TRAINING AI FOR ASSET PRICING

HOW TIME-WEIGHTING A MACHINE LEARNING PROGRAM'S TRAINING DATA AFFECTS ITS ABILITY TO PREDICT STOCK RETURNS

AXEL JERNBÄCKER

HENRIK ANGELSTIG

Bachelor Thesis

Stockholm School of Economics

2022

Training AI for Asset Pricing : How Time-Weighting a Machine Learning Program's Training Data Affects Its Ability to Predict Stock Returns

Abstract:

This bachelor's thesis examines how a machine learning program's ability to predict stock prices is affected by how the data on which the program is trained has been time-weighted. Building upon the work by Gu, Kelly, and Xiu (2020), we examine how the same machine learning program performs when its training data is time-weighted using one of three methods: exponential weighting, linear weighting, or cutoff. Our results show a tiny improvement when using either exponential or linear weighting, and a small worsening for cutoff, compared to no time-weighting. However, these differences were so small as to be almost negligible. While our study shows no evidence that weighting training data affects the program's accuracy, there is a vast number of other possible combinations of different machine learning program designs and datasets that could be examined. Therefore, a more comprehensive study examining a greater variety of ML program designs and datasets should be undertaken before a final conclusion on the subject can be made.

Keywords:

AI, machine learning, asset pricing, stock price prediction, time-weighting, concept drift

Authors:

Axel Jernbäcker (24808)
Henrik Angelstig (24829)

Tutors:

Adrien d'Avernas, Assistant Professor, Department of Finance

Examiner:

Adrien d'Avernas, Assistant Professor, Department of Finance

Bachelor Thesis
Bachelor Program in Business & Economics
Stockholm School of Economics
© Axel Jernbäcker and Henrik Angelstig, 2022

Table of Contents

Introduction **1**
- Background 1
- The Problem Today and Purpose of Thesis 2
- Question at Issue 3
- Delimitations 3

Literature Review **3**

Theory **4**
- Concept Drift 4
- Lucas' Critique 5
- Neural Networks 5
- Backpropagation and training 5

Data and Methodology **6**

Results **8**

Analysis of Results **10**

Discussion of Results **11**

Conclusion **12**

Suggestions for Further Research **12**

References **12**

Appendix Matlab **12**

Appendix Neural Network **13**

Introduction

Background

Correct asset pricing is a crucial component in allocating capital to its most productive uses in an economy. When stocks are mispriced, it may cause either an over- or underinvestment of capital in affected firms, resulting in a loss of value to both investors and the economy at large.

For this reason, researchers have devoted much effort to understanding what factors that influence stock prices. One approach toward this goal has been to develop theoretical frameworks, such as the Capital Asset Pricing Model or the Fama-French three-factor model. But the drawback of these models is that they only capture a small portion of all potential factors. Another approach has been to empirically investigate how certain factors have affected stock returns. These empirical studies consider many more potential factors, but their downside is that they only define the relationships between stock prices and factors as static, i.e that they will always affect each other in the same way. In reality, these relationships are likely to be more dynamic.

However, recent advancements in AI and machine learning have now enabled researchers to develop more complex models, where relationships between different factors can be considered. For example, a study by researchers Gu, Kelly, and Xiu managed to create a portfolio strategy that, by timing S&P 500 using neural network-based ML forecasts, more than doubled the portfolio's risk-reward ratio compared to the leading literature's regression-based analysis.

The Problem Today and Purpose of Thesis

AI and machine learning (ML) lend themselves nicely to stock price prediction, where decision-making requires the analysis of countless variables with many complex interrelations. But ML has a major caveat: its predictions rely on the assumption that the same patterns that held in the past will continue to hold true in the future.

In the stock market, however, there is no guarantee that cause and effect relationships will remain constant over time. The Lucas critique makes this point specifically by arguing that the economy regularly undergoes fundamental transformations that change how different factors relate to each other over time. This means that patterns that held true in the past may not continue to hold true in the future. In machine learning, this notion of changing cause and effect relationships is called *concept drift*.

