Machine Learning 2020: Homework 1

張軼峰(交:A091501) (清:109033804)

October 31 2020

1 Bayesian Linear Regression

1. Why we need the basis function $\phi(x)$ for linear regression? And what is the benefit for applying basis function over linear regression?

   We are trying to predict t from x, for some future test case, but we are not trying to model the distribution of x. Suppose also that we don't expect the best predictor for t to be a linear function of x, so ordinary linear regression on the original variables won't work well. We need to allow for a non-linear function of x.
   The **basis function** arises by taking linear combinations of a fixed set of some nonlinear functions of the input variables. Such models are linear functions of the parameters (read, simple analytical properties), and yet can be **nonlinear with respect to the input variables**.

2. Prove that the predictive distribution just mentioned is the same with the form

$$p(t|x, \mathbf{x}, \mathbf{t}) = \mathcal{N}(t|m(x), s^2(x))$$

where

$$m(x) = \beta\phi(x)^T\mathbf{S}\sum_{n=1}^{N}\phi(x_n)t_n$$

$$S^2(x) = \beta^{-1} + \phi(x)^T\mathbf{S}\phi(x).$$

Here, the matrix $\mathbf{S}^{-1}$ is given by $\mathbf{S}^{-1} = \alpha\mathbf{I} + \beta\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T$

Ans:

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

First,

$$MAP \propto ML \cdot prior$$

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\alpha)$$

in p.93 marginal Gaussian $p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})$

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{t}|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) = \mathcal{N}(t|\mathbf{w}^T\Phi(x), \beta^{-1}\mathbf{I}) = \mathcal{N}(\mathbf{t}|\mathbf{w}^T\mathbf{A} + b, \mathbf{L}^{-1})$$

$$\rightarrow \mathbf{A} = \Phi(x)^T, b = 0, \mathbf{L} = \beta\mathbf{I}$$

in p.93 marginal Gaussian $p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) = \mathcal{N}(\mathbf{w}|\mu, \Lambda^{-1}) \rightarrow \mu = 0, \Lambda = \alpha\mathbf{I}$$

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}) = \mathcal{N}(\mathbf{w}|\Sigma\{\mathbf{A}^T\mathbf{L}(\mathbf{w} - b) + \Lambda\mu\}, \Sigma) \rightarrow \Sigma = (\alpha\mathbf{I} + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1}$$

$$\Rightarrow \mathcal{N}(\mathbf{w}|(\alpha\mathbf{I} + \Phi(x)^T\beta\Phi(x))^{-1}\{\Phi(x)^T\beta\mathbf{t}\}, (\alpha\mathbf{I} + \Phi(x)\beta\Phi(x)^T)^{-1})$$

$$= \mathcal{N}(w|\mathbf{S}(\Phi^T(x)), \mathbf{S}), \text{ where } \mathbf{S} = (\alpha\mathbf{I} + \Phi(x)\beta\Phi(x)^T)^{-1})$$

repeat

$$p(\mathbf{t}|\mathbf{x},\mathbf{w}) = \mathcal{N}(\mathbf{t}|y(\mathbf{x},\mathbf{w}),\beta^{-1}) = \mathcal{N}(t|\mathbf{w}^T\Phi(x),\beta^{-1}\mathbf{I}) = \mathcal{N}(\mathbf{t}|\mathbf{w}^T\mathbf{A}+b,\mathbf{L}^{-1})$$
$$\rightarrow \mathbf{A} = \Phi(x)^T, b = 0, \mathbf{L} = \beta\mathbf{I}$$
$$p(\mathbf{w}|\mathbf{x},\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{S}(\Phi^T(x)),\mathbf{S}) = p(\mathbf{w}|\mu,\Lambda^{-1}) \rightarrow \mu = \mathbf{S}(\beta\Phi(x)t),\ \Lambda^{-1} = \mathbf{S}$$

$$\begin{aligned} p(t|x,\mathbf{x},\mathbf{t}) &= \mathcal{N}(y|\mathbf{A}\mu+b,\mathbf{L}^{-1}+\mathbf{A}\Lambda^{-1}\mathbf{A}^T) \\ &= \mathcal{N}(t|\beta\Phi(x)^T\mathbf{S}\Phi(x)t,\ \beta^{-1}+\Phi(x)^T\mathbf{S}\Phi(x)) \\ &= \mathcal{N}(t|m(x),\mathbf{S}^2(x)) \end{aligned} \tag{1}$$

3. Could we use linear regression function for classification? Why or why not? Explain it!

   Yes, we can extend linear regression back to a linear model and apply on classification tasks. For regression problems, the target variable **t** was simply the vector of real numbers whose values we wish to predict. For classification problems, considering probabilistic models, in the case of two-class problems the binary representation in which there is a single target variable $t \in \{0, 1\}$ such that $t = 1$ represents class C_1 and $t = 2$ represents class C_2. In multi-output regression, **k outputs** could related to **k classes**. Furthermore, we can take a linear function as linear discriminant, such discriminant function could generate decision surface. For example, logistic regression is a powerful model for classification.

