基因演算法分類器

機器學習期末報告

指導老師 ： 陳瑞彬 教授

學生 ： 黃乙修 張銓晉 陳柏愷 于立宏

摘要

我們使用的分類方法，一般而言時常來自兩種測度，一是資料點間的距離(例如 KNN 法)，二是透過變數估計機率(如羅吉斯迴歸法)。然而，實際上能同時且正確處理分類以及變數選擇是不容易的。我們想要透過基因演算法盡可能演化出固定條件下所能達到最好的分類效果，其條件包含使用變數的個數，演化過程的迭代次數等。

過程中我們另外以線性判別分析法以及廣義線性模型作為分類標準，並配合 5 折交叉驗證得出每次資料測試集依該次模型所得的準確率作為最後判斷建議使用變數的依據。我們共使用三組資料演示我們的分類器，分別是一組模擬生成的資料，一組與紅酒白酒有關的真實資料以及另一組有關於鮑魚年齡大小的真實資料。

關鍵字 : 基因演算法(genetic algorithm); 線性判別分析(Linear Discriminant Analysis); 羅吉斯迴歸(Logistic regression)

目錄

圖目錄

表目錄

一、基因演算法簡介

基因演算法的原始用途在於最佳化的求算。在我們有興趣的領域裡，我們把最佳化解釋為分類的準確率，因為在 Supervised Learning 之下這是一個合理的目標。以下我們逐個解釋基因演算法的過程，並會對各個步驟依照我們的應用方式另加介紹。

- **隨機生成染色體:**

  在求解最佳化問題時，每一組染色體皆代表一組可能的解。這一組解的型態可以自行去定義。常見的方式有以二進位表示變數等。在我們的分類器裡，每一組染色體代表一種可能的解釋變數組合，以該組染色體每個位置的 0 或 1 表示該位置編號的變數在這次組合裡有無被納入。在此階段同時會依照所設定每次迭代所使用的數目來生成數個染色體，以進行後續步驟。

- **計算適應函數:**

  適應函數用來決定每組染色體表現的好壞，通常會跟目標函數直接相關。以極值求解為例，適應函數很自然會視該問題的目標函數。在我們的分類器裡，適應函數是每次資料測試集套用了訓練集模型所得到的準確度。

- **選擇、複製:**

  計算適應函數的目的即在於，透過該染色體的表現來決定演算法的此階段中，這個染色體的去留。以極值求算(求最大)為例，每次生成幾組染色體並計算出適應函數值後，其值越大我們給該組染色體較大的機率被留下。以每次迭代生成 5 組染色體為例，我們在指派完此 5 組染色體各自被留下的機率後，就依照其機率抽出下次進入迭代所使用的 5 組新染色體，而很重要的是這一次抽取是採取後放回的方式，讓表現較好的染色體組合不僅有更高的機率被留下，還有更高的機率加入下一輪的迭代。

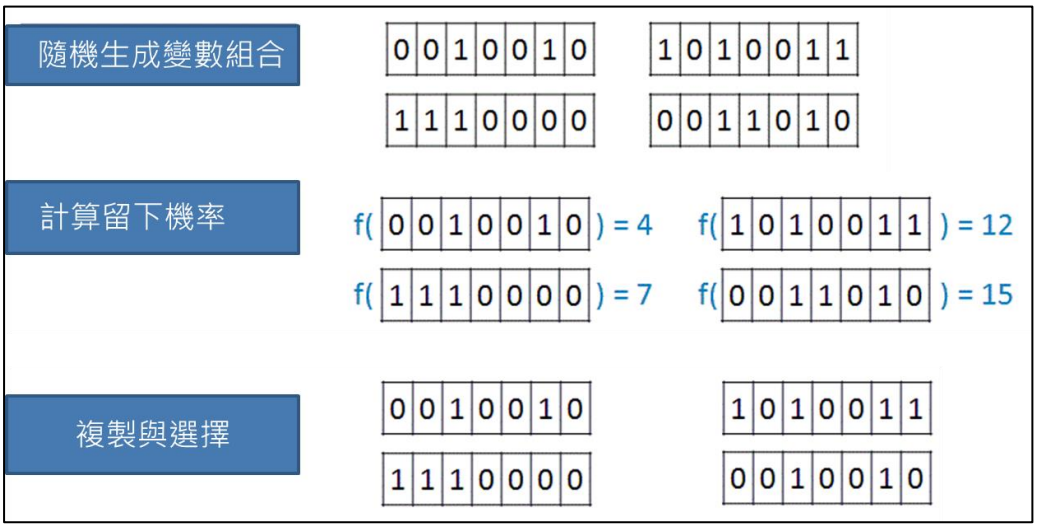

圖 1-1:演算法流程實例之一

- **交配:**

交配的過程，旨在經過一些狀況改變，卻能留住部分親代的特徵。以生物為例就非常好理解了。在我們的分類器裡，某兩組染色體的交配，表示其中一些變數組合(仍為 0 或 1)會進行互換。雖然製造出差異，但是若染色體差異已經漸小，或者此時還保留了實際上欲經過此過程汰除的非重要變數，只進行交配的意義不大，因為會過於快速的收斂。因此，需要有下一步驟突變的存在。