The problem with previous ML-based research is that they have given *equal weight* to the data used for training the ML program. But assuming that the Lucas' critique holds true, this implies that the ML program may be subject to concept drift, i.e it has trained itself to identify cause and effect relationships that held true in the past but which may no longer hold for the future period the program seeks to predict.

Our study aims at minimizing this error by *time-weighting* the data used for training an ML program. By giving greater weight to more recent data on stock prices and their influencing factors, we hypothesise that an ML program will make more accurate predictions compared to when no time-weighting is applied. In addition to studying whether time-weighting is

effective, our study also aims to answer which method of time-weighting will lead to the highest performance.

Question at Issue

Our thesis question at issue is:

> *When training a neural network-based machine learning program to predict stock prices, does time-weighting the program's training data lead to more accurate predictions? If so, which method of time-weighting training data is best?*

Delimitations

The delimitations of our study are as follows:

1. The data used is up to 60 years old, which may cause some inaccuracies.
2. The data only encompasses US stocks during the period 1957-2016.
3. The study is based on the same 93 different factors used by Gu et. al. (2000). Other potential price-influencing factors are not considered.
4. The training and validation split is using a rolling scheme where the most recent 12 years are the validation data and the prior years the training data. This is done to recreate the results from the study by Gu et. al. (2000).

Literature Review

Several papers have investigated the effectiveness of machine learning for stock price prediction, and research has suggested several methods for dealing with concept drift in asset pricing. However, there is a lack of empirical data on how effective these suggested weighting scheme methods are. Providing that data is the extension to the existing literature that our study aims to make.

One of the closest papers to our research is Gu, et. al. (2020), *Empirical asset pricing via machine learning*. Their study uses a neural network ML program to predict returns of nearly 30,000 US-based stocks during the period 1957-2016, based on a set of 94 characteristics for each stock. The paper validates that machine learning indeed is a potent tool for predicting stock returns, managing to double the Sharpe ratio compared to foremost linear regression. For training their ML program, Gu et. al. utilize a combination of the expanding and rolling window approaches. Their ML program is refitted once every year on an expanding window basis, where the training sample size is increased by 1 year each time the program is refitted. However, the size of the validation sample is kept constant on a rolling window basis of 12 years. An expanding window approach carries the risk that data patterns from one period may no longer hold true in future periods because of structural changes in the economy.

Consequently, the ML program trained on expanding windows gets stuck with 'outdated' data to navigate a radically different environment.

Another paper, Drobetz & Otto (2021). *Empirical asset pricing via machine learning: evidence from the European stock market*, builds upon the ML research of Gu et. al. by investigating their findings for European stocks. In contrast to Gu et. al., this study utilizes a pure rolling window approach with 8 years of data for training, 2 years for validation, and 1 year for out-of-sample testing. Furthermore, the ML program is trained on just 22 predictors instead of 94. While Drobetz & Otto's study produced a slightly higher Sharpe ratio than Gu et. al, 1.41 compared to 1.35, it is unclear how much of this improvement was caused by 1) using a different dataset, 2) using a different number of predictors, or 3) using a pure rolling window compared to a hybrid recursive-rolling window approach.

A third paper, Pesaran (2009). *Forecast Combination across Estimation Windows*, investigates how asset price forecasting can be improved by using data windows of different lengths and formats. Based on the analysis of 20 world equity index futures, the study concludes that a strategy of averaging forecasts based on windows of different lengths generates the lowest forecast error among a variety of approaches. This study gives credibility to the idea that more recent data should be given greater weight when predicting the future, while still maintaining the need to include some older data in the analysis as well. Furthermore, the paper found exponential time-weighting produced similar results as averaging window estimates, but noted that the accuracy of the forecast is impacted significantly by the choice of the discount coefficient.

Finally, Nagel (2021). *Machine Learning in Asset Pricing* discusses methods for dealing with ML concept drift in asset pricing, and specifically mentions weighting schemes such as cutoff and exponential weighting as possible solutions. But the discussion is reserved only on a theoretical level without any comprehensive study on actual stocks.

