Scenario Generation and Data Augmentation

Raymond Luo, Ziheng Chen, Xiaobin Ou, Elaine Huang

1 Introduction

Statistical models aimed at risk management, such as Value at Risk (VaR), have proven largely ineffective in addressing various market movements amidst past crises. Historical data used to train models tend to be sparse in representing the whole set of possible outcomes and are biased against extreme yet possible events. Scenario analysis addresses many of the shortcomings of the statistical approach, since it is both forward looking and also able to draw on historical experience that might not be represented in the development samples for statistical approaches.

It follows that there is a need for more rigorous and systematic scenario analysis. It is important to point out that a key challenge in applying scenario analysis is to consider the extremely severe events that have yet to happen but are very much possible. Statistically, this refers to the need for constructing "tail event" scenarios that are severe but plausible. We can assess scenario's plausibility by reference to the behavior of observable variables used to define the scenario (e.g., asset prices, bond yields, credit spreads, etc.). To generate realistic scenarios in all relevant market regimes we can use use machine learning approaches to identify regimes. This investigation approaches these issues with scenario analysis by attempting to generate data through "data augmentation" algorithms used in machine learning (e.g., through transfer learning or generative adversarial networks GANs).

2 Methodology

We proceed our investigation in the following manner. We first recreate the results given by two major toolsets for scenario generation: the variational autoencoder (VAE) and the time series generative adversarial network (timeGAN) models. This procedure involves identifying the code syntax used by the respective authors of the models as well as updating the code for our own usage - some of the machine learning and tensorflow packages have deprecated due to the rapid development of deep learning packages. We then extrapolate the usage of the code onto our own choice of initial data. As equity indices are both abundant and diverse, we proceed with a selection of 10 years of various daily and weekly equity index data. For theoretical considerations of the financial behavior of stock returns under the efficient market hypothesis and the crucial rate of convergence for our deep learning models, we focus on normalized stock returns in data processing for the entirety of our investigation.

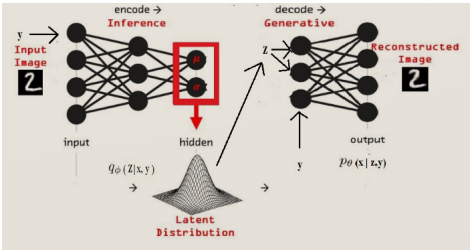
After establishing baseline models, we proceed in modifications in two ways. Firstly, we combine the two models in sequence to varying degrees in order to see whether our results are for better or worse. Secondly, we modify parameters in the VAE and timeGAN models to see if we have increased success in our results. These modifications include but are not limited to adding discrete jump processes to the baseline GAN data as noise or observing regime shift periods in the underlying data to add market shift imitations. For VAE data this could include adding variation to the path-signature of generated data paths to add noise.

3 Variational Autoeoncoders

3.1 Base Model Results

VAEs have historically found less success in comparison to GAN models due to the limited structure they impose on the generated data. One paper that has particularly shown great promise proposes the use of

Conditional Variational Autoencoders (cVAE) on path-signatures. Our first step aims to replicate the model proposed in the paper using an updated version of Tensorflow (the paper uses deprecated code from Tensor v.1.0) and to extrapolate the mathematical aspects to extend to our own data sets. The diagram below demonstrates the pipeline process for the cVAE model:



And the signatures for the paper are defined by:

Definition 3.1 (Signature)**.** Let $X : [0, T] \to^d$ d be a continuous path of bounded variation. Then the signature of X is defined by the sequence of iterated integrals given by:

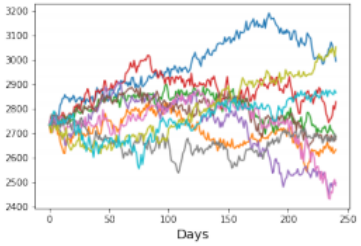
$$\mathbf{X}_T^{<\infty} := (1, \mathbf{X}_T^1, \cdots, \mathbf{X}_T^n, \cdots)$$

where

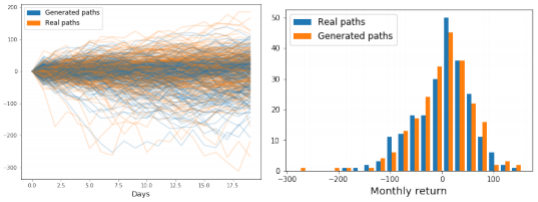
$$\mathbf{X}_T^n := \int_{0<u_1<\cdots<u_k<T} dX_{u_1} \otimes \cdots \otimes dX_{u_k} \in ({}^d)^{\otimes n}$$

.

This definition is a generalization of smooth paths from stochastic analysis with $\otimes$ being the tensor product. The key to this approach is to consider the signature a feature for the training of the cVAE model and the integral of the tensor product has a *postulated* value of $\mathbf{X}_T^n$.



We have recreated the results from the paper as follows:



4 Generative Adversarial Networks

The generative adversarial network was originally and mainly used for image generation, but in this section, we apply it to the time-series domain, and more specifically, to the generation of financial time series.

We implement the Time-series Generative Adversarial Networks (TimeGAN), which takes temporal dynamics of sequences into consideration, combining the unsupervised GAN framework with supervised training via adding an embedding space. In other words, in addition to minimizing the unsupervised loss, we also want to capture the stepwise dynamics in the data and minimize the corresponding supervised loss. Suppose we have static features ($\mathbf{S}$) and temporal features ($\mathbf{X}$) in our data set. And let $\mathcal{S}, \mathcal{X}$ be the vector spaces of static features and temporal features ($\mathbf{S} \in \mathcal{S}, \mathbf{X} \in \mathcal{X}$), respectively. We aim to approximate the joint distribution of $(\mathbf{S}, \mathbf{X}_{1:T})$, $p(\mathbf{S}, \mathbf{X}_{1:T})$, which may be difficult under the standard GAN framework. Therefore, we break down the objective into a sequence of stepwise objectives, and approximate $p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$ for each time t by

$$p(\mathbf{S}, \mathbf{X}_{1:T}) = p(\mathbf{S}) \prod p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$$

Thus we transform our goal from global,

$$\min_{\hat{p}} D(p(\mathbf{S}, \mathbf{X}_{1:T})||\hat{p}(\mathbf{S}, \mathbf{X}_{1:T}))$$

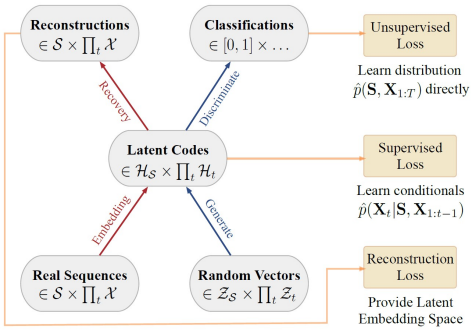
to local,

$$\min_{\hat{p}} D(p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})||\hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}))$$

where $\hat{p}(\cdot)$ is the learned density and D is a distance measure. The local minimization can

be better achieved since it only depends on ground-truth sequences.

The TimeGAN network consists of four steps: embedding function, recovery function, sequence generator and sequence discriminator. Embedding and recovery functions are the first two mappings between feature and latent space, where the embedding function takes static features ($\mathcal{S}$) and temporal features ($\prod_t \mathcal{X}$) to their latent spaces $\mathcal{H}_\mathcal{S}$ and $\prod_t \mathcal{H}_\mathcal{X}$, respectively, and the recovery function, on the contrary, takes static and temporal codes from the spaces back to features. Sequence generator and discriminator are the latter two networks operating within the latent space, where the generator takes static and temporal random vectors from vector spaces $\mathcal{Z}_\mathcal{S}, \mathcal{Z}_\mathcal{X}$ to the latent space, and the discriminator returns classifications (either real or synthetic data) by receiving static and temporal codes from latent space. The bellow diagram shows the interactions and objectives of the TimeGAN functions.



