# The Four Horsemen of Machine Learning in Finance

Matthew Dixon
Department of Applied Mathematics,
Stuart School of Business (Courtesy),
Illinois Institute of Technology
matthew.dixon@iit.edu

Igor Halperin
Fidelity Investments

September 15, 2019

**Abstract**

Machine Learning has been used in the financial services industry for over 40 years, yet it is only in recent years that it has become more pervasive across investment management and trading. Machine learning provides a more general framework for financial modeling than its linear parametric predecessor, generalizing archetypal modeling approaches, such as factor modeling (Feng et al., 2018; Gu et al., 2018; Chen et al., 2019), derivative pricing, portfolio construction, optimal hedging with model-free, data-driven approaches (Halperin, 2017) which are more robust to model risk and capture outliers. Yet despite their demonstrated potential, barriers to adoption have emerged — most of them artifacts of the sociology of this inter-disciplinary field. Based on discussions with several industry experts and the authors' multi-decadal experience using machine learning and traditional quantitative finance at investment banks, asset management and securities trading firms, this position article identifies the major red flags and sets out guidelines and solutions to avoid them. Examples using supervised learning and reinforcement in investment management & trading are provided to illustrate best practices.

## 1 Introduction

The practice of machine learning in finance has grown somewhat commensurately with both theoretical and computational developments in machine learning. Early adopters have been the quantitative hedge funds, including Bridgewater Associates, Renaissance Technologies, Worldquant, D.E. Shaw, and Two Sigma who have embraced novel machine learning techniques although there are mixed degrees of adoption and a healthy skepticism exists that machine learning is a panacea for quantitative trading. In 2015, Bridgewater Associates announced a new artificial intelligence unit, having hired people from IBM Watson with expertise in deep learning. Anthony Ledford, chief scientist at MAN AHL: "It's at an early stage. We have set aside a pot of money for test trading. With deep learning, if all goes well, it will go into test trading, as other machine-learning approaches have'". Winton Capital Management's CEO David Harding: "People started saying, 'There's an

Opinions expressed in this paper are the authors' own, and do not necessarily reflect his or her employer's opinions.

amazing new computing technique that's going to blow away everything that's gone before'. There was also a fashion for genetic algorithms. Well, I can tell you none of those companies exist today — not a sausage of them".

ML in finance as a research area is still nascent, and in important ways incomplete. Nevertheless, some of the earliest and most active users and developers of this technology for business decision making have been in the realm of finance, see for example (Martin, 1977; Trippi and Desieno, 1992; Swanson and White, 1995). Finance has all the ingredients needed to make ML work — vast amounts of data, the computational and human resources, direct profit and loss implications, and a highly competitive environment where every advantage is needed if one is to succeed. But it is also increasingly under scrutiny by regulators and some of the core research activities are resistant to "black-box" methodologies that are typical with "empirical" or "engineering" approaches.

Nevertheless, the growth of machine-readable data to record and communicate activities throughout the financial system combined with persistent growth in computing power and storage capacity has significant implications for every corner of financial modeling. Since the financial crises of 2007-2008, regulatory supervisors have reoriented toward "data-driven" regulation, a prominent example of which is the collection and analysis of detailed contractual terms for the bank loan and trading book stress-testing programs in the United States and Europe, instigated by the crisis (Flood et al., 2016).

"Alternative data" — which refers to data and information outside of the usual scope of securities pricing, company fundamentals, or macroeconomic indicators — is playing an increasingly important role for asset managers, traders, and decision makers. Social media is now ranked as one of the top categories of alternative data currently used by hedge funds. Trading firms are hiring experts in machine learning with the ability to apply Natural Language Processing (NLP) to financial news and other unstructured documents such as earnings announcement reports and SEC 10K reports. Data vendors such as Bloomberg, Thomson Reuters and RavenPack are providing processed news sentiment data tailored for systematic trading models. See Guida (2019) and de Prado (2018) for extensive discussion of machine learning and big data in quantitative investing and trading.

Machines are able to model complex and high-dimensional data generation processes, sweep through millions of model configurations and then robustly evaluate and correct the models in response to new information (Dhar, 2013). By continuously updating and hosting a number of competing models, they prevent any one model leading us into a data gathering silo effective only for that market view. Structurally, one of the by-products of data science is that we have shifted our mindset — the way we reason, experiment, and shape our perspectives from data using ML is cultivating more robust practices to empirically driven trading and investment decision processes.

By nature of the field's obsession with the ultimate thinking machine, it's ironic that the factors contributing to negative outcomes in adopting machine learning in finance should be human. In a field that is engineering heavy and the AI gurus are iconized, it's perhaps not surprising that the contextual challenges of producing successful financial applications in machine learning haven't always materialized. Finance is a unique field in that it borrows much of its innovation from other disciplines, some which applaud open innovation, yet it often operates in different and clandestine ways. Whereas, the academic literature addresses applications of machine learning in finance, offering pointwise solutions to particular challenges, there is little material which approaches the subject of adoption from a holistic perspective.

We begin with a brief overview of machine learning in finance, before identifying the major actors and types of reasoning which has led to serious defects in machine learning applications.

Some of these are not unique to machine learning in finance, indeed history repeats itself with earlier examples in finance which predate machine learning. We shall then set out a principled approach to experiment design which seeks to offset these problems.

## 2 What is Machine Learning in Finance?

Machine learning in finance is a set of learning algorithms and frameworks for financial modeling from data and broadly falls into three branches:

1. *Unsupervised machine learning*: is a data mining technique for partitioning and reducing the dimension of data. Unsupervised learning augments and generalizes statistical approaches to data reduction, such as principal component analysis. An example of unsupervised learning used in finance is K-means clustering for portfolio selection;

2. *Supervised machine learning*: is either a parametric or non-parametric, algorithmic or probabilistic method of learning the relationship between regressors and regressands. Supervised machine learning generalizes statistical techniques such as ordinary least squares (OLS) regression or time series methods such as auto-regressive models. Supervised machine learning assumes that the decisions made by the model are inconsequential to the input. Only a handful of supervised machine learning are suitable for non-stationary data; and

3. *Reinforcement learning*: is a method of stochastic control, with feedback, which learns a policy based on decisions which do change the state of inputs. Reinforcement learning generalizes stochastic dynamic programming and is likely to be the most impactful for trading and investment management. However, because of its complexity, it is the most under-exploited method in finance. Example applications in finance include derivative pricing, optimal hedging, Merton's portfolio problem and optimal trade execution.

The distinction between supervised and unsupervised learning is usually well understood — if the data is not labelled with a response, e.g. loan status, then unsupervised learning is the correct approach to discover patterns and lower dimensional representations of the data. Conversely, the presence of a label suggests supervised learning is more appropriate. This paradigm is strictly for estimating the parameters of a map between input data and an output through minimizing an error over training samples, i.e. fitting a regression over covariates. Performance generalization is achieved through estimating regularization parameters on cross-validation data.

Once the parameters are learned, they are not updated in response to new data. For this reason, supervised learning can be considered as an "offline" form of learning, i.e. the model is calibrated offline. Note that we avoid referring to the model as static since it is possible, under certain types of architectures, to create a dynamical model in which the map between input and output changes over time. For example, a Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU) maintains a set of hidden state variables, much like Markov switching models, which result in a different form of the input-output map over time (Alonso et al., 2018).

In *supervised learning*, a "teacher" provides an exact right output for each data point in a training set. This can be viewed as "feedback" from the teacher, which for supervised learning amounts to informing the agent with the correct label each time the agent classifies a new data point in the training dataset. Note that this is opposite to unsupervised learning, where there is no teacher to provide correct answers to a ML algorithm, which can be viewed as a setting with no teacher, and respectively no feedback from a teacher.

*Reinforcement learning* uses an agent to make a sequence of decisions (referred to as *actions*) depending on the input (referred to as a *state-space*). The key difference of this setting from supervised learning is feedback from the teacher is somewhat in between of the two extremes of unsupervised learning (no feedback at all) and supervised learning, that can be viewed as feedback by providing the right labels. Instead, such partial feedback is provided by "rewards" which encourage a desired behavior, but without explicitly instructing the agent which exact decision it should take, as in supervised learning.

