33

Black-Box Model Risk in Finance

Samuel N. Cohen[a], Derek Snow[b] and Lukasz Szpruch[c]

Abstract

Machine learning models are increasingly used in a wide variety of financial settings. The difficulty of understanding the inner workings of these systems, combined with their wide applicability, has the potential to lead to significant new risks for users; these risks need to be understood and quantified. In this chapter, we will focus on a well studied application of machine learning techniques, to pricing and hedging of financial options. Our aim will be to highlight the various sources of risk that the introduction of machine learning emphasises or de-emphasises, and the possible risk mitigation and management strategies that are available.

Acknowledgements.

We are grateful to Katia Babbar for helpful comments and suggestions on a draft of this chapter. We acknowledge the support of the Alan Turing Institute under EPSRC grant no. EP/N510129/1

33.1 Introduction

Traditionally, the tractability of pricing and hedging methods was arguably more critical than their accuracy, and the limits of computation determined what methods were useful. The Black–Scholes formula is concise, simple to understand, can be implemented on a handheld calculator (Lo, 2019); these features were critical to its wide adoption. Similarly, the Heston model benefits from convenient (fast) Fourier transformation methods (see, for example, Gatheral, 2006) and the SABR model from a convenient approximation (see Hagan et al., 2002; Oblój, 2008), which have formed a key part of their attractiveness. While many more sophisticated and accurate models have been developed, computational bottlenecks have impeded their wide adoption.

In recent years, machine learning models in finance have become streamlined; in just a few lines of packaged code, modellers can develop state-of-the-art models with online computing power and open-source software (Snow, 2020; Dixon et al.,

[a] University of Oxford and The Alan Turing Institute. samuel.cohen@maths.ox.ac.uk.
[b] The Alan Turing Institute. dsnow@turing.ac.uk.
[c] University of Edinburgh and The The Alan Turing Institute. l.szpruch@ed.ac.uk.
Published in *Machine Learning And Data Sciences For Financial Markets*, Agostino Capponi and Charles-Albert Lehalle

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

2020). However, the risks of blindly using machine learning solutions, without understanding their inner workings and inherent drawbacks, are significant.

In this chapter, we seek to give an overview of the key issues which arise when using machine learning in finance, and some remedies which have been suggested. Rather than concentrating on developing a particular algorithm, we take a higher-level view of the risks and challenges which arise in these contexts. We wish to highlight that machine learning is not a panacea for financial markets, instead it provides tools which allow practitioners to shift between different sources of risk, some of which have not been a primary concern in the past.

We will focus on those risks which are a core part of machine learning – the risks inherent in data and in the modelling algorithms used. We will not discuss what The World Economic Forum calls the erosion of "human financial talent" where humans lose the skill to challenge and disagree with machine learning systems (McWaters et al., 2019), although this is potentially a significant concern in many financial applications.

There are two broad uses of machine learning in finance. The first is to remove computational barriers and enable use of advanced models in day to day business operations. When used in this manner, 'machine learning' is providing the next generation of computational tools, which are used to speed up and improve traditional modelling. For example, when calibrating an option pricing model one often needs to price many derivatives many times, using a variety of potential parameter values – this is a task that can be improved by using a machine-learnt approximation for the pricing operator. Hutchinson et al. (1994) trained a neural network on simulated data to learn the Black–Scholes option pricing formula. A number of efficient algorithms have recently been developed to approximate parametric pricing operators with flexible modelling assumptions (for example, see Horvath et al., 2020; Jacquier and Oumgari, 2019; Ferguson and Green, 2018; McGhee, 2018; Sabate-Vidales et al., 2018, 2020). This in turn can eliminate the calibration bottlenecks commonly found in using realistic pricing models.

The second application involves a more fundamental change in the approach to modelling and working with data, where traditional, low-dimensional, hand-crafted models are replaced with abstract over-parameterized models. These models may be used to represent the statistical features of underlying assets, to determine prices, hedges and risk properties of portfolios in terms of market observables, as well as a combination of these tasks. This application of machine learning depends in a far more significant way on the historical data available, leading to various challenges: for example, it becomes hard to understand what is driving the price of a derivative, and the data modelling and preprocessing steps might introduce an additional set of risks, which can be a cause of unease for regulators and risk managers.

In this chapter, for the sake of concreteness, we focus our attention on the challenge of pricing and hedging derivatives, which we outline in §33.2, and principally on the use of one machine learning method (deep neural networks) in this challenge. In §33.3 we discuss issues connected with the sources of data that are used as inputs into machine learning algorithms, while in §33.4 we

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

are concerned with the risk associated with the way probabilistic modelling is incorporated within machine learning.

33.2 A practical application of machine learning

To get acquainted with a neural network solution, we will take a closer look at the problem of pricing and hedging an exotic derivative, by trading in a financial market. Here we give a non-technical description; for a technical primer on neural networks see, for example, the work of Bengio et al. (2017). The inputs to our problem will be a combination of historical market data and commonly accepted handcrafted models (depending on the precise approach), the latter may be used to generate additional simulated data for training purposes. The key outputs are prices, hedging strategies and risk assessments for exotic options and portfolios of exotic options and other assets.

33.2.1 How to use neural networks for derivative modelling

The precise role of machine learning in options pricing and the data used to support it can vary significantly. If we consider one particular class of machine learning methods – neural networks – in Figure 33.1 we present one way of classifying some applications of this method, looking at whether they principally are concerned with the processing and generation of data, or with building models for financial markets, and how these contribute to different outputs.

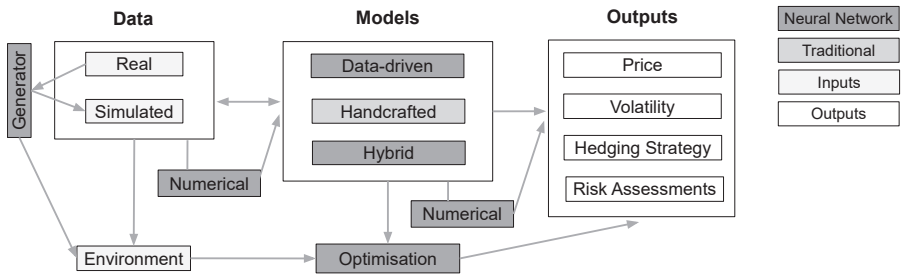

Figure 33.1 *Neural Network Pricing and Hedging*. This figure illustrates five broad roles for neural networks as part of developing pricing, volatility, and hedging models. Neural networks can learn directly from real data (Data-driven), can be used as a numerical or computational tool (Numerical), can enhance handcrafted models (Hybrid), can generate novel simulated data (Generator), and can be used in reinforcement learning models to develop strategies in a dynamic way (Optimisation).

Many of the use-cases for neural networks boil down to their ability to learn complex, high-dimensional, non-linear relationships; for example, to solve partial differential equations in high dimension, to develop data-driven models with large feature sets, and to find optimal policies in large state-spaces via reinforcement learning. The effectiveness of neural nets in high-dimensional settings suggests

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

they have ability to overcome the computational curse of dimensionality[1]. We divide the neural network use-cases into five broad modes of application: Data-driven models, Hybrid models, Numerical approximations, Online Optimisation methods, and Generator models.

(a) **Numerical approximations** are based on traditional parametric models, and exploit neural networks to, for example, approximate pricing or hedging functionals in the form of solutions to parametric families of PDEs. This approximation can significantly speed up calibration problems. In these applications, the neural network is not trained against real-world historical data, but is used purely to approximate complex functions, in an efficient way. The key difficulty in this area is the calibration of hyperparameters and network architectures, and the implementation to industrial standards.

- Some applications depend on solving a high-dimensional and/or non-linear PDE, even if underlying model is simple. For example to price and hedge path-dependent options or to compute XVAs[2]. Neural networks can be used as a function approximation tool which works well in high dimension, and are particularly efficient at solving PDEs when blended with Monte Carlo simulation (Barucci et al., 1997; Beck et al., 2021; Sabate-Vidales et al., 2020; Sirignano and Spiliopoulos, 2018; Han et al., 2018; Gnoatto et al., 2020).
- Some problems, in particular in calibration of handcrafted models (for example, Heston or SABR models), require repeated calculation of various option prices prices under a variety of parameters. By providing an efficient means of approximating this calculation for a range of parameter choices, neural networks speed up the process of calibration, allowing a more efficient use of data (Andreou et al., 2010; Bayer et al., 2019; Sabate-Vidales et al., 2018). These methods often depend on simulating option values from the handcrafted model, under a range of parameter values.

(b) **Data-driven models** rely on real market data to approximate pricing and hedging functions. These models disregard handcrafted models in their entirety and simply use historical, synthetic or simulated data of any type to learn new relationships and features (Ghaziri et al., 2000; Montesdeoca and Niranjan, 2016).

(c) **Hybrid models** rely on historical, simulated, or synthetic data to approximate pricing and hedging functions and also constrain or impose knowledge onto the architecture of an otherwise unconstrained neural network.

- Some models first leverage a handcrafted model to estimate prices and then build a data-driven model to learn the difference or residuals between

[1] It is widely conjectured that neural nets are effective in situations where the problem at hand admits an accurate low dimensional representation, however, this representation is not known a priori (Fefferman et al., 2016).

[2] This is similar to the now classic use of regression in the Longstaff–Schwarz approach to pricing American options (Longstaff and Schwartz, 2001), where neural networks can give a more flexible approximating class of functions.

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

the observed price and the handcrafted model estimate (Lajbcygier and Connor, 1997).
- Other models constrain a universal neural network by adding domain knowledge to the architecture to learn more realistic relationships that increases the interpretability or efficiency of the model; for example, forcing monotonous relationships towards one direction by adding penalties to the loss function (Garcia and Gençay, 2000; Dugas et al., 2009; Gierjatowicz et al., 2020).

(d) **Online Optimization methods**: A number of option types, for example American options, benefit from learning optimal stopping rules using neural networks in a reinforcement learning framework; others may benefit from learning a value function or a hedging strategy that benefits from temporal optimal control, e.g. a model that takes evolving market frictions into account in an environment or control system (Buehler et al., 2019; Kolm and Ritter, 2019).

