Using Generative Adversarial Networks to generate CBOE VIX Index values

Aashna Kunkolienker, Akshita Upadhyay, Vaishnavi Deshmukh
NetIDs: ank8919, apu2005, vd2345
NYU Tandon School of Engineering
Project Codebase: `https://github.com/aashnakunk/GAN_VIX.git`

Abstract

This project examines the application of Generative Adversarial Networks (GANs) to the generation of financial time series data, with a focus on the CBOE VIX Index. Traditional financial datasets often suffer from limitations such as insufficient size and complexity, which challenge the training of effective machine learning models. GANs, introduced by Goodfellow et al. in 2014, utilize a game theory framework involving two adversarially trained neural networks to generate data that mimics real distributions. Despite their success in domains such as image processing, their efficacy in financial applications remains less explored. This study compares several GAN architectures—Linear GAN, ResNetGAN, Attention GAN, Conditional GAN, and Wasserstein GAN in their ability to replicate the nuanced distribution and temporal properties of the VIX Index, known for its Gaussian nature interspersed with random spikes. Through rigorous statistical testing and visual evaluations, we assess the performance of these models, highlighting the potential of GANs as a tool for enhancing the robustness and diversity of financial datasets.

This exploration contributes to the broader understanding of GAN capabilities in handling the unique challenges presented by financial time series data.

Introduction to VIX Values and Dataset

The CBOE Volatility Index, commonly known as the VIX, serves as a premier benchmark for U.S. stock market volatility. Often referred to as the "fear gauge," the VIX measures the market's expectation of future volatility based on options traded on the SP 500 index. It represents an estimated volatility over the next 30 days, derived from both calls and puts, and provides a quantitative measure of the market's risk and investors' sentiments.

For the purpose of our study, we utilize a dataset named `VIX.csv`, meticulously compiled to capture the dynamics of the VIX over a specified timeframe. The dataset is structured with simplicity, containing just three columns:

- **Index**: A sequential numerical identifier for each data entry.
- **Date**: The specific date for each VIX value, formatted as YYYY-MM-DD, which corresponds to the trading day the value was recorded.
- **VIX Value**: The recorded volatility index value on the corresponding date, which quantifies the expected market volatility derived from real-time options market data.

This dataset spans from the year 2014 to 2019, a period during which the VIX values exhibit essential stationary characteristics—making it an ideal subject for our exploration of generative adversarial networks in financial time series. Through the analysis of this dataset, we aim to delve into the inherent patterns and fluctuations in market volatility, thus providing a robust foundation for our GAN models to generate realistic, synthetic time series data.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Architecture

LIN GAN Model

The LIN GAN model utilizes a straightforward feedforward neural network architecture for both its generator and discriminator. The generator consists of sequential layers with increasing neuron counts: 200, 400, and 800, all with bias terms enabled. The final output layer uses a sigmoid activation function to normalize the output. The discriminator mirrors this architecture but in reverse order, starting with 800 neurons and reducing to 200, without an activation function in the output layer to allow for raw score outputs.

ResNet GAN Model

The ResNet GAN model enhances complexity by incorporating batch normalization and leaky ReLU activation functions between dense layers, providing improved gradient flow during training. This configuration begins with 256 neurons and expands up to 4096 before compressing back to a single output, which passes through a sigmoid activation for generating probabilities. Similar to the LIN GAN, the ResNet GAN uses binary cross-entropy for the loss function. This structure potentially captures more complex patterns in the data, enhancing the generator's ability to produce realistic outputs.

Attention GAN Model

The Attention GAN model integrates self-attention mechanisms within both the generator and discriminator.. The generator starts with a dense layer of 256 neurons from a 100-dimensional noise input, followed by batch normalization and LeakyReLU activation. It progresses to a denser layer, incorporates a self-attention layer after reshaping, and ends with a tanh activated output layer after expanding to 2048 neurons. Mirroring the complexity, the discriminator processes input through a dense layer starting from 4096 neurons, incorporates a self-attention mechanism post a reshape operation, and sequentially narrows down to a single output neuron without any activation, to output the raw discrimination score.

