks that require, for example, *reasoning* or abstract planning. There is a fundamental divide between the capability of a computer model to map inputs to outputs, versus our own human intelligence [Chollet (2017)]. In grouping these methods under the moniker "artificial intelligence," we risk imbuing them with faculties they do not have. Doing so has lead to fears about an "AI takeover," which perhaps overestimate the capabilities of intelligent computer systems conceivable in the near future.

To give an overview of machine learning, we first establish context by reviewing some of the history of the field, beginning with very early efforts to think about designing intelligent machines. We then consider modern concepts and methods, and hope to clarify some jargon in the process. We conclude with a brief discussion of implications for investment management.

2. A Brief History of AI

Here we present a brief history of human efforts to build an AI. By necessity, we omit a vast amount of important detail, but we make an effort to capture the dominant historical arc.

Automatons and Reason

The idea of building intelligent machines can be traced back to (at least) classical antiquity, and is woven into the mythologies of the Greeks and other advanced civilizations of the period. These cultures were fascinated by the idea of *automatons*, or self-operating machines. Automatons themselves were not necessarily intelligent; they were depicted as simply following

sets of instructions[1]. But the Greeks believed that if they had faith, the automatons could be imbued with intelligence [Homer and Fagles (1999)]. For the ancients, an AI would then be a mechanical device that the gods had gifted with human reason and decision making.

Figure 1. Medeia and Talos: this modern painting depicts an automaton in Greek mythology. Automatons were some of the first entities imagined in the search for an AI.

Interest in the topic continued in the European Renaissance, with the work of Leibniz, Descartes, and others. Leibniz (1685) believed that all rational thought could be reduced to a symbolic logic (his "calculus ratiocinator"). His work suggested that if such a formalism was found, it could be ported to a complex automaton, and a truly rational agent could be built—an AI. Philosophically, this lighted the path to building an AI using the tools of mathematics and computation.

Leibniz's efforts to construct his calculus failed, but the work was picked up in the early 20th century by mathematicians such as Frege, Russell, and Whitehead. Frege introduced a formal machinery for exploring the foundations of all mathematics [Frege (1879)]. Whitehead and Russell (1910) used these new tools in their seminal work, *Principia Mathematica*, where they attempted to describe a simple set of rules from which all mathematical truths could be derived. These ambitious goals reignited the debate about whether a formal foundation for all of human reasoning could be constructed, and whether Leibniz's program to build a calculus for human reason could be completed.

Computability Theory and Turing Machines

In 1928, David Hilbert, inspired by the *Principia Mathematica*, proposed what came to be known as Hilbert's Program. Hilbert's idea was to uncover a set of axioms which could prove the consistency of more abstract branches of mathematics (e.g., topology or analysis) in terms of basic concepts, ultimately down to elementary arithmetic. Once again, the focus was mathematics, but human rationality was never far from mind. If this goal could be achieved in the limited scope of mathematical reasoning, perhaps it could be generalized to rational choice, decision theory, and general human intelligence.

However, from 1931 to 1936, Kurt Gödel, Alonzo Church, and Alan Turing all independently published results that exposed the limitations of mathematical logic and formal systems in general. It seemed clear that Leibniz's dream of a calculus ratiocinator simply could not be realized for any system powerful enough to represent arithmetic.

In order to prove their respective results, Gödel, Turing, and Church all had to come up with some notion of "effectively calculable" functions. Both Church and Turing used Gödel's work on incompleteness theorems to build a formal concept of computability; for Alonzo Church this was his lambda calculus (the precursor to the Lisp programming language), and for Turing it was his idealized, mechanical Turing Machine. Church and Turing quickly realized that their systems covered the same class of functions. They then went a step further, with the Church-Turing thesis, where they state that computable functions are those that can be computed—via a prescribed sequence of steps—by a mechanical device given arbitrary time and resources. In other words, *a function is computable if and only if it has an algorithm*. In this way, Church and Turing had introduced the notion of an *algorithm* to the calculation of computable functions, and Turing had introduced a mechanical device for carrying out these algorithms[2]. The modern computer had been born.

Computers and AI

While mathematicians were busy working out computability theory, mechanical and electrical engineers (and tinkerers) were making considerable progress on building the first Turing-complete computers[3]. In 1837, Charles Babbage and Lady Ada Lovelace released a design for the "analytical engine," which would have been the first general purpose computer[4]. Its design—the actual machine wasn't built until many years later—won Babbage the moniker of "father of the computer," and Lady Lovelace is credited with writing the first algorithm to be carried out on such a device.