(e) **Generator models** can take any data as input and generate new data that has the same statistical properties. Data can be generated by applying a calibrated 'handcrafted' model (or from a range of handcrafted models) or from a machine learning generative model. Alternatively, they may be learn from data observed in one situation to generate representative data in a related setting. The first of these uses (where the generated data matches statistical properties of historical observations) is called 'synthetic data', and is a subset of 'simulated data', which includes scenarios that were not present in the historical data. The generated data's purpose is principally to aid the performance of machine learning pipelines, for example to provide an environment to train further models with reinforcement learning. It's worth noting that the generator, and hence the simulated data, should be seen as a statistical model[3] for our observations. This approach can be viewed as model-based data boosting (Buehler et al., 2020; Ni et al., 2020; Mariani et al., 2019).

Using a combination of these approaches, we can now build an abstract pipeline for learning to price and hedge options.

(1) Using historical data and current market data, build up a collection of training trajectories of the assets under consideration, as well as a representation of the state of the market. As we typically only have one trajectory of past data, one often needs to augment historical observations with models or simulated data. We have discussed two main approaches to this:
   (i) Design and train *generative* models to provide additional realistic data, or provide a rich parametric class with which to work.
   (ii) Train *handcrafted* models (possibly using neural networks as a *numerical* tool to speed up calibration) from which to simulate.

(2) Using this data, either

[3] Here the network architecture, loss function and training method of the generator are all modeling choices.

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

(i) Use reinforcement learning to *optimize* hedging strategies (and thus determine initial prices), using the simulator as a training environment.
(ii) Learn *data-driven* pricing relationships and hedging strategies by observing prices in historical data.
(iii) Learn a *hybrid* model that first trains on simulated data, and then transfers this learning over to real data for efficient training.
(iV) Use a further *numerical* approach, to solve the PDEs arising from the (possibly high dimensional) models. Equivalently, one can ensure the calibrated model generates trajectories with probabilities from a risk-neutral measure, and use Monte Carlo simulation to estimate prices.

33.2.2 Black-box trade-offs

The pipeline we have outlined above gives a very flexible approach to modelling and pricing of financial derivatives, however this is not a 'free lunch'. Neural networks are notoriously opaque as a modelling tool, and are often implemented simply as a 'black-box' approach to function approximation (the hybrid models discussed above being a partial exception to this). An important practical question is whether the potential disadvantages of black-box hedging can be justified by increased performance, and whether the risks associated with this approach can be distinguished and quantified.

An industry standard for assessing the quality of a new model is to compare it with simpler benchmarks such as Black–Scholes delta-hedging with the presence of transaction costs (Davis et al., 1993; Whalley and Wilmott, 1999). There is preliminary evidence that suggests, at least in simple, constant volatility settings, these benchmark models have performance close to that of reinforcement learning agents (Mikkilä, 2020). If that is true, these benchmarks should be preferred because they have easy to explain analytical solutions.

Ruf and Wang (2021) have shown, on an out-of-sample test set, that a simple fixed hedging strategy that hedges calls by $0.9 \times \delta_{\mathrm{BS}}$ and puts by $1.1 \times \delta_{\mathrm{BS}}$, where δ_{BS} is the delta under the classic Black–Scholes model, outperformed 14 out of 16 models, including all the supervised neural network models with 1-day rebalancing, and outperformed all models with 2-day rebalancing. It should be noted, that these tests were performed in a simple one period setting, with no transaction costs. These results do not directly extend to a large basket of derivatives; as a result, more tests are needed. However, these results suggest that more complex black-box models may fail to outperform simpler ones.

It is also possible that improved feature selection and simple models might be a better solution than a direct application of neural networks. Ruf and Wang (2021) compared a neural network model with a linear regression model to estimate the hedge ratio, on simulated and real data. Predictors in the linear model included standard model sensitivities under the Black–Scholes model: moneyness, Delta, Vega, implied volatility, time to maturity and Vanna. They conclude that the classical option sensitivities already contain the non-linearities necessary to build an effective hedging strategy for common options, in a financially significant and

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

efficient way. They further showed that this linear regression model outperformed their neural network model.

Their approach diverges from Buehler et al. (2019), who do not use option sensitivities as variables, but instead rely on the belief that the Greeks indirectly present themselves as non-linear functions that the agent has access to via the market state, in the form of hedging instrument prices. Future experiments should test hedging performance on a basket of derivatives, in a multi-period setting, with dynamic volatility, transaction costs, and environmental feedback. Such a real-life setup could benefit the deep hedging approach (Buehler et al., 2019), as this has the capacity for direct feedback from the environment and online training.

33.3 The role of data

33.3.1 Data risks

We now focus our attention on how the use of machine learning methods highlights risks associated with the underlying financial data. One can identify three primary sources of risk: biases in the training data, erroneous data or erroneous preprocessing, and legal and regulatory data risks. We will not focus on legal and regulatory risks. As we are moving the dial from handcrafted towards data-driven models, the data risk increases significantly. On the one hand, handcrafted models are more robust to biases and errors in the data, and the risk of using inadequate models is easier to detect. On the other other hand, for data-driven models the training data becomes an integral part of the model, making them more sensitive to data risk.

Biased data

A key issue in financial data is that the majority of data is backward-looking, and there is no guarantee that future behaviour of financial markets will be represented by historical observations. We typically only have one trajectory of historical data – we cannot see what *might* have happened in different scenarios – which makes it difficult to build a clear view of the range of likely outcomes in the future. Any recent changes in the true underlying state of a system that are not incorporated in a model's training dataset will also lead to biased predictions.

These risks are not particular to machine learning – they are well known issues in financial markets; however, the use of machine learning models, which often depend on observed data in a more significant way than traditional handcrafted models, means understanding and managing these risks is critical for the success of these approaches. Here we summarize some key forms of bias that historical data could exhibit:

(a) **Backward looking**: most data reflect prices and signals obtained in the past. This means that data could reflect a state of affairs that no longer applies, e.g. an options model might only have access to data from a low volatility period, or from an old regulatory regime. Financial markets are reactive and don't follow universal laws – for example, the increase of high-frequency

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

trading has changed the nature of many financial markets (see, for example, the discussion in MacKenzie (2018)). Restricting to only the very recent past, and projecting a model's predictions only into the near future, can mitigate this concern somewhat, but often results in significantly less data being available for training.

(b) **Spurious correlations**: for some financial domains it is prudent to record and collect attributes that have some theoretical basis. For example, it is questionable whether an option pricing model should contain sentiment features. It is often difficult to identify intuitively unreasonable relationships within a black-box model, and the increase in dimensionality of the models being used results in a vast increase in the range of potential relationships that could be inferred (Fan and Zhou, 2016). Spurious correlations are well known in finance, but the increased use of machine learning techniques can exacerbate this problem.

(c) **Sample disparity**: biases in the sampling procedure might lead to data that doesn't fairly represent the state of the market. For example, a firm may wish to use the same algorithm for trading over multiple exchanges and geographic locations, each of which has subtly different conventions and data. This introduces biases within the data which can be magnified through the use of machine learning models, particularly when a model is trained in one setting then deployed for use elsewhere.

(d) **Imbalanced inputs**: some evidence shows that even when your sample accurately reflects the true state of the market, it remains imbalanced – rare events may be significant, but are only infrequently represented in historical samples. Many data-driven models are known to favour the performance of the majority outcome, to limit overall model errors (Provost, 2000). Within a financial setting, this might correspond to a model which performs well in low-volatility regimes, even when high-volatility periods are observable (infrequently) within the training data. A related idea is the use of stressed periods for calculating risks within the Basel accords – these infrequent periods are significant to overall performance, and need to be explicitly taken into account.

(e) **Insufficient data**: one often has insufficient data to use machine-learning models well. The calibration of neural networks requires significant quantities of data, which are often not available for training in a financial context (Gu et al., 2018). The richer the context in which an algorithm is to be run, and the more finely tuned its behaviour needs to be, the larger the quantity of data needed. It is worth emphasising that this is not to say that the data in finance is 'small', but that often it is not 'large' in the directions needed – we may have enormous datasets due to high-frequency observations of a large number of asset's order books, but these will be of little use in determining good models over long time periods.

Data errors and preprocessing

A further concern in many applications is that data may display subtle errors, which need to be addressed before it can be effectively used. This is a common concern in many applications of machine learning, and data-cleaning methodologies form a key part of the implementation of these methods.

(a) Observed financial data can fail to satisfy fundamental economic constraints, which can be subtle. For example, as discussed in Cohen et al. (2020), historic options price data, for both listed and OTC contracts, may be inconsistent with no-arbitrage constraints, particularly in emerging markets. If such data is naïvely used when training a trading system, it is plausible that the system would learn to exploit this apparent arbitrage opportunity. Given these errors could arise due to multiple sources (for example, stale quotes being listed as live in historical data), this can lead to significant error in the resulting learnt behaviour.

(b) When working with time-series data, it is critical to respect information flow when e.g splitting data into training, validation, and test sets; engineering features; or normalising data. Errors in this process can 'leak' information from the future, leading to unrealistic performance.

(c) Financial data often has a particular concern around precise timekeeping, which may not be reflected in the accuracy of the data given. Particularly when working with very high-frequency data, failing to take into account latency and other implementation issues can have a significant effect, which may not be well reflected or available in historic data (see, for example, the effect of latency in Cartea and Sánchez-Betancourt (2021)). This is particularly the case with the increased attention being given to non-market data sources (for example signals from online news sources), where historic time-stamping may be of low quality.

(d) Financial data is often heavy tailed and not stationary, making it difficult to detect and exclude erroneous data. Typical methods (such as Winsorizing) have the potential to introduce significant bias, particularly when considering extreme events.

33.3.2 Data solutions

There are many process improvements that can be implemented to decrease data risks, for example, performing data quality monitoring, documenting and reviewing the manipulation of input data, and educating and training individuals involved in data manipulation tasks. Another key approach, to fix biased and limited data, is to generate synthetic or simulated data which is free from (or even corrects for) these issues. We will outline two key approaches – the top-down approach of synthetic data generation, and the bottom-up approach of agent based modelling.

