Generative Models & Generative Adversarial Networks (GAN)

Idea: Construct 2 neural networks that compete with each other

↳ 1) Creates new data with the same statistics as the training data (generator)

2) Differentiates between real & fake data (adversary / discriminator)

↳ The generator is trying to create data that tricks the adversary

The adversary is trying to catch the generator

These 2 models are trained together in a zero-sum game

Training continues until the discriminator is fooled about 50% of the time

↳ it cannot distinguish between real & fake data

Before going into details, let's consider some applications in Finance:

1) CoVaR + stress testing

There are limited number of stress scenarios in real data

Can lead to overfitting the risk management to the prior crisis (2008, 2020, ..)

↳ misses other types of stress events

↳ Use GANs to generate fictitious time series data from stress scenarios

2) Derivative Pricing

The Black-Scholes equation assumes a parametric model for the underlying asset (Geometric Brownian motion with parameter σ)

This has been relaxed for, e.g., stochastic volatility or rough volatility

↳ still parametric forms

GAN can generate data for Monte Carlo simulations that does not assume a parametric form $\Rightarrow$ can be more robust to "irrational" periods in the market

3) Outlier detection

Generally more data helps make outliers clearer to find

4) Small data environments

If you just do not have enough data to train a model to the desired accuracy

For example, 10 years of data ≈ 2500 data points ← much smaller than required for many advanced ML techniques

Generally, we can think of GANs as a market simulator to replace GBM in Black-Scholes

Other generative modelling paradigms exist

- ↳ often proposed for "static" problems
- ↳ adoption to financial time series modelling is not straightforward

(This is ongoing research)

↳ Generally: train a neural network to "transport" some source distribution μ (ex: $\mu \sim N(0,1)$) to a target distribution observed from the data (ex: simplest is to use the empirical distribution ← results in bootstrapping) ex: Deep Neural Network Generators:

- Restricted Boltzmann Machine (RBM)

  2 layers (1 hidden layer) but repeated to reach the desired distribution

- GANs: idea discussed both above + below

- Variational Autoencoders (VAE)

  Attempt to create "blurrier" images

All data-driven with adaptation to time series is not straightforward

Why is time series data hard?

↳ Dependencies over time need to be accounted for

Some properties of financial time series:

- asset returns have heavier tails than the normal distribution
- asset returns are more "peaked" than the normal distribution
- asset returns have "volatility clustering" (periods of high + low activity)
- volatility is negatively correlated with returns ("leverage effect")
- Often assume uncorrelated returns over time (autocorrelations ≈ 0)

  but not independent

Generative Adversarial Networks (for static data)

<u>Recall</u>: Generator network tries to produce new data

Discriminator network attempts to classify data as real or fake

↪ Generator is "accepted" if it fools the discriminator often enough

The generator network produces samples $x = g(z \,;\, \theta^{(g)})$

"input" such as the seed of a random number generator

The discriminator network gives the probability that data is real: $d(x \,;\, \theta^{(d)})$

Simplest formulation of training is as a zero-sum game

↪ discriminator has value $v(\theta^{(g)}, \theta^{(d)})$ (or $v(g, d)$

generator has value $-v(\theta^{(g)}, \theta^{(d)})$ or $-v(g, d)$)

↪ each network is trying to maximize its value given the actions of the other

$$g^* = \arg\min_g v(g, d^*)$$
$$d^* = \arg\max_d v(g^*, d)$$
fixed point
so convergence to a solution is important

at convergence: $g^* = \arg\min_g \max_d v(g,d)$

$d^* = \arg\max_d \min_g v(g,d)$

Because zero-sum

The value function v can take many forms

A common example: $v(g,d) = \mathbb{E}[\log(d(x)) \mid x \sim p_{data}] + \mathbb{E}[\log(1-d(g(z))) \mid z \sim p_z]$

cost for real data

cost for fake data

Forces the discriminator to try to separate real + fake data

& the generator to try to fool the classifier into believing its samples are real

