Machine Learning HW3

0560206 蔡孝謙

1.(a)(hw3_1_a.m)

Algorithm

[Training Phase]

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}.$$

先用Training data的kernel算出CN(右式)

一開始我設定beta = 0.01，theta = [(1/beta)^2;(1/beta)^2]，ita = [1/beta;1/beta];

後來根據助教的設定beta = 100，theta = [1;0.5]，ita = [1;1];

[Testing Phase]

Regression 出來的結果是 k' * inv(C_N) * t(右式)

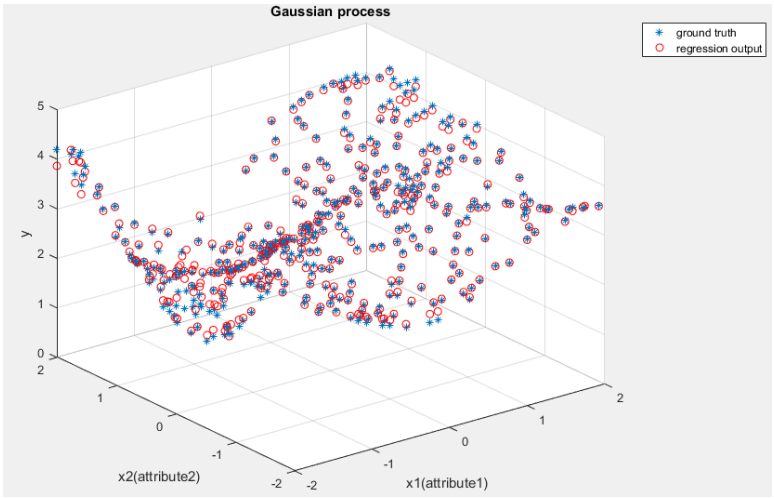
$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t}$$

這邊的k 是[k(x_1,x_{N+1}); k(x_2,x_{N+1})...; k(x_N,x_{N+1})]

Result

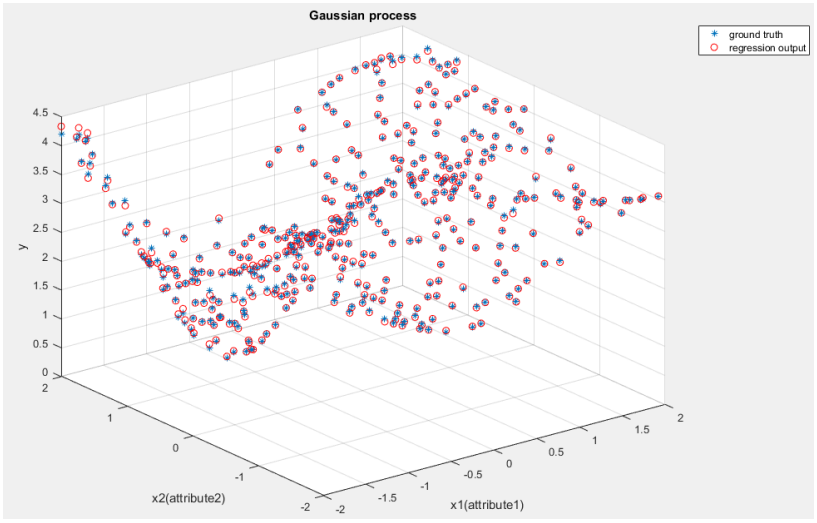
[By my parameter]

Erms = 0.1436



[By suggestion parameter]

Erms = 0.0387



1.(b)(hw3_1_b.m)

Algorithm

[Training Phase]

我設定mu = 0.01，theta = [(1/mu)^2;(1/mu)^2]，ita = [1/mu;1/mu];

根據助教給的演算法，先算出a_N，但是中間會遇到一個inv(I+WC)，而這個矩陣是6000*6000的，所以運算時間會很長

我設定跳出迴圈的條件是，a_{new} - a_{old} 的Erms<0.6 就進入testing Phase，但這樣還是要進行8次iteration，訓練時間超長(約150秒)…

