Beating the market is possible, when using GANs for investment strategies tuning and combination

Generative Adversarial Networks for Financial Trading Strategies

Adriano Koshiyama, Nick Firoozye and Philip Treleaven

Computer Science Department, University College London

[adriano.koshiyama.15, n.firoozye, p.treleaven]@ucl.ac.uk

Introduction

To obtain an edge in a highly competitive environment, the analyst needs to proper fine-tune its strategy, or discover how to combine weak signals in novel alpha creating manners

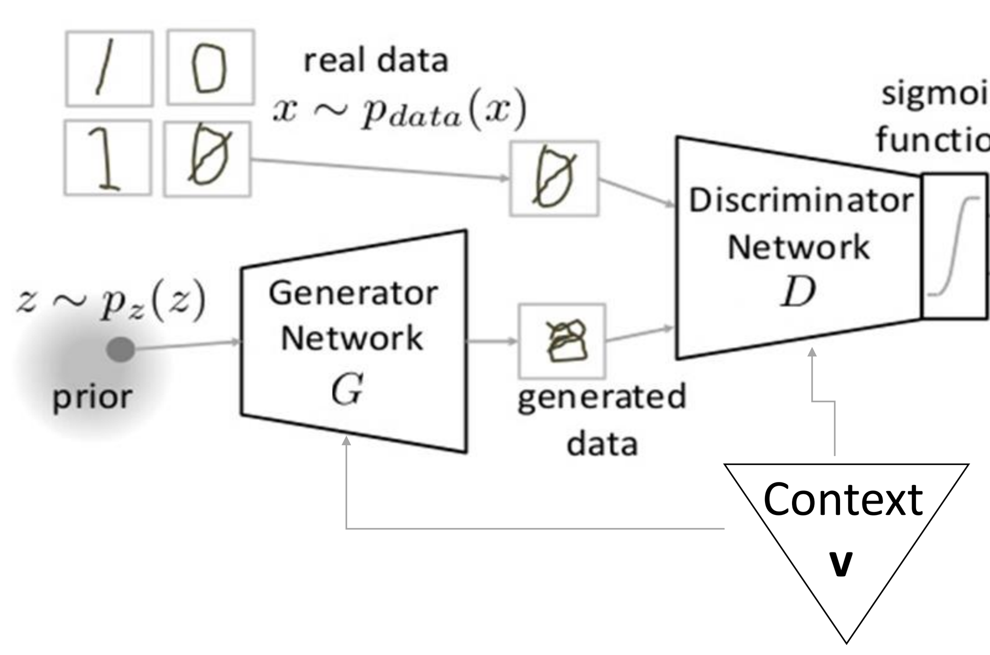
Conditional Generative Adversarial Networks (cGANs) can have an impact into both aspects of trading strategies; also, we can list a few advantages of such method, like:

i. generating more diverse training and testing sets, compared to traditional resampling techniques;
ii. able to draw samples specifically about stressful events, ideal for model checking and stress testing; and
iii. providing a level of anonymization to the dataset, differently from techniques that (re)shuffle/resample data.

In this work we provide a full methodology on (i) the training and selection of a cGANs for time series; (ii) how each sample is used for strategies calibration; and (iii) how all samples can be used for ensemble modelling.

