Machine Learning HW3

0510894 電機 4D 翁紹恩

一、 Gaussian Process for Regression

1、 實驗步驟：

a、 Without ADR

i、 Calculate $\mathrm{k}(x_n, x_m) = \theta_0 \exp\left(-\frac{\theta_1}{2}\|x_n - x_m\|^2\right) + \theta_2 + \theta_3 {x_n}^T x_m$

ii、 Calculate $\mathrm{C}(x_n, x_m) = \mathrm{k}(x_n, x_m) + \beta^{-1}\delta_{nm}$

iii、 Calculate $\mathrm{m}(x_{N+1}) = \mathrm{k}^T C^{-1} t$

iv、 Calculate $\sigma^2(x_{N+1}) = \mathrm{C} - \mathrm{k}^T C^{-1} k$

將 testing set 資料作為x_m，training set 資料為x_n算出預測的平均和變異數，即可得出結果。

b、 With ARD

i、 Initialize θ with arbitrary value(事實上初始值參數若設定的不好，ARD 也無法算出良好的結果，可能會 overfitting 等等，因此仍要靠人工 trial and error，若是 overfitting 則改挑選其他值作為初始值)

ii、 Calculate $\mathrm{k}(x_n, x_m) = \theta_0 \exp\left(-\frac{\theta_1}{2}\|x_n - x_m\|^2\right) + \theta_2 + \theta_3 {x_n}^T x_m$

iii、 Calculate $\mathrm{C}(x_n, x_m) = \mathrm{k}(x_n, x_m) + \beta^{-1}\delta_{nm}$

iv、 Calculate $\frac{\partial \mathrm{C}}{\partial \theta_0}$ 、$\frac{\partial \mathrm{C}}{\partial \theta_1}$ 、$\frac{\partial \mathrm{C}}{\partial \theta_2}$ 、$\frac{\partial \mathrm{C}}{\partial \theta_3}$

$$\frac{\partial \mathrm{C}}{\partial \theta_0} = \exp\left(-\frac{\theta_1}{2}\|x_n - x_m\|^2\right)$$

$$\frac{\partial \mathrm{C}}{\partial \theta_1} = \theta_0 \times (-\frac{1}{2}\|x_n - x_m\|^2)\exp\left(-\frac{\theta_1}{2}\|x_n - x_m\|^2\right)$$

$$\frac{\partial \mathrm{C}}{\partial \theta_2} = 1$$

$$\frac{\partial \mathrm{C}}{\partial \theta_3} = {x_n}^T x_m$$

v、 Calculate $\frac{\partial \ln P(t|\theta)}{\partial \theta_i} = -\frac{1}{2} \times \mathrm{Tr}\left(C^{-1}\frac{\partial \mathrm{C}}{\partial \theta_i}\right) + \frac{1}{2}\mathrm{t}^T C^{-1}\frac{\partial \mathrm{C}}{\partial \theta_i}C^{-1}t$ for four theta

vi、 Calculate $\theta_{new} = \theta_{old} + \eta \frac{\partial \ln P(t|\theta)}{\partial \theta}$，$\eta = 0.0001$(η is one of hyperparameter, here set η as 0.0001)

