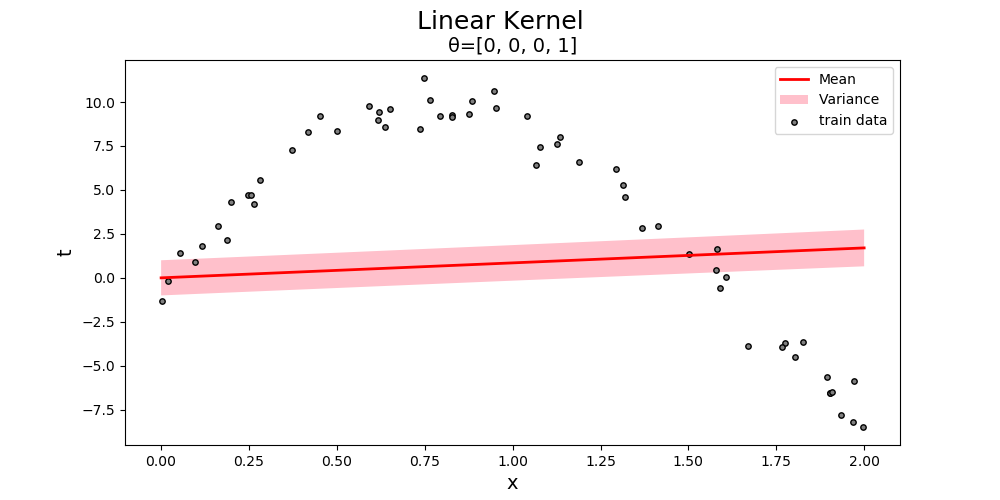
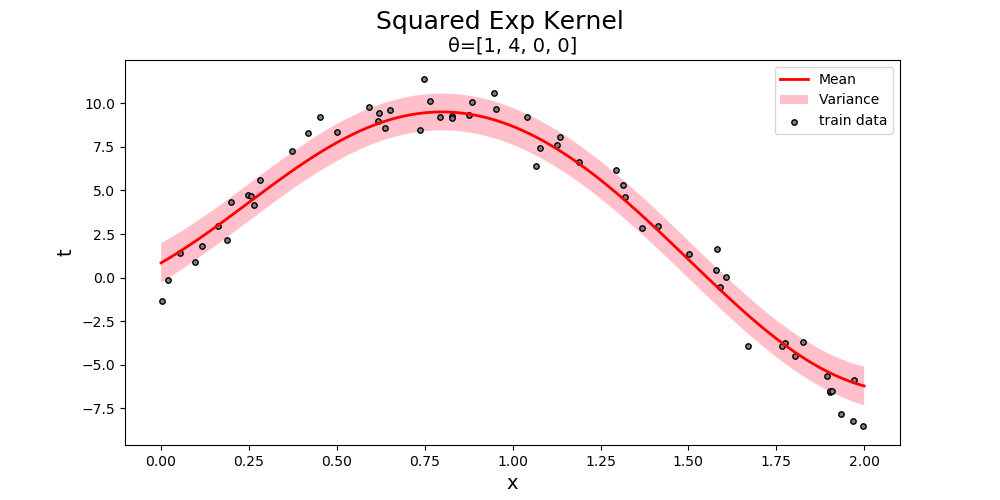
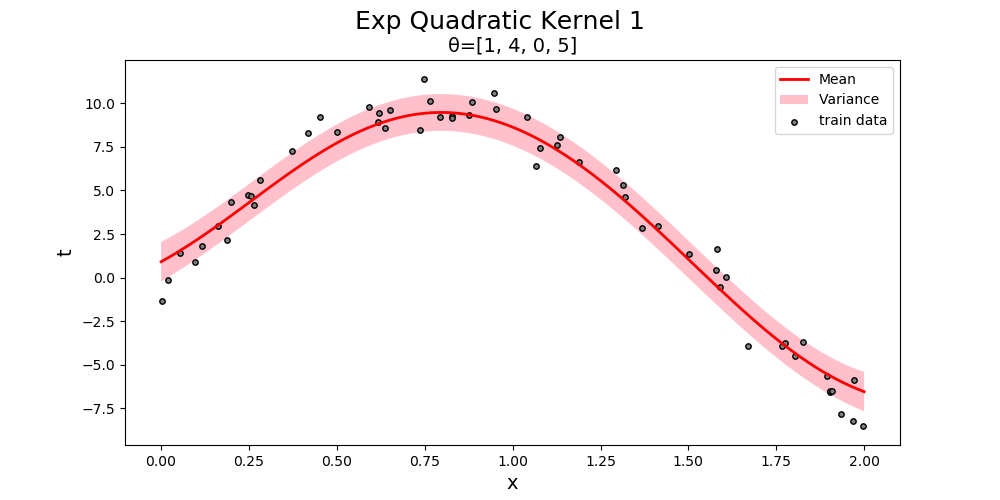