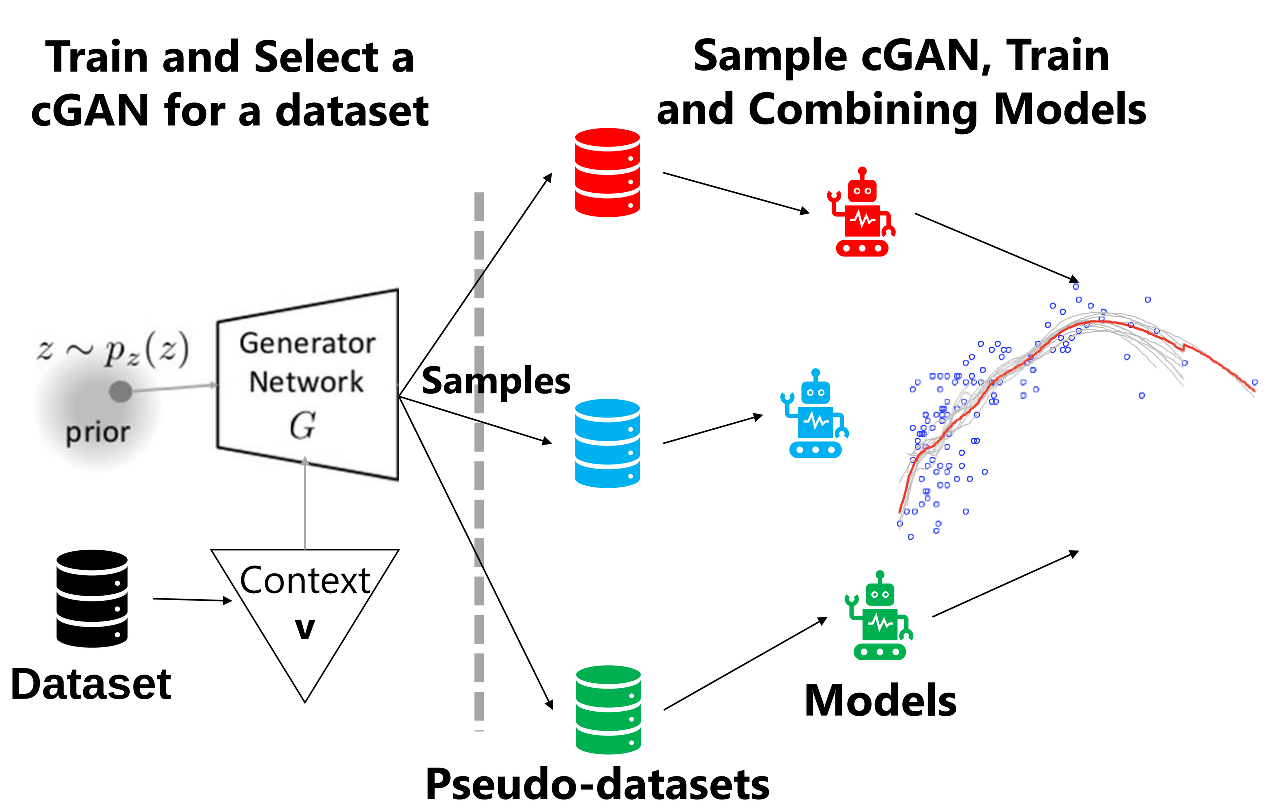
cGAN: Training and Combining

Conditional GANs are an extension of a traditional GAN, when both G and D decision is based not only in noise or generated inputs, but include an information set $\mathbf{v}$.



In our case, given a time series $y_1, \dots, y_t, \dots, y_T$, our conditional set is $\mathbf{v} = [y_{t-1}, \dots, y_{t-p}]$ and we are sampling/discriminating is $\mathbf{x} = [y_t]$. We train the cGAN by solving this min-max problem:

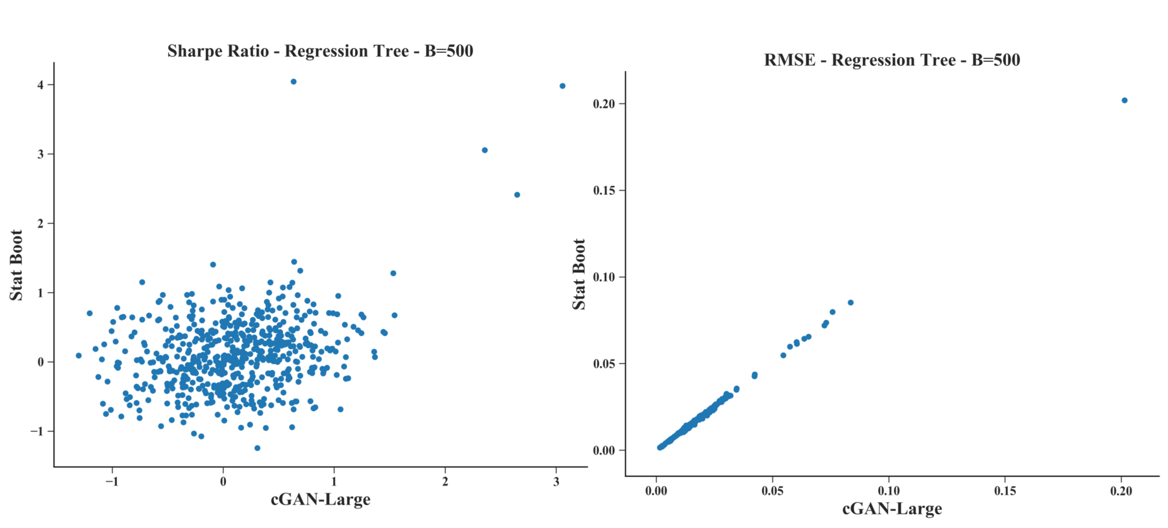
$$\min_G \max_D V(D,G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{v})] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{v})))]$$



Results

❖ SPX Index Results

❖ All results: Regression Tree



Outcomes

❖ In combination of trading strategies, our results suggest that both approaches are equivalent in aggregate;

❖ In fine-tuning of trading strategies, we have evidence that cGANs can be used for model tuning, bearing better results in cases where traditional schemes fail.

Appendix

Abbreviation	Individual Configuration
cGAN-Small (G, D)	number of neurons = (5, 5)
cGAN-Medium (G, D)	number of neurons = (100, 100)
cGAN-Large (G, D)	number of neurons = (500, 500)

Other Configuration	Values
Architecture	Multilayer Perceptron
Number of hidden layers	1
Hidden layer activation function	rectified linear
G Output layer activation function	linear
D Output layer activation function	sigmoid
Epochs	20000
Batch Size	252
Solver	Stochastic Gradient Descent
Solver Parameters	learning rate = 0.01
Conditional info	$r_{t-1}, r_{t-2}, \dots, r_{t-252}$ ($p = 252$)
Noise prior $p_{\mathbf{z}}(\mathbf{z})$	$N(0,1)$
Noise dimension $dim(\mathbf{z})$	252
Snapshot frequency ($snap$)	200
Number of samples for evaluation	$C = 50$
Input features scaling function	Z-score (standardization)
Target scaling function	Z-score (standardization)

