TimeGAN Synthetic Data: evaluation

paper (https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf)

- supplement (https://www.vanderschaar-lab.com/papers/NIPS2019_TGAN_Supplementary.pdf)
- github (https://github.com/jsyoon0823/TimeGAN)

The real sample is a timeseries of OHLC+ (Open/High/Low/Close/Adj Close/Volume) of a single stock ticker.

We use Jansen's notebook (https://github.com/stefan-jansen/synthetic-data-for-finance/blob/main/02_evaluating_synthetic_data.ipynb) to evaluate the synthetic data created by the TimeGAN:

- An example is
    - a sequence of length T
    - each element of the sequence has n features

To be concrete, we will refer to

- the sequence dimension as the time/date dimensions
- the feature dimensions as the ticker return dimension

So an example is a timeseries across T time steps, each element being a one-step sample of the returns of n tickers.

n.b., Jansen's notebook is derived from (but not identical to) the author's evaluation code (https://github.com/jsyoon0823/TimeGAN/blob/master/metrics/visualization_metrics.py)

The raw real data (file `stocks.csv`) is of shape $(N \times n)$ where N is the number of dates over 18 years.

An example is formed by taking rolling slices (across the date dimension) of length T.

A *sample* is a fixed number (`sample_size`) of sequences: shape $(\text{sample\_size} \times T \times n)$

In order to evaluate TimeGAN, we compare properties of

- the real sample `real_sample`
- the synthetic sample `synthetic sample`

Diversity

We want to compare the distributional properties of the two samples.

Because samples are 3 dimensional, there are two types of distributional properties

- cross-sectional: ticker versus ticker
    - relationship between returns of all tickers at each time
- timeseries: time step versus time step
    - relationship of the return of a *single* ticker across time

This is not completely straight forward:

Suppose we want to examine the cross-sectional properties

- e.g., examining the $(n \times n)$ correlation matrix of ticker versus ticker

There are 2 other dimensions (sample index, time index)

- but we need to create a one-dimensional vector for each ticker
- in order to compute the pairwise correlation
- so we need to pool the sample and time dimensions
    - a ticker's one-dimensional vector has entries for every sample and every time step

Similarly for the timeseries relationship

- need to pool the ticker and sample dimensions
    - each time step's vector has entries for every sample and ticker
- so we can create a one-dimensional vectors for each time step
- in order to compute the pairwise correlation of one time step versus another

Although either method of pooling feels like we are mixing apples and oranges

- as long as we apply the sample pooling to the real and synthetic samples
- we are aligning comparable returns
    - same (sample, time) pair or same (sample, ticker) pair
    - so never really mix

Which to choose ?

Why not both !

From a statistical point of view

- all samples are draws from the same distribution
- so seems reasonable to pool across samples

Visual comparison: PCA

Rather than comparing the correlation matrix of the real sample to the correlation matrix of the synthetic sample

- the authors used dimensionality reduction (PCA)
- to reduce
  - the T time steps to 2 component "time factors" for timeseries property analysis
  - the n tickers to 2 component "ticker factors" for ticker property analysis
- the PCA is performed **only on the real sample**
  - the "true" factors

Reduction to two dimensions allows us to create a scatter plot of each entry in the pooled data of a sample.

- Places each entry in the 2D space of Component 1 versus component 2

By superposing the scatter plot of the real and synthetic samples

- we hope to see if the factors identified for the real sample
- are a good explanation for the structure of the synthetic sample-

Pooling methods: Author versus Jansen

Both Jansen and the authors create a second dimension equal to T

- the authors eliminate the ticker dimension by taking the *mean* across all tickers at a single time
- Jensen combines the `sample_size` and ticker dimension to one dimension of size $\text{sample\_size} * n$
    - pooling observations across tickers

This is certainly something we can discuss !

- taking the mean eliminates distinctions between tickers
- combining sample and ticker dimensions is equivalent to saying that the "time factor"
    - that generates daily returns for one ticker
    - is shared by all tickers (by the pooling across tickers)

Visual comparison: t-SNE

The authors (and Jansen) plot the real and synthetic samples in 2 dimensions using t-SNE

- another dimensionality reduction technique

Again, the superposing of plots of the two samples enables a visual comparison.