None of these papers, however, present any concrete results of how time-weighting data affects an ML program's effectiveness in predicting stock prices. While the existing literature suggests that time-weighting might be an improvement, the research field is lacking tangible results regarding both 1) the merits of time-weighting in general, and 2) how effective different methods of time-weighting are. Shedding light on these two questions is the purpose of our study.

The contribution this paper aims to add to the literature is two-fold. First, to AI research, we hope that our analysis of different weighting schemes will provide guidance on how ML programs should be trained to avoid concept drift in general. Second, to the finance research, we aim to generate a higher-performing algorithm for predicting stock prices that other researchers may build upon in the future.

Theory

Concept Drift

In machine learning terminology, concept drift refers to changes over time in the target variable trying to be predicted (Lu et al., 2018). It is a well-studied area in machine learning because it worsens the performance of most models that are trained on historical data. The idea is that a model trained on historical data will learn patterns that enable it to predict, for example, share price. However, the patterns that exist in the historical data would have been rendered obsolete by concept drift and therefore the model will perform poorly. To counter this there are different approaches, most notably refreshing and reactive solutions. Reactive solutions work by detecting deterioration in prediction accuracy which triggers a retraining of the model to keep error due to concept drift within an acceptable error bound. Refreshing is a solution where models are continuously being trained on newer data and replacing old models in an ensemble. The solution used in this thesis is an implementation of refreshing where an ensemble of models is trained on data weighted towards more recent data. This effectively makes the model more sensitive to patterns in recent data but also enables it to find patterns that could persist for the entire historical period.

Lucas' Critique

To understand why concept drift relates to predicting patterns in the stock market there is a need to look at the underlying economic theory. The most notable example here is the Lucas critique. In essence, the critique says that it is naive to predict the effects of a change in economic policy only based on relationships in historical data. This is due to the change of behavior in different economic policies. More closely related to this thesis one could explain it such that, when patterns are detected in the stock market, the trading changes which in turn alter the patterns. Therefore it would be naive to assume the patterns that could be used in evaluating returns are invariant of time and behavior. This is a brief reasoning as to why a certain measure of concept drift is to be expected when relating historical data and patterns to current or future data.

Neural Networks

There exist many different machine learning models, each one with its own benefits and drawbacks. One of the most widely used is Neural Networks which are excellent and versatile predictors. Without going too deep into the technical nature of a neural network the basics of them will be explained here. A network consists of "layers" which hold "neurons". Each layer has a number of inputs and outputs. These inputs each go through a neuron, where some weighted function is applied to it, in our case the ReLU (a standard function in neural networks). This produces an output that is sent to the next layer of neurons. The idea is that a properly weighted network becomes a complex function that takes some complex input, for example accounting information, and then produces some desired output, for example, the expected share price. These types of problems hold great interest in society and therefore neural networks are widely used.

Backpropagation and training

After explaining the value of a neural network one might wonder what holds them from being universal predictors that perfectly predict all complex dynamics in society. This relates to the training of networks. The training is the method with which you set the desired weights to the neurons in the different layers. Since there can be thousands of weights and these only make sense in a context with each other this is far from a straightforward process. There will not be much technical detail in this explanation but it serves conceptually to create an understanding of how weighting the training data affects the created model. The training method used in this thesis and most popular in general is backpropagation. It works by inputting training data and the desired outcome. The model then runs and calculates a loss, this can be defined in different ways, for example, mean squared difference. This loss is used to approximate a value for how "wrong" each weight is. This is done using a complicated process of derivative approximations that in the end can update the weights to be more "right". The weights of training data are normally 1 for all data points which means that having a difference between desired and actual outcome is equally important regardless of where in the training set that particular datapoint resides. This study experiments with weights that make the training more focused on minimizing errors on recent data points. This relates to the issue of concept drift above where the older data is more likely to contain larger differences from current stock return patterns.

