TimeGAN

December 8, 2022

1 Abstract

Modeling the stock market has tremendous financial implications for investors, consumers, and policymakers. For asset prices though, we observe only one price path over time, recorded usually at the daily level at the finest granularity. To address this, many have turned to deep learning methods such as generative adversarial networks (GANs) as a form of data augmentation. In this paper, we apply a time series GAN (TimeGAN) in a limited data setting, determining its veracity using properties that hold in practice as well as a "train on synthetic, test on real" approach. We find that with limited data and a relatively high-dimensional feature space, GANs may not be able to generate data that well approximates the true data-generating distribution.

2 Introduction

The stock market exhibits very complex temporal dynamics. Given its seemingly randomly movements, models of the stock market have taken various forms of random walks and Brownian motions. However, market movements are entirely determined by economic agents who tend to behave in a predictable manner. As a result, models to predict the movements in stock prices have been undertaken by many. Of course, the successful prediction of the stock market has tremendous financial implications. Moreover, well-calibrated predictions would have applications for policymakers in the federal government and the Federal Reserve. ARIMA, GARCH, and other traditional time series models have been used towards this end, but in recent years, deep learning methods have grown in their popularity. The drawback of deep learning methods is their overparameterization, requiring a substantial amount of data for learning. For financial time series, this poses a problem as we only observe one time series for each asset price. To address this, recurrent generative adversarial networks (rGANs) to generate realistic financial time series have been used, but these networks may not efficiently learn the temporal dynamics of the time series. Instead, we will turn to the recently introduced Time Series GAN (TimeGAN). Since GANs are most useful when we have limited data, we will operate in this setting, determining if the generated data are similar to the true data by checking that they recover the properties exhibited by the observed series and provide similar predictive performance to the true data when trained on a model to predict a holdout test set. The stock prices we will model are those of Amazon, Apple, Tesla, Google, and JP Morgan from 2012 to 2022. These companies are each among the top 10 in market capitalization currently and provide a diverse basket of sectors. This information was obtained from the Yahoo Finance module in Python while general market information was obtained from the St. Louis Federal Reserve. The economic data has been seasonally adjusted using an ARIMA model. A list of all economic indicators and their description can be found in the appendix.

3 Previous Work

In recent years, GANs have been applied to a myriad of domains with success. With respect to finance however, there has been limited work done toward applying GANs in this context. Takahashi et al. apply generative adversarial networks to stock prices for S&P 500 firms from 1960 to 2018, showing that the generated time series recover the fundamental properties of asset returns. They use a multi-layer perceptron (MLP) and convolutional neural network (CNN) as their generator, which does not readily allow for the generator to learn the temporal dependencies in the data. Koshiyama et al. demonstrate the ability of conditional generative adversarial networks to simulate realistic data which can then be used for assembling and fine-tuning trading strategies. However, their CGAN structure requires a fixed time lag for time series generation. A shortcoming of these modeling approaches is the failure to account for the economic and political environment in which the market is operating. For example, the unemployment rate and asset prices are negatively correlated, on average. If we do not incorporate this information in our generator, then our generated data may not observe properties that we know hold in practice.

To ensure properties of the true data distribution hold, a few different approaches have been offered. Goodfellow et al. introduce the notion of feature matching. They adjust the objective of the generator so that a penalty is applied when features for the generated data differ from those of the true data. However, the discriminator learns these features, so if the discriminator does not learn the features of interest, the generator may not actually learn the properties we desire. To capture the distributions of features within each time point, Jarrett et al. introduce the TimeGAN. To learn relationships among features, The TimeGAN embeds the time series in a latent space. In this lower dimensional space, the generator-discriminator play the minimax game to generate data, which is then mapped back into the feature space using a recovery network. Though providing excellent empirical results, the model has not been applied to stock data with general market information and in a setting in which we possess limited data.