Supervised learning is typically used in applications which traditionally rely on OLS or time series regression, such as fundamental factor modeling and autoregressive modeling. Feedforward and recurrent neural networks, respectively, directly generalize linear regression to non-linear regression and autoregressive models to non-linear autoregressive models. They scale well to high dimensional input data and avoid the colinearity problem in OLS, for example. They are also fitted for out-of-sample performance rather than in-sample performance.

Reinforcement learning is the most suitable paradigm when there is a need to both optimize a utility function and the decisions change the state. For example, a trade execution algorithm should maximize a risk adjusted return and reduce the effect of price impact by splitting large block orders into smaller orders. The problem of how exactly to execute, liquidate, or consume in order to optimize risk adjusted returns, for example, is often the goal of reinforcement learning in financial modeling.

## 2.1 Is Deep Learning in Finance Necessary?

Deep Learning is a type of supervised learning method which involves composition of neural network layers. It generalizes linear regression or logistic linear regression by adding additional layers — allowing any type of non-linear function between input and output to be represented. In fact, a so-called "feedforward" neural network with one hidden layer (a.k.a. shallow learner) has a universal representation theorem[1] which proves that the network can be used for approximating any Borel measureable function between input and output. So why do we need more elaborate types of neural networks including deep learning?

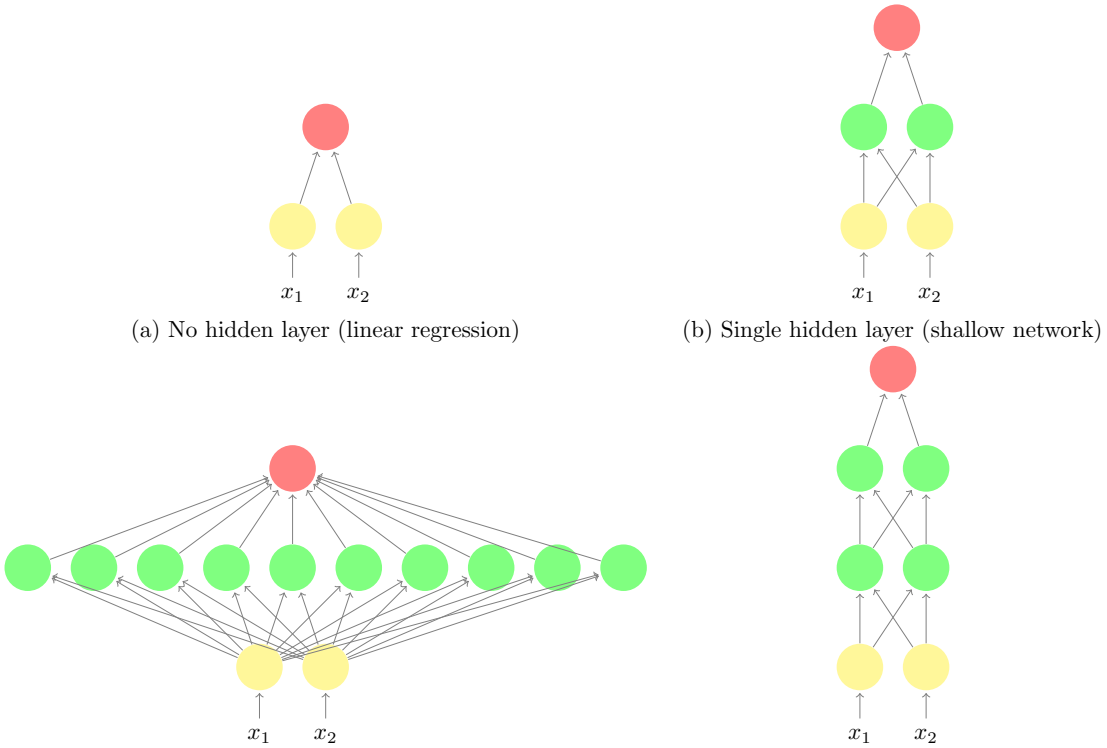
Outside of natural language processing or text mining, it turns out that in most use cases in finance, deep learning is not strictly necessary but is more of a convenience, allowing different graphs to be chained together through back-propagation. In certain applications, especially with high dimensional inputs and strong non-linearity, it might be advantageous to use deep learning versus a "fanned out" shallow network (Sirignano et al., 2016; Dixon et al., 2016; Gu et al., 2018). However the advantage is primarily efficiency — fewer parameters might be needed, resulting in faster computations[2]. Deep networks are in fact interpretable[3] — the sensitivity of the output to each input variable can be determined, so the notion that they are black-boxes is too simplistic (Dixon and Polson, 2019). Deep learning has become synonymous with neural networks and standard tools for training neural networks such as `PyTorch` and `TensorFlow` are designed for deep networks.

In reality, most of the neural networks which are used in finance needn't be deep and the hype

[1]The Universal representation theorem suggests that we only need one hidden layer (Kolmogorov, 1957; Arnold, 1957).

[2]Deep networks are implicitly self-regularizing (Martin and Mahoney, 2018), meaning that fewer parameters are needed than a shallow network to obtain the out-of-sample performance on the same data.

[3]We should add the caveat that the weights and biases do not carry individual meaning, as in say linear regression, rather algebraic expressions can be written for the Jacobian and interaction effects which can be used to rank the feature importance.




(a) No hidden layer (linear regression)

(b) Single hidden layer (shallow network)

(c) Fanned-out hidden layer (shallow network)

(d) Two hidden layers (deep network)

Figure 1: *Simple two variable feedforward networks with and without hidden layers. The yellow nodes denote input variables, the green nodes denote hidden units and the red nodes are outputs. A feedforward network without hidden layers is a linear regressor. A feedforward network with one hidden layer is a* shallow learner *and a feedforward network with two or more hidden layers is* deep learner*.*

is not grounded in reality. To be clear, while the notion that deep learning is synonymous with neural networks is approximately valid, it is not true that deep learning is synonymous with AI. AI represents a far greater universe of algorithms and methods, many of which have yet to make their debut in finance.

However, the use of deep learning for financial modeling from market data is not all negative. As it turns out, there is a silver lining. Aside from LSTMs and GRUs, the deep learning candidate which stands to be very relevant for financial time series prediction has yet to be firmly put to the test on financial data. A certain type of neural network, referred to as a convolutional neural network, is capable of capturing patterns at different time scales when applied to financial times series (Borovykh et al., 2017; van den Oord et al., 2016). Referred to as a *dilated convolutional neural network*, this architecture can learn autocorrelation structures at different frequencies, for example, daily, weekly, monthly, annual historical patterns. Each layer captures a different time scale, much like composing auto-regressive models at different frequencies.

The field of machine learning in finance has suffered from being too easily enamoured by inno-

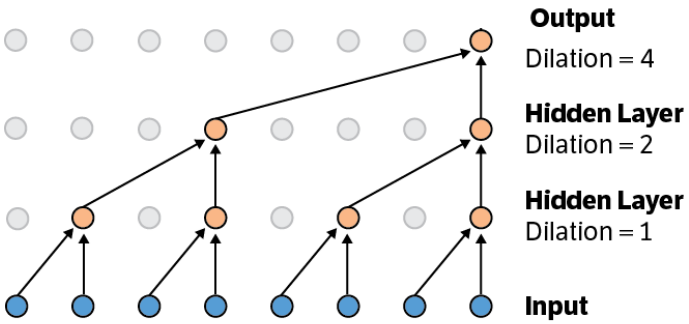



Figure 2: *A dilated convolutional neural network with three layers. The receptive field is given by* $r = 8$*, i.e. one output value is influenced by eight input neurons. Source van den Oord et al. (2016).*

vations from AI research — deep learning is one example where simply borrowing ideas directly from AI researchers has led to square pegs in round holes. In our view, what is needed is a broad awareness of the human factors leading to these adoption problems and a framework, or at least guidelines, for how to cast AI research innovation into the field of financial modeling. The remainder of this article sets out to explore this further.

