Korea Advanced Institute of Science and Technology

Introduction to Financial Engineering

Market Analysis and Portfolio Optimization

Professor:
Woo Chang Kim

Student:
Federico Berto

Course ID:
IE471

ID number:
20204817

Introduction

The goal of this report is to describe the experimental process and result for analyzing the market and using prediction results for optimizing a portfolio. In particular, after predicting the future time-series of stock prices, we will optimize the weigthings for obtaining an optimal portfolio.
Let us briefly introduce the AI models we will use in the code.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) [5] are capable of dealing with sequences of data. As Figure 1 [1] shows, they form an unrolled, sequential undirected graph in which input data are fed sequentially. The vanilla implementation's most notable problem

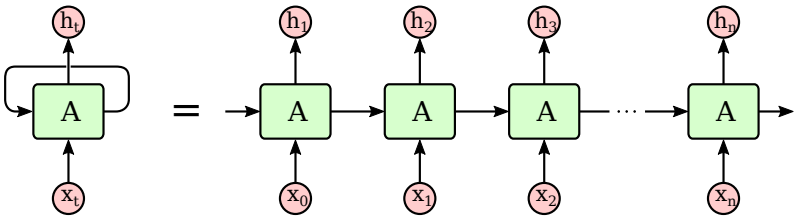

Figure 1: Recurrent Neural Network (RNN) base model

is the so-called *vanishing gradients* problem, in which long time series cannot be dealt with due the parameter updates using gradients; due to the sequential nature of RNNs, the further inputs in the time series are subject to more activation functions, causing updates to be almost independent from them.
For this reason, more advanced versions of Recurrent Neural Networks i.e GRU and LSTM have been created. Nonetheless, the simplicity of RNNs makes them suitable for shorter and less complex time series, since they are faster and also less prone to overfitting due to less parameters.

Gated Recurrent Units

Gated Recurrent Units (GRU) [1] were proposed as an RNN variant in 2014: they are similar to LSTMs without an output gate and has fewer parameters. Figure 2 [2] shows an overview of the architecture. Having fewer parameters than LSTM , GRUs have been shown to perform better on certain dataset and have better generalization capabilities with fewer data.

[1] Source: Github
[2] Source: Wikimedia

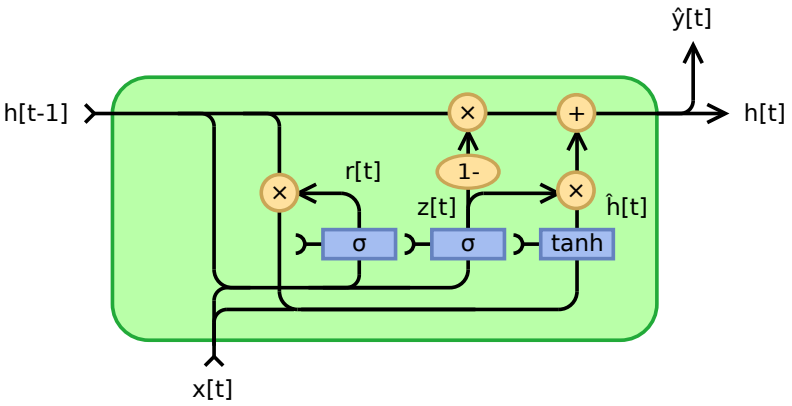

Figure 2: Gated Recurrent Unit (GRU) base model

Long Short-Term Memory

The prediction model is based on the Long Short-Term Memory (LSTM) [3] module in Figure 3[3], which is able to store past information of the data and is thus suitable for time series prediction.

