Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Магистерская программа Финансовые технологии и анализ данных

КУРСОВАЯ РАБОТА

На тему (рус) Моделирование высокочастотных финансовых временных рядов с помощью глубоких генеративных моделей
На тему (англ.) Modeling high-frequency financial time series using deep generative models

Студент
Попов Никита Сергеевич

Руководитель КР:
Канд. экон. наук
Науменко Владимир Викторович

МОСКВА
2024

Оглавление

ВВЕДЕНИЕ

В последнее время, во многом благодаря языковым моделям типа ChatGPT, интерес к генеративным моделям сильно вырос. Они позволяют генерировать данные различной структуры: тексты, картинки, таблички. Обучение генеративных моделей отличается и существенно сложнее обучения дискриминативных моделей. Это объясняется тем, что последние работают с намного более простыми распределениями. Например, предсказать вероятность конкретного вида животного по изображению намного проще, чем сгенерировать это животное. Основная задача генеративных моделей – приближение распределения данных и генерация новых данных. Смоделировать распределение генеральной совокупности можно двумя способами.

1. Явное моделирование. Здесь явно находится плотность p(x) распределения данных и из нее сэмплируются объекты. Примеры: авторегрессионные, диффузионные модели, вариационные автокодировщики.

2. Неявное моделирование. Здесь у нас нет доступа к плотности, однако мы можем делать из него сэмплы. В основном это генеративно-состязательные сети.

Генераторы данных полезны, когда реальные данные не доступны в достаточном количестве, либо не могут быть использованы из соображений конфиденциальности, либо при необходимости моделирования ситуаций, еще не встречавшихся в реальности, исключительных случаев.
В области временных рядов генерация синтетических данных является сложной задачей из-за временных закономерностей в данных. Генеративный процесс должен охватывать как распределение признаков,

так и временные отношения. Методы глубокого обучения особенно хорошо подходят для моделирования таких сложных конструкций. Применение таким моделям можно найти, например, на фондовом рынке, в задачах предсказания стоимости недавно выпущенных бумаг. Генеративные модели помогут обогатить небольшой набор имеющихся данных. В данной работе будет исследовано качество генерации моделей минутных свечек с помощью различных архитектур VAE и GAN.

1. Модели генерации временных рядов.

1.1. TimeVAE

Сначала коротко рассмотрим принцип работы обычного VAE. VAE, описанный Kingma, Welling (2013), предполагает, что наблюдаемые данные генерируются в результате двухэтапного процесса. Сначала значение z генерируется из некоторого распределения $p_\theta(z)$, далее, значение x генерируется из некоторого другого распределения $p_\theta(x|z)$. Эти распределения нам неизвестны, однако мы предполагаем, что они дифференцируемы относительно θ и z. Отношения между входными данными и скрытым представлением следующие: — $p_\theta(z)$ - априорное распределение, $p_\theta(x|z)$ - правдоподобие, $p_\theta(z|x)$ – апостериорное распределение. Вычисление $p_\theta(z)$ довольно ресурсозатратно. Чтобы преодолеть это, VAE вводит некоторую аппроксимацию апостериорного распределения: $q_\phi(z|x) \approx p_\theta(z|x)$. Энкодер моделирует $q_\phi(z|x)$, декодер - $p_\theta(x|z)$. Распределение z мы выбираем сами как стандартное нормальное. В процессе обучения добавляем в функцию потерь KL-дивергенцию между z и стандартным нормальным распределением. Это позволяет нам удобно генерировать данные просто подавая числа из $N(0, 1)$ в декодер. В вариационных автокодировщиках используется следующая функция потерь:

$$L_{\theta,\phi} = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q\phi(z|x)||p_\theta(z))$$

Первая часть – отрицательное логарифмическое правдоподобие, отвечает за восстановление данных, вторая часть – KL-дивергенция между латентным пространством и априорным распределением.

Энкодер базового TimeVAE состоит из стандартных слоев: одномерные свертки, многосвязные слои. В статье(1) в декодер так же добавляется некоторая интерпретируемость: параллельные блоки, которые

представляют различные временные структуры, которые суммируются для получения итогового результата. Полная аритектура из статьи изображена на рисунке 1.1.

Блок тренда: N -размер батча, D - размерность фичи, T - количество точек времени. Зададим максимальную степень полинома P. Дальше тренды как полиномы степеней $p = 0, 1, 2, .., P$ моделируются в 2 шага. Сначала латентный вектор z используется для оценки коэффициентов базисного разложения Θ_{tr}. $\Theta_t r$ имеет размерность $N \times D \times P$. Θ_{tr} нужен для восстановления тренда V_{tr} в исходный сигнал. Реконструкция тренда в сигнале в матричной форме выглядит следующим образом: $V_{tr} = \Theta_{tr} \times R$. $R = [1, r, ..., r_P]$ матрица стапеней r, где $r = [0, 1, 2, ..., T - 1]/T$ - вектор времени. V_{tr} имеет размерность $N \times T \times D$. Таким образом матрица Θ_{tr} интерпретируется как тренд. Ее з начения определяют 0-й, 1-й, ..., P-й порядок тренда для каждого семпла(из N) и признака(из D).