# 3 The Four Horsemen of Machine Learning in Finance

Despite an impressive array of recent successful applications of supervised machine learning (Dixon et al., 2016; Heaton, Polson, and Witte, Heaton et al.; Bayer and Stemper, 2018; Feng et al., 2018) and reinforcement learning (Halperin, 2017; Bühler et al., 2018) in financial modeling, there is an equally if not larger crowd of anecdotal stories of machine learning leading to fund failure, poor performance on financial data, or even seed research projects failing to make the light of day. While problem areas requiring natural language processing and text mining fall heavily into well established engineering practices, financial modeling requires a blend of engineering and financial modeling experience which is still unchartered territory. Based on our experience and communications with industry experts, we dissect some of the common problems which plague the adoption of machine learning in financial institutions and introduce the following four horsemen:

- *The Tinkerer*: Armed with a Swiss army knife, the tinkerer is statistically illiterate and has an inability and patent reluctance to co-integrate their discipline with financial time series analysis, financial modeling and econometrics, and dynamic programming. Some of this inability appears to stem from habitual success in applying machine learning to simple pattern recognition problems outside of finance, often accessible through Kaggle. The end result is a Rube-Goldberg machine which can not produce defensible, interpretable results reconciled against a more established methodology. Moreover, changing incomprehensible hyper-parameters substantially changes the result and, at best, heuristics are used to stabilize the result.

- *The Historian*: The historian posits that the world will continue to be as it always was in the past. In making predictions, a common mistake is to assume covariance stationarity of

the data. Methods for time series prediction such as Recurrent Neural Networks can only be used reliably if the data is autocorrelated and covariance stationary[4]. Any attempt to arrive at high predictive performance using almost all supervised machine learning methods on non-stationary data is likely futile. Only Gated Recurrent Units (or closely related LSTMs) and change point detection based machine learning methods are suitable for non-stationary data.

- *The Miner*: The miner operates myopically on the premise that they are one-step away from striking gold without considering the wider context of competing in a market with other participants. Once the model generates a relatively high accuracy signal for an algo-trading strategy, it is then tested against the market or back-tested, only to deliver mediocre P&L. The miner then exhaustively applies every machine learning classifier or regression technique available in their arsenal of tools at their disposal, before finally giving up and reluctantly concluding that strategy performance using supervised machine learning faired no better than linear regression. It is well known by experts in algorithmic trading that latency, slippage, price impact and trade execution perform an important role in strategy performance and reinforcement learning is the correct paradigm to account for these effects. In general, highly accurate supervised learning pattern recognition methods do not necessarily translate into P&L maximizing signals.

- *The Puritan*: The puritan operates on strictly rationalistic terms and wants every detail to be meticulously documented and scrutinized — the results must match their expectations and beliefs before they are willing to advocate machine learning, only then as a tool for automation. The puritan is incapable of accepting that machine learning can discover new and meaningful patterns, not consistent with well established theory and stylized facts inscribed in the canons of finance. In doing so, they are overlooking some of the most important advantages of machine learning as a tool for capturing outliers and non-linear effects.

Each horsemen embodies a pathological line of reasoning when taken to an extremity. Of course, the mindsets for innovation and risk management can be diametrically opposite, and a degree of experimentation and healthy skepticism is an artifact of research. In our experience, some simple guidelines can pave the way to more successful adoption of machine learning for financial modeling.

# 4 Principles for Effective Machine Learning in Finance

With the large caveat that there is no substitute for critical human reasoning skills, knowledge of machine learning theory, expertise in the application area, sufficient and reliable data[5] and technical resources to process the data and apply machine learning, some guidelines for improving the success of machine learning in investment management are now described:

1. *Problem definition:* Define the problem that is being solved. Consider carefully whether the application of machine learning rests squarely in data snooping — with trading strategy and execution treated separately — or whether the problem has a scheduling component and requires reinforcement learning. Further determine whether it is computationally feasible to analyze the full set of data, or whether data reduction (unsupervised learning), is first needed.

[4]The only case when this may not be true is if the modeler is willing to retrain the model on a frequent basis which may not be practical and result in stability challenges.

[5]The data should also have a sufficiently high signal-to-noise ratio.

2. *Modeling assumptions:* State the main assumptions of the modeling approach, does the machine learning method assume that the observations are identical and independently distributed? If the data is a time series, does the model assume covariance stationarity? Apply basic statistical tests, as listed in Table 2 to characterize the data.

3. *Develop intuition:* Establish a toy model, a subset of the full problem, in order to gain comfort with machine learning in a setting where less can go wrong. In the case of neural networks, for example, we might establish a baseline with the linear version of the model, i.e. when the neural network is functionally equivalent to linear regression[6];

4. *Defensible results:* Design the experiment so that the model output can be explained and is defensible by reduction ad absurdum. This doesn't mean the model should be statistically interpretable, but rather any mis-classifications or large errors should be accountable. In many cases, it may be necessary to revert to the toy model to unfold the mystery.

5. *Diagnostics:* Employ several diagnostics, both from machine learning and statistics, to characterize the data and the bias-variance and model significance compared to a baseline. Appendix A contains a brief summary of some of the most important statistical tests useful for machine learning in finance;

6. *Keep it simple:* Minimize the use of elaborate algorithms and widgets, in favor of approaches which rely on as few parameters and are as transparent as possible;

7. *Choice of utility function:* Ahead of analyzing performance of specific algorithms, review the appropriateness of your assumption about a loss function you use to train you machine learning algorithms. For example, is the ordinary least squares loss function appropriate when analyzing analysts' forecasts of companies earnings?;

8. *Solution constraints:* Analyze all possible prior views or constraints that the expected solution should satisfy (e.g. price positively, sparsity, convexity, no-arbitrage constraints etc.), and enforce these prior views on the solution, either as hard constraints, or as soft constraints (regularization);

9. *Is your data biased?* In many cases, datasets available for training machine learning algorithms are mis-balanced. For example, the very event of interest is a black-swan. A mis-balance of data may propagate into biases of your predictive model, which is known in statistics as the "inspection paradox". Depending on the planned application of your algorithm, such effects may or may not be potentially harmful for you - if your future data is expected to keep mis-balances of your training data, you can ignore it, but otherwise you have to make corrections;

10. *Feature engineering:* If possible, avoid the temptation to prematurely feature engineer, switch between various machine learning models, without first assessing the signal in a back-testing or live simulation environment. In particular, establish a strategy performance range, by assessing what the best and worse case P&L is using the perfect signal (or in-sample signal) and white noise respectively. Attribute how the strategy loses and makes money based on the

[6]One advantage of neural networks is that the activation functions can be switched off so that the network functionally becomes a linear regression, which is easier to understand. There is a potential degeneracy in this case; There may exist "flat directions"— hyper-surfaces in the parameter space that have exactly the same loss function.

signal, before spending valuable time on feature engineering. If the signal is misconfigured for the strategy and market, there is no guarantee that feature engineering will improve strategy performance.

# 5 Examples of Best Practices in Machine Learning in Finance

While there are countless examples of machine learning in financial modeling, we focus on just a few of the most archetypal problems, each of which illustrates, in different but important ways, the guidelines for best practice in the previous section.

## 5.1 Factor modeling with Deep Learning

Linear cross-sectional factor models, such as (Fama and MacBeth, 1973), FF(3) and FF(5) (Fama and French, 1993, 1992, 2015) and BARRA factor models (see Rosenberg and Marathe (1976); Carvalho et al. (2012)) are appealing because of their simplicity and their economic interpretability, generating tradable portfolios. Factor realizations are estimated in the BARRA model by generalized least squares regression.

However, least squares linear regression can have poor expressability and relies on independent Gaussian errors, which motivates considering *non-linear* models. Yet, introducing non-linearities and incorporating interaction effects may be a harder task. Asset managers seek novel predictive firm characteristics to explain anomalies which are not captured by classical capital asset pricing and factor models. Recently a number of independent empirical studies, rooted in a data science approach, have shown the importance of using a higher number of economically interpretable predictors related to firm characteristics and other common factors Moritz and Zimmermann (2016); Harvey et al. (2015); Gu et al. (2018); Feng et al. (2018). Gu et al. (2018) analyze a dataset of more than 30,000 individual stocks over a 60 year period from 1957 to 2016, and determine over 900 baseline signals. Both Moritz and Zimmermann (2016) and Gu et al. (2018) highlight the inadequacies of OLS regression in variable selection over high dimensional datasets.