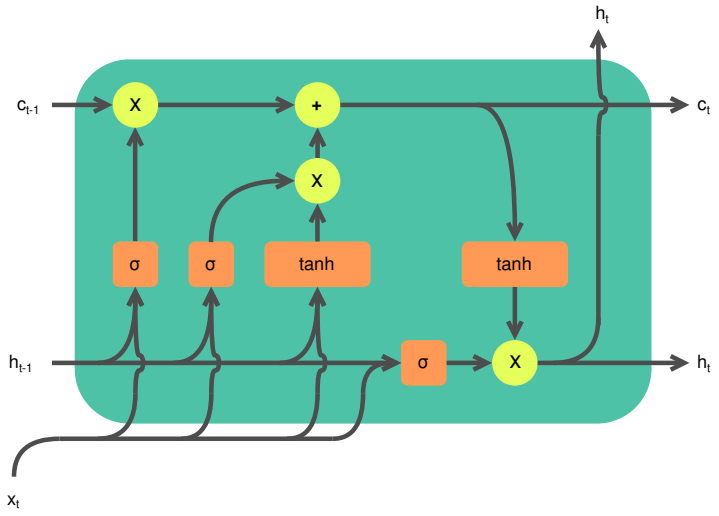

Figure 3: LSTM cell

Further details into the PyTorch [4] implementation can be found in the code, also available on the following Github repository.[4]

[3]Source: Wikimedia
[4]Link: `https://github.com/Juju-botu/financial-engineering-ai`.

Experiments

Portfolio Optimization on top US companies

In this section we provide the required data for the report.

1. Regimes analysis in Figures 4 and 5.

Figure 4: Market analysis with Gaussian Mixture Models

2. Mean values and covariance values of four regimes:

	Regime 0	**Regime 1**	**Regime 2**	**Regime 3**
Mean	137.5967	284.536	197.8587	106.826
Covariance	89.7873	1148.4455	298.8313	142.6559

As we can see, Regime 3 is the one with lowest mean and indeed is the one corresponding to low cumulative returns, while Regime 1 corresponds with high cumulative returns and has high covariances.

- Mean comparison: Regime 1 > Regime 2 > Regime 0 > Regime 3
- Covariance comparison: Regime 1 > Regime 2 > Regime 3 > Regime 0