Блок сезонности: пусть S - число различных сезонных паттернов, которые будем моделировать. Каждый паттерн j определяется двумя параметрами: m - число сезонов и $d -$. Например, для сезонности дня недели для дневных данных $m = 7, d = 1$
Аналогично блоку тренда для компонента j вычисляется в 2 этапа. Сначала используем латентный вектор z для оценки Θ^j_{sn} (размерности $N \times D \times m$). Далее индексируем элементы Θ^j_{sn} соответствующие конкретной сезонности для каждого временного шага для получения сезонного паттерна V^j_{sn}(размерности $N \times T \times D$). Финальная оценка сезонности V_{sn} - поэлементная сумма V^j_{sn} по $j = 1, 2, \cdots, S$
В сезонном блоке j, матрицу V^j_{sn} можно итерпретировать как один сезонный паттерн для сэмпла(каждого из N) и признака(каждого из D). Всего их m штук.

Финальный выход декодера - поэлементная сумма выхода блока трнеда V_{tr}, выхода блока сезонности $V^j_{sn}, j = 1, 2, \cdots, S$, и выхода residual Base Decoder. На картинке можно увидеть схему нейронной сети.

Для обучения используется функция потерь ELBO, определенная

ранее с одной модификацией: добавляется вес ошибки реконструкции, который увеличивает или уменьшает акцент на потерях реконструкции по сравнению с потерями KL-дивергенции между аппроксимированным апостериорным $q_\phi(z|x)$ и априорным $p_\theta(z)$. Весовой коэффициент ошибки реконструкции может быть выбран путем визуальной проверки качества сгенерированных выборок.

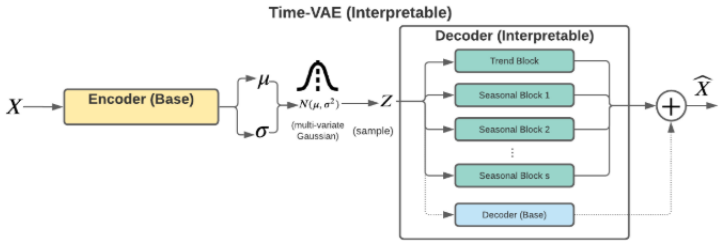

Figure 2: Block diagram of components in Interpretable TimeVAE

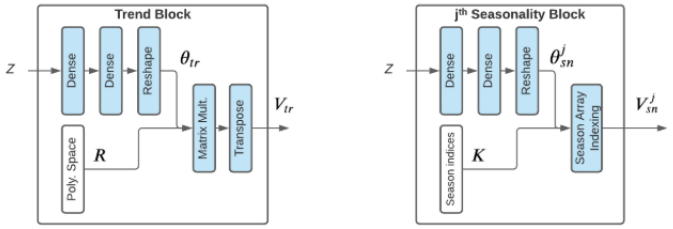

Figure 3: Trend and Seasonality Blocks in Interpretable TimeVAE

Рис. 1.1: Трендовый и сезонный блоки, рисунок из статьи

1.2. GAN модели

Как было отмечено, модели GAN не оценивают в явном виде плотность распределения данных. GAN вдохновлен теорией игр: две модели, генератор и дискриминатор, конкурируют друг с другом посредством состязательного процесса обучения. Пусть у нас есть генератор G_θ с параметрами θ, и дискриминатор $D\phi$ с параметрами ϕ. Гереператор отображает латентные векторы в векторы размерности исходных данных $\hat{x} \sim q(x)$, распределение которых приближает реальное распределение данных $p(x)$, так, чтобы приблизить распределение исходных данных. Дискриминатор решает задачу бинарной клиссификации: объекту x ставит в соответствие веростность $D(x)$, того, что этот объект реальный (сгенерирован из $p(x)$). Это можно сделать с помощью бинарной кросс-энтропии. Причем после некоторых

математических трюков можно записать одну лосс-функцию для обеих задач, просто генератор ее минимизирует, а дискриминатор - максимизирует.

$$\min_{\theta} \max_{\phi} E_{x \sim p(x)}[\log D_{\phi}(x)] - E_{z \sim p(z)}[\log (1 - D_{\phi}(G_{\theta}(z)))]$$

1.2.1. TimeGAN

TimeGAN состоит из 4 компонент: функции перевода в латентное пространство и восстановление, генератор последовательности, дискриминатор последовательности. Блок-схема модели и схема обучения приведены на рисунке 1.2. Разница с обычным GAN в том, что для построения эмбеддингов и генерации последовательностей нужно использовать эмбеддинги предыдущих элементов, чтобы учесть временную структуру. В модели для этого используются реккурентные сети. TimeGAN одновременно учится делать энкодинг, генерировать представления и итерироваться по времени.

