Machine Learning (Homework 1)

電子所 311510061 陳柏翰

1 Bayesian Linear Regression

$(\vec{x}, \vec{t})$ are training data ; (x, t) are new test point

$$p(t|x, \vec{x}, \vec{t}) = \int_{-\infty}^{\infty} p(t, \vec{w} | \vec{x}, \vec{t}) d\vec{w}$$

$$= \int_{-\infty}^{\infty} p(t|\vec{w}, x, \vec{x}, \vec{t}) p(\vec{w}|x, \vec{x}, \vec{t}) d\vec{w}$$

$$= \int_{-\infty}^{\infty} p(t|x, \vec{w}) \, p(\vec{w}|\vec{x}, \vec{t}) d\vec{w}$$

from $p(\vec{w}|\vec{x}, \vec{t}) \propto p(\vec{t}|\vec{x}, \vec{w}) p(\vec{w})$

且 $p(t|x, \vec{w}, \beta) = N(t|y(x, \vec{w}), \beta^{-1}) = N(t|\vec{w}^T\phi(x), \beta^{-1})$

$$p(\vec{w}|\alpha) = N(\vec{w}|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{\frac{M+1}{2}} \cdot e^{-\frac{\alpha}{2}\vec{w}^T\vec{w}}$$

$$\Rightarrow p(\vec{w}|\vec{x}, \vec{t}) = k \cdot e^{-\frac{\beta}{2}\sum_{n=1}^{N}(\vec{w}^T\phi(x_n) - t_n)^2 - \frac{\alpha}{2}\vec{w}^T w} \quad (k = const.)$$

$$= k \cdot e^{-\frac{\vec{w}^T}{2}\left(\beta\sum_{n=1}^{N}\phi(x_n)\cdot\phi(x_n^T) + \alpha I\right)\vec{w} + \vec{w}^T\beta\sum_{n=1}^{N}\phi(x_n)t_n}$$

題目已知 $S^{-1} = \alpha I + \beta \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T$

$$\Rightarrow P(\vec{w} \mid \vec{x}, \vec{t}) = k \cdot e^{-\frac{1}{2}\vec{w}^T S^{-1}\vec{w} + \vec{w}^T \beta \sum_{n=1}^{N} \phi(x_n) t_n}$$

$$= N\left(S\beta \sum_{n=1}^{N} \phi(x_n) t_n ,\ S\right)$$

by equations from pg. 93

$$\begin{cases} P(x) = N(x \mid \mu, \Lambda^{-1}) ,\quad P(y \mid x) = N(y \mid Ax + b, L^{-1}) \\ P(y) = N(y \mid A\mu + b, L^{-1} + A\Lambda^{-1}A^T) \\ P(x \mid y) = N(x \mid \Sigma\{A^T L(y-b) + \Lambda\mu\}, \Sigma),\ \Sigma = (\Lambda + A^T L A)^{-1} \end{cases}$$

$$\Rightarrow ① P(\vec{w} \mid \vec{x}, \vec{t}) = N\left(S\beta \sum_{n=1}^{N} \phi(x_n) t_n ,\ S\right) = N(\vec{w} \mid \vec{\mu}, \vec{\Lambda}^{-1})$$

we get $\vec{\mu} = S\beta \sum_{n=1}^{N} \phi(x_n) t_n$ and $S = \vec{\Lambda}^{-1}$

$$② P(t \mid x, \vec{w}) = N(t \mid A\vec{w} + b, L^{-1}) = N(t \mid \vec{w}^T \phi(x), \beta^{-1})$$

we get $A = \phi(x)^T$, $b = 0$, $L = \beta$

$$③ P(t \mid x, \vec{x}, \vec{t}) = N(t \mid A\mu + b, L^{-1} + A\Lambda^{-1}A^T) = N(t \mid m(x), s^2(x))$$

$$\Rightarrow m(x) = A\mu + b = \beta \phi(x)^T S \sum_{n=1}^{N} \phi(x_n) t_n$$

and $s^2(x) = L^{-1} + A\Lambda^{-1}A^T = \beta^{-1} + \phi(x)^T S \phi(x)$

where $S^{-1} = \alpha I + \beta \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^T$ 得証 #

