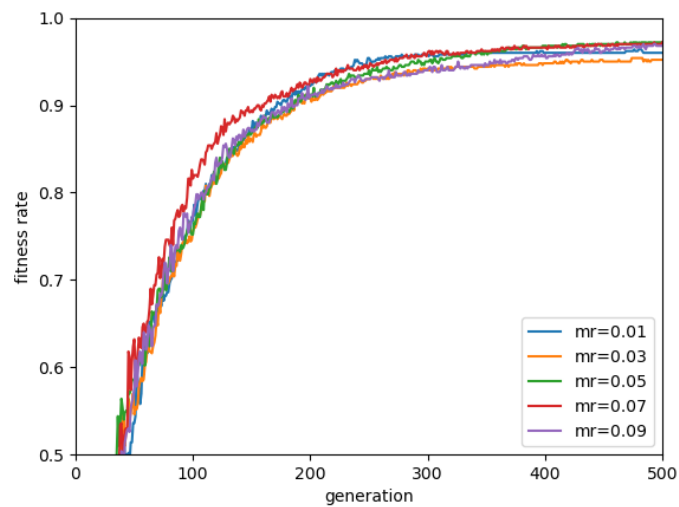


自主學習

機器學習與人工智慧入門
遺傳演算法專案
延伸應用：彈簧擺軌跡亂度預測



作者：

范佑誠

~ 目錄 ~

1 摘要.....	- 1 -
2 動機.....	- 1 -
3 基礎人工智慧知識的學習	- 1 -
3.1 Linear Regression、CNN.....	- 1 -
3.2 遺傳演算法	- 2 -
3.2.1 hyperparameters.....	- 3 -
4 應用—結合物理探究	- 4 -
4.1 前言—物理探究內容	- 4 -
4.2 解決方式—類神經網路	- 4 -
4.3 結果.....	- 4 -
4.4 訓練解果	- 4 -
4.5 預計改善方式.....	- 5 -
5 心得.....	- 5 -
5.1 不同任務真的需要不同 hyper parameter	- 5 -
5.2 加速之路—從 python 到 C++.....	- 5 -
5.3 物理探究—失敗學更多	- 5 -
6 參考資料	- 6 -
7 連結.....	- 6 -

1 摘要

本次自主學習之主要目標在於遺傳演算法的學習、基本模型之建構，目標是想看到其跳脫 local optimize 的能力，及先前物理探究之延伸應用。

遺傳演算法以建立 python, C++ 的模型為主，並以自定義之資料進行訓練。結合物理探究則是以類神經網路輔助預測軌跡亂度（直接模擬並進行歸納後發現亂度有規律，大致跟總能量有關，因此預期亂度是可預測的）。雖然最終結果不完全成功，但是測資訓練正常，但也已經找到資料集的問題所在，並提出修正方式。

2 動機

高一加入新創人工智慧研究社，開始接觸人工智慧領域。在時代潮流及兩位社長的先前作品的影響下，逐漸有興趣。也因此在一聽完理論課程之後高二開始簡單的實作並自己研究遺傳演算法，高三則繼續延伸。