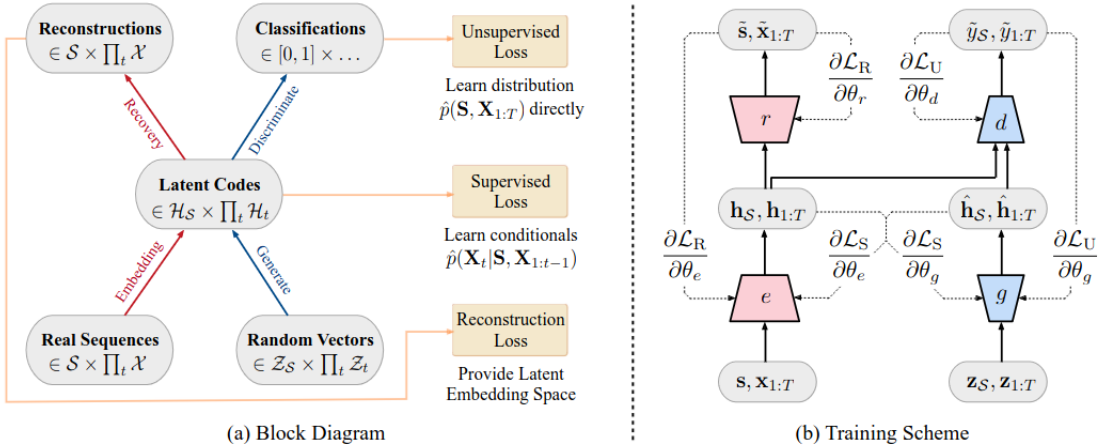

Figure 1: (a) Block diagram of component functions and objectives. (b) Training scheme; solid lines indicate forward propagation of data, and dashed lines indicate backpropagation of gradients.

Рис. 1.2: Устройство TimeGAN из статьи

1.2.2. CGAN

CGAN - модель генерации по уловию. Условие может быть как категориальным, так и непрерывным. Также как и в обычном GAN GAN, CGAN - minmax соревнование между генератором и

дискриминатором. Разница в том, что на вход генератору подается также некоторый вектор - условие, который просто конкатенируется с латентным вектором. Таким образом мы можем, наприемр генерировать следующие точки многомерного ряда при условии предыдущих, получая как бы предсказательную модель. В качестве декодера и энкодера изначально брались просто полносвязные слои. Но также есть реализации с использованием LSTM слоев, которые способны извлекать как краткосрочные, так и долгосрочные зависимости благодаря большому количеству путей течения данных.

1.2.3. CWGAN

CWGAN использует Wasserstein distance. Это метрика на пространстве вероятностных мер в метрическом пространстве. (функция расстояния, определенная между двумя распределениями вероятностей P и Q в заданном метрическом пространстве M). Интуитивно, если каждая мера измеряет распределение «грунта» по метрическому пространству M, то расстояние Васерштейна измеряет минимальную стоимость преобразования одного распределения грунта в другое, в простейшем случае предполагается, что стоимость прямо пропорциональна количеству грунта и расстоянию, на которое его надо перетащить. Из-за этой аналогии эта метрика известна в информатике как earth mover's distance

Для двух эмперических распределений P с выборкой $p_1, p_2, \cdots, p_n$ и Q с выборкой $q_1, q_2, \cdots, q_n$ расстояние Васерштейна определяется как:

$$W(P,Q) = \inf_{\gamma \in \prod(P,Q)} E_{(p,q)\sim\gamma}[||p-q||]$$

$\prod(P,Q)$ обозначает множество всех совместных распределений $\gamma(p,q)$, предельные распределения которых - P и Q. Для конкретных p, q совместное распределение $\gamma(p,q)$ означает как много массы нужно перенести из p в q чтобы превратить P в Q. Интуитивно, одно распределение можно рассматривать как правильно разбросанную массу, в второе - дыры в том же пространстве. Тогда расстояние Вассерштейна измеряет наименьший объем работы, нужный, чтобы засыпать ямы землей.

GANы сложно обучать, часто возникают проблемы сходимости моделей, такие как mode collapse или vanishing gradients. Как раз для борьбы с такими проблемами WGAN использует расстояние Вассерштейна в качестве функции потерь. Обучение становится более стабильным, становится проще отлаживать и искать оптимальные гиперпараметры. Функция потерь WGAN строится с использованием двойственности Канторовича-Рубинштейна как:

$$\min_G \max_{D \in \mathbf{D}} E_{x \sim P_r}[D(x)] - E_{\hat{x} \sim P_g}[D(\hat{x})]$$

где $\mathbf{D}$ функции, удовлетворяющие условию Липшица, P_r -реальное распределение, P_g - модельное.
Чтобы заставить выполняться условие Липшица есть 2 способа: weight clipping, gradient penalty. В CWGAN используется weight clipping. CWGAN также принимает условие y на вход и генератора и дискриминатора(который в данном случае уже не просто бинарный классификатор). Итоговый лосс запишется в виде:

$$\min_G \max_{D \in \mathbf{D}} E_{x \sim P_r}[D(x, y)] - E_{\hat{x} \sim P_g}[D(\hat{x, y})]$$

.

