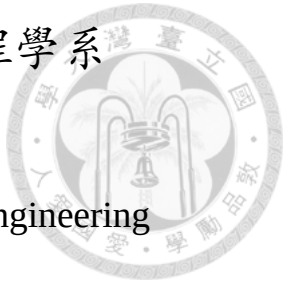


國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

應用 GAN 於回測交易策略以避免過擬合

Backtesting Trading Strategies with GAN To Avoid Overfitting

孫奧

Ao Sun

指導教授：呂育道 博士

Advisor: Yuh-Dauh Lyuu, Ph.D.

中華民國 107 年 8 月

August, 2018

doi:10.6342/NTU201801645

誌謝

時光匆匆，兩年的研究生生涯就此畫上了句話。在我的人生的一個里程碑，我想要：

感謝呂育道教授的悉心指導，呂教授教會了我很多。呂教授是我見過最聰明的人。無論是學識上還是待人接物，呂教授都是我學習的榜樣。也是老師的教誨與督促，才使得這篇論文得以圓滿完成。

感謝家人的無私付出，老爸老媽的默默付出讓我可以沒有生活壓力的度過了這兩年以及前面的二十二年，非常非常謝謝你們。

感謝實驗室的同學們的幫助，認識了很多實驗室的朋友，與你們的討論與交流是我人生的一大財富，祝你們前程似錦。

感謝我的朋友們，張晨、吉宇陽、畢銘洋、益鈞、周維夫（排名依照年齡順序），謝謝你們的陪伴以及帶來的歡笑，沒有你們我的人生不會有這麼多的歡笑。

感謝 Vic，你慷慨的幫助與對於量化交易見解讓我學到了很多關於量化交易的知識以及促成了對於論文的課題的選擇。

感謝搜到這篇論文的你，因為你，我感到我的兩年沒有被浪費。

doi:10.6342/NTU201801645

摘要

有很多研究表明，在選擇交易策略的這一過程中存在過擬合的問題。而問題的原因是在樣本內表現最好的策略不見得在樣本外。但是大部分的研究只是證實了這個現象存在，並沒有給出一套行之有效的方法來避免或減輕該問題。在本論文中，我們提出了自己的對抗過擬合的方法。我們將策略在生成的資料上做回測，而生成資料的方法我們使用了對抗生成網絡（GAN）。實驗表明，使用模型生成的資料可以避免過擬合的發生，以及使用對抗生成網絡的確可以學到股價模型的內在的關聯。因此，使用對抗生成網絡在某種程度上避免過擬合的風險。

中文關鍵詞：深度學習、對抗生成網絡、回測、回測過擬合、量化交易;

doi:10.6342/NTU201801645

Abstract

Many works have shown the overfitting hazard of selecting a trading strategy based only on good IS (in sample) performance. But most of them have merely shown such phenomena exist without offering ways to avoid them. We propose an approach to avoid overfitting: A good (meaning non-overfitting) trading strategy should still work well on paths generated in accordance with the distribution of the historical data. We use GAN with LSTM to learn or fit the distribution of the historical time series . Then trading strategies are backtested by the paths generated by GAN to avoid overfitting.

Keywords: Deep learning、Generative Adversarial Network、Backtest、Backtest Overfitting、Algorithm trading

doi:10.6342/NTU201801645

目錄

doi:10.6342/NTU201801645

doi:10.6342/NTU201801645

doi:10.6342/NTU201801645

1 緒論

1.1 研究動機

在量化交易中，策略和回測（backtesting）是很重要的元素。所謂策略指的是決定什麼時候買、什麼時候賣以及交易數量的規則[1]。當我們有一個策略以後，最通常的做法會是在歷史的資料上去做回測, 即將這個策略運用在過去的資料上看看表現如何，作為我們評估策略的一個依據。

這聽起來很好：我們通過某些方法去尋找策略，然後把這些策略應用在歷史股價上，然後得到一個諸如 Sharpe ratio（夏普值）等指標來評估策略的表現，然後我們選擇指標優秀的諸策略，最後在真實世界採用這些策略，接著我們就可以享受美好人生了。但是真實的情況真的是這樣嗎？從實際上來說，更多的情況會像是圖 2.1 那樣：綠色的是我們回測所得到的權益曲線，而紅色的資金曲線則代表了殘酷的現實。

這部分原因很大一部分來源於對於資料的過度勘探（data snooping），過度勘探代表著於對於某一個模型嘗試過多的配置（configuration）而不考慮因嘗試次數增多而增加的出現偽陽性（false positives）的幾率。實際上，大多數的情況策略和配置並不賺錢。

本研究的目的是希望可以提出一個回測的架構，一個基於資料的生成的架構。希望借用對抗生成網絡（GAN）來學到金融市場的某些內涵的性質和資料的生成過程。

而之所以選擇 GAN 的原因如下：首先 GAN 作為 2014 年提出的方法，在圖像、文字方面有諸多應用如 [2], [3]等等，但是在金融上的應用還沒有人研究，

doi:10.6342/NTU201801645

所以本研究的一個貢獻在於將 GAN 應用到了金融市場資料上。

其次，如果把時間序列看待成向量則格式和圖像一樣，GAN 在圖像上的優異表現讓我們對於他在金融時序上的表現有信心。另外 GAN 與現有的其他時間序列資料生成方法相比，即使兩者皆不假設模型，它們仍有一個很大的區別：後者大多數不改變資料，而是將現有資料進行切割再組合[4]，這很大的一個原因在於，我們並沒有簡單的公式來定義這些資料的分佈。例如，我們很難用數學去描述什麼樣的三維矩陣代表是人臉的 RGB 圖的分佈。然而 GAN 卻可以利用類神經網絡去趨近真實的分佈以及其特殊的訓練方法找到一個分佈，並利用該分佈來生成資料。

最後相對於在語音、圖像甚至一些 IOT 中對於物件發生故障的預測中的表現優異[5]，類神經網絡在預測股票價格、股票漲跌方面的表現其實並不如人意[6]–[9]。雖然 GAN 的生成器與判別器皆為類神經網絡，但是作為一個生成模型，並不被要求不能看到未來的資料，本論文顯示類神經網絡可以在金融領域發揮更多的用處。

1.2 相關研究

對於策略過擬合（overfitting）的研究不少。在[10]中，作者們嚴厲的批評了在回測中被忽視的一些問題，包括對於資料的過度的資料挖掘而不考慮過擬合等原因，並且提出一個辦法就是保證回測的資料的最小長度(MinBTL)，作者們的想法是：假設我們從標準常態分佈中獨立且相同分佈抽取 N 次，並計算這 N 個數中最大值的期望值$\mathrm{E}\left[\max_n\right]$是多少。則在 N 次的抽取之下（對應我們嘗試策略的次數），即使出現了$\mathrm{E}\left[\max_n\right]$我們也應該將其視為並不顯著。從年化的夏普值角度來說

doi:10.6342/NTU201801645

$\mathrm{E}_{\mathrm{annual}}\left[\max_n\right] \approx y^{-1/2}\mathrm{E}\left[\max_n\right]$，其中 y 為樣本的長度，則如果加長 y 我們可以使得$\mathrm{E}_{\mathrm{annual}}\left[\max_n\right]$從而使實驗的夏普值變得更加合理（例如$\mathrm{E}_{\mathrm{annual}}\left[\max_n\right]$為零，則大於零的夏普值視為有限而不是一個過擬合的結果）。這一部分的結果，[11]有更深入的探討。

而後[12]中提出了 probability of backtest overfitting 的概念，並給了一個數學上的定義如下：

$$\sum_{n=1}^{N} E[\overline{r_n} \mid r \in \Omega_n^*]Prob[r \in \Omega_n^*] < \frac{N}{2} \tag{1.1}$$

其關於回測過擬合的定義。關於回測過擬合的定義。其中$E[\overline{r_n} \mid r \in \Omega_n^*]$是指樣本內最好的策略 r 在樣本外的排名。公式的直覺是：當有最佳 IS 表現的策略的 OOS 的表現低於平均值。

但是注意一點的是，這個方法對應的是：有多個策略，需要從這多個策略裡選一個，目標是選取的過程不要過擬合。

然而這個定義其實並不是針對某一個策略在某一個資料集上而言，而是針對從多個策略中最好的那個的策略的過程做的探討，即這一過程是否合適（僅關注某一項指標在樣本內的表現是否代表這個策略在樣本外的表現同樣出色），定義的直覺是樣本內的排名最好的策略在樣本外的排名的期望值小於其他策略的平均值。而計算的部分，作者的方法是觀察策略-表現矩陣（即這個矩陣的 column 是，row 是時間）則M_{st}表示策略s在t時刻的表現（譬如損益多少錢），給定一個常數 K，則將M_{st}按照時間分成 K 等分，並且從，這樣便形成一對（樣本內，樣本外），再用這些樣本對去做模擬回測，觀察是否會容易選擇過擬合的策略，這個方法針對的是選取的過程，亦或者說某一個標準是否容易使得樣本內最佳其實

doi:10.6342/NTU201801645

是偽陽性（false positive）。

在[13]從統計角度探討了這個問題：當對同一個數據集、同一個模型的不同配置最大量的搜索的時候，以夏普值為例，多重檢定需要替代原先的單一檢定，這意味著對於測試得到的夏普值需要一個額外的調整（haircut）。雖然對於夏普值的調整在實際應用中已經被使用，但是作者的結果是夏普值的調整並不是線性的，通常來說對於很大的夏普值的調整會很小，因為這暗含著這個策略的顯著性，相對的，對於不那麼大的夏普值的調整就需要大一些，因為這很可能是一個偽陽性的結果。

最後從模擬路徑的角度來探討解決回測過擬合的是[1]，作者將股價的過程假設為 Ornsterin-Uhlenbeck（OU）過程，然後從過去的資料中算出 OU 過程中的參數。接著通過 Monte Carlo 的方法去讓模型生成很多條路徑，在這些生成的路徑上去測試每一個策略的表現，並且觀察討論了同一個策略在這些生成的路徑上的表現。

至於將深度學習應用於金融領域的研究著實不少，在預測金融時間序列方面尤甚，但是似乎並沒有很好的結果的論文出現，這其實可以理解：如果有表現好的模型被發表，勢必會引來人們蜂擁而至的使用，然後根據效率市場假說（efficient market hypothesis），這個模型就失去效果，市場重新再變得效率。此外，我們的研究並未包含預測股價的漲跌。舉一個例子，假設從今天開始，未來的股價只有永遠上漲或者永遠下跌兩種可能，且其概率各為 0.5，則任何人無論對歷史資料做如何的處理或挖掘，皆無法預測股價的漲跌，而只能丟銅板。即使如此，我們藉由回測並對策略未來表現的推論依舊可能是可靠的，比如趨勢性的策略，即那些根據過去的走勢來預測未來的策略，在我們假設的永遠漲的情況下，

doi:10.6342/NTU201801645

趨勢性的策略也會認為股價會一直漲，故會做出買入的動作，反之亦然，所以無論是永遠上漲或永遠下跌都可以得到正收益。因此我們的方法並不需要可以預測股價漲跌的前提保證，即我們的假設比「預測股價漲跌」所要求的要少。

1.3 論文架構

論文的架構如下，第一部分是前言、簡介和過往學者在這個領域的研究的總結。第二部分背景知識會探討一些需要用到的定義以及生成資料的生成方法包括 GAN 及其變形以及它所依賴的循環類神經網絡。第三部分會談論實驗設計以及對於實驗的評估方法。第四部分是實驗結果以及實驗結果的討論。第五部分是結論。

doi:10.6342/NTU201801645

2 背景知識

2.1 符號

我們用 $\{y_t\}^T$ 表示一個長度為 T 的時間序列，而 $\{\{y_t\}^T\}$ 表示長度為 T 的時間序列的集合，$\{y_t\}_k^T$表示第 k 個長度為T的時間序列，有的時候簡寫成 $\{y_t\}_k$。$\{\{y_t\}^T\}^N$ 則表示收集了 N 個長度為 T 的時間序列，有的時候簡寫成 $\{\{y_t\}^T\}^N$ 如果 y 是一個時間序列，則 y_t 代表時間序列第 t 個的值。