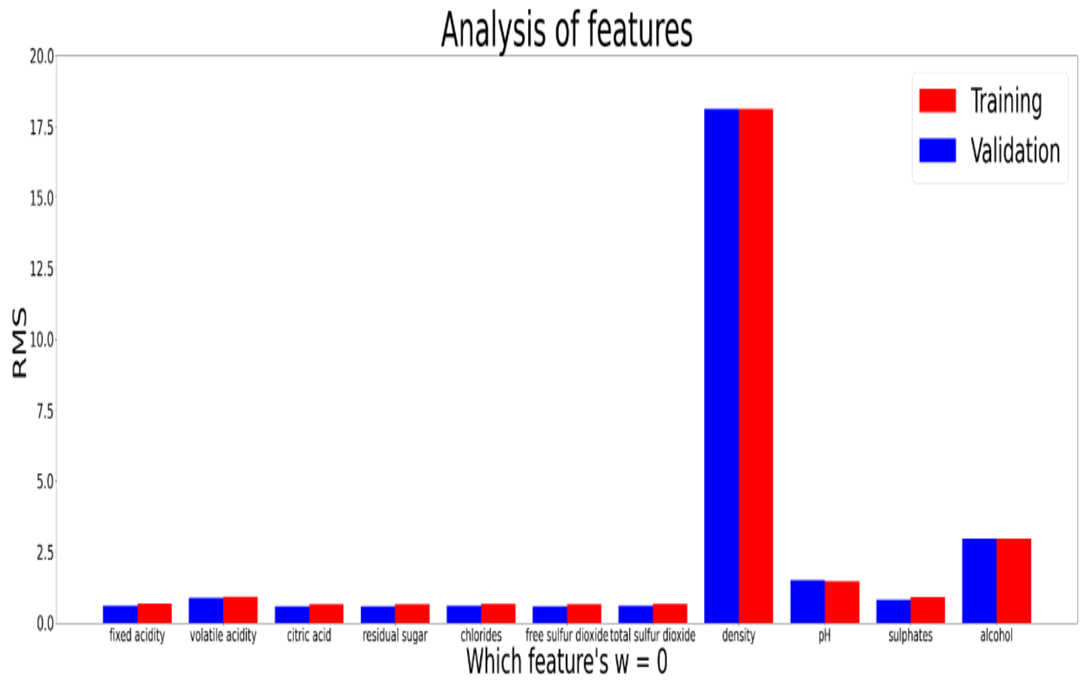
2 Linear Regression

2.1 Feature selection

2.1(b)

要找出 weight 中影響最大的 feature 大概有兩個方法，(1)由於 M = 1 是線性函數，可以透過觀察 weight 中各項數值的大小來判斷，但是可能會因為 data 的大小不同而出錯 (2)因此透過將各個 feature 的 w 分別設為 0，這樣可以觀察 w*feature value 的值對 RMS error 所造成的影響，使 RMS error 劇烈上升的即為 contributive value，我將以第二種作分析。

```
fixed acidity weight=0:
RMS_train:  [0.69102306]
RMS_validation:  [0.60047125]

volatile acidity weight=0:
RMS_train:  [0.90683545]
RMS_validation:  [0.87107072]

citric acid weight=0:
RMS_train:  [0.6539666]
RMS_validation:  [0.58238378]

residual sugar weight=0:
RMS_train:  [0.6510944]
RMS_validation:  [0.57989009]

chlorides weight=0:
RMS_train:  [0.67394036]
RMS_validation:  [0.60161653]

free sulfur dioxide weight=0:
RMS_train:  [0.65514664]
RMS_validation:  [0.5822055]

total sulfur dioxide weight=0:
RMS_train:  [0.67488678]
RMS_validation:  [0.60403608]

density weight=0:
RMS_train:  [18.12152547]
RMS_validation:  [18.11711781]

pH weight=0:
RMS_train:  [1.47476665]
RMS_validation:  [1.49349494]

sulphates weight=0:
RMS_train:  [0.88883226]
RMS_validation:  [0.81356367]

alcohol weight=0:
RMS_train:  [2.97004235]
RMS_validation:  [2.97827644]
```