2 Linear Regression

1. Feature select

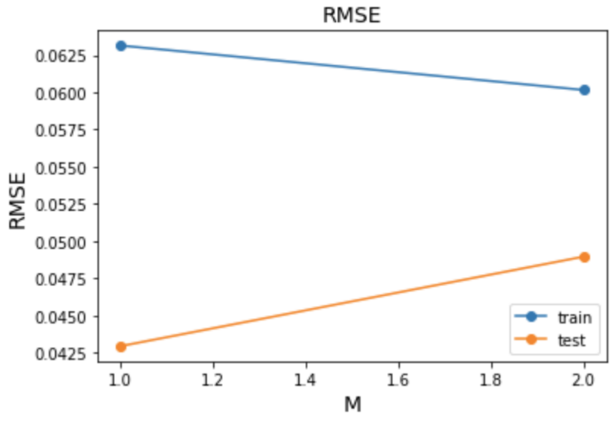
   Polynomial function

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

   Error function

$$E(w) = \frac{1}{2}\sum_{n=1}^{D} y(x_n, w) - {t_n}^2$$

   (a) In the feature selection stage, please apply polynomials of order $M = 1$ and $M = 2$ over the dimension $D = 7$ input data. Please evaluate the corresponding RMS error on the training set and valid set. **Code Result**



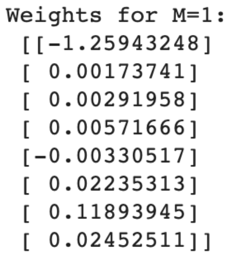
```
M=1, training RMSE: 0.063142, testing RMSE: 0.042932
M=2, training RMSE: 0.060157, testing RMSE: 0.048945
```

將Data import 後，先將data 做standarize。
Training data與Testing data的設定training 前80%, testing 後20%。
此題結果testing error 較training error 來的低，試過data 不同切法，即會得出不同結果。主要原因可能data set 筆數較少，且未將data shuffle，shuffle 後結果也可能不盡相同。

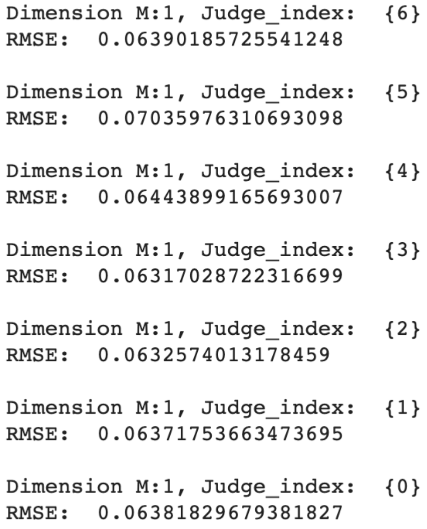
(b) How will you analysis the weights of polynomial model M = 1 and select the most contributive feature? **Code Result, Explain**

Ans:

首先印出M=1 時，各weight 的值(第一項對應bias w_0)，可以看到w_5(對應第六項feature)的值最大。接著輪流剔除一項feature，放入model 計算RMSE。Judge index代表剔除哪一項做regression，可以看到剔除feature[5] 的RMSE 最大。由以上結果可推知第六項feature - **CGPA** 的貢獻最大。



```
Weights for M=1:
 [[-1.25943248]
 [ 0.00173741]
 [ 0.00291958]
 [ 0.00571666]
 [-0.00330517]
 [ 0.02235313]
 [ 0.11893945]
 [ 0.02452511]]
```



```
Dimension M:1, Judge_index:  {6}
RMSE:  0.06390185725541248

Dimension M:1, Judge_index:  {5}
RMSE:  0.07035976310693098

Dimension M:1, Judge_index:  {4}
RMSE:  0.06443899165693007

Dimension M:1, Judge_index:  {3}
RMSE:  0.06317028722316699

Dimension M:1, Judge_index:  {2}
RMSE:  0.0632574013178459

Dimension M:1, Judge_index:  {1}
RMSE:  0.06371753663473695

Dimension M:1, Judge_index:  {0}
RMSE:  0.06381829679381827
```

2. Maximum likelihood approach

(a) Which basis function will you use to further improve your regression model, Polynomial, Gaussian, Sigmoidal, or hybrid? **Explain**

I choose Gaussian as the basis function and the advantages of Gaussian basis function are:

- The prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals and decide based on those if one should refit (online fitting, adaptive fitting) the prediction in some region of interest.
- The prediction interpolates the observations (at least for regular kernels).

