

# 人工智慧導論 HW-1

Q56104076 陳哲緯

## ● brute force

實作方式:

在 x 和 y 的數值範圍內枚舉落點，並找出最好的點。

枚舉落點方式：將x和y分別等距等切成n段 並讓他們互相配對產生  $n^*n$  個點。

假設  $x \text{ range} = 1 \sim 5$  ,  $y \text{ range} = 1 \sim 5$  ,  $n = 5$  就會有 25 個起始值

pairs = [ (1,1) , (1,2) , (1,3) , (1,4) , (1,5) , (2,1) , (2,2) , (2,3) (2,4) , (2,5) .... ]

優點: 在運算資源無限和精度無限小的情況能找出全局最佳解。

缺點: 一切都依賴選取的點，無法再更逼近local or global minimum。

n	run times	X	Y	minimum Z
10	0.0002S	-17.0	10.0	-31.529
100	0.0161S	-14.79	10.0	-36.943

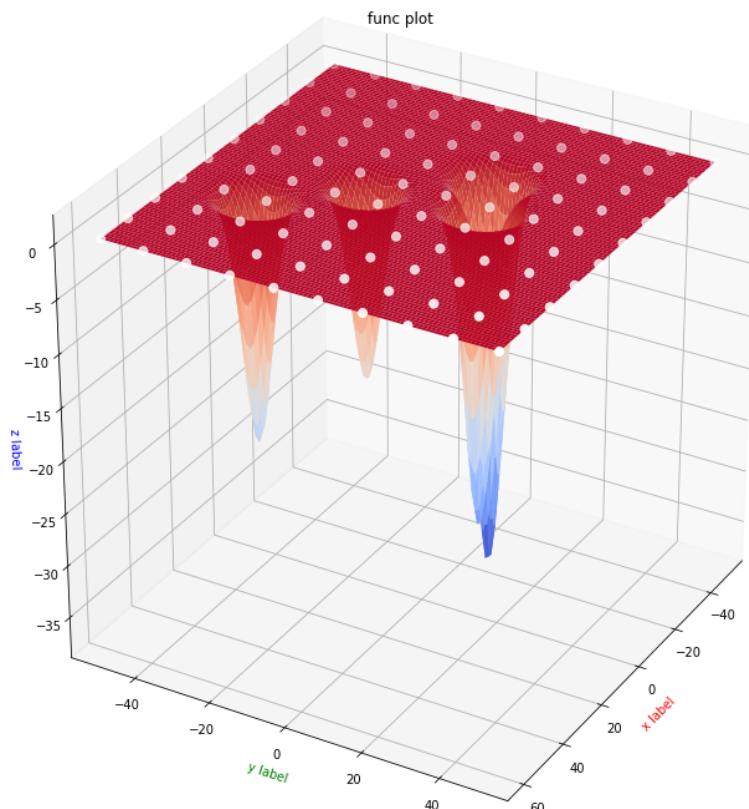


示意圖: 白點為x,y的枚舉方式

## ● beam search

在每個step:

1. 先選出K個起始值
2. 讓每個起始值產生K個successors
3. 篩選出最好的N個successors
4. 從這N個successors 附近產生 successors，直到有  $N \times K$  個 newsuccessors，重複3-4循環直到找出符合條件的最小值，

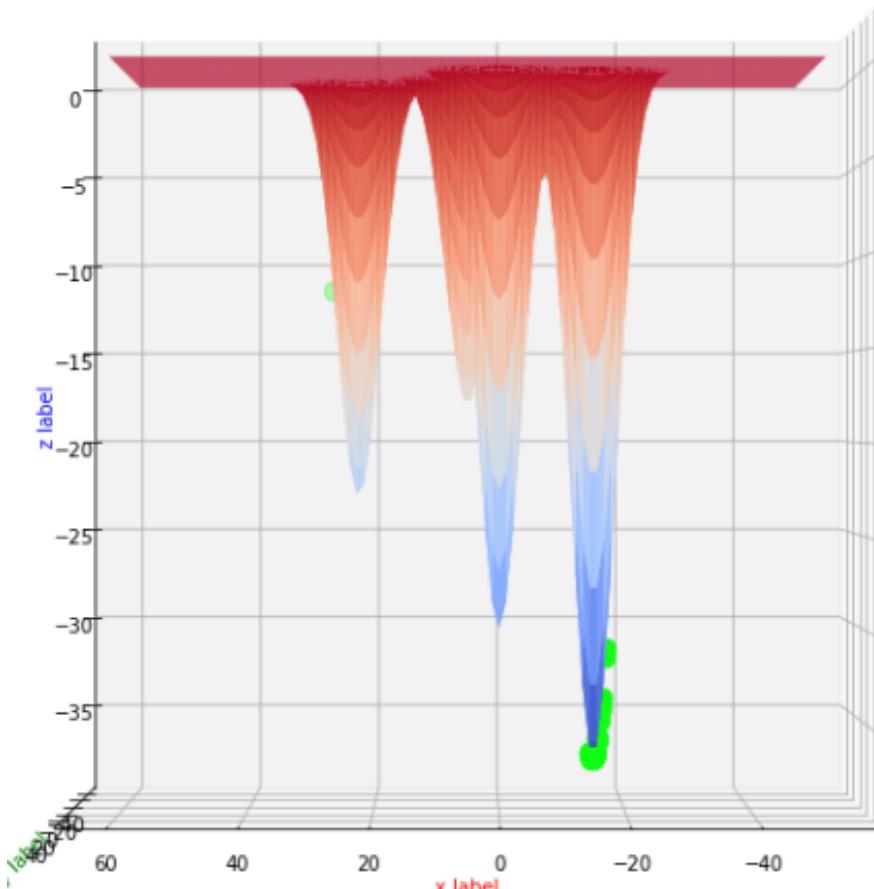
提早結束條件為:

1. 如果上一個 step 中  $N \times K$  successors 中最好的點，跟歷史以來找過最好的點差距不到 0.0001，那麼就提早結束搜索。
2. 超過特定次數不能更新最小值，可以判斷為掉入 local minimum 並退出。

產生successors的方式:

從當下的點往四個方向隨機移動隨機 0~1 內的長度

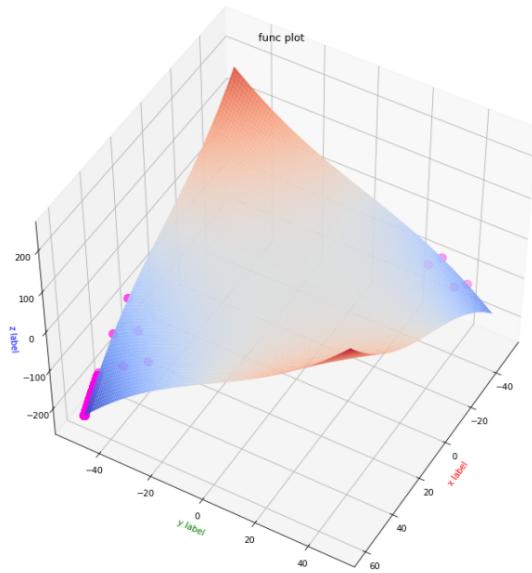
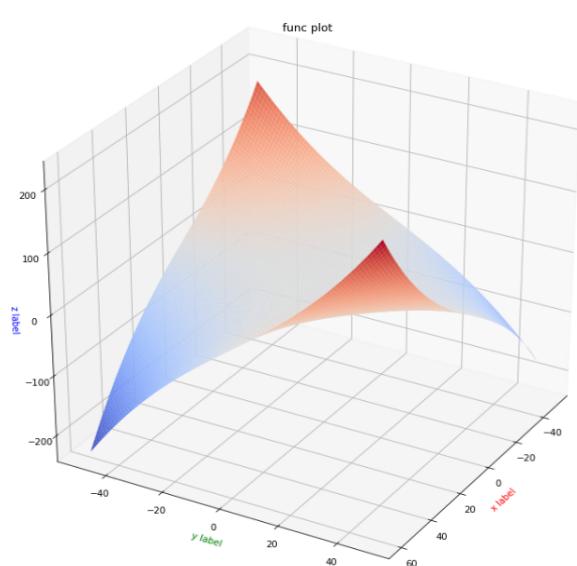
K	N	Steps	run times	X	Y	minimum Z
100	10	100	0.013S	-14.98	10.03	-37.0
1000	100	100	1.07S	-14.99	9.99	-37.001



從圖中可以看到，beam search 會逐漸逼近 minimum(綠點為每次選出的 N 個 best successors)，就算不小心掉入左邊的 local minimum 也是有機會拯救回來

## 額外測試:

```
func(x,y) = 0.001 * x * y * math.sqrt(x**2 + y**2)  
x range = [-50, 60]  
y range = [-50, 50]
```



左圖: function圖形, 右圖: beam search的搜索方式

method	time	X	Y	minimun
brute force	0.0036	58.9	-50.0	-227.53
beam search	0.0033	59.9	-49.99	-234.30

結論: 在時間差不多的情況, beam search可以更快的找到minimum, brute force則需要去枚舉其他的點, 但beam search無法避免陷入local minimum