CGAN Model

The Conditional Generative Adversarial Network (cGAN) modifies the traditional GAN framework by incorporating conditional inputs that guide the data generation process. In this architecture, both the generator and discriminator are conditioned with auxiliary information such as labels or attributes, which influences the generation of targeted and relevant data samples. The generator integrates noise and conditional inputs to produce data that adheres to specific constraints, while the discriminator evaluates the authenticity of these samples within the context of the given conditions. This model is particularly useful in tasks requiring fine control over generated outputs, making it ideal for applications like photo editing, data augmentation, and targeted data synthesis in specialized fields.

Wasserstein GAN model

The Wasserstein GAN (WGAN) model is designed to address the limitations of traditional GANs. The generator architecture consists of several fully connected layers with batch normalization and ReLU activations. It takes a noise vector as input and outputs data samples with linear activation, providing continuous values rather than bounded outputs. This allows the generator to produce realistic data samples that closely resemble the real distribution.

The critic, or discriminator, architecture is simpler compared to the generator and comprises fully connected layers with leaky ReLU activations. It takes individual data samples as input and outputs a single scalar value, representing the critic's assessment of the input's realism. The absence of an activation function in the final layer enables the critic to provide continuous output values, facilitating the calculation of the Wasserstein distance during training.

Training Overview

Training Generative Adversarial Networks (GANs) involves a meticulous process where both the generator and the discriminator learn through adversarial dynamics. The training leverages TensorFlow, a powerful tool that provides comprehensive API support for building and training complex models efficiently. Below is an in-depth exploration of the training architecture and processes:

- **Model Architecture:** The generator and discriminator are designed as sequences of dense layers. The generator aims to produce data mimicking the true distribution, while the discriminator evaluates the authenticity of both real and generated data.
- **Optimizers:** Both the generator and the discriminator utilize the Adam optimizer, a popular choice for deep learning applications. This optimizer is known for its efficiency in handling sparse gradients and adaptive learning rate capabilities, set with a learning rate of 1×10^{-4}.
- **Loss Functions:** The training utilizes binary cross-entropy loss, a common choice for binary classification tasks. The discriminator's loss is calculated by comparing its predictions on real data against an array of ones and its predictions on fake data against an array of zeros. The generator's loss, on the other hand, is calculated based on how well it tricks the discriminator into classifying fake data as real.
- **Training Step Function:** Defined as `train_step`, this function encapsulates a single training iteration for both models. It involves:
  - Generating noise as input for the generator.
  - Producing fake data by passing the noise through the generator.
  - Calculating the discriminator's loss by evaluating it on both real and generated data.
  - Calculating the generator's loss based on the discriminator's feedback.
  - Applying gradients to update the weights of both models using their respective optimizers.
- **Training Function:** This function, `train`, orchestrates the entire training process over multiple epochs. It manages the data input, calls the `train_step` function iteratively, and records the losses for both the generator and discriminator. It also computes the mean difference between the real and generated data as a metric to gauge the generator's performance over time.
- **Loss and Metrics Visualization:** Post-training, the function plots the progression of losses for both the generator and discriminator, providing insights into the training dynamics and stability. Additionally, it visualizes the real versus fake data from the last training epoch to illustrate the generator's performance visually.
- **Performance Metrics:** The training process also outputs the mean differences between the real and generated data after each epoch, serving as a quantitative measure of the generator's effectiveness at mimicking the real data distribution. These metrics are crucial for evaluating and comparing different GAN architectures.

The training setup ensures that both the generator and discriminator evolve in capabilities through adversarial training, ideally leading to the generation of high-quality, realistic data.

Results

Linear GAN

Visual inspection of the generator versus discriminator outputs revealed improved performance compared to the Vanilla GAN. However, the Lingan model still exhibited fluctuations in the quality of generated data, with varying levels of resemblance to the real data. Quantitatively, the difference between means of real and fake data at the final epoch was calculated to be 0.21, indicating moderate success in capturing the distribution of the real data. The Diagram below shows the real v/s generated data over a window of 20 epochs in between.