此外，如果申明了 P 是一個時間序列的集合（即 P 是$\{\{y_t\}^T\}^N$），則 P_i 表示其中第 i 個時間序列，P_{ij} 表示其中第 i 個時間序列在第j的時刻的值。而用 P_{*j} 表示 P 中所有的時間序列在時間 j 的值的集合。

2.2 策略

2.2.1 定義

策略的定義似乎並沒有一個標準，在[1]中所定義的交易規則代表的是買入和賣出的實際。為了後面討論方便，我們也對策略做了一個自己的定義：我們可以把它看作是將時間序列（例如股價）映射到 $\{-1,0,1\}$ 的函數，其中-1 為做空，0 為不操作，1 為做多。

通常策略 S 會有一些參數 θ 來確定這個策略具體的執行方案，我們將參數 θ 代表策略 S 的配置。我們將 $S(p;\theta)$ 定義為策略 S 在 θ 配置之下，在時間序列 p 上的結果。當我們不關心時間序列時，我們用 S_θ 來表達在 θ 配置之

doi:10.6342/NTU201801645

下的策略 S，具體定義如下：

$$S_\theta \ : \ \{p_t\}^{\mathrm{T}} \to \{-1,0,1\}^{\mathrm{T}}. \tag{2.1}$$

本研究會用到兩種策略，介紹如下。

2.2.2 MAC 策略

MAC 策略即 moving average cross[14]。此模型接受兩個參數 p_1 和 p_2。策略會計算距今 p_1 期的股價的均值 MA_1 以及距今p_2期的股價的均值 MA_2。若$p_1 > p_2$ 則讓部位（position）變成 1，反之則讓倉位變成-1。更具體的資訊參考表格 2.1。

MAC	策略配置	
	p_1	p_2
參數範圍	1~50	1~50
參數解釋	用於過去p_1期的均值	用於過去p_2期的均值

表格 2.1 MAC 策略代表的信息

2.2.3 BH 策略

買入持有策略(buy and hold，簡稱 BH)會接受四個參數 entry、hold、stop-loss 和 side。策略的邏輯是將交易日切割成月份，交易在每個月如此進行：在 entry（例如:1~31）日進入，持有 hold 天然後賣出[15]。且如果在持有期間的損失超過了 stop-loss 則 提前賣出。具體請見表格 2.2

doi:10.6342/NTU201801645

BH	策略配置			
	entry	hold	stop-loss	side
參數範圍	1~30	1~30	0~20	$\{-1,1\}$
參數解釋	一個月哪一天操作	在進場以後持有多久	持有期間最大容忍損失	做多或做空

表格 2.2 買入持有策略的信息

2.3 回測

2.3.1 介紹

所謂回測是指在歷史上去模擬一個交易策略，然後通過計算某個指標來幫助判斷交易策略的優劣。

定義一個回測函數 B 為

$$B\colon \{p_t\}^{\mathrm{T}} \times \{-1,0,1\}^{\mathrm{T}} \to \mathrm{R} \tag{2.2}$$

將策略S的結果以及路徑映射到一個實數值作為衡量策略S的好壞。具體的B會根據不同的交易目標而有所不同，但是我們規定 B 的值越大代表策略越好（否則就乘以-1），我們將 B 的值稱為策略的表現。

我們研究中使用的是常見的夏普值：

$$SR\colon \{p_t\}^{\mathrm{T}} \times \{-1,0,1\}^{\mathrm{T}} \to \frac{\mathrm{E}[X] - R_f}{\sigma[X]} \tag{2.3}$$

其中X是用$\{p_t\}^{\mathrm{T}}$和$\{-1,0,1\}^{\mathrm{T}}$算出每一對交易的損益。

定義了回測函數B和策略S之後，我們接著探討交易員的目標。給定策略S，

doi:10.6342/NTU201801645

通常交易員希望找到一個配置 θ^* 以最大化策略的未來表現，即

$$\theta^* = \underset{\theta}{\operatorname{argmax}}[S(P_{\text{future}}; \theta)]. \tag{2.4}$$

由於我們無法獲得未來的資料, 所以交易員只能在過去的資料上去找 θ^*, 即：

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\left[S\left(P_{\text{history}}; \theta\right)\right]. \tag{2.5}$$

2.3.2 回測的過擬合

解(2.5 的過程很有可能發生量化交易中最令人頭疼的問題：過擬合，詳細的討論請見後面章節。以下這邊用一個例子來說明。

假定我們有一個買入持有策略，稱其為 S'。我們假設股價的生成是隨機遊走：

$$P(t+1) = P(t) + \epsilon(t) \;\; \epsilon(t) \sim N(0,1) \tag{2.6}$$

我們用(2.6 生成 600 天的資料，將前 300 天當作歷史資料，而將後 300 天當作未來資料，我們的目標是用(2.5 在歷史資料上找到最好的配置 θ'^* 得到策略 S'^*，然後將 S'^* 應用到未來資料上，並考察其表現。

首先，一個結果是，在(2.6 所生成的路徑上，策略 S' 的期望值理論上應該不會大於 0。這是因為策略S'如果提前離場，則損益一定是負的。但如果沒有提前離場，則損益僅和入場和出場的價格有關，假設在 t 時間入場，持有 L 天，又因為

$$E[P(t+L) - P(t)] = E\left[\sum_{k=t}^{t+L} \epsilon(k)\right] = 0 \tag{2.7}$$

可知價差的期望值為 0。

但是如果我們遍歷表格 2.2 所有的參數的組合，選取在歷史資料上有最佳表

doi:10.6342/NTU201801645

現的配置圖 2.1 會是結果。綠色的部分是我們的在歷史資料上所得到的權益曲線（equity curve，記錄投資過程中的資金的變化），而紅色則是在未來的資料上得到的權益曲線。綠色的權益曲線顯然報酬不為 0。另外，如果我們不考慮上述期望值為 0 的結論，僅僅從圖上我們也可以看出一個很明顯的結果：歷史的表現雖然很好，但是未來的表現則與歷史的表現相差甚遠。

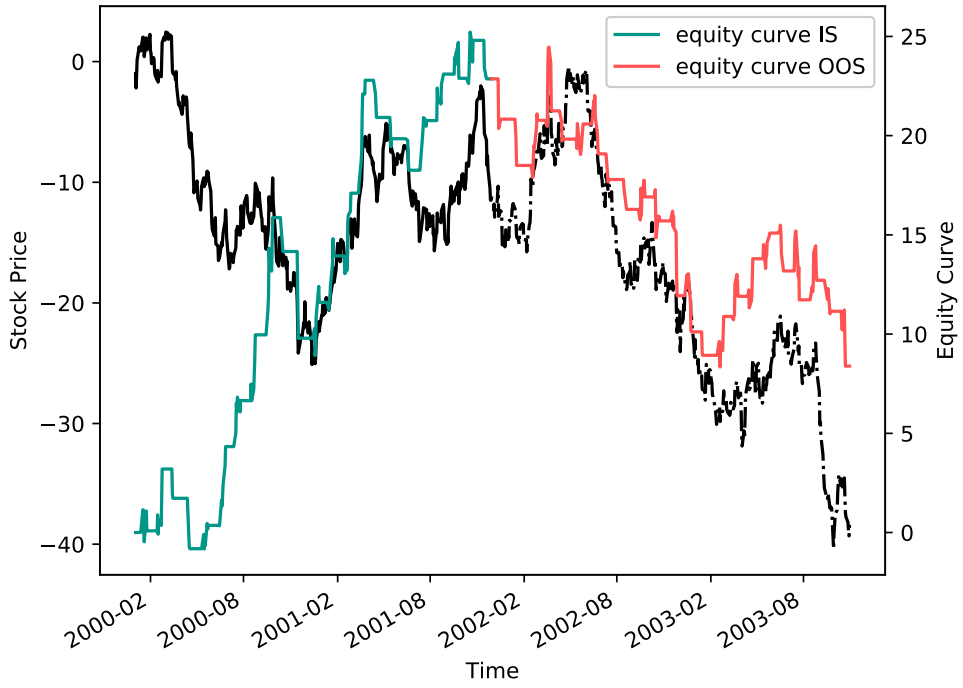


圖 2.1 對應右邊的 Y 軸其中綠色部分為樣本內的權益曲線，而紅色為樣本外同一個策略的權益曲線。對應左邊的 Y 軸，黑色實線是樣本內的價格，黑色虛線是樣本外的價格。

doi:10.6342/NTU201801645

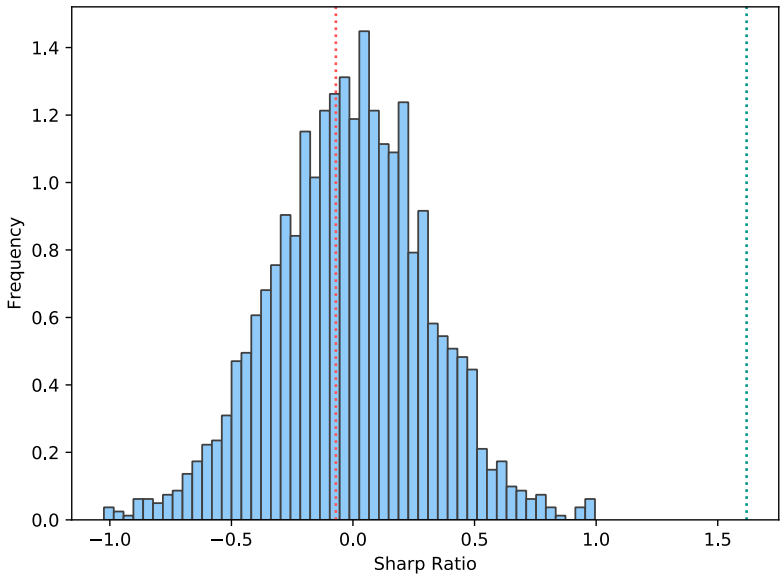


圖 2.2 統計圖 2.1 的策略在生成的 2000 個路徑上的夏普值的分佈，綠色虛線代表著圖 2.1 中 IS 的夏普值的分佈，紅色虛線代表 OOS 的。

而這樣的結果原因很簡單：這個策略的配置在樣本內之所以表現這麼好是因為這個策略的配置恰好擬合在樣本內的雜訊上，例如回測的資料恰好在每個月的 12 日到 17 日都是上漲，且策略的參數恰好也在 12 日買入持有 5 天，但(2.6 表明未來 12 日到 17 日上漲和下跌的概率是一樣的，所以該策略在 12 日到 17 日實際上並不會賺錢。

為了更進一步的說明，由於我們知道股價的生成公式(2.6，我們利用蒙地卡羅模擬（Monte Carlo）來做另一個實驗：我們將 S'^* 應用到 2000 個用(2.6 生成的股價上並得到回測的結果。結果是圖 2.2。其中藍色的頻率圖代表 S'^* 在所有生成的路徑上的夏普值的頻率分佈，而綠色垂直線對應的是在樣本內的表現，我們可以看到，這一表現明顯是異常值（outlier），另外我們可以看到這個近似常態分佈的頻率圖中的均值落在 0 附近，這和我們前文所說期望值為 0 相吻合。從這個小的實驗我們可以感覺到也許參考多個路徑上的回測結果是一種更好的回

doi:10.6342/NTU201801645

測的目標，更進一步的討論見下文。

另外的一個角度考察過擬合可以從參數配置對應表現的平滑程度來考察，我們用 MAC 以及(2.6 和白噪音：

$$P(t) = \epsilon(t) \;\; \epsilon(t) \sim N(0,1) \qquad (2.8)$$

作為例子。要說明的是 MAC 在(2.6 的隨機遊走程序上其期望值為 0，而在(2.8 上則存在期望值大於 0 的策略，結果參考圖 2.3 及圖 2.4。圖上展現的是遍歷兩個參數所有配置的表現：越偏向藍色代表夏普值高即回測效果好，越偏向紅色則相反。在隨機漫步上，深藍色的部分（即對應配置的夏普值顯著大於 0）並不是平滑的呈現的而是跳躍的呈現，而在白噪音上的結果則光滑的多。這個例子表明了，如果找到一策略的配置發現其回測的結果很好，然而他周圍的配置的回測的表現很差如圖 2.3，則我們應該要憂慮這可能是一個過擬合的結果。反之，如果回測函數越平滑，則表明配置過擬合的幾率會越小如圖 2.4。

