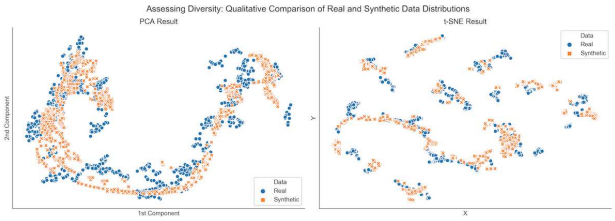
Generative Adversarial Nets for Synthetic Time Series Data

This chapter introduces generative adversarial networks (GAN). GANs train a generator and a discriminator network in a competitive setting so that the generator learns to produce samples that the discriminator cannot distinguish from a given class of training data. The goal is to yield a generative model capable of producing synthetic samples representative of this class. While most popular with image data, GANs have also been used to generate synthetic time-series data in the medical domain. Subsequent experiments with financial data explored whether GANs can produce alternative price trajectories useful for ML training or strategy backtests. We replicate the 2019 NeurIPS Time-Series GAN paper to illustrate the approach and demonstrate the results.



Generative adversarial networks for synthetic data

This book mostly focuses on supervised learning algorithms that receive input data and predict an outcome, which we can compare to the ground truth to evaluate their performance. Such algorithms are also called discriminative models because they learn to differentiate between different output values. Generative adversarial networks

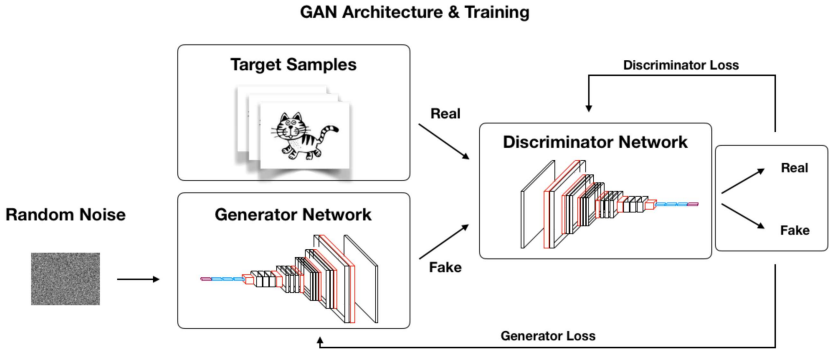
Comparing generative and discriminative models

Discriminative models learn how to differentiate among outcomes y, given input data X. In other words, they learn the probability of the outcome given the data: p(y | X). Generative models, on the other hand, learn the joint distribution of inputs and outcome p(y, X).

While generative models can be used as discriminative models using Bayes Rule to compute which class is most likely (see Chapter 10), it appears often preferable to solve the prediction problem directly rather than by solving the more general generative challenge first.

Adversarial training: a zero-sum game of trickery

The key innovation of GANs is a new way of learning the data-generating probability distribution. The algorithm sets up a competitive, or adversarial game between two neural networks called the generator and the discriminator.



Code example: How to build a GAN using TensorFlow 2

To illustrate the implementation of a generative adversarial network using Python, we use the deep convolutional GAN (DCGAN) example discussed earlier in this section to synthesize images from the fashion MNIST dataset that we first encountered in Chapter 13.

convolutional GAN (DCGAN) example to synthesize images from the fashion MNIST dataset

Code example: TimeGAN: Adversarial Training for Synthetic Financial Data

Generating synthetic time-series data poses specific challenges above and beyond those encountered when designing GANs for images. In addition to the distribution over variables at any given point, such as pixel values or the prices of numerous stocks, a generative model for time-series data should also learn the temporal dynamics that shapes how one sequence of observations follows another (see also discussion in Chapter 9: Time Series Models for Volatility Forecasts and Statistical Arbitrage).

Very recent and promising research by Yoon, Jarrett, and van der Schaar, presented at NeurIPS in December 2019, introduces a novel Time-Series Generative Adversarial Network (TimeGAN) framework that aims to account for temporal correlations by combining supervised and unsupervised training. The model learns a time-series embedding space while optimizing both supervised and adversarial objectives that encourage it to adhere to the dynamics observed while sampling from historical data during training. The authors test the model on various time series, including historical stock prices, and find that the quality of the synthetic data significantly outperforms that of available alternatives.

Learning the data generation process across features and time

A successful generative model for time-series data needs to capture both the cross-sectional distribution of features at each point in time and the longitudinal relationships among these features over time. Expressed in the image context we just discussed, the model needs to learn not only what a realistic image looks like, but also how one image evolves from the next as in a video.