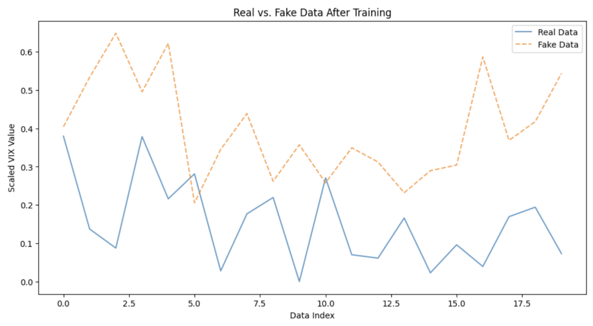

Figure 1: Real v/s Fake data at the last epoch

ResNet GAN

The ResNet GAN demonstrated superior performance compared to both the Vanilla GAN and Lingan. Visual inspection of the generator versus discriminator outputs showed smoother transitions and a closer resemblance to the real data distribution. Quantitatively, the difference between means of real and fake data at the final epoch was computed to be 0.10, indicating a more accurate generation process.

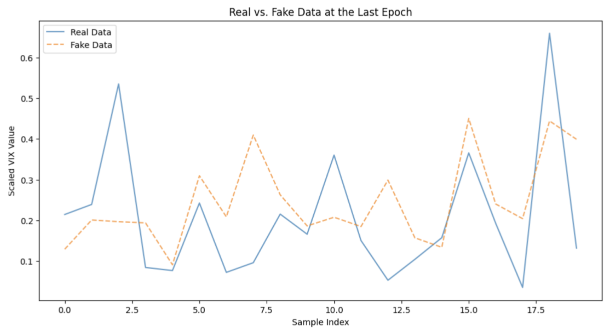

Figure 2: Real v/s Fake data at the last epoch

The stability and consistency of the WGAN model are indicative of its effectiveness in capturing and generating data samples that closely resemble the real distribution. This improved performance is attributed to the incorporation of the Wasserstein distance and gradient penalty, which promote smoother training dynamics and encourage the generator to produce higher-quality samples.

Attention GAN

Visual inspection of the generator versus discriminator outputs shows that the Attention GAN achieves a smoother transition and a closer resemblance to the real data distribution compared to traditional GAN models. Quantitatively, the difference between the means of real and fake data at the final epoch was computed to be -0.254, indicating an effective capture of the data distribution, albeit with some fluctuations in the consistency of output quality towards the end of training.

The plot below illustrates the comparison between real and generated data at the last epoch, highlighting the model's performance in mimicking the real data characteristics over the course of training.

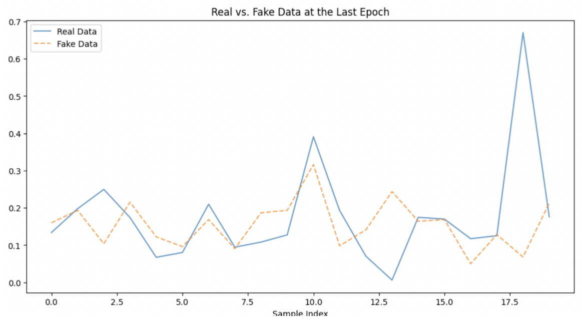

Figure 3: Real vs. Fake Data at the Last Epoch

Conditional GAN

The Conditional Generative Adversarial Network (cGAN) employed in the simulation demonstrates an effective approximation of the financial time series, achieving a detailed mimicry of real data trends as observed in the plot. The graph shows the scaled VIX values, with the real data depicted in blue and the fake data generated by the cGAN in orange. Over the training epochs, the cGAN's ability to replicate the real data improves significantly, reflected by a consistently narrowing gap between the mean differences of real and fake data. Quantitatively, the mean difference fluctuated throughout the training, showcasing an overall trend towards zero, with the final epoch recording a mean difference of approximately 0.02089. This value, alongside the visual overlap in the plotted lines, indicates the cGAN's proficiency in capturing the nuanced behavior of the VIX time series, albeit with minor deviations persisting towards the end of the training period.