- **突變:**

回到生物演化的例子，演化的精髓就在於適時的突變所造成的新可能性。突變係指規模大過個體差異許多的變異型態，以我們的分類器而言，突變會在某個偏低的機率下直接改變染色體裡一個基因的數值(由 0 改為 1 或反之)。突變的用意在於在當親代交配所產生的子代過於相似，演化容易收斂而落入區域最佳解。有了突變則能某種程度增加基因池的變化性，就算是失敗的突變在後續的迭代裡也很容易被淘汰。而在一輪進行到決定不突變或者突變完成後，即根據滿足終止條件與否決定下一步。若滿足則停止，不滿足則進入下一輪迭代直到滿足為止。

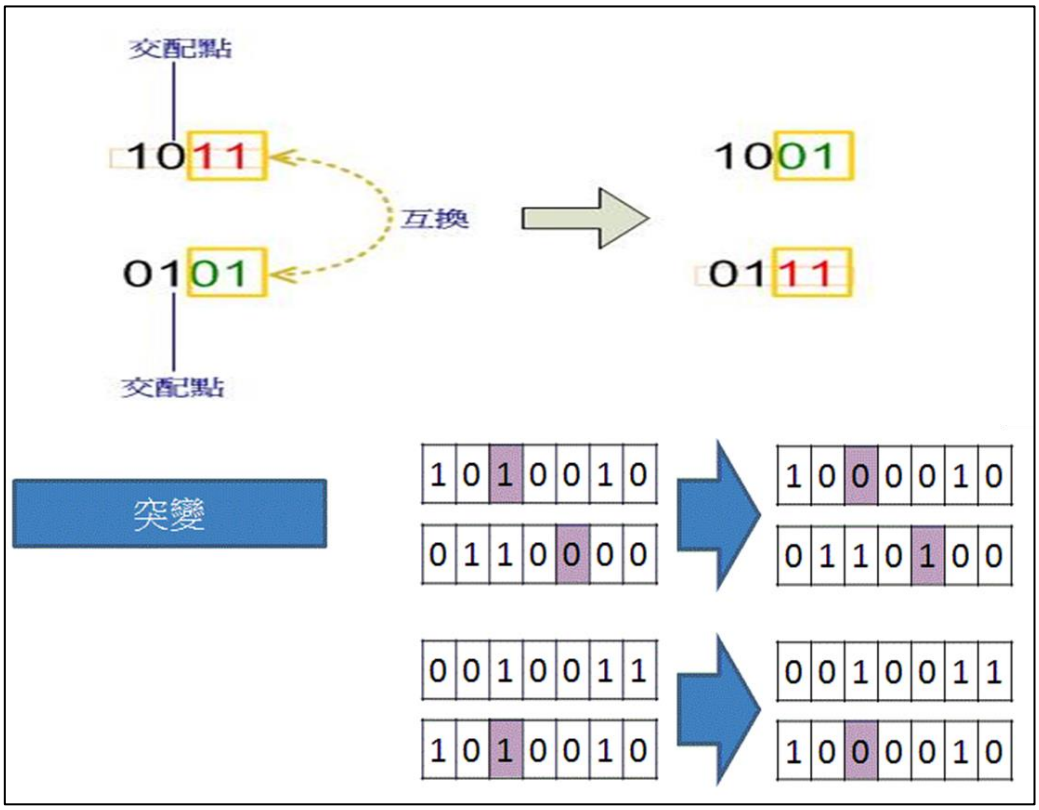

圖 1-2:演算法流程實例之二

- **滿足終止條件:**

  基因演算法是透過迭代不斷進步逼近最佳解的方法，因此設定終止條件是必要的。對於一筆資料其準確率究竟能逼近什麼水準我們並不能事先得知。以股市漲跌為例，一般而言在長期只要準確率大過 50%就足以稱之為勝利。然而若是利用分類器設定了準確率 90%以上才終止，那麼程式很可能運算完成之前就負荷過重。因此替代的方案是直接設定好迭代次數。只要能事先知道解釋變數的數量，使用者可以自行透過簡單的排列組合運算概估需要多少次的迭代較為保守。