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

Synthetic data generation

Synthetic data generation (SDG) is a top-down data generation solution. It can help to address some of the data biases and errors listed above. It does so by augmenting the quantity and quality of historical data, but it does not attempt to provide a simulator which can model feedback effects for an agent's interventions in a market.

At a high level, a synthetic data generator attempts to build a probabilistic model which would generate observations similar to historical data. Generative models such as generative adversarial networks (GANs) and variational autoencoders (VAEs) have demonstrated great success in seemingly high dimensional setups (Wan et al., 2017; Lin et al., 2020). If used correctly, SDGs could allow for a more comprehensive approach to future-proofing and validating machine learning pipelines; ameliorating some structural deficiencies in data and amending distributional biases (Louizos et al., 2016).

In a financial context, Takahashi et al. (2019) have shown that (GANs) can be used to generate synthetic data that matches most known stylized features of returns; Ni et al. (2020) have shown how mathematically principled feature extraction methods such as signature models can be used to efficiently implement conditional GANs for generic time series data. Related ideas, but combined with VAEs, are presented in Buehler et al. (2020). Henry-Labordere (2019) developed efficient algorithms building upon optimal transport theory and highlighted an interesting application of data generators for detecting anomalies. Algorithms based on restricted Boltzmann machines have been developed in Kondratyev and Schwarz (2019), who coined the term 'market generators'. An alternative approach is to learn the underlying dynamics of the system, allowing a path to evolve through time – this is the approach taken by neural-SDE models (Gierjatowicz et al., 2020). Fu et al. (2019) have shown how conditional GANs can be used to produce synthetic data for different market scenarios. Koshiyama et al. (2020) have shown how these methods can be used to validate trading strategies.

SDGs still pose a form of modelling risk, since a generator is only as good as the data from which it constructs its generating function; building an SDG involves choosing a metric, a loss function, and a training algorithm for parameter selection. As such, SDGs introduce model risks within the data used to train downstream models, and these risks may be difficult to identify depending on the use-case.

At the present time, research in this area lacks standardised benchmarks and theoretical guarantees. Most off-the-shelf methods are not built with financial applications in mind, and are therefore likely to generate simulated data which exhibits arbitrage or other economically unrealistic phenomena. Moreover, many of these models remain black-box and are not easily interpretable.

The key benefit, however, is that these methods are expressive and work in high dimensions. This is the main difference when comparing with traditional methods using handcrafted features. Synthetic data can also be used to generate data according to expert opinions and known facts, e.g., can be conditioned to

form the observed volatility smiles. And SDGs generally offer a more accurate and robust oversampling method than traditional methods like SMOTE (Synthetic Minority Oversampling Technique) that simply repeat existing records (Chawla et al., 2002). They also provide a convenient solution for missing data imputation and outlier treatment (Xu and Veeramachaneni, 2018).

Synthetic data generation tools can be used as part of a larger solution to address some of the most common upstream data errors. They can be used side-by-side with federated learning techniques to improve the quality of single standing resources, by pooling data across, departments, subsidiaries, companies, or data-providers (Goetz and Tewari, 2020).

Deep generative models for synthetic data generation remains a new field, and although they have potential to alleviate some of the known issues of neural network models, it is clear that they have the potential to introduce further risks. Overall, as with other methods, they can be seen as shifting risks away from the quantity and quality of data, by including probabilistic modelling (with its associated risks) at a very early stage in the analysis pipeline.

Market simulator engine

Agent-based model (ABM) simulators, unlike SDGs, are a bottom-up data solution and date back to the 1990s. Notable early models include those by Levy et al. (1994) and the Santa Fe Artificial Stock Market (Palmer et al., 1994; Arthur et al., 1996). ABMs model markets as evolving systems of competing, autonomous interacting agents (LeBaron, 2000).

The development of ABMs has seen multiple waves of interest. The first wave of market simulators in the 1990s was a deliberate move away from classical economic theories to advance financial market knowledge, the second wave was a reaction to the failure of economic models in foreseeing the financial crises of 2008, the third wave was a call to understand high frequency trading and the flash-crashes in 2010 and 2013, and the fourth and current wave combines the concerns with the past, but emphasises the use of simulators to train machine learning agents.

A key advantage of ABMs is that, as bottom-up models, they attempt to learn the feedback effects of agents acting within the market. This has the advantage that these effects can be modelled, but makes training much more difficult – usually involving explicit modelling decisions, and requiring more data to train. Again we see that the issues of historical data not containing counterfactual histories, or being too limited for our purposes, are being addressed, but doing so introduces increases our reliance on statistical models, rather than on observed data.

Modeling feedback is important for training environments to be realistic. For example when hedging or trading strategies are trained and tested on historical data, the success of the model still cannot be reliably demonstrated, even when using holdout sets for validation. Training environments with appropriately modelled feedback can, at least partially, mitigate this issue. Such training environments are also critical for deploying on-line reinforcement learning solutions as they allow pre-training of these systems before implementation in the real

market. This is critical in applications where the costs and risks of exploration are significant.

Agent-based modelling has, in recent years, allowed for the design of high-fidelity simulated markets (Belcak et al., 2020; Byrd et al., 2020). These artificial markets can run millions of in-silico trials to test counterfactual theories, research emergent phenomena, and train and test algorithms.

A current trend is that quantitative funds are looking to establish risk management systems that develop scenarios with no historical precedent[4]. With a simulator, one can perform training and backtesting for trading, execution, and placement algorithms under various conditions. Causal assessments can be performed for market impact and market slippage. Lastly, simulators can also be used as a means of generating synthetic data, given that financial data of sufficient granularity is often highly proprietary and/or expensive to access.

Standardized cleaning and preprocessing methodologies

Issues surrounding data quality often are specific to the particular use-case. The increasing use of varied data sources, often with little standardization, will inevitably result in the preprocessing of financial data becoming more important.

Some approaches, for example the no-arbitrage constraints for option books in Cohen et al. (2020), rely on preprocessing data to conform with prescribed characteristics. In this case, given the no-arbitrage constraints restrict the range of possible option prices significantly, imposing these requirements has the potential to address errors coming from a variety of sources. These methods can also be run on data coming from a SDG or ABM, in order to ensure that the simulated data is economically reasonable.

An approach which can serve to highlight potential concerns, is to look at the sampling frequency and periods of data. By comparing the results of using different but comparable datasets, it is possible to gauge the stability of calibration and models, and hence to identify causes for concern.

More generally, learning from other areas of machine learning, the development of common examples, codebases and resources, in an open-source manner, has the potential to improve the identification and processing of data errors. A significant risk is that inappropriate methods for dealing with data errors will be separately developed, implemented and used, without sufficient oversight or criticism. The use of well-developed, understood, and standardized tools is a key part of modern machine learning, and the development of preprocessing tools appropriate to finance should be seen in this light.

As part of this, the development of publicly discussed use-cases, with realistic data, would allow for new methods to be evaluated in a consistent manner, and for best-practice to be developed. While this is the case in other areas of machine learning, there is still much scope for improvement when it comes to financial data and problems.

[4] In 2017, Jane Street published a technical presentation of their own exchange, motivated to train and test new algorithms and models. It has been reported to handle messages in the rate of 500k/second with latencies in the single digit microseconds (Nigito, 2021).

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

33.4 The role of models

33.4.1 Model risks

As we have discussed, the use of machine learning changes, but does not eliminate, the use of classical mathematical modelling. Classical models may appear explicitly in machine learning methods (for example, in a hybrid pricing model), or may be subtly incorporated in the simulations used to support more explicitly data-driven approaches. Typically, however, machine learning methods aim to construct models from flexible ('non-parametric') families, combining the classical tasks of model selection and calibration into a single step. In this section, we will discuss the risks which arise from these modelling decisions, in a machine learning context.

It is worth noting that the importance of model risk depends strongly on how machine learning methods are used. Using machine learning tools for numerical procedures typically introduces little additional model risk, as one can often verify the solutions using other techniques. For example, when using a neural network to estimate option prices, for the sake of quickly calibrating the parameters of a classical model, it is straightforward to verify (using traditional PDE or Monte Carlo methods) that the calibrated model gives the correct prices of those options – the neural network is only serving as a numerical tool.

Conversely, end-to-end deep reinforcement learning, for example of a hedging strategy, exposes users to risks in multiple forms: models with too many parameters risk overfitting to available data, leading to both poor performance and a misunderstanding of a model's accuracy; the common use of synthetic and simulated data hides an additional layer of model risk in the training environment; complex models are more exposed to reward hacking, poisoning attacks, and other adversarial concerns and are typically less interpretable than simpler models.

The problem of calculating a price for a financial derivative which is consistent with the market can be seen as equivalent to finding a map that takes market data (e.g. prices of underlying assets, interest rates, prices of liquid options) and returns the no-arbitrage price of the derivative. One way to do this is to select a martingale model (to prevent arbitrage) that can be calibrated to market data, by which we mean that the model matches the observed prices of liquid assets.

While this is a dominating approach in the industry, the introduction of a model necessarily introduces model risk, and there are infinitely many models that can fit market data. In the robust finance paradigm, see Hobson (1998); Cox and Obłój (2011), one takes a conservative approach and, instead of computing a single price, one constructs pricing intervals that are consistent with market data. Without imposing further constraints, the class of all calibrated models might be too large, and consequently, the corresponding pricing intervals too wide to be of practical use (Eckstein et al., 2021). It is therefore natural to consider a smaller search space of models (e.g. SDEs with continuous coefficients) and use data and machine learning to select an appropriate model (i.e. the coefficients of the SDE).

This approach has been recently applied in Gierjatowicz et al. (2020). The key

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

idea is to use SDEs to describe the model dynamics but, instead of fixing its coefficients, to allow the drift and diffusion to be given by an overparametrized neural network. These 'neural-SDE' models not only provide a systematic framework for model selection, but can also produce robust estimates on the derivative prices.

A concern for model risk is not new in finance, but the use of machine learning methods can be seen as typically emphasising some risks over others. In Table 33.1, we present an overview of the typical distinctions between handcrafted and machine learning perspectives on model risk.