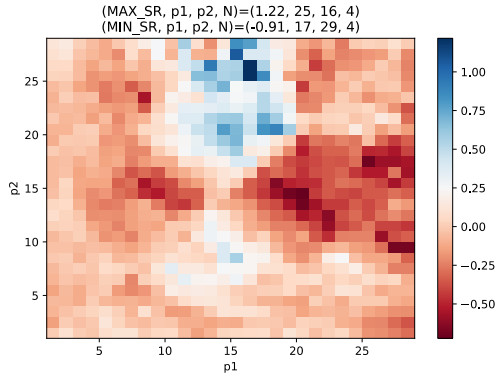


圖 2.3 在隨機漫步上 MAC 的夏普值熱點圖

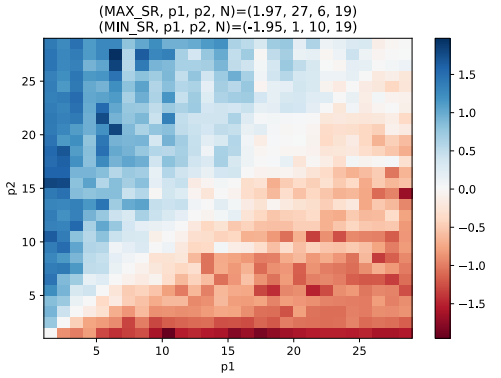


圖 2.4 在標準白噪音上 MAC 的夏普值熱點圖

2.3.3 回測的目標

上面顯示了過擬合是如此容易地發生，這很大程度是上因為 θ^* 是來源自擬合了過去的雜訊，尤其是在配置θ的空間很大的時候。

doi:10.6342/NTU201801645

為了說明這一點，舉個例子。如果有一個策略S_θ，我們令其配置 θ 是一個維度為 100 的向量，每一維度的向量取值$-1/0/1$，且 S_θ 就將 θ 當作其輸出：

$$S_\theta: \{p_t\}^{\mathrm{T}} \to \theta \tag{2.9}$$

這顯然是一種非常糟糕的策略：給定一個配置 θ，θ 並不會根據回測的資料而改變，則無論資料是什麼，這個策略給出同樣的交易訊號，這顯然不合理。但當我們將這個策略用在任意長度為 100 的歷史資料上的時候會發生什麼呢？如果我們搜索整個θ可以取的空間：$\{-1/0/1\}^{100}$，總共有3^{100}個可能，則我們一定可以找到一個配置 θ^*，它恰好就對應到完美的那個交易訊號的向量，對應的回測效果是：策略 S_{θ^*} 在回測的這一筆資料的所有交易均在低點買入，在高點賣出。之所以會有這個結果是因為我們定義的策略及 θ 使得產生的交易信號覆蓋了所有的交易可能，當然也包括那個完美的交易策略。本論文的核心目標是盡可能避免這個情況的發生。

在我們前面的假設之下，即時間序列 P 有一個資料產生過程 μ，此時我們將交易員的目標由(2.5)改為：

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathrm{E}_\mu \left[B\left[S(P_\mu; \theta) \right] \right]. \tag{2.10}$$

其中 P_μ 為 μ 抽樣的路徑。公式即最大化在某個資料生成模型所生成的樣本上的表現的期望值。這是合理的，理由如下：用期望值的方法我們也許可以去掉某一條特定的路徑中的雜訊。這在機器學習的其他領域也有類似的概念，被稱之為 bagging [16]。

實際操作過程中，我們用抽樣的平均值來近似期望值，所以(2.10 變成：

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{\mu \in A_N} B\left[S(P_\mu; \theta) \right]. \tag{2.11}$$

doi:10.6342/NTU201801645

其中 A_N 是抽取樣本路徑的集合。綜上來說，我們定義了好的策略的配置 θ^* 為最大化資料生成模型之下生成的樣本上的表現的平均值的那一個。

2.4 類神經網絡

2.4.1 類神經網絡

類神經網絡被提出已久，最早的起源是模仿大腦神經的結構。

從數學上的抽象的結果可以參考圖 2.5。其直覺是，用許多簡單的神經元組成一個龐大的系統以逼近任何的函數。從某一個神經元的監督來看，其輸入的神經元的值會被加權的傳遞到其所鏈接的神經元上，接著，加權的值會通過一個激發函數（activation function），常用的激發函數參考表格 2.3。最後輸出，當作這個神經元所連接的神經元的輸入。

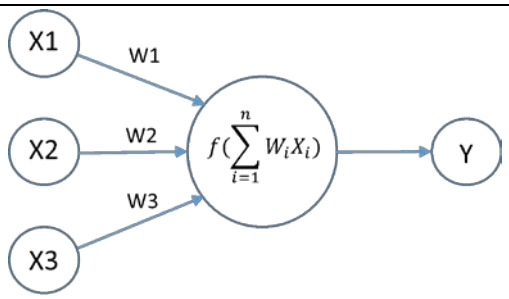


圖 2.5 神經元。展現的是單個神經元的輸入與輸出。f就是激發函數，而 $\sum_{i=1}^{n} W_i X_i$是被加權的綜合。

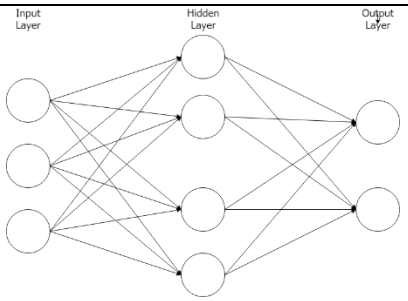


圖 2.6 類神經網絡。由圖 2.5 組成。

傳統的神經網絡分為輸入層、隱藏層以及輸出層，如圖 2.6。現在深度學習時代一大特徵便是隱藏層越來越深，這得益於 CPU 和 GPU 能力的提升，使得訓練深的網絡變得可行。

名稱	公式
sigmoid	$\frac{1}{1+e^{-x}}$

doi:10.6342/NTU201801645

tanh	$\frac{e^{2x}+1}{e^{2x}-1}$
RELU	$\max(0, x)$

表格 2.3 常見的激發函數。

以上是類神經網絡的概念。接下來我們需要訓練網絡中的參數，最著名的方法便是使用反向傳播算法（backpropagation）。在常用的深度學習庫中，反向傳播算法的實作是基於計算圖（computational graph），這使得算法的實作變得簡單以及維護變得容易。

計算圖的定義如圖 2.7 所示。每一個 node 定義變量的名稱以及計算方式，計算可以使用的變量只能是被指向的 node 所定義的變量。

在 forward 的階段，計算很直覺，就是將輸入（即圖 2.7 中的 a 和 b）代入，然後根據拓撲順序去計算即可。

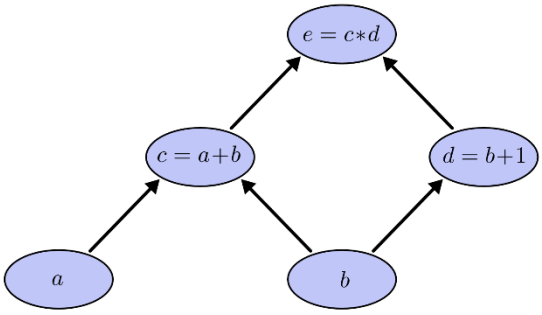


圖 2.7 計算圖的 forward 過程。圖片來源：[17]。

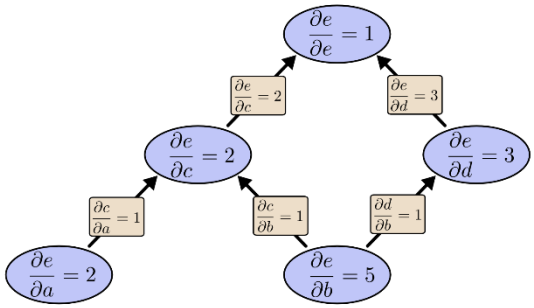


圖 2.8 計算圖的 back propagation 過程。圖片來源：[17]。

而在反向傳播算法的過程，首先計算每個 node 中的偏微分，如圖 2.8，接著用 forward 反方向計算即可，如果一個 node 有多個流入，則將他們的偏微分相加即可。　可以看到，所謂的反向傳播算法實際上是鏈鎖定則（chain rule）的一個表現。

doi:10.6342/NTU201801645

2.4.2 長短期記憶模型類神經網路

長短期記憶模型類神經網路（LSTM）是由[18]所提出的。不同與前面所提到的類神經網絡，LSTM 不光用到了當前的輸入與輸出，LSTM 還用到了前面時刻所計算的結果，如圖 2.9 所示，其中σ代表的是 sigmoid 函數。

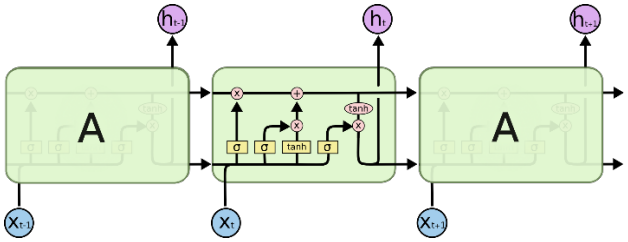


圖 2.9 LSTM 的結構，圖片來源[19]。可以看到這邊每一期的輸入除了X_t以外，前面的X_{t-1}所產生的結果也當做了第t期的輸入。其中每一個圓圈里定義了流出的資料是如何由流入的資料計算所得的。

LSTM 的資料流可以參考圖 2.10、圖 2.11、圖 2.12 和圖 2.13。

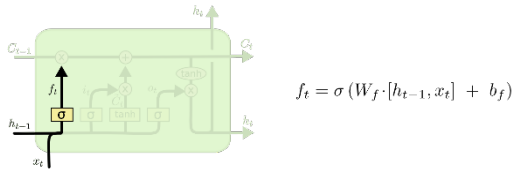


圖 2.10 計算遺忘的比例的示意圖。圖片來源：[19]。

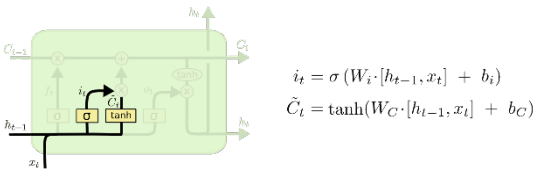


圖 2.11 新的信息的計算示意圖。圖片來源：[19]。

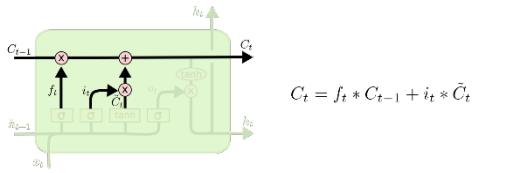


圖 2.12 更新記憶的示意圖。圖片來源：[19]。

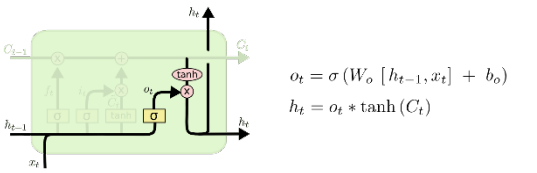


圖 2.13 選擇輸出的示意圖。圖片來源：[19]。

同樣的，在循環神經網絡的訓練部分，同樣是使用反向傳播算法，不過不同於一般的 back propagation，由於用到了前面的資訊，所以當前面的權重需要訓練的時候，必須要考慮到後面的結果，這被稱之為 BPTT (back propagation through

doi:10.6342/NTU201801645

time)。具體過程可以參考圖 2.14 和圖 2.15。

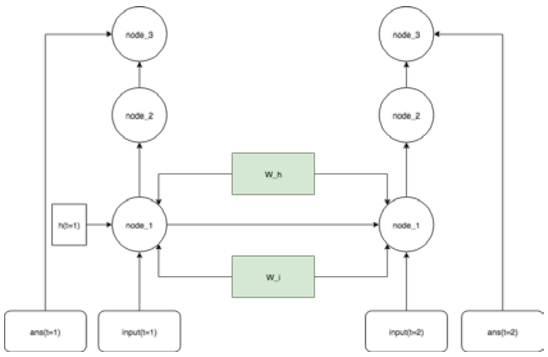


圖 2.14 循環類神經網絡計算示意圖。可以看到與一般的類神經網絡不通的地方在於，node_1 使用了共享的權重

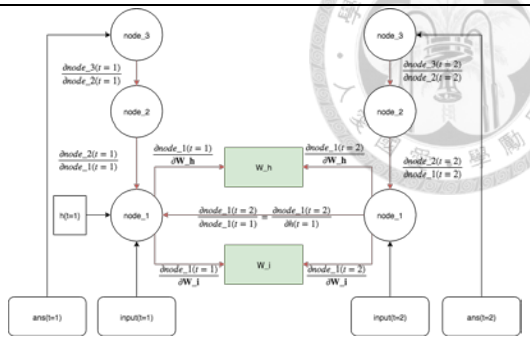


圖 2.15 BPTT 示意圖。

2.5 生成對抗網絡 (GAN)