4.1 Data Processing

In the current stage, we constructed our basic portfolio using AAPL, AMZN, DIS, JNJ, JPM and MSFT daily prices, and then calculated the daily return as our dataset. We used data between the period 2001/01/01 and 2021/06/18. After obtaining returns, we used MinMaxScalar to normalize the returns, and then cut the data by sequence length. Then we mixed data via random permutation to make it similar to independent and identically distributed random variables, then it is ready for model training.

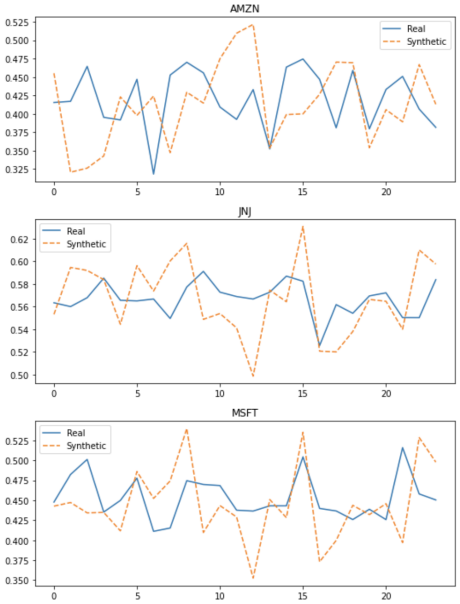
4.2 GAN Model Building

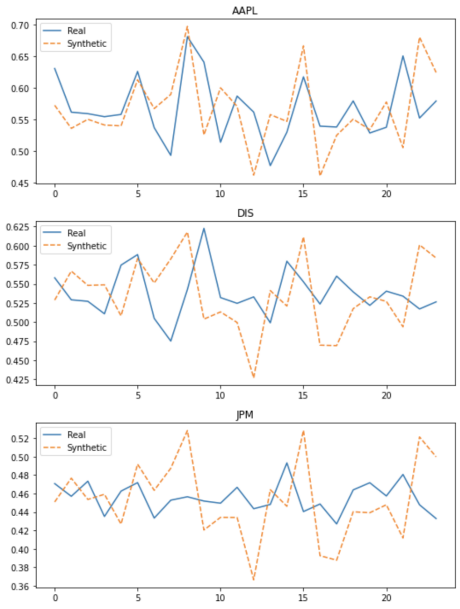
We use TGAN to generate synthetic time-series data, which is based on RNN networks. It is consisted of four elements which are generator, discriminator, embedder and recovery network. In the model, the architectures of each element would be optimized and tailored to the dataset. Here we defined sequence length, number of sequence, hidden dimension, gamma, noise dimension, batch size, log step, and learning rate. After finishing defining model hyperparameters, we trained the timeGAN synthetizer using 50000 train steps. It took around two hours to finish training in the current stage.

Hyperparameters	value
sequence length	24
number of sequence	6
hidden dimension	24
noise dimension	32
batch size	128
log step	100
gamma	1
learning rate	$5e^{-4}$

4.3 GAN Model Results

After model training was done, we plotted some generated samples, which included both synthetic and original data standardized with values between [0,1]. The results are shown below.





4.4 GAN Model Evaluation

To evaluate the generated synthetic data, we used PCA and TSNE. For the sake of comparing them visually, we need the data to be 2-Dimensional. For that reason we used only two components for both the PCA and TSNE. The results are shown below.

