

# 商業智慧與營運策略

## Business Intelligence and Operations Strategy

# 特徵選取、退火演算法 和基因演算法

**授課老師：林冠成 教授**

Kuan-Cheng Lin

5501.mis.nchu.edu.tw

# 課程大綱

- **多特徵選取 (feature selection)**
  - 特徵選取的概念與重要性
  - 特徵選取的方法 ( 包裝法-Wrapper、過濾法-Filter )
- **退火演算法 (Simulated Annealing, SA)**
  - 退火演算法的基本原理與步驟
  - 退火演算法在特徵選取中的應用
- **基因演算法 (Genetic Algorithms, GA)**
  - 基因演算法的基本原理與步驟



# 一、特徵選取 (feature selection)

二、退火演算法 (Simulated Annealing)

三、基因演算法 (Genetic Algorithms)

# What is Feature Selection for classification?

什麼是分類的特徵選取？

- Given: a set of predictors (“features”)  $V$  and a target variable  $T$

已知：一組預測變數（「特徵」） $V$  和一個目標變數  $T$

- Find: minimum set  $F$  that achieves maximum classification performance of  $T$  (for a given set of classifiers and classification performance metrics)

目標要找：實現  $T$  最大分類性能的最小集合  $F$   
（對於給定的一組分類器和分類性能指標）

# Why feature selection is important?

為何特徵選取很重要？

- May Improve performance of classification algorithm  
可以提高分類演算法的效能
- Classification algorithm may not scale up to the size of the full feature set either in sample or time  
分類演算法可能無法在樣本或時間上擴展到完整特徵集的大小
- Allows us to better understand the domain  
讓我們更好的理解這個領域
- Cheaper to collect a reduced set of predictors  
收集減少的預測變數集更便宜
- Safer to collect a reduced set of predictors  
收集減少的預測變數集更安全

## Filters vs. Wrappers: **Wrappers**

- Say we have predictors A, B, C and classifier  $M$ .  
We want to predict T given the smallest possible subset of {A,B,C}, while achieving maximal performance (accuracy)

假設我們有預測器 A、B、C 和分類器 M。

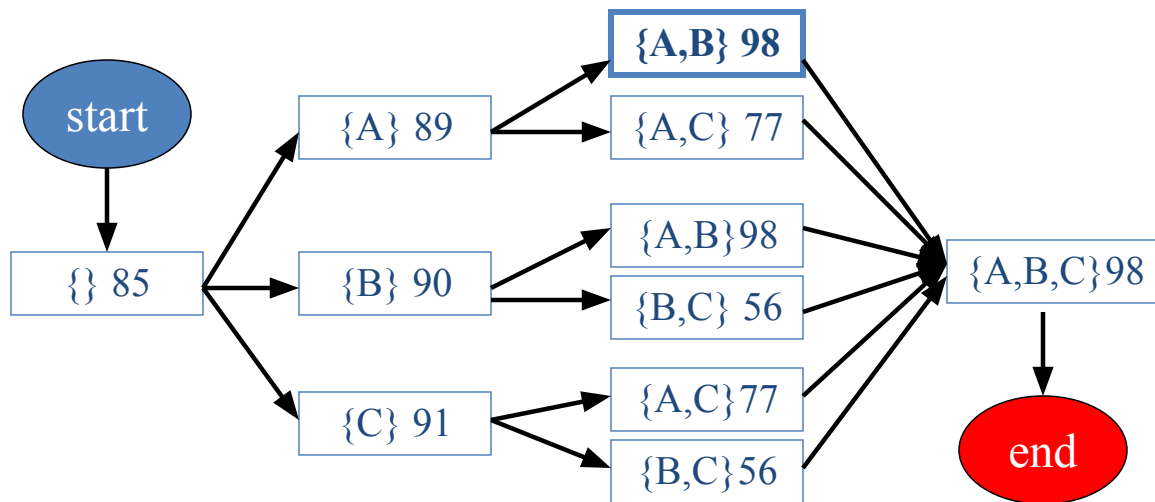
我們希望在給定 {A, B, C} 的最小可能子集的情況下預測 T，同時實現最大性能（準確度）

FEATURE SET	CLASSIFIER	PERFORMANCE
{A,B,C}	$M$	<u>98%</u>
<u>{A,B}</u>	$M$	<u>98%</u>
{A,C}	$M$	77%
{B,C}	$M$	56%
{A}	$M$	89%
{B}	$M$	90%
{C}	$M$	91%
{.}	$M$	85%

## Filters vs. Wrappers: Wrappers

- The set of all subsets is the power set and its size is  $2^{|V|}$ . Hence for large  $V$  we cannot do this procedure exhaustively; instead we rely on heuristic search of the space of all possible feature subsets.

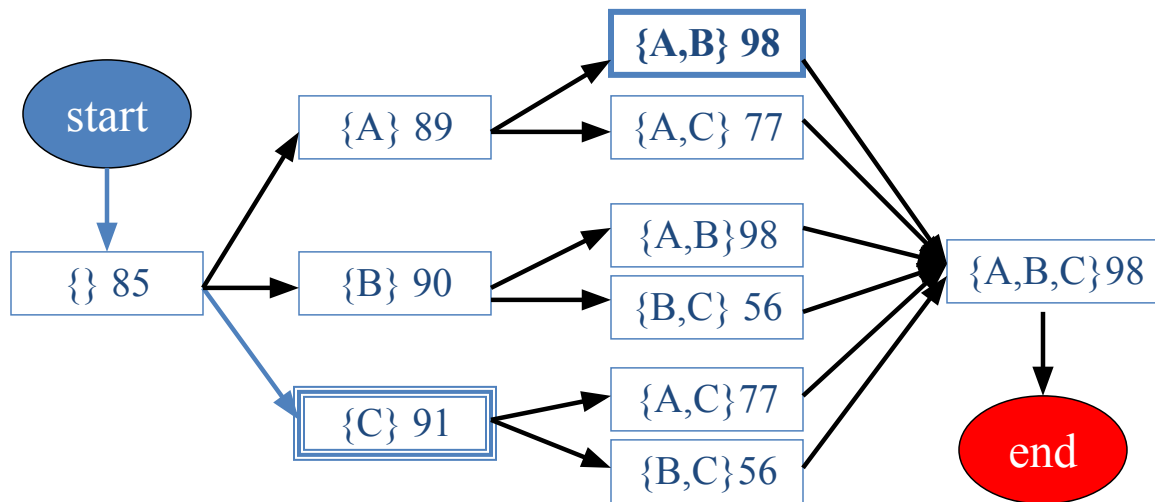
所有子集的集合是冪集，其大小為  $2^{|V|}$ 。因此，對於大  $V$ ，我們無法詳盡地執行此過程；相反，我們依靠對所有可能的特徵子集的空間進行啟發式搜尋。



## Filters vs. Wrappers: Wrappers

- A common example of heuristic search is hill climbing: keep adding features one at a time until no further improvement can be achieved.