There are several recent articles in the literature which find evidence of superior performance of non-linear regression techniques for fundamental factor models such as regression trees and neural networks. (Gu et al., 2018; Feng et al., 2018; Dixon and Polson, 2019). The nuances of deep versus shallow architecture should not over-shadow the importance of having a non-linear regression. There are many techniques for non-linear regression, but neural networks are provably representative of all non-linear functions. The same is not true for trees, unless they are infinitely deep.

In the following neural network for fundamental factors from Dixon and Polson (2019), we demonstrate a recommended practice for adopting ML. The details of the experiments are incomplete and there are other tests and experiments which should be performed such as tuning the amount of regularization and viewing the learning curves to understand the effect of sample size on the bias-variance trade-off. Moreover, the results are applied to a simple data set of six fundamental factors only —however the principal of robust experimental design remains valid here.

**Problem formulation:** We follow the BARRA approach — the inputs to the feedforward network are the monthly reported factor loadings for each (asset, time) pair and the output, here, is

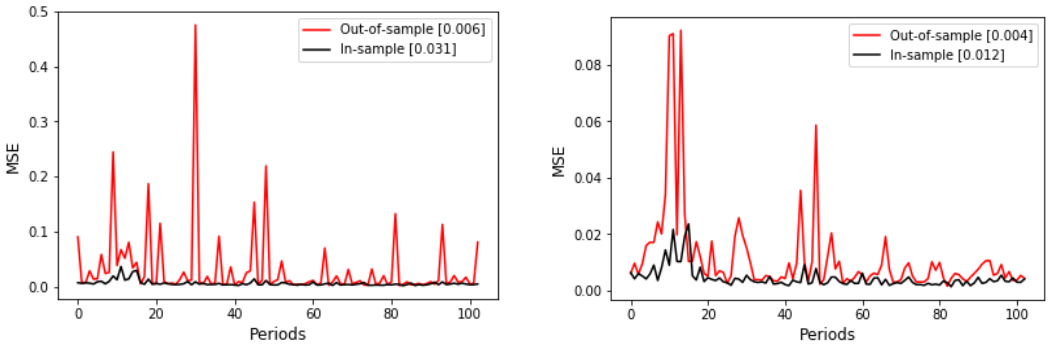
a 1 month return[7]. The problem is to find the non-linear map between the factor loadings and the asset returns across the universe of stocks for a particular historical period. In linear regression, the sensitivities of the model to each factor correspond to the factor realizations. We shall see how this concept generalizes to neural networks (shallow or deep) too.

More precisely, we use neural networks to find a map between the factor loadings and the asset returns. If there are $K$ fundamental factors and $N$ assets then the input data is the known time invariant factor loadings (betas): where $B = [\mathbf{1} \mid \boldsymbol{\beta}_1 \mid \cdots \mid \boldsymbol{\beta}_K]$ is the $N \times K + 1$ matrix of known factor loadings (betas): $\beta_{i,k} := (\boldsymbol{\beta}_k)_i$ is the exposure of asset $i$ to factor $k$ at time $t$. The output are the $N$ asset returns, $\mathbf{r}_t$, at time $t$. The problem is how to fit a neural network to predict the returns if the noise is assumed $i.i.d.$ :

$$\mathbf{r}_t = F_t(B) + \epsilon_t. \tag{1}$$

For each monthly period, the network is trained as a cross-sectional regression model over all assets in the chosen universe — in this case, 218 assets from the S&P 500 index. For each historical period, the neural network is re-trained and the performance with the realized asset returns are compared.

**Establishing a baseline:** To assess the strength of a neural network for this problem, we demonstrate how we establish a neural network baseline with OLS regression and we compare the P&L of a simple stock selection strategy using the ML signal with a white noise signal. Note, if the neural network had underperformed OLS regression, then we simply troubleshoot by turning off the activation functions (i.e. use linear activation) and check that the results are, on average, comparable[8]




(a) OLS Regression (b) FFWD Neural Network

Figure 3: *This figures compares the out-of-sample performance of an OLS estimator with a feedforward neural network, as measured by the mean squared error (MSE).*

**Interpretability:** Neural networks are, in fact, interpretable — the sensitivities of the output to the inputs, and the interaction terms, can be evaluated analytically. Figure 4 shows the sensitivities

[7] A 3-month return is more commonly used in practice.

[8] Recall that a neural network, shallow or deep, with linear activation of the hidden units is a linear regression.

of the OLS regression model and neural network models to each of six fundamental factors over a 48 month period from November 2014 to November 2018. The black line shows the sensitivities evaluated by our neural network based method. The red line shows the same sensitivities using linear regression.

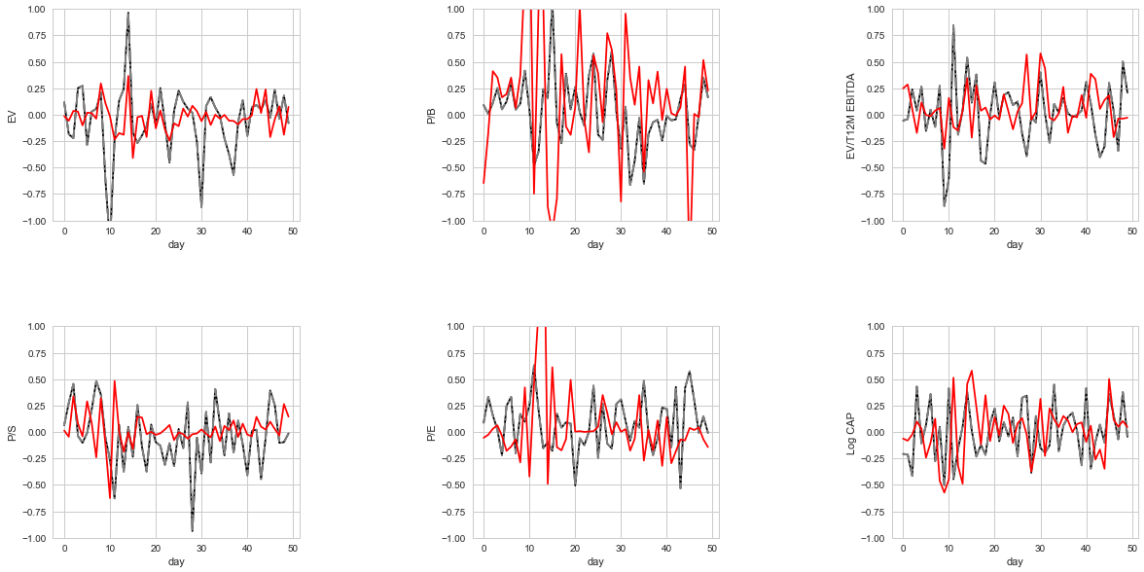



Figure 4: *This figure shows the sensitivities of the model to each factor over a 48 month period from November 2014 to November 2018. The black line shows the sensitivities evaluated by our neural network based method. The red line shows the same sensitivities using linear regression. Note, for ease of comparison across factors, the scaling of the plot has been adjusted so that the y-axis range is fixed.*

Figure 5 compares the distribution of sensitivities to each factor over a 48 month period from November 2014 to November 2018 as estimated by neural networks and OLS regression.