我們將整套演化過程以簡單的流程圖表示之。並附上較為實際具體的範例圖示。

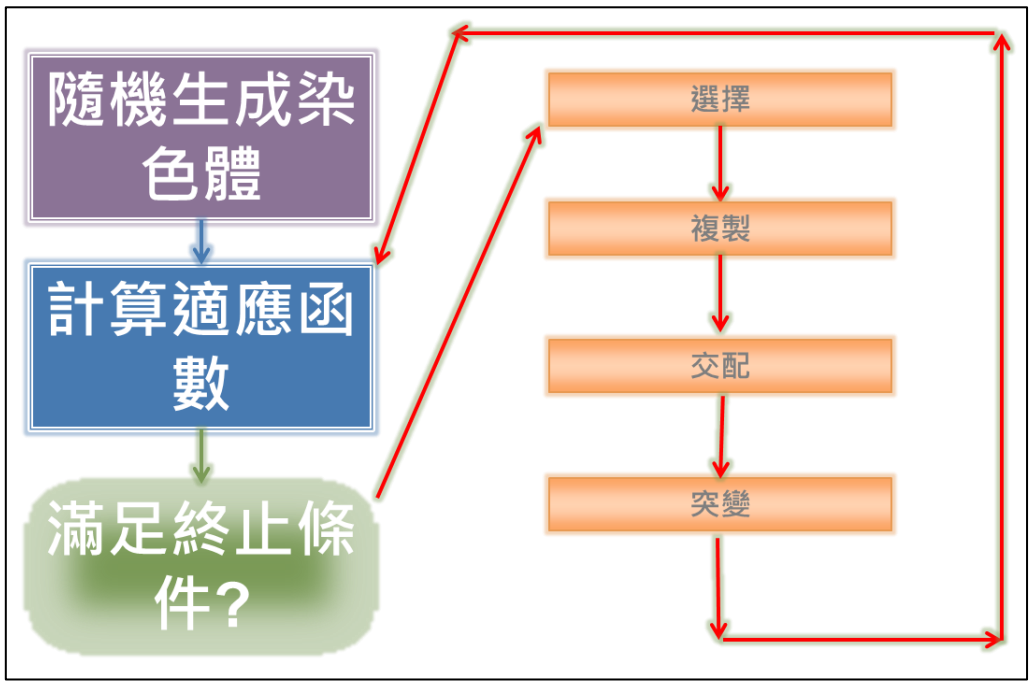

圖 1-3: 基因演算法流程圖

二、實作內容

以下我們將大致講解分類器演算法的函數編寫過程。其中我們使用到 R 裡面 MASS package 的一些內容。

- **GENE 函數**

這是我們主要的函數。段落包含在基因演算法裡的流程。隨機生成、選擇、複製、交配、變異以及終止條件。其參數包括每次迭代的染色體數 k，迭代次數(即我們分類器的終止條件)n，我們所定義的重要度係數 imp，突變機率 evo.prob，交配機率 sex.prob，選擇模式 choose.type，讀入資料檔 data，模型種類 model.type。

其中重要係數 imp 是我們所設計，用來調整模型準確度與複雜度的參數。時常我們在改進模型時會遇到表現能進步，複雜度卻也隨之提高的狀況。因此我們使用此係數(0 到 1 之間)讓使用者調整兩者權重。imp 高則選擇機率評比將偏向準確度，而 imp 低時，則選擇機率表現偏向納入變數個數考量的情況。

選擇模式 choose.type 則是有 rank 與 unrank 兩種可能的輸入。當輸入為 rank 時，選擇的機率將以各組染色體適應函數之”排序”再配合重要度係數作計算。而當輸入為 unrank 時則不加以排序，以準確度及變數個數之原始情況做選擇機率的計算。我們考量在函數參數中加入選擇模式 choose.type 的原因是為了能在資料量龐大時多少能簡化計算。而使用與否則完全由使用者決定。

另外對於讀入資料檔 data 的部分，我們期待格式是 R 當中的 data frame，並且反應變數在最後一直行上。如此能夠進一步避免使用者在輸入參數時還得去對照變數名稱。

至於模型種類 model.type，我們在函數中設計了兩種建模方法供使用者選擇。其一是線性判別分析法(LDA)，二是使用二項分配家族的廣義線性模型(GLM)。為此，若資料反應變數的種類多過兩種，使用上就需特別注意。

- **GENE4InBox 函數**

在 GENE 函數裡，我們加入了表示演算次數的參數 r。根據使用者輸入的演算次數，GENE 函數會被做 r 次。同時在 GENE4InBox 函數裡，會回報 4 種表現良好的模型。事實上，如果表現良好的變數組合非常少，在此函數裡是有可能看到兩組或以上同樣的模型的。然而，就算是一樣的變數組合，也可能會因為該次演算裡使用的訓練集和測試集不同，導致所計算出的準確率亦有不同。而在函數最後，我們將此函數的輸出更改其類別為我們所定義使用的 GENE。

- **print.GENE 函數**

  給定一群由 GENE4InBox 函數所生成的 GENE 物件，print.GENE 函數會表列出這些模型所使用的變數組合，同時每一變數組合底下會印出根據 GENE4InBox 函數所算出的單一次準確率。