1.2.4. Conditional Signature Wasserstein GAN

Вместо использования нейронных сетей для классификации в дискриминаторе Signature Conditional Wasserstein GAN вычисляет расстояние между сигнатурами реальных и синтетических путей. Сигнатура S последовательности временного ряда X представляет собой нелинейную функцию $S(X)$, достаточную для того, чтобы уловить определенные особенности временного ряда. Модель обучается генерировать будущие сигнатуры по предыдущим. Генератор принимает шум Z и предыдущие p точек: $X_{gen} = G(Z, X_{t,-p:0})$. Сгенерированный семпл имеет сигнатуру $S(X_{gen})$. Цель генератора сравнить ожидаемую будущую сигнатуру синтетических данных $E(S(X_{gen}))$ и обученную будущую сигнатуру реальных данных $S_{t,1:q}$ и минимизировать

$(E(S(X_{gen})) - S_{t,1:q})^2$.

Функция сигнатуры S задана. Первым шаг - обучение функции прогноза сигнатуры L. Второй шаг — оптимизировать генератор G с учетом функции потерь, рассчитанной на основе сигнатур реальных и синтетических данных. S и L получаются при настройке модели генератора G.

2. Сравнение VAE и GAN без условия.

Для экспериментов был использован открытый датасет https://www.kaggle.com/datasets/nickdl/snp-500-intraday-data с минутными свечками по акциям SNP-500 в период с 2017-09-11 по 2018-02-16.

Для начала посмотрим два подхода для генерации данных без условия. Сгенерируем синтетические данные с помощью моделей TimeVAE и TimeGAN. Причем TimeVAE двух реализаций - с блоком тренда(блок сезонности делать бессмысленно, тк данные минутные) и без блока тренда(по сути - просто VAE). Для данного пункта возьмем данные по акциям Apple, Nvivia и Oracle. Пример свечек на рисунке 2.1, также есть данные по объему. Будем генерировать такой многомерный ряд(размерность = 5) после нормировки.

Рис. 2.1: Минутные свечки по акциям Apple

1. Для количественной меры сходства реальных и сгенерированных данных обучаем модель классификации апостериорных временных рядов. В качестве модели берем двухслойный LSTM. Помечаем единичками настоящие ряды и нулями сгенерированные. Считаем метрику $accuracy - 1$ на отложенной выборке. Чем ближе она к нулю, тем сложнее отличать реальные и сгенерированне данные.

	TimeGAN	TimeVAE_base	TimeVAE_trend
AAPL	0.0692	0.1217	0.1087
NVDA	0.2163	0.0504	0.0553
ORCL	0.1692	0.1246	0.0340

Рис. 2.2: accuracy - 1

2. Проверим, могут ли сгенерированные данные быть полезны для предсказания, выучивают ли они условные распределения с течением времени. Для этого обучаем двухслойную LSTM предсказывать ряд на одну точку вперед и оцениваем обученную модель на исходном наборе данных. В качестве метрики будем брать mean absolute error(MAE). Для сравнение приведено также MAE для предсказания ряда этой модели, обученной на тех же данных(real).

	TimeGAN	TimeVAE_base	TimeVAE_trend	real
AAPL	0.0069	0.0074	0.0071	0.0059
NVDA	0.0140	0.0123	0.0120	0.0113
ORCL	0.0020	0.0019	0.0025	0.0021

Рис. 2.3: Predictive score

3. Визуализируем сгенерированные данные. Для этого усредним по размерности одной точки(Open, Close, High, Low, Volume) после нормализации. Получим данные размерности [num_samples, seq_len]. Понизим размерность с seq_len до 2 с помощью PCA и TSNE, чтобы посмотреть, насколько близки сгенерированные и реальные данные.