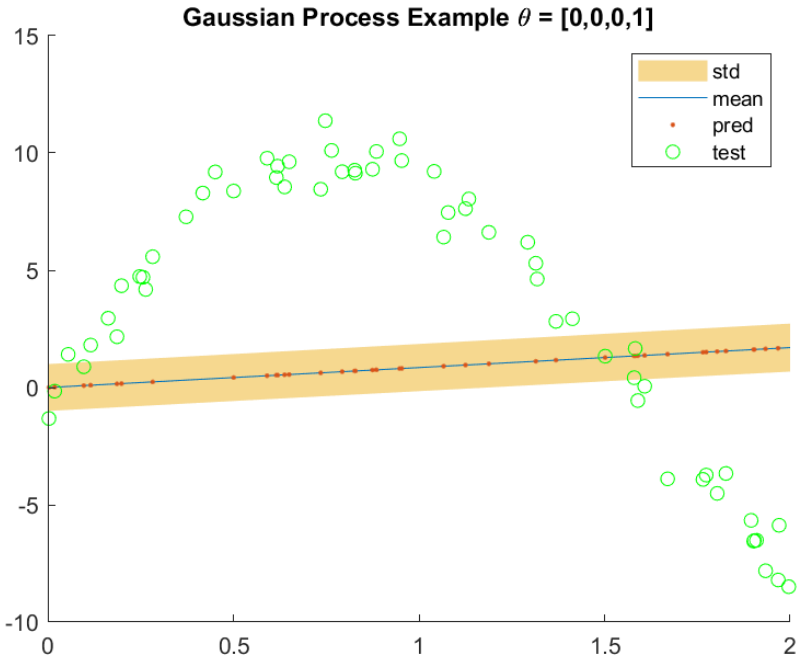
vii、 持續做到收斂為止，並接續一、1、a 步驟。

2、 問題與討論：

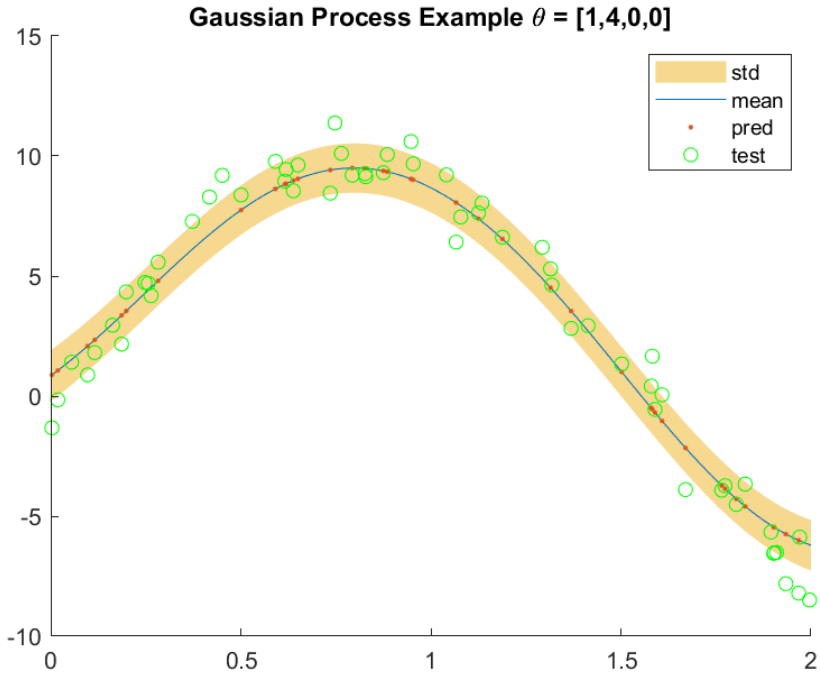
a、 The prediction result

i、

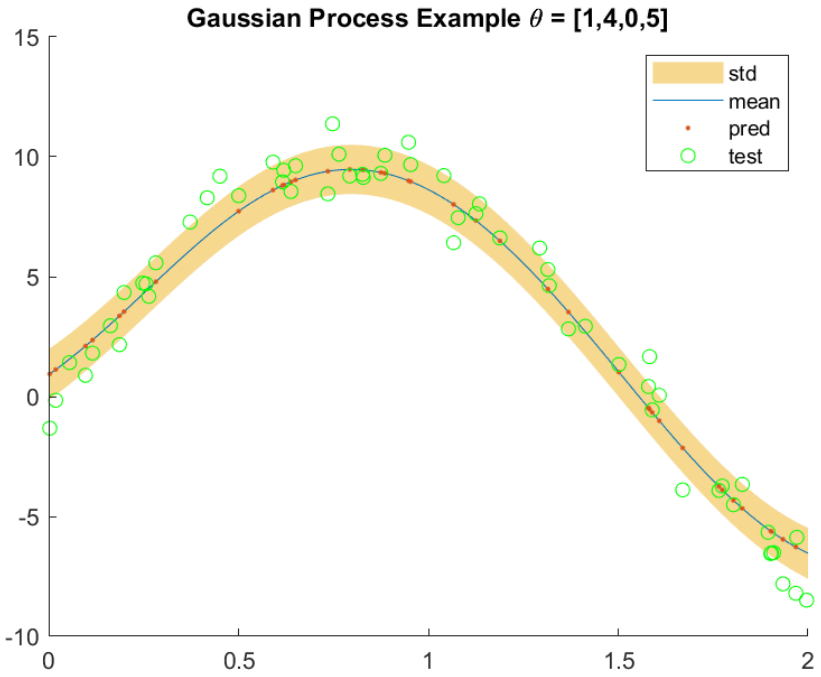
θ =[0,0,0,1]



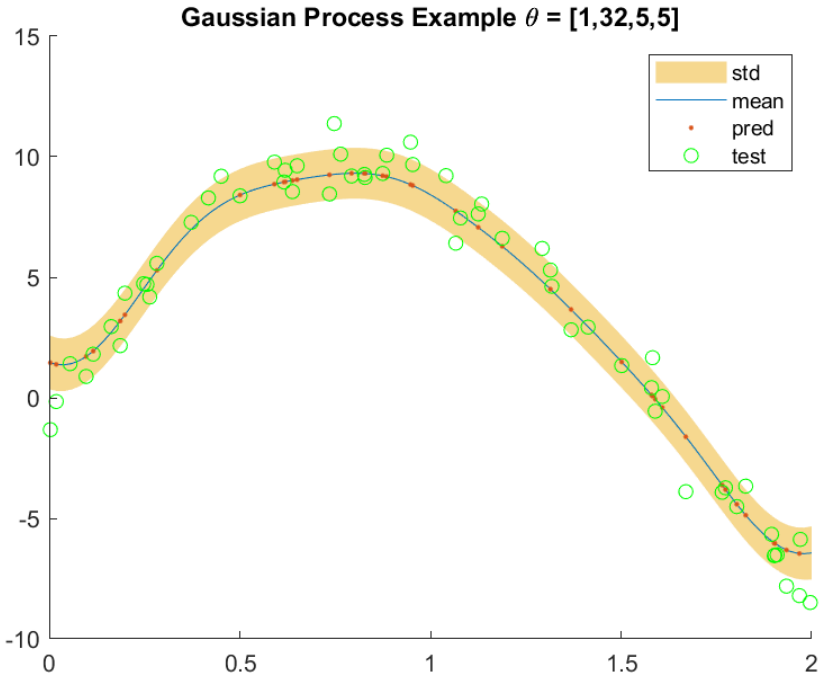
θ =[1,4,0,0]