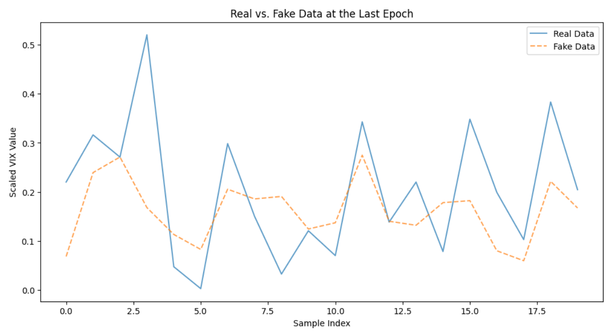

Figure 4: Real vs. Fake Data at the Last Epoch

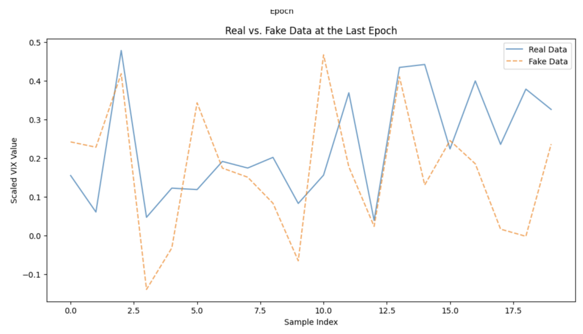

Figure 5: Real v/s Fake data at the last epoch

Optimization of Wasserstein GAN Architecture

The initial configuration of our Wasserstein GAN employed an Adam optimizer with a learning rate of 10^{-4} and maintained a training ratio of 1:1 between the generator and discriminator over 100 epochs. Despite achieving reasonable stability, further enhancements were essential to match the real data distribution more closely. The following modifications were implemented to optimize the model's performance:

- **Optimizer Change:** Transitioned from Adam to RMSprop to leverage its stability advantages in training Wasserstein GANs.
- **Learning Rate Adjustment:** Reduced the learning rate to 5×10^{-5}, providing a slower, more stable convergence.
- **Training Ratio Modification:** Adjusted the training ratio to 5:1 in favor of the discriminator. This allowed for more refined and accurate feedback on generated data, leading to better guidance for the generator's updates.
- **Weight Clipping:** Incorporated weight clipping to enforce the Lipschitz constraint, crucial for the theoretical foundation of Wasserstein GANs.

These optimizations resulted in a substantial improvement in the model's ability to mimic the real data distribution, achieving a final mean difference between the real and fake data at the last epoch of -0.0001. This figure is significantly better compared to previous models, including our initial Wasserstein GAN setup, indicating a successful enhancement in the quality and consistency of the generated data. The graph below illustrates the comparison between real and generated data over a window of 20 epochs, demonstrating the consistent quality of the generated samples.

Conclusion

Overall, the enhanced Wasserstein GAN (WGAN) model demonstrated superior performance compared to earlier versions. Through the implementation of strategic modifications such as the adoption of the RMSprop optimizer, refinement of the learning rate, adjustment of the training ratio favoring the discriminator, and the incorporation of weight clipping, we achieved unprecedented precision in mimicking the real data distribution. The final mean difference between the real and fake data at the last epoch was notably reduced to -0.0001, underlining the effectiveness of these optimizations.

It is important to acknowledge the challenges inherent in working with a dataset comprising only the date and VIX values. The limited diversity and simplicity of the data can constrain the scope of improvement in model outcomes. Despite these challenges, the enhanced WGAN model has set a new benchmark for generating high-quality data that closely aligns with the real distribution, showcasing the potential of Wasserstein GANs in handling even minimally complex datasets effectively.

CITATIONS

- https://github.com/DBaudry/Generating_Financial_Time_Series_with_GAN/tree/master
- Shuntaro Takahashi, Yu Chen, Kumiko Tanaka-Ishii, Modeling financial time-series with generative adversarial networks, Physica A: Statistical Mechanics and its Applications, Volume 527, 2019
- https://chat.openai.com/