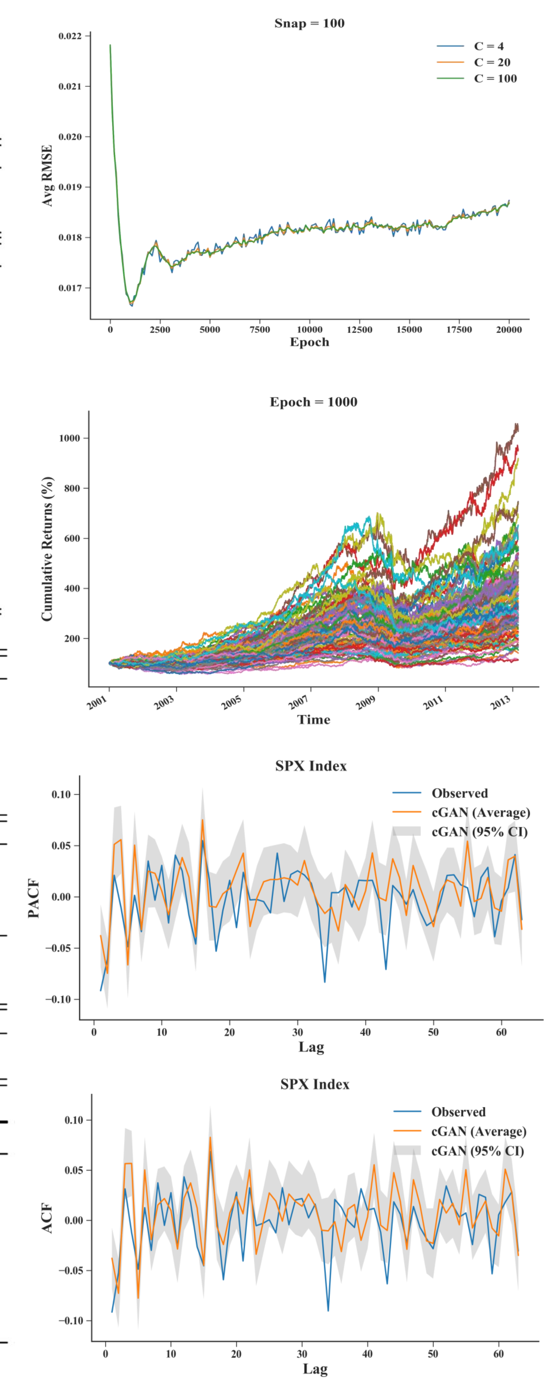
Resampling Scheme (RS)	Parameters
Stationary Bootstrap [24]	$B = \{20, 100, 500\}$ samples and block size = 20
cGAN-Small	$B = \{20, 100, 500\}$ samples
cGAN-Medium	$B = \{20, 100, 500\}$ samples
cGAN-Large	$B = \{20, 100, 500\}$ samples

Trading Strategy (M_λ)	Hyperparameters (λ)
Regression Tree (Reg Tree) [12]	unlimited depth, with minimum number of samples required to split an internal node of 2
Multilayer Perceptron (MLP) [12]	number of neurons = $\{200\}$, weight decay = $\{0.00001\}$ and activation function = $\{tanh\}$

Other details	Values
Number of lags used as features	$r_{t-1}, r_{t-2}, \dots, r_{t-252}$
Aggregation function (Agg)	Mean

Algorithm 4 Generic loop for combination of strategies

1: **for** $b \leftarrow 1, B$ **do**
2: $\mathbf{X}^{(train)} := \{r_1^*, \dots, r_{T-h}^*\} = RS(r_1, \dots, r_{T-h})$
3: fit trading strategy: $M_\lambda^{(b)}(\mathbf{X}^{(train)})$
4: add to ensemble: $ES \leftarrow M_\lambda^{(b)}(\mathbf{X}^{(train)})$
5: **end for**
6: test ensemble: $P(r_{T-h}, \dots, r_T; Agg(ES))$



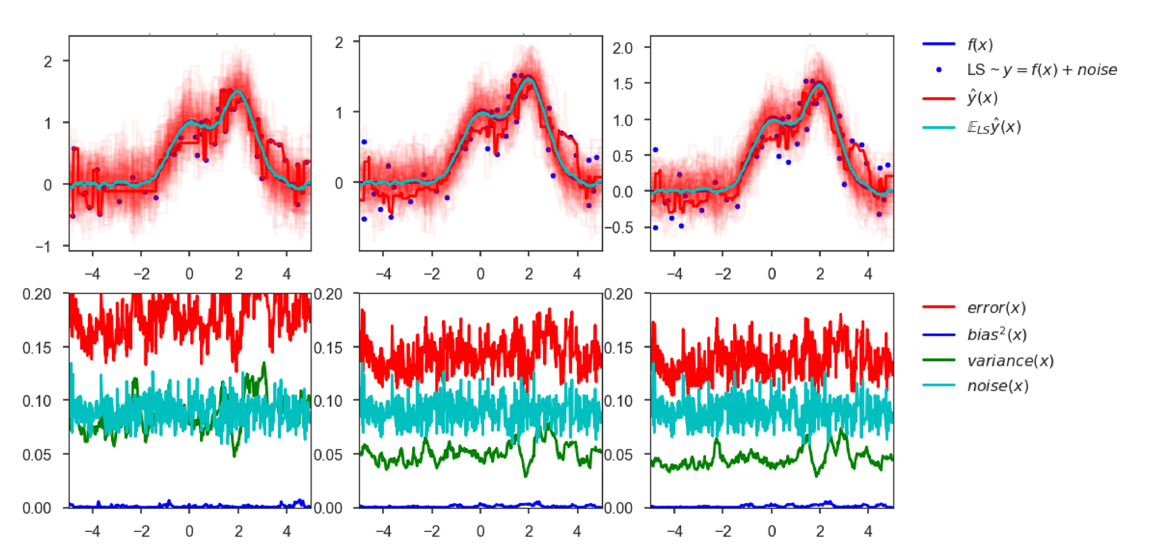
MEDIAN AND MEAN ABSOLUTE DEVIATION (MAD) RESULTS OF TRADING AND ENSEMBLE STRATEGIES ON THE OS SET.