2.5.1 介紹

Ian Goodfellow 在 2014 年在[2]提出的 Generative Adversarial Networks 的概念在近幾年中掀起了一陣研究熱潮。有數不勝數的論文出現，均為其各種變形。這邊先介紹一下 GAN 的基本原理然後再介紹幾個著名的變形。

2.5.2 GAN

幾乎所有的 GAN 的思想都是一致的：有一個生成器 G，他負責將某一個潛在空間（latent space）上的分佈 Z 轉換到我們想要的分佈 T 上。做法是從潛在空間中抽出一個樣本，然後將其丟給生成器，生成器會生成一個分佈 T 的樣本。

在過程還需要一個鑒別器 D，他負責區分資料是來自 T 分佈亦或是生成器從 Z 分佈生成而來的。

Goodfellow 在[2]中提出的第一代的 GAN 是這樣的

doi:10.6342/NTU201801645

$$\min_G \max_D V(D,G),$$

其中

$$V(D,G) = E_{x\sim p_T(x)}[\log D(X)\}] + E_{z\sim p_z(z)} \log[1 - D(G(z))]$$

$$= E_{x\sim p_T(x)}[\log D(X)\}] + E_{x\sim p_G(x)} \log[1 - D(x)]$$

這是一個博弈論中 Min-Max-Problem 的體現。上面式子直覺上可以這樣理解。首先，在$\max_D V(D,G)$的部分，我們固定生成器G的參數，此時想要找一個鑒別器D使得$V(D,G)$最大越好：這意味著找一個新的鑒別器D'，D'可以相比當前的D更能分別資料是否來自 T 分佈。接著，在找到D'之後，我們要做的是$\min_G V(D',G)$，即尋找新的生成器G'，它要使得$V(D',G)$越小越好：這意味著找一個新的生成器 G′，G′可以相比當前的 G 更能欺騙D'資料是否來自 T 分佈。如上面所說，循環往復的訓練以達到這樣的效果：D越來越強大，分辨能力越來越強，而 G 也會因為變得生成的樣本越來越像來自 T 分佈的樣本。

不妨看一下$V(D,G)$的意義是什麼，

$$V(D,G) = E_{x\sim p_T(x)}[\log D(X)\}] + E_{x\sim p_G(x)} \log[1 - D(x)]$$

期望值沒有辦法計算，所以我們由抽樣的方式來替代：

$$V(D,G) = \int P_T(x) \log D(X)\, \mathrm{d}x + \int P_G(x) \log[1 - D(G(z))]\, \mathrm{d}x$$

$$= \int P_T(x) \log D(X) + P_G(x) \log[1 - D(G(z))]\, \mathrm{d}x$$

則令：

$$D^* = \operatorname*{argmax}_D P_T(x) \log D(X) + P_G(x) \log[1 - D(G(z))]$$

求導可得

$$D^* = \frac{P_T(x)}{P_T(x) + P_G(x)}$$

接下來，

doi:10.6342/NTU201801645

$$\min_G \max_D V(D,G) = \min_G V(D^*,G)$$
$$= \min_G \int P_T(x)\log\frac{P_T(x)}{P_T(x)+P_G(x)} + P_G(x)\log\frac{P_G(x)}{P_T(x)+P_G(x)}\mathrm{d}x$$

化簡之後得到

$$V(D^*,G) = -2\log 2$$
$$+ \int P_T(x)\log\frac{P_T(x)}{\frac{(P_T(x)+P_G(x))}{2}} + P_G(x)\log\frac{P_G(x)}{\frac{(P_T(x)+P_G(x))}{2}}\mathrm{d}x$$
$$= -2\log 2 + \mathrm{KL}\left(P_T(x)||\frac{(P_T(x)+P_G(x))}{2}\right)$$
$$+ \mathrm{KL}\left(P_G(x)||\frac{(P_T(x)+P_G(x))}{2}\right) = -2\log 2 + 2\mathrm{JSD}\left(P_T(x)||P_G(x)\right)$$

我們可以看出$V(D^*,G)$的意義其實衡量的是P_T和P_G的 Jensen-Shannon divergence，這時我們最小化的其實是 Jensen-Shannon divergence。

此外，對於 GAN 的變形有很多參考，可以將其大概分為幾種。一為從 D 或者 G 的架構的改變，例如從 fully connected 的 NN 到 convolutional 的 NN 或是改用 LSTM 等 RNN 的模型。二為修改輸入的結構，例如加入某些各位的資訊將其當做 GAN 生成、鑒別之條件，如[20]中就將 MNIST（一個手寫數字的資料集）中的數字的標籤當做條件，使得生成器生成對應標籤的數字的手寫圖片。還有一種便是修改對於$V(D,G)$的定義，諸如變成其他的 divergence，具體[21]做了更多更深入的探討。

2.5.3 RGAN

RGAN 中的 R 對應的意思是生成器和鑒別器均是 recurrent neural network。RGAN 是今年早些時候在[20]中被提出的。RGAN 被提出的目的是為了解決真實病人資

doi:10.6342/NTU201801645

料被用作研究資料時的隱私問題，RGAN 的示意圖可以參考圖 2.16 與圖 2.17。

在[20]中，目標的資料是一個維度為四的時間序列，四個維度分別是：脈搏血氧、心跳頻率、呼吸頻率和平均動脈壓。可以看到，如果不關心數值具體的為物理意義，我們完全可以講開盤價、最高價、最低價和收盤價代換資料帶到 RGAN 的框架中。

另外，在本研究中，我們將 RGAN 的判別器的部分從 LSTM 換成了雙向的 LSTM，這樣做的目的在於增強判別器的能力。

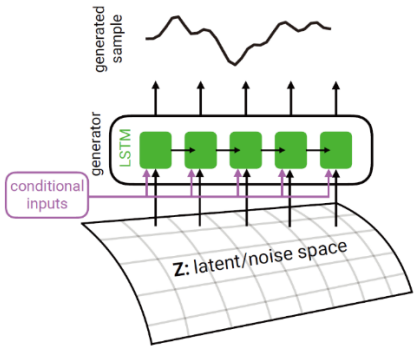


圖 2.16 RGAN 的生成器的架構，輸入包含來自原始分佈和條件，經過 LSTM 的結果變成生成器的輸出。圖片來源：[20]。

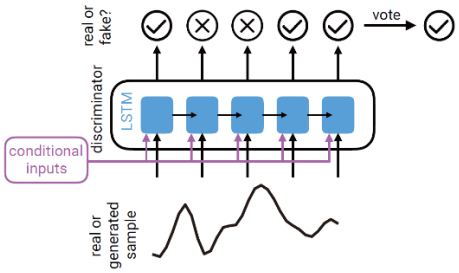


圖 2.17 RGAN 的鑒別器的架構。輸入是目標分佈的資或者生成器所生成的資料，經過 LSTM 以後輸出的結果是 0/1向量，這裡並沒有最後再經過一個全連接層變成一個 0/1 而是就當作有時間長度這麼多的 binary。圖片來源：[20]。

3 研究方法

3.1 學習路徑

3.1.1 實驗目的

首先，我們在常見的資料生成模型上看看 GAN 在其上面的表現：包括能否學到

doi:10.6342/NTU201801645

這個模型、對於資料量的要求等等。從驗證的角度，我們使用的金融領域常假定的股價模型 GBM。

3.1.2 模型說明

圖 3.1 是我們的模型的一個總覽: 從 latent space 抽樣的資料會丟給一個生成器。生成器的輸出和真實資料的形狀是一樣的，因此他們會分別丟給鑒別器，由於生產的資料以及真實的資料是時間序列, 故每個時刻的鑒別器也會對應的有一個輸出，當全部時間的資料輸入完畢後，我們將每一期的鑒別器對於生成資料的輸出以及鑒別器對於真實資料的輸出分別匯總，最後放在一起算 cross entropy 當作 loss。

圖 3.1 是模型的總體架構，分開來看，在 latent space 的部分，我們選取的是 5 維，且各個維度為獨立的標準常態分佈。

在生成器部分，我們採用的是 LSTM 模型，其中隱藏層節點的個數，我們在實驗部分有測試不同的數量做比較。隱藏層傳遞的激活函數使用 sigmoid，輸出層的激活函數使用 tanh。

而在判別器部分，我們嘗試了單項的 RGAN 和雙向的 RGAN，隱藏層傳遞的激活函數使用的 sigmoid 以及輸出層的激活函數使用的是 tanh，而其中隱藏層節點的數量是我們另外做了實驗來比較。