啟發式搜尋的一個常見範例是爬山法：一次不斷地添加一個特徵，直到無法實現進一步的改進。



# Filters vs. Wrappers: Wrappers

- A common example of heuristic search is hill climbing: keep adding features one at a time until no further improvement can be achieved (“forward greedy wrapping”)

啟發式搜尋的一個常見範例是爬山：一次不斷地添加一個特徵，直到無法實現進一步的改進（「前向貪婪包裝」）

- Alternatively we can start with the full set of predictors and keep removing features one at a time until no further improvement can be achieved (“backward greedy wrapping”)

或者，我們可以從一整套預測器開始，一次一次刪除一個特徵，直到無法實現進一步的改進（「向後貪婪包裝」）

- A third alternative is to interleave the two phases (adding and removing) either in forward or backward wrapping (“forward-backward wrapping”).

第三種替代方案是在前向或後向環繞（「前向-後向環繞」）中交錯兩個階段（新增和刪除）。

- Of course other forms of search can be used; most notably:

當然也可以使用其他形式的搜尋；最值得注意的是：

- Exhaustive search

詳盡的搜索

- Genetic Algorithms

遺傳演算法

- Branch-and-Bound (e.g., cost=# of features, goal is to reach performance *th* or better)

分支定界（例如，成本=特徵數量，目標是達到性能或更好）

## Filters vs Wrappers: **Filters**

- In the filter approach we do not rely on running a particular classifier and searching in the space of feature subsets; instead we select features on the basis of statistical properties. A classic example is univariate associations:

在過濾方法中，我們不依賴運行特定的分類器並在特徵子集的空間中進行搜尋；相反，我們根據統計特性選擇特徵。一個典型的例子是單變量關聯：

---

FEATURE	ASSOCIATION WITH TARGET	
{A}	91%	Threshold gives suboptimal solution
{B}	90%	Threshold gives optimal solution
{C}	89%	Threshold gives suboptimal solution

# 比較Filters和Wrapper的優缺點

比較面向	Filter 方法	Wrapper 方法
是否使用模型	否	是
評估依據	統計指標（如相關係數、卡方值）	模型效能（AUC、Accuracy等）
考慮特徵間交互作用	否	是
計算成本	低	高
適用情境	高維資料初步特徵篩選	需要最佳化模型表現時
範例方法	皮爾森、卡方	前向選取、遺傳演算法、模擬退火



一、特徵選取 (feature selection)

## 二、退火演算法 (Simulated Annealing)

三、基因演算法 (Genetic Algorithms)

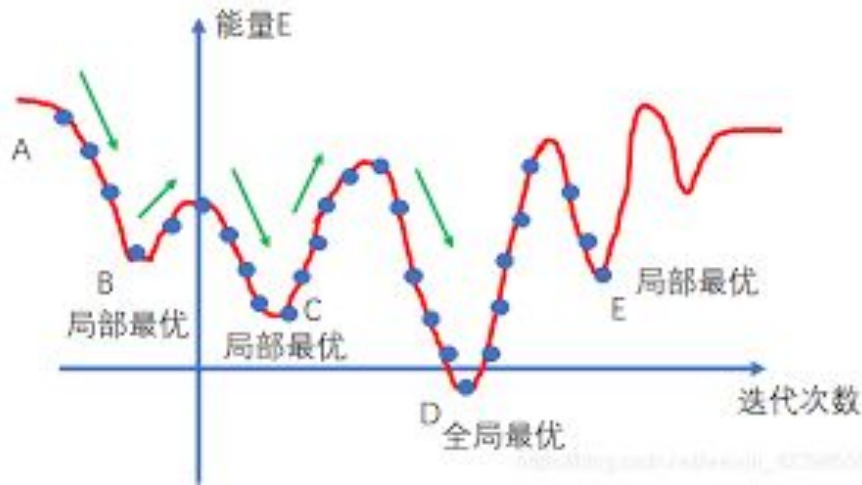
# 退火演算法

## 退火概述：

- 退火（Annealing）是一種冶金過程，指的是將材料加熱後再緩慢冷卻，使內部原子得以重新排列，最終達到內部能量最小化的穩定狀態。這樣的排列會改變材料的性質。
- 在這個過程中，原子的狀態可以被視為「內能的局部最小值」。若要讓材料重新排列原子，必須給予能量（加熱），使其能越過局部最小值，前往全域最小能量的狀態。這樣的「動力」正是透過加熱材料、提升溫度來實現的。

# 退火演算法

- 退火演算法 (Simulated Annealing) 便是模擬這個物理過程。
- 演算法從一個隨機初始狀態（相當於加熱的材料）開始，並設定一個高溫。此時演算法允許接受能量較高的狀態（較差的解），因為**高溫提供了足夠的隨機性，幫助它跳出局部最小值**，朝向全域最小值前進。
- 隨著時間推進、溫度逐漸降低，演算法會越來越傾向拒絕高能量的狀態，最終收斂到目前找到的最佳解。



# 退火演算法

## 步驟一：定義問題 (Define the Problem)

- 明確設定要優化的問題，並定義「能量函數 (Energy Function)」，也就是目標函數
  - 若是最小化問題，能量即為要最小化的函數值。
  - 例：旅行推銷員問題 (TSP) 中，能量 = 總旅行距離
- 同時設定：
  - 初始溫度  $T_{\max}$
  - 初始候選解（可隨機或用啟發式方法產生）
  - 初始能量  $E(x)$