θ =[1,4,0,5]



θ =[1,32,5,5]



由以上圖形可以發現當θ_1變大的時候，整個圖形的準確度提高，而只剩θ_3的時候預測結果變成線性，因此精準度低。

b、 The corresponding root-mean-square errors

	Training	Testing
θ =[0,0,0,1]	6.6575	6.5502
θ =[1,4,0,0]	1.0522	1.0518
θ =[1,4,0,5]	1.0288	1.0288
θ =[1,32,5,5]	0.9640	0.9856

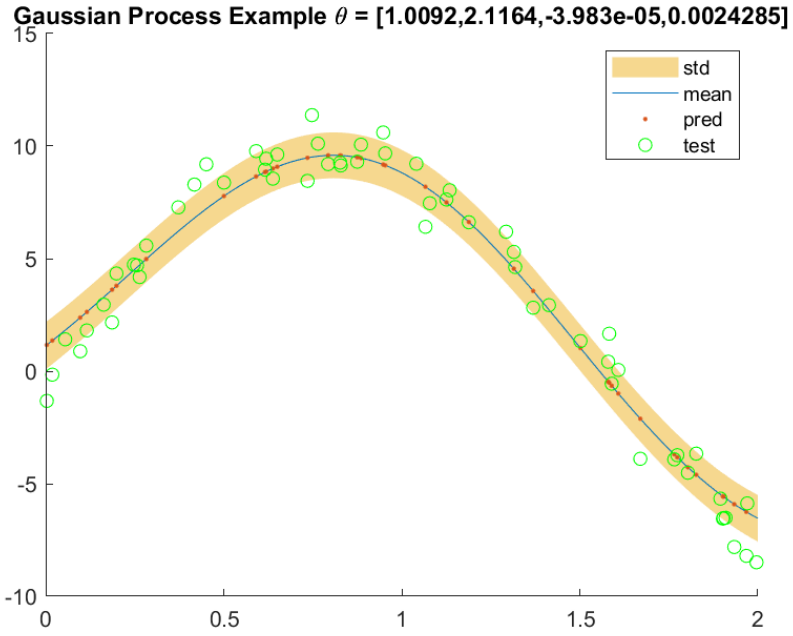
可以從 θ=[1,4,0,0]和 θ=[1,4,0,5]兩組數據觀察發現，似乎θ_3對數據的影響並不大。

c、 Tune the hyperparameters by **ARD**

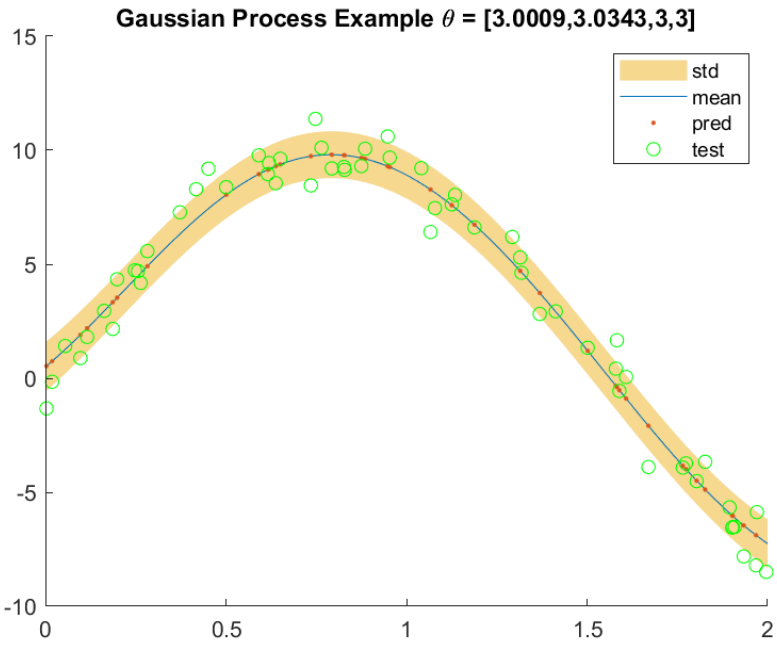
	Training	Testing
Initial θ=[1,2,0,0]	1.0397	1.0642
Initial θ=[3,3,3,3]	0.8926	0.9162
Initial θ=[2,2,2,2]	0.9471	0.9722
Initial θ =[10,10,10,10]	0.7890	0.8031
Initial θ =[1,10,1,1]	0.9470	0.9374
Initial θ =[1,100,1,1]	1.1300	1.1649