對模型影響最大的部分是 LSTM 的隱藏層節點的數量，其對於參數的影響參考表格 3.1。

doi:10.6342/NTU201801645

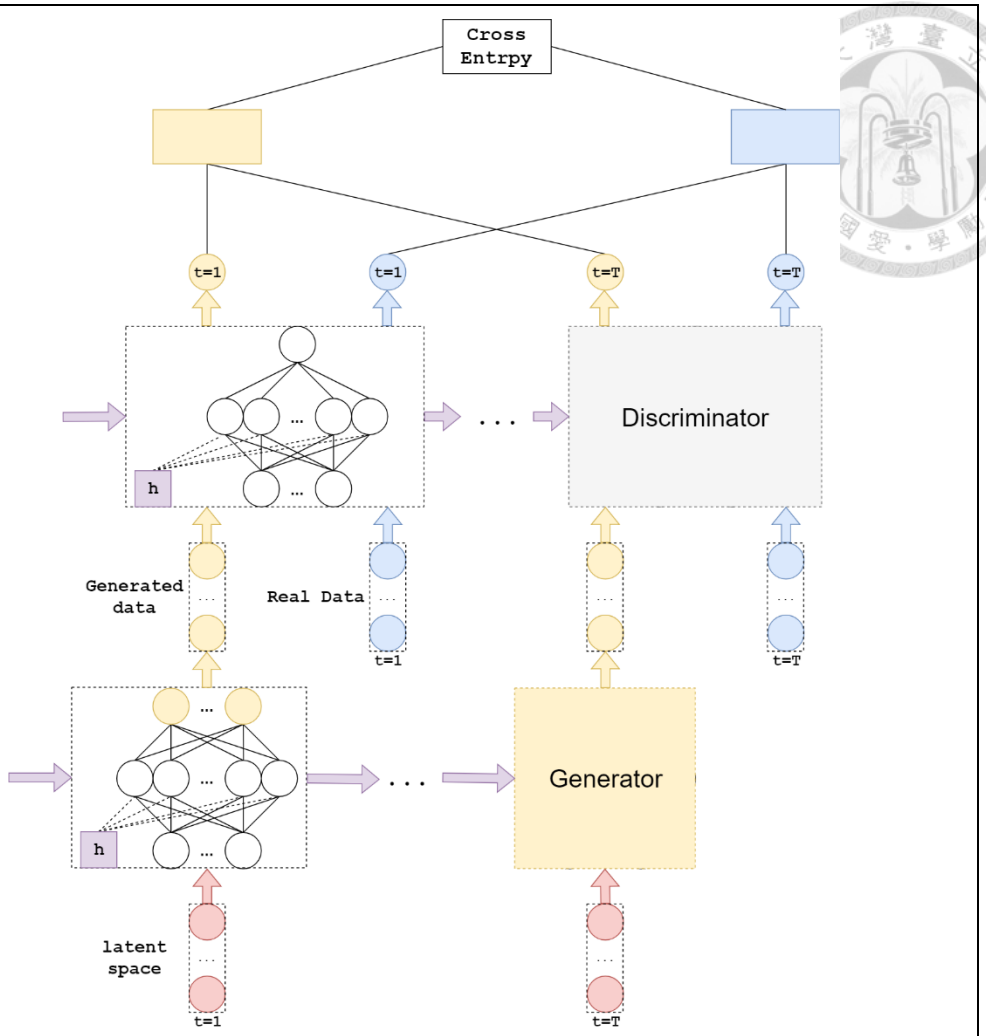


圖 3.1 模型的架構。圖中黃色的部分是生成器相關，藍色的部分是真實資料相關，紅色資料則是來源於 latent space 的資料。

		參數量		
		生成器	鑒別器	總數
隱藏層節點的數量	100	42,901	82,800	125,701
	50	11,451	21,400	32,851
	10	691	1,080	1,771

表格 3.1 隱藏層節點數量與參數量的關係。

doi:10.6342/NTU201801645

3.1.3 資料集

3.1.3.1 模型

我們選取 RGAN 的目標隨機過程選擇的是 GBM:

$$\mathrm{d}y_t = \mu y_t \mathrm{d}t + \sigma y_t \mathrm{d}W_t \tag{3.1}$$

其中W_t是 Brownian motion，解上述 SDE 可以得到:

$$y_t = y_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right) \tag{3.2}$$

根據其性質我們有:

$$\ln y_t \sim N\left(\left(\mu - \frac{\sigma^2}{2}\right)t, \sigma^2 t\right) \tag{3.3}$$

選擇這個隨機過程基於以下的原因：GAN 的評估以及對於時間序列是否來自某一個樣本的檢定一直是一個開放的問題（參考[20][22]），尤其在 GAN 常用的圖像領域的判斷條件多是人眼，而時間序列卻很難採用同樣標準。使用(3.2 時我們可以做統計上的檢定：當我們有模型生成的時間序列的集合 $\{\tilde{y}^T\}^N$，我們可以檢驗 $\ln \tilde{y}_{t,*}$ 的分佈：對於每一個t，$\ln \tilde{y}_{t,*}$ 是否為常態分佈以及其分佈的均值和變異數是否趨近線性方程 $\left(\mu - \frac{\sigma^2}{2}\right)t$以 及 $\sigma^2 t$？在實驗中我們選擇 $\mu = 0.1$, $\sigma = 0.1$ 以及 $y_0 = 1$。

3.1.3.2 生成樣本

有了模型，接下來是樣本的部分。我們的樣本是這樣來的：用我們假設的模型生成相同長度為 *T* 的時間序列一共 *N* 個，這 *N* 個便作為我們的訓練樣本。試驗中我們固定 *T*=100。

doi:10.6342/NTU201801645

而資料量 N 這一點相當的重要。因為如前面所述，我們的樣本是在給定模型下生成的，而真實的情況下我們並沒有真實的模型（否則我們也不用這麼麻煩地去學習）。參看金融股價資料，資料量其實並沒有我們想象的那麼多：為了要得到樣本，我們勢必要用某種假設（例如每個月的資料是相似的，則我們可以將每個月的資料當作是一個樣本），用某些方法將真實的資料切割成多份，當作此假設之下的多個時間序列樣本，這樣生成的樣本數量其實並不多不再那麼多了: 比如，我們有一檔股票為期 10 年的金融資料，假設每個月的股價有統計上的相似性，這樣其實真正可以被我們當做樣本數量只剩 120 筆。即使我們把假設限定到每日的相關性，10 年的資料總量上是 3600 左右，比圖像領域的常見的資料集如 MNIST 要少得多。N 的個數的研究在實驗結果當中有討論。

3.1.3.3 預處理

另外，由於生成器最後的激活函數是 tanh，這意味著值域在$(-1,1)$，所以我們需要將樣本標準化到$(-1,1)$的區間，這邊我們了使用了：

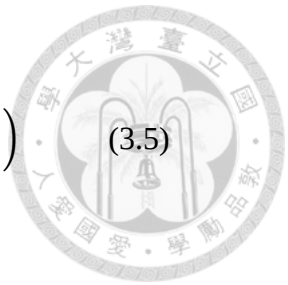
$$\text{sample}_{\text{new}} = 2 * \left(\frac{\text{sample}_{\text{raw}} - \text{sample}_{\text{min}}}{\text{sample}_{\text{max}} - \text{sample}_{\text{min}}}\right) - 1 \qquad (3.4)$$

其中 $\text{sample}_{\text{raw}}$ 泛指模型生成的樣本，$\text{sample}_{\text{min}}$ 和 $\text{sample}_{\text{max}}$ 分別代表模型生成的樣本的最小值和最大值，而$\text{sample}_{\text{new}}$代表我們真正給 RGAN 使用的樣本。

另外，由於(3.4 會使得樣本內最大（小）值為 1（-1），所以當 RGAN 要生成長度超過 T 的時間序列會碰到問題: 生成的最大值被限制了。所以我們對資料預處理從(3.4 變成了

doi:10.6342/NTU201801645

$$\text{sample}_{\text{new}} = \frac{1}{\text{scaling}} * \left(2 * \left(\frac{\text{sample}_{\text{raw}} - \text{sample}_{\text{min}}}{\text{sample}_{\text{max}} - \text{sample}_{\text{min}}}\right) - 1\right) \quad (3.5)$$



其中 scaling 是一個我們挑選的常數。

3.1.4 評估方法

對於不同的模型的評估理應不同，由於對於時間序列的分佈的檢定相當的困難[20], [22]，但是由於我們選擇的 GBM，使得我們可以對它每一期的值取自然對數，則每一期的分佈將會是一個常態分佈，並且其在時間尺度上的期望值和變異數也會和時間成線性的關係。在訓練中，要評判 GAN 學得是否出色的標準很模糊[20]，但是由於我們選的時間序列的特徵征很明顯（期望值和變異數在時間上成線性增長），所以我們用理論上的線性方程當作我們的目標，與 GAN 生成的資料去計算R^2，如圖 3.2 和圖 3.3，而整個訓練的過程的討論，包括資料量、參數量等等會在實驗結果與分析中給出討論。

doi:10.6342/NTU201801645

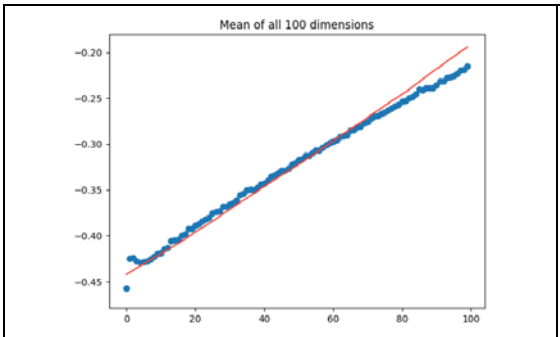


圖 3.2 顯示的是生成的時間序列(取 10000 條)在每一個時刻的期望值。橫軸是時間序列的時間點，縱軸是期望值。顯示出理論值(紅線)和實際值(藍點線)的擬合的情況（GBM 模型），這邊的顯示已經相當不錯了，此時可以停止學習。

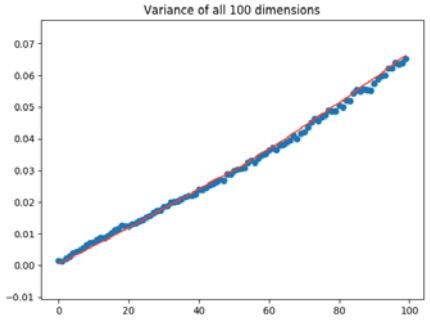


圖 3.3 橫軸是生成的時間序列的每一個時刻，縱軸是變異數。其餘描述同圖 3.2。

3.2 策略與模型路徑

3.2.1 實驗目的

接著，從交易策略的角度来看看 GAN 的结果。實務上，GAN 的訓練是很困難的[23]，有時候 GAN 所學習到的結果並不是完全的那麼理想，這樣的話，真的對於回測就完全沒有幫助了嗎？

並不是這樣的。舉一個簡單的例子，假設我們真正的模型是隨機遊走模型，即下一期的價格等於這一期價格加上一個來源於標準常態分佈的值，比如每一期的常態分佈的均值，標準差不同不會影響，甚至不通過常態分佈的檢定也沒有影響，重要的是它是對稱，並且均值在 0（這意味著下一期的盈利的期望值是可以計算並且有信賴區間來保障的），可見，是否學到完整的股價的生成分佈對於策略來說可能並不是那麼的重要。

這一部分的具體的做法是使用有假設的股價的生成過程，並且我們研究在這些股價上理論上可以期望值為正的策略以及期望值為零的策略（在 OOS 上），這樣

doi:10.6342/NTU201801645

我們便有一個 OOS 上策略表現的參考值。基於這個參考值我們便可以比較集中生成策略的方法，比較誰的結果與參考值更接近。

3.2.2 模型介紹

在 RGAN 的部分我們選擇和上述一樣的機構。在這部分實驗，考慮策略的部分。本實驗一共測試了兩種策略。

3.2.3 資料集

在 GBM 的基礎上，這邊實驗加進了一個新的過程 AR(2)：

$$y_t = a + by_{t-1} + cy_{t-2} + \epsilon_t \quad \epsilon_t \sim N(0,1) \tag{3.6}$$

為了 GBM 做對照，我們選取了$a = 0$，$b = 1.1$，$c = -0.5$，這樣選取的原因是，可以使得(3.6 的特徵多項式（Characteristic Polynomial）的根的絕對值小於 1，這保證了(3.6 是穩定的（stationary）。

另外，我們將理論上的各個策略在各個過程上理論上的結果列在了底下。

BH 和 GBM

由於 GBM 的對數期望值在時間上是線性增加的（見(3.3），所以 BH 在持有的期望值的確應該為正。

BH 和 AR(2)

由於我們的 AR(2)在時間上的期望值是不變的（因為我們的 AR(2)是平穩過程），所以持有多久的期望值都為 0。

MAC 和 GBM

doi:10.6342/NTU201801645

由於GBM的過去對於未來並沒有影響, 且在我們的模型以及參數之下, GBM對數的期望值在時間上是線性增加的，所以 MAC 在持有的期望值的確應該為正。

MAC 和 AR(2)

由於我們的 AR(2)的 stationary，即過去的均值可以當作期望值的一個估計。所以當策略發現當期價格低於這個均值的時候（即p_1選夠長而p_2為 1 即當期），選擇買入反之賣出便可以保證其網上是盈利的。

表格 3.2 對於是上面所說的總結。有、無表示理論上是否有使得策略期望值為正的參數。

	GBM	AR(2)
BH	有	無
MAC	有	有

表格 3.2

3.2.4 評估方法

如[20]中所提到的，如何評估 GAN 所生成的資料的質量是很具挑戰性的。與傳統的機器學習的模型不一樣, 生成器和鑒別器的 loss 對於模型表現的評估的意義並不大。在圖像部分，通常判斷 GAN 所生成圖像的品質的評估多數依賴人類評分。在[20]中提出的 TSTR (Train on Synthetic, Test on Real)和 TRTS (Train on Real, Test on Synthetic) 。他們在於在生成（真實）的資料集上訓練一個傳統分類器，然後在真實（生成）的資料上測試結果，如果二者接近則認為 GAN 學到了。