[1] For example, in Greek mythology, Talos was a giant automaton made of bronze, built to protect Europa in Crete from pirates and invaders. He circled the island's shores three times daily.

[2] Turing firmly believed that the human mind was a Turing machine, and the functions underlying intelligence were computable by his device. He, and many others at the time, saw this device as the means by which to create an AI. In 1950, he even introduced a test to determine whether a mechanical device exhibited human like intelligence [Turing (1950)].

[3] "Turing-complete" simply means that the computer could calculate all the functions identified by computability theory as being computable. Most modern, widely-used programming languages are Turing-complete.

[4] Lady Lovelace famously dismissed the idea that the analytical engine could be used to build an AI; however, that didn't keep others, such as Turing himself, from considering such a possibility.

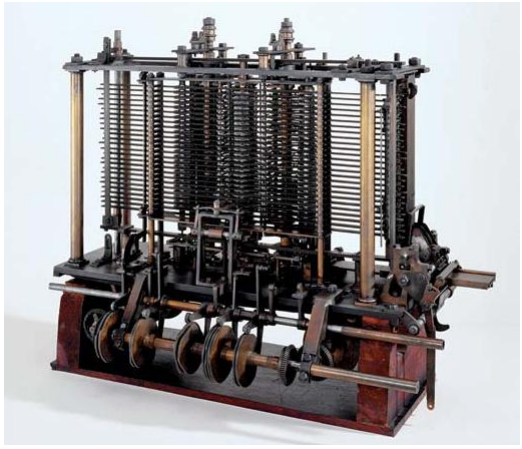


Figure 2. A model of Babbages analytical engine.

The first electronic computers were developed by the British and US militaries for use in code-breaking during WWII[5]. Turing-complete digital computers were available by the 1950s, and a large community developed around the notion that an AI could be built from these devices. Philosophers, mathematicians, psychologists, and neuroscientists all debated the possibility that human intelligence might soon be replicated.

With theory and technology coming together, computer scientist John McCarthy[6] called for a Summer Research Project on Artificial Intelligence to be held at Dartmouth College in the summer of 1956. McCarthy coined the term "Artificial Intelligence" to encompass the diverse set of disciplines comprising the burgeoning effort. The conference brought together some of the greatest minds of the age, including Marvin Minsky, Claude Shannon, Allen Newell, and Herbert Simon. The result was a perhaps misguided optimism, and a renewed determination to build "a Machine that Thinks" [Solomonoff (1956)].

The Linear Perceptron

By 1957, material progress had been made in the form of Frank Rosenblatt's *perceptron* algorithm, the first trainable classifier to be implemented via computer. Given a set of data points ostensibly belonging to two classes (say, "0" and "1"), the perceptron is a program which learns how to distinguish between the two groups by fitting a linear separating hyperplane (imagine an infinite sheet dividing the cloud of points into two distinct regions). Though simple, the perceptron proved to be an effective means for learning how to separate two classes of data, albeit in rather contrived settings. It was used, for example, to distinguish pictures of cats from dogs[7].

With this model, Rosenblatt had built the first and most basic kind of *artificial neural network* (about which we'll have more to say later on), and he (correctly) believed that it could become the foundation for work on more sophisticated computer learning systems. This generated a lot of excitement in the burgeoning field, and for the next decade research moved along at a considerable clip—due in part to significant in-flows of money from DARPA and other organizations.

Researchers made bold statements about the progress of AI. In 1958, Simon and Newell declared that "within ten years a digital computer will be the world's chess champion"; Marvin Minsky stated in 1967 that "within a generation the problem of creating an artificial intelligence will be substantially solved." Unfortunately, this exuberance was soon tempered by a series of negative results. In 1969 Papert and Minsky published a paper demonstrating the significant limitations of perceptrons. This effectively ended research into artificial neural networks.

By 1974, DARPA and the broader funding community had begun moving away from AI-based projects, weary of systematic over-promising and under-delivering. These setbacks inaugurated the first of the major AI winters.

Figure 3. Rosenblatt with the perceptron machine.

