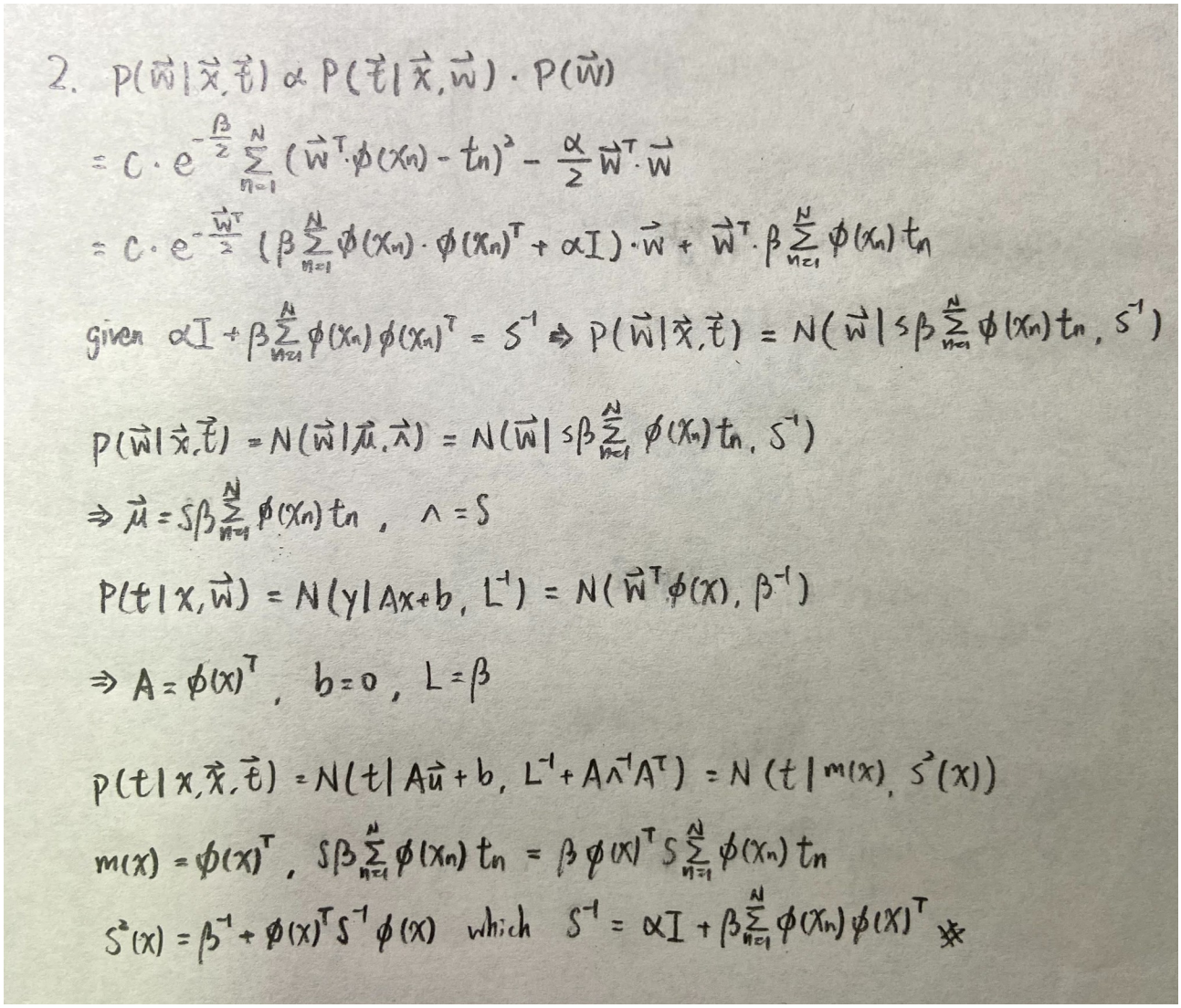
1. Bayesian Linear Regression

1.因為 basis function 是最常見的數學模型，透過這些模型做 linear regression 能夠讓我們在預測上或設計新的數學模型時更有依據與參考價值。



3.可以，但是效果會比起使用邏輯回歸來做分類還差，因為它的輸出是連續的，不是概率的。在二元分類問題中，我們感興趣的是結果發生的可能性，概率介於 0 到 1 之間，但是在線性回歸中，我們預測的是絕對數，其範圍可能在 0 和 1 之外。