我們將演算法做了一些調整，如演算法 3.1 所表示。我們現用訓練集訓練一個

doi:10.6342/NTU201801645

生成器，接著我們再訓練集上生成的資料中找到一個表現最好的策略，並且得到一個策略的分數指標（例如夏普值），最好我們將這個策略再應用到測試集上得到測試集上的分數指標，我們對比兩個分數指標對其打分。

```
1: train_data, test_data = split(data)
2: discriminator, generator = train_GAN(train_data)
4: synthetic_data = generator.generate_synthetic()
5: for config in configs:
5:  strategy = argmax [backtest (synthetic_data, config)]
6: synthetic_score = backtest (test_data, strategy)
7: test_score = backtest (test_data, strategy)
8: score = score(synthetic_score, test_score)
```

演算法 3.1 對於 RGAN 生成的資料之於回測表現的評估演算法。

由於有模型，我們首先找出策略，接著用我們的模型生成路徑，將這個策略應用在這些生成的路徑上，這樣我們就會得到一個回測的表現的分佈，我們成為目標回測結果分佈。

第二，我們再將上面所找到的策略應用到我們用 GAN 所學習的，這樣我們也可以得到一個分佈。

最後我們比較目標回測結果分佈和實驗回測結果分佈，包括對於分佈設一個信賴區間，比較 Monte Carlo 的結果和 GAN 的是否拒絕的結果，這變成了一個分類問題。

4 實驗結果與分析

實驗分為兩大部分。第一個部分是用 GAN with LSTM 來說明的確 GAN 可以合理的生成一些有 underlying 公式的時間序列。第二個部分是在第一個部分的基礎上，我們衍生到生成金融的時間序列(這裡採用的是蠟燭圖)並且用於 backtesting 的部分，並說明這的確可能為 backtesting 帶來好處。

doi:10.6342/NTU201801645

4.1 路徑學習

4.1.1 學習效果

圖 4.1 展示了 GBM 這個模型生成的樣本的樣子。圖 4.2 則顯示的是生成的這些樣本在每個時刻的分佈的頻率圖，就跟(3.3 所說一樣，每一期的分佈接近常態分佈。

從圖 4.2 可以看到, 似乎分佈是在移動的且分佈的標準差也隨著時間在增加，這一點通過圖 4.3 和圖 4.4 可以看得更加清楚一些。圖 4.3 和圖 4.4 展示了 GBM 這個時間序列的在各個時間的期望值和變異數，跟理論上一樣，與時間呈現了線性的關係。如前面所說，這一線性關係在我們訓練中很有幫助，我們訓練的目標之一就是觀察 RGAN 生成的時間序列在每一個時間點的期望值是否與時間呈線性關係。

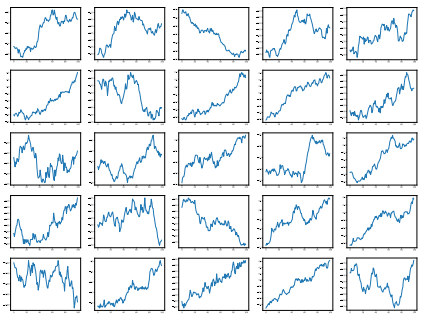


圖 4.1 模型生成的資料的樣子。

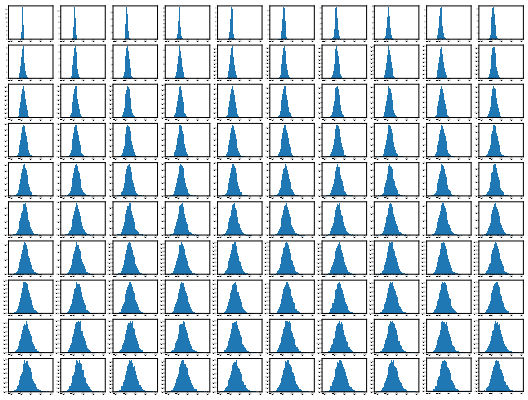


圖 4.2 每一個時間點（共有 100 個）模型生成的資料的頻率統計。

doi:10.6342/NTU201801645

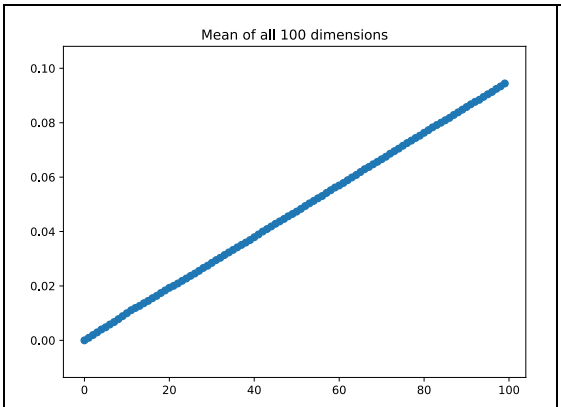


圖 4.3 橫軸是時間。顯示了模型生成的時間序列資料在每個時間點上的期望值。

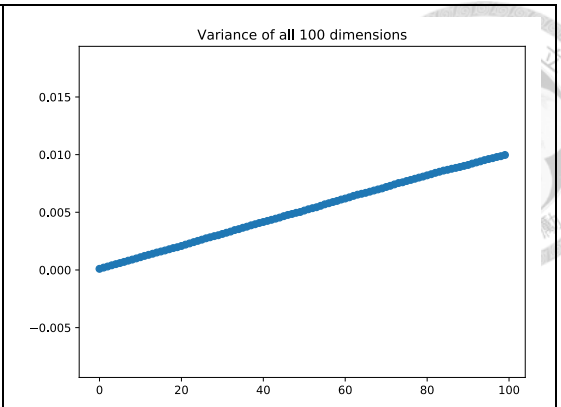


圖 4.4 橫軸是時間。顯示了模型生成的時間序列資料在每個時間點上的變異數。

圖 4.5 展示的是（訓練完的）RGAN 所生成的路徑的樣子。跟我們所說的一樣，用肉眼將它與圖 4.1 作比較是很困難的。參考圖 4.6 要容易一些，與圖 4.2 相比，圖 4.6 的分佈似乎的確是學到了圖 4.2 的某些性質：對稱性、隨著時間的增加期望值在變大、隨著時間的增加變異數在增加。

後兩者的話比較圖 4.7 和圖 4.8 以及圖 4.3 和圖 4.4 會更加有幫助。我們可以看到，圖 4.7 和圖 4.8 的確都表現出了和時間成線性慣性的這一性質，圖 4.7 和圖 4.8 中的紅線代表了理論的那一條線：參考(3.3。可以看到藍色的對於其擬合度相當的高，其中期望值的R^2為 0.994935，變異數的R^2為 0.9946425。

doi:10.6342/NTU201801645

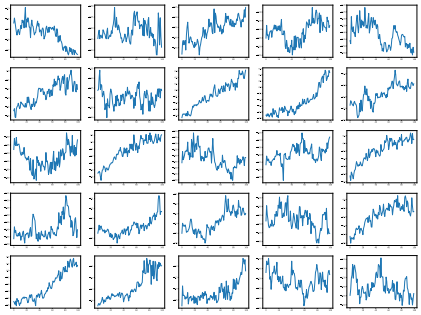


圖 4.5 RGAN 生成的資料的樣子。

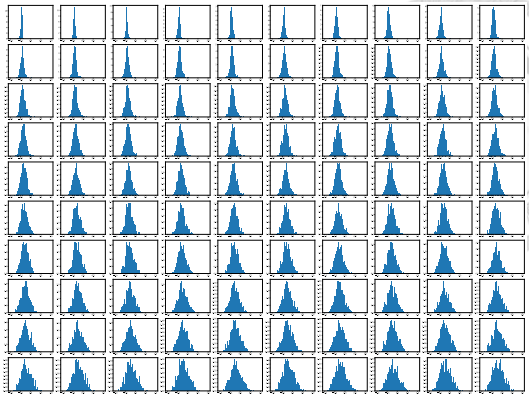


圖 4.6 每一個時間點（共有 100 個）RGAN 生成的資料的頻率統計。

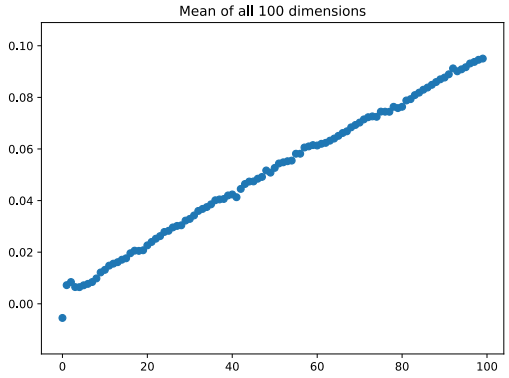


圖 4.7 橫軸是時間。顯示了模型生成的時間序列資料在每個時間點上的期望值。R^2 為 0.994935。

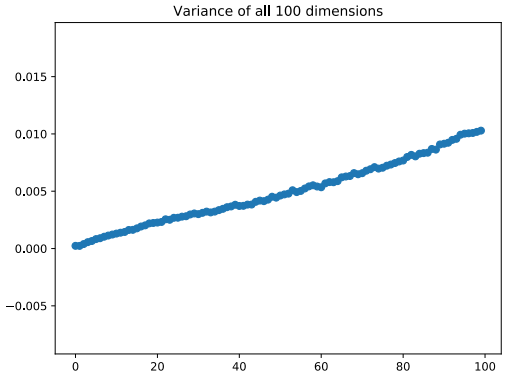


圖 4.8 橫軸是時間。顯示了模型生成的時間序列資料在每個時間點上的變異數。R^2 為 0.9946425。

4.1.2 資料量和參數量

在上一節中，我們現實了 RGAN 從某些解讀的確可以相當不錯的學到 GBM 的結果，但是如前面所說，RGAN 的訓練並不容易，比如訓練的結果容易停止在局部最優或者與超參數（hyperparameter）的選擇（學習率等）有比較強的關係[24]，在這一節我們更進一步實驗以下幾個參數對於 RGAN 的結果的影響，包括模型的角度和資料的角度。資料的角度在研究方法中已經討論。

首先，我們比較了隱藏層節點的數量以及資料量對於訓練的影響。如圖 4.9

doi:10.6342/NTU201801645

和圖 4.10 表示的是不同配置在時間上的表現，橫軸代表的是餵給模型的 batch 的數量，固定 batch 的大小是 50，而縱軸則是當時訓練的模型的期望值和理論的期望值直線的R^2，則對應的是變異數，R^2的意義可以參考圖 3.2 和圖 3.3。

從圖上可以看到，隱藏層節點的數量太少的模型的學習效果並不理想，隱藏層節點數量為 10 的模型的R^2的表現很差。這其實很合理，參數數量之巨大本來就是 deep learning 的一個特質，至於參數個數與模型複雜度和模型泛化的能力的部分的更深一步探討可以參考[25]。