Machine Learning: A Re-branding

The 1980s and early 1990s saw more failures and setbacks, including the widespread abandonment of "expert systems," rules-based programs which were too inflexible to be of lasting value. There was a clear need by this point to redefine the scope of the field, and—for the sake of funding—a need to essentially "re-brand" the field. It was no longer fashionable to make grandiose claims about AI; researchers settled instead for progress on what came to be known as "*machine learning*," or the design of computer programs which are able to learn automatically from data.

These new methods, born largely out of statistics, optimization, and functional analysis, proved to be effective, especially as the availability of digitized data became ever greater. There were a number of important developments during this period.

[5]The British Colossus was the world's first electronic programmable digital computer, famously used by British intelligence at Bletchley Park to attack the German encryption system Enigma. It was not, however, a Turing-complete machine. A short while later, the US built the more powerful Electronic Numerical Integrator and Computer (ENIAC), which was in fact Turing-complete.

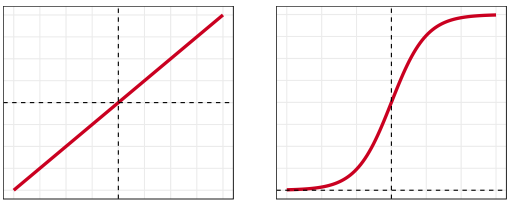
[6]Considered to be one of the fathers of AI, McCarthy was an assistant professor at Dartmouth College at the time. He went on to work as a full professor at Stanford University from 1962 until his retirement in 2000.

[7]Rosenblatt imagined that a similar machine might one day be sent to space to "take in impressions for us" [Mason et al. (1958)].

Neural Networks and Backpropagation

Neural networks—generalizations of the simple perceptron described earlier—were becoming a powerful tool in both *supervised learning* (the problem of learning to predict specific outputs given certain input data) and *unsupervised learning* (the problem of automatically extracting meaningful patterns—*features*—from data, irrespective of any relationships with associated target quantities). A neural network is merely the composition of many simple mathematical operations—or *layers*—applied to input data. Each layer is typically some kind of linear transformation followed by a (differentiable) nonlinearity (see Fig. 4). The network represents nothing more than a series of smooth geometric transformations, which, when linked together, give a very complicated transformation of the input data. (Imagine the data as a cloud of points being successively stretched, compressed, and rotated into a form that reveals some hidden underlying structure.)

The power of these objects is in their status as *universal approximators*. They are able to model extremely complex phenomena that simply cannot be represented by other means. The parameters (or *weights*) of the network must be learned by the computer, which turns out to be a nontrivial problem. A major breakthrough came with the introduction of *backpropagation* in the 1980s [Rumelhart et al. (1986), le Cun (1986)]. An application of the chain rule for derivatives in calculus, backpropagation is a procedure by which the weights of the network are iteratively "pushed" toward a desired configuration. With this new scheme, networks could be trained more reliably and much more quickly than had previously been possible. Variations of backpropagation are used to this day to train very deep networks (i.e. networks with many stacked hidden layers).



(a) linear function **(b)** nonlinear function

Figure 4. A linear function changes according to its inputs in a constant fashion. A nonlinear function, on the other hand, has a varying response.

Statistical Learning Theory and Kernel Methods

Vladimir Vapnik and others established *statistical learning theory*, laying a formal groundwork for supervised learning [Vapnik (1995)]. Particularly important here are the notions of *capacity* and *regularization*, or, respectively, the representational power of a learning algorithm vs. control over its tendency to *overfit*. Overfitting refers to the problem of a model "memorizing" the data it was trained on, but outputting meaningless results for unseen data. It is a serious and often subtle issue for any sufficiently flexible algorithm.

Vapnik and collaborators had meanwhile introduced the *support vector machine* (SVM), making vital use of what came to be known as the *kernel trick*. Kernel methods are an elegant class of flexible, nonlinear models with explicit capacity control, so named for their use of "kernel functions." Kernel functions are mathematical constructs which output a measure of similarity in an implicit, high (perhaps infinite) dimensional feature space, and the kernel trick is a prescription for translating an intractable problem in the original domain to a simple problem in a high dimensional domain (Fig. 5).

