2022 NYCU Machine Learning – Homework1 Report

IEE 陳冠瑋 310510221 / Oct.18, 2022

1 Bayesian Linear Regression

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

其中 $\begin{cases} P(t|x,\vec{w}) = \mathcal{N}(t|y(x,\vec{w}),\beta^{-1}) = \mathcal{N}(t|w^T\phi(x),\beta^{-1}) \\ P(w) = \mathcal{N}(w|0,\alpha^{-1}I) \end{cases}$

$$\because P(\vec{w}|\vec{x},\vec{t}) \propto P(t|\vec{x},\vec{w}) \times P(\vec{w}) \propto \prod_{n=1}^{N}\mathcal{N}(t_n|w^T\phi(x_n),\beta^{-1})\cdot\mathcal{N}(w|0,\alpha^{-1}I)$$

$$\propto \exp\left[-\frac{\beta}{2}(t_1-w^T\phi(x_1))^2+(t_2-w^T\phi(x_2))^2+\cdots+(t_N-w^T\phi(x_N))^2\right]\exp\left(-\frac{\alpha}{2}w^Tw\right)$$

$$= \exp\left[-\frac{\beta}{2}\sum_{n=1}^{N}(t_n^2+w^T\phi(x_n)\phi(x_n)^Tw-2w^T\phi(x_n)t_n)-\frac{\alpha}{2}w^Tw\right]$$

$$\propto -\frac{1}{2}w^T(\beta\sum_{n=1}^{N}\phi(x_n)\phi^T(x_n)+\alpha I)w-2\beta w^T\sum_{n=1}^{N}(\phi(x_n)t_n)$$

$$\Rightarrow S_N^{-1} = \alpha I+\beta\sum_{n=1}^{N}\phi(x_n)\phi^T(x_n),\quad m_N = S(\sum_{n=1}^{N}\beta\phi(x_n)t_n)$$

$$\therefore P(\vec{w}|\vec{x},\vec{t}) = \mathcal{N}(w|m_N,S_N)$$

$$P(t|x,\vec{x},\vec{t}) = \int_{-\infty}^{\infty}P(t|x,\vec{w})P(\vec{w}|\vec{x},\vec{t})\,dw \propto \int_{-\infty}^{\infty}\exp\left(-\frac{\beta}{2}(t-w^T\phi(x))^2\right)\cdot\exp\left(-\frac{1}{2}(w-m_N)^TS_N^{-1}(w-m_N)\right)dw$$

$$\propto \int_{-\infty}^{\infty}\exp\left[\frac{-\beta}{2}(t^2-2(w^T\phi(x))t+(w^T\phi(x))^2\right]\cdot\exp\left[-\frac{1}{2}(w^TS_N^{-1}w-2w^TS_N^{-1}m_N+m_N^TS_N^{-1}m_N)\right.$$

$$\propto \int_{-\infty}^{\infty}\exp\left[-\frac{1}{2}(\beta t^2-2\beta w^T\phi(x)\,t+\beta w^T\phi(x)\phi^T(x)w+(w^TS_N^{-1}w)-(2w^TS_N^{-1}m_N))\right]dw$$

$$= \int_{-\infty}^{\infty}\exp\left[-\frac{1}{2}[w^T(\beta\phi^T(x))+S_N^{-1}]w-2w^T(\phi(x))t\beta+S_N^{-1}m_N)+\beta t^2\right]dw$$

Compare with $\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) = -\frac{1}{2}(x^T\Sigma^{-1}-2x^T\Sigma^{-1}\mu+\mu^T\Sigma^{-1}\mu)$

令 $\Sigma^{-1} = \beta\phi(x)\phi^T(x)+S_N^{-1}$, $\mu = \Sigma(\beta\phi(x)t+S_N^{-1}m_N)$

$$= \int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2}[(w^T\Sigma^{-1}w-2w^T\Sigma^{-1}\mu+\mu^T\Sigma^{-1}\mu)-\mu^T\Sigma^{-1}\mu+\beta t^2]\right\}dw$$

$$= \int_{-\infty}^{\infty}\exp\left\{-\frac{1}{2}[(w-\mu)^T\Sigma^{-1}(w-\mu)]\right\}\cdot\exp\left\{-\frac{1}{2}(\beta t^2-\mu^T\Sigma^{-1}\mu)\right\}dw$$