3 基礎人工智慧知識的學習

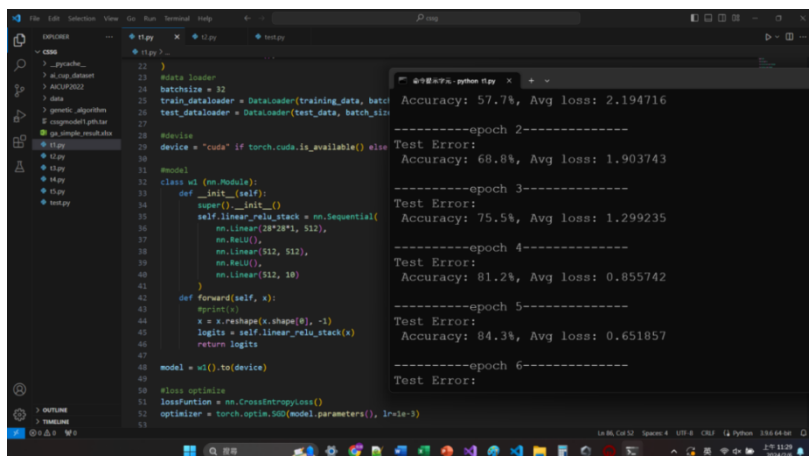
3.1 Linear Regression、CNN

學習資料來源：

主要是參考 PyTorch 官網的 Tutorial。

學習內容：

大致上是了解 PyTorch 如何操作，module 怎麼寫，各個 function 的功能（尤其各種不同的 loss function）。加上與經驗豐富的老師討論，所以在這一節沒什麼太大的問題。



```
22 }
23 #Data loader
24 batchsize = 32
25 train_dataloader = DataLoader(training_data, batchsize=batchsize)
26 test_dataloader = DataLoader(test_data, batchsize=batchsize)
27
28 #Device
29 device = "cuda" if torch.cuda.is_available() else "cpu"
30
31 #Model
32 class Net(nn.Module):
33     def __init__(self):
34         super().__init__()
35         self.linear_relu_stack = nn.Sequential(
36             nn.Linear(28*28, 512),
37             nn.ReLU(),
38             nn.Linear(512, 512),
39             nn.ReLU(),
40             nn.Linear(512, 10)
41         )
42     def forward(self, x):
43         print(x)
44         x = x.reshape(x.shape[0], -1)
45         logits = self.linear_relu_stack(x)
46         return logits
47
48 model = Net().to(device)
49
50 #optimizer
51 lossfunction = nn.CrossEntropyLoss()
52 optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
53
54 # Training loop
55 for epoch in range(1, 7):
56     # Training
57     train_loss = 0.0
58     train_acc = 0.0
59     for batch_idx, (data, target) in enumerate(train_dataloader):
60         data, target = data.to(device), target.to(device)
61         optimizer.zero_grad()
62         output = model(data)
63         loss = lossfunction(output, target)
64         loss.backward()
65         optimizer.step()
66         train_loss += loss.item()
67         train_acc += output.argmax(-1).eq(target).sum().item()
68     train_loss /= len(train_dataloader)
69     train_acc /= len(train_dataloader)
70
71     # Testing
72     test_loss = 0.0
73     test_acc = 0.0
74     with torch.no_grad():
75         for batch_idx, (data, target) in enumerate(test_dataloader):
76             data, target = data.to(device), target.to(device)
77             output = model(data)
78             loss = lossfunction(output, target)
79             test_loss += loss.item()
80             test_acc += output.argmax(-1).eq(target).sum().item()
81     test_loss /= len(test_dataloader)
82     test_acc /= len(test_dataloader)
83
84     print(f'Epoch: {epoch}, Train Loss: {train_loss}, Train Acc: {train_acc}, Test Loss: {test_loss}, Test Acc: {test_acc}')
```

Figure 3.1-1

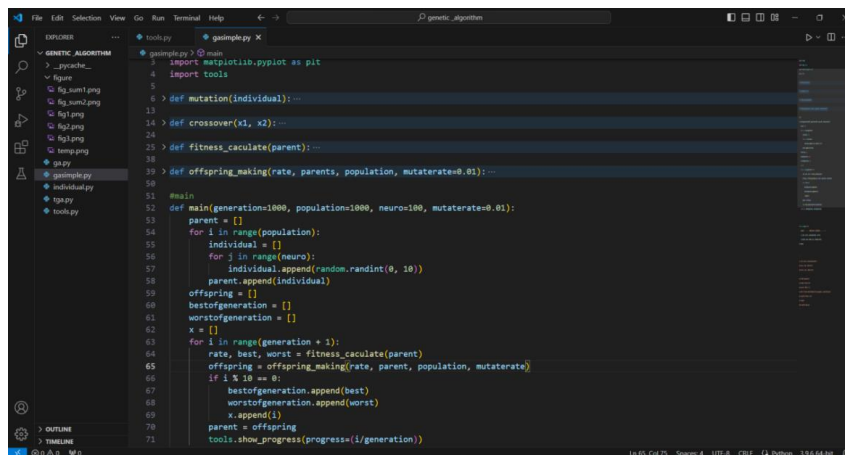
3.2 遺傳演算法

學習資料來源：

MIT OpenCourseWare

學習方式：

學習遺傳演算法的精神，自己重頭寫出遺傳演算法的架構，並透過自定義的正確率計算方式作為判斷依據。



```
1 import matplotlib.pyplot as plt
2 import tools
3
4 def mutation(individual):...
5
6 def crossover(x1, x2):...
7
8 def fitness_calculate(parent):...
9
10 def offspring_making(rate, parents, population, mutaterate=0.01):...
11
12 #main
13 def main(generation=1000, population=1000, neuro=100, mutaterate=0.01):
14     parent = []
15     for i in range(population):
16         individual = []
17         for j in range(neuro):
18             individual.append(random.randint(0, 10))
19         parent.append(individual)
20     offspring = []
21     bestofgeneration = []
22     worstofgeneration = []
23     x = []
24     for i in range(generation + 1):
25         rate, best, worst = fitness_calculate(parent)
26         offspring = offspring_making(rate, parent, population, mutaterate)
27         if i % 10 == 0:
28             bestofgeneration.append(best)
29             worstofgeneration.append(worst)
30             x.append(i)
31         parent = offspring
32     tools.show_progress(progress=1/generation)
```