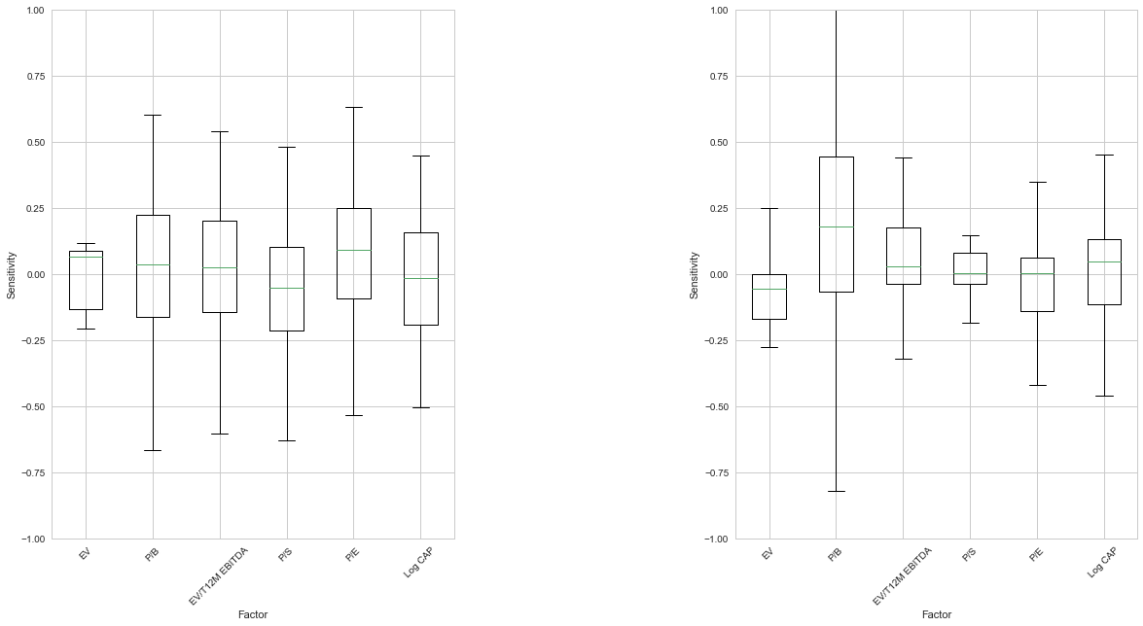
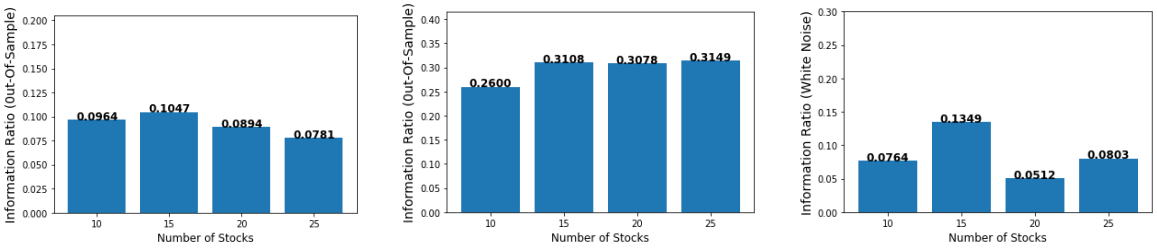



Figure 5: *The distribution of sensitivities to each factor over a 48 month period from November 2014 to November 2018 as estimated by neural network (left). The same sensitivities using generalized linear regression (right). Note, for ease of comparison across factors, the scaling of the plot has been adjusted so that the y-axis range is fixed.*

**Strategy performance:** Finally, although not an extensive a result to warrant a conclusion, we provide evidence that our neural network factor model generates higher information ratios than the linear factor model when used to sort portfolios from our universe. Figure 6 shows the information ratios of a portfolio selection strategy which selects the $n$ stocks with the highest predicted monthly returns. The information ratios are evaluated for various size portfolios, using the S&P 500 index as the benchmark.

The information ratios are observed to increase by a factor of approximately three. Also shown, for control, are randomly selected portfolios[9]. We observe in some cases that even though the factors in the linear regression barely exceed white noise, they are indeed predictive when used in a neural network model. Even for this limited set of factors, one can produce positive information ratios with neural networks.

[9]Note that the reason why the information ratios are always positive, even when under random portfolio selection, is because we have defined a universe of only 218 stocks, with sufficient historical data available over the period from November 2014 to November 2018.




(a) OLS Regression (b) FFWD Neural Network (c) White Noise

Figure 6: *This figure shows the information ratios of a portfolio selection strategy which selects the n stocks from the universe with the highest predicted monthly returns. The information ratios are evaluated for various size portfolios. The information ratio are based on out-of-sample predicted asset returns using (left) generalized linear regression; (center) neural networks; and (right) randomized with no predictive model.*

The take-away from this first example application is that the principles of sound experiment design are just as important as the machine learning method. To simply try an out-of-the-box neural network and conclude that it "doesn't work", should be supported by a trail of evidence of out-of-sample performance. Since a neural network with linear activations of hidden units is OLS regression, one should first establish comparable results with a standard OLS regression tool. Only then, revert to non-linear activated units and tune the $L_1$ regularization to obtain generalization performance on out-of-sample predictions. In general, we should suspect that neural networks will outperform linear regression, but this is only likely with sufficient training data and careful treatment of the bias-variance tradeoff with cross-validation. Although not demonstrated here, learning curves can be used to establish the effect of increasing the training sample size on the bias-variance tradeoff (see, for example, Dixon (2018)).

You may have noticed that in the above experiment, the experimental results had a time component. Strictly speaking, the factor loadings are both cross-sectional and temporal (i.e. panel data). However, the prediction horizon is typically quarterly and the loading are updated on a monthly basis. For this reason, we can expect to retrain the model each month and dynamical aspect of the model can be ignored. Of course, lagged factor loading may be important for out-of-sample performance and time series models, such as recurrent neural networks, are useful.

## 5.2 Prediction with RNNs

Prediction of asset prices from time series has traditionally been the domain of financial econometrics. Recall that because plain RNNs generalize vector auto-regressive models, we can use them for prediction from historical inputs, either endogenous or exogenous. Moreover, dynamic RNNs such as GRUs and LSTMs, can be used when the data is non-stationary, providing a dynamic auto-regressive structure.

**Problem formulation:** We consider the problem of how to predict the one-step ahead mid-prices from equi-distant timestamps of historical mid prices of Bitcoin. There are many reasons of course to use alternative input data, especially at a higher frequency, but the purpose of this exercise is

to illustrate the main challenges and guidelines for prediction from time series data with machine learning.

**Statistic tests:** To apply RNNs to univariate time series modeling, we first establish whether the time series is auto-covariance stationary. Using 199918 samples of 1 minute snapshots of Bitcoin mid-prices in USD, we find the augmented Dickey-Fuller (ADF) test statistic is $-2.0945$, above the $10^{th}$ percentile ($-2.5667$) with a p-value of 0.26. Since the test statistic is for a null hypothesis that the data is not stationary, we can not reject it. In this case, the test incorporates the effect of a constant and trend term.

Sometimes, the time series can be transformed by taking the difference or detrending it. While there are advantages to this transformation, it is far from full-proof and can lead to loss of predictive signal. Here, we shall simply proceed with prediction on the original time series, but caution the reader to explore differencing or detrending.

**Model identification:** Machine learning is finance hasn't replaced the classical Box-Jenkins approach. Rather it builds on it, using (i) model identification; (ii) fitting; and (iii) diagnostics to arrive at the "best" model, where out-of-sample performance becomes the ultimate measure of success. Model identification can be challenging for time series —- one can either compare the loss across varying maximum lags ("sequence length") in a RNN because it is a special type of AR model whose number of parameters (i.e. weights and biases) do not increase with the sequence length.

This approach, while full-proof, can be quite computationally intensive and we can use the estimated partial autocorrelation function (PACF) to identify the number of lags to include in the model.

Figure 7 shows the estimated partial correlation function (PACF) for minute snap-shots of Bitcoin mid-prices. The signature of a plain RNN is to exhibit a zero PACF once the lag exceeds the sequence length. In other words, it has a cut-off after p-lags if the sequence length is p. With the caveat that the data is not stationary, so the PACF will vary in time, we view the estimated PACF to identify a sample average model order. From the plot, we would start our initial model training using a lag of 4 or 5. This is favored to guessing the sequence length, or adding yet another hyper-parameter to the tuning phase, and avoids specifying excessive sequence lengths which result in longer training times.