4 Methods

Though generative adversarial networks have been shown to recover the fundamental structure of asset returns, using general economic conditions in data generation has not been done. We believe that by incorporating this information, we can create more realistic samples by preserving economic relationships that hold in the markets. In this paper, we will use a TimeGAN along with general market information to produce synthetic financial time series. To determine if our generated data is similar to the observed data, we will use two methods. First, we will check that the stock returns adhere to the following properties used by Takahashi: (a) $\mu \approx 0$ where μ is the mean log return, (b) linear unpredictability: autocorrelation function for the price return is absent for any $k > 1$, i.e. $\frac{E[(r_t-\mu)(r_{t+k}-\mu)]}{\sigma^2} \approx 0$,(c) fat-tailed distribution: the tails of the price return distribution follows power-law decay, i.e. $P(r)\, \alpha\, r^{-\alpha}$ where α typically ranges from 3 to 5, and (d) volatility clustering: large/small price fluctuations tend to cluster together temporally, i.e. $Corr(|r_t|, |r_{t+k}|)\, \alpha\, k^{-\beta}$. Takahashi also uses the leverage effect and coarse-fine volatility correlation. We will not consider these properties in this paper because the leverage effect is market dependent, so with only five stocks, the true underlying relationship is unclear, and the coarse-fine volatility correlation is very noisy, so with a small amount of data, we cannot reliably estimate this property. In addition to these properties, we will also check that the resulting stock prices are inversely correlated with bond rates and unemployment and positively correlated with GDP per capita. These will serve as our measures for the relationships with general economic conditions. We will then conduct a "train on

synthetic, test on real" experiment to judge the quality of the generated data. In particular, two recurrent neural networks (RNNs) with gated recurrent units (GRUs) will be trained on the stock prices from the beginning of 2012 to the end of 2021. The other will be trained on the synthetic data created from a recurrent GAN using this data as examples of the true data distribution. These two networks will then be tested on the current year's stock prices to compare performance. If our TimeGAN learns the data-generating distribution well, then we would expect the predictions to be similar.

In our analysis of stock prices, we will use log returns to measure price movements rather than absolute price movements. We will motivate this by introducing geometric Brownian motion. Geometric Brownian motion is a popular model for stock price movement and is the foundation of the famous Black-Scholes-Merton model. Geometric Brownian motion is derived from the stochastic differential equation (SDE):

$$dS_t = \mu S_t dt + \sigma S_t dB_t$$

where B_t is standard Brownian motion. Heuristically, we can think of this SDE as

$$\log(\frac{S_{t+1}}{S_t}) \approx \frac{\Delta S_t}{S_t} = \mu \Delta t + \sigma Z$$

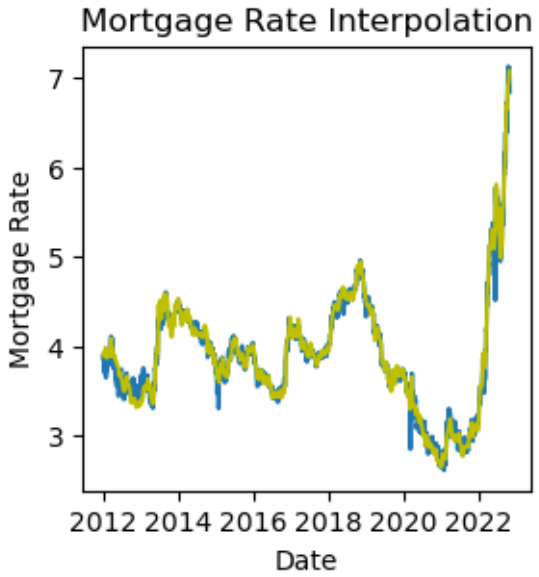
where Z is $N(0, \Delta t^2)$. This implies then that stock returns are $N(\mu \Delta t, \sigma \Delta t^2)$. Since we are dealing with the movement of daily stock prices, Δt corresponds to a one-day change. As returns depend only on the time difference and not the time itself, returns are stationary with respect to time under this model. This is a desirable property because it means that the statistical properties of returns do not change over time. This is important in this context because implementing our TimeGAN will require numerous time series as samples from the data-generating distribution. To meet this requirement, we will have to split the time series for each of the five companies. If our time series are not stationary, which would be the case under normal stock prices, then each time series will have different statistical properties, preventing us from learning the data-generating distribution. By using log returns, we can hope to avoid this issue. To further mitigate any possible effects of seasonality, we proceed by splitting each time series by the year and do not implement a sliding-window approach.

In order to leverage general market conditions to create our timeGAN, one issue that arises is economic data is not released daily. Though bonds are actively traded which allows us to have daily interest rates, mortgage rates are released weekly, inflation and unemployment are released monthly, and GDP and debt figures are released quarterly. To interpolate these figures to a daily scale, we used daily bond interest rates to perform Gaussian process regression. Gaussian process regression assumes $ y MVN(, (x)) $ but for our purposes, we assume $ = 0 $. For $\Sigma(x)$, we use a white kernel along with a squared exponential kernel. In particular,

$$\Sigma_{i,j}(x) = \sigma_n^2 + \sigma_s^2 e^{\frac{-||x_i - x_j||_2^2}{\ell^2}}$$

We find $\sigma_n^2, \sigma_s^2, \ell^2$ using maximum likelihood estimation. For new data (x_p, y_p), we can find the conditional predictive distribution, $y_p|x_p, y, x$ using the fact that (y_p, y) are jointly normal along with the properties of the conditional distribution for the multivariate normal distribution. Given the abrupt changes in economic conditions brought on by the COVIV-19 pandemic, a smooth

interpolation function induced by the above kernel had difficult in interpolating all 10 years of economic data well. To address this, we split our economic data into pre- and post-pandemic data using the date March 1st, 2021. After doing this, Gaussian process regression managed to interpolate the data quite well as we can see in the figure below.