Note: convergence can be a problem for GANs

Training can get stuck in "cycles"

Other objectives can perform better for convergence

Generative Adversarial Networks (for financial time series)

Generally follows the same framework as (static) GAN

Want to think about choice of neural network design more carefully

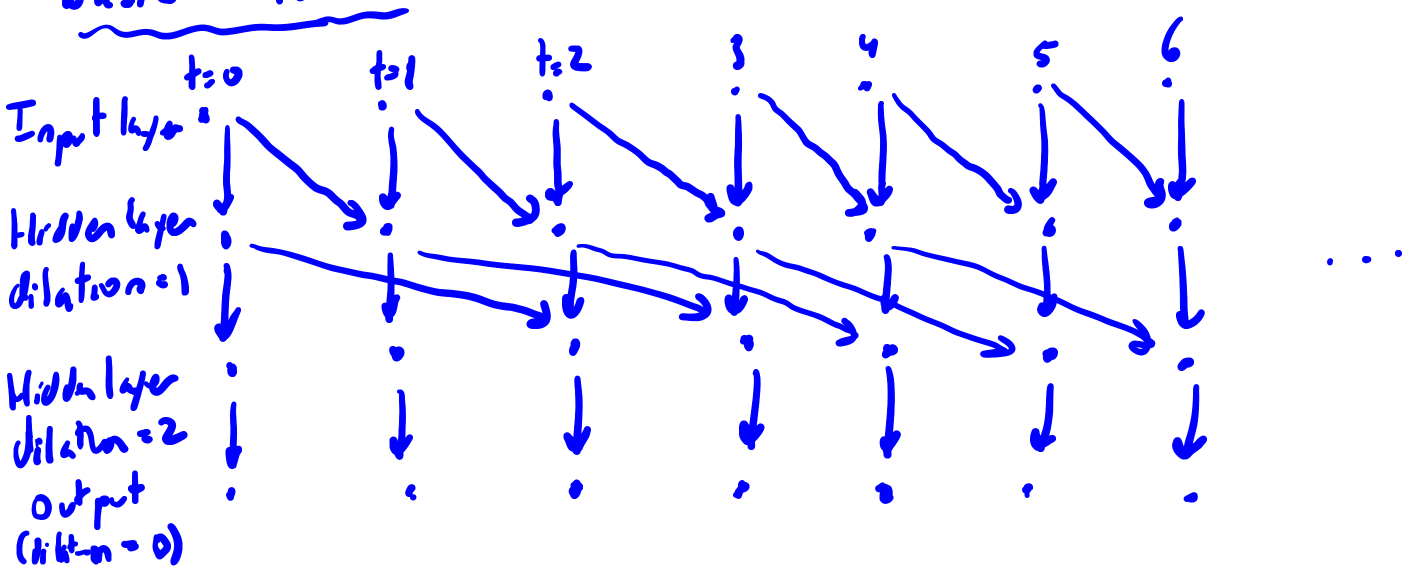
Quant GAN (paper available on Canvas, but not the only approach)

To model stochastic volatility, it uses Temporal Convolutional Networks (TCN)

↳ TCNs were first introduced for video data

↳ Can capture long-range dependencies more effectively than some RNNs

Basic structure



Choice of number of hidden layers + dilations can matter a lot
(Typical: dilation = 1 unless strong feeling otherwise)

⌜Discriminator is also using TCN to capture historical dependencies
You can use LSTM instead (for instance) ⌟

To improve the generator, use a TCN for returns + a ~~2^{nd}~~ network to model the noise/innovations (since this can change over time aswell)
↳ often not a TCN

↳ creates a SVNN (stochastic volatility neural network)

$(z_{t-k}, \ldots, z_{t-1}) \mapsto (\mu_t, \sigma_t)$ mean + variance through TCN

$z_t \mapsto \varepsilon_t$ innovation through feed-forward neural network

Paper goes into choices of activation functions, etc... in order to match properties of financial time series data (ex: heavy tails)

⌜No pre-built implementation of this is available online ⌟