Table 33.1 Comparing Typical Risks Between Handcrafted and Machine Learning Methods

Risk	Handcrafted	Machine Learning
Structural Risk	Lower-dimensional models which are easy to calibrate, but fail to capture all aspects of the market's behaviour. Generally a higher bias than variance and more prone to underfitting.	High-dimensional models which require large amounts of data to calibrate, but can capture fine detail when fitted well. Can often incorporate new sources of information in a convenient manner. Generally a higher variance than bias and more prone to overfitting.
Model Sensitivity	Few parameters and model inputs. Model outputs vary smoothly with calibration and input. Well understood sensitivities to erroneous inputs.	High-dimensional parameters and data inputs. Model outputs can vary sharply with inputs. Sensitivities to erroneous inputs can vary significantly.
Adversarial Attacks	Reasonably robust calibration and not susceptible to data poisoning attacks. Calibration can be easily monitored by users. Adversarial defences not a key part of most models.	Susceptible to attacks, require robust training and adversarial defences, but these can be incorporated as a key part of the model. Not easily monitored by users.
Model Drift	Models naturally incorporate economic intuition and underpinnings. Few parameters to update online, but do not often incorporate updating as a core part of the model.	Model based on data patterns which may change over time. Many parameters need to be updated dynamically, which can lead to unstable behaviour. Model updating can be included as a core part of the approach.

Structural risk

Within a machine-learning paradigm, one usually combines the stages of model selection and calibration. Given data on a supposed relationship or phenomenon, one aims to directly fit a model to this data with which to predict, simulate and build understanding.

For our example of pricing and hedging of options, we can focus on the task of pricing an option given historical market data. Our data consist of historical observations of market data, and we aim to build a function which can take new

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

observations and provide us with prices in the future. To do this, some basic modelling assumptions are unavoidable:

- Does the price of an option depend only on the current market state, on the recent past, or on a long history of market observations. Equivalently, what are the inputs to the pricing function that I wish to find?
- Do I wish to make conditional predictions (say of an option price given a stock price) or do I wish to give simulations of both simultaneously?
- Does the relationship between market observations and prices remain stable through time? If not, how do I choose training periods which are representative of the situations where I will apply my function in the future?
- If the observed prices are not perfectly predicted by market data, so I have noisy observations, are the noises independent, or are they correlated between times and assets?

In each case, the answer given to these questions will be incorporated in our machine learning model, and introduces model risk at a structural level.

These general concerns are common to both classical and machine learning methods, however the increased flexibility of machine learning methods may suggest that (as one can include more observations in a model), they would be less present in a machine learning approach.

Even after these general concerns are addressed, machine learning methods introduce risks similar to the 'model risk' of classical mathematical finance. Within the paradigm of machine learning, models are not chosen explicitly but implicitly, through the choice of training data, training algorithm and the often *ad hoc* choice of a large parametric model (e.g. a neural network and its architecture). Unlike handcrafted methods that are explicitly specified, or hybrid approaches relying on feature engineering, neural networks construct an internal representation of features to capture and approximate functions.

With neural networks, model specification is not in the direct control of the modeller. Due to this flexibility in feature specification, a larger space of plausible models are explored than in traditional or many other machine learning approaches. The cost of this flexibility is that the model selected may not be the 'best' available. Since the fitting of traditional models typically involve solving some convex optimisation problem, a best model can be identified due to the existence of a unique minimum. Neural networks fitting techniques are typically non-convex and many good solutions can be found.

Adding fuel to the fire, neural networks are known to be sensitive to initialisation conditions (McMormack and Doherty, 1993). Moreover, many sources of randomness are often injected into the training phase of neural networks, this includes the use of dropout (where some neurons are randomly set to zero for network regularisation), early stopping (where the process of gradient descent stops when the performance on a validation set stops improving), and stochastic gradient descent (where random selections of observations are used to fit the network). These additional factors introduce uncertainty in the output of neural network models. The injection of noise during training is critical to the

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

performance of these methods, and it leads to, so called, implicit regularisation (Neyshabur, 2017). That means that stochastic gradient descent methods select regularised solutions, even though regularisation is not explicitly incorporated at the training stage (Heiss et al., 2019). In this sense, the model selection step of classical approaches is replaced by the choice of training algorithm, which has a less easily understood connection with model performance.

Drawing from interpretability research by Lipton (2018), any model's transparency can be broken down in *simulatibility*, *decomposability*, and *algorithmic transparency*. With simulatibility, a human should be able to step through each of the operations in a reasonable time; with decomposability, each part of the model has an intuitive explanation that is understood in isolation; with algorithmic transparency, there are theoretical guarantees about the behaviour of the algorithm, for example certainty of convergence. Going down this checklist it is clear that neural networks lack simulatibility and decomposability because the parameters in the hidden layers do not have an intuitive explanation. Moreover, for non-convex problems stochastic gradient descent is not guaranteed to converge. Instead, one can show that the weights of neural networks are represented by Monte Carlo samples from optimal distribution over the parameter space. This perspective allows one to establish convergence guarantees, but does not help with the issue of interpretability (Hu et al., 2019; Jabir et al., 2019).

Model sensitivity

A key selling point of neural networks is their ability to work with high dimensional inputs. However, this comes with a well documented issue of sensitivity, where the learnt relationships vary wildly with small perturbations to the underlying inputs.

Models are known to be fragile when using high dimensional inputs. The reasons are numerous: given the randomness involved in training neural networks, some inputs may spuriously be considered important. This is a particular issue when only limited data is available, or simulated data (from a low dimensional model) is used as training data – simulated data will typically not explore a full range of market conditions (as it is constrained by the model from which it's generated), and so the neural net will not learn to provide good answers when novel conditions are encountered. Secondly, when many inputs are used within a model, there is an increased probability that some variables might not be available when a model is put into practice.

Since the model specification of neural networks is implicit, the modeller and end-user of these methods will often no longer understand how the model has been fit, significantly increasing model specification risk. Consequently, it is not clear how we can quantify sensitivity of the model. The field is therefore largely left with developing more interpretable model alternatives (Nakagawa et al., 2019) or using post-hoc explanations to assess and visualise what models have learned (Li et al., 2020). This however also comes with risk as many post-hoc explanation are not robust and may lead to false sense of security (Anders et al., 2020).

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

Robustness and adversarial attacks

The competitive nature of financial markets often leads to particular concerns for machine learning models. As models are used in increasingly automated ways, they need to be able to respond to the pressures placed on them by competitive forces, who have strong incentives to identify and exploit potential weaknesses of a model.

For example, we could consider our challenge of managing an options portfolio, but in a context where market price impact reduces the efficiency of trading. A classic model for order execution with market impact, Almgren and Chriss (2001), yields deterministic policies for executing a large buy or sell order, which may have the undesirable effect of 'information leakage' (revealing your strategy to other market participants) when used in an illiquid market. In the more complex situation of managing a portfolio, one could consider building a neural network model to perform this task (for optimal execution, a model of this type is given in work by Ning et al. (2018)). The additional randomness of the neural network model would arguably assist in preventing information leakage, when compared to the traditional model. Nevertheless, it is *a priori* unclear whether this additional randomisation would be sufficient, or whether further precautions against information leakage would be needed.

Adversarial attacks can be grouped into many categories, for example, attacks can either be intentional or unintentional. Behzadan and Munir (2018) splits them into attacks on model confidentiality, integrity (does the model behave as intended?), and availability (can the model be disabled by an external actor?). Attacks could also be split into the components that are susceptible to the attack, for reinforcement learning this includes the environment, the observation channel, the reward channel, the decision making system, and the online training system.

We can consider various way in which a financial reinforcement learning agent can be attacked, with a simple description and illustrative example. These classifications have been adapted from the adversarial threat a matrix developed by MITRE in collaboration with Microsoft, IBM, NVIDIA, and Bosch (Kumar et al., 2020).

In Table 33.2, we present examples of adversarial attacks against a trading system. We first list those which are internal to the company, many of which can arise inadvertently in building and implementing machine learning methods and then follow with examples of attacks that an adversary can exploit without having direct access to a trader's codebase. The examples are our own, and are purely illustrative.

These intentional and unintentional attack examples are hypothetical, and relate to problems seen in other machine learning domains. Nonetheless, these examples have significant implications for financial model risk management. A substantial level of compounded risks could exist where multiple of these susceptibilities overlap.

Although there is a need to test and benchmark the robustness and resilience of trading agents with private systems and historical data, these agents ultimately

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

Table 33.2 Examples of Adversarial Attacks in Finance

Failures	Description	Example
Reward Hacking	When training, the stated reward differs from a true reward.	A learning agent was trained to create a perfect hedge, however transaction costs were poorly modelled, leading to poor performance.
Side Effects	A reinforcement learning system disrupts the environment by advancing its goal.	A model has learned an order execution strategy for an illiquid asset, but by executing this strategy, changes the dynamics of the order book significantly, leading to increased risk.
Distributional Shifts	The system is trained on one environment, but unable to adapt to changes.	A pricing model was trained on data during normal times, and is unable to react to the higher correlations between assets during crises.
Natural Adversarial Examples	Even without being attacked, the system fails from natural errors.	A pricing model was trained individually for each strike and maturity, resulting in arbitrageable prices being offered in the market.
Common Corruptions	The system is not able to deal with common corruptions.	A pricing model failed due to a halt on trading being placed on a closely related underlying instrument.
Incomplete Testing	The system is not tested on the right environment nor over multiple periods.	A pricing model is tested only on one exchange, but is deployed in multiple locations with differing market behaviours.
Poisoning attack	Contaminate training phase.	Contaminated data is introduced into a pricing model, for example when using sentiment analysis based on social media.
Model stealing	Recover the entire model.	A proprietary model is trained and can be queried online by counterparties. By repeated queries it is possible that the inputs can be matched with the outputs, to reverse engineer the original model.
Model inversion	Recover hidden features.	A pricing model is trained using proprietary trading data on market impact. The fitted model is then made public, without the underlying data. By repeated queries, it may be possible to extract the training data used (Fredrikson et al., 2015).
Reprogramming system	Repurpose system for other use.	An online pricing model is used to identify expected future market volatility.
Adversarial example in physical domain	Fool a system by changing some interface component.	An adversary determines that a pricing model has sensitivity to the volumes deep in the order book – by posting to this part of the book, they influence the model's behaviour.
Exploit software dependencies	The use of traditional software exploits.	The model relies on code dependencies; these dependencies are exploited by modifying the code to introduce nonsensical values, leading to a trading halt. (The 2016 NPM/left-pad debacle illustrates this external dependency risk, where a disgruntled developer deleted a tiny piece of code that 'broke' the internet (Collins, 2016).)