- **GENE.cv 函數**

  在 GENE.cv 函數裡，主要的目標是根據 GENE4InBox 函數裡所生成的 r 種模型進行 k 折交叉驗證的運算，其中 k 是 GENE.cv 的參數之一。另一個參數就是經過 GENE4InBox 函數得出的 GENE 物件。定義類別的好處在於，能夠避免非類別成員的誤用，而此函數則是在我們的分類器裡最後一個階段。它會回報 r 種變數組合作為建模的參考，並在每一種變數組合之下回報 k 組測試集依訓練集模型建模的準確度。值得一提的是，由於是依序處理 r 種組合，不同組合間的訓練集測試集都是不同的，如此更能另外避免重複使用到一組代表性不佳的樣本。

三、資料模擬

為了驗證我們的分類法有無效果，我們首先採用模擬的資料來驗證。在生成模擬資料時，我們設計一函數 simdata，輸入生成筆數 n 以及變數個數 p 後，此函數首先會生成 n 筆平均皆為 1，變異數亦皆為 1 且服從 p 維常態的模擬資料，其中各維度間互相獨立。在這些變數都被生成了之後，我們拿其中一些維度利用羅吉斯迴歸的公式倒算出一機率，並將其依照 0.5 作為界限分成兩類。
以下我們表列生成的模擬資料，共取十筆。

表 3-1:模擬之多元常態資料

編號	V1	V2	V3	V4	V5	resp
1	0.282508	0.324065	0.757916	0.159798	0.438729	small
2	0.010999	-2.03707	1.701842	1.892278	-0.11151	big
3	-1.30777	3.163392	1.386577	1.316738	0.174927	small
4	3.125604	0.026483	0.028144	1.295841	-0.20301	big
5	1.710225	1.151577	0.07274	0.584553	1.243777	big
6	1.487735	0.592485	1.027044	0.868408	2.5395	big
7	1.229253	0.894658	-0.08126	0.533236	1.7233	big
8	0.802396	0.013167	-0.41237	-0.27347	1.964815	big
9	1.750469	-0.54293	1.294753	0.124179	0.750528	big
10	-0.27747	1.88536	2.160511	1.992976	1.81802	small

在有了這樣 5 維的資料，並且取了當中的 V1，V2，V3 出來做羅吉斯迴歸的運算得出機率值 Y 並依其生成二維的反應變數 resp 後，我們即套用我們的基因演算法分類器執行運算，所得出的交叉驗證結果如下圖所示。

```
> GENE.cv(GENE4InBox(data=trial1,model.type="glm"))
$`resp~V1+V2+V5`
[1] 0.780 0.785 0.735 0.800 0.795

$`resp~V1+V3`
[1] 0.740 0.750 0.760 0.750 0.745

$`resp~V1+V2`
[1] 0.725 0.780 0.800 0.820 0.785

$`resp~V1+V2+V3+V5`
[1] 0.725 0.745 0.810 0.815 0.775
```

圖 3-1:模擬資料分類結果

從結果中可以看見，雖然部分組合誤選了不重要變數 V5，但整體而言表現不錯，也從來沒有一個組合有漏掉所有重要變數過。我們推測選入 V5 的原因一是樣本數少，加上 5 組變數皆是平均 1 變異數 1 量級過於接近，二是測試集抽取上抽到的樣本代表性不足所致。

四、真實資料及分析結果

接著是使用我們的分類器面對兩筆真實資料做應用。以下我們將簡介各筆資料以及介紹其分類的用途。

4.1 資料一

我們的資料一是有關於紅酒和白酒的真實資料。筆數共有 6497 筆，加上反應變數(red/white)共有 13 個變數。以下我們簡單列出其中一些。

表 4-1:紅白酒資料簡示

編號	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	…….	class
1	7.4	0.7	0	1.9	……	red
2	7.8	0.88	0	2.6	……	red
3	7.8	0.76	0.04	2.3	……	red
4	11.2	0.28	0.56	1.9	……	red
5	7.4	0.7	0	1.9	……	red
6	7.4	0.66	0	1.8	……	red
7	7.9	0.6	0.06	1.6	……	red
8	7.3	0.65	0	1.2	……	red
9	7.8	0.58	0.02	2	……	red
10	7.5	0.5	0.36	6.1	……	red

而在這筆資料中我們共有兩個目標。第一個目標是按照其他變數的情況判斷一筆資料在類別上是屬於紅酒還是白酒。二是對於紅白兩種酒類，按照其他變數的情況去預測一瓶酒的品質會是幾分。我們將兩目標分述於下。