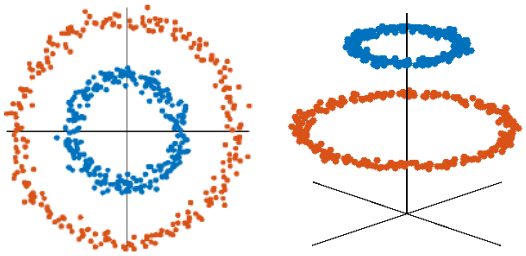


Figure 5. In two dimensions, the blue and orange data are not *linearly separable*—we cannot draw a straight line which separates one set of points from the other. Projecting the data to a higher dimension, however, reveals extra structure. Separating the points is trivial in this new space; this is the intuition behind the kernel trick.

SVMs are equivalent to a very specific kind of single-layer neural network. They became quite popular in the 1990s and 2000s thanks to a number of attractive properties. An SVM is learned by means of a *convex* optimization, meaning it admits a unique, globally optimal solution. By contrast, neural networks generically admit a combinatorially large number of "local" solutions; there is no single best set of model parameters[8]. It was thought that networks were susceptible to getting "trapped" in poor local solutions during training. Although they continued to play an important role in many applications, neural networks fell largely out of favor until the late 2000s.

Boosting, Bagging, and Ensembles

Alongside the developments happening in the neural networks and kernel communities, a set of practical and powerful results were being worked out for *ensembling* models—the practice of combining multiple learning algorithms to produce better results than those obtainable by any single model. Ensemble theory deals with the question of how best to construct a set of base learners, and then optimally combine them into a single overall model. Doing so turns out to be less straightforward than simply coming up with any arbitrary combination of constituent models.

[8]The error of a model with respect to its target values may be pictured as a hilly landscape in a high-dimensional space. The parameters of the model form a kind of coordinate system for this landscape, and the error is a measure of altitude. We wish to find the low-lying valleys of this space, which correspond to regions of small error. A "global" minimum refers to a point lower than any other, whereas a "local" minimum is only lower than points in an immediate neighborhood.

A key idea here is that of the *bias-variance-covariance tradeoff*. If one were to simply average the predictions of N sub-models, the expected error of the ensemble would depend on the average bias, variance, and covariance of the sub-models as

$$E\{\text{error}\} = \overline{\text{bias}} + \frac{1}{N}\overline{\text{var}} + \left(1 - \frac{1}{N}\right)\overline{\text{covar}}$$

where the bar indicates an average over models [Brown et al. (2005)]. The bias of each model is the expected amount by which its predictions will tend to differ from the "true" values it aims to match (the targets). Variance is essentially the opposite of the *precision* of a model, giving a measure of how "noisy" predictions are relative to the targets. Finally, the covariance term here describes the degree to which the constituent models behave similarly or differently from one another.

One of the implications is that the underlying models must have some *diversity*. Combining many diverse (but, on average, unbiased) models will have the effect of washing out some idiosyncratic variance, while hopefully reinforcing any predictive edge. (There is a deep analogy here with the relative fitness of hybrid organisms having a heterogeneous genetic makeup.)

A great many approaches have been proposed for building ensembles. *Bagging* is a procedure for fitting base models to random subsets of the data, so that each model gets a different view of the problem. *Boosting* is a more advanced form of bagging, in which learners are trained in rounds such that subsequent learners focus more heavily on training examples which were incorrectly learned at the previous stage. Boosting is less robust to very noisy data, since the ensemble may end up focusing on instances which simply cannot be reliably dealt with. Model *blending* or *stacking* is the practice of training higher-level models on the outputs of base models; this can be done recursively to obtain layered networks which are not so fundamentally different from deep neural architectures.

Ensemble methods comprise a broad class of standard techniques in statistical modeling. Nearly all winning solutions to popular machine learning competitions, such as the Netflix Prize or the well-known contests hosted by Kaggle[9], make use of ensembles of models.

The Modern Deep Learning Renaissance

Kernel methods enjoyed a period of ubiquity, and are still considered to be powerful tools for a wide array of problems. However, they are ultimately rather limited in their ability to find anything other than generic nonlinear features, which do not generalize far beyond the training data. Despite the significant achievements of the 1990s and 2000s, machine learning in practice still required careful, domain-specific feature-engineering by human experts. That is, specialists needed to spend significant time and effort in thinking about how best to transform and represent their data before feeding it to a model (such as an SVM), such that the model could best extract sought-after relationships.