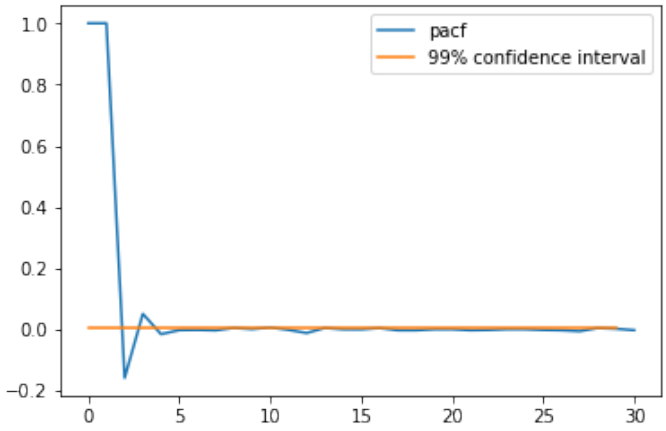


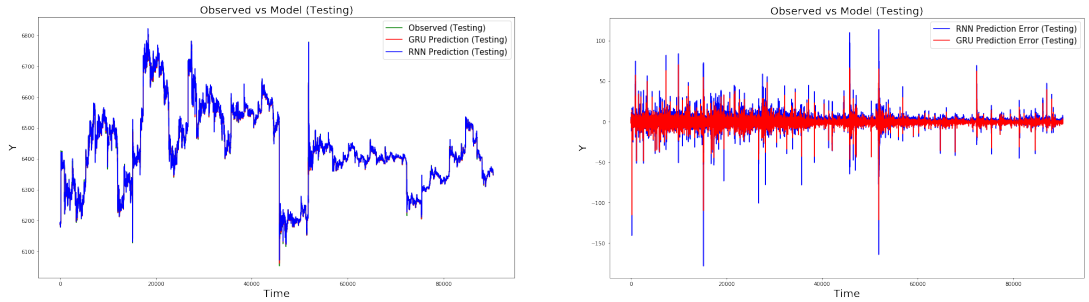

Figure 7: *This figure shows the estimated partial correlation function (PACF) for minute snap-shot Bitcoin mid-prices.*

**Results:** We perform next minute prediction over a test-sample of 10,000 samples[10] as shown in Figure 8(a) using a RNN and a GRU. Note that the model is not re-trained over the test period, only the training data immediately prior to this test period is used by the model for training.

As expected, the out-of-sample error is lower for the GRU (see Figure 8(b)), which does not assume stationarity of the data. While the errors might seem insignificant, we point out that the y-scale is in USD. We also apply a Mariano-Diebold test to determine that the difference in the two models is statistically significant at the 99% level.

While, again, the details of the above experiment are not complete, it serves to illustrate the importance of adopting statistical tests to guide experiment and model design. There are further statistical tests that should be performed, such as the Ljung-Box test on the out-of-sample residuals to determine whether the model is under-fitting by not capturing all of the time series structure.

[10]To avoid look-ahead bias, the test-set is set in the future of the training set.




(a) One step-ahead Prediction with RNNs (b) One step-ahead Prediction Error with RNNs

Figure 8: *(left) This figure compares the next minute plain RNN and GRU predictions with a test-sample of 10,000 Bitcoin mid-prices. (Right) As expected the out-of-sample performance is lower for the GRU than the plain RNN – the latter assumes stationarity.*

We do not develop this example further here, as we did for the previous example, by backtesting a trading strategy using the model prediction as a signal. But this of course would be a necessary step before productionizing a prediction model for alpha generation. More often that not, an immediate challenge that must be addressed before such a strategy is successful are trade execution concerns, such as slippage and, for larger trades in less liquid markets, price impact.

The study of price impact warrants an entirely different paradigm of machine learning — rather than offline learning[11], in which the model must be fitted offline, we use online learning; The model is a RL agent which is capable of updating its optimal policy as new information arrives and new actions are taken which affect future information.

In the next example, we consider the execution problem of how to break-up block sell orders to maximize risk adjusted returns. In the example, we simulate the stock price rather than predict it and try to determine the best approach as a scheduling problem, contingent on the stock price. The problem formulation is therefore more challenging and it takes more skill to apply off-the-shelf machine learning to such problems.

## 5.3 Optimal stock execution with Reinforcement Learning

Assume that a broker has to sell $N$ blocks of shares with $n$ shares in each block, e.g. we can have N = 10, n = 1000. The problem is to how to break up the $N$ blocks to maximize some reward as illustrated in Figure 9.

In applying reinforcement learning, we place the emphasis on defining the (i) state— often an observed variable; (ii) the actions, which are the state dependent decisions made by an agent; and (iii) the rewards for the decisions.

The state of the inventory at time $t$ is then given by the variable $X_t$ taking values in a set $\mathcal{X}$ with $N = 10$ states $X^{(n)}$, so that the starting point at $t = 0$ is $X_0 = X^{(N-1)}$ and the target state is $X_T = X^{(0)} = 0$, i.e. the broker seeks to sell the entire inventory. In each step, the agent has four possible actions $a_t = a^{(i)}$ that measure the number of blocks of shares sold at time $t$ where $a^{(0)} = 0$ stands for no action, and $a^{(i)} = i$ with $i = 1, \ldots, 3$ is the number of blocks sold. The

[11]The exception to this rule are Bayesian supervised learning methods, such as change point detection, which can update the posterior in response to new data.

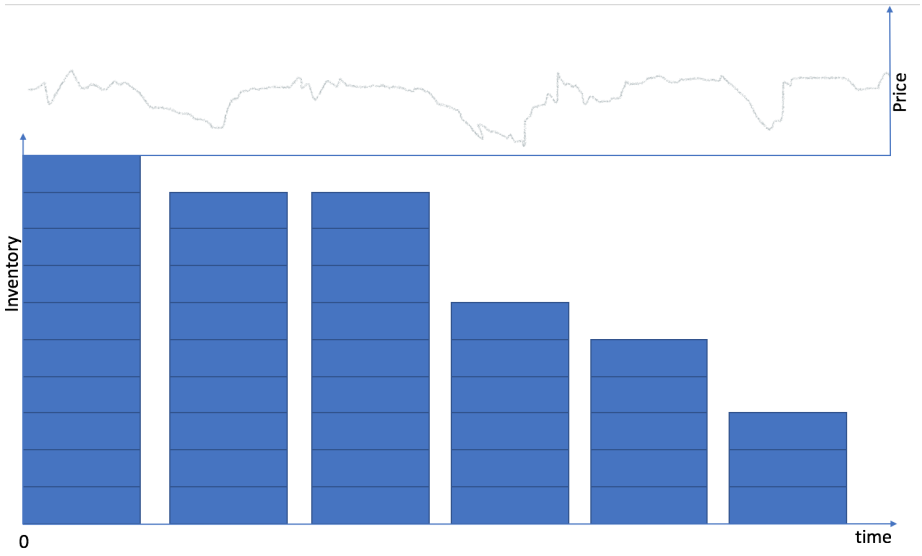



Figure 9: *The optimal execution problem: how to break up large market orders into smaller orders with lower market impact. We choose the state space to be the inventory, shown by the number of blocks, stock price and time. In this illustration, the agent decides whether to sell* $\{0, 1, 2, 3\}$ *blocks at each time step. The problem is thus whether to retain inventory, increasing market risk but reducing the market impact, or quickly sell inventory to reduce exposure but increase the market impact.*

update equation is

$$X_{t+1} = (X_t - a_t)_+ . \tag{2}$$

We assume that trades influence the stock price dynamics through a linear market impact

$$S_{t+1} = S_t e^{(1-\nu a_t)} + \sigma S_t Z_t, \tag{3}$$

where $\nu$ is a market friction parameter. To map onto a finite Markov Decision Process problem, a range of possible stock prices $\mathcal{S}$ can be discretized to $M$ values, e.g. $M = 12$. This makes it possible to visualize the optimal policy in tabulated form — convenient for interpreting the output[12]. The state space of the problem is dimension $N \times M = 10 \cdot 12 = 120$. The dimension of the extended space including the time is then $120 \cdot 10 = 1200$.