可以從算出的結果和作出來的圖看出 density 這個 feature 對 RMS 的影響特別大，因此 density 即為 contributve feature。

2.2 Maximum likelihood approach

2.2(a)

我分別使用三種 Basis Function 去進行 M = 1 和 M = 2 時 RMS 的計算，發現用 Polynomial 當 Basis Function 的效果稍微好一些，RMS 的值稍微小一些，雖然 Polynomial 和 Sigmoid 的 RMS 非常接近，但 Polynomial 對 validation 的效果稍微好一點，因此我選擇使用 Polynomial 當作 Basis Function。

以下為各種 Basis Function 在 M = 1 和 M = 2 時的 RMS。

```
Polynomial M = 1:
RMS_M1_train:  [0.64078163]
M1_train_accuracy:  81.08903605592347 %

RMS_M1_validation:  [0.67766329]
M1_validation_accuracy:  78.75 %
```

```
Polynomial M = 2:
RMS_M2_train:  [0.59846099]
M2_train_accuracy:  84.03237674760854 %

RMS_M2_validation:  [0.70396571]
M2_validation_accuracy:  78.33333333333333 %
```

```
Gaussian M = 1:
RMS_Gaussian_M1_train:  [0.77212188]
M1_Gaussian_train_accuracy:  79.10228108903605 %

RMS_Gaussian_M1_validation:  [0.8393091]
M1_Gaussian_validation_accuracy:  73.75 %
```

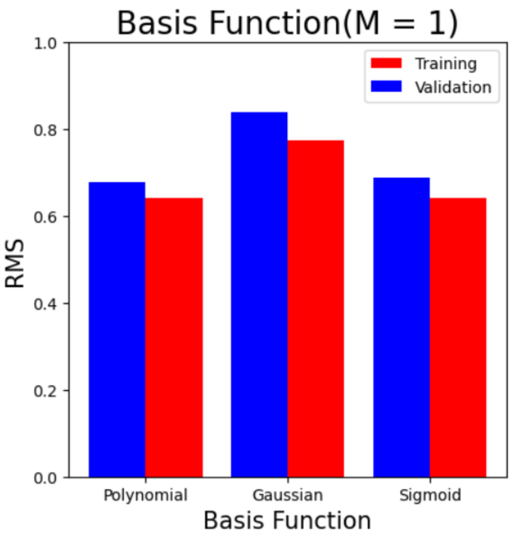
```
Gaussian M = 2:
RMS_Gaussian_M2_train:  [0.71650612]
M2_Gaussian_train_accuracy:  82.04562178072112 %

RMS_Gaussian_M2_validation:  [0.907709]
M2_Gaussian_validation_accuracy:  70.0 %
```

```
Sigmoid M = 1:
RMS_sigmoid_M1_train:  [0.63962191]
M1_sigmoid_train_accuracy:  81.67770419426049 %

RMS_sigmoid_M1_validation:  [0.68668618]
M1_sigmoid_validation_accuracy:  78.33333333333333 %
```