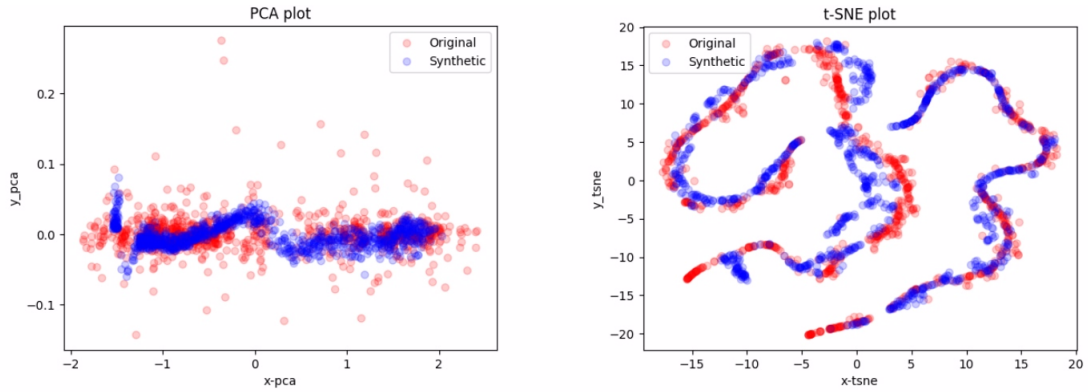

Рис. 2.4: GAN visualization

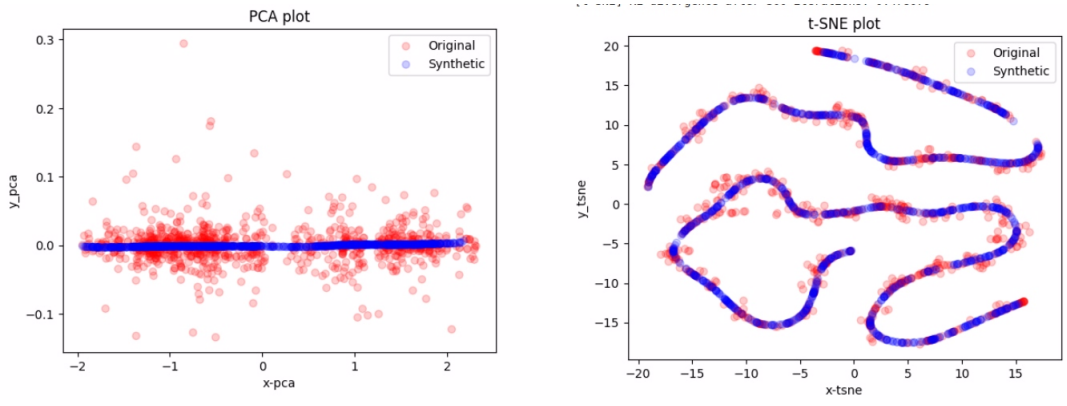

Рис. 2.5: VAE visualization

Видим, что как по мере сходства, так и по предсказательной способности на разных датасетах побеждают разные алгоритмы. Результаты будут отличаться при разных запусках, корректнее было бы смотреть усреднение метрик по множеству сгенерированных выборок, но на это не было ресурсов. Понятно, что результаты также сильно зависят от слоев внутри TimeVAE и TimeGAN и от гиперпараметров.

Этот пункт скорее показывает, что обе модели способны генерировать реалистичные данные, что видно и по метрикам и по визуализациям с понижением размерности.

3. Сравнение GANов

Теперь сравним различные GAN модели, которые позволяют генерировать следующие точки при условии предыдущих. Будем проводить сравнение по аналогии со статьей 5 для моделей TimeGAN с условием, CWGAN, SigCWGAN. Данные - минутные свечки по акциям Apple. Причем брать будем 2-мерный ряд: цена закрытия и объем, тк опыт показал, что лучше всего получается выучить распределения именно такого ряда.

Будем проверять некоторые стилистические факты, поэтому предобрабатываем данные, меняя точки на прирост логарфима ряда. В качестве условия в результатах далее используем 3 точки до, генерируем 3 точки после.

Для начала посмотрим, как распределены приросты на реальных и сгенерированных данных: построим гистограммы. Также посмотрим скоррелированность приростов: построим автокорреляционную функцию для первых 3 лагов.
На картинках слева изображены гистограммы реальных и сгенерированных данных, в центре - то же самое в логарифмических координатах, справа - автокрорреляционная функция. По вертикали - координаты ряда(close_price, volume). Все результаты показаны на отложенной выборке (20 % от всей выборки).