$$= \exp\left\{-\frac{1}{2}(\beta t^2-\mu^T\Sigma^{-1}\mu)\right\} = \exp\left\{-\frac{1}{2}(\beta t^2-(\Sigma(\beta\phi(x)t+S_N^{-1}m_N))^T\Sigma^{-1}(\Sigma(\beta\phi(x)t+S_N^{-1}m_N)\right)$$

$$= \exp\left\{-\frac{1}{2}(\beta t^2-(\beta\phi(x)t+S_N^{-1}m_N)^T\Sigma^T(\beta\phi(x)t+S_N^{-1}m_N))\right\}$$

$$= \exp\left\{-\frac{1}{2}[\beta t^2-((\beta\phi(x)t)^T\Sigma(\beta\phi(x)t)+2(S_N^{-1}m_N)^T\Sigma(\beta\phi(x)t)+\text{const}]\right\}$$

$$\propto \exp\left\{-\frac{1}{2}[\beta t^2-\beta^2\phi(x)^T\Sigma\phi(x)t^2+2(S_N^{-1}m_N)^T\Sigma\beta\phi(x)t]\right\}$$

$$= \exp\left\{-\frac{1}{2}[(\beta-\beta^2\phi(x)^T\Sigma\phi(x))t^2-2(S_N^{-1}m_N)^T\Sigma\beta\phi(x)t]\right\}$$

$$\propto \exp\left\{-\frac{1}{2}(\beta-\beta^2\phi(x)^T\Sigma\phi(x))\left(t-\frac{S_N^{-1}m_N^T\Sigma\beta\phi(x)t}{\beta-\beta^2\phi(x)^T\Sigma\phi(x)}\right)^2\right\}$$

$$\Rightarrow S^2(x) = (\beta-\beta^2\phi(x)^T\Sigma\phi(x))^{-1},\quad \Sigma = (\Sigma^{-1})^{-1} = [S_N^{-1}+\beta\phi(x)\phi^T(x)]^{-1} = S_N-\frac{S_N\beta\phi(x)^TS_N}{1+\phi(x)^TS_N\beta\phi(x)}\phi(x))^{-1}$$

$$\Rightarrow S^2(x) = (\beta-\beta^2\phi(x)^T(S_N-\frac{S_N\beta\phi(x)\phi^T(x)S_N}{1+\phi(x)^TS_N\beta\phi(x)})\phi(x))^{-1}$$

$$= (\beta-\beta^2\phi(x)^TS_N\cdot\frac{\phi(x)+\beta\phi(x)\phi(x)^TS_N\phi(x)-\beta\phi(x)\phi(x)^TS_N\phi(x)}{1+\beta\phi(x)^TS_N\phi(x)})^{-1}$$

$$= (\beta-\beta^2\frac{\phi(x)^TS_N\phi(x)}{1+\beta\phi(x)^TS_N\phi(x)})^{-1} = [\beta(1-\beta\frac{\phi(x)^TS_N\phi(x)}{1+\beta\phi(x)^TS_N\phi(x)})]^{-1}$$

$$= (\beta\cdot\frac{1}{1+\beta\phi(x)^TS_N\phi(x)})^{-1} = \frac{1+\beta\phi(x)^TS_N\phi(x)}{\beta}$$

$$m(x) = y(x,m_N) = \phi(x)^Tm_N = \phi(x)^T[S_N(\sum_{n=1}^{N}\beta\phi(x_n)t_n)]$$

$$= \beta\phi(x)^TS_N\sum_{n=1}^{N}\phi(x_n)t_n$$

$$\therefore P(t|x,\vec{x},\vec{t}) = \mathcal{N}(t(m(x),S^2(x)),\begin{cases} m(x) = \beta\phi(x)^TS_N\sum_{n=1}^{N}\phi(x_n)t_n \\ S^2(x) = \beta^{-1}+\phi(x)^TS_N\phi(x) \\ S_N^{-1} = \alpha I+\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T \end{cases}$$ *