Figure 3.2-1

學習的過程去調過各種參數，包括 mutation rate、population 等，討論其對 learning speed 及 ultimate accuracy (fitness rate)的影響，也使用不同的 fitness function 觀察其最佳的 mutation rate (mr)等參數，以及如何從 fitness rate 定義 crossover rate（後續稱為 f2c function）。

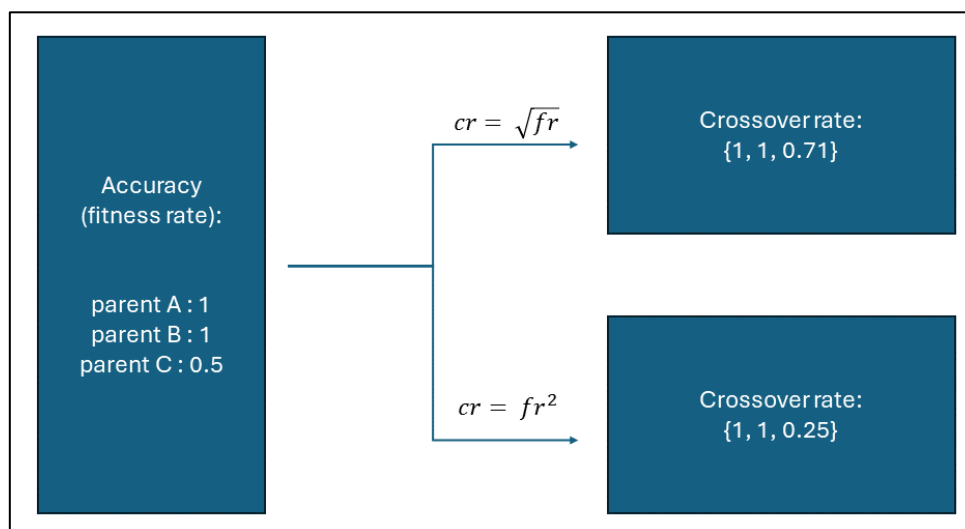


Figure 3.2-2

3.2.1 hyperparameters

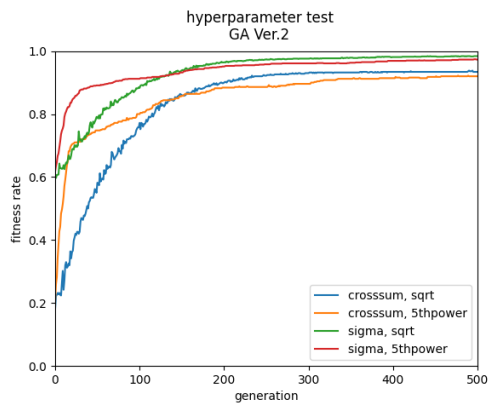


Figure 3.2.1-1

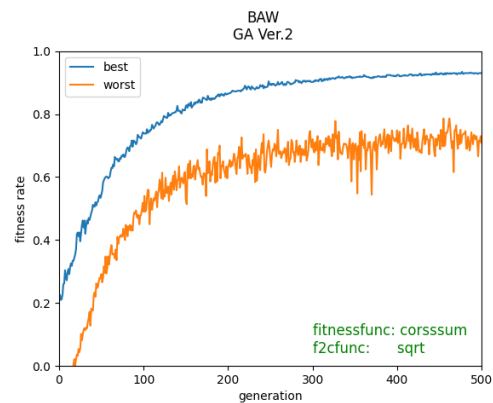


Figure 3.2.1-2

Figure 3.2.1-1 為不同 fitness function 與 f2c function 配對產生的結果，可以看到使用 5thpower 的模型正確率可以上升較快，但提前收斂，相對的，使用 sqrt 則可以表現出較佳的最終正確率。

Figure 3.2.1-2 為同一次訓練中每個 generation 中最好與最差的模型。可以看到 best 幾乎是上升後收斂，而 worst 雖會上升但仍處於隨機的狀態。