## 退火演算法

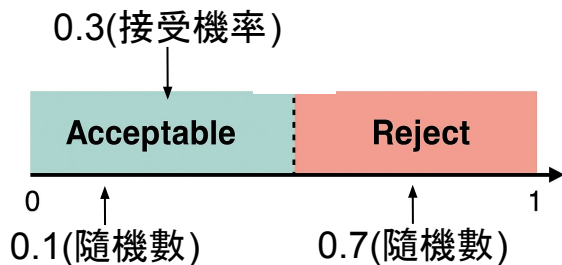
### 步驟二：定義擾動函數 (Define the Perturbation Function)

- 用來產生新的候選解  $x'$
- 新解應與當前解接近，但不完全相同
- 例如：
  - 在函數最小化中：可在目前解上加上隨機值（如  $\pm 0.1$ ）。
  - 在 TSP 中：可隨機交換兩個城市的位置。

# 退火演算法

## 步驟三：設定接受準則 (Acceptance Criterion)

- 判斷新解是否被接受。
- 若新解能量較低 ( $\Delta E < 0$ )，則直接接受。  $P[\text{accept}] = e^{-\frac{E_{\text{current}} - E_{\text{new}}}{T_i}}$
- 若新解能量較高 ( $\Delta E > 0$ )，則以機率接受：  
 $P = e^{-\Delta E/T}$ 
  - 接下來，隨機在  $[0, 1]$  範圍內均勻產生一個隨機數 ( $r$ )
  - 比較接受機率與隨機數 ( $r$ )，若隨機數 ( $r$ ) 小於機率則接受該新解
- 溫度 ( $T$ ) 越高，越有可能接受較差解；溫度越低，則越傾向拒絕壞解。



接受機率  $P=0.3$

隨機數  $r=0.1 \rightarrow$  代表「中了 30% 的機率」

隨機數  $r=0.7 \rightarrow$  沒中，落在拒絕區

## 退火演算法

### 步驟四：溫度下降策略 (Temperature Schedule)

- 利用溫度控制演算法探索與收斂的平衡
- 每次迭代後，依冷卻因子  $\alpha$  進行降溫： $T_{\text{new}} = \alpha \times T_{\text{old}}$ 
  - $\alpha$  介於 (0, 1)，例如 0.9 或 0.95。
  - 高溫 → 探索性強，允許壞解。
  - 低溫 → 收斂性強，只接受更佳解。

## 退火演算法

### 步驟五：主流程 (Run the Algorithm)

- 利用重複以下步驟直到停止條件滿足（如溫度  $<$  最小值或能量  $<$  閾值）：
- 產生新候選解  $x'$
- 計算能量差  $\Delta E = E(x') - E(x)$
- 根據接受準則決定是否更新當前解。
- 更新溫度  $T \leftarrow \alpha T$

# 退火演算法範例

以最小化函數  $f(x, y) = x^2 + y^2$  為例

- 搜尋空間 (Search Space) : 建立一個  $101 \times 101$  的網格, 範圍為

$$(x, y) \in [-10, 10] \times [-10, 10]$$

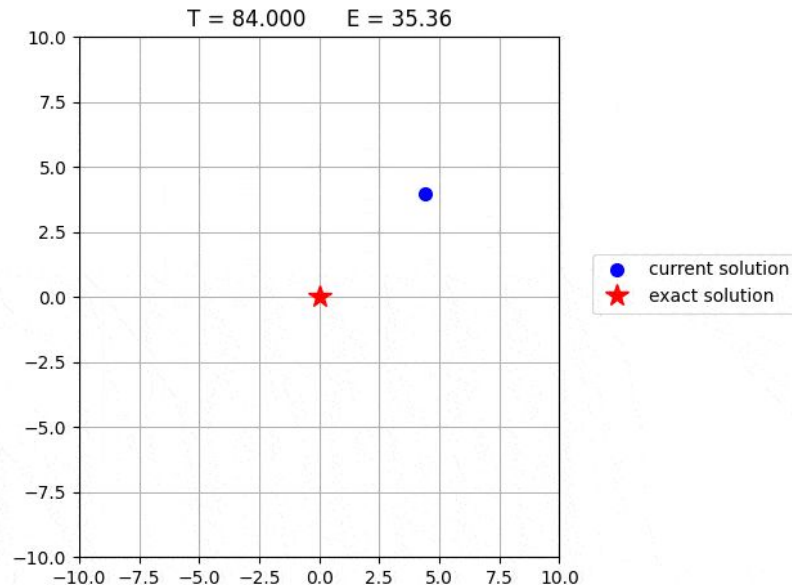
- 冷卻率 (Cooling Rate) : 設定為

$$\alpha = 0.84$$

- 初始解 (Initial Solution) :

$$(x, y) = (4, 4)$$

- 候選解生成方式: 每次迭代時, 隨機將目前解  $(x, y)$  在  $x$  與  $y$  方向上平移  $\pm 0.2$ , 以產生新的候選解。



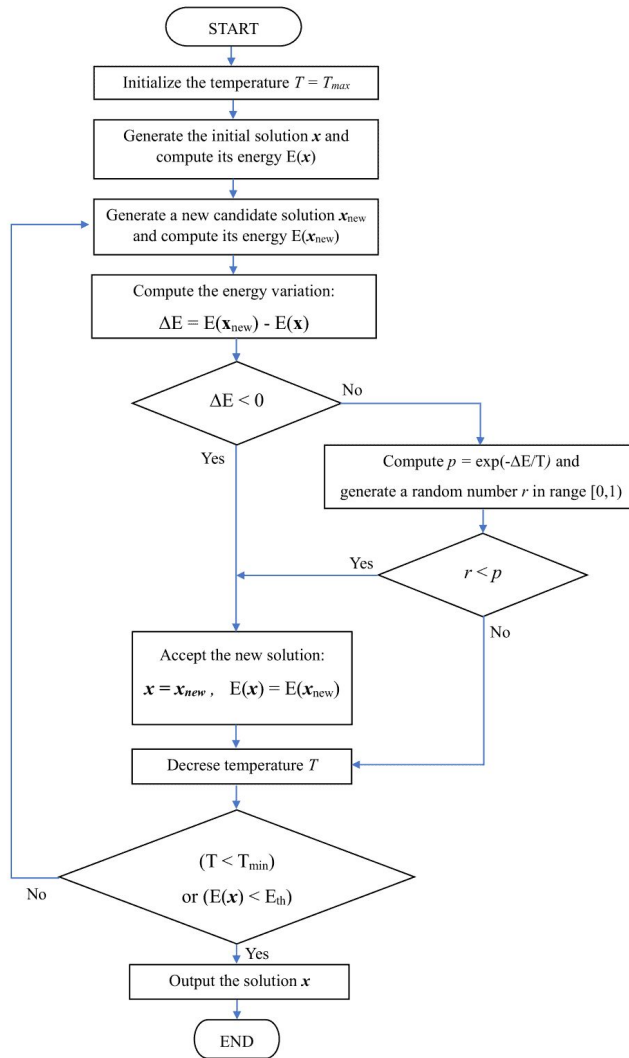
當溫度較高 (例如  $T > 1$ ) 時:

- 系統傾向於**接受較差的解** (能量較高), 以避免陷入局部最小值。
- 模型能「自由探索」搜尋空間。

當溫度較低 (例如  $T < 1$ ) 時:

- 系統變得**更挑剔**, 僅接受能量較低的候選解。
- 模型逐漸收斂到**全域最小值**

# 退火演算法流程圖



# 退火演算法的優缺點

優點：

- **可跳出局部最小值**：透過「以機率接受較差解」的機制，能避免陷入局部最佳。
- **結構簡單、易於實作**：不需維護族群或複雜結構，只需一組解即可運作。
- **計算成本較低**：單一解搜尋，所需記憶體與計算量小。
- **適合連續或非線性問題**：能處理多峰值或非凸的目標函數。

缺點：

- **收斂速度慢**：若冷卻過慢，運算時間長；若過快，容易錯過全域最佳解。
- **缺乏全域探索能力**：僅沿單一路徑進行搜尋，可能忽略其他潛在區域。
- **結果不穩定**：因帶有隨機性，多次執行結果可能略有差異。

## 退火演算法於特徵篩選範例

概述：[\(範例參考來源\)](#)

- 使用[泰坦尼克號資料集](#)建立一個二元分類器
- 採用隨機森林模型（以預測乘客生存）
- 以ROC-AUC 分數作為效能指標。

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	Title
0	0	3	0	22.0	1	0	7.2500	2.0	2	2
1	1	1	1	38.0	1	0	71.2833	0.0	2	3
2	1	3	1	26.0	0	0	7.9250	2.0	1	1
3	1	1	1	35.0	1	0	53.1000	2.0	2	3
4	0	3	0	35.0	0	0	8.0500	2.0	1	2

訓練資料共有**九個**預測特徵和一個**二元目標變數** (Survived\*\*)

# 泰坦尼克號資料集特徵說明

## 資料欄位說明

欄位名稱	意涵說明
Survived	是否存活 (目標變數) 0 = 死亡, 1 = 生還。
Pclass	客艙等級 (社會經濟地位指標) 1 = 頭等艙 (Upper), 2 = 二等艙 (Middle), 3 = 三等艙 (Lower)。
Sex	乘客性別 1 = 男性, 0 = 女性 (或反之, 視資料編碼方式)。
Age	年齡 (以歲為單位)。若缺失通常以平均值或中位數補齊。
SibSp	與乘客同行的兄弟姊妹或配偶數量。
Parch	與乘客同行的父母或子女數量。
Fare	船票價格 (以英鎊計)。通常與 Pclass 正相關。
Embarked	登船港口 0 = Southampton (南安普頓)、1 = Cherbourg (瑟堡)、2 = Queenstown (昆士鎮)。
FamilySize	家庭總人數 (由 SibSp + Parch + 1 組成)。反映是否獨自旅行。
Title	乘客頭銜 (由姓名中提取), 例如 Mr., Mrs., Miss., Master 等, 轉換為數值型特徵以表示社會角色或身份。

## 退火演算法於特徵篩選範例

步驟一：產生特徵的隨機初始子集

- Pclass (客艙等級)、Age、Parch (與乘客同行的父母或子女數量)、Title (Mr., Mrs., Miss., Master 等社會角色或身份)

步驟二：指定要執行的運算數目上限 (例如 50)

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	Title
0	0	3	0	22.0	1	0	7.2500	2.0	2	2
1	1	1	1	38.0	1	0	71.2833	0.0	2	3
2	1	3	1	26.0	0	0	7.9250	2.0	1	1
3	1	1	1	35.0	1	0	53.1000	2.0	2	3
4	0	3	0	35.0	0	0	8.0500	2.0	1	2

## 退火演算法於特徵篩選範例

### 步驟三：評估目前子集上的模型效能

- 在篩選到目前 4 個特徵子集（Pclass、Age、Parch、Title）的資料上訓練隨機森林分類器
- 檢索交叉驗證的 AUC 分數。

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	Title
0	0	3	0	22.0	1	0	7.2500	2.0	2	2
1	1	1	1	38.0	1	0	71.2833	0.0	2	3
2	1	3	1	26.0	0	0	7.9250	2.0	1	1
3	1	1	1	35.0	1	0	53.1000	2.0	2	3
4	0	3	0	35.0	0	0	8.0500	2.0	1	2

## 退火演算法於特徵篩選範例

### 步驟四：變更目前子集中的特徵清單

- 透過新增、取代或移除一個特徵來產生新的子集
- 如加入FamilySize特徵，產生新的 5 特徵子集

注意：如果之前訪問過子集，會重複此步驟，直到產生新的未看到的子集。

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	Title
0	0	3	0	22.0	1	0	7.2500	2.0	2	2
1	1	1	1	38.0	1	0	71.2833	0.0	2	3
2	1	3	1	26.0	0	0	7.9250	2.0	1	1
3	1	1	1	35.0	1	0	53.1000	2.0	2	3
4	0	3	0	35.0	0	0	8.0500	2.0	1	2

## 退火演算法於特徵篩選範例

步驟五至七：依據接受機率（準則）選擇特徵子集

- 在新的子集上運行模型，並將性能指標（即 AUC 分數）與當前子集進行比較。
- 如果新的子集效能較佳，請接受並更新新的子集作為目前狀態，並結束迭代。  
如果效能較差，請繼續執行步驟 7。
- 首先計算接受機率，評估是否接受表現較差的新子集。接下來，在  $[0, 1]$  範圍內均勻產生一個隨機數 ( $r$ )。
  - 如果隨機數大於接受機率，則拒絕新的子集並保留目前的子集。否則，請接受並更新新的子集作為目前狀態。