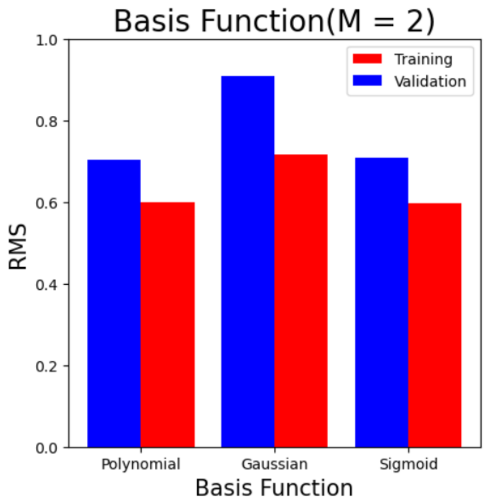
```
Sigmoid M = 2:
RMS_sigmoid_M2_train:  [0.5975065]
M2_sigmoid_train_accuracy:  84.4738778513613 %

RMS_sigmoid_M2_validation:  [0.70727267]
M2_sigmoid_validation_accuracy:  77.5 %
```

2.2(b)

```
M = 1
Polynomial :

RMS_train : 0.6407816319224392
RMS_validation : 0.677663293201 8127

Gaussian :

RMS_train : 0.7721218842579711
RMS_validation : 0.8393091006265121

Sigmoid :

RMS_train : 0.6396219090305546
RMS_validation : 0.6866861807428465
```



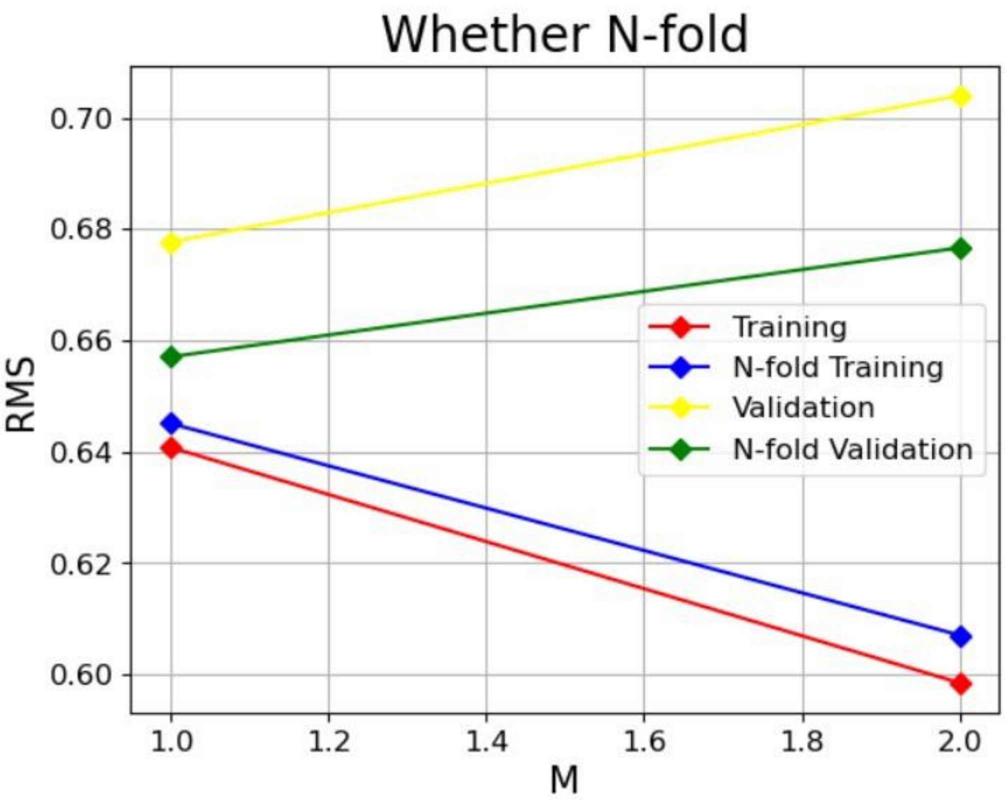
```
M = 2
Polynomial :

RMS_train : 0.5984609851256365
RMS_validation : 0.7039657061849073

Gaussian :

RMS_train : 0.7165061245175866
RMS_validation : 0.9077090023386037

Sigmoid :

RMS_train : 0.5975064974280028
RMS_validation : 0.7072726737896333
```