Having covered the necessary details, we can now introduce our model. Our problem consists of creating data that prescribe not only to temporal trends but trends across features as well. The TimeGAN addresses this problem by modeling across both dimensions of time and space. The model is composed of four networks: the embedder, recoverer, generator, and discriminator. The embedder maps the high-dimensional time series at each time step into a latent space. In so doing, the embedder learns a lower-dimensional representation and discovers relationships between the features. The generator and discriminator operate in this lower-dimensional space, playing the minimax game to generate realistic data. In addition, however, the generator is also given examples of the true data in the latent space. The generator performs supervised learning at each time step of the sequence to learn the conditional distribution of the next time point given the past time points. Since the generator operates at the level of each time step, the discriminator does as well, determining whether the vector at each time point is true or fake. This stands in contrast to the normal GAN framework in which the discriminator evaluates the sequence in aggregate, which may hinder the learning of the autoregressive component of the data. The recovery network then maps the low-dimensional representation back to the ambient space. The objective of the model then is composed of three components:

$$L_R = E_{x \sim p_{data}(x)}[\sum_{t=1}^{T} ||x_t - \tilde{x_t}||_2]$$

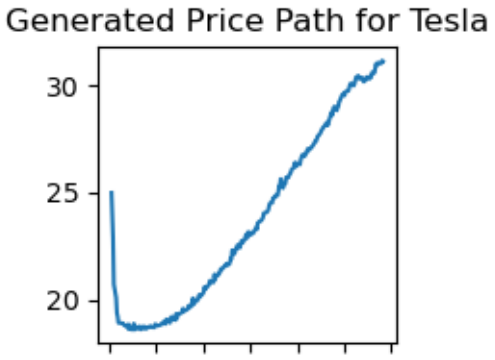
$$L_U = E_{x \sim p_{data}(x)}[log \sum_{t=1}^{T} y_t] + E_{z \sim p_z(z)}[log(\sum_{t=1}^{T} 1 - \hat{y_t})]$$

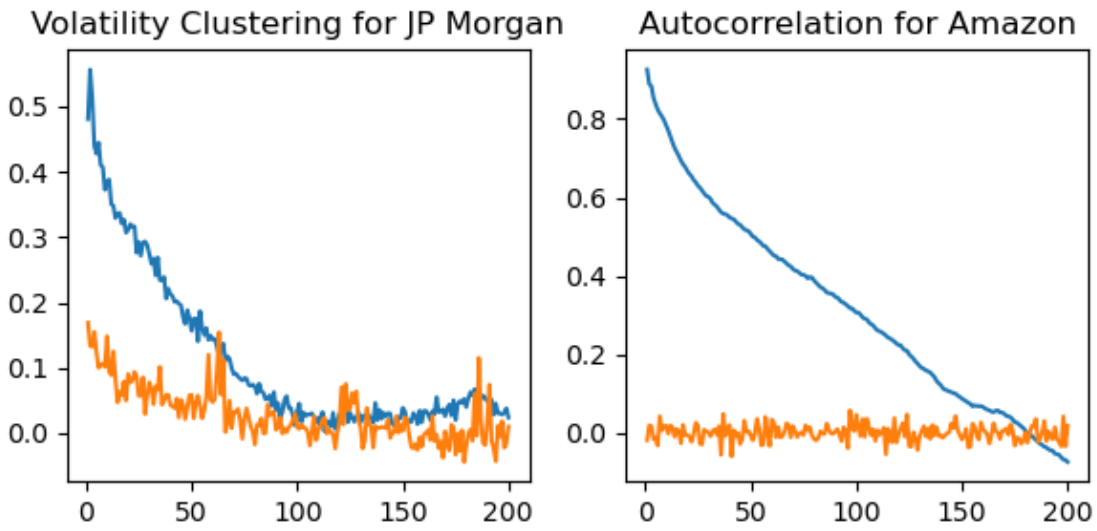
$$L_S = E_{x \sim p_{data}(x)}[\sum_{t=1}^{T} ||h_t - g(h_{1:t-1}, z_t)||_2]$$