Trad Strat	Metric	B	Ensemble Strategy: Stat Boot	cGAN-Small	cGAN-Medium	cGAN-Large
Reg Tree	Sharpe	20	0.042560 (0.380039)	0.053867 (0.378896)	0.044741 (0.380228)	**0.080540** (0.360695)
		100	0.062837 (0.378920)	0.058820 (0.387749)	0.030588 (0.390575)	**0.086423** (0.406171)
		500	0.074116 (0.397212)	0.067905 (0.392788)	0.072071 (0.392382)	**0.098094** (0.424621)
	Calmar	20	0.019442 (0.230641)	0.022619 (0.201044)	0.018987 (0.200625)	**0.035473** (0.191353)
		100	0.027235 (0.241023)	0.024254 (0.209783)	0.011890 (0.201523)	**0.036523** (0.239046)
		500	0.034422 (0.266419)	0.031174 (0.212710)	0.032761 (0.221514)	**0.042232** (0.251194)
	RMSE	20	0.014397 (0.005570)	0.014561 (0.005604)	**0.014289** (0.005414)	0.014411 (0.005432)
		100	0.014096 (0.005486)	0.014281 (0.005545)	**0.013988** (0.005357)	0.014099 (0.005373)
		500	0.014035 (0.005470)	0.014203 (0.005531)	**0.013912** (0.005346)	0.014033 (0.005361)
MLP	Sharpe	20	0.080722 (0.390515)	0.079428 (0.416847)	**0.087960** (0.393913)	0.069866 (0.398197)
		100	**0.097576** (0.382028)	0.063012 (0.415537)	0.091344 (0.397506)	0.087216 (0.414697)
		500	**0.092262** (0.390161)	0.059344 (0.415700)	0.073652 (0.389588)	0.085333 (0.414096)
	Calmar	20	0.035525 (0.223141)	0.030805 (0.217727)	**0.037877** (0.219139)	0.031145 (0.214533)
		100	**0.045916** (0.227827)	0.023479 (0.223602)	0.040718 (0.217648)	0.040572 (0.223359)
		500	**0.038678** (0.237459)	0.024014 (0.225413)	0.035688 (0.215691)	0.035885 (0.222552)
	RMSE	20	0.014030 (0.005416)	0.013999 (0.005408)	**0.013910** (0.005345)	0.014055 (0.005369)
		100	0.013924 (0.005399)	0.013973 (0.005403)	**0.013878** (0.005339)	0.014028 (0.005363)
		500	0.013924 (0.005400)	0.013974 (0.005402)	**0.013887** (0.005337)	0.014033 (0.005362)

❖ Tree (Max Depth = 7): 0.1781 (error) = **0.0009** (bias^2) + 0.0873 (var) + 0.0889 (noise)

❖ Ganning (B=10): 0.1420 (error) = 0.0010 (bias^2) + 0.0511 (var) + 0.0889 (noise)

❖ Ganning (**B=100**): **0.1366** (error) = 0.0010 (bias^2) + **0.0457** (var) + 0.0889 (noise)



Take a picture to download the full paper

or to watch the presentation