上方為我三種 Basis Fuction 在 M=1 和 M=2 時的數據和圖表，我所選擇的 polynomial 雖然 RMS 是最小的，但是似乎在階數(M)上升時，validation 會出 overfitting 的情形。

另外也可以觀察到階數上升可以有效幫助降低三者 training 的 RMS，但是皆會有 overfitting 的情況發生。其中 Gaussian 的 RMS 又比另外兩個大上許多為其最大缺點。

2.2(c)



此題我做 N-fold 的函數是我所選擇的 polynomial，我選擇使用階數(M)當作我的 Hyperparameter，並作圖比較 N-fold 前後 RMS 的變化。

我用的是 N=15 的 N-fold，也就是將所有 Data 切成 15 份並分別輪流將每份當作 validation 去計算 RMS 並取 15 份的平均值。

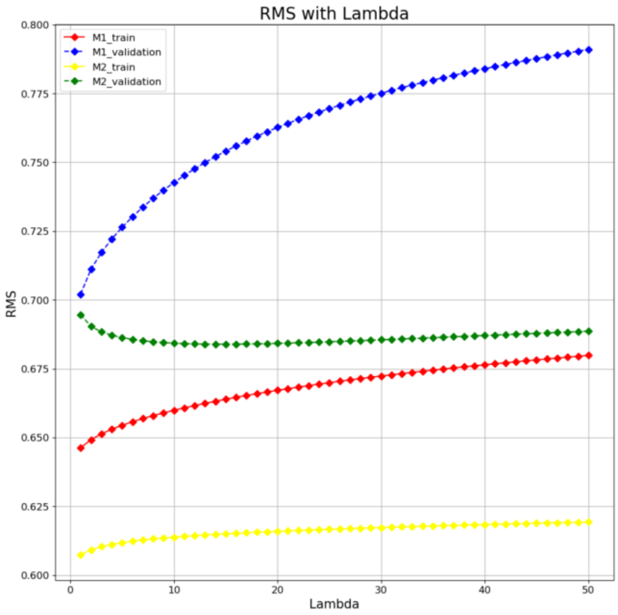
可以從上圖中觀察到做完 N-fold 後依然可以隨著階數上升而減少 RMS 值，也能有效的減少 Overfitting 的狀況，可以看出 N-fold validation 的 RMS 值比沒做之前的小。

2.3 Maximum a posteriori approach

(a)Maximum likelihood 和 MAP 的差別在於 MAP 有考慮進 prior 的影響，Maximum likelihood 只會受到 Data 的影響曲尋找最可能的結果，而 MAP 則會同時考慮 Data 和 prior 的影響來尋找結果。

而其中的 key difference 則是 MAP 的 RMS 加上了 Regularization 項$E(w) = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n,\omega) - t_n\}^2 + \frac{\lambda}{N}\|w\|^2$，目的是為了有效減少 overfitting 的發生，並使 w = $(\emptyset^T \quad \emptyset \quad +\lambda I)^{-1}\emptyset^T t$。

(b)