※紅字為 error 最小，藍字為比原指定四個參數 error 還要小的參數組合。

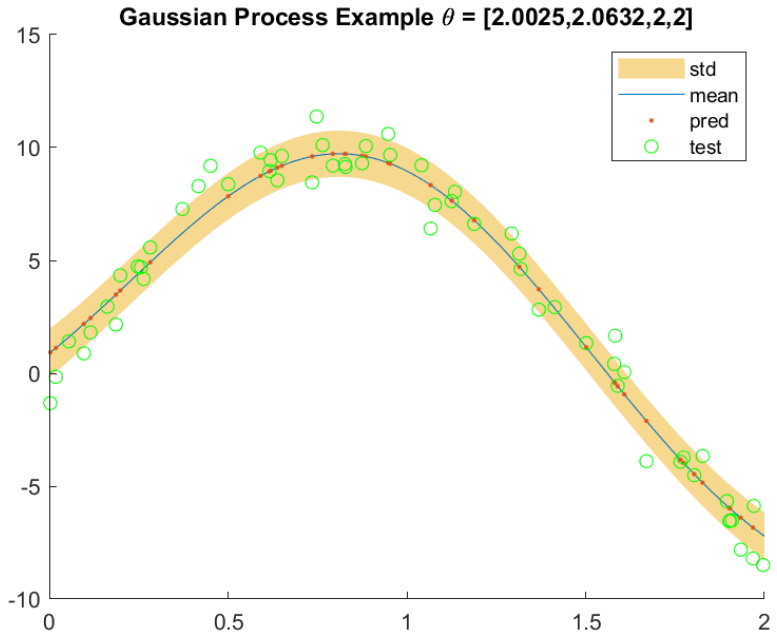
Initial θ=[1,2,0,0]



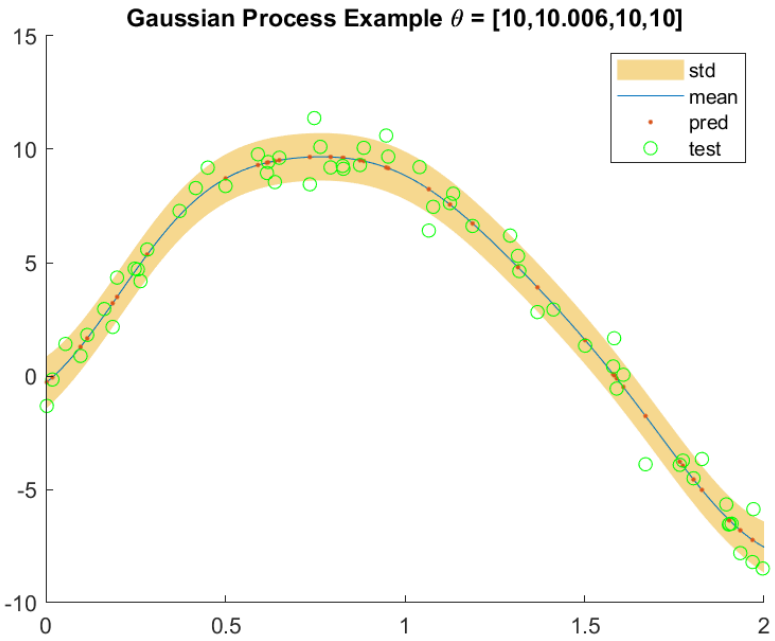
Initial θ=[3,3,3,3]



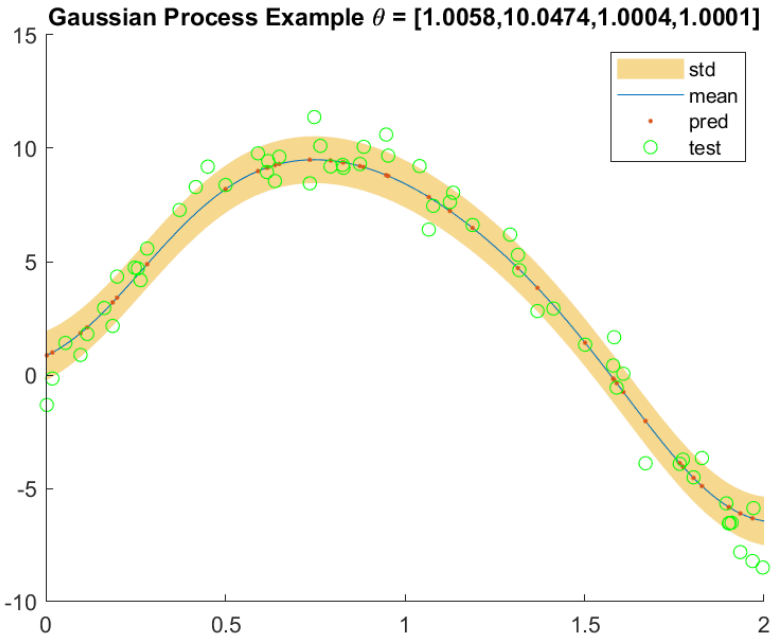
Initial θ=[2,2,2,2]