2. Linear Regression

1. Feature select

(a)首先將所有 data 跟對應的 target 分成兩部份，分別是 training 以及 testing 的，這邊我取前面 450 筆當 training data，最後面 50 筆當 testing data。接著透過公式(1)找出 M=1 和 M=2 各別的 phi matrix，再利用公式(2)找出各別的 weight，最後用公式(3)算出 RMS error。

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j \quad (M = 2) \text{ ---(1)}$$

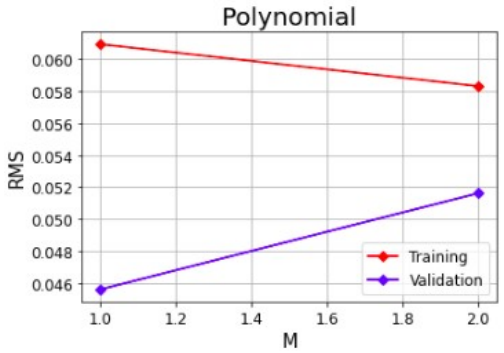
$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \text{ ---(2)}$$

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}. \text{ ---(3)}$$

- Result & Discuss

  從 RMS 的結果以及圖可以看出來，當階數(M)變高的時候，training data 的 RMS 確實有因為 fit 更好而下降，但卻造成 testing data 有 overfitting 的現象，也就是 RMS 上升。

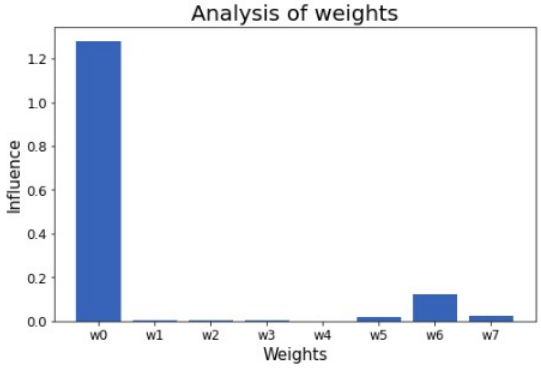
```
1.(a)
Polynomial

RMS_M1_train:   0.06095143237851235
RMS_M1_test:    0.04559177564490297

RMS_M2_train:   0.05832880424300963
RMS_M2_test:    0.05160960927185425
```

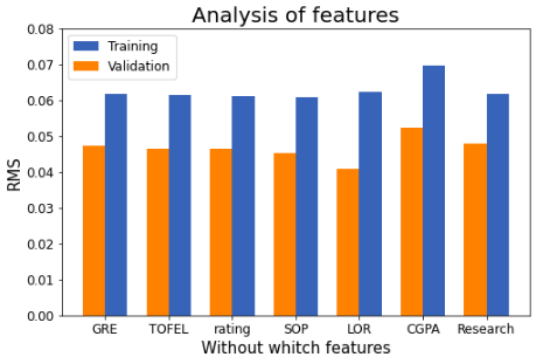


(b)為了看出哪個 feature 對整個 model 的影響最大，可以透過兩種方式來分析。第一，算出 model M=1 時的所有 weight，並觀察 weight 的大小。第二，將每個 feature 依序從 data 中拿掉，並觀察 RMS 的變化。

- Result & Discuss

  由下方左圖可以看出各個 weight 的大小，也就是對應 feature 的影響程度，但因為 w0 不對應到任何 feature，所以我們不將它考慮在內，因此由圖可看出 feature 6 (CPGA)對整個 model 的影響最大。接著，由下方右圖可以看出在去掉哪個 feature 時所計算出的各個 RMS，而當在去掉 CGPA 時，可以發現其 training 跟 testing 的 RMS 皆最大，因此可以得知其對整個的 model 影響最大。