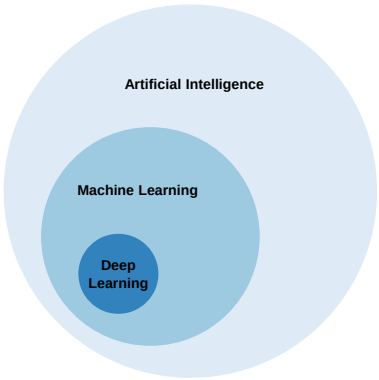


Figure 6. The distinction between topics is not always made clear in discussions about AI. "Deep learning" refers to a particular branch of machine learning, itself a subset of efforts to develop and understand AI.

Deep neural networks (DNNs)—neural networks with many stacked hidden layers—are able to compose meaningful features *automatically*. The model's hidden layers each learn some geometric transformation of the previous layer; when these layers are chained together, the model builds up hierarchical levels of abstraction, discovering fundamental and invariant features of the data [LeCun et al. (2015)].

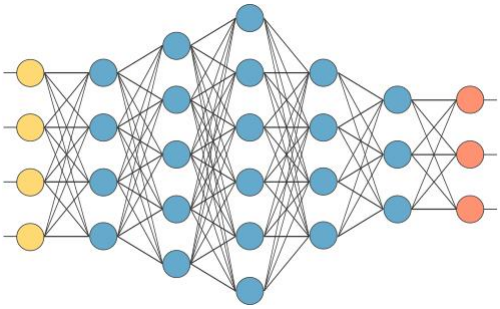


Figure 7. A neural network with five hidden layers. "Hidden layer" refers to any set of operations happening in between the input (yellow) and the final output (red).

The ability of DNNs to carry out automatic, multi-stage feature engineering had long held promise as a means for circumventing expensive human involvement in this stage of the learning process. And yet, beset by the difficulties involved in training deep networks, researchers' hopes had been tempered. This began to change after a series of seminal results starting in 2006, and continuing especially in the 2010s.

[9]The Netflix Prize challenged contestants to improve upon Netflix's own algorithm for predicting user movie ratings, offering a $1 million prize for any team which could demonstrate a significant improvement. Kaggle—a startup acquired by Google in March of 2017—regularly hosts data science contests awarding large cash prizes. Competitions range from identifying cancerous lesions in high-resolution lung scans to to predicting the outcomes of the March Madness college basketball tournament.

DNNs now come in many flavors, depending on their architecture and the context in which they are deployed. *Convolutional networks*, whose hidden layers learn discrete local filters, have met with fantastic success in problems whose data has some local context—images, speech, and text being prime examples. *Recurrent neural networks* are useful for processing sequentially ordered data[10]. *Deep reinforcement learning* leverages DNNs for the problem of an agent learning by trial-and-error interactions with its environment, as for robot locomotion[11].

A host of factors underlaid the dramatic progress. Researchers made use of specific nonlinearities in their networks, which facilitated faster and more reliable training. New and more powerful forms of model regularization were developed. Crucially, there was an explosion in available data and processing power. It turned out that many of the issues encountered in fitting these models can be avoided simply by having a sufficiently large body of training data. Meanwhile, models could be trained faster—by orders of magnitude—using computing architectures based on graphics processing units (GPUs). As for the problem of local solutions, it was realized that these usually have similar (and close to globally optimal) quality. With better architectures, more data, and faster computers, DNNs surpassed benchmark after benchmark on standard datasets in disparate fields. In most cases, in fact, DNNs made quantum leaps forward relative to prior state-of-the-art methods.

As an example, consider the evolution from IBM's Deep Blue to Google DeepMind's AlphaGo system. Deep Blue famously bested chess grandmaster Garry Kasparov in a sequence of matches from 1996-1997. Celebrated as a milestone in AI, Deep Blue was not really an *intelligent* system. It would search among possible moves and evaluate their quality using a chess-specific reward function. Its strength came mainly from its massive processing power; in 1997, Deep Blue was one of the most powerful supercomputers in the world. In March 2016, AlphaGo beat Lee Sedol in a five-game series of the ancient Chinese board game Go. Go's larger board makes the space of possible configurations too large for brute-force searches like those of Deep Blue. AlphaGo instead used convolutional neural networks to process the board as an image and evaluate the state of the game, and reinforcement learning to improve its own play [Silver et al. (2016)].