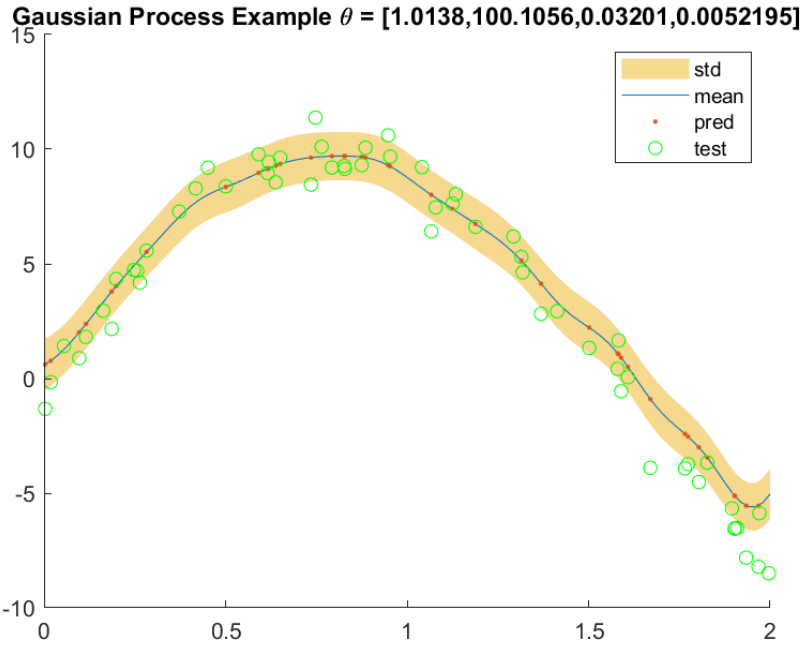
Initial θ=[10,10,10,10]



Initial θ=[1,10,1,1]



Initial θ=[1,100,1,1]



可以發現當θ_1太大(Initial θ=[1,100,1,1])的時候，整個曲線開始 overfitting，但是其他三個數值不用設太大(Initial θ=[1,10,1,1])即可得出不錯的結果，稍微略調θ_1以外的三個數值可以得出不錯的結果(Initial θ=[10,10,10,10] ➔θ=[10,10.006,10,10]是嘗試的幾組數據中 error 最低的。)

d、 Discussion

使用 ARD 可以找出數值精確度較高的參數，但是還是需要自己設定初始參數，若設定不好則容易 overfitting，因為在此題 error 並無差很多，因此會傾向使用 trial and error 的方法，並且從觀察數據可以發現，θ_1為影響最多的參數，θ_0至少要為 1，而θ_2、θ_3影響不大，但是略微調整θ_0、θ_2、θ_3可使得 error 降低一些(由 Initial θ=[10,10,10,10]和 Initial θ=[1,10,1,1]比較)。

二、 Support Vector Machine

1、 實驗步驟：

a、 先對資料進行作業二做過的 PCA 降維

b、 分別做不同 kernel 的 SVM

i、 Lineral kernel

$k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$，$\varphi(x_i) = x_i$

ii、 Polynomial (homogeneous) kernel of degree 2

$k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$，$\varphi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$、$x = [x_1, x_2]$

c、 代入 smo()，計算 α 和 bias

d、 $w = (\alpha .* t) * \varphi$

e、 $y = \varphi * w^T + bias$

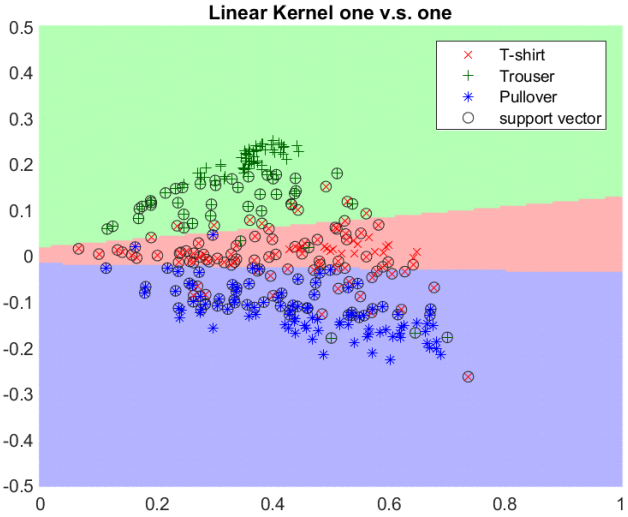
f、 其中 $\alpha > 0$ 的向量為 support vector

g、 分別根據不同組和“one-versus-one”、“one-versus-all”得出最後的預測值

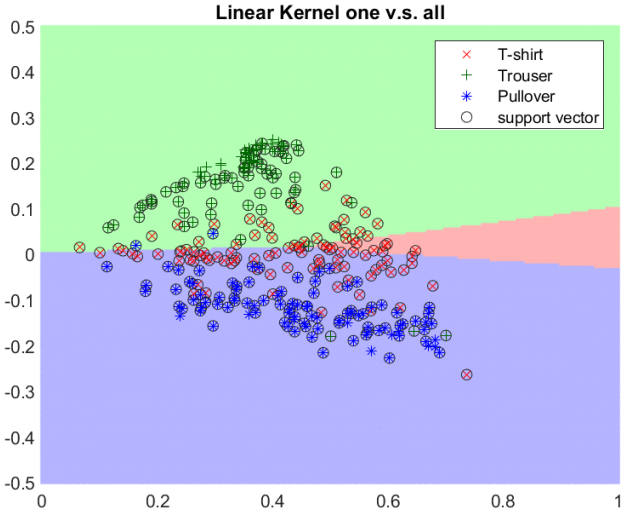
2、 問題與討論：

a、 Analyze the difference between two decision approaches, one is “one-versus-the-rest”, and another is “one-versus-one”.