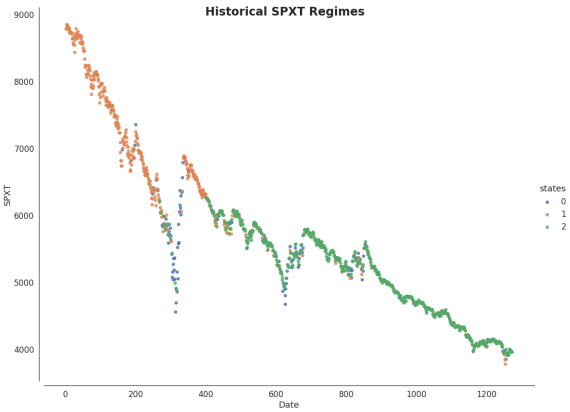
Then we trained on synthetic datset and tested on real dataset to further evaluate the model. We first implemented a simple RNN model for prediction. To prepare the dataset for the regression model, we splited data on train dataset and test dataset, and then we defined s and y for both the synthetic and real datasets. Finally, we trained the model with both the synthetic data and the real train data with parameters epochs 200, batch size 128, and callbacks to be early stopping. We summarize the metrics and the result is shown below.

	r2	MAE	MRLE
Real	0.906701	0.012914	0.000629
Synthetic	0.913467	0.018099	0.000715

5 Evaluation of Real Time Series

An initial data analysis that we can provide that is an analysis of the market regimes that the real data goes through. This allows us to understand some interesting perspectives: what is a reasonable length of a market regime, what are the statistical qualities that shift during a market regime, and etc. We attempt to observe these market regimes tentatively with use of Hidden Markov Models (HMMs) with a varying parameter on the number of regimes to segment our historical data into various data regimes. We will continue to conduct a similar analysis on the generated data to see how our synthetic data segments into various regimes. The idea here is that a synthetic data that has few and sparse regime shifts is less representative of realistic data and the possible scenarios in reality.

Alternatively, we can also use the metrics in our market regimes to determine the qualities that our various asset groups (from various styles of equities to other asset classes) that are shared within the groups. This allows us to identify key parameters to hold for future data-augmentation models and which qualities are to be tested for our evaluation. The diagrams below display an basic analysis of one of the training sets in a 20 year period for 3 market regimes.



6 Evaluation of Synthetic Data

6.1 Methodology of Evaluation

One of the foremost challenges to generating synthetic data is the evaluation of whether the data is ”good”. Scenarios need to be realistic enough to be a possible alternative scenario but also different enough to decrease the bias in training existing real data samples.

Here, we try to evaluate whether a time series from the aspect that whether synthetic time series exhibit similar stylized features as real data. These are some financial time series stylized features:

1) Non-stationarity. Future returns can behave quite different from history. We could not expect financial return series is a statistical stationary series.

2) Fat-tails. Financial returns are not in the form of normal distribution. They always have fatter tails, which means there are more extreme events happening.

3) Volatility clustering. Financial time series tend to have clustering volatility, which means another high volatility period is likely to happen after a previous volatility period.

4) Market Regime. Financial time series could behave totally different in different market regimes. In crashing regimes, for example, S&P index drop over 10 percent in month, the correlations among stocks could be really high, while in booming regimes, there are nearly no correlations among these stocks.

Using the statistics which can describe the above features, combined with some other statistical features, such as first, second and third moments. We can use a set of statistics to describe time series. Using these real data and data with different statistical features, such as white noise, we train a classifier which can tell whether a time series is real. After that, we let the classifier to tell whether synthetic data looks real. If the classifier always classify synthetic data as real, we think our synthetic data show similar stylized features as real financial times series data.

7 Conclusion

8 Future Plan

We will construct a more diverse portfolio including stocks of different industries, bonds, and other derivatives. We will further improve GAN model via using parameters containing additional information.

9 Contribution

Raymond Luo: VAE model building, evaluation of real time series

Ziheng Chen: VAE model building, evaluation of synthetic time series

Elaine Huang: TimeGAN model building, Evaluation of synthetic data using PCA, TSNE and TSTR

Xiaobin Ou: data download and processing, cooperate in TimeGAN model building, model training and data generating

References

https://github.com/vanderschaarlab/mlforhealthlabpub
https://github.com/jsyoon0823/TimeGAN
https://github.com/imanolperez/market_simulator/tree/master/notebooks