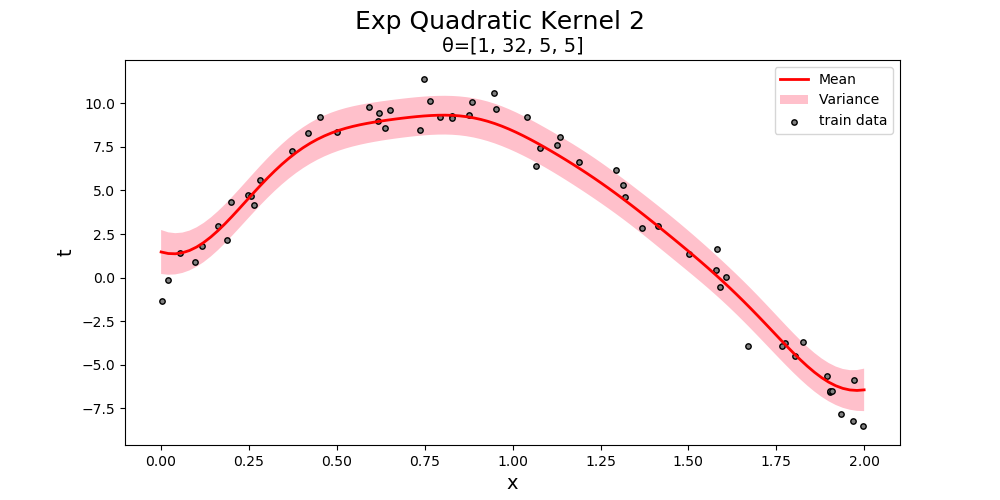
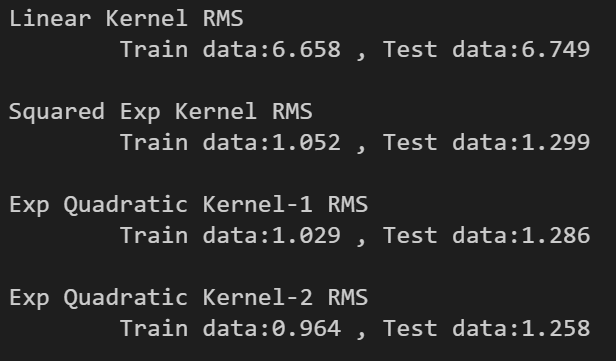
ML Hw3 Report