而令人驚訝的是，RGAN 在這次試驗中，對於樣本的數量要求並不高！當樣本的數量僅僅 100 的時候依舊可以達到想對理想的訓練狀態。

然而這可能是過擬合所導致的，GAN 里的過擬合是指 GAN 完全的背下來了樣本而不是生成樣本的分佈。

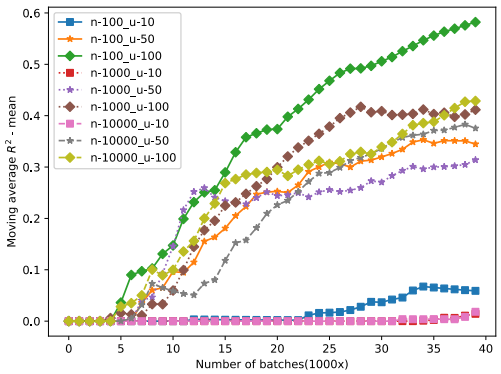


圖 4.9 資料量和參數與訓練時間的影響（期望值）。

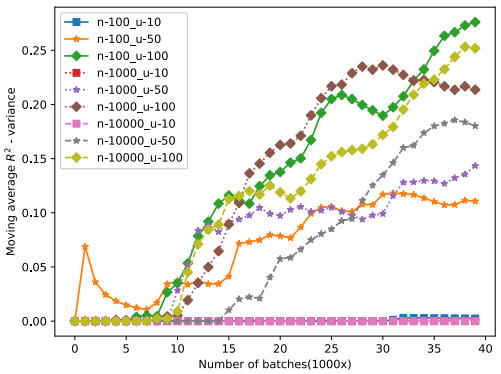


圖 4.10 資料量和參數與訓練時間的影響（變異數）。

4.1.3 Scaling

我們比較了資料量以及 scaling 的對於 RGAN 的學習的影響。從圖 4.11 圖 4.12 上可以看到而在資料量和 scaling 對於學習速度的影響。

doi:10.6342/NTU201801645

結果顯示，在相同的 scaling 之下，以 epoch 來算，資料量越大學習的越快（當然一個 epoch 對於資料量大的資料實際的樣本也更多）。

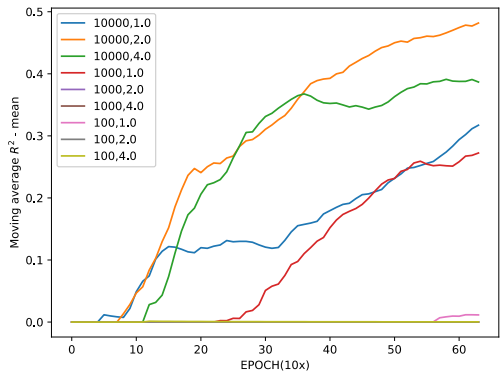


圖 4.11 sample 的數量以及 scaling 的訓練的結果（期望值）。

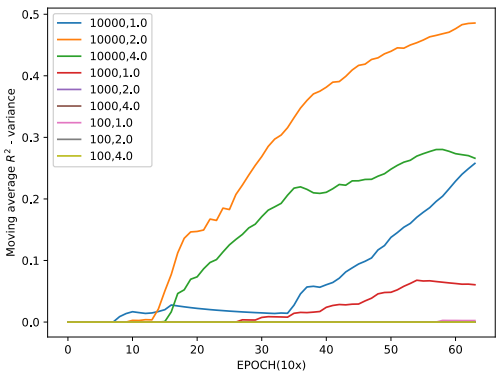


圖 4.12 sample 的數量以及 scaling 的訓練的結果（變異數）。

而在學習的效果部分，圖 4.13 和圖 4.14 表現了在 GAN 學習的樣本長度為 T=100 的時候，生成樣本為 200 的結果。可以看到，在平均值的部分，GAN 所生成的結果依舊穩定（繼續固定斜率線性增加），而在變異數的部分有一個明顯的轉折。

當我們把生成的樣本的長度增加到 300 的時候，即圖 4.15 和圖 4.16 的結果，我們可以看到此時的變異數依舊沒有繼續保持線性了，而平均的部分甚至出現了下降，這部分的原因很有可能是，當我們將用樣本的最大值作為總體的最大值（以此做(3.4）的時候所帶來的問題：由於我們的生成器的最後的激活函數是 tanh，則值域被固定在(−1,1)，且 1（−1）對應到樣本的最大值（最小值），當 GAN 試圖生成更大的值的時候，便會出現圖 4.15 最後的結果。

當我們使用了 scaling=4 之後，相當於某種程度上把 0.25（−0.25）對應到樣本的最大值（最小值），此時前面的問題會得到一定程度上的緩解。如圖 4.17 和圖 4.19 所示：在時間上，尤其是超過了 $T = 100$ 的部分，其期望值的線性依舊平滑。

doi:10.6342/NTU201801645

但是這樣會有一個明顯的代價：從圖 4.18 和圖 4.20 也能看出，樣本的變異數被壓縮了，這某種程度上改變了學習的樣本的一些統計性質。

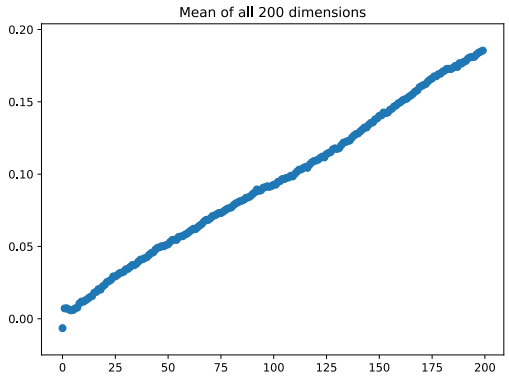


圖 4.13 scaling 為 1 的時候 RGAN 生成的長度為 200 的時間序列的在每個時刻的期望值的分佈。

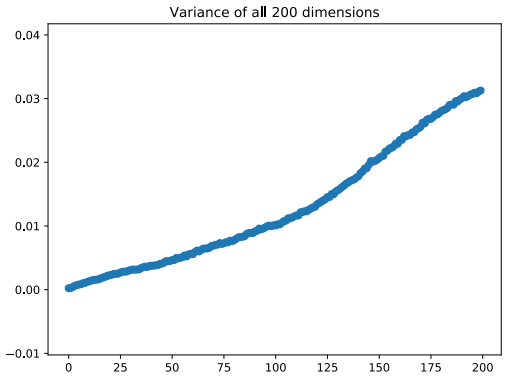


圖 4.14 scaling 為 1 的時候 RGAN 生成的長度為 200 的時間序列的在每個時刻的變異數的分佈。

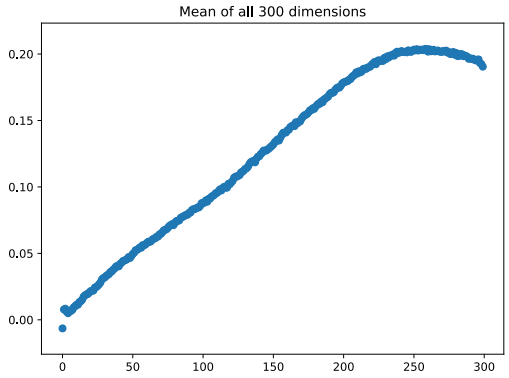


圖 4.15 scaling 為 1 的時候 RGAN 生成的長度為 300 的時間序列的在每個時刻的期望值的分佈。

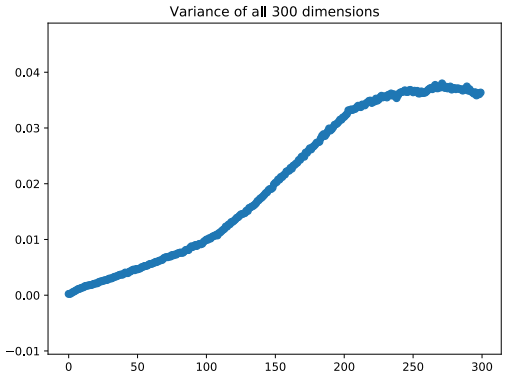


圖 4.16 scaling 為 1 的時候 RGAN 生成的長度為 300 的時間序列的在每個時刻的變異數的分佈。

doi:10.6342/NTU201801645

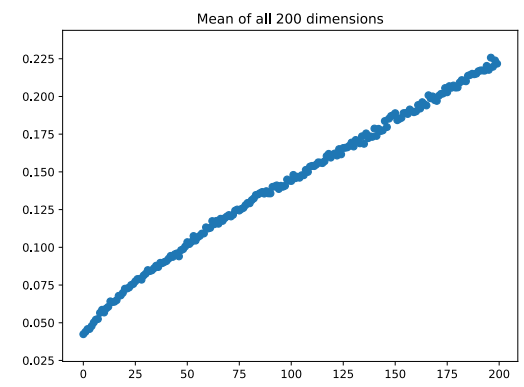


圖 4.17 scaling 為 4 的時候 RGAN 生成的長度為 200 的時間序列的在每個時刻的期望值的分佈。

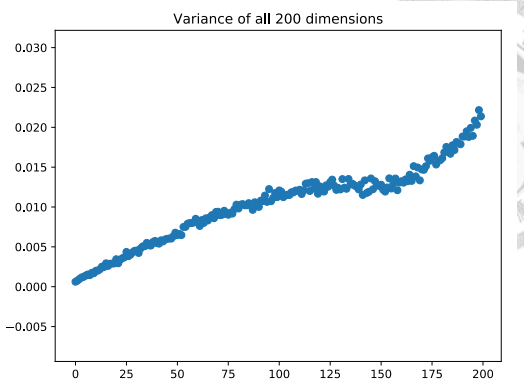


圖 4.18 scaling 為 4 的時候 RGAN 生成的長度為 200 的時間序列的在每個時刻的變異數的分佈。

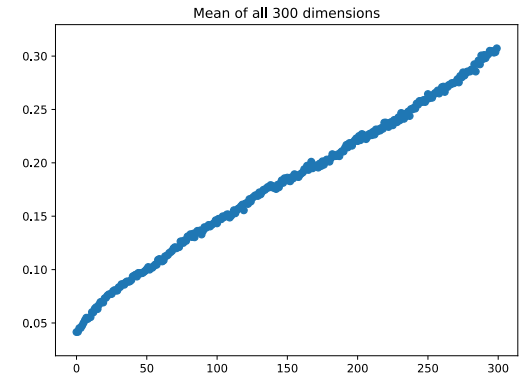


圖 4.19 scaling 為 4 的時候 RGAN 生成的長度為 300 的時間序列的在每個時刻的期望值的分佈。

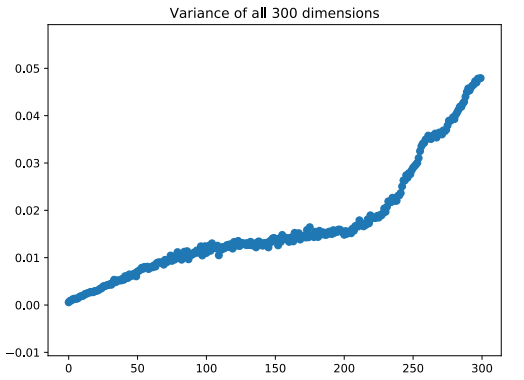


圖 4.20 scaling 為 4 的時候 RGAN 生成的長度為 300 的時間序列的在每個時刻的變異數的分佈。

綜合上述的實驗結果，我們有以下結論：（一） GAN 可以學習到常用的股價模型的一些性質。（二） 樣本數在這個試驗中的重要性並不是那麼大。（三） 用 scaling 可以緩解一些在生成階段值域的相關問題，但是代價是犧牲了變異數的統計性質。

另外，值得指出的我們並沒有說 GAN 完全學習到了 GBM 這個模型：因為顯然在各個方面，離完美的擬合理論值還有不小的距離（包括樣本的均值、變異數、分佈的形狀等等），但是的確在各個方面和理論值已經接近了。那麼，一個很重要的問題是，對於我們希望應用的的回測而言，這樣的學習效果到底足夠不足夠呢？在下面我們進行了討論。

doi:10.6342/NTU201801645

4.2 策略在路徑上

4.2.1 表現

如前面所說, GAN 可以近似常用的股票過程, 但是這樣的近似足夠我們回測嗎? 我們在這邊做了實驗。

圖 4.21 和圖 4.22 表現的是 BH-GBM 的實驗結果。我們策略的選取是挑樣本內的最佳。可以看到，紅色和藍色的 PDF 相當的接近：這意味著，在 BH-GBM 組合之下，GAN 所學習的模型在回測上的參考價值和真正的模型的相差不多。