- **紅白酒分類**

  在紅白酒分類的問題裡，重要的反應變數即是 class。我們利用分類器所得出的最終驗證結果如下圖所示。

```
> winecv
$`class~residual.sugar+chlorides+free.sulfur.dioxide+density+alcohol`
[1] 0.9853734 0.9838462 0.9684615 0.9807840 0.9738663

$`class~fixed.acidity+residual.sugar+total.sulfur.dioxide+sulphates`
[1] 0.9568899 0.9384615 0.9369231 0.9477325 0.9431207

$`class~fixed.acidity+citric.acid+residual.sugar+density+alcohol`
[1] 0.9822941 0.9861538 0.9776923 0.9800154 0.9784781

$`class~residual.sugar+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+quality`
[1] 0.9830639 0.9838462 0.9907692 0.9869331 0.9846272

$`class~volatile.acidity+citric.acid+density+pH+sulphates`
[1] 0.9253272 0.9276923 0.9092308 0.9231360 0.9385088

$`class~total.sulfur.dioxide+density+quality`
[1] 0.9538106 0.9623077 0.9607692 0.9538816 0.9592621
```

圖 4-1:紅白酒分類結果

而我們也能選出一組在 k 折交叉驗證裡準確度平均表現最佳的一組，列出其平均準確度以及在此次交叉驗證裡準確度的變異情況。圖示如下。

```
> winecv[1]
$`class~residual.sugar+chlorides+free.sulfur.dioxide+density+alcohol
[1] 0.9853734 0.9838462 0.9684615 0.9807840 0.9738663

> mean(winecv[[1]])
[1] 0.9784663
> var(winecv[[1]])
[1] 5.081945e-05
```

圖 4-2:紅白酒分類最終變數組合及其表現

- **紅酒品質**

在品質的分類部分，我們的目標是辨識 quality 變數。此變數原本是離散的有序變量，評分從 0 到 10。我們將其分為兩種品質，一種是 0 到 5 分，另一種是 6 到 10 分。同樣的，我們將 GENE.cv 函數所得到的結果列出如下。

```
> redcv
$`quality~volatile.acidity+citric.acid+total.sulfur.dioxide+sulphates+alcohol`
[1] 0.7406250 0.7562500 0.7429467 0.7304075 0.7241379

$`quality~fixed.acidity+volatile.acidity+citric.acid+chlorides+alcohol`
[1] 0.7406250 0.7437500 0.7115987 0.7335423 0.7335423

$`quality~citric.acid+chlorides+sulphates`
[1] 0.6218750 0.6281250 0.6802508 0.6708464 0.6645768

$`quality~volatile.acidity+citric.acid+total.sulfur.dioxide+pH+sulphates+alcohol`
[1] 0.7312500 0.7500000 0.7084639 0.7648903 0.7304075

$`quality~fixed.acidity+citric.acid+chlorides+total.sulfur.dioxide+pH+alcohol`
[1] 0.7031250 0.7218750 0.7115987 0.7053292 0.7115987

$`quality~volatile.acidity+chlorides+total.sulfur.dioxide+density+sulphates`
[1] 0.6968750 0.7500000 0.7147335 0.7021944 0.6708464
```

圖 4-3:紅酒品質分類結果

另外一樣的，我們挑選一組平均表現最佳的組合作為最終使用的模型，並列出其平均表現與變異程度。

```
> redcv[2]
$`quality~fixed.acidity+volatile.acidity+citric.acid+chlorides+alcohol
[1] 0.7406250 0.7437500 0.7115987 0.7335423 0.7335423

> mean(redcv[[2]])
[1] 0.7326117
> var(redcv[[2]])
[1] 0.0001578878
```

圖 4-4: 紅酒品質分類最終變數組合及其表現

- **白酒品質**

白酒的品質大致情況都與紅酒品質分類相同，因此我們直接列出兩項同樣的主要結果如下。

```
> whitecv
$`quality~free.sulfur.dioxide+pH+sulphates`
[1] 0.6642857 0.6578141 0.6874362 0.6462168 0.6707566

$`quality~volatile.acidity+citric.acid+free.sulfur.dioxide+total.sulfur.dioxide+pH+alcohol`
[1] 0.7438776 0.7262513 0.7334014 0.7331288 0.7351738

$`quality~fixed.acidity+volatile.acidity+density+pH+sulphates+alcohol`
[1] 0.7551020 0.7364658 0.7517875 0.7658487 0.7433538

$`quality~citric.acid+residual.sugar+total.sulfur.dioxide+density`
[1] 0.6806122 0.6455567 0.6629213 0.6963190 0.7085890

$`quality~residual.sugar+free.sulfur.dioxide`
[1] 0.6418367 0.6813075 0.6486210 0.6768916 0.6728016

$`quality~total.sulfur.dioxide+density+pH+alcohol`
[1] 0.6938776 0.6945863 0.7201226 0.6789366 0.7085890
```

圖 4-5:白酒品質分類結果

```
> whitecv[3]
$`quality~fixed.acidity+volatile.acidity+density+pH+sulphates+alcohol
[1] 0.7551020 0.7364658 0.7517875 0.7658487 0.7433538

> mean(whitecv[[3]])
[1] 0.7505116
> var(whitecv[[3]])
[1] 0.0001266113
```