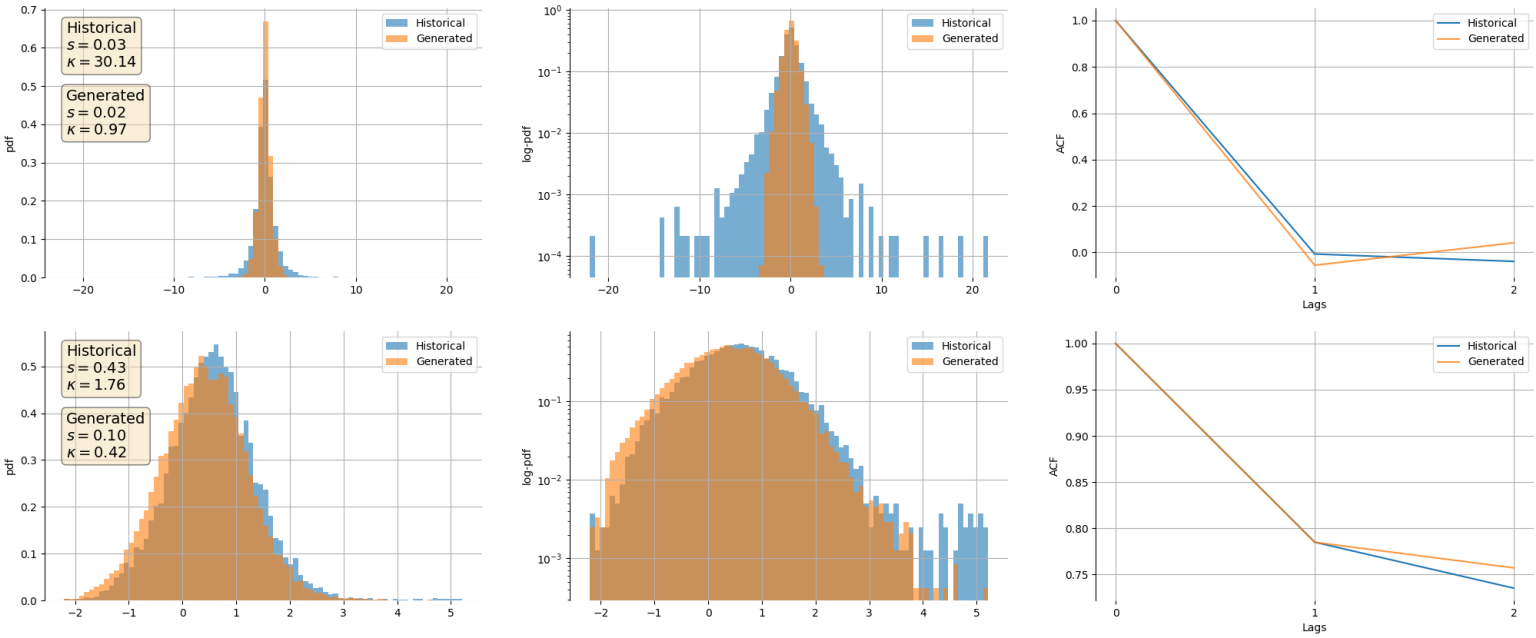

Рис. 3.1: TimeGAN

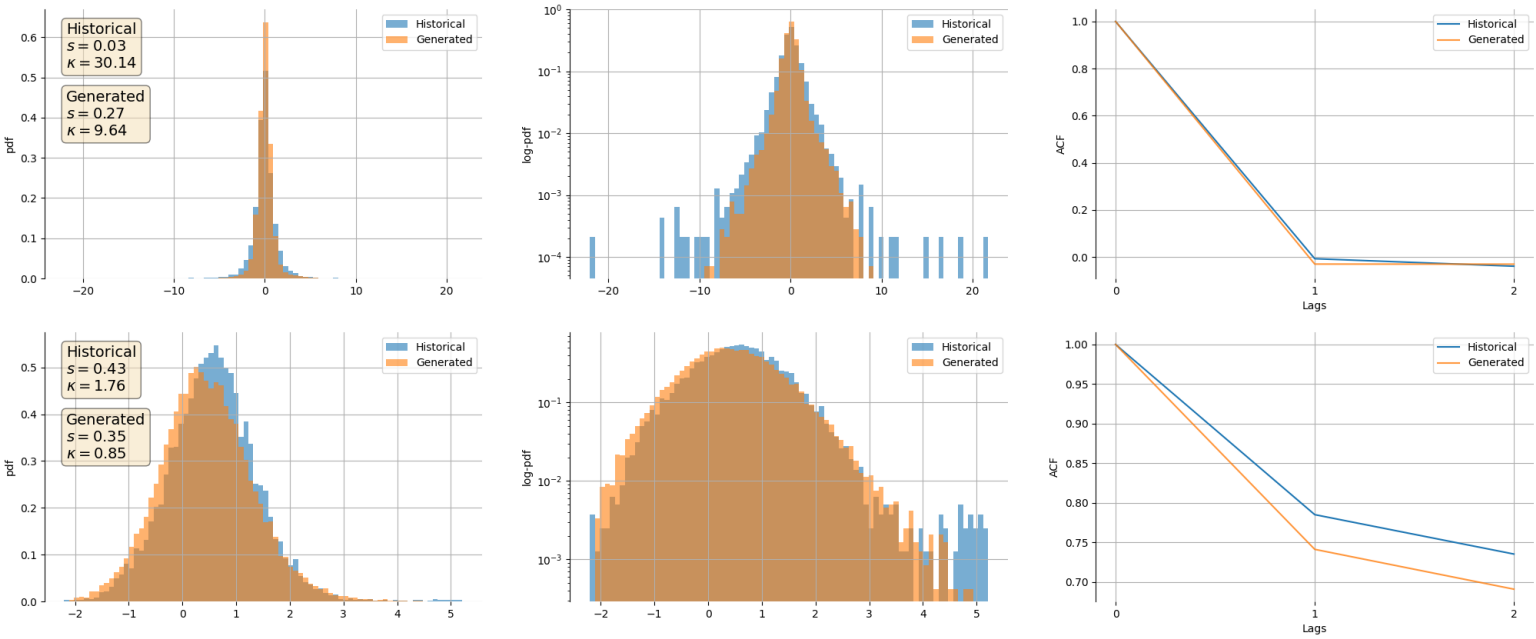

Рис. 3.2: CWGAN

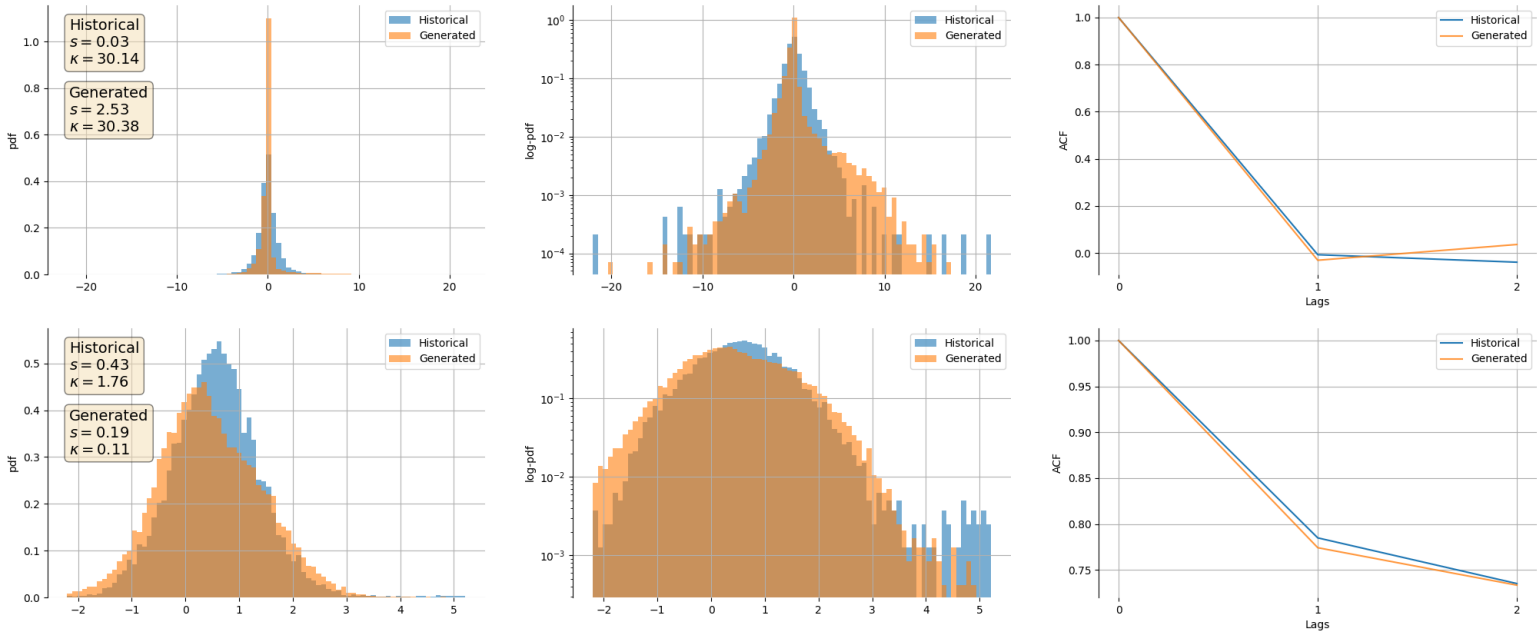

Рис. 3.3: Sig-CWGAN

Визуально распределения похожи, особенно у CWGAN. Видно, что sigcwgan хорошо выучивает kurtosis, а skew плохо, остальные алгоритмы - наоборот. По автокорреляции результаты неплохие. Посмотрим метрики.