圖 4.23 和圖 4.24 表現的是 MAC-GBM 的實驗結果。此時, 紅色和藍色的 P.D.F 並不那麼接近，且藍色的均值顯然會比紅色的更大，不過從 CDF 來看，兩者的差別並不大，所以當我們選一個合理的信賴區間來拒絕偽陽性。

圖 4.25 及圖 4.26、圖 4.27 及圖 4.28 的表現出色, 結論類似圖 4.21 和圖 4.22。

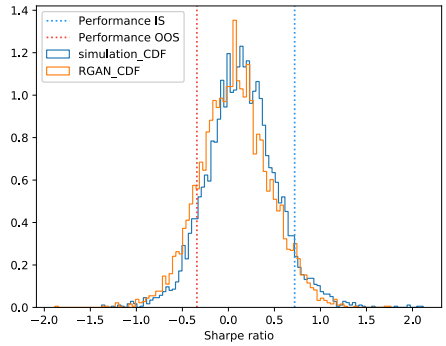


圖 4.21 策略 BH 在學習了 GBM 的 GAN 所生成的樣本路徑上以及 GBM 所生成的樣本路徑上的 SR 的 P.D.F。

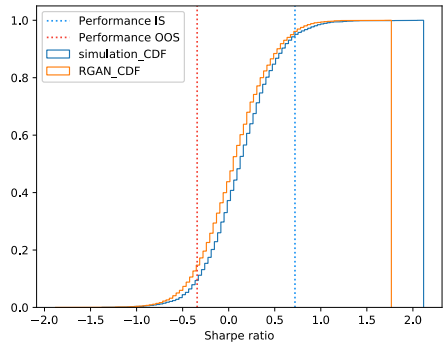


圖 4.22 策略 BH 在學習了 GBM 的 GAN 所生成的樣本路徑上以及 GBM 所生成的樣本路徑上的 SR 的 C.D.F。

doi:10.6342/NTU201801645

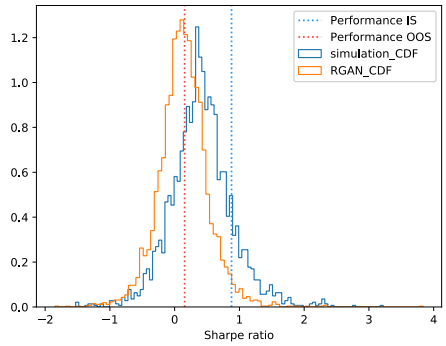


圖 4.23 策略 MAC 在學習了 GBM 的 GAN 所生成的樣本路徑上以及 GBM 所生成的樣本路徑上的 SR 的 P.D.F。

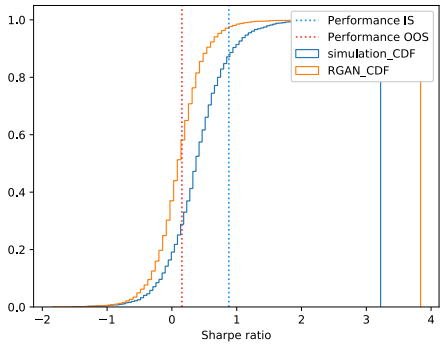


圖 4.24 策略 MAC 在學習了 GBM 的 GAN 所生成的樣本路徑上以及 GBM 所生成的樣本路徑上的 SR 的 C.D.F。

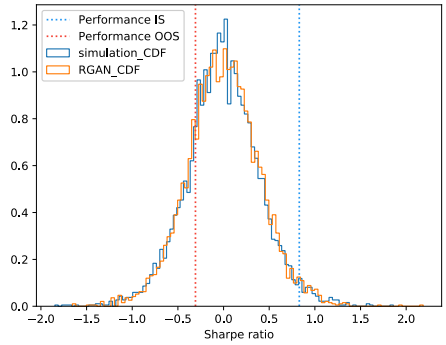


圖 4.25 策略 BH 在學習了 AR(2)的 GAN 所生成的樣本路徑上以及 AR(2)所生成的樣本路徑上的 SR 的 P.D.F。

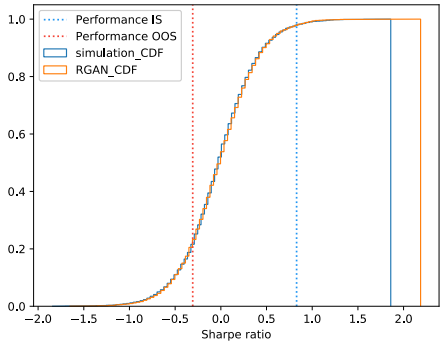


圖 4.26 策略 BH 在學習了 AR(2)的 GAN 所生成的樣本路徑上以及 AR(2)所生成的樣本路徑上的 SR 的 C.D.F。

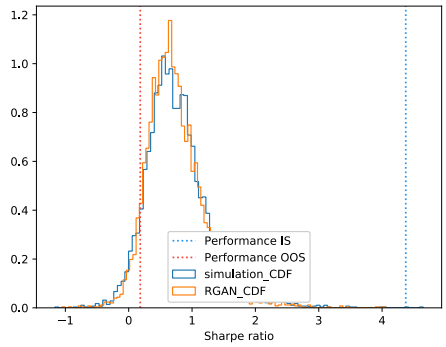


圖 4.27 策略 MAC 在學習了 AR(2)的 GAN 所生成的樣本路徑上以及 AR(2)所生成的樣本路徑上的 SR 的 P.D.F。

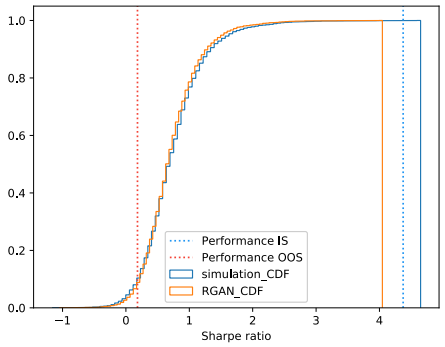


圖 4.28 策略 MAC 在學習了 AR(2)的 GAN 所生成的樣本路徑上以及 AR(2)所生成的樣本路徑上的 SR 的 C.D.F。

4.2.2 混淆矩陣

doi:10.6342/NTU201801645

此外，我們也做了測試，對於每一種（過程，策略）的組合均抽樣 100 個，並且與 Monte Carlo 的結果作比較，此外我們定義策略何為有效：夏普值 >0 的頻率超過 0.75（同時對 Monte Carlo 和 RGAN 適用。我们也可以調整這個值來控制我們的 recall 和 precision）。

表格 4.1 表現的是策略 BH 和過程 GBM 的結果。可以看到總體上的準確率是不錯的，只有 RGAN 認為無效而 Monte Carlo 認為有效的部分也許可以通過調整 RGAN 的閾值來調整。

表格 4.2 表現的是策略 MAC 和過程 GBM 的結果並不是很理想，這也許和圖 4.23 的表現有關。

表格 4.3 表現的是策略 BH 和過程 AR(2)的結果，可以看到正確率是 100%，在前面有提過的確 BH 和 AR(2)的期望值是不可能為正的。

表格 4.4 表現的是策略 BH 和過程 AR(2)的結果，可以看到正確率是 100%，在前面有提過的確 BH 和 AR(2)的期望值是不可能為正的。

通過以上的結果，我們發現，在這幾個股價模型和策略之下，我們的方法與期望的正確答案的結果相當的接近，即在回測的部分，GAN 所學習到的分佈可以替代真正的分佈。

GBM-BH		Monte Carlo	
		有效	無效
RGAN	有效	72	0
	無效	8	20

表格 4.1 GBM 和 BH 的混淆矩陣。

doi:10.6342/NTU201801645

GBM-MAC		Monte Carlo	
		有效	無效
RGAN	有效	32	9
	無效	18	41

表格 4.2 GBM 和 MAC 的混淆矩陣。

AR(2)-BH		Monte Carlo	
		有效	無效
RGAN	有效	0	0
	無效	0	100

表格 4.3 AR(2)和 BH 的混淆矩陣。

AR(2)-MAC		Monte Carlo	
		有效	無效
RGAN	有效	39	1
	無效	1	59

表格 4.4 AR(2)和 MAC 的混淆矩陣。

doi:10.6342/NTU201801645

5 總結

本論文從量化交易中極易碰到的回測過擬合問題出發，希望利用 GAN 來學習到股價的分佈來緩解這個問題。從機器學習的角度，我們研究了 GAN 學習的性質，包括生成器、鑒別器的架構以及參數數量，以及在緩解過擬合這個任務下 GAN 對於樣本數量的要求。從應用的角度，我們研究 GAN 所學習的分佈是否有助於減緩回測過擬合。

我們通過實驗得到以下結論。在我們的假設模型以及選取策略之下：（一） GAN 可以學習到常用的股價模型的一些性質。（二） 樣本數在這個試驗中的重要性並不是那麼大。（三） 用 scaling 可以緩解一些在生成階段其值域的相關問題，但是代價是犧牲了變異數的部分統計性質。（四） GAN 所學到的分佈已經足夠在我們選擇的一些策略之下某種程度上避免了過擬合這個問題，可以幫助我們更好的篩選策略。

doi:10.6342/NTU201801645

參考文獻

[1] P. Carr and M. Lopez de Prado, “Determining Optimal Trading Rules without Backtesting,” 2014.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, and S. Ozair, “Generative Adversarial Networks,” in *NIPS*, 2014, pp. 2672–2680.

[3] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, “Generating images with recurrent adversarial networks,” 2016.

[4] D. N. Politis and J. P. Romano, “The Stationary Bootstrap,” in *Journal of the American Statistical Association*, vol. 89, no. 428, 1994, pp. 1303–1313.

[5] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, “A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction,” 2017.

[6] C. W. J. Granger, “Forecasting stock market prices: Lessons for forecasters,” *Int. J. Forecast.*, vol. 8, no. 1, pp. 3–13, 1992.

[7] G. Gidófalvi, “Using News Articles to Predict Stock Price Movements,” *Dep. Comput. Sci. Eng. Univ. Calif. San Diego*, p. 9, 2001.

[8] M. O. Afolabi and O. Olude, “Predicting Stock Prices Using a Hybrid Kohonen Self Organizing Map (SOM),” *Proc. 40th Hawaii Int. Conf. Syst. Sci.*, pp. 1–8, 2007.

[9] T. S. B. Fletcher, “Machine Learning for Financial Market Prediction,” p. 207, 2012.

[10] D. H. Bailey, J. M. Borwein, M. López de Prado, and Q. J. Zhu, “Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance,” *Not. AMS*, vol. 61, no. 5, pp. 458–471, 2014.

[11] D. H. Bailey and M. Lopez de Prado, “The Deflated Sharpe Ratio: Correcting for

doi:10.6342/NTU201801645

Selection Bias, Backtest Overfitting and Non-Normality," *J. Portf. Manag.*, vol. 40, no. 5, pp. 94–107, 2014.

[12] M. Lopez de Prado, D. H. Bailey, J. M. Borwein, M. López de Prado, and Q. J. Zhu, "The Probability of Backtest Overfitting," *J. Comput. Financ.*, p. Forthcoming, 2013.

[13] C. R. Harvey and Y. Liu, "Backtesting," *SSRN Electron. J.*, 2015.

[14] C. M. Investopedia, "Moving Averages: Strategies." [Online]. Available: https://www.investopedia.com/university/movingaverage/movingaverages4.asp.

[15] Investopedia, "Buy And Hold." [Online]. Available: https://www.investopedia.com/terms/b/buyandhold.asp.

[16] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[17] C. Olah, "Calculus on Computational Graphs: Backpropagation," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Backprop/.

[18] S. Hochreiter and J. Urgen Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[20] S. L. Hyland, C. Esteban, and F. Ratsch, "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs." 2017.

[21] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization," 2016.

[22] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *ICLR*, 2015, pp. 1–10.

doi:10.6342/NTU201801645

[23] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, “Optimizing the Latent Space of Generative Networks,” 2017.

[24] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[25] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, “Generalization in Deep Learning,” 2017.

doi:10.6342/NTU201801645