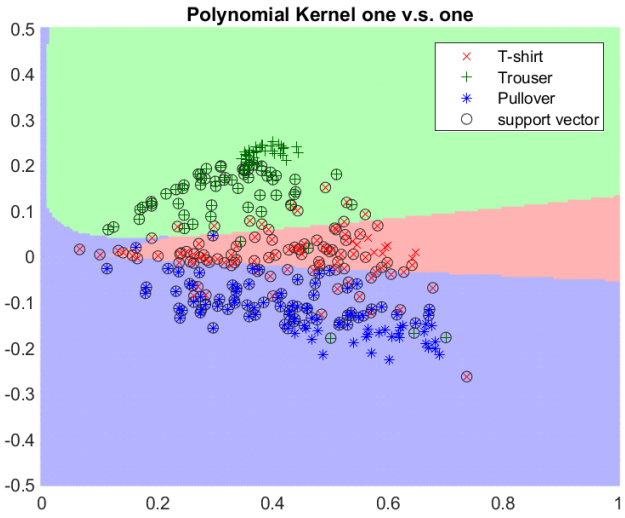
Linear kernel one versus one



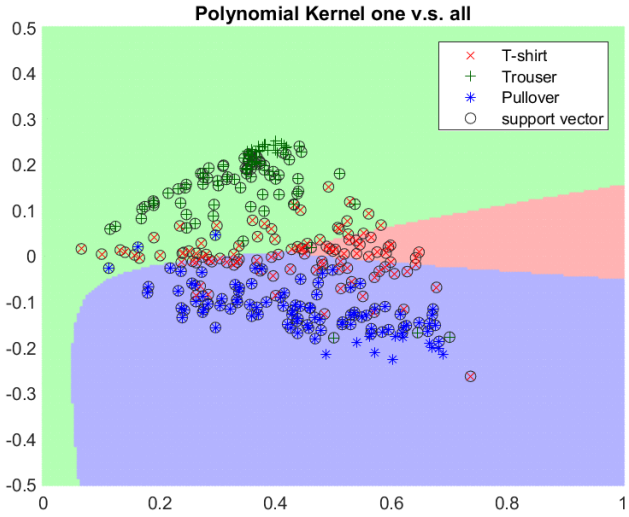
Linear kernel one versus all



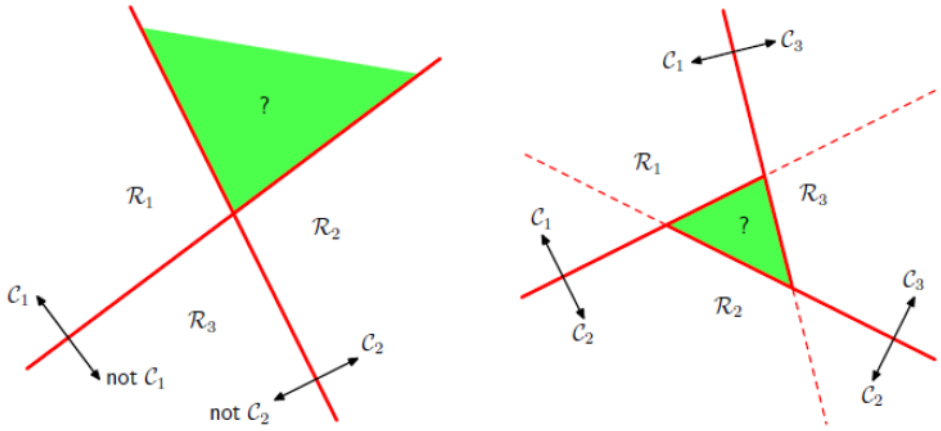
Polynomial kernel one versus one



Polynomial kernel one versus all



由上述四張圖可以發現，不論是 Linear kernel 還是 Polynomial kernel 都是 one versus one 做出來的預測較為精準，one versus one 的方法使用的是兩兩比大小，再用最判斷出的區域分類別，而 one versus all 是使用三者直接比大小取大的，雖然 one versus one 實作較為麻煩複雜，但可能是因為經過多次比較，預測考量較多，所以準確率相對也較高，會產生問題的區域也較小，如下圖所示：



One-verus-the-rest classifier

One-verus-one classifier

b、 SVM with linear kernel versus SVM with polynomial kernel (degree = 2)

由上圖可以發現 linear kernel 的邊界為線性的，而 polynomial kernel 的為曲線，因此當分布有彎曲變化的時候用 polynomial kernel 預測較為準確，可以避免 linear kernel 一刀兩斷、硬性分割區域的問題。

三、 Gaussian Mixture Model(only do k-means part)