3. Machine Learning in Finance

Unsurprisingly, professionals have been eager to bring machine learning methods to bear in quantitative finance. The potential applications are plentiful. ML systems might be able to uncover subtle nonlinear effects in asset returns, where more traditional methods (e.g., linear autoregressive models) might fail. Many problems in finance boil down to time series prediction, for which one might consider using temporal convolutional networks or recurrent neural networks. The idea of *ensembling* models—combining multiple independent models to reduce predictive variance—is a natural thing to do here. In fact, ensembles are intimately related to *portfolio theory*; many interesting results for ensembles were first established in the context of portfolios of financial assets.

Even tried and true investment styles can benefit from more sophisticated approaches founded on ML. Identifying long-term trends in markets, for example—or predicting when trends will fall apart—is not a simple problem. Anticipating swings in market volatility is another challenging task that may be better attacked with newer techniques rather than, say, classical GARCH models [Rizvi et al. (2017)]. Machine learning has given us many tools for dealing with data in high dimensions and having many degrees of freedom, as is the case for long-short equity portfolios. With computer programs now able to process and interpret text in real-time, news- and event-based trading is being revolutionized. Apart from the direct applications, advances in ML also contribute indirectly to investing; dealing with spurious correlations or estimating covariance matrices are problems central to both fields, for instance.

From one perspective, deploying computerized statistical models in investment management may seem like an obvious choice. Compared with human traders, computers are passionless, tireless, and better at boring through mountains of data in order to find meaningful, trade-able market signals. For better or for worse, the reality of the situation is more complicated. Many recent breakthroughs in machine learning are due to the recent availability of vast, clean datasets. Financial time series, on the other hand, constitute relatively small datasets with very low signal-to-noise ratios. Further, they are largely nonstationary; the generating distributions for these data change over time. One can always expect to see new—and arguably *fundamentally unpredictable*—behavior in the markets. The risk of over-fitting a model to historical observations, a concern for any systematic trader, is magnified here. Finally, managers must come to terms with a certain amount of opacity when relying on a black-box trading algorithm. On the one hand, there would be little point in invoking these methods if they were only capable of finding patterns that happen to be intuitive to humans; on the other hand, one must have confidence that the model is not fitting to pure noise. Any attempt to apply these methods in a naïve, out-of-the-box fashion would be badly misguided.

With these enormous caveats in mind, there remains ample potential for disruption. In the hands of careful practitioners, machine learning provides a powerful toolset for systematic investing. The industry is in the midst of a veritable gold-rush for quantitative talent. No doubt there will be hand-wringing and soul-searching over the efficacy of these approaches. And

[10] Recurrent networks have cyclical connections between units, allowing them to model temporal behavior.

[11] *Reinforcement learning* is an older field, dating back to optimal control for dynamic systems in the 1950s and 1960s (for e.g. calculating aerospace trajectories). *Deep* reinforcement learning refers to a modern variant which uses many-layered (i.e. deep) neural networks for some portion of its calculations.

yet, progress is irreversible. Whether or not they seem novel today, some subset of the methods we have described will become canonical tools in statistical modeling and data science. They will likewise find their place in the universe of investing.

References

G. Brown, J. L. Wyatt, and P. Tiňo. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, (6):1621–1650, 2005.

F. Chollet. The limitations of deep learning. http://blog.keras.io/the-limitations-of-deep-learning.html, 2017. Accessed: 2017-08-09.

G. Frege. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Halle, 1879.

Homer and R. Fagles. *Iliad*. Cambridge, MA., Harvard University Press, 1999.

Y. le Cun. Learning processes in an asymmetric threshold network. *Disordered systems and biological organization*, pages 233–240, 1986.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, (521):436–444, 2015.

G. Leibniz. *The Art of Discovery*. Wiener 51, 1685.

H. Mason, D. Stewart, and B. Gill. Rival. *The New Yorker*, (Dec. 6, 1958), 1958.

T. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

S. A. A. Rizvi, S. J. Roberts, M. A. Osborne, and F. Nyikosa. A novel approach to forecasting financial volatility with gaussian process envelopes. *ArXiv e-prints*, 2017.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, (323): 533–536, 1986.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, (6529):484–489, 2016.

R. Solomonoff. Personal notes. 1956.

A. Turing. Computing machinery and intelligence. *Mind*, (236):433–460, 1950.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, 1995.

A. N. Whitehead and B. Russell. *Principia mathematica*. Cambridge: Cambridge University Press, 1910.

Legal Disclaimer