2. Maximum likelihood approach

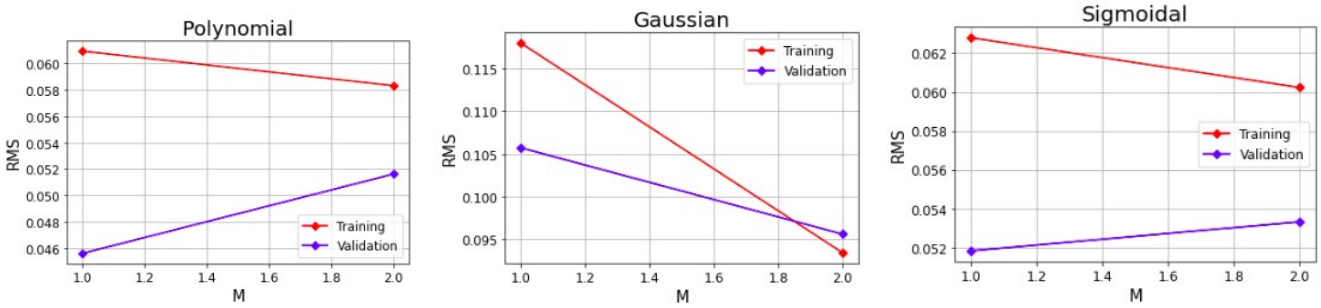
(a)直接透過 Polynomial、Gaussian 以及 Sigmoidal 三種 basis function 去計算各自在 M=1 和 M=2 時 training 與 testing 的 RMS，發現使用 Polynomial 的效果最好(RMS 最小)，因此我選擇使用 Polynomial 來改善我的 regression model。

- Result & Discuss

```
Polynomial                                Gaussian                                  Sigmoidal

RMS_M1_train:  0.06095143237851235        RMS_M1_train:  0.11796033218956312        RMS_M1_train:  0.06278908522875996
RMS_M1_test:   0.04559177564490297        RMS_M1_test:   0.10570638491253403        RMS_M1_test:   0.05182633693994323

RMS_M2_train:  0.05832880424300963        RMS_M2_train:  0.09344113230064592        RMS_M2_train:  0.060233039149453146
RMS_M2_test:   0.05160960927185425        RMS_M2_test:   0.09561352963742394        RMS_M2_test:   0.053313789580317336
```

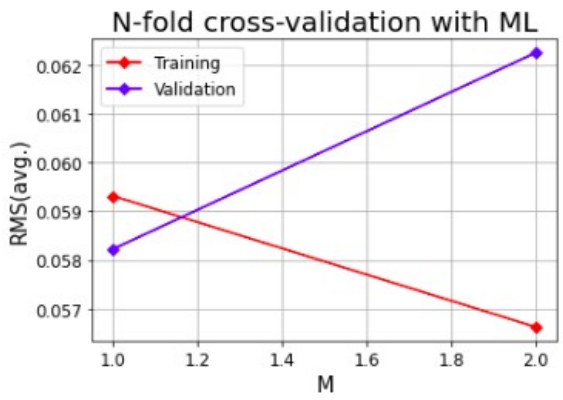
(b)我選用的 Polynomial basis function 就是公式(1)，雖然它的效果在上述三者之中最好的(RMS 最小)，但是當階數變高時，testing data 會出現 over fitting 的情況。

- Result & Discuss
  從下面三張圖可以發現 Polynomial 跟 Sigmoidal 在高階時會出現 over fitting，而 Polynomial over fitting 的程度較大。Gaussian 在高階時 under fitting 的效果不管是對 training 還是 testing 都很顯著，但缺點就是其 RMS 相較其他兩者最差。



(c)我選用階數來當作 hyper-parameter 去跟 1.(a)做有無使用 N-fold 的比較。方法是將所有 data 跟對應的 target 分成 N 個子集合(這邊我設 N=10)，並取任一組當作 testing data，剩餘的 N-1 組當作我的 training data 去計算 RMS，並重複做 N 遍，直到每個子集合都當過 testing data 為止。

- Result & Discuss
  以下是利用 Polynomial 經過 N-fold 後的結果，這邊將各別算出來的 N 組 RMS 平均以呈現更接近實際的情形。可以看到透過 N-fold 後，階數變高依然能夠有效下降 training data 的 RMS，但是 testing data 的 over fitting 依然不會因為 N-fold 後而減少。




```
N-fold cross-validation with ML

RMS_M1_train :
mean:      0.0593113767871043
variance:  8.1094424689400 44e-06

RMS_M1_test :
mean:      0.058225498477515604
variance:  0.00043942534086194394

RMS_M2_train :
mean:      0.05662403816652206
variance:  6.603056004833078e-06

RMS_M2_test :
mean:      0.062252808785435244
variance:  0.0004380016477467137
```