Data and Methodology

Data Collection

In this thesis, the data used comes from the WRDS database combined with a lagged dataset made by Gu, Kelly, and Xiu in their paper: *Empirical Asset Pricing via Machine Learning.* This dataset has firm characteristics that are lagged to mimic how the public would receive the information. Therefore, monthly characteristics are delayed 1 month, quarterly are delayed 4 months and annual data is delayed by a year. The dataset contains 94 lagged firm characteristics, ID:s for every stock and date. Furthermore the returns of the stocks given a certain date are appended, these are downloaded from the WRDS database using the CRSP annually updated stock returns.

Description of Methodology

The methodology used in this thesis is predominantly quantitative. The work begins by downloading the data set used by Gu, Kelly, and Xiu which contains nearly 30 000 stocks over a period of 60 years. This data contains the lagged firm characteristics and only needs to be appended to the monthly return of stocks, data which can be found in the WRDS database.

Secondly, the neural network used in this thesis has to be constructed according to the specifications of Gu, Kelly, and Xiu. This neural network is described extensively in their paper and could, apart from small ambiguities, be replicated exactly. The network consists of an input layer with all the lagged firm characteristics, 3 dense hidden layers with 32, 16, and

8 neurons respectively, and a single output node. The network uses a ReLU activation between all layers and an ensemble training of 10 networks for stability. A batch normalization function between layers for faster training. An early stopping algorithm that detects when the error in the validation set starts to increase again, has patience 5. This means it will run for 5 epochs even after detecting an increase in validation set errors. A normal Adam optimization algorithm is used and the learning rate is between 0.001 and 0.01.

There is an L1 penalty that causes parameters close to zero to tend to zero, to remove irrelevant inputs. As one can see there is extensive documentation on how to replicate their network but there still remain some ambiguities. Firstly, and most notably, the L1 penalty is between $10^{-5}\ to\ 10^{-3}$. This is most likely different values for the hidden layers but the paper does not state if it is in ascending or descending order. Secondly, the learning rate decreases from 0.01 to 0.001 but they do not state at what rate the decrease happens. Thirdly, they don't specify which function they use to compute loss in backpropagation training. This could be either Huber loss or least squares as both are used in their paper. They do, however, seem to imply that least squares are used for neural networks, which is also used in this bachelor thesis. Fourthly, they do not specify whether they use a rollback in their early stopping or not. It is quite common to "revert back" to the best result which minimized the error in the validation set. But they could also use the latest iteration of their trained network, omitting a rollback scheme. Our network uses a rollback scheme.

After replicating the neural network custom loss functions are implemented. These custom loss functions are used to calculate a weighted loss which potentially could improve the performance of the neural network by reducing the issue of concept drift and prioritizing the training of more recent data. Here 3 additional functions are created, one with exponential decay of weight. One with linear decay and one cutoff scheme which ignores data older than 10 years later than the validation set. Worth noting is that we use the rolling validation split they implemented in all different training schemes. This is to improve the comparability between different weighting methods.

When the different loss functions are created the program is run to generate the R-squared out-of-sample values for all different years and weighting schemes. This is a time-consuming process since all networks need to be retrained for every year they should predict. Therefore a loop was implemented that consecutively runs the program and outputs the R-squared out of sample values for all different years. Allowing the computer to run for a few hours yields results for all years with the used weighting scheme. The results can then be aggregated to detect performance differences between the different weighting schemes, most importantly the mean of R-squared out-of-sample.

Rolling scheme

For all different weighting schemes and years predicted the same data split is used. The split consists of training data, validation data, and test data. The model is retrained for every year it is trying to

predict, the year it predicts is the test data. The model uses the 12 most recent years as validation data. It then uses the remaining years as training data. As illustrated in the figure below.