The payoff of selling $a_t$ blocks of shares when the stock price is $S_t$ is $na_tS_t$. A risk-adjusted payoff adds a penalty on variance of the remaining inventory price at the next step $t + 1$: $r_t = na_tS_t - \lambda n \mathrm{Var}\left[S_{t+1}X_{t+1}\right]$. All combinations of state and time can then be represented as a three-dimensional grid of size $N \times M \times T = 10 \cdot 12 \cdot 10$. At each time step, a time-dependent optimal policy is therefore found with $10 \times 12$ (for inventory and stock price level) states and four possible actions $a_t = \{a_0, a_1, a_2, a_3\}$ can be viewed as a $10 \times 12$ matrix as shown, for the second time step, in Table 1.

[12] Additionally, when the state space is discrete, we can use simple RL methods such as SARSA and Q-learning, as opposed to more complex methods such as policy gradient or continuous-state Q-learning, with a function approximation (e.g. deep Q-learning).

We can now apply SARSA to learn the solution to the price impact problem in such a simplified setting. For exploration needed for on-line learning, one can use a $\varepsilon$-greedy policy. The $\varepsilon$-greedy policy is a simple stochastic policy where the agent takes an action that maximizes the action-value function with probability $1-\varepsilon$, and takes a purely random action with probability $\varepsilon$.

The $\varepsilon$-greedy policy is used to produce both actions $a, a'$ in the SARSA update, while with Q-learning it is only used to pick the action at the current step. For sufficiently small $\alpha$ and under tapering of $\varepsilon$ (see Figure 10), both methods are shown by Figure 11 to converge to the same cumulative reward.

The optimal policy, at time step $t = 2$, for the trade execution problem using SARSA. The rows denote the inventory level and the columns denote the stock price level. Each element denotes an action to sell $\{0, 1, 2, 3\}$ blocks of shares. Note that a different tabular matrix is available for each time step, but is not shown here for brevity of exposition.

| $t = 2$ | Price level | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Inventory | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 3 | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 3 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 1: *The optimal policy, at time step $t = 2$, for the trade execution problem using SARSA. The rows denote the inventory level and the columns denote the stock price level. Each element denotes an action to sell $\{0, 1, 2, 3\}$ blocks of shares.*

While the problem formulation is simpler than needed for production grade applications, our example served to illustrate the importance of interpretability and the ability to compare different methods, in this case Q-learning and SARSA, which are known to converge as $\varepsilon$ tends to zero. In general, Q-learning is favored as it is off-policy and does not assume that the optimal policy generated the data.

The above example offers a simple illustration of discrete-state and discrete-action reinforcement learning. For many problems, thinking in terms of continuous states and/or actions may be preferred. For such cases, one should use function approximations to represent the Q-function or/and an action policy. As is shown in Halperin (2017); Halperin (2019), continuous-state Q-learning can be used to provide a model-independent way of option pricing that, in addition to mitigating model risk, also improves on the classical Black-Scholes model (and other "risk-neutral" models) by tackling mis-hedging risk in discrete time[13]. When a simple quadratic Markowitz-type utility function is used for this setting, linear function approximation for the Q-function suffices to

[13]This risk disappears from the original BS model, and all "risk-neutral" models, due to the fact that these models are formulated in continuous time by construction, rather than obtaining the continuous-time limit as an approximation to discrete-time dynamics.

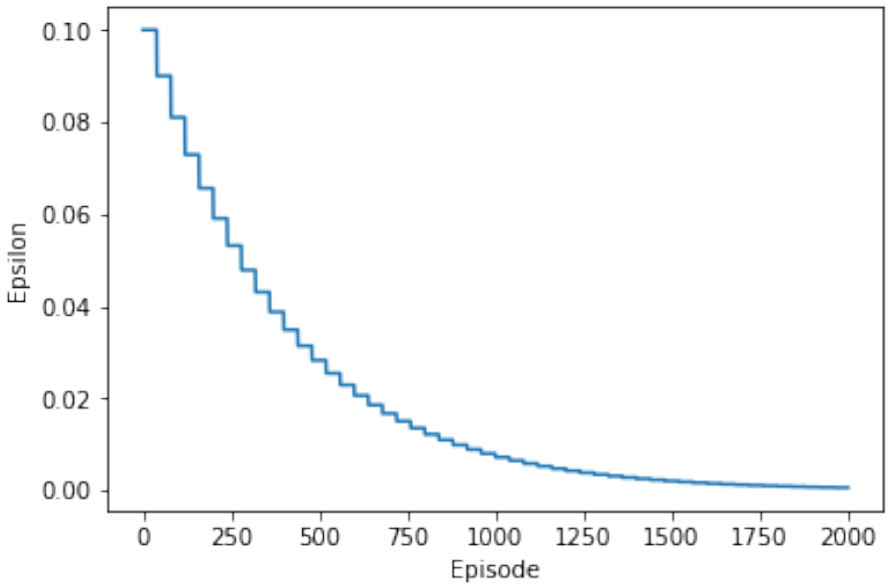



Figure 10: *This figure illustrates how $\varepsilon$ is tapered in the price impact problem problem with increasing episodes so that Q-learning and SARSA converge to the same optimal policy and cumulative reward.*

obtain both optimal option price and optimal hedge. For more complex utility functions, e.g. a CVaR utility, one may use more complex function approximations such as neural networks. This amounts to deep reinforcement learning (e.g. deep Q-learning) that combines supervised learning and reinforcement learning. Extensions to portfolio constructions with market impact effects using an entropy-regularized version of Q-learning called G-learning are described in (Halperin and Feldshteyn, 2018). For applications of RL to other classical financial problems such as wealth management and financial planning, see the forthcoming book (Bilokon et al., 2020).

Other interesting directions are offered by Inverse Reinforcement Learning (IRL) whose objective is to find a reward function of an agent given its observed behavior. In addition to applications for learning reward (utility) functions of specific traders, IRL can also be applied to a market-wide "agent" that embodies the "Invisible Hand" of the market, i.e. IRL can be used to model the dynamics of a market as a whole (Halperin and Feldshteyn, 2018). Interestingly, a model of market dynamics obtained in this way implicitly assumes that the market is in a benign regime, where corporate defaults or crises do not exist. Using methods from statistical and quantum physics, Halperin and Dixon (2018) extended the resulting model to include also market crashes and corporate defaults - phenomena that are both very important, yet absent in standard models for equity markets (unless one introduces special degrees of freedom such as stochastic credit spreads)[14].

We note that in a sense, this last work goes 'beyond IRL' (or beyond ML in general, for that sake) because it shows the importance of *rare* data (or even unobserved data!) in construction of a right dynamic model of the environment. While pure ML algorithms are driven by data, constraints

[14] An introductory exposition on these topics, along with a general introduction on reinforcement learning, with examples from option pricing and portfolio optimization, can be found in a Coursera specialization (Halperin, Halperin).

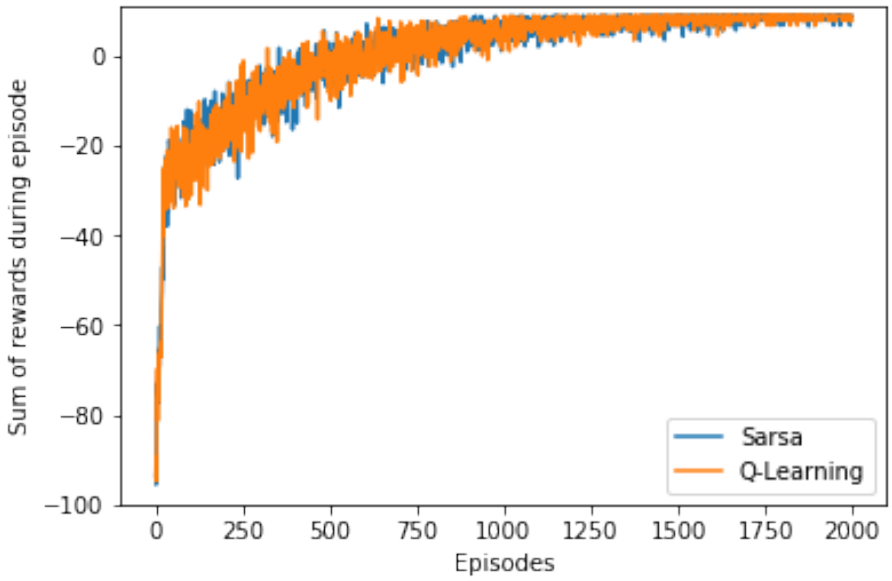



Figure 11: *Q-learning and SARSA are observed to converge to the same optimal policy and cumulative reward in the price impact problem under the $\varepsilon$-tapering schedule shown in Figure 10. Note that the cumulative rewards are averaged over twenty simulations.*

or beliefs about the "true" nature of the world can be enforced by using appropriate regularization penalties - that can themselves be based on models, see Halperin and Dixon (2018) for one example.