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

have to move to the real world, where a slight distributional shift could impair performance. In other areas of machine learning, in addition to internal testing, models can be subjected to public audits. However, in finance the competitive risks from revealing private models are significant, leading to a far lower level of transparency.

Model drift

A good model not only fits historical data well, but also captures changes in the environments in which it is deployed. The challenge of updating models exists in both handcrafted and machine learning models, and reflects the basic challenge that finance does not operate according to stable physical laws, but arises from the interactions of many agents.

The challenge of changing market behaviour can be significant: the overwhelming belief is that the value of a derivative and its underlying are kept in line due to no-arbitrage. However, during the 2007-08 financial crisis, these relationships were observed to break down, as arbitrage calculations did not account for counterparty creditworthiness. As a result, a theoretical arbitrage opportunity was observable in the market, but was not available in practice (Baba and Packer, 2009).

Handcrafted models, typically, require updates of few parameters to capture the shift of the data distribution. For overparameterised models, this may not be the case, and a small change in the data may require a significant change in the model. For example: fraud detection models lose their discriminatory power against maliciously evolving strategies, hedging strategies have to evolve as market conditions are changing.

Off-line machine learning suffers from a lack of robustness to distribution shifts, and hence a lack of on-line monitoring can significantly impair its performance (Sugiyama and Kawanabe, 2012). This has become particularly clear in recent years in other applications of machine learning. For example, in the airline industry it was quickly realised that the standard machine learning pricing models that study flight patterns, fuel costs, and user behaviour became useless during the covid-19 pandemic, with data scientists choosing to fall back on traditional macroeconomic modelling (McCartney, 2020).

On the other hand, online learning approaches have the promise of being able to dynamically and naturally adapt to new situations (Zeng and Klabjan, 2019; Soleymani and Paquet, 2020). This comes with significant issues, however, as these methods require training at a meta-level: the rate at which they adapt to new information needs to be tuned and adjusted, with rapid adjustment speeds typically associated with increased volatility in performance.

33.4.2 Model solutions

Any given model provides only a crude approximation to reality; the risk of using an inadequate model is often hard to detect and quantify. While modern data science techniques are opening the door to more data-driven model selection

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

mechanisms, this comes with new risks, as described previously. In this section, we argue that by combining old and new approaches, it is possible to regain control over newly emerging risks (e.g. lack of interpretability) while improving over classical models currently favoured by industry. We base our presentation on a few hybrid modelling approaches which have recently emerged in the research literature.

A natural idea is to incorporate prior knowledge/modelling into deep learning. This can be achieved through incorporating modelling constraints during the training. However, as the number of constraints increases, and hence the search space of possible network parameters decreases, stochastic gradient descent algorithms struggle to find good solutions, so bespoke machine learning methods need to be developed.

Machine learning as a numerical tool

As mentioned above, using machine learning as a numerical tool introduces only modest model risks, while potentially providing significant speed and accuracy benefits. In Sabate-Vidales et al. (2018, 2020), the authors developed deep learning algorithms for solving parametric families of (path-dependent) partial differential equations (P)PDEs that arise in pricing and hedging. The key idea in these works is to use a probabilistic representation of the (P)PDE, and learn both the solution and its gradient simultaneously. An advantage of this approach is that the gradient of the solution to the (P)PDE provides access to the hedging strategies. While this method is of interest in its own right, it can also be used as a control variate for unbiased Monte Carlo pricing. In other words, by combining deep learning with standard Monte Carlo pricing, one can remove the bias due to approximation with neural nets and easily compute confidence intervals (which are, in general, hard to obtain for large networks). This approach has been tested on several models and (path dependent) payoffs. We stress that while the literature on deep learning for PDEs is growing rapidly, for finance applications it is critical to approximate parametric *families* of PDEs, where parameters correspond to the possible values of calibrated coefficients of the model. A similar observation has been made in Horvath et al. (2020).

Another interesting approach, one that combines ideas emerging from ML and classical modelling has been put forward in Lyons et al. (2019). The key idea here is to lift both modelling and pricing into the signature space. Intuitively, signatures provide efficient basis functions for representing functionals defined on the path space (e.g exotic derivatives or non-Markovian models) and play a similar role to polynomials on Euclidean space. In particular, the signature expansion of a path represents the values of integrals against that path, and so can capture the effect of dynamic trading and hedging. The classical idea of replicating an option via trading in the market then reduces to regressing the option payoff on the signature of the underlying and other vanilla securities.

It has been shown that one can effectively represent many exotic derivatives using this signature expansion, and consequently obtain the prices of derivatives in terms of the expectation (under the pricing measure) of the signature expansion

https://doi.org/10.1017/9781009028943.034 Published online by Cambridge University Press

terms. Consequently, one only needs to calibrate expected signatures to market data, which in some settings can be done efficiently. The advantage of using signatures when compared with recursive neural networks is that the computational cost does not increase with the number of time points in a time-series.

The idea of model selection using signatures has been proposed in Arribas et al. (2020). Here, one still works with the familiar SDEs type model but aims to learn (possibly non-Markovian) coefficients from data.

Expert knowledge

A viable approach to controlling the risk of non-transparent model specifications is to develop algorithms and training methods that embed expert knowledge into the architecture or training stage of machine learning. A handful of papers have attempted to embed financial domain knowledge into their models. These methods can offer regularisation, efficiency, consistency, and stability benefits.

Drawing from the review by Ruf and Wang (2020), methods that adjust the *architectural* design of neural networks include models that incorporate a homogeneity hint by training a neural network in two parts, the first part controls for moneyness, and the other for time-to-maturity (Garcia and Gençay, 1998). Other methods restrict the shape of outputs (Dugas et al., 2001) or enforce no-arbitrage conditions such as the convexity of a neural network pricing function and monotonicity (Zheng et al., 2019).

Approaches that impart expert knowledge at the *training* stage include data augmentation, which involves the generation of synthetic data to help with neural network training (Yang et al., 2017), adjustments in the penalty terms of the loss function to promote no-arbitrage (Itkin, 2019; Ackerer et al., 2019), as well as the development of bespoke training algorithms for neural networks for options hedging, including the use of the extended Kalman filter, sequential Monte Carlo, and evolutionary algorithms (Niranjan, 1996; de Freitas et al., 2000; Palmer, 2019).

Benchmarks

A safe and efficient transition toward using machine learning in finance is only possible when models and methods are well understood and tested on reliable data sets. In other areas of machine learning, standard benchmarks and data sets are a common way to proof-test new methodologies. For example, recent advances in computer vision or reinforcement learning were significantly accelerated due to the emergence of challenging benchmarks, such as ImageNET (Deng et al., 2009) or ALE (Bellemare et al., 2013). These benchmarks have enabled open, systematic cross-validation of various AI solutions.

In machine learning, the term ‘benchmarking’ has been used to refer to the evaluation and comparison of machine learning models, particularly regarding their ability to learn patterns from benchmark datasets (Olson et al., 2017). This process can be thought of as a check to validate the improvement of a new method, but also more broadly to identify the respective advantages and disadvantages of each method. Comparisons can be made across a wide range of metrics,

for example accuracy in detecting signals, interpretability, and computational complexity.

Currently, in finance, various algorithms and machine learning methods are tested on disparate data sets, which are often only accessible to a small community or at high cost. A consequence of this is that very little comparison of methods is done, and we have little understanding of the appropriateness or optimality of these methods. In addition, evaluating new AI techniques on real-world applications often requires expert domain knowledge and consideration of scalability and the cost of development.

A key difficulty, in financial applications, is that a more open approach to benchmarking will often involve revealing details of each participant's methodologies. While this is reasonable within the academic community, within industry it is clear that confidentiality is needed, both regarding algorithms and, in some cases, their performance. For this reason, it is important to build our understanding of which problems can be discussed and benchmarked in a public way, and which related data science problems provide insight for those cases where confidentiality is needed.

The typical datasets which the benchmarking literature has well studied come from real-world data and simulated data with known underlying patterns. As alluded to before, in finance there are relatively few datasets that have been made publicly available, and often these contain only a small sample of the data that would be needed in practice. There is therefore a clear opportunity for Finance to benefit from synthetic data generators. Synthetic data has been used in other fields[5] but has not yet flourished in the financial literature.

Benchmarking has its own problems, many of which are not new to machine learning. There has been an increasing concern that published research findings are misleading due to the number of studies addressing the same question and datasets (Ioannidis, 2005). Benchmarking has a similar problem, in that a lot of models are prodding the same unchanging datasets leading to a lack of generalisation. Studies reveal that the accuracy of state-of-the-art deep learning models can drop from 4%-10% when moving to a new test-set, highlighting the risk of overfitting (Recht et al., 2018). For this reason, the regular evaluation and updating of benchmarks remains important for future development.

Adversarial defenses

In order to be reliably implemented, algorithms must be robust with respect to a variety of objectives (e.g. safety, accuracy). Summarizing the range of adversarial challenges outlined above, we see that machine learning pipelines should come with robustness guarantees against: (i) shifts in data distribution (distributional robustness), (ii) intentional input manipulations (adversarial robustness) and (iii) intentional feature manipulation to 'game' the system (strategic robustness).

Recent work (Huang et al., 2017) has begun to address these issues for neural

[5] For example, the Open Graph Benchmark, released in 2020, has become a popular repository of challenging and realistic benchmark datasets to help facilitate scalable, robust, and reproducible graph machine learning research (Hu et al., 2020).

network based models. Drawing on adversarial machine learning and distributionally robust optimisation (Rahimian and Mehrotra, 2019; Cohen et al., 2019a; Wicker et al., 2020), it is possible to certifiably train models to provably ensure robustness, by providing guaranteed bounds on the probability of the model output (decision) satisfying a combination of objectives.

Data-driven models cannot automatically guarantee model robustness (Kwiatkowska, 2019). An adversarial defence is anything that decreases the efficacy of adversarial attacks. There are a range of techniques that can be used to provide adversarial defences; they can generally be classified into adversarial training methods, randomisation-based schemes, denoising methods, and provable defences.