Iteration Number	Feature Count	AUC-ROC Score	Acceptance Probability	Random Number	Outcome
1	4	0.91	-	-	Initial Run
2	5	0.80 ↓	0.975	0.741	Accept New Subset

## 退火演算法於特徵篩選範例

接受機率 (準則) :

$$P[\text{accept}] = e^{-\frac{E_{\text{current}} - E_{\text{new}}}{T_i}}$$

- E 術語是指績效指標

- AUC 分數。E<sub>new</sub> 是在新子集上訓練的模型的交叉驗證 AUC 分數，而 E<sub>current</sub> 則基於當前子集。

- 兩個項之間的差異表示模型效能的變化

- T<sub>i</sub> 代表運算第 i 次時的溫度。

- 預設最大值 1 來啟動加熱階段

- 溫度會從初始 T<sub>0</sub>冷卻並降低，常見策略為幾何縮減

- 每次迭代後溫度都會按冷卻係數 alpha (α) 縮放。

## 退火演算法於特徵篩選範例

參數和性能影響接受機率(準則):

$$P[\text{accept}] = e^{-\frac{E_{\text{current}} - E_{\text{new}}}{T_i}}$$

- 溫度

- 隨著溫度的降低，接受機率會降低。
- 隨機數大於接受機率的可能性更高，更有可能拒絕表現不佳的新子集。

$E_{\text{current}} - E_{\text{new}}$	Temperature	Acceptance Probability
0.1	1	0.905
0.1	0.85 ↓	0.889 ↓

- 新子集模型性能不佳到更大程度（即 $E_{\text{current}}$ 和  $E_{\text{new}}$ 之間的差異更大）
  - 接受機率會降低，更有可能拒絕表現不佳的新子集。

$E_{\text{current}} - E_{\text{new}}$	Temperature	Acceptance Probability
0.1	1	0.905
0.2 ↑	1	0.819 ↓

## 退火演算法於特徵篩選範例

終止條件：

- 達到可接受的效能門檻 (AUC分數)
- 達到特定終端溫度 (例如,  $T=0.01$ )
- 達到預先決定的連續反覆運算次數, 而沒有效能改善

# 退火演算法於特徵篩選範例

Iteration	Feature Count	Feature Set	Metric	Best Metric	Acceptance Probability	Random Number	Outcome
0	1	4	[8, 2, 4, 5]	0.838	0.838	-	- Improved
1	2	5	[1, 2, 4, 5, 8]	0.842	0.842	-	- Improved
2	3	5	[1, 2, 4, 5, 8]	0.834	0.842	0.988412	0.995764 Reject
3	4	5	[0, 1, 2, 5, 8]	0.867	0.867	-	- Improved
4	5	5	[0, 1, 2, 6, 8]	0.838	0.867	0.943198	0.800376 Accept
5	6	5	[0, 1, 3, 6, 8]	0.848	0.867	-	- Improved
6	7	5	[0, 1, 2, 3, 6]	0.845	0.867	0.991662	0.532136 Accept
7	8	4	[0, 2, 3, 6]	0.69	0.867	0.601126	0.157681 Accept
8	9	4	[0, 2, 3, 6]	0.617	0.867	0.754274	0.842234 Reject
9	10	5	[0, 2, 3, 6, 8]	0.845	0.867	-	- Improved
10	11	4	[0, 8, 2, 3]	0.855	0.867	-	- Improved
11	12	4	[8, 2, 3, 7]	0.794	0.867	0.681332	0.344293 Accept
12	13	4	[8, 2, 5, 7]	0.855	0.867	-	- Improved
13	14	4	[8, 4, 5, 7]	0.844	0.867	0.908674	0.570306 Accept
14	15	5	[4, 5, 6, 7, 8]	0.833	0.867	0.893446	0.385079 Accept
15	16	5	[3, 4, 5, 7, 8]	0.842	0.867	-	- Improved
16	17	5	[3, 4, 5, 7, 8]	0.798	0.867	0.535919	0.984426 Reject
17	18	4	[8, 3, 4, 5]	0.84	0.867	0.967193	0.152562 Accept

Pclass、Sex、Age、Fare 和 Title



一、特徵選取 (feature selection)

二、退火演算法 (Simulated Annealing)

## 三、基因演算法 (Genetic Algorithms)



# 基因演算法

## 遺傳學概述：

- 在比較高等的動物中，這些傳遞下來的DNA，藉由雜交融合了其它適者的DNA，這種技術，稱之為雜交（Cross over）
- 有時候傳遞基因給下一代的過程也會有出錯，稱之突變（mutation）
- 這所有的過程加起來，一代一代傳遞，  
使得生物高度適應其環境：這就是演化過程

## 電腦上的遺傳學

一個簡單的例子來解釋基因演算法如何運作：

在一個只有一個參數『p』簡單函數，p值為在0 至31之間，該函數  $31p - p^2$  見右圖。

這個例子可視為一個基因組 (genome)，只含有一個5位元基因參數p。（需要5位元來表達0到31）這個函數的最大值出現在p為15與16時，分別以01111與10000代表。這個例子也顯示即使有數個最適狀態，基因演算法依然適用。

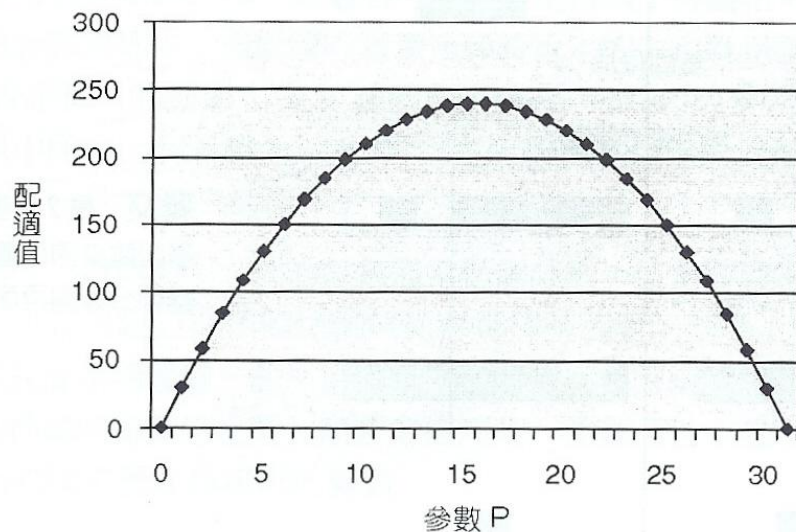


圖 14.2 這個簡單的函數有助於解釋基因演算法

# 電腦上的遺傳學

我們必須選擇所謂的**適配函數** (fitness function) 來解決這個問題。在這個例子裡，適配函數就是  $31p - p^2$ ，然後進行以下一些步驟：