2 Linear Regression

2.1 Feature selection

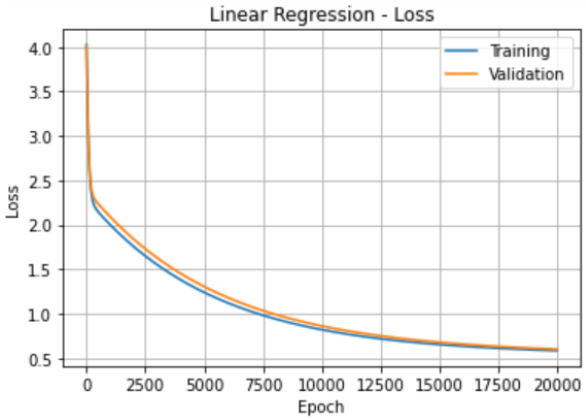
(a) In the feature selection stage, please apply polynomials of order M = 1 and M = 2 over the input data with dimension D = 11. Please evaluate the corresponding RMS error on the training set and valid set.

我先將 data X 與 T 合併一起 shuffle，然後以 8:2 拆成 training 與 validation data。這邊嘗試使用兩種方法：

第一種方法是利用 gradient descent 的方法($w^* = w - lr \times \frac{\partial L}{\partial w}$)去更新 linear regression model 的 weight，然後去求取每個 epoch 帶入 RMS 計算出來的 loss。第二種方法是直接利用公式$w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t$，其中$\Phi$可以由$y(x,w) = \sum_{j=0}^{M-1} w_j\Phi_j(x) = w^T\Phi(x)$推得，最後代入RMS $= \sqrt{\frac{\sum_{t=1}^{n}(y-\hat{y})^2}{n}}$，計算 RMS error 的結果。

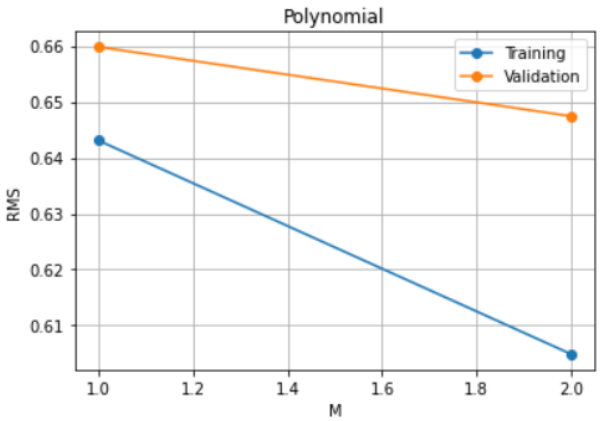
Method 1: Gradient descent

```
---------------------------------
epoch: 19995 / 20000
traing loss: 0.5837713392134929
validation loss: 0.5999877232894505
---------------------------------
epoch: 19996 / 20000
traing loss: 0.5837625801755816
validation loss: 0.5999776262632301
---------------------------------
epoch: 19997 / 20000
traing loss: 0.5837538225995597
validation loss: 0.5999675308622933
---------------------------------
epoch: 19998 / 20000
traing loss: 0.5837450664851666
validation loss: 0.5999574370863594
---------------------------------
epoch: 19999 / 20000
traing loss: 0.5837363118321403
validation loss: 0.599947344935149
---------------------------------
```



Method 2: $w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t$

```
----------Polynomial----------

M=1
RMS train:   0.64310530043101
RMS val   :   0.6598255614779065

M=2
RMS train:   0.6048685472553934
RMS val   :   0.6474326189304004
```



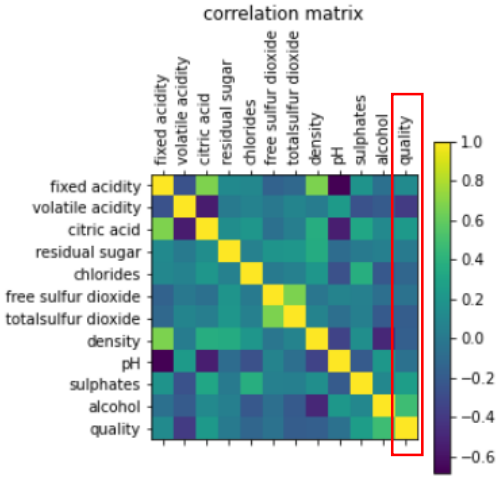
從上圖可知，當 Polynomial order M=1 時，training set 的 RMS error 約為 0.64310，validation set 的 RMS error 約為 0.65982；當 Polynomial order M=2 時，training set 的 RMS error 約為 0.6048，validation set 的 RMS error 約為 0.6474。可以發現當 Polynomial order M 變高的時候，training 與 validation 的 RMS 都有下降的趨勢，表示較複雜的模型(Polynomial order M=2)對於這個 dataset 能有較好的學習結果。

(b) How will you analyze the weights of the polynomial model M = 1 and select the most contributive feature?