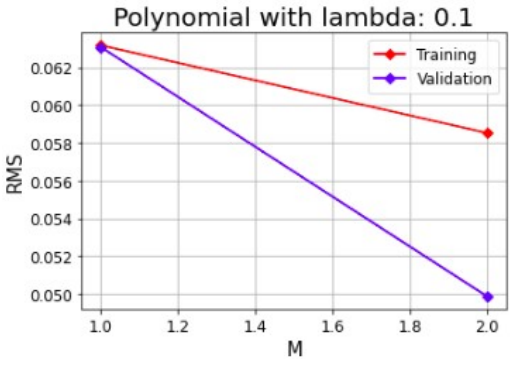
3. Maximum a posterior approach

(a)MAP 和 ML 最大的差異在於 lambda 的加入，將原本的 weight function 由公式(2)變成公式(4)，目的是為了能夠在高階時有效降低 over fitting 的現象。

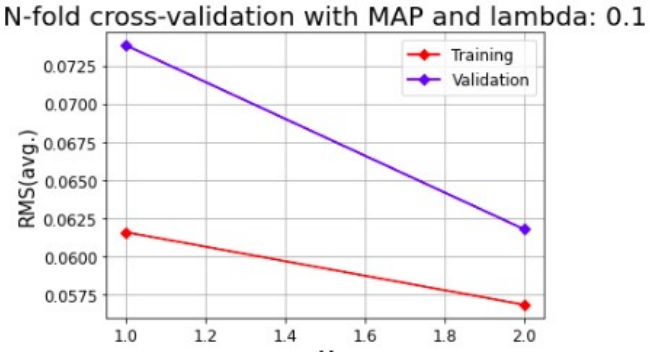
$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \text{---(4)}$$

(b)利用 MAP 並選用 lambda=0.1 將第二題重做一遍後，結果如下：

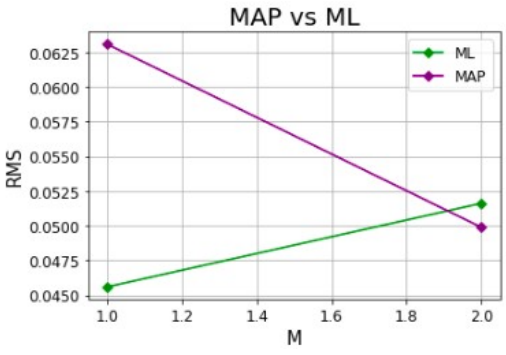
```
Polynomial
lambda =  0.1

RMS_M1_train_with_lambda:  0.06316570857304941
RMS_M1_test_with_lambda:   0.06306933971315273

RMS_M2_train_with_lambda:  0.05852795272158788
RMS_M2_test_with_lambda:   0.049883038573420996
```

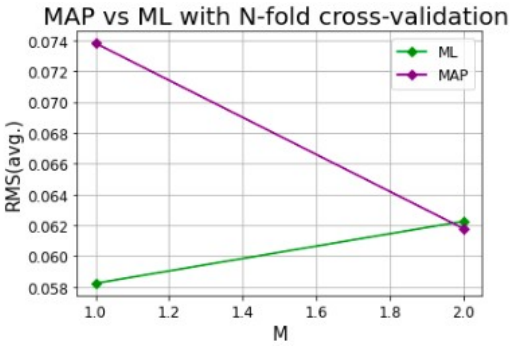



```
N-fold cross-validation with MAP

RMS_M1_train :
mean:      0.061576845913199896
variance:  8.102784571076458e-06

RMS_M1_test :
mean:      0.07381409987728829
variance:  0.00031440976018294863

RMS_M2_train :
mean:      0.05683214259208362
variance:  6.556640630596912e-06

RMS_M2_test :
mean:      0.06177471083894624
variance:  0.0004062184103747651
```



(c)由以下比較圖可以發現，不管是有無使用 N-fold，加入 lambda 的 MAP 均能有效下降 testing data 在高階時的 RMS，從原本向上趨勢的 over fitting 變成向下趨勢的 under fitting。





- Result & Discuss

  上面兩張是單純比較 testing 的 RMS 所畫出來的圖，雖然高階時可以有效降低 RMS 以達到 under fitting，但有趣的是因為加入了 lambda，因此在 M=1 時，MAP 所算出來的 RMS 會較 ML 的高。