1、 實驗步驟：

a、 任意設定 μ 的初始值

b、 計算 x 和 μ 之間的距離，取得 k 個類別中最小的，設定$\gamma_{nk} = 1$，其餘為 0

c、 得到新的 γ 後利用$\mu_k{}^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}\,x_n$重新計算 μ ，$N_k = \sum_{n=1}^{N}\gamma_{nk}$

d、 在此設定 iterate 100 次後即跳出

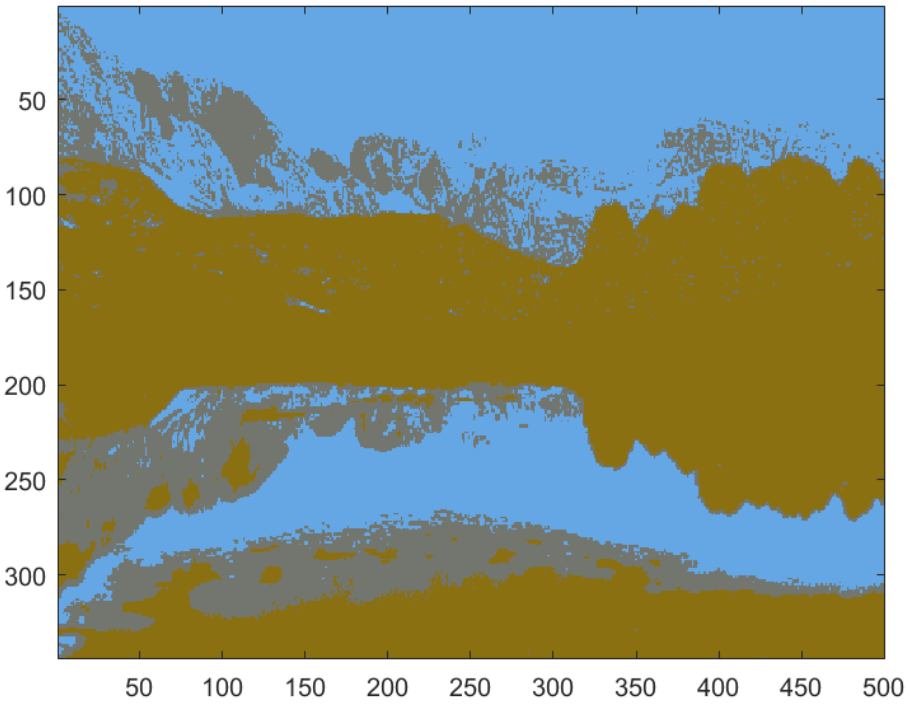
e、 x 和新的 μ 計算，取距離最短的 μ 為新的數值，產生新的圖片。

2、 問題與討論：

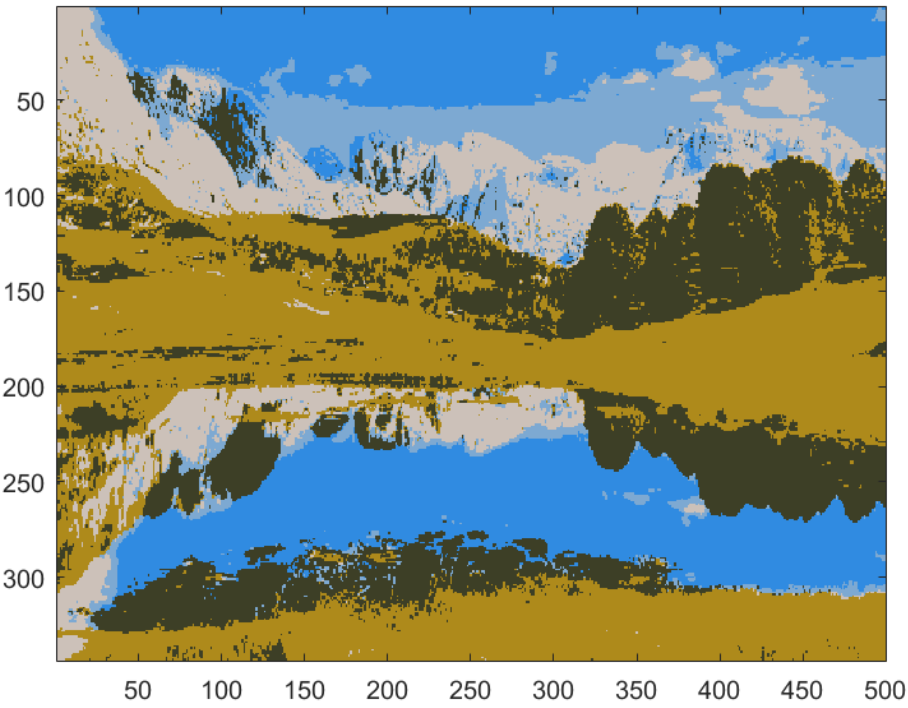
a、 Result

i、 Picture

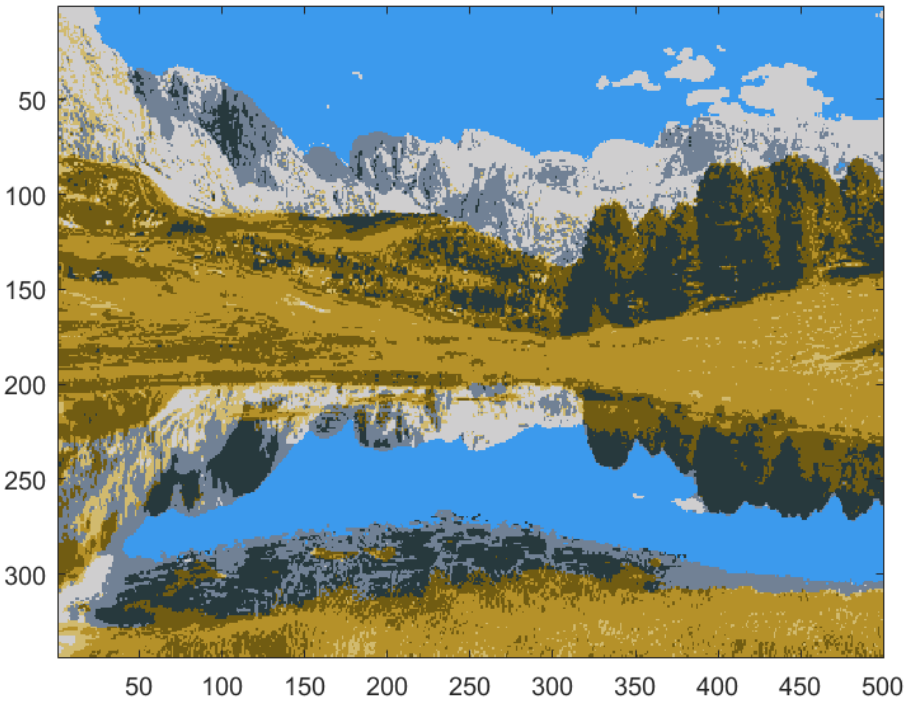
K=3



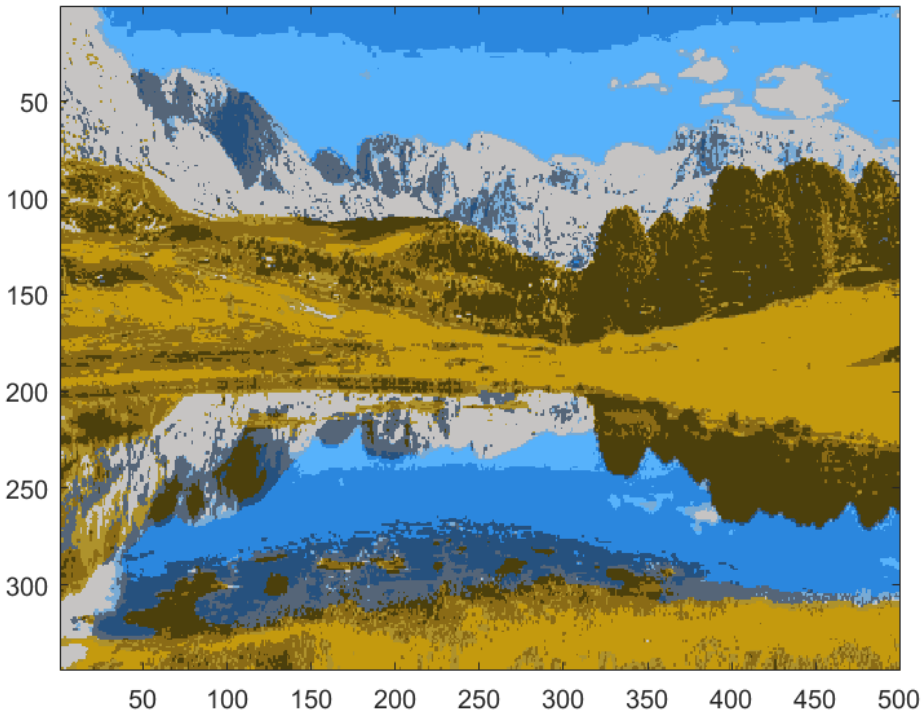
K=5



K=7



K=10



ii、 Mu table

K=3

0.4107	0.6153	0.8110
0.1642	0.5089	0.8396
0.5495	0.4634	0.1785

K=5

0.7187	0.5864	0.1946
0.6034	0.6106	0.6236
0.8363	0.8324	0.8209