這邊也嘗試了使用兩種方法來決定哪一個 feature 對整個模型的影響是最大的。

Method 1: Correlation

第一種方法是去觀察 quality 與所有 feature 之間的 correlation。Correlation 越高代表兩者之間相互的關係越強，如果拿一個 feature 與 quality 去做計算 correlation，就可以知道不同 feature 與 quality 之間的相關性。觀察以下的 correlation matrix 可以發現這些 feature 與 quality 之間的 correlation 相差不大，代表每一個 feature 之間的 contribution 是差不多的，如果硬要選出 contribution 前幾名的 feature，alcohol 大概比其他的 feature 的 contribution 再高一些。

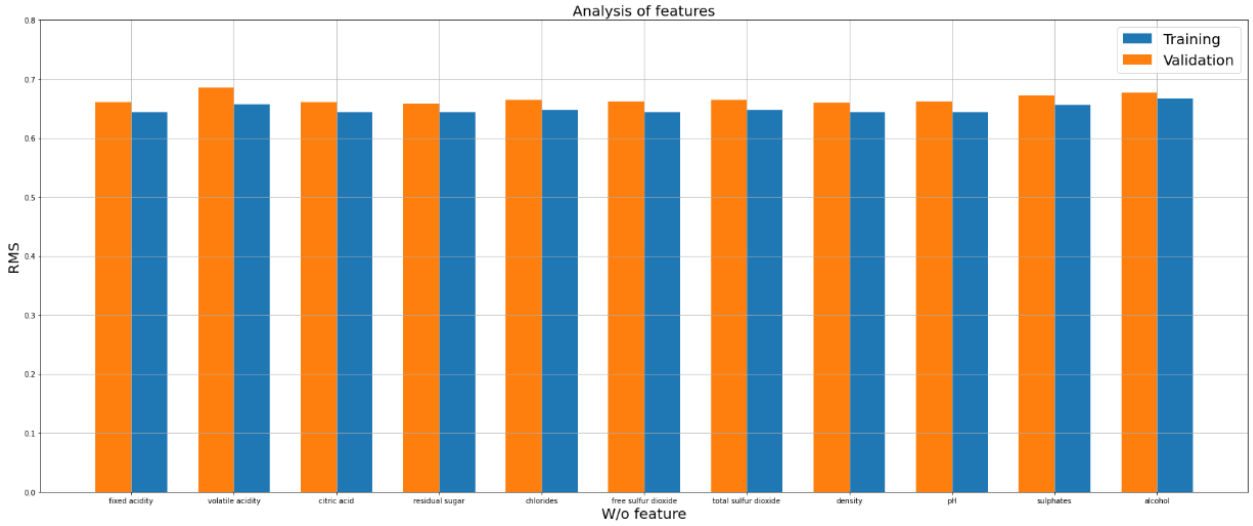


	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

Method 2: Remove one of the features to calculate the corresponding RMS error

第二種方法是透過移除其中一種 feature，並保留其他的 feature，再計算出 Polynomial order M=1 時對應的 Training RMS error 與 Validation RMS error。由下圖可知，移除其中一種 feature 對應算出來的 RMS error 相差沒有很大，所以推測每一個 feature 之間的 contribution 差不多。如果要選出 most contributive 的 feature，alcohol 的移除會導致 error 提升最顯著，所以 most contributive 的 feature 是 alcohol (但每一個 feature 的 contribution 差不多)。



2.2 Maximum likelihood approach

(a) Which basis function will you use to further improve your regression model, polynomial, Gaussian, Sigmoid, or hybrid?