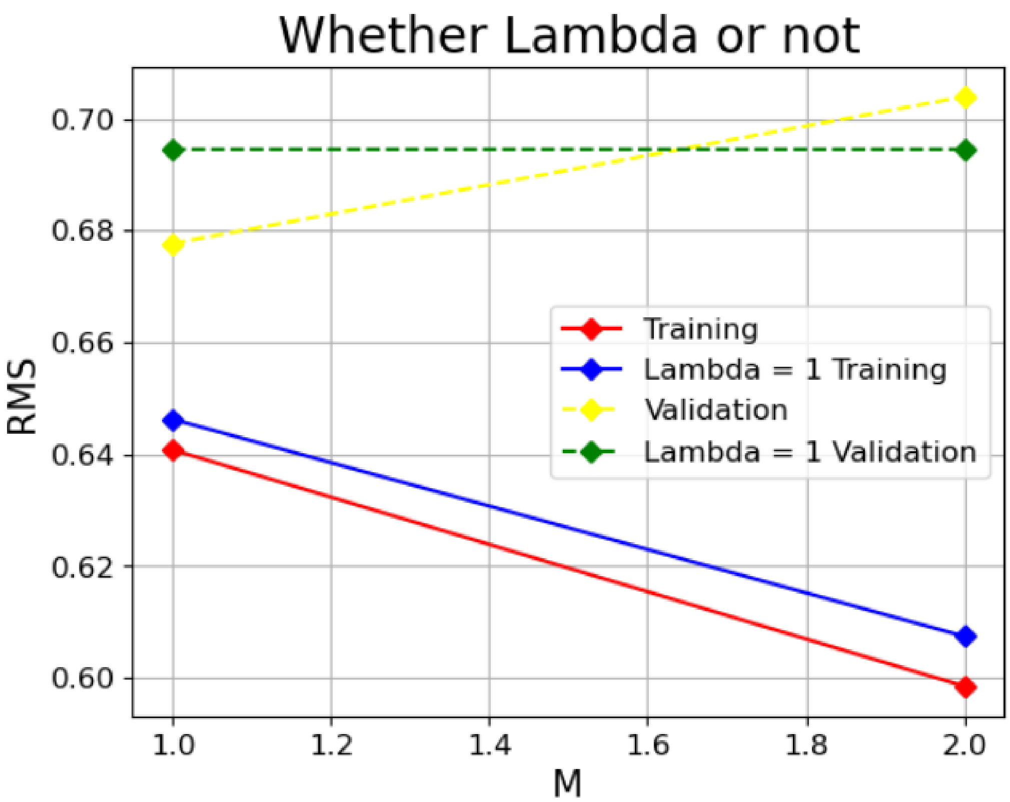


```
Lambda =  1
RMS_M2_train_posterior : 0.6073963580743086
RMS_M2_validation_posterior : 0.6946550181633245

Lambda =  2
RMS_M2_train_posterior : 0.6091786821598503
RMS_M2_validation_posterior : 0.6903526966959063

Lambda =  3
RMS_M2_train_posterior : 0.6103077746693011
RMS_M2_validation_posterior : 0.6883944315474206

Lambda =  4
RMS_M2_train_posterior : 0.6111292842754024
RMS_M2_validation_posterior : 0.6871708085298471

Lambda =  5
RMS_M2_train_posterior : 0.6117718576299316
RMS_M2_validation_posterior : 0.6863024442766865
```


```
Lambda =  1
RMS_M2_train_posterior : 0.6073963580743086
RMS_M2_validation_posterior : 0.6946550181633245

Lambda =  2
RMS_M2_train_posterior : 0.6091786821598503
RMS_M2_validation_posterior : 0.6903526966959063

Lambda =  3
RMS_M2_train_posterior : 0.6103077746693011
RMS_M2_validation_posterior : 0.6883944315474206

Lambda =  4
RMS_M2_train_posterior : 0.6111292842754024
RMS_M2_validation_posterior : 0.6871708085298471

Lambda =  5
RMS_M2_train_posterior : 0.6117718576299316
RMS_M2_validation_posterior : 0.6863024442766865
```


上圖為 MAP 後隨著 Lambda 的變化 RMS 的變化曲線以及 Lambda 從 1 到 5 的 RMS 值，可以看出 RMS 和 Lambda 的大小是呈正比的。

(c)



由上圖的結果可以發現 MAP 的 training 雖然變大但是 validation 變小了，表示有幫助減少 overfitting 的問題。

在沒有加入 prior 前，隨著階數上升， overfitting 也跟著變嚴重；在有 prior 加入後，即便階數上升，雖然還有一點 overfitting，但明顯減少很多。從作圖的結果可發現，做 MAP(加入 prior)確實有降低 overfitting 的效果並且可以保護 model 對測試資料的預測，有效降低 training 和 validation 之間的誤差，與我 a 小題的結論一致。