3. Losses for Netflix `NFLX` every 1000 epochs:

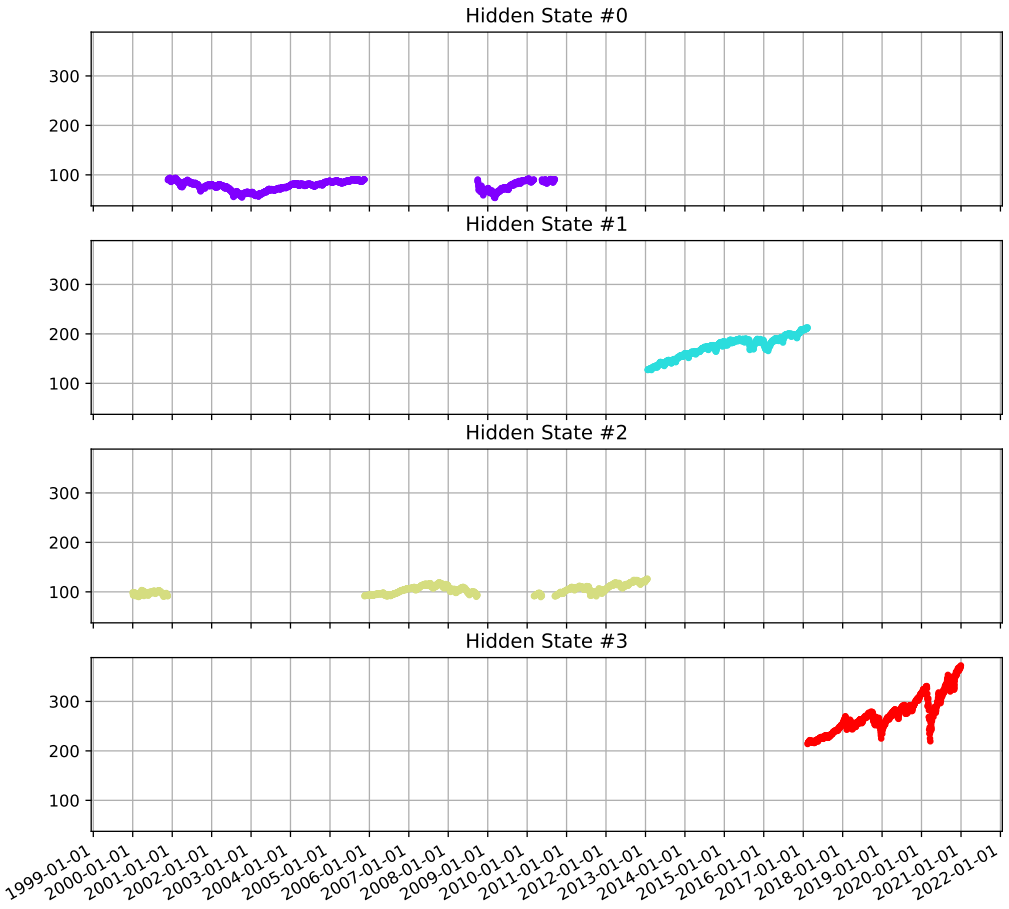

Figure 5: Market analysis with Hidden Markov Models

Epoch	Loss
0	0.3728
1000	0.0003
2000	0.0002
3000	0.0001
4000	0.0001
5000	0.0001
6000	0.0001
7000	0.0001
8000	0.0000
9000	0.0000
10000	0.0000

4. `ResultsofErrorMetrics(LSTM).csv`: attached to the zip file.

5. Three portfolios' last five cumulative returns:

Cumulative Returns of Portfolios			
Date	**LSTM**	**Equally Weighted**	**Capitalization Weighted**
2020-12-24 00:00:00	4.070533	0.667722	0.960323
2020-12-28 00:00:00	4.083723	0.682366	0.991312
2020-12-29 00:00:00	4.099815	0.684749	0.989239
2020-12-30 00:00:00	4.318745	0.687135	0.985936
2020-12-31 00:00:00	4.400515	0.696872	0.991695

6. Comparison of best porfolios: we can clearly see that in this case the LSTM portfolio outperformed both the Equally Weighted and the Capitalization Weighted portfolios; the final return was more than four times the cumulative return of the CPW portfolio.

7. Portfolios performances in cumulative returns over time in Figure 6:

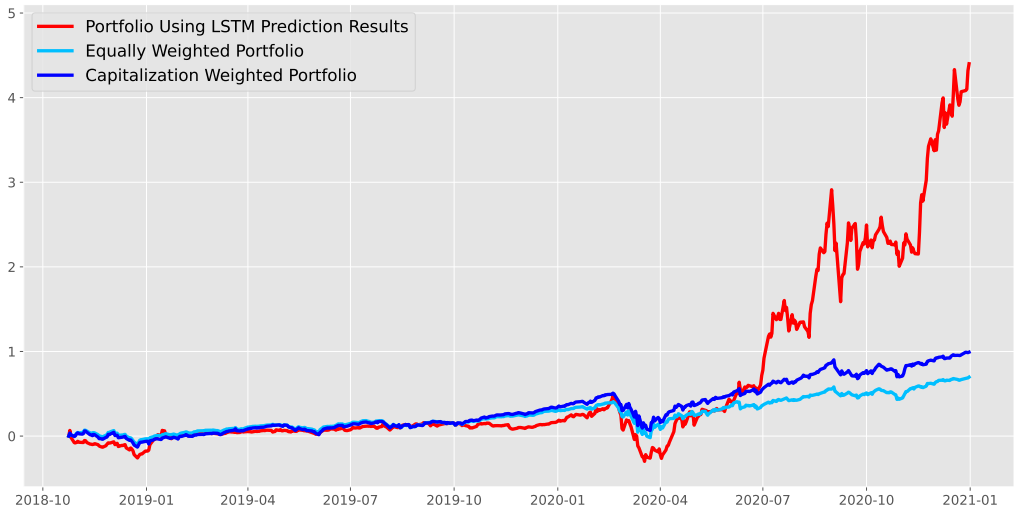

Figure 6: Portfolios performances in cumulative returns over time

Comparison to other algorithms and datasets

In this section, we show how to improve the algorithm and use different datasets and AI algorithms.

Using Pytorch Lightning

We revise the code by using Pytorch Lightning [2]: this PyTorch framework provides a high-level interface by which it is easier to control architectures, results, logging and more. More importantly, it makes the process of moving models to GPU, Tensor Processing Units (TPUs) and even multiple GPUs and TPUs easier while having a negligible overhead. This open-source library is actively maintained by a community highly focused on efficiency and code readability. We refactor the code to be used with Pytorch Lightning.