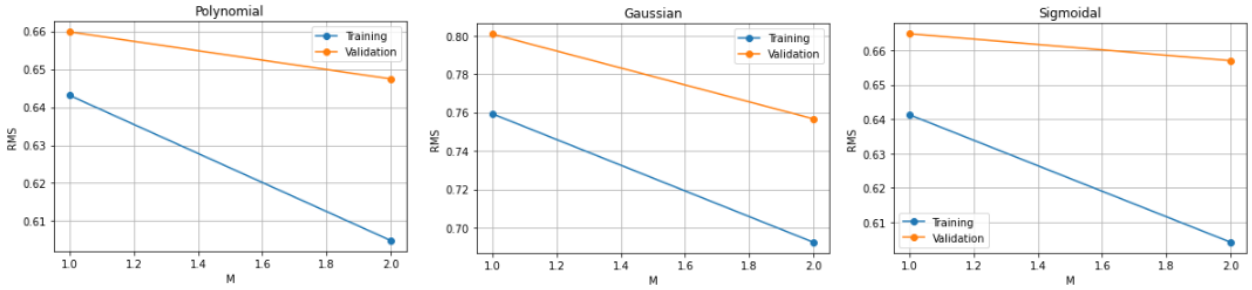
利用 Polynomial, Gaussian 與 Sigmoid 三種 basis function 分別計算在 M=1 和 M=2 時 training 和 validation 的 RMS。可以發現使用 Polynomial 時，training 與 validation RMS 都有不錯的表現(RMS 小)，因此使用 Polynomial 來提升我的 regression model。

```
----------Polynomial----------  ----------Gaussian----------  ----------Sigmoidal----------

M=1                              M=1                           M=1
RMS train:  0.64310530043101     RMS train:  0.7592669615485866  RMS train:  0.6412945700979797
RMS val  :  0.6598255614779065   RMS val:    0.8008020247068542  RMS val:    0.6648415660339035

M=2                              M=2                           M=2
RMS train:  0.6048685472553934   RMS train:  0.6923484347609844  RMS train:  0.6042039194969999
RMS val  :  0.6474326189304004   RMS val:    0.7566264445030028  RMS val:    0.6569981607249855
```

(b) Introduce the basis function you just decided in (a) to the linear regression model and analyze the result you get. (Hint: You might want to discuss the phenomenon when the model becomes too complex.)

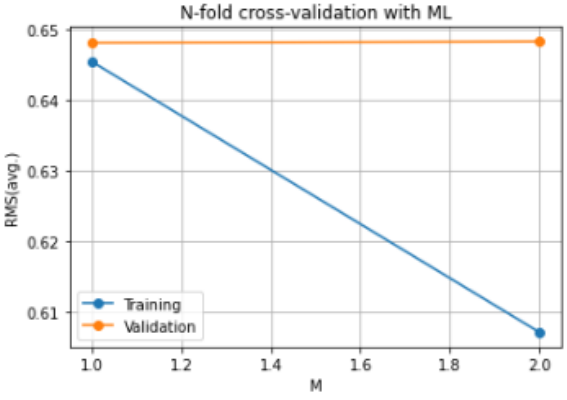
我使用 Polynomial 作為 basis function，相較於其他兩個，在 training 與 validation 的 RMS 上有更好的表現。可以觀察到 Polynomial, Gaussian 與 Sigmoid 在 M=1 時，RMS 都是相對於 M=2 時還要高，代表在這個問題中，較複雜的模型(M=2)並不會讓模型 over fitting。

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



(c) Apply N-fold cross-validation in your training stage to select at least one hyper-parameter (order, parameter number, ...) for the model and do some discussion (underfitting, overfitting).

```
N-fold cross-validation with ML

RMS train (M=1) :
mean:      0.6453792623844965
variance:  2.6877314763708544e-05

RMS val (M=1) :
mean:      0.6480426520189771
variance:  0.002541810107620268

RMS train (M=2) :
mean:      0.6070894502478673
variance:  2.3612270898815526e-05

RMS val (M=2) :
mean:      0.6482231719460065
variance:  0.0021716330378396065
```



N-fold cross-validation 主要作用是要防止因為模型過於複雜所導致的 overfitting。這裡使用 N-fold cross-validation 並將 N 設為 10，作法是將資料隨機平均分成 10 個集合，然後將某一個集合作為 Validation data，剩下 9 個集合作為 Training data，如此重複進行直到每一個集合都被當作 Validation data 為止。由圖可知，在 Polynomial order M=2 時，training 的 RMS 優於 M=1 時的 RMS，而且有明顯的下降。但 Validation 的 RMS 並沒有特別顯著的提升。

2.3 Maximum a posterior approach

(a) What is the key difference between the maximum likelihood approach and the maximum posterior approach?