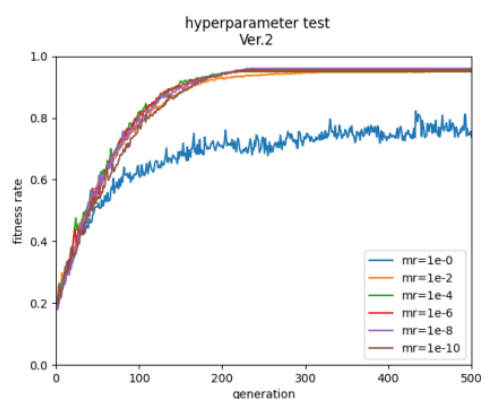
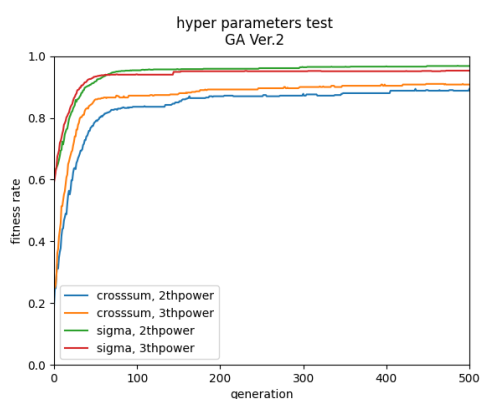
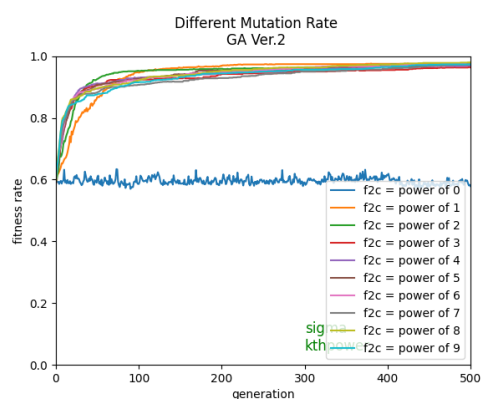
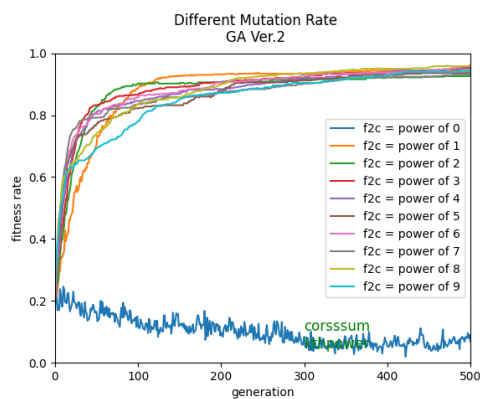


Figure 3.2.1-3 ~ Figure 3.2.1-6

上面四張則是實驗過程中一些較具代表性的實驗記錄

4 應用—結合物理探究

4.1 前言—物理探究內容

目標：觀察彈簧擺之各種釋放參數對其軌跡亂度的影響

方式：自行架設器材，用 Tracker 追蹤並劃出軌跡

問題：Tracker 取樣頻率過低，導致快速傅立葉轉換（FFT）之有效頻域太小；實驗太花時間；彈簧只可伸長、不可壓縮

解決方式：自己用 python 寫模擬器（Elastic Pendulum Simulator (EPS)）

問題：由於軌跡是用遞迴的方式求出，若取樣頻率高則花時間，取樣頻率低則誤差大，因此不利於做更精細完整的分析。

見 參考資料[1]

4.2 解決方式—類神經網路

想要用快速、計算量少的方式預測亂度，回歸可能會是一個可行的方法。

基於之前學過的人工智慧，我發現可以用最基礎的線性回歸解決此問題，而相較於 excel 能支援的 one-layer 線性回歸，neuron network 所具有的 multilayer 可能會產生更好的解果。