圖 4-6: 白酒品質分類最終變數組合及其表現

4.2 資料二

我們的第二筆資料是有關於鮑魚及其乾貨的真實資料。我們的目的在於就其它變數去預測鮑魚的年齡。鮑魚的年齡可以透過其年輪數計算出，而我們在這筆資料中定義 15 或以上算是年長，15 以下為年輕。以下我們簡單列出部分資料的型態以供參考。

表 4-2:鮑魚年齡資料簡示

編號	Sex	Length	Diameter	Height	.....	Rings
1	M	0.455	0.365	0.095	.....	Old
2	M	0.35	0.265	0.09	.....	Young
3	F	0.53	0.42	0.135	.....	Young
4	M	0.44	0.365	0.125	.....	Young
5	I	0.33	0.255	0.08	.....	Young
6	I	0.425	0.3	0.095	.....	Young
7	F	0.53	0.415	0.15	.....	Old

8	F	0.545	0.425	0.125	……	Old
9	M	0.475	0.37	0.125	……	Young
10	F	0.55	0.44	0.15	……	Old

經過整理以後，便能利用我們的分類器進行分類。以下是使用 GENE.cv 做交叉驗證的最終結果。

```
> abalone.result
$`Rings~Shell.weight`
[1] 0.9052369 0.9215442 0.8904110 0.9278607 0.9017413

$`Rings~Sex+Diameter+Height+Shucked.weight+Shell.weight`
[1] 0.9214464 0.9115816 0.9115816 0.9129353 0.9216418

$`Rings~Whole.weight+Shucked.weight+Shell.weight`
[1] 0.9127182 0.9165629 0.9053549 0.9241294 0.9203980

$`Rings~Height+Shucked.weight+Shell.weight`
[1] 0.9102244 0.9165629 0.9165629 0.9154229 0.9191542
```

圖 4-7:鮑魚年齡分類結果

可以觀察到這筆資料有良好的準確度結果。接著，我們一樣列出這幾組可行的變數組合中，平均而言表現最佳的一組及其平均和變異。

```
> abalonecv
$`Rings~Sex+Diameter+Height+Shucked.weight+Shell.weight`
[1] 0.9214464 0.9115816 0.9115816 0.9129353 0.9216418

> mean(abalonecv[[1]])
[1] 0.9158373
> var(abalonecv[[1]])
[1] 2.744947e-05
```

圖 4-8:鮑魚年齡分類最終變數組合及其表現

五、結論與建議

我們的分類方法可以歸納出一些優缺點。它的好處包含了使用概率機制進行迭代維持隨機性、並不用計算所有解以及我們所設定的高度自由性，可以讓使用者依其需求去改動。使用概率機制能確保不以一次計算準確率的結果決定一組變數組合的去留，如此能夠避免遠離真正的最佳解。加上透過變異的設計，我們能夠更放心演算過程不致收斂過快，同時現行組合若全部太過接近，變異的發生更能加入變化，持續引導演算通往最佳解。

然而，擁有這些優點以外，確實也存在一些缺點。一是參數的問題，由於參數多，需要使用者自行去嘗試。雖然我們曾提到過像迭代次數 n 或者進行幾次演算的參數 r 是容易估算或自行選用的，但其他如交配機率，突變機率甚至是過程中排不排序等，都需要多方思量。不同的資料天性不同，想像如果本來差異性就低的資料配合偏低的交配機率，很可能在到達目標迭代次數前就已經急遽收斂了。也有可能因為可能的組合數多或著交配機率偏高，導致另一筆資料在到達目標迭代次數時最佳化僅完成了一半而已。另外就是透過迭代演進的方法，實在很難保證所得到的是全域最佳解，但在決定簡化計算前，我們就已經有此認知，因此也盡力做了相當的措施增加分類器突破區域最佳解桎梏的機會。

未來若還有機會繼續深入研究，我們會考量在分類器裡模型的部分更進一步。也許是加入不同模型比較的機制，或者在使用模型的部分加入一些設計上的巧思使模型結果更能符合分類器的要求等。

參考文獻

[1] James, G., Witten, D., Hastie, T., Tibshirani, R (2013). An introduction to statistical learning with Applications in R. New York: Springer.

[2] Adrian Barbu, Yiyuan She, Liangjing Ding, Gary Gramajo (2013). Feature Selection with Annealing for Regression, Classification and Ranking

[3] Trevor. Hastie, Robert. Tibshirani, & Friedman, J. J. H. (2001).*The elements of statistical learning*(Vol. 1). New York: Springer.