- Adversarial learning techniques simply train a neural network using adversarial samples. It is one of the most effective defences against attacks as revealed in benchmark studies (Madry et al., 2018). These can be thought of as a preprocessing technique.
- Randomisation schemes can also protect against perturbation in inputs. These generally involve some transformation, such as random resizing, or can also be achieved by adding a noise layer to the neural network (Liu et al., 2018).
- Denoising inputs in the prediction phase can help to rectify or remove adversarial perturbations. This denoising can be done with generative adversarial networks or autoencoders and can be thought of as a postprocessing technique (Xie et al., 2019).
- Provable defences are unlike the above approaches in that they are theoretically proven, rather than purely being experimentally validated. These methods can certify a level of robustness before the prediction stage (Balunovic and Vechev, 2019).

The defences listed here can only verify and protect a system against a limited number of attacks. Security vulnerability attacks will have to be dealt with using domain expertise, rather than relying on generalist defence mechanisms. Adversarial defences will not protect against a badly developed model, and appropriate fail-safe mechanisms and human oversight remain a critical part of implementation.

Explainability

Explainability allows for human oversight of machine learning to be carried out effectively, ensuring that model risk is understood and controlled. Understanding the causes behind performance is a common part of risk management – for example, the 'Profit and Loss attribution test', which forms part of the Fundamental Review of the Trading Book (BIS, 2019), requires a bank's hypothetical profits using front-office pricing models to be explained against their back-office risk models and factors, as part of the validation of those risk models.

The understanding of models and their risks is a significant challenge in finance. The 2007–2008 financial crises demonstrated that copulas, especially those proposed by Li (2000), were underpowered for modelling the risks of CDOs, but

yet still were too large and complex to be understood and critiqued by users. In contrast, machine learning models are overpowered, have shiny user-interfaces, but are even more obscure. Machine learning has been promoted in much-cited papers as a method for systemic risk analysis, with only limited discussion of the risks of using machine learning and its lack of interpretability (Kou et al., 2019; Aziz and Dowling, 2019).

Neural networks are not inherently explainable, as input features become entangled and compressed into a single value via repeated non-linear transformations of a weighted sums (Ras et al., 2022). Explainability can be improved by *prima facie* selecting a more interpretable 'white box' model: that is, adopting models which intrinsically are easier to query and understand. Neural network models can be designed to be more interpretable through joint training (Hendricks et al., 2016; Iyer et al., 2018) or including attention mechanisms (Bahdanau et al., 2016; Devlin et al., 2019; Anderson et al., 2018).

Although these solutions apply for neural networks in general, they do not necessarily apply in a reinforcement learning framework. In this setting, rule-based (Verma et al., 2018; Hein et al., 2017), or hierarchical (Shu et al., 2017) methods are available. The purpose of rule-based methods is to present the policies in high level human-readable language, e.g. IF-THEN sequences. Hierarchical methods divide policies into simpler sub-tasks, each of which are separately more interpretable than a flat policy, and are therefore useful to explain individual decisions, i.e. they provide 'local' interpretability.

The above interpretable models generally forgo some performance for enhanced comprehensibility. As a result, as performance is often the primary concern, explainability techniques which can be applied to a black-box model need to be identified. These techniques can be grouped under the name 'post-hoc' explainability.

The types of post-hoc explanation methods are broad and include perturbation analysis, gradient analysis, example based explanations, and surrogate-modelling for local and global explanations (Adadi and Berrada, 2018). Different applications and tasks require a different balance between explainability and performance.

'Deep' reinforcement learning is based on neural network models, and adds an additional layer of incomprehensibility to the modelling process (Mnih et al., 2013). Reinforcement learning models are complex, but it is often possible to use interpretable surrogate models as a means of simplifying and representing their actions; this is often easier than developing inherently interpretable models (Puiutta and Veith, 2020). A range of surrogates are available for this purpose, and include genetic programming techniques (Hein et al., 2018), causal DAGs (Madumal et al., 2020) and the use of tree-based models to approximate predictions (Coppens et al., 2019). However, when using surrogate models for explainability, it is wise to keep the underlying model as simple as possible, in order to make it easier for a surrogate model to reproduce its outputs.

Monitoring and control

Models not only have to be validated on historical data, i.e. benchmarked, they also have to be monitored and controlled when running 'live'. In machine learning, this is related to 'concept drift', which refers to data distributions changing over time, leading to faulty predictions (Žliobaitė et al., 2016). The hope is that, with online learning incorporated in the approach, models can self-diagnose and self-correct when this occurs, but this is not always the case. Continuous recalibration may not be possible in all settings, due to regulatory requirements and the cost of recalibration (Cohen et al., 2019b). A good survey of concept and data drift and how to deal with it can be found in Gama et al. (2014). The importance of monitoring, recalibrating and updating systems, and ensuring sufficient human control, is a key part of the implementation of most automated systems in practice.

In a financial setting, we might also want to base the criteria for drift on the execution of other methods (for example, handcrafted strategies) that are run in parallel as 'controls' for performance. This allows one to study those occasions in which the performance of controls differed significantly from the model, highlighting points of concern.

Machine learning models need more extensive monitoring procedures than handcrafted approaches due to the various risks they come with. Nevertheless, the promise of improved performance, the flexibility of modelling, and the speed advantages associated with embracing these new technologies means that there is no doubt about their broad incorporation into many parts of the finance industry.

References

Ackerer, Damien, Tagasovska, Natasa, and Vatter, Thibault. 2019. Deep smoothing of the implied volatility surface. Available at SSRN 3402942.

Adadi, Amina, and Berrada, Mohammed. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access*, **6**, 52138–52160.

Almgren, Robert, and Chriss, Neil. 2001. Optimal execution of portfolio transactions. *Journal of Risk*, **3**, 5–40.

Anders, Christopher, Pasliev, Plamen, Dombrowski, Ann-Kathrin, Müller, Klaus-Robert, and Kessel, Pan. 2020. Fairwashing explanations with off-manifold detergent. Pages 314–323 of: *International Conference on Machine Learning*.

Anderson, Peter, He, Xiaodong, Buehler, Chris, Teney, Damien, Johnson, Mark, Gould, Stephen, and Zhang, Lei. 2018. Bottom-up and top-down attention for image captioning and visual question answering. Pages 6077–6086 of: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Andreou, Panayiotis C., Charalambous, Chris, and Martzoukos, Spiros H. 2010. Generalized parameter functions for option pricing. *Journal of Banking & Finance*, **34**(3), 633–646.

Arribas, Imanol Perez, Salvi, Cristopher, and Szpruch, Lukasz. 2020. Sig-SDEs model for quantitative finance. In: *Proc. First ACM International Conference on AI in Finance*. Article 7, 1–8. DOI: https://doi.org/10.1145/3383455.3422553.

Arthur, W, Brian, Holland, John H., LeBaron, Blake, Palmer, Richard, and Tayler, Paul. 1996. Asset pricing under endogenous expectations in an artificial stock market. Pages 15-44 in: *The Economy as an Evolving Complex System II*, W. B. Arthur, S. Durlauf, and D. Lane (eds). Addison-Wesley. Available at SSRN 2252.

Aziz, Saqib, and Dowling, Michael. 2019. Machine learning and AI for risk management. Pages 33–50 of: *Disrupting Finance*. Palgrave Pivot.

Baba, Naohiko, and Packer, Frank. 2009. Interpreting deviations from covered interest parity during the financial market turmoil of 2007–08. *Journal of Banking & Finance*, **33**(11), 1953–1962.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. 2016. Neural machine translation by jointly learning to align and translate. ArXiv:1409.0473v7.

Balunovic, Mislav, and Vechev, Martin. 2019. Adversarial training and provable defenses: Bridging the gap. In: *International Conference on Learning Representations*.

Barucci, Emilio, Cherubini, Umberto, and Landi, Leonardo. 1997. Neural networks for contingent claim pricing via the Galerkin method. Pages 127–141 of: *Computational Approaches to Economic Problems*. Springer.

Bayer, Christian, Horvath, Blanka, Muguruza, Aitor, Stemper, Benjamin, and Tomas, Mehdi. 2019. On deep calibration of (rough) stochastic volatility models. ArXiv:1908.08806v1.

Beck, Christian, Becker, Sebastian, Cheridito, Patrick, Jentzen, Arnulf, and Neufeld, Ariel. 2021. Deep splitting method for parabolic PDEs. *SIAM Journal on Scientific Computing*, **43**(5), A3135–A3154.

Behzadan, Vahid, and Munir, Arslan. 2018. The faults in our pi stars: Security issues and open challenges in deep reinforcement learning. ArXiv:1810.10369v1.

Belcak, Peter, Calliess, Jan-Peter, and Zohren, Stefan. 2020. Fast agent-based simulation framework of limit order books with applications to pro-rata markets and the study of latency effects. ArXiv:2008.07871v2.

Bellemare, Marc G., Naddaf, Yavar, Veness, Joel, and Bowling, Michael. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, **47**, 253–279.

Bengio, Yoshua, Goodfellow, Ian, and Courville, Aaron. 2017. *Deep Learning*. MIT Press.

BIS. 2019 (Dec). MAR32 – Internal models approach: backtesting and P&L attribution test requirements. `https://www.bis.org/basel_framework/chapter/MAR/32.htm?inforce=20220101`, [Accessed Feb. 1, 2021].

Buehler, Hans, Gonon, Lukas, Teichmann, Josef, Wood, Ben, Mohan, Baranidharan, and Kochems, Jonathan. 2019. Deep hedging: hedging derivatives under generic market frictions using reinforcement learning. *Swiss Finance Institute Research Paper*.

Buehler, Hans, Horvath, Blanka, Lyons, Terry, Perez Arribas, Imanol, and Wood, Ben. 2020. A data-driven market simulator for small data environments. Available at SSRN 3632431.

Byrd, David, Hybinette, Maria, and Balch, Tucker Hybinette. 2020. Abides: towards high-fidelity market simulation for AI research. Pages 11–22 of: *Proc. ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. DOI: https://doi.org/10.1145/3384441.3395986.