To summarize, for problems of sequential decision-making with a possible feedback loop from actions of the agent, reinforcement learning algorithms often provide the most natural and transparent approach. The ability to choose the state space, actions, rewards and exploration tapering schedule, and produce defensible results are the most important considerations.

Our general recommendation is to avoid hastily applying deep reinforcement learning ahead of first using "shallow" reinforcement learning — establishing, at a minimum, a benchmark for more advanced techniques such as deep Q-learning.

# 6 Summary

Machine learning provides a more general framework for financial modeling than its linear parametric predecessors, generalizing archetypal modeling approaches, such as factor modeling, derivative pricing, portfolio construction and consumption, optimal hedging with model-free, data-driven approaches which are more robust to model risk and capture outliers.

In this short position article, we evaluated some of the barriers to adoption which have emerged – most of them artifacts of the sociology of this inter-disciplinary field. We identify the major red flags and set out guidelines and solutions to avoid them. Examples using supervised learning and reinforcement in investment management and trading are provided to illustrate best practices in successful application of machine learning.

Beyond applications in NLP and text mining, machine learning does have a more central role to play in financial modeling. The key to successful adoption is marrying new with old, casting machine learning into financial modeling and econometrics frameworks which generalize their classical counterparts and therefore inherit the interpretability and robustness which is necessary for machine learning in finance to be successful.

# References


Alonso, M. N., G. Batres-Estrada, and A. Moulin (2018). *Deep Learning in Finance: Prediction of Stock Returns with Long Short-Term Memory Networks*, Chapter 13, pp. 251–277. John Wiley Sons, Ltd.

Arnold, V. I. (1957). On functions of three variables. Volume 114, pp. 679–681.

Bayer, C. and B. Stemper (2018, Oct). Deep calibration of rough stochastic volatility models. *arXiv e-prints*, arXiv:1810.03399.

Bilokon, P., M. Dixon, and I. Halperin (2020). *Machine Learning in Finance: from Theory to Practice* (1st ed.). Springer.

Borovykh, A., S. Bohte, and C. W. Oosterlee (2017, Mar). Conditional Time Series Forecasting with Convolutional Neural Networks. *arXiv e-prints*, arXiv:1703.04691.

Bühler, H., L. Gonon, J. Teichmann, and B. Wood (2018, Feb). Deep Hedging. *arXiv e-prints*, arXiv:1802.03042.

Carvalho, C. M., H. Lopes, O. Aguilar, and M. Mendoza (2012, 01). Dynamic stock selection strategies: A structured factor model framework. *Bayesian Statistics 9 9*.

Chen, L., M. Pelger, and J. Zhuz (2019, March). Deep learning in asset pricing. Technical report, Stanford University.

de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley.

Dhar, V. (2013, December). Data science and prediction. *Commun. ACM 56*(12), 64–73.

Dixon, M. (2018). A high frequency trade execution model for supervised learning. *High Frequency 1*(1), 32–52.

Dixon, M., D. Klabjan, and J. H. Bang (2016). Classification-based financial markets prediction using deep neural networks. *CoRR abs/1603.08604*.

Dixon, M. F. and N. G. Polson (2019, Mar). Deep Fundamental Factor Models. *arXiv e-prints*, arXiv:1903.07677.

Fama, E. and J. D. MacBeth (1973). Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy 81*(3), 607–36.

Fama, E. F. and K. R. French (1992). The cross-section of expected stock returns. *The Journal of Finance 47*(2), 427–465.

Fama, E. F. and K. R. French (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics 33*(1), 3 – 56.

Fama, E. F. and K. R. French (2015). A five-factor asset pricing model. *Journal of Financial Economics 116*(1), 1–22.

Feng, G., J. He, and N. G. Polson (2018, Apr). Deep Learning for Predicting Asset Returns. *arXiv e-prints*, arXiv:1804.09314.

Flood, M., H. V. Jagadish, and L. Raschid (2016). Big data challenges and opportunities in financial stability monitoring. *Financial Stability Review* (20), 129–142.

Gu, S., B. T. Kelly, and D. Xiu (2018). Empirical asset pricing via machine learning. Chicago Booth Research Paper 18-04.

Guida, T. (2019). *Big Data and Machine Learning in Quantitative Investment*. Wiley Finance. Wiley.

Halperin, I. Machine learning and reinforcement learning in finance. *Coursera specialization, https://www.coursera.org/specializations/machine-learning-reinforcement-finance*.

Halperin, I. (2017, Dec). QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *forthcoming in Journal of Derivatives*, arXiv:1712.04609.

Halperin, I. (2019). The QLBS Q-Learner goes NuQLear: fitted Q iteration, inverse RL, and option portfolios,. *Quantitative Finance 19*(9), 1543–1553.

Halperin, I. and M. F. Dixon (2018). "Quantum Equilibrium-Disequilibrium": Asset Price Dynamics, Symmetry Breaking, and Defaults as Dissipative Instantons. *forthcoming in Physica A abs/808.03607*, arXiv:1808.03607.

Halperin, I. and I. Feldshteyn (2018, May). Market Self-Learning of Signals, Impact and Optimal Trading: Invisible Hand Inference with Free Energy (Or, How We Learned to Stop Worrying and Love Bounded Rationality). *https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3174498*.

Harvey, C. R., Y. Liu, and H. Zhu (2015, 10). ... and the Cross-Section of Expected Returns. *The Review of Financial Studies 29*(1), 5–68.

Heaton, J. B., N. G. Polson, and J. H. Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry 33*(1), 3–12.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR 114*, 953–956.

Martin, C. H. and M. W. Mahoney (2018). Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *CoRR abs/1810.01075*.

Martin, D. (1977). Early warning of bank failure: A logit regression approach. *Journal of Banking & Finance 1*(3), 249–276.

Moritz, B. and T. Zimmermann (2016). Tree-based conditional portfolio sorts: The relation between past and future stock returns.

Rosenberg, B. and V. Marathe (1976). Common factors in security returns: Microeconomic determinants and macroeconomic correlates. Research Program in Finance Working Papers 44, University of California at Berkeley.

Sirignano, J., A. Sadhwani, and K. Giesecke (2016, July). Deep Learning for Mortgage Risk. *ArXiv e-prints*.

Swanson, N. R. and H. White (1995). A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks. *Journal of Business & Economic Statistics 13*(3), 265–275.

Trippi, R. and D. Desieno (1992, 01). Trading equity index futures with a neural network. *19*, 27–33.

van den Oord, A., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu (2016). Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*.

# A Summary of Statistical Tests

| **Name** | **Description** |
|---|---|
| Augmented Dickey-Fuller test | Used to detect whether time series data is stationary. |
| Ljung-box test | Used to determine whether the data is correlated or i.i.d. |
| Granger Causality test | Used to identify input variables (a.k.a features) for use in supervised learning. |

Table 2: *A summary of some of the most useful exploratory data analysis tests for supervised machine learning in finance.*

| Name | Description |
|---|---|
| Chi-squared test | Used to determine whether<br>the confusion matrix of a classifier<br>is statistically significant, or merely white noise. |
| t-test | Used to determine whether the output<br>of two separate regression models<br>are statistically different on i.i.d. data. |
| Mariano-Diebold test | Used to determine whether the output of two separate<br>time series models are statistically different. |
| White test | Used to determine whether the<br>the error is heteroschedastic or homoscedastic |
| Portmanteau test | A general test for whether the error<br>in a time series model is auto-correlated.<br>Example tests include the Box-Ljung and the Box-Pierce test. |

Table 3: *A summary of some of the most useful diagnostic tests for supervised machine learning in finance. An example of supervised machine learning for time series methods are recurrent neural networks.*