電機乙 0850736 楊登宇

1. Gaussian Process for Regression

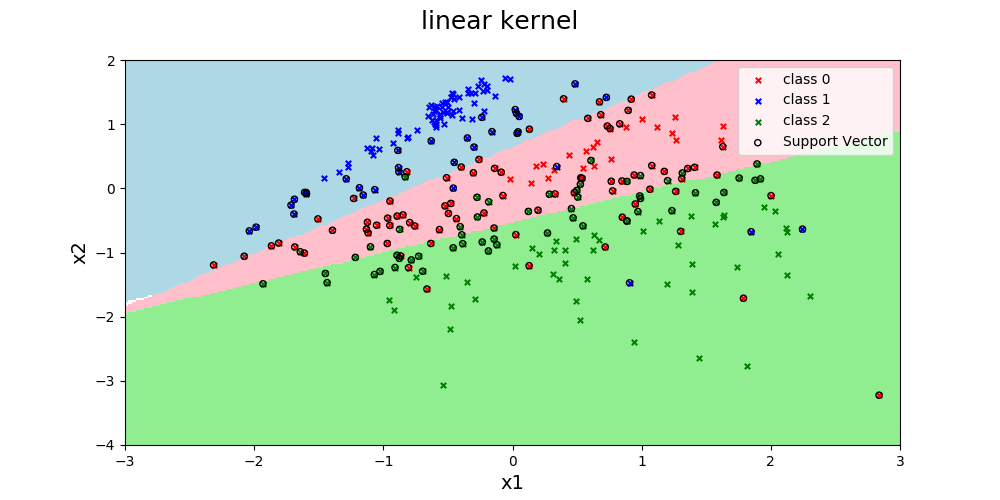
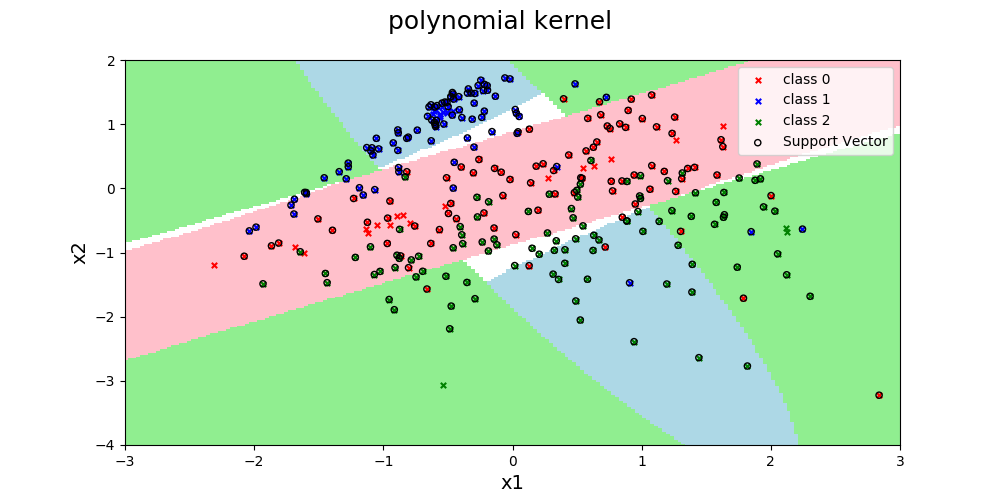
這一題在練習使用kernel的方法，一開始在寫的時候一直不太懂所謂的kernel到底有甚麼好處，但實作後才懂原來使用kernel可以減少一些對於超參數的假設與計算（雖然最後還是需要去調整），讓整體計算平均值與變異數的過程更加直覺。

其中，關於平均值與變異數的計算方法如下：

而結果可以清楚看見，透過去調整，可以讓kernel所代表的意義有所變化，而此作業所使用的

訓練資料來看，使用Exponential Quadratic Kernel的參數最可以貼近該組資料，其中第二組在訓練資料的RMS可以到達最低值0.964。但是在使用測試資料來計算誤差時會發現跟第一組差別不大，我懷疑可能是因為第二組的參數有overfitting的趨勢，使得測試資料RMS下降程度不像使用訓練資料來做RMS一樣的大。也有可能是這已經是Exponential Quadratic Kernel參數對該題測試資料所能影響的最大範圍了。

1. Support Vector Machine

第二題我使用python的scikit-learn套件來求出該題SVM所要計算的Lagrange Multipliers與支援向量（已跟助教確認過可以直接求得）。在這一題支援向量機的求法，我使用的是one versus one的方法來實作。其中關於我選擇方法的原因如下；