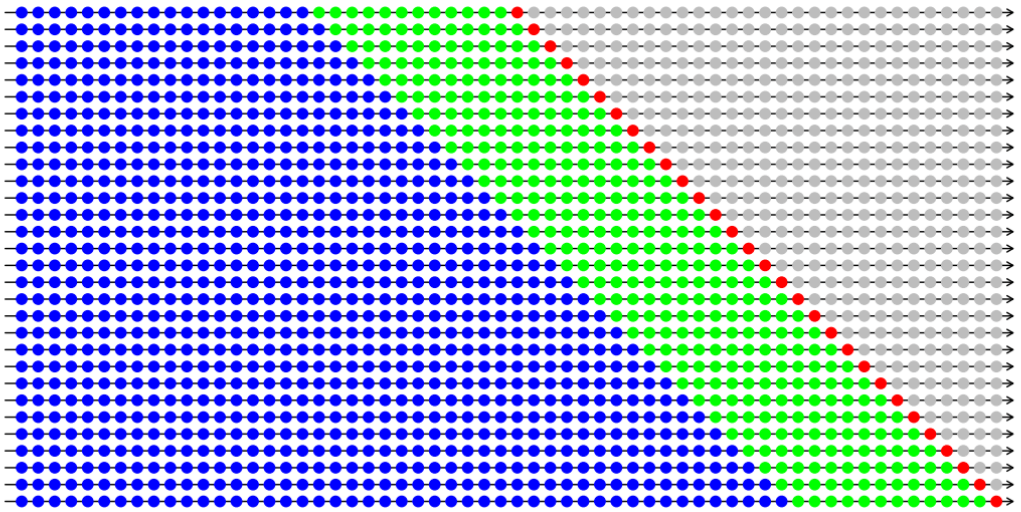


Figure 1: dataset split during different years

The blue dots here correspond to the years used for training data. The green is the validation data used for the early stopping. The red years are the out-of-sample years that are predicted. The model is retrained with the different datasets for every new year to be predicted.

Weighting schemes

The weighting schemes used are exponential, linear, cutoff, and normal. They work such that the exponential decay makes data that is ten years earlier twice as important. With a continuous exponential curve between. This has the benefit of treating data that is 40 respectively 30 years old with the same amount of concept drift as data that is 20 respectively 10 years old.

The second scheme is linear. This is a very simple approach where the oldest data is set to matter zero and the most recent is 100 percent. The data in the middle as such matters half of what the newest data does. The drawback with this is that data points that are 58 years old might matter 2 times as much as data points that are 59 years old due to the nature of the linear relationship. Furthermore, it decays very slowly in the beginning when counting as a proportion of the weight of each data point.

The third scheme is a cutoff scheme, sometimes also called a rolling window. This is even more simple than the linear and means that the model only trains on the ten latest years in the training set. This should deal with concept drift very well, however it reduces the size of the training set drastically and as such, could lead to other issues during training. It is beneficial for a neural network model to train on as large of a dataset as possible.

Results

Running the Neural Network with the four different weighting schemes – Exponential, Linear, Cutoff, and No Weight – gives the results found in the table below. The percentages displayed in the table represent the out-of-sample R-squared value when predicting the year shown at the top.

R-squared is a statistical measurement that indicates how good a model is at explaining the results. An R-squared of 100% means the model predicts the results perfectly every time, while an R-squared of 30% means that 30% of the results can be explained by the model. Generally, the higher the R-squared value, the better the model is.

R-squared of ML model's predicitions for period 1988-2016, based on the four different weighting schemes

	Weighting Scheme			
Year	**Exponential**	**Linear**	**Cutoff**	**No weight**
1988	0.67%	0.63%	0.38%	0.64%
1989	0.27%	0.30%	0.13%	0.33%
1990	-0.45%	-0.19%	-0.87%	-0.62%
1991	1.24%	1.15%	0.96%	1.35%
1992	0.80%	0.75%	0.83%	0.78%
1993	1.09%	1.24%	1.22%	0.95%
1994	-0.80%	-0.84%	-1.03%	-0.73%
1995	1.26%	1.24%	1.33%	1.34%
1996	-0.09%	-0.05%	-0.50%	-0.13%
1997	-0.54%	-1.01%	-1.10%	-0.41%
1998	0.02%	-0.01%	-0.22%	0.09%
1999	1.66%	1.51%	1.28%	1.54%
2000	-0.70%	-0.52%	-0.74%	-0.63%
2001	0.48%	0.82%	0.56%	0.72%
2002	-0.78%	-0.66%	-0.67%	-0.69%
2003	2.42%	2.32%	1.67%	2.33%
2004	1.34%	1.12%	1.20%	0.78%
2005	-0.13%	-0.42%	-0.19%	-0.14%
2006	1.10%	1.21%	1.40%	1.32%
2007	-1.07%	-1.16%	-1.32%	-1.10%
2008	-2.14%	-2.41%	-2.83%	-2.34%