Maximum Likelihood Estimation (MLE)

假設資料 $x_1, x_2, \dots, x_n$ 是 i.i.d (Independent and identical distribution)的一組抽樣， $X = (x_1, x_2, \dots, x_n)$。MLE 的估計方法推導如下：

$$\theta_{MLE} = \text{argmax } P(X|\theta)$$
$$= \text{argmax } P(x_1;\theta)P(x_2;\theta)\cdots P(x_n;\theta)$$
$$= \text{argmax } \log\prod_{i=1}^{n} P(x_i;\theta)$$
$$= \text{argmax } \sum_{i=1}^{n} logP(x_i;\theta)$$
$$= \text{argmin } -\sum_{i=1}^{n} logP(x_i;\theta)$$

Maximum A Posterior (MAP)

假設資料 $x_1, x_2, \dots, x_n$ 是 i.i.d (Independent and identical distribution)的一組抽樣， $X = (x_1, x_2, \dots, x_n)$。MAP 的估計方法推導如下：

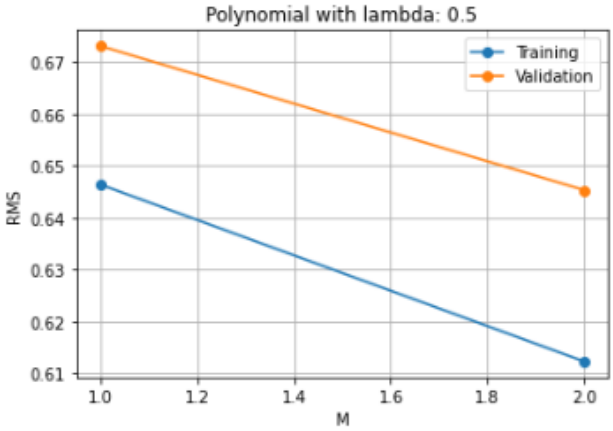
$$\theta_{MAP} = \text{argmax } P(\theta|X)$$
$$= \text{argmin } -\log P(\theta|X)$$
$$= \text{argmin } -\log P(X|\theta) - logP(\theta) + logP(X)$$
$$= \text{argmin } -\log P(X|\theta) - \log P(\theta)$$

由以上公式，可以得出，當 prior follow uniform distribution 時，MLE 是 MAP 的一種特殊情況。另外，在 MAP 中使用一個 Gaussian distribution 的 prior probability 等價於 MLE 中採用的 L2 regularization。因此，原本的 weight function，$w_{ML} = (\Phi^T\Phi)^{-1}\Phi^T t$ 會加入 lambda 修正項變成 $w_{ML} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T t$，目的是為了能夠在高階時有效降低 over fitting 的現象。

(b) Use the maximum a posteriori approach method to retest the model in 2.2 you designed. You could choose Gaussian distribution as a prior.

下面是使用 MAP 的方法，並取 lambda = 0.5 ，重做第二題所得到 RMS 的結果。從圖中可知，當 Polynomial order M= 1 時，training RMS 約為 0.64641，validation RMS 約為 0.67315。在 Polynomial order M=2 時，training RMS 約為 0.61227，validation RMS 約為 0.64534，可以發現加上 lambda 修正項後，有讓模型在更高階時讓 RMS error 下降的更明顯。

```
Polynomial
lambda =  0.5

M=1
RMS train with lambda:  0.6464153177053508
RMS val   with lambda:  0.6731507962562808

M=2
RMS train with lambda:  0.6122687209026715
RMS val   with lambda:  0.6453353906737497
```



(c) Compare the result between the maximum likelihood approach and maximum a posteriori approach. Is it consistent with your conclusion in (a)?

下圖是比較使用 MLE(左圖)與 MAP(右圖)，可以發現 MAP 在 M=2 時讓 RMS error 下降的更明顯，能夠降低 over fitting 的現象。