Predictive score. Для сравнения предсказательной способности аналогично предыдущему пункту будем сравнивать качество предсказания модели, обученной на реальных данных(trtr - train_real_test_real) и на синтетических(tstr - train_synthetic_test_real). Берем в качестве признаков предыдущие 3 шага, каждый размерностью 2, и считаем R2_score для предсказания следующего шага каждой компоненты с помощью линейной регрессии. Усредняем полученные 2 значения R2_score - получаем r2_trtr для модели, обученной на реальных данных и r2_tstr для модели, обученной на синтетических данных. Метрика $predictive\_score = \|r2\_trtr - r2\_tstr\|$. Метрика посчитана при использзовании бустинга, случайного леса и линейрой регрессии.

model_id	r2_trtr_boosting	r2_tstr_boosting	score_boosting	r2_trtr_forrest	r2_tstr_forrest	score_forrest	r2_trtr_linreg	r2_tstr_linreg	score_linreg
CWGAN	0,49450	0,28352	0,21098	0,90090	0,12245	0,77846	0,33378	0,30863	0,02515
SigCWGAN	0,49450	0,28003	0,21447	0,89857	0,01431	0,88426	0,33378	0,31617	0,01761
TimeGAN	0,49450	0,31317	0,18133	0,90227	0,21389	0,68838	0,33378	0,31920	0,01458

Рис. 3.4: predictive scores

Видно, что значительной разницы в скоре для разных алгоритмов нет. Также можно заметить довольно странный результат: линейная регрессия почти не чувствует разницы при обучении на реальных данных и синтетических, в то время как бустинг и лес показывают на синтетике качество значительно хуже, чем на реальных данных. Возможно, дело просто в плохо подобранных параметрах.

Метрика маргинального распределения. Для каждого измерения объекта $i \in 1, \cdots, d$ вычисляем две эмпирические функции плотности(epdf) на основе гистограмм реальных данных и синтетических данных, соответственно $\hat{df}_r^i$ и $\hat{df}_G^i$. Далее берем абсолютную разницу этих двух epdf, усредненную по размерности объекта. $abs\_metric = \frac{1}{d}\sum_{i=1}^{d} \|\hat{df}_r^i - \hat{df}_G^i\|$.

Метрики skew и kurtosis означают разность этих мер для реальных и синтетических данных.

Временные зависимости. Для каждого измерения признака $i \in 1, \cdots, d$ вычисляем автоковариацию $i-$ координаты данных временного ряда X с лагом k для реальных данных и синтетических, обозначаемой $\rho_r^i k$ и $\rho_G^i k$. Для сравнения по данной размерности можно использовать $\frac{\rho_r^i(1)}{\rho_r^i(0)} / \frac{\rho_G^i(1)}{\rho_r^G(0)}$. После усреднения получаем метрику

$$\text{ACF\_score} = \frac{1}{d}\sum_{i=1}^{d} \|\frac{\rho_r^i(1)}{\rho_r^i(0)} / \frac{\rho_G^i(1)}{\rho_r^G(0)}\|.$$

Взаимодействия между компонентами. Пусть $\tau_r^{i,j}$ и $\tau_G^{i,j}$ - корреляции между i-й j-й компонентами реальных и сгенерированных данных.

Используем следующую метрику для похожести с точки зрения межпризнаковых корреляций $\sum_{i=1}^{d}\sum_{i=j}^{d} \|\tau_r^{i,j} - \tau_G^{i,j}\|$.

Сводная таблица со всеми указанными метриками приведена в таблице 3.5.

model_id	abs_metric	acf_id_lag=1	cross_correl	kurtosis	skew
CWGAN	0,01343	0,02693	0,00408	10,59053	0,10896
SigCWGAN	0,02337	0,01443	0,01761	1,14222	1,53383
TimeGAN	0,01481	0,02202	0,04126	15,14765	0,17475

Рис. 3.5: Метрики

Как уже было замечено, sigCWGAN хорошо выучил kurtosis, остальные - skew. По метрикам распределения и автокорреляциям особой разницы у клгоритмов нет. Корреляции между объемом и ценой закрытия лучше всего выучивает CWGAN.

ЗАКЛЮЧЕНИЕ

Были рассмотрены различные алгоритмы генерации временных рядов и исследована возможность применения этих алгоритмов к моделированию высокочастотных финансовых данных: минутных свечек по акциям.

Оказалось, что генеративные модели могут неплохо выучивать статистические свойства таких данных. Также было показано, что в некоторых слуаях при обучении на синтетических данных качество модели предсказания падает не очень значительно. Это означает, что данные методы можно потенциально использовать для генерации дополнительных данных в обучении прогнозных моделей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Abhyuday Desai, Cynthia Freeman, Zuhui Wang, Ian Beaver. TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation.
2. Jinsung Yoon. Time-series Generative Adversarial Networks.
3. Ying Yu; Bingying Tang; Ronglai Lin; Shufa Han; Tang Tang; Ming Chen. CWGAN: Conditional Wasserstein Generative Adversarial Nets for Fault Data Generation.
4. Shujian Liao, Hao Ni, Lukasz Szpruch, Magnus Wiese, Marc Sabate-Vidales, Baoren Xiao. Conditional Sig-Wasserstein GANs for Time Series Generation.
5. Lars Ericson, Xuejun Zhu, Xusi Han, Rao Fu, Shuang Li, Steve Guo, Ping Hu. Deep Generative Modeling for Financial Time Series with Application in VaR: A Comparative Review.