[4]From Wikipedia: Genetic Algorithm
https://en.wikipedia.org/wiki/Genetic_algorithm

[5]
http://jerryyang-wxy.blogspot.tw/search/label/%E5%9F%BA%E5%9B%A0%E6%BC%94%E7%AE%97%E6%B3%95%20%28Genetic%20Algorithm%29

[6]
https://dotblogs.com.tw/dragon229/2013/01/03/86692

附錄

我們將所使用的 R-code 檢附如下：

```
#################################################################
library(MASS)
wineRed=read.csv("C:/winequality-red.csv")
wineWhite=read.csv("C:/winequality-white.csv")
wineRed$class=c(rep("Red",nrow(wineRed)))
wineWhite$class=c(rep("White",nrow(wineWhite)))
wine=rbind(wineRed,wineWhite)
rm(wineRed); rm(wineWhite)
#################################################################
###GENE4
GENE =
function(k=5,n=100,imp=0.5,evo.prob=0.05,sex.prob=0.5,choose.type="unrank",data
=NULL,model.type="lda",glm_family=binomial){
   ldaac = vector( length = k)
   pac = vector("list", k)
   same = c(rep(F,n))
   tfflag = matrix(nrow=length(data)-1,ncol=k)

   m1 = names(data)[length(data)]
   m2 = names(data)[-length(data)]

   for(i in 1:k){
      start = T
      while(start){
      tf = rbinom(length(m2),1,prob = 0.5)
      if(sum(tf)>0){
         start=F
      }
      tf2<-ifelse(tf==1,T,F)
      tfflag[,i] = tf2
      }
   }
   for(iteration in 1:n){
      train = sample(1:nrow(data),round(nrow(data)*4/5))
```

```
    trainWine = data[train,]
    testWine = data[-train,]
    for(i in 1:k){
      choose = m2[tfflag[,i]]
      a = paste(choose,sep = "",collapse = "+")
      b = paste(m1,"~",a,sep = "")
      switch (model.type,
              lda = {
                model = lda(as.formula(b),data= trainWine)
                model_pred = predict(model,testWine)
                ldaac[i]    =
sum(diag(table(model_pred[[1]],testWine[,length(data)])))/length(testWine[,length(
data)])

              },
                glm ={
                levels(trainWine[,length(data)]) = c(0,1)
                levels(testWine[,length(data)]) = c(0,1)

                model = glm(as.formula(b),family = glm_family,data=
trainWine,control = list(maxit = 50))
                model_prob = predict(model,testWine,type="response")
                model_pred =
rep(levels(trainWine[,length(data)])[1],length(model_prob))
                model_pred[model_prob>0.5] =
levels(trainWine[,length(data)])[2]
                ldaac[i]    =
sum(diag(table(model_pred,testWine[,length(data)])))/length(testWine[,length(data)
])

              },
              stop("Wrong model")
      )

      pac[[i]]=choose
      names(ldaac)[i] = b
    }
```

```
    ##選擇

    tempFlag =switch (choose.type,
                                rank =
sample(1:k,size=k,replace=T,prob=(imp*rank(round(1/apply(tfflag,2,sum),digit=4))+(
1-imp)*rank(ldaac))/(sum(imp*rank(round(1/apply(tfflag,2,sum),digit=4)))+(1-imp)*s
um(rank(ldaac))))
                                ,unrank =
sample(1:k,size=k,replace=T,prob=(imp*round(1/apply(tfflag,2,sum),digit=4)+(1-imp)
*ldaac)/(sum(imp*round(1/apply(tfflag,2,sum),digit=4))+(1-imp)*sum(ldaac)))
                                ,{
                                    stop("Wrong choose.type") }
    )
    tfflagChoose = tfflag[,tempFlag]
    ##交配

    sexCOM = combn(1:k,2)[,sample(1:ncol(combn(1:k,2)),replace = F,prob =
rep(1/ncol(combn(1:k,2)),ncol(combn(1:k,2))) )]
    sexTF = sample(c(T,F),size
=ncol(combn(1:k,2)),replace=T,prob=c(sex.prob,1-sex.prob) )
    sexFlag = sexCOM[,sexTF]
    sexPOS =
sample(1:length(m2),size=sum(sexTF),replace=T,prob=rep(1/length(m2),length(m2)))
    if(sum(sexTF>0)){
      for(i in 1:length(sexPOS)){
        if(is.matrix(sexFlag)==T){
          for(j in 0:(length(m2)-sexPOS[i])){

if(tfflagChoose[sexPOS[i]+j,sexFlag[1,i]]!=tfflagChoose[sexPOS[i]+j,sexFlag[2,i]]){
              tfflagChoose[sexPOS[i]+j,sexFlag[1,i]]
= !tfflagChoose[sexPOS[i]+j,sexFlag[1,i]]
              tfflagChoose[sexPOS[i]+j,sexFlag[2,i]]
= !tfflagChoose[sexPOS[i]+j,sexFlag[2,i]]