=====12/29更新 training data====

因為training data變少所以訓練時間變短了

[Testing Phase]

Regression 出來的結果是 sigmoid(k' * inv(C_N) * t)

因為sigmoid出來的解在0和1之間，所以我們看上面那個式子出來的結果，如果>0.5的分為1，反之分為0

這邊的k 是[k(x_1,x_{N+1}); k(x_2,x_{N+1})…; k(x_N,x_{N+1})]

Result

[Old training data](6000 training data and 591 testing data)

[By my parameter]

Accuracy = 0.8576

Known \ Predict	0	1	accuracy
0	484	52	91.3%
1	32	22	40.7%
accuracy	93.7%	29.7%	85.7%

[By suggestion parameter]

Accuracy = 0.9034

Known \ Predict	0	1	accuracy
0	527	9	98%
1	48	6	11%
accuracy	91.6%	40%	90.3%

[New training data](2000 training data and 720 testing data)

因為training data變少且testing data變多，所以正確率變低了

[By suggestion parameter]

Accuracy = 0.7306

Known \ Predict	0	1	accuracy
0	255	113	69.2%
1	81	271	76.9%
accuracy	75.8%	70.5%	73.0%

2.(One-versus-one – Linear kernel / Polynomial kernel)(hw3_2_linear.m, hw3_2_poly.m)

Algorithm

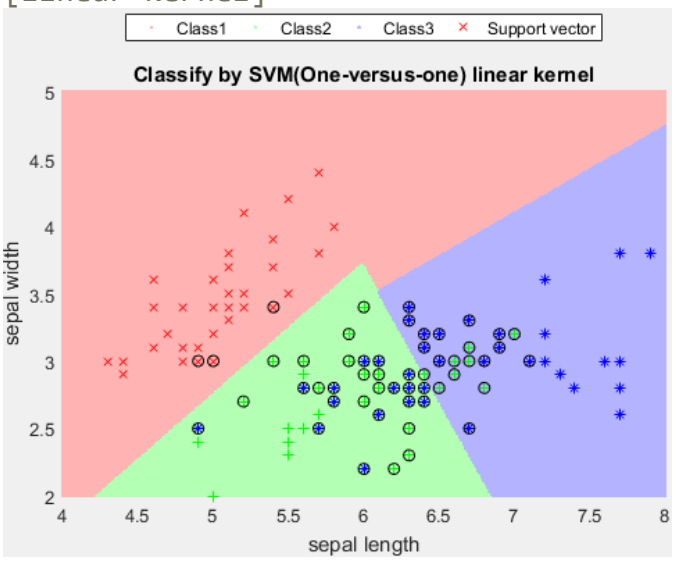
[Other file]

1. svm.m
   - input是phi_x, t, C, tol，在中間會進行SMO，值得注意的是input是phi_x，所以在主要的檔案中要先把training data轉成phi的feature space，再用svm去train
   - output是train出來的w, b, support vector，support vector是所有a>0的training data
2. sigmoid
   - 就是一般的sigmoid function: 1./(1+exp(-a));