Cartea, Álvaro, and Sánchez-Betancourt, Leandro. 2021. The shadow price of latency: Improving intraday fill ratios in foreign exchange markets. *SIAM Journal on Financial Mathematics*, **12**(1), 254–294.

Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., and Kegelmeyer, W. Philip. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, **16**, 321–357.

Cohen, Jeremy M., Rosenfeld, Elan, and Kolter, J. Zico. 2019a. Certified adversarial robustness via randomized smoothing. Pages 1310–1320 of: *Proc. ICML*. `http://proceedings.mlr.press/v97/cohen19c.html`.

Cohen, Samuel N., Henckel, Timo, Menzies, Gordon D., Muhle-Karbe, Johannes, and Zizzo, Daniel J. 2019b. Switching cost models as hypothesis tests. *Economics Letters*, **175**, 32–35.

Cohen, Samuel N, Reisinger, Christoph, and Wang, Sheng. 2020. Detecting and repairing arbitrage in traded option prices. *Applied Mathematical Finance*, **27**(5), 345–373.

Collins, Keith. 2016. How one programmer broke the Internet by deleting a tiny piece of code. *Quartz Magazine*, `https://qz.com/646467`, [Accessed Feb. 1, 2021].

Coppens, Youri, Efthymiadis, Kyriakos, Lenaerts, Tom, Nowé, Ann, Miller, Tim, Weber, Rosina, and Magazzeni, Daniele. 2019. Distilling deep reinforcement learning policies in soft decision trees. Pages 1–6 of: *Proc. IJCAI 2019 Workshop on Explainable Artificial Intelligence*.

Cox, Alexander M.G., and Obłój, Jan. 2011. Robust pricing and hedging of double no-touch options. *Finance and Stochastics*, **15**(3), 573–605.

Davis, Mark H.A., Panas, Vassilios G., and Zariphopoulou, Thaleia. 1993. European option pricing with transaction costs. *SIAM Journal on Control and Optimization*, **31**(2), 470–493.

de Freitas, João F.G., Niranjan, Mahesan, and Gee, Andrew H. 2000. Hierarchical Bayesian models for regularization in sequential learning. *Neural Computation*, **12**(4), 933–953.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. 2009. Imagenet: A large-scale hierarchical image database. Pages 248–255 of: *2009 IEEE Conference on Computer Vision and Pattern Recognition*.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. Pages 4171–4186 of: *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. `https://aclanthology.org/N19-1423`.

Dixon, Matthew F., Halperin, Igor, and Bilokon, Paul. 2020. *Machine Learning in Finance*. Springer.

Dugas, Charles, Bengio, Yoshua, Bélisle, François, Nadeau, Claude, and Garcia, René. 2001. Incorporating second-order functional knowledge for better option pricing. Pages 472–478 in: *Advances in Neural Information Processing Systems*.

Dugas, Charles, Bengio, Yoshua, Bélisle, François, Nadeau, Claude, and Garcia, René. 2009. Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, **10**(6), 1239–1262.

Eckstein, Stephan, Guo, Gaoyue, Lim, Tongseok, and Obloj, Jan. 2021. Robust pricing and hedging of options on multiple assets and its numerics. *SIAM Journal on Financial Mathematics*, **12**(1), 158–188.

Fan, Jianqing, and Zhou, Wen-Xin. 2016. Guarding against spurious discoveries in high dimensions. *Journal of Machine Learning Research*, **17**(1), 7123–7156.

Fefferman, Charles, Mitter, Sanjoy, and Narayanan, Hariharan. 2016. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, **29**(4), 983–1049.

Ferguson, Ryan, and Green, Andrew. 2018. Deeply learning derivatives. ArXiv:1809.02233.

Fredrikson, Matt, Jha, Somesh, and Ristenpart, Thomas. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. Pages 1322–1333 of: *Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security*.

Fu, Rao, Chen, Jie, Zeng, Shutian, Zhuang, Yiping, and Sudjianto, Agus. 2019. Time series simulation by conditional generative adversarial net. ArXiv:1904.11419.

Gama, João, Žliobaitė, Indrė, Bifet, Albert, Pechenizkiy, Mykola, and Bouchachia, Abdelhamid. 2014. A survey on concept drift adaptation. *ACM Computing Surveys*, **46**(4), 1–37.

Garcia, René, and Gençay, Ramazan. 1998. Option pricing with neural networks and a homogeneity hint. Pages 195–205 of: *Decision Technologies for Computational Finance*. Springer.

Garcia, René, and Gençay, Ramazan. 2000. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, **94**(1-2), 93–115.

Gatheral, Jim. 2006. *The Volatility Surface: A Practitioner's Guide*. Wiley.

Ghaziri, H., Elfakhani, S., and Assi, J. 2000. Neural networks approach to pricing, options. *Neural Network World*, **1**(2/00), 271–277.

Gierjatowicz, Patryk, Sabate-Vidales, Marc, Siska, David, Szpruch, Lukasz, and Zuric, Zan. 2020. Robust pricing and hedging via neural SDEs. Available at SSRN 3646241.

Gnoatto, Alessandro, Reisinger, Christoph, and Picarelli, Athena. 2020. Deep xVA solver – a neural network based counterparty credit risk management framework. Available at SSRN 3594076.

Goetz, Jack, and Tewari, Ambuj. 2020. Federated learning via synthetic data. ArXiv:2008.04489.

Gu, Shihao, Kelly, Bryan, and Xiu, Dacheng. 2018. Empirical asset pricing via machine learning. Tech. Rept. National Bureau of Economic Research.

Hagan, Patrick S., Kumar, Deep, Lesniewski, Andrew S., and Woodward, Diana E. 2002. Managing smile risk. *The Best of Wilmott*, **1**, 249–296.

Han, Jiequn, Jentzen, Arnulf, and E, Weinan. 2018. Solving high-dimensional partial differential equations using deep learning. *Proc. National Academy of Sciences*, **115**(34), 8505–8510.

Hein, Daniel, Hentschel, Alexander, Runkler, Thomas, and Udluft, Steffen. 2017. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence*, **65**, 87–98.

Hein, Daniel, Udluft, Steffen, and Runkler, Thomas A. 2018. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, **76**, 158–169.

Heiss, Jakob, Teichmann, Josef, and Wutte, Hanna. 2019. How implicit regularization of neural networks affects the learned function – Part I. ArXiv:1911.02903.

Hendricks, Lisa Anne, Akata, Zeynep, Rohrbach, Marcus, Donahue, Jeff, Schiele, Bernt, and Darrell, Trevor. 2016. Generating visual explanations. Pages 3–19 of: *European Conference on Computer Vision*. Springer.

Henry-Labordere, Pierre. 2019. Generative models for financial data. Available at SSRN 3408007.

Hobson, David G. 1998. Robust hedging of the lookback option. *Finance and Stochastics*, **2**(4), 329–347.

Horvath, Blanka, Muguruza, Aitor, and Tomas, Mehdi. 2020. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, **21**(1) 11–27.

Hu, Kaitong, Ren, Zhenjie, Siska, David, and Szpruch, Lukasz. 2019. Mean-field Langevin dynamics and energy landscape of neural networks. *Ann. Inst. H. Poincaré Probab. Statist.*, **57**(4), 2043–2065.

Hu, Weihua, Fey, Matthias, Zitnik, Marinka, Dong, Yuxiao, Ren, Hongyu, Liu, Bowen, Catasta, Michele, and Leskovec, Jure. 2020. Open graph benchmark: Datasets for machine learning on graphs. ArXiv:2005.00687.

Huang, Xiaowei, Kwiatkowska, Marta, Wang, Sen, and Wu, Min. 2017. Safety verification of deep neural networks. Pages 3–29 of: *International Conference on Computer-Aided Verification*. Springer.

Hutchinson, James M., Lo, Andrew W., and Poggio, Tomaso. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, **49**(3), 851–889.

Ioannidis, John P.A. 2005. Why most published research findings are false. *PLoS Medicine*, **2**(8), e124.

Itkin, Andrey. 2019. Deep learning calibration of option pricing models: some pitfalls and solutions. ArXiv:1906.03507.

Iyer, Rahul, Li, Yuezhang, Li, Huao, Lewis, Michael, Sundar, Ramitha, and Sycara, Katia. 2018. Transparency and explanation in deep reinforcement learning neural networks. Pages 144–150 of: *Proc. AAAI/ACM Conference on AI, Ethics, and Society*.

Jabir, Jean-François, Šiška, David, and Szpruch, Łukasz. 2019. Mean-field neural odes via relaxed optimal control. ArXiv:1912.05475.

Jacquier, Antoine Jack, and Oumgari, Mugad. 2019. Deep PPDEs for rough local stochastic volatility. Available at SSRN 3400035.

Kolm, Petter N, and Ritter, Gordon. 2019. Dynamic replication and hedging: A reinforcement learning approach. *Journal of Financial Data Science*, **1**(1), 159–171.

Kondratyev, Alexei, and Schwarz, Christian. 2019. The market generator. Available at SSRN 3384948.

Koshiyama, Adriano, Firoozye, Nick, and Treleaven, Philip. 2020. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, **21**(5) 797–813.

Kou, Gang, Chao, Xiangrui, Peng, Yi, Alsaadi, Fawaz E., and Herrera-Viedma, Enrique. 2019. Machine learning methods for systemic risk analysis in financial sectors. *Technological and Economic Development of Economy*, **25**(5), 716–742.

Kumar, Ram Shankar Siva, Nyström, Magnus, Lambert, John, Marshall, Andrew, Goertzel, Mario, Comissoneru, Andi, Swann, Matt, and Xia, Sharon. 2020. Adversarial machine learning-industry perspectives. Pages 69–75 of: *2020 IEEE Security and Privacy Workshops (SPW)*.

Kwiatkowska, Marta Z. 2019. Safety verification for deep neural networks with provable guarantees. In: *Leibniz International Proceedings in Informatics, LIPIcs*.

Lajbcygier, Paul R., and Connor, Jerome T. 1997. Improved option pricing using artificial neural networks and bootstrap methods. *International Journal of Neural Systems*, **8**(04), 457–471.