          }
          }
        }else{
          for(j in 0:(length(m2)-sexPOS[i])){
```

```
if(tfflagChoose[sexPOS[i]+j,sexFlag[1]]!=tfflagChoose[sexPOS[i],sexFlag[2]]){
                    tfflagChoose[sexPOS[i]+j,sexFlag[1]]
= !tfflagChoose[sexPOS[i],sexFlag[2]]
                    tfflagChoose[sexPOS[i]+j,sexFlag[1]]
= !tfflagChoose[sexPOS[i],sexFlag[2]]
                }
                }
            }
          }
      }
      ##相同指標
      if(length(table(ldaac))==1){
         same[iteration]=T
      }
      ##終止

      if(iteration>4){

if(n>100&same[iteration]==T&same[iteration-1]==T&same[iteration-2]&same[iterati
on-3]&same[iteration-4]){
            break
          }
      }


      ##突變
      for(i in 1:ncol(tfflagChoose)){
         evoFlag = rbinom(1,1,prob=evo.prob)
         if(evoFlag==1){

evoPOS=sample(1:nrow(tfflagChoose),1,prob=rep(1/nrow(tfflagChoose),nrow(tfflagC
hoose)))
            tfflagChoose[evoPOS,i] = !tfflagChoose[evoPOS,i]
         }
      }
      ##防 BUG
      for (i in ncol(tfflag) ) {
```

```
    if(sum(tfflagChoose[,i])>0)
      tfflag[,i] = tfflagChoose[,i]
    }

  }
  monkey = list(mod =
ldaac[which.max(imp*rank(round(1/apply(tfflag,2,sum),digit=4))+(1-imp)*rank(ldaac
))],para=pac[which.max(imp*rank(round(1/apply(tfflag,2,sum),digit=4))+(1-imp)*ran
k(ldaac))])
  return(monkey)
}
#####################################################################
GENE4InBox =
function(k=5,n=100,imp=0.5,evo.prob=0.05,sex.prob=0.5,choose.type="unrank",data
=NULL,model.type="lda",glm_family="binomial",r=4){
  re = c()
  pac = vector("list", r)
  for(i in 1:r){
    model =GENE(k=k,n=n,imp = imp,evo.prob =
evo.prob,sex.prob=sex.prob,choose.type = choose.type,data =
data,model.type=model.type,glm_family=glm_family)
    re=c(re,model$mod)
    pac[[i]] = model$para
  }
  ans = list(mod=re,para=pac,data=data)
  class(ans)="GENE"
  return(ans)
}

#####################################################################
##CV

GENE.cv = function(GENE=NULL,k=5){
  a = vector("list",length(GENE$mod))
  names(a)= names(GENE$mod)
  ldaac = vector(length = k)
  for (i in 1:length(GENE$mod)) {
    data2 =GENE$data[sample(1:nrow(GENE$data),nrow(GENE$data)),]
```

```
      dataFlag = vector("list",k)
      for(x in 0:(k-1)){
         dataFlag[[x+1]] =
((round(nrow(GENE$data)/k))*x+1):(round((nrow(GENE$data)/k)*(x+1)))
      }


      for(j in 1:k){
         trainWine = data2[-dataFlag[[j]],]
         testWine = data2[dataFlag[[j]],]
         model = lda(as.formula(names(GENE$mod)[i]),data= trainWine)
         model_pred = predict(model,testWine)
         ldaac[j]
=sum(diag(table(model_pred[[1]],testWine[,length(testWine)])))/length(testWine[,le
ngth(testWine)])


      }
      a[[i]] = ldaac

   }
   return(a)
}

#########################################################################
print.GENE = function(gene){
   print(gene$mod)

}
#########################################################################
simdata <- function(n,p){
   rawsim <- matrix(0,n,p)
   for(i in 1:n){
      for(j in 1:p){
         rawsim[i,j]=mvrnorm(n,c(rep(1,p)),Sigma=diag(p))[i,j]
      }
   }
   return(rawsim)
```

```
}
trial1 <- as.data.frame(simdata(1000,5))
Y<-exp(0.004+1.5*trial1[,1]-0.6*trial1[,2]+0.000000007*trial1[,3])/(1+exp(0.004+1.5
*trial1[,1]-0.6*trial1[,2]+0.000000007*trial1[,3]))
Yprob=c()
for(i in 1:nrow(trial1)){
   Yprob[i]=sample(c(0,1),size=1,prob=c(1-Y[i],Y[i]))
}
class=as.factor(ifelse(Yprob==1,"big","small"))
trial1$resp=class
#####################################################################
abalone=read.csv("C:/abalone.csv")
abalone$Rings=ifelse(abalone$Rings>=15,"Old","Young")
GENE.cv(GENE4InBox(data=abalone))

abmean=c()
for(i in 1:4){
   abmean[i]=mean(abalone.result[[i]])
}
abmean
abalone.result=GENE.cv(GENE4InBox(data=abalone))
abalonecv=abalone.result[which.max(abmean)]
mean(abalonecv[[1]])
var(abalonecv[[1]])
```