0.2335	0.5907	0.9098
0.2534	0.2606	0.1599

K=7

0.6934	0.5528	0.0989
0.8684	0.8502	0.8396
0.2602	0.2430	0.1058
0.5050	0.4967	0.4977
0.6934	0.6737	0.6661
0.1247	0.4671	0.7831
0.2994	0.6598	0.9727

K=10

0.5421	0.4184	0.0882
0.4747	0.6715	0.8357
0.6841	0.5733	0.1727
0.7762	0.7689	0.7642
0.3315	0.3972	0.4717
0.1701	0.5275	0.8701
0.7698	0.6042	0.0539
0.1485	0.3157	0.4926
0.2986	0.2512	0.0478
0.3420	0.6970	0.9847
0.5421	0.4184	0.0882
0.4747	0.6715	0.8357
0.6841	0.5733	0.1727

b、 Discussion

可以看到當 k 愈大，圖片和原本的愈相近，因為種類愈多，愈可以區分如光影色彩等，一開始計算時即對圖形做 normalize，因此數值皆為在 0~1 之間，而非 0~255；因為使用 k-means 會選擇最接近輸入資料的 mean 因此在多次遞迴更新後可以取得適當的 mean 畫出一個大致的圖，而本次實驗大約使用 k=5 以上就可以產生一張和原圖相似的圖片。