	Weighting Scheme			
Year	**Exponential**	**Linear**	**Cutoff**	**No weight**
2009	1.72%	1.55%	1.14%	1.53%
2010	1.70%	1.67%	1.90%	1.69%
2011	-1.34%	-1.40%	-1.32%	-1.32%
2012	1.21%	1.12%	1.04%	1.01%
2013	2.91%	2.82%	3.06%	2.41%
2014	-0.72%	-0.37%	-0.50%	-0.43%
2015	-1.01%	-1.05%	-0.69%	-1.20%
2016	0.95%	0.97%	0.70%	0.80%

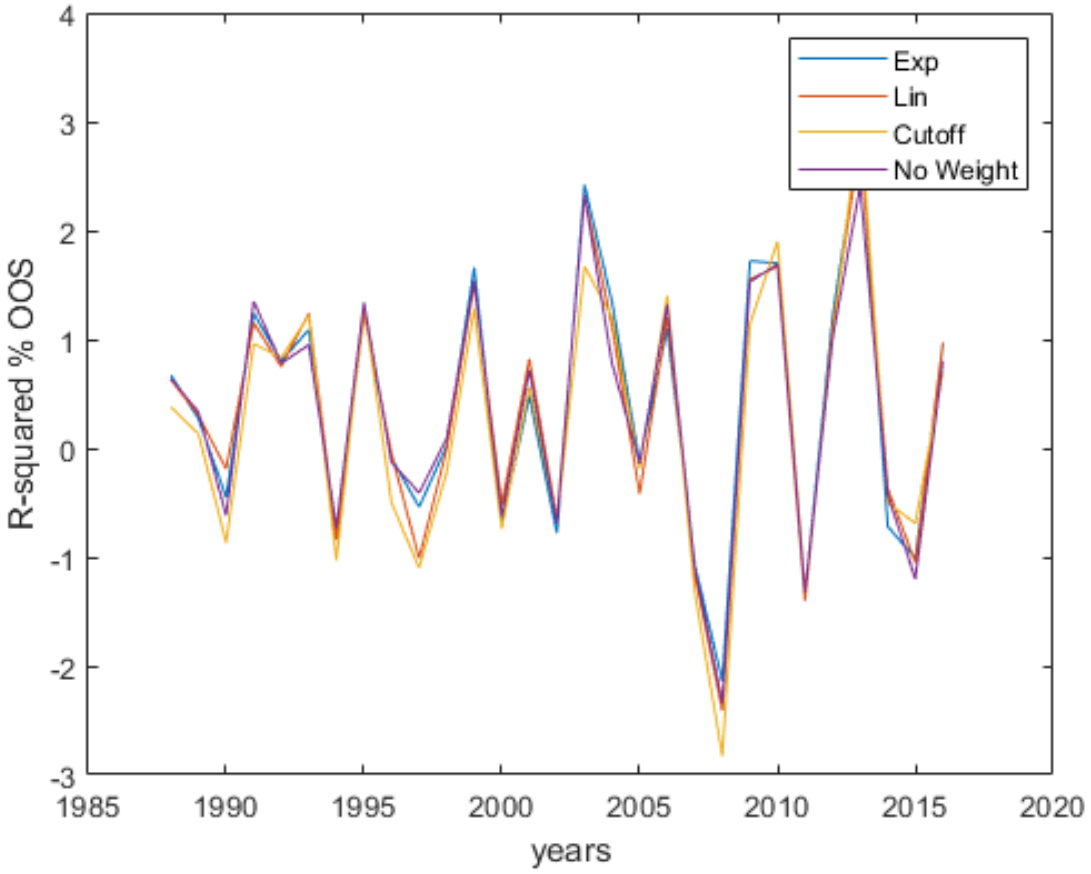
From this table above, the mean R-squared is calculated using the program found in Appendix Matlab. The results are shown below.