1. 定義基因組以及適配函數，創造第一代的基因組。
2. 藉由選擇雜交，以及突變，來修訂起始群體。
3. 重複步驟2，直到這個群體不再進步為止。

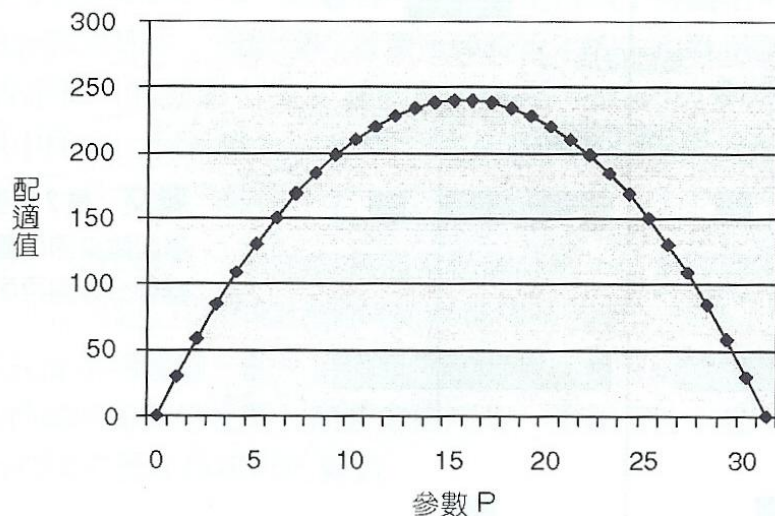


圖 14.2 這個簡單的函數有助於解釋基因演算法

## 電腦上的遺傳學

運用基因演算的**第一個步驟**就是**設定問題**。

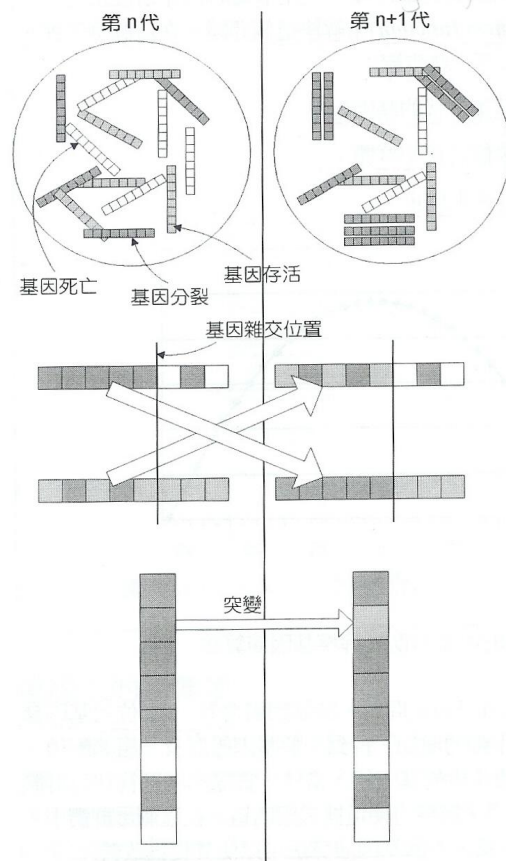
- 該基因組含有一個5位元基因參數p。
- 如下表顯示，第一代有四個隨產生的基因組。
- 平均適配值是117.75，已經相當不錯，不過基因演算法可以使其更為改善。

適配值	基因	p
176	10110	22
	00011	3
87		

# 電腦上的遺傳學

如右圖所示，基本演算過程利用三個運算元來修訂起始群體：

- 選擇(selection)
- 雜交(crossover)
- 突變(mutation)



**選擇** 保持母體個數不變，但新一代的適合度會比上一代更好。

有較高的適合度的基因(深色)會存活，增殖而陰影部份的基因(淺色)死亡

**雜交** 是 2 個基因結合的方式。基因雜交的位置決定了基因分裂與再度重組

**突變** 是基因在某個位置上突然的改變這會出現與原先母體不曾有過的特徵

圖 14.3 基因演算法中三個基本運算元是，選擇、基因雜交、突變

## 電腦上的遺傳學— 選擇

在我們的計算過程中，母群體的數量是保持恆定的，也就是說該群體不致滅種。基因組存活並繁衍的機率，與其**適配值（fitness value）成正比**：**值越高者，在下一代基因組中所佔比例就越高**。按比例計算後，遺傳給下一代的數量通常是個分數，但是群體中基因組的數量卻絕對是整數。生存是建立在依照比例再任意選擇基因組的基準上。

基因 代數	適配值	適配值百分比	期望子
10110 1.50	176	37.4%	
00011 0.74	87	18.5%	
00010 0.42	58	12.3%	

## 電腦上的遺傳學- 選擇

這種選擇的過程，如下表所示，使得 4 個基因組出現生存的勝利者。  
值得注意的是，傳遞下來的基因組，適者較多，不適者較少。

- 例如：00011就沒有在這一次的选择过程中生存下来
- 而10110這個最適者則佔了2個名額
- 母群體的平均適配值從117.75提高140

基因	適配值
10110	176
11001	150
00010	
58	
10110	176

## 電腦上的遺傳學- 雜交

下一個運算元的因素則是雜交。雜交也發生在自然界中，將 2 組基因組原有的某一部分與另一組雜交，產生了新的 2 組基因組。

如前圖所示，雜交發生在 2 組基因組之間，且發生的節點位置乃隨機決定。舉例來說：10110 與 00010 這 2 組基因組在 2 與 3 之間的位置做雜交。過程如下：