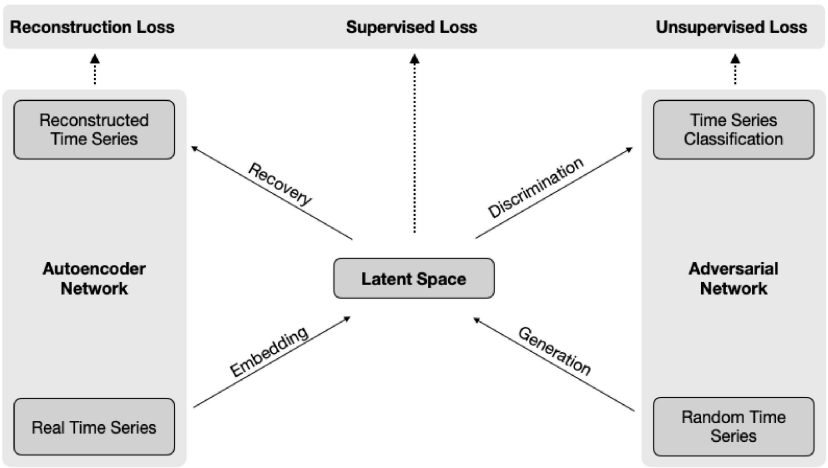
time-series embedding

Prior attempts at generating time-series data like the recurrent (conditional) GAN relied on recurrent neural networks (RNN, see Chapter 19, RNN for Multivariate Time Series and Sentiment Analysis) in the roles of generator and discriminator.

TimeGAN explicitly incorporates the autoregressive nature of time series by combining the unsupervised adversarial loss on both real and synthetic sequences familiar from the DCGAN example with a stepwise supervised loss with respect to the original data. The goal is to reward the model for learning the distribution over transitions from one point in time to the next present in the historical data.

The four components of the TimeGAN architecture

The TimeGAN architecture combines an adversarial network with an autoencoder and has thus four network components as depicted in Figure 21.4: Autoencoder: embedding and recovery networks Adversarial Network: sequence generator and sequence discriminator components



Implementing TimeGAN using TensorFlow 2

In this section, we implement the TimeGAN architecture just described. The authors provide sample code using TensorFlow 1 that we port to TensorFlow 2. Building and training TimeGAN requires several steps: 1. Selecting and preparing real and random time series inputs 2. Creating

Running the training loops and logging the results 3. Generating synthetic time series and evaluating the results

The notebook TimeGAN_TF2 shows how to implement these steps.

Evaluating the quality of synthetic time-series data

The TimeGAN authors assess the quality of the generated data with respect to three practical criteria: 1. **Diversity**: the distribution of the synthetic samples should roughly match that of the real data 2. **Fidelity**: the sample series should be indistinguishable from the real data, and 3. **Usefulness**: the synthetic data should be as useful as their real counterparts for solving a predictive task

The authors apply three methods to evaluate whether the synthetic data actually exhibits these characteristics: 1. **Visualization**: for a qualitative diversity assessment of diversity, we use dimensionality reduction (principal components analysis (PCA) and t-SNE, see Chapter 13) to visually inspect how closely the distribution of the synthetic samples resembles that of the original data 2. **Discriminative Score**: for a quantitative assessment of fidelity, the test error of a time-series classifier such as a 2-layer LSTM (see Chapter 18) let's us evaluate whether real and synthetic time series can be differentiated or are, in fact, indistinguishable. 3. **Predictive Score**: for a quantitative measure of usefulness, we can compare the test errors of a sequence prediction model trained on, alternatively, real or synthetic data to predict the next time step for the real data.

The notebook evaluating_synthetic_data contains the relevant code samples.

Resources

How GAN's work

- Why is unsupervised learning important?, Yoshua Bengio on Quora, 2018
- GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation, Minsuk Kahng, Nikhil Thorat, Duen Horng (Polo) Chau, Fernanda B. Viégas, and Martin Wattenberg, IEEE Transactions on Visualization and Computer Graphics, 25(1) (VAST 2018), Jan. 2019
- GitHub
- Generative Adversarial Networks, Ian Goodfellow, et al, 2014
- Generative Adversarial Networks: an Overview, Antonia Creswell, et al, 2017
- Generative Models, OpenAI Blog

Implementation

- Deep Convolutional Generative Adversarial Network
- CycleGAN
- Keras-GAN, numerous Keras GAN implementations
- PyTorch-GAN, numerous PyTorch GAN implementations

The rapid evolution of the GAN architecture zoo

- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (DCGAN), Luke Metz et al, 2016
- Conditional Generative Adversarial Net, Medhi Mirza and Simon Osindero, 2014
- Infogan: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, Xi Chen et al, 2016
- Stackgan: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, Shaoting Zhang et al, 2016
- Photo-realistic Single Image Super-resolution Using a Generative Adversarial Network, Alejando Acosta et al, 2016
- Unpaired Image-to-image Translation Using Cycle-consistent Adversarial Networks, Juan-Yan Zhu et al, 2018
- Learning What and Where to Draw, Scott Reed, et al 2016
- Fantastic GANs and where to find them

Applications

Ratsch, 2016

- GitHub Repo
- MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks, Dan Li, Dacheng Chen, Jonathan Goh, and See-Kiong Ng, 2019
- GitHub Repo
- GAN—Some cool applications, Jonathan Hui, 2018
- gans-awesome-applications, curated list of awesome GAN applications