4.3 結果

(1). 在先前的模擬器加一個紀錄用的 function: `writedata2()`

(2). 建立 custom dataset

(3). 建立類神經網路並開始訓練

4.4 訓練解果

很遺憾的，由於資料是沿用先前 EPS 中的 `chaosidx`，並沒有將數值控制在 0 跟 1 間，導致沒辦法用現有類神經網路做出有效的訓練。

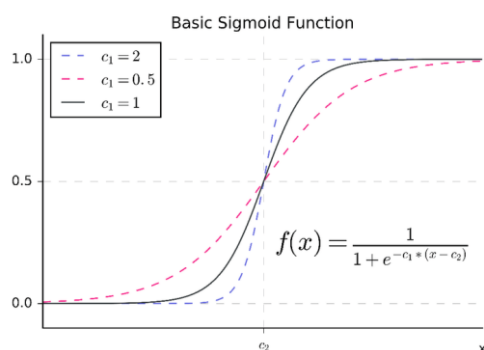


Figure 4.5-1

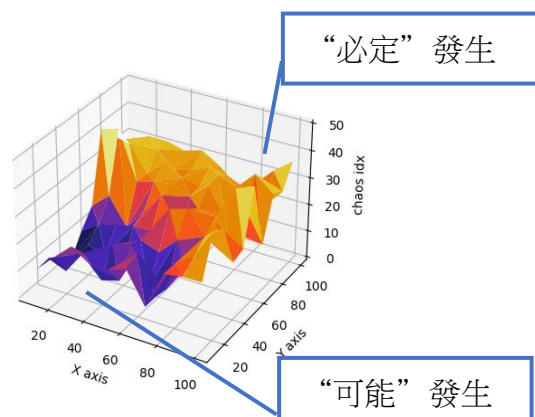


Figure 4.5-2

4.5 預計改善方式

如前結所提，改善目標在於把 chaosidx 轉換成一個介於 0 和 1 的數。因此我預計引入 sigmoid function 的概念，這將可以達到下列兩個好處：

(1). 所有數直可控制在 0 到 1 之間

(2). 有利討論原物理探究之命題：

增強討論“可能”發生混沌的地方，省去討論“必定”發生的範圍，如 *Figure 4.5-2* 所示。

見 參考資料[1]

5 心得

5.1 不同任務真的需要不同 hyper parameter

一開始是在李弘毅的生成式 AI 2024 線上課中聽到，直覺也不是沒有道理，但我是在這次的實作後意外體會到。在整個實驗最一開始，不同的 hyper parameter 產生了一定的影響，這些都反映在模型正確率的提升速度及最終正確率，只是沒有預期那麼明顯，個人覺得可能是任務差異不夠大。如 *Figure 3.2.1-4* 所式。

5.2 加速之路－從 python 到 C++

當時因為是在學期間，還有課業要顧，所以選擇編輯速度較快的 python，然而這卻導致後期的一大麻煩：超長的執行時間。由於計算量大，再加上要討論就需要有資料比對，因此每次改完參數後都需要花費很長的時間在執行上。因此決定改用 C++。

5.3 物理探究－失敗學更多

雖然這次實驗並非完全失敗，至少在測資的訓練下模型的運作皆如預期，然而當行到真正的資料時卻直接秀出我從未遇過的異常代碼，後來才意識到是資料標記的有問題。

由於想直接以數值作為判斷，就不得不想出一個可以重新標記的方式，也就是前面提到加入 sigmoid function 的想法。而加入這個也會遇到一個問題： C_1 要等於多少？這會關係到原本的值域到底多大，而我們也知道 FFT 的強度數值是會跟隨時間進行疊加的，所以後來也想到直接把時間這個因素除掉，也就是把 output 變成單位時間的強度。這除了作出符合模型的標記方式外，也統一資料的格式，人為判斷上也會較方便。

6 參考資料

- [1] 范佑誠 B-物理探究與實作書面報告
- [2] 生程式 AI 導論 2024 李弘毅
<https://speech.ee.ntu.edu.tw/~hylee/genai/2024-spring.php>
- [3] MIT Open Course Ware 2010 Fall Genetic Algorithm
<https://www.youtube.com/watch?v=kHyNqSnzP8Y>
- [4] PyTorch official tutorial
<https://pytorch.org/tutorials/beginner/basics/intro.html>

7 連結

- (1). 為避免系統傳送中資料有誤，或其他特殊狀況，以下為 2 個資料備份連結
[https://drive.google.com/drive/folders/13tSIPH37VxL2sehbljRZnTZyTsxV623z?usp=drive link](https://drive.google.com/drive/folders/13tSIPH37VxL2sehbljRZnTZyTsxV623z?usp=drive_link)
<https://github.com/bilililililily/bilililililily.git>
(folder: 學習歷程備份資料夾)
- (2). 程式碼
<https://github.com/bilililililily/bilililililily.git>
(folder: ElasticPendulumSimulator, GeneticAlgorithm)