Results Comparison

We show in Figure 7 a comparison of the algorithms run with the same parameters with Pytorch Lightning and including RNN and GRU results and a version of LSTM with dropout.

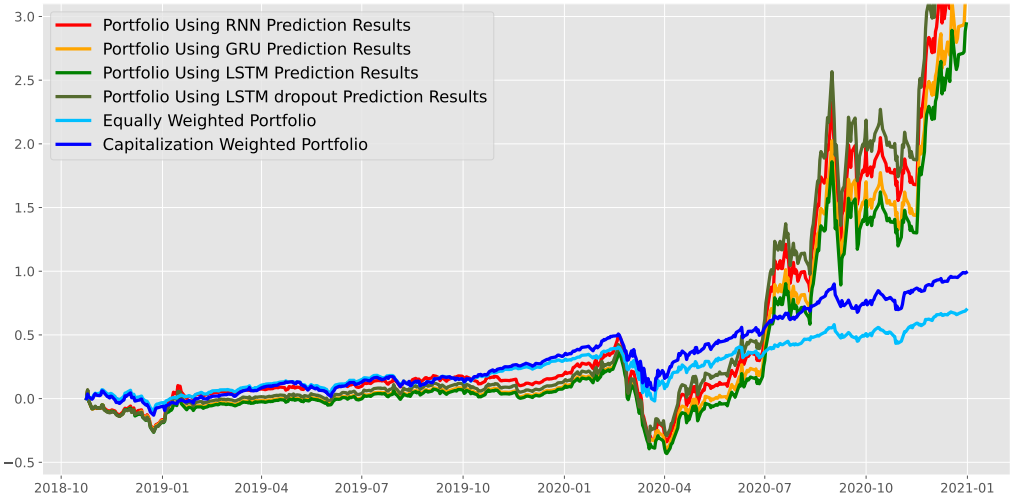

Figure 7: Portfolios performances in cumulative returns over time. As we can see, RNN and GRU perform similarly, while applying dropout as a regularization yields the best results.

Moreover, we train on a new dataset: supposing we want to follow some online advice to invest in 10 stocks [5]. We can train our models and successfully get working portfolios with AI, which we show in Figure 8

[5]Source: US News Money

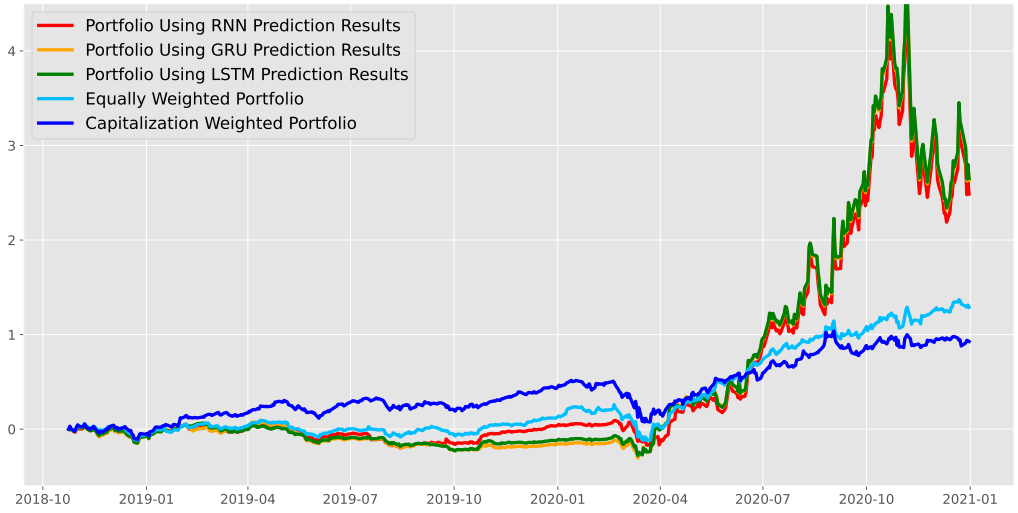

Figure 8: Portfolios performances in cumulative returns over time. As we can see, RNN and GRU perform similarly to LSTM. Even though we may just have 10 stocks, the AI-driven portfolios are still able to outperform the Equally Weighted and the Capitalization Weighted ones.

References

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[5] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.