R-squared out-of-sample mean

Exp	**Linear**	**Cutoff**	**No weight**
0.38%	0.36%	0.24%	0.34%

The graph below shows how the different schemes perform over time.

R-squared values of each weighting scheme



Analysis of Results

The results of this study show that different weighting schemes perform similarly in predicting stock returns. The most notable exception is the cutoff scheme, which not only performed worse than the other two weighting schemes, but also worse compared to training without any weighting at all. As shown by the graph above, it is evident that the predictions made by the different networks are very similar, and the difference that weighting historical data causes is small or negligible in the setup described in this thesis methodology.

There are a few areas in the setup that should be considered before drawing any conclusions that weighting historical data for training does not impact predictive performance.

First and foremost, this network is trained on historical data with a training/validation split that uses the last 12 years of the known training data as validation data. This is useful because it indicates when the model starts overfitting. Therefore, the setup utilizes an early stop function which prevents overfitting during training. This could, however, lead to an effect that “hides” the impact of weighting schemes. Since the training stops when the model fits the validation data well, it can be viewed as such that the model is trained to fit the validation data, which is not weighted, only the training data is. This would intuitively make the model more likely to converge to solutions that are good on the validation data which is unrelated to the used weighting schemes.

Secondly, the R-squared measurement is slightly crude and not necessarily that interesting when analyzing the predictive power of a model. The returns of a model with a slightly lower R-squared could still be higher than a model with a higher R-squared. R-squared does however indicate how well the model predicts returns overall very well, and is thus still an interesting measurement.

Thirdly, a neural network is only one of many different models that could be suited for predicting stock returns. In the paper by Gu et al., they had the most success with a network of this design which is why it is used in this paper. But it is not obvious by any means that the impact of weighting historical data would be the same for a neural network as for an elastic net or Lasso model.

With these caveats in mind, these results still indicate that weighting schemes do not impact the predictive power of a neural network trained on stock returns. More elaborate ML training schemes might have to be designed to combat the issue that concept drift poses to predicting stock returns using AI.

Discussion of Results

The results shown above show that the performance of different weighting schemes is very similar. The most notable exception is the poor performance of the cutoff scheme. This could, however, be caused by the fact that the cutoff scheme consistently trained on a smaller training dataset. With other schemes using 18-48 years of data it always only used 10 years. This leads to shorter epochs during training and increases the likelihood of an early stop being premature. Since the training uses 10 000 data points per batch in each epoch there are far fewer steps for the cutoff scheme. This means it has to go through more epochs than other schemes to “travel” to a local maximum in predictive power.

Furthermore, the split between training data and validation data could likely have had a norming effect on the impact of weighting schemes. Therefore it is not guaranteed that the results of this study confirming small or no impact with weighting schemes would be replicated in a study with a different training/validation split.

The dataset is not complete, there are missing returns for some stocks in different time periods. These rows with missing returns are excluded but could nonetheless affect the training of the network. For example, there are many more missing data points for the oldest data, from the 50s and 60s. This leads to different classes of data points being represented disproportionally. With fewer old data points and more newer ones, the training will naturally tend to fit newer data better. This could inadvertently account for concept drift and actually reduce the effect weighting schemes would have. It could be imagined that on a complete dataset a larger effect could be seen from weighting schemes.

Overall the predictive power is low, with R-squared being measured in fractions of a percent. This gives credibility that the stock market as a whole is almost truly random. That does not mean that the model performs poorly or well, however. An analysis showing the return of trading strategies using networks trained with the different weighting schemes might show larger discrepancies between returns.

This work gives an indication that weighting data during training might be too naive of an approach to combat concept drift, at least by itself, especially with the training/validation split used which partially accounts for concept drift. There is more work to be done that would further strengthen or refute the results of this work. Some ideas are written under "Suggestions for Further Research".