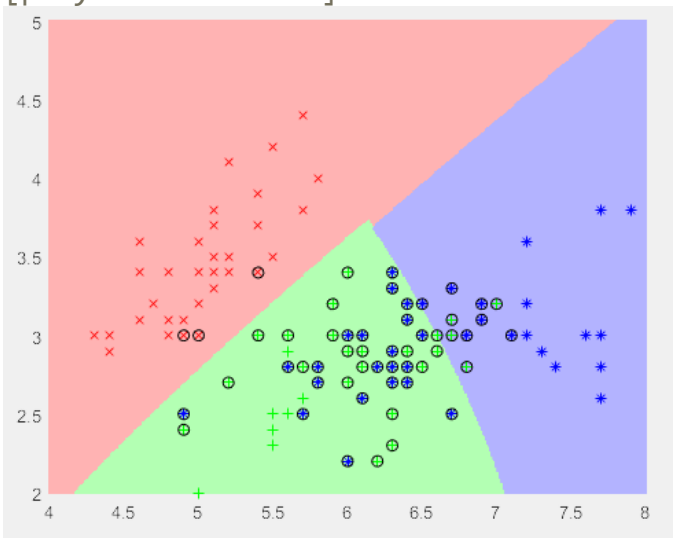
[Training Phase]
一開始要先做把training data轉換成svm要的feature，因為linear kernel的feature就是原本的資料，所以不用動，不過polynomial kernel就要把[x1,x2]轉成三維的vector才能進行svm
因為是One-versus-one，所以我們要做3次svm，分別是1 vs 2, 2 vs 3, 1 vs 3，並給相對應的t，t = {-1 , 1}
接下來我們要把每個training data用這三個classifier分類(根據y是否>0來判斷)，並且Voting，最後把分類的結果算出來
[Testing Phase]
在Testing Phase就直接用SVM算出來的y，再去voting是哪一個class，出來就是結果了

Result

[linear kernel]



[polynomial kernel]

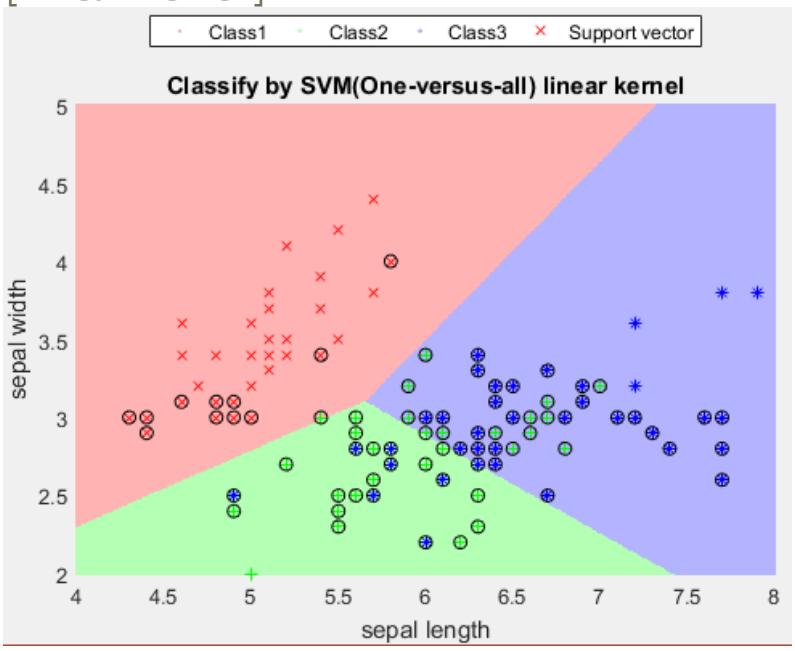


P.S
背景要怎麼著色花了一點時間去想，後來跟Lab的同學討論我們是用linspace將x軸和y軸各產生N=300個點(總共有N*N個點)，再把每個點去做分類，分類完在畫圖，才有這個結果!

從結果可以看出來分類的結果蠻合理的，在boundary附近的training data是support vector，不過class 2和class 3有很多overlapping，所以support vector就多很多

2.(One-versus-rest – Linear kernel / Polynomial kernel)
(file: hw3_2_one_to_all_linear.m, hw3_2_one_to_all_poly.m)
其實稍微改一下code就可以變成one-versus-rest了
主要是把svm的kernel變成120筆data(one-versus-one是80筆data)，40筆t = 1，另外80筆t = -1，就可以算出w_1,w_2,w_3,b_1,b_2,b_3，也能夠進行test
結果圖如下:

[linear kernel]



[polynomial kernel]