where $\tilde{x_t}$ is the reconstruction of x_t, $y_t, \hat{y_t}$ are the labels and estimates for true and fake data, respectively, h_t is the latent representation of x_t, and z_t is a standard Brownian motion. L_R corresponds to the reconstruction loss, L_U is the usual objective for the minimax game of the generator and discriminator, and L_S is the supervised loss for the generator when given the latent representations of the true data. For all four networks, we will use a recurrent neural network architecture with gated recurrent units along with a feedforward output layer. In particular, we use two GRUs with 50 hidden units each. We will use GRUs here to better learn long-term dependencies while still using a smaller number of parameters than a long short-term memory (LSTM). Moreover, we will use a Leaky RELU(0.2) activation function as this function has been demonstrated to allow better learning of long-term dependencies.

For training, we first pretrain the embedding and recovery networks on the reconstruction loss before pretraining the generator using just the supervised loss. The four components are then trained jointly. For the embedding and recovery networks, the objective $\min_{\theta_e, \theta_r} \lambda L_s + L_R$. By including L_R, we encourage the embedder and recoverer to learn representations that assist the generator in identifying the temporal relationships in the data. For the generator and discriminator, we have $\min_{\theta_g}(\eta L_S + \max_{\theta_d} L_U)$. λ and η control the balance between the two losses, but empirically, we found that the model is not sensitive to these parameters; as a result, both have been set to 1. We have ten sequences of 245 time steps and will use the TimeGAN to generate ten sequences of 245 time steps as well. Each sequence is composed of the five log returns for each stock, the five corresponding trading volumes, and 14 economic indicators and has been standardized via min-max scaling. We should note as well that the latent space is 12-dimensional. Training was conducted using the Adam optimizer with a learning rate of 0.001 and beta parameters of 0.9 and 0.999 for 2000 iterations (for joint training and each pretraining phase) with a batch size of 3. Even with a relatively small batch size and number of iterations, training took several hours due to the time horizon of the data.

5 Results





	Tsla	Aapl	Jpm	Amzn	Goog
true mean	0.002	0.0011	0.0007	0.0011	0.0009
syn mean	-0.0024	-0.0042	-0.0047	0.0006	-0.0027
true alpha	3.22	3.98	3.74	3.52	3.8
syn alpha	4.11	9.09	1.94	2.45	5.34
true bond corr	-0.00595112	-0.570613	-0.719304	-0.555363	-0.72001
syn bond corr	0.093474	0.567045	0.440488	0.565131	0.550004
true unemploy corr	0.0173472	-0.0160959	-0.040835	-0.363327	-0.0892327
syn unemploy corr	0.242222	-0.943866	-0.882979	-0.959	-0.948461
true gdp corr	0.0125982	0.654677	0.797923	0.960065	0.866503
syn gdp corr	-0.287698	0.947001	0.917909	0.980322	0.976439

To analyze our results, we first examine some realized stock price paths produced by the generator. In the figure above, it seems like the path could represent an actual stock price; the price path captures the tendency of stocks to have sudden drops followed by a general upward trend. Upon closer look though, we can see that there seems to be a momentum effect, i.e. price increases are subsequently followed by more price increases. For closer inspection, we have plotted the autocorrelation function for the generated time series for Tesla compared to that of the true series. Clearly, the generated series is not observing the linear unpredictability property. The inability of the generated data to capture zero autocorrelation holds generally as well. Aside from this, the zero mean property is followed quite well. The volatility clustering property seems decent as it generally captures the levels observed in practice though with much higher levels for low time lags. With respect to the fat-tailedness property, the generated data's alpha parameters are close for some stocks, namely Tesla, Amazon, and Google, but for Apple, the generated data has learned a much lighter-tailed distribution. The converse can be said for JP Morgan. For our economic indicators, we observe the correct relationship for GDP per capita and unemployment (albeit with a much lower magnitude), but the completely wrong relationship for the bond interest rate.

With respect to the "train on synthetic, test on real" approach, we use an RNN with three layers of

GRU cells with corresponding hidden dimensions of 245, 150, and 100. The output is a feedforward layer with five outputs, one for each return. The model was trained using the Adam optimizer with the same parameters as discussed previously. For the true training set, we obtain a mean absolute error of just over 0.02 on the test set, the 2022 stock data. For the generated data, we obtain a mean absolute error of over 0.04. Moreover, the test set performance worsened for a large duration of the learning process. In other words, as the model learned more about the features of the generated data, the performance on the test set worsened. This is a clear indication that the generated data is different in meaningful ways from the true data.