Conclusion

To summarize the results of this study, it is not impactful to time-weight data when training a neural network that predicts monthly stock returns. The performance of the model remains similar across different weighting schemes compared to no weighting, and has similar predictivity between years as shown in the plot above. While a few caveats that have been discussed above exist, this work demonstrates that weighting historical data is most likely not sufficient to avoid concept drift in its entirety during training.

Suggestions for Further Research

While our study shows little evidence that weighting training data improves the program's accuracy, there is a vast number of other possible combinations of different machine learning program designs and datasets that could be examined.

To illustrate, ML program designs can vary in: the number of neural network layers used, the number of neurons within each layer, size of training data, size of validation data, and the discount factor used, just to mention a few dimensions. And within each of these dimensions, there is a lot of room for variability. For example, the number of neurons in each layer can be anywhere from two to several hundred.

In addition, the dataset of stock prices used can vary in the geography of firms, size of firms, time period examined, and the number of stock price factors examined. And within each of these dimensions, there is again a lot of room for variability.

In total, the number of combinations of ML program design and datasets that could be studied easily exceeds one million. Naturally, if each combination was examined we should expect many of them to yield similar results as our study; that time-weighting training data has little to no effect on ML prediction performance. However, it is not unreasonable to believe that there could also be combinations that *do* show an effect of time-weighting training data. We,

therefore, suggest that future research should examine a larger quantity of diverse combinations of datasets and ML program designs, and perform a comparative meta-study of these examinations, in order to reach a final conclusion on the matter.

References

Drobetz, W. and Otto, T., 2020. *Empirical Asset Pricing via Machine Learning: Evidence from the European Stock Market.* SSRN Electronic Journal.

Gu, S., Kelly, B. and Xiu, D., 2018. *Empirical Asset Pricing via Machine Learning.* SSRN Electronic Journal.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. and Zhang, G., 2018. *Learning under Concept Drift: A Review*. IEEE Transactions on Knowledge and Data Engineering, p.1-1.

Nagel, Stefan. *Machine Learning in Asset Pricing*. Princeton University Press, 2021, https://doi.org/10.2307/j.ctv19fvx8r. p.59-63.

Pesaran, M. and Pick, A., 2011. *Forecast Combination Across Estimation Windows*. Journal of Business & Economic Statistics, 29(2), p.307-318.

Appendix Matlab

```
Years = 1988:2016;
Exp =
[0.67,0.27,-0.45,1.24,0.80,1.09,-0.80,1.26,-0.09,-0.54,0.02,1.66,-0.70,0.48,-0.78,2.42,1.34,-0.13,1.10,-1.07,-2.14,1.72,1.7,-1.34,1.21,2.91,-0.72,-1.01,0.95];
Lin =
[0.63,0.3,-0.19,1.15,0.75,1.24,-0.84,1.24,-0.05,-1.01,-0.01,1.51,-0.52,0.82,-0.66,2.32,1.12,-0.42,1.21,-1.16,-2.41,1.55,1.67,-1.40,1.12,2.82,-0.37,-1.05,0.97];
Cutoff =
[0.38,0.13,-0.87,0.96,0.83,1.22,-1.03,1.33,-0.50,-1.1,-0.22,1.28,-0.74,0.56,-0.67,1.67,1.20,-0.19,1.40,-1.32,-2.83,1.14,1.9,-1.32,1.04,3.06,-0.5,-0.69,0.70];
NoWeight =
[0.64,0.33,-0.62,1.35,0.78,0.95,-0.73,1.34,-0.13,-0.41,0.09,1.54,-0.63,0.72,-0.69,2.33,0.78,-0.14,1.32,-1.1,-2.34,1.53,1.69,-1.32,1.01,2.41,-0.43,-1.2,0.8];

plot(Years, Exp)
hold on
plot(Years, Lin)
plot(Years, Cutoff)
plot(Years, NoWeight)
legend('Exp','Lin', 'Cutoff', 'No Weight')

mean(Cutoff)
mean(Exp)
mean(NoWeight)
mean(Lin)
```

Appendix Neural Network

See Python files for machine learning models.