LeBaron, Blake. 2000. Agent-based computational finance: Suggested readings and early research. *Journal of Economic Dynamics and Control*, **24**(5-7), 679–702.

Levy, Moshe, Levy, Haim, and Solomon, Sorin. 1994. A microscopic model of the stock market: cycles, booms, and crashes. *Economics Letters*, **45**(1), 103–111.

Li, David X. 2000. On default correlation: A copula function approach. *Journal of Fixed Income*, **9**(4), 43–54.

Li, Yimou, Turkington, David, and Yazdani, Alireza. 2020. Beyond the black box: an intuitive approach to investment prediction with machine learning. *Journal of Financial Data Science*, **2**(1), 61–75.

Lin, Zinan, Jain, Alankar, Wang, Chen, Fanti, Giulia, and Sekar, Vyas. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. Pages 464–483 of: *Proc. ACM Internet Measurement Conference*.

Lipton, Zachary C. 2018. The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. *Queue*, **16**(3), 31–57.

Liu, Xuanqing, Cheng, Minhao, Zhang, Huan, and Hsieh, Cho-Jui. 2018. Towards robust neural networks via random self-ensemble. Pages 369–385 of: *Proc. European Conference on Computer Vision*.

Lo, Andrew W. 2019. *Adaptive Markets: Financial Evolution at the Speed of Thought*. Princeton University Press.

Longstaff, Francis A., and Schwartz, Eduardo S. 2001. Valuing American Options by simulation: a simple least-squares approach. *Review of Financial Studies*, **14**, 113–147.

Louizos, Christos, Swersky, Kevin, Li, Yujia, Welling, Max, and Zemel, Richard. 2016. The variational fair autoencoder. In: *Proc. 4th International Conference on Learning Representations*.

Lyons, Terry, Nejad, Sina, and Arribas, Imanol Perez. 2019. Nonparametric pricing and hedging of exotic derivatives. *Applied Mathematical Finance*, **27**(6), 457–494.

MacKenzie, Donald. 2018. Material signals: A historical sociology of high-frequency trading. *American Journal of Sociology*, **123**(6), 1635–1683.

Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, Tsipras, Dimitris, and Vladu, Adrian. 2018. Towards deep learning models resistant to adversarial attacks. *Proc. 4th International Conference on Learning Representations*.

Madumal, Prashan, Miller, Tim, Sonenberg, Liz, and Vetere, Frank. 2020. Explainable reinforcement learning through a causal lens. pages 2493–2500 of: *Proc. AAAI Conference on Artificial Intelligence.*

Mariani, Giovanni, Zhu, Yada, Li, Jianbo, Scheidegger, Florian, Istrate, Roxana, Bekas, Costas, and Malossi, A. Cristiano I. 2019. PAGAN: Portfolio Analysis with Generative Adversarial Networks. Arxiv:1909.10578.

McCartney, Scott. 2020. Coronavirus has upended everything airlines know about pricing. *Wall Street Journal*, Aug 5.

McGhee, William A. 2018. An artificial neural network representation of the SABR stochastic volatility model. *Journal of Computational Finance*, **25**(7), 1–27.

McMormack, C., and Doherty, James. 1993. Neural network super architectures. Pages 301–304 of: *Proc. International Conference on Neural Networks*, vol. 1.

McWaters, R.J., Blake, M., and Galaski, R. 2019. *Navigating uncharted waters: a roadmap to responsible innovation with AI in financial services* Part of the *Future of Financial Services Series*. World Economic Forum.

Mikkilä, Oskari. 2020. *Optimal Hedging with Continuous Action Reinforcement Learning*. Master's Thesis, Tampere University

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. 2013. Playing Atari with deep reinforcement learning. ArXiv:1312.5602.

Montesdeoca, Luis, and Niranjan, Mahesan. 2016. Extending the feature set of a data-driven artificial neural network model of pricing financial options. Pages 1–6 of: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*.

Nakagawa, Kei, Ito, Tomoki, Abe, Masaya, and Izumi, Kiyoshi. 2019. Deep recurrent factor model: interpretable non-linear and time-varying multi-factor Model. ArXiv:1901.11493.

Neyshabur, Behnam. 2017. Implicit regularization in deep learning. ArXiv:1709.01953.

Ni, Hao, Szpruch, Lukasz, Wiese, Magnus, Liao, Shujian, and Xiao, Baoren. 2020. Conditional Sig-Wasserstein GANs for Time Series Generation. ArXiv:2006.05421.

Nigito, Brian. 2021. How to Build an Exchange: Jane Street. `https://www.janestreet.com/tech-talks/building-an-exchange`, [Accessed Jan. 25, 2021].

Ning, Brian, Lin, Franco Ho Ting, and Jaimungal, Sebastian. 2018. Double deep q-learning for optimal execution. ArXiv:1812.06600.

Niranjan, Mahesan. 1996. Sequential tracking in pricing financial options using model based and neural network approaches. *Advances in Neural Information Processing Systems*, **9**, 960–966.

Oblój, Jan. 2008. Fine-tune your smile: Correction to Hagan et al. *Wilmott Magazine*, **35**, 102–104.

Olson, Randal S., La Cava, William, Orzechowski, Patryk, Urbanowicz, Ryan J., and Moore, Jason H. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, **10**(1), 1–13.

Palmer, Richard G., Arthur, W. Brian, Holland, John H., LeBaron, Blake, and Tayler, Paul. 1994. Artificial economic life: a simple model of a stockmarket. *Physica D: Nonlinear Phenomena*, **75**(1–3), 264–274.

Palmer, Samuel. 2019. *Evolutionary Algorithms and Computational Methods for Derivatives Pricing*. PhD thesis, University College London.

Provost, Foster. 2000. Machine learning from imbalanced data sets 101. Pages 1–3 of: *Proc. AAAI Workshop on Imbalanced Data Sets*, vol. 68. AAAI Press.

Puiutta, Erika, and Veith, Eric. 2020. Explainable reinforcement learning: a survey. In *Machine Learning and Knowledge Extraction*, Holzinger, A., Kieseberg, P., Tjoa, A., Weippl, E. (eds). Lecture Notes in Computer Science, vol. 12279.

Rahimian, Hamed, and Mehrotra, Sanjay. 2019. Distributionally robust optimization: A review. ArXiv:1908.05659.

Ras, Gabrielle, Xie, Ning, van Gerven, Marcel, and Doran, Derek. 2022. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, **73**, 329–396.

Recht, Benjamin, Roelofs, Rebecca, Schmidt, Ludwig, and Shankar, Vaishaal. 2018. Do CIFAR-10 classifiers generalize to CIFAR-10? ArXiv:1806.00451.

Ruf, Johannes, and Wang, Weiguan. 2020. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, **24**(1), 1–46.

Ruf, Johannes, and Wang, Weiguan. 2021. Hedging with neural networks. *Journal of Business and Economic Statistics*, DOI: 10.1080/07350015.2021.1931241.

Sabate-Vidales, Marc, Siska, David, and Szpruch, Lukasz. 2018. Unbiased deep solvers for parametric PDEs. ArXiv:1810.05094.

Sabate-Vidales, Marc, Šiška, David, and Szpruch, Lukasz. 2020. Solving path dependent PDEs with LSTM networks and path signatures. ArXiv:2011.10630.

Shu, Tianmin, Xiong, Caiming, and Socher, Richard. 2017. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. ArXiv:1712.07294.

Sirignano, Justin, and Spiliopoulos, Konstantinos. 2018. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, **375**, 1339-1364.

Snow, Derek. 2020. Machine learning in asset management – Part 2: Portfolio construction – weight optimization. *Journal of Financial Data Science*, **2**(2), 17–24.

Soleymani, Farzan, and Paquet, Eric. 2020. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder – DeepBreath. *Expert Systems with Applications*, **156**, 113456.

Sugiyama, Masashi, and Kawanabe, Motoaki. 2012. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press.

Takahashi, Shuntaro, Chen, Yu, and Tanaka-Ishii, Kumiko. 2019. Modeling financial time-series with generative adversarial networks. *Physica A*, **527**, 121261.

Verma, Abhinav, Murali, Vijayaraghavan, Singh, Rishabh, Kohli, Pushmeet, and Chaudhuri, Swarat. 2018. Programmatically interpretable reinforcement learning. Pages 5045–5054 of: Proc. ICML. `https://proceedings.mlr.press/v80/verma18a.html`.

Wan, Zhiqiang, Zhang, Yazhou, and He, Haibo. 2017. Variational autoencoder based synthetic data generation for imbalanced learning. Pages 1–7 of: *2017 IEEE Symposium Series on Computational Intelligence*.

Whalley, A.E., and Wilmott, Paul. 1999. Optimal hedging of options with small but arbitrary transaction cost structure. *European Journal of Applied Mathematics*, **10**(2), 117–139.

Wicker, Matthew, Laurenti, Luca, Patane, Andrea, and Kwiatkowska, Marta. 2020. Probabilistic safety for Bayesian neural networks. Pages 1198–1207 of: *Conference on Uncertainty in Artificial Intelligence*.

Xie, Cihang, Wu, Yuxin, Maaten, Laurens van der, Yuille, Alan L., and He, Kaiming. 2019. Feature denoising for improving adversarial robustness. Pages 501–509 of: *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Xu, Lei, and Veeramachaneni, Kalyan. 2018. Synthesizing tabular data using generative adversarial networks. ArXiv:1811.11264.

Yang, Yongxin, Zheng, Yu, and Hospedales, Timothy. 2017. Gated neural networks for option pricing: Rationality by design. Pages 52-58 of: *Proc. 31st AAAI Conference on Artificial Intelligence*.

Zeng, Yaxiong, and Klabjan, Diego. 2019. Online adaptive machine learning based algorithm for implied volatility surface modeling. *Knowledge-Based Systems*, **163**, 376–391.

Zheng, Yu, Yang, Yongxin, and Chen, Bowei. 2019. Gated deep neural networks for implied volatility surfaces. ArXiv:1904.12834.

Žliobaitė, Indrė, Pechenizkiy, Mykola, and Gama, Joao. 2016. An overview of concept drift applications. Pages 91–114 in: *Big Data Analysis: New Algorithms for a New Society*, Nathalie Japkowicz, Jerzy Stefanowski (eds). Springer.