Here are some visualizations

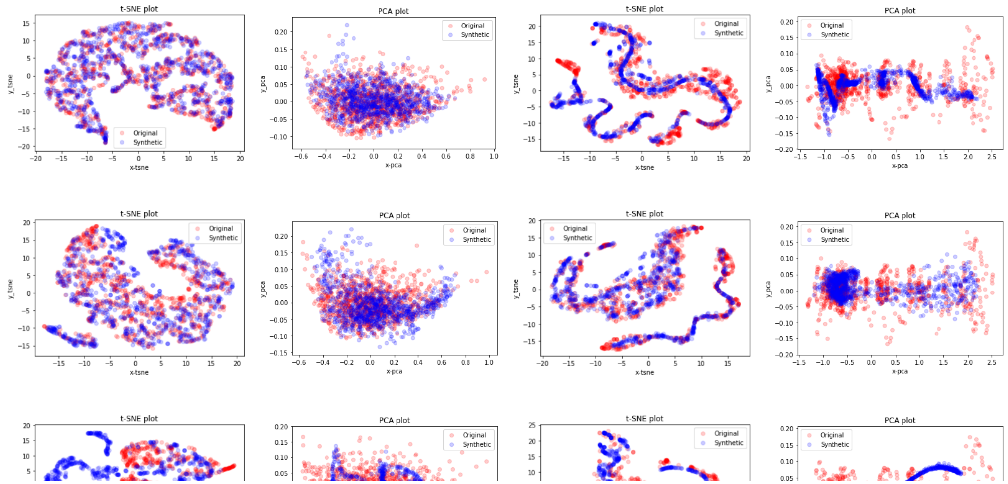
- Each row visualizes the result of a different model
    - TimeGAN is first row
- Red is real sample; Blue is synthetic
- Columns 1 and 2 show results on the "Sines" real dataset": t-SNE and PCA visualization
- Columns 3 and 4 shows results on the "Stocks" real dataset: t-SNE and PCA visualization

Note

t-SNE coordinates are dependent on the data

- so the "real" (red) plots on different rows may not appear identical
- because the t-SNE computation takes the blue points into account too

Additional Visualizations with t-SNE and PCA



Fidelity

Obtain a sample of real and synthetic data

- Pooled into 2 dimensional structure of dimension $(\text{sample\_size} * n \times T)$
- Split each sample into a training and test cohort
- Train a simple sequence Classifier
    - RNN on sequences of length T
        - reduces sequence to a fixed length vector
    - Binary Classifier: `Dense` layer with 1 unit and a sigmoid activation

The Sequence Classifier's performance metric is *worse* out of sample

- suggesting that a classifier cannot distinguish between real and synthetic examples
- hence: the quality of synthetic examples is good

Predictive: Train Synthetic, Test Real (TSTR)

The author's define a Target task

- given a prefix of the timeseries: predict the one-step ahead element

The model

- Uses an RNN accepting input sequences of length at most $(T-1)$
- Followed by a `Dense` layer with n units

It is trained on the synthetic sample

```
synthetic_train = synthetic_data[:, :T-1, :]
synthetic_label = synthetic_data[:, -1, :]
```

Two models are trained; both are evaluated on the same *real* sample

- one trained on a *synthetic* sample
- one trained on a *real* sample

The authors find that the Performance Metric MAE (Mean Absolute Error)

- is lower for the model trained on synthetic
- compared to the model trained on real

Concluding that the synthetic sample is "good enough" for the Target task.

Suggestions

The OHLC data has a number of mathematically defined constraints for each step of the timeseries

$$\text{Low} \leq \text{Close} \leq \text{High}$$
$$\text{Low} \leq \text{Open} \leq \text{High}$$

Does the synthetic sample obey these constraints ?

- Nothing in the Loss function to constrain it
- Could we add a constraint to the Loss to enforce this ?