基因	適配值
10110	176
11001	150
00010	
58	
10110	176

10		110
00		010

## 電腦上的遺傳學- 雜交

經過雜交後（加底線者為原本第 2 個基因組）

$$\begin{array}{cc|c}
 10 & 110 & \\
 00 & 010 & \\
 \hline
 \end{array}
 \longrightarrow
 \begin{array}{cc|c}
 10 & & \underline{010} \\
 \underline{00} & & 010
 \end{array}$$

新產生的基因組，稱為**子嗣 (children)**，皆繼承了來自雙親的某部份。

雜交進行的過程就是選擇某一對基因後，再如擲銅板般隨機決定是否雜交基因。

這個機率稱之為**雜交機率 (crossover probability)**，通常以 **pc** 表示。

0.5 的雜交機率（與擲銅板擲到人頭那一面的機率相同）通常會帶來好結果。

## 電腦上的遺傳學- 雜交

在這個例子中，10110 與 00010 這2個基因組被選擇在2與3位元之間的位進行雜交，如下表。

經過選擇與雜交，母群體的平均適配值由117 . 75提高到178 . 5。

這對僅僅經過一代是相當了不起的進步。

適配值	基因	p
234	10010	18
	11001	25
150	00110	6

## 電腦上的遺傳學- 突變

最後一種運算元的因素是突變。突變在自然界中極少發生，是因為親代在繁衍後代時，基因編碼產生了錯誤所致。

在提高適應性上，選擇和雜交相當有助益，但與其原先的狀況有密切相關，隨機的特性也可能阻礙一些基因產生更好的組合。

突變提供了另一種可能。自然界中，突變極少發生，在基因演算法裡其比例也很低，每一代的發生機率小於1為合理。

## 電腦上的遺傳學- 突變

假設接下來這一代第2個基因組第3個位元的位置發生了突變。下表顯示了經過突變後的基因組。如同多數突變一般，這個突變造成毀滅性的結果：該突變基因組也不可能存活至下一代。

適配值	基因	p
234	10010	18
58	11101	29
150	00110	6

## 基因演算法總結

基因演算法就是基因在一代一代傳遞的過程中，藉由選擇，雜交，以及突變來不斷改善自身的適應力。它不一定會導向**最適境界**（optimal solution），但卻可以很快地接這個最適境界。

# 基因演算法-手寫範例

假設所處理的函數是  $f(x)=x^2$ ，為了縮小搜尋範圍，將變數  $x$  的範圍限制在  $[0, 32]$ ，希望能找到在限制範圍內的函數最大值，我們選定以五個位元來編碼變數  $x$ ，交配機率設定為 1.0，突變機率設定為 0.1：

## • 步驟一、產生初始族群

設定族群數目 (Population) 的大小為 4，隨機地產生如下表所示四個染色體：

染色體編號	染色體內容	對應之數字
1	01110	14
2	11000	24
3	10001	17
4	00111	7

## • 步驟二、複製

直接以此函數當做為適配函數：

$x$	$f(x)$	$f_i / \sum f$	複製個數
14	196	0.18	1
24	576	0.52	2
17	289	0.26	1
7	49	0.04	0

## 基因演算法-手寫範例

假設所處理的函數是  $f(x)=x^2$ ，為了縮小搜尋範圍，將變數  $x$  的範圍限制在  $[0, 32]$ ，希望能找到在限制範圍內的函數最大值，我們選定以五個位元來編碼變數  $x$ ，交配機率設定為 1.0，突變機率設定為 0.1：

- 步驟三、交配
  1. 第一個染色體被複製一個至交配池中
  2. 第二個染色體被複製二個至交配池中
  3. 第三個染色體被複製一個至交配池中
  4. 第四個染色體被淘汰

# 基因演算法-手寫範例

- 投擲兩枚硬幣，以決定交配池中字串的配對
- 交配池中的第一個字串與第二個字串交配，第三個字串與第四個字串交配

交配池	配對	交配點	下一子代	$x$	$f(x)$
01110	2	2~5	01000	8	64
11000	1	2~5	11110	30	900
11000	4	4~5	11001	25	625
10001	3	4~5	10000	16	256

- **步驟四、突變：**突變過程的作法是將染色體中的基因隨機的由 0 變 1 或由 1 變 0。
- 由於設定的突變率為 0.1，而族群中的基因總數為族群大小乘上每個染色體的編碼位元數，也就是  $4 \times 5 = 20$
- 因此，族群中將被突變的基因總數為  $20 \times 0.1 = 2$ ，亦即，有兩個基因將被突變。我們隨機地挑選一個字串中的一個基因來作突變，舉例來說，我們突變第一個字串中的第三個基因，則：

突變前：01000

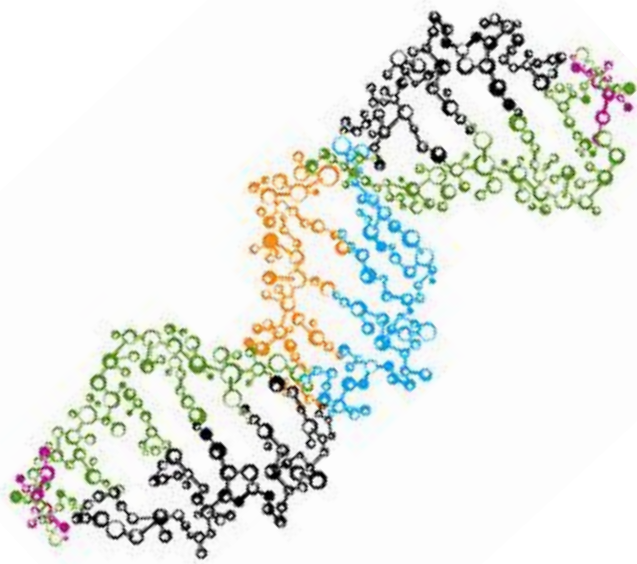
突變後：01100

## 基因演算法-手寫範例

- **步驟五、終止搜尋**：判斷目前的最佳適配函數值是否已達所須要的標準
- 若是，則終止搜尋
- 若否，則回到步驟二，以進行下一代的演化。

# 基因演算法之主要特性

- 基因演算法是以參數集合之**編碼**進行運算而不是參數本身，因此可以跳脫搜尋空間分析上的限制。
- 基因演算法同時考慮**搜尋空間上多個點**而不是單一個點，因此可以**較快地獲得整體最佳解**（global optimum），同時也可以避免陷入區域最佳值（local optimum）的機會，此項特性是基因演算法的最大優點。