With these results, we can see the effect of limited data within a higher-dimensional feature space. Takahashi managed to recreate the first four properties above with extreme precision without rigorously constructing the conditional distribution over time as our model does. The difference lies in that they used 24 sequences of 8,192 time steps with no additional features other than stock prices to train their GAN whereas we used ten sequences of 245 time steps with 19 additional features. Though not treating these properties rigorously like we have, just by visual inspection, Jarrett's model creates much more realistic data than ours. In this setting, the model is unable to learn both the relationships across time and features with limited data, even when we use a latent space to reduce the dimensions of the features. The implication is that if we do not have sufficient data, turning to GANs to perform data augmentation may create synthetic data that does not well approximate the data-generating distribution.

6 Conclusions

In sum, we have shown that even with a theoretically solid model with state-of-the-art empirical results, GANs can still fail to provide an accurate approximation of the true data. Unfortunately, the setting in which we would most like GANs to succeed is precisely the one in which GANs are most likely to fail. We did introduce an approach to determine the veracity and practical utility of generated financial data by comparing its relationships with general market conditions to the relationships exhibited by the true data. An obvious extension to our work would be to determine at which point GANs are able to perform sufficiently well. With varying datasets, this would never be a strict rule but would help determine when data augmentation is dubious and when it is efficient. To compare the generated and true data, we used the generated data as part of a regression-based supervised learning problem. We could have discretized the distribution of log daily returns (to do density estimation in a sense) and then treated this as a multi-class classification problem. Using the Kullback-Leibler divergence $D_{KL}(p||q) = \sum_{k=1}^{n} p_k \log(\frac{p_k}{q_k})$, we could have compared the two probability vectors generated by each model. This would provide a rigorous framework in which to compare models with proper accounting for uncertainty.

7 References

Koshiyama, N. Firoozye, P. Treleaven, Generative Adversarial Networks for Financial Trading Strategies Fine-Tuning and Combination, arXiv preprint arXiv:1901.01751, 2019.

Takahashi, S. Chen, Y. Tanaka-Ishii K. Modeling Financial Time Series with Generative Adversarial Networks. Physica A 527 (2019).

Salimans Tim, Goodfellow Ian, Zaremba Wojciech, Cheung Vicki, Radford Alec, and Chen Xi. 2016. Improved techniques for training GANs. In Advances in Neural Information Processing

Systems. 2234–2242.

Fekri MN, Ghosh AM, Grolinger K. Generating Energy Data for Machine Learning with Recurrent Generative Adversarial Networks. Energies. 2020; 13(1):130.

Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. 2018. RecGAN: recurrent generative adversarial networks for recommendation systems. In Proceedings of the 12th ACM Conference on Recommender Systems. 372-376.

Daniel Jarrett, Jinsung Yoon, Mihaela van der Schaar, "Time-series Generative Adversarial Networks," Neural Information Processing Systems (NeurIPS), 2019.

8 Appendix

For general economic indicators, the following were used:

Effective Federal Funds Rate: the rate banks charge each other for overnight loans to meet deposit requirements, serves as a measure of policy action by the Federal Reserve

3 Month Treasury Bill Rate and 1 Year Treasury Bill Rate: prevailing yield of 3 month and 1 year Treasury bonds; serves as a measure of short-term interest rates

5 and 10 Year Treasury Bond Rates: prevailing yield of 5 year and 10 year Treasury bonds; serves as a measure of intermediate term interest rates

30 Year Treasury Bond Rate: prevailing yield of 30 year Treasury bonds; serves as a measure of long-term interest rates

30 Year Mortgage Rate: prevailing average interest rate for a 30 year fixed rate mortgage; measure of long-term interest rates as well as quantitative easing for the Federal Reserve and overall health of the housing market

Real GDP Per Capita: total economic output divided by the midyear population in terms of chained 2012 dollars; serves as a measure of the economic cycle and overall health of the economy

Unemployment Rate: number of people employed divided by the number of people in the labor force; serves as an additional measure of the economic cycle and overall health of the economy

Consumer Price Index: price of a basket of goods and services; serves as a measure of inflation

Consumer Price Index for Gas: price level of gasoline and related products; serves as a measure of inflation as well as a proxy for foreign policy relations

Real Potential GDP: total potential economic capacity; serves as a measure of technological improvement and capital accumulation

Public Debt as Percent of GDP: ratio of total outstanding government debt to GDP; captures the spending and taxing policies employed by the federal government

Household Debt as Percent of GDP: ratio of total outstanding household debt to GDP; measures the borrowing habits of consumers and serves as a proxy for consumer confidence and the credit cycle