One vs. one：可一次分類多類別的資料（multiclass），假設有n個類別，可以一次找出其中一個類別與其他(n-1)類別的邊界y(x)，這樣子可以一次就訓練好模型，在預測的時候比較方便，所以我是選擇使用此方式。

One vs. the rest：一次只會判斷是不是屬於該類別，一次只會計算出一條邊界y(x)。假設有n個類別，就需要訓練n次才能進行後續的預測，因為需要耗費的計算量我覺得比較大，因此我並沒有選擇此方法。

在得到multipliers後，根據以下的公式：

可以求出三條決策邊界，我這邊計算出來的邊界定義如下：

接著，可以帶入x來計算出在這三條邊界上的正負來判斷是屬於哪一類別：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | - | + | + | - | - | + | - |
|  | + | + | - | + | - | + | - | - |
|  | + | + | + | - | + | - | - | - |
| **class** | 0 | 1 | X | 0 | 1 | X | 2 | 2 |

其中會發現，會出現三種皆為無法分類的情形，在我的實作中我會將分不出哪一類的情形畫成白色底，用以跟其他類別作出區分。

實作部分我是先將原本的資料進行PCA降成2維後，再進行標準化。因為若沒有進行標準化的話，每個樣本點之間的離散程度會滿大的，在畫圖時的邊界會差距過大，所以為了繪圖與分析資料方便，我線進行標準化讓資料漂亮一點。

接著，將資料透過scikit-learn來求出參數後以求得，接著繪圖。

在線性部分的結果，分起來的效果看起來還算不錯，且幾乎沒有分不出來的部分（白色底）；而2次多項式的結果看起來就有明顯差異了，其中更多出了許多白色底的無法分類的區域，而且分類的情況也沒有線性的理想。我推測是因為此題應該使用到線性kernel即可，使用到多項式反而有點過頭，才導致分類反而會分不好。

1. Gaussian Mixture Model
2. kmeans only

|  |  |
| --- | --- |
| K=3 | K=5 |
| K=7 | K=10 |

我將每個pixel的rgb顏色取出來，然後進行kmeans來做初步分類。程式在運作時，會發現在K=5,7時，計算次數會比較久一點，反而k=10還比較快，這讓我感到蠻驚訝的。我猜測可能與kmeans一開始取初始中心點有關係。我本身的取法是把所有資料平均取k個點，有可能我在k=10取的點比較好，所以收斂的速度較為快速。

其中各個不同k最後得到的中心點與數量結果我都額外存在3\_kmeans\_output.txt檔案中，可以透過以下公式，來將kmeans求出的k點中心來做為GMM的初值，進行後續EM計算：

比較麻煩的點是，我的是三維的向量，所以便藝術的部分是一個矩陣，後續在GMM中還需要進行一些矩陣的反矩陣與行列式，會消耗掉比較多計算資源。

1. GMM with kmeans initialization

|  |  |
| --- | --- |
| K=3 | K=5 |
| K=7 | K=10 |

根據kmeans所求得的k個中心點來做為GMM的起始條件，接著使用以下的EM方法來做計算：

E step：

M step：

透過不斷更新，我們可以得到GMM後的新群集中心。比較特別的是GMM考量的都是機率，不像Kmeans是直接將資料分給特定群體中心，GMM討論的是在每個類別中可能的機率。

1. Likelihood

|  |  |
| --- | --- |
| K=3 | K=5 |
| K=7 | K=10 |

因為我的輸入資料是三維資料(rgb三個顏色)，一直找不到可以畫出4D空間方法。因此，我將原本的顏色rgb先算出likelihood的值後，再將rgb轉成灰階值，藉此畫出來。

所以會看到相同灰階值會有不同的區間值，是因為同一個灰階值有可能來自不同的rgb，所以才會得到此結果。