## 基因演算法之主要特性

- 基因演算法只使用適配函數的資訊而不需要其它輔助的資訊(例如梯度)，因此可以使用各種型態的適配函數，並可節省計算機資源避免繁複的數學運算。
- 基因演算法使用機率規則方式去引導搜尋方向，而不是用明確的規則，因此較能符合各種不同類型的最佳化問題。



## 基因演算法之細部探討

- 編碼範圍
- 字串長度
- 族群大小
- 交配機率
- 突變機率
- 適配函數之設計
- 菁英政策

## 基因演算法之細部探討

- 編碼及解碼過程：

假設受控系統中有三個參數要編碼，三個參數值均界於  $[0, 1]$  之間，且每個參數使用五個位元加以編碼，則二進制字串編碼流程如下：

隨機設定三個變數值於  $[0, 1]$ ，假設： $x = \frac{1}{2^5 - 1}$ ,  $y = \frac{2}{2^5 - 1}$ ,  $z = \frac{4}{2^5 - 1}$

則編碼為：
$$\begin{cases} x = 00001 \\ y = 00010 \\ z = 00100 \end{cases}$$

最後的字串為：00001 00010 00100.  
而解碼流程則以反對順序即可完成。

## 基因演算法之細部探討

- **字串長度：**  
長度越長則精準度越高，但所須的編碼、解碼運算也相對增加。
- **交配機率：**  
交配率越高，則新物種進入族群的速度越快，整個搜尋最佳值的速度也越快。
- **突變機率：**  
突變是一項必須的運算過程，因為在複製及交配過程中可能使得整個族群裡，所有染色體中的某一特定位元皆一樣。

## 基因演算法之細部探討

- **避免陷入區域最佳值：**  
須視所搜尋空間的維度及參數範圍大小與編碼時所採用的精確度(字串長度)一起考量
- **適配函數之設計原則：**  
原則上，適配函數須能反應出不同物種間適應程度的差異即可
- **搜尋終止之條件：**  
對於某些線上即時系統而言；為了結省時間，當適配函數值到達系統要求後即可終止搜尋程序。

## 基因演算法之細部探討

- 菁英政策:

在某些特殊情況下，交配後的結果可能會讓子代比父代差。

例如：11100 (父1)    fitness:  $28^2$

00001 (父2)    fitness:  $1^2$

10001 (子1)    fitness:  $17^2$

01101 (子2)    fitness:  $13^2$

如果子代完全取代父代，則演化結果反而是退步的，所以目前大部分的基因演算法會將**菁英政策**列為必要步驟。

## 實數型基因演算法則

- 基因演算法則在編碼及解碼的運算上相當地**耗時**，尤其是當目標函數**參數增多**時，**編碼及解碼**所需的浮點數運算將嚴重地減緩搜尋程式的執行速度。
- 而且編碼時所使用的**字串長度**若不夠，將可能使基因演算法則雖然搜尋到系統整體最佳值出現的區域，但由於準確度不足的關係，只能搜尋到**整體最佳值的附近**而無法真正搜尋到整體最佳值。

## 實數型基因演算法則

- 大部份自然界中的最佳化問題的參數型式皆為實數參數，因此最好是直接以實數參數來運算，而不是透過離散式的編碼型式。
- 直接以實數運算的好處是，不但可以免去做編碼及解碼運算，而且可以提高系統準確度。
- 二元編碼：01001
- 實數編碼：9

## 實數型基因演算法則

- 實數型的**交配**過程：

交配的過程會使得兩個物種之間的距離變得更近或更遠。

- $$\begin{cases} \underline{x}'_1 = \underline{x}_1 + \sigma(\underline{x}_1 - \underline{x}_2) \\ \underline{x}'_2 = \underline{x}_2 - \sigma(\underline{x}_1 - \underline{x}_2) \end{cases} \quad \begin{cases} \underline{x}'_1 = \underline{x}_1 + \sigma(\underline{x}_2 - \underline{x}_1) \\ \underline{x}'_2 = \underline{x}_2 - \sigma(\underline{x}_2 - \underline{x}_1) \end{cases}$$

- $\sigma$  是隨機選取的微量正實數。

## 實數型基因演算法則

- 實數型的**突變**過程：

突變過程為隨機選取一個基因，並且加入微量或較大量的雜訊即可；但加入雜訊後必須確保新的數值仍然保留在所定的參數範圍中。

$$\underline{x} = \underline{x} + s \times random\_noise$$

- 其中  $s$  控制所加入雜訊之大小。

## 實數型基因演算法的優缺點

優點：

- **無需編碼與解碼**：直接以實數表示染色體，減少運算轉換的繁瑣過程。
- **精度更高**：克服二進位編碼帶來的精確度限制，適合連續參數優化。
- **全域搜尋能力強**：透過交叉與突變機制，提升搜尋全域最佳解的機率。

缺點：

- **參數敏感**：交叉率、突變幅度需依問題特性調整，否則易收斂過慢或陷入局部最小值。
- **早熟收斂風險**：若突變率太低，族群多樣性下降，容易集中於局部區域。
- **運算成本高**：需同時評估多個候選解（族群），計算資源需求大。

## 比較退火演算法和基因演算法優缺點

項目	退火演算法 (SA)	遺傳演算法 (GA)
搜尋方式	單一解逐步隨機搜尋	多解並行族群演化
優點	結構簡單、易實作 可跳出局部最小值 計算量較小、參數少	全域搜尋能力強 可同時探索多區域 適合高維與複雜問題
缺點	探索範圍有限 對降溫設定敏感 易陷局部最小	計算成本高 參數多難調整 可能早熟收斂
收斂速度	較快但可能不穩定	較慢但較穩定
適用情境	小規模、連續優化	高維度、全域最佳化

## Week9 程式作業 程式碼說明

使用退火、基因演算法  
做特徵篩選

[參考程式碼連結](#)