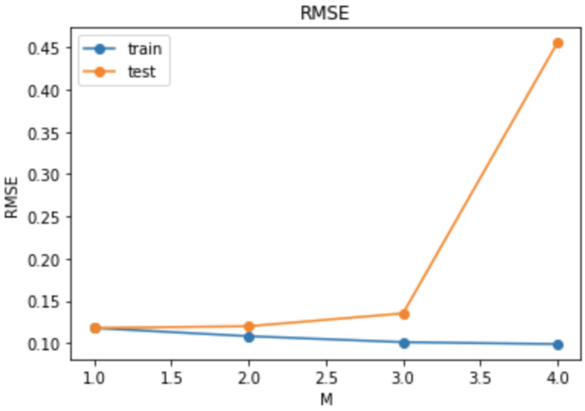
(b) Introduce the basis function you just decided in (a) to linear regression model and analyze the result you get. **Code Result, Explain**

選擇Gaussian 作爲basis function，

$$\phi_j(x) = exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$$

μ_j: govern the locations of the basis function in input space, s: governs spatial scale.

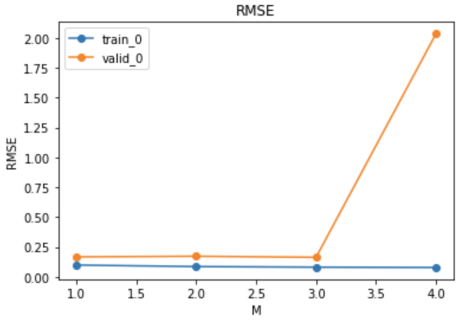
可以看到training error 隨M 增加而下降，顯示隨著model 的維度增加，能夠將model train得更好。然而，testing error 緩慢上升(可能爲資料存在一定的誤差值，且資料量較小)，並在$M = 4$ 時顯著增加，出現overfitting 的問題。



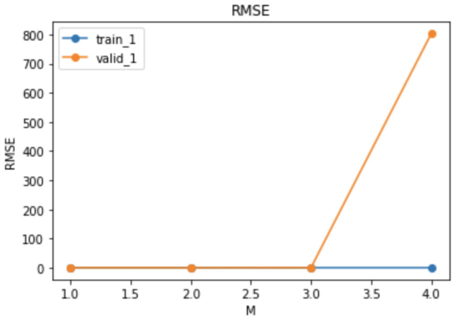
```
M=1, training RMSE: 0.117800, testing RMSE: 0.117878
M=2, training RMSE: 0.108118, testing RMSE: 0.119880
M=3, training RMSE: 0.101038, testing RMSE: 0.135055
M=4, training RMSE: 0.098770, testing RMSE: 0.455512
```

(c) Apply N-fold cross-validation in your training stage to select at least one hyperparameter(order, parameter number, ...) for model and do some discussion. **Code Result, Explain**

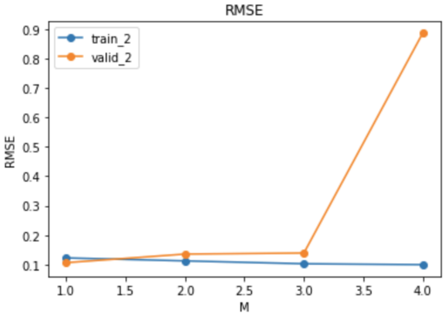
在training data 中分割N fold，分別執行N 次training 作為各次的validation set，並平均N 次validation 的error，以此方法來選擇model 或選擇hyperparameter 的設定。此處用以不同維度M 的model 來做練習。



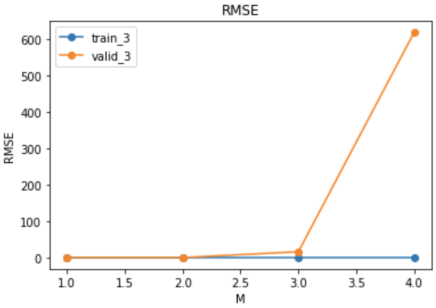
```
M=1, training RMSE: 0.098338, validation RMSE: 0.166663
M=2, training RMSE: 0.085581, validation RMSE: 0.172014
M=3, training RMSE: 0.080584, validation RMSE: 0.162829
M=4, training RMSE: 0.078030, validation RMSE: 2.034917
```



```
M=1, training RMSE: 0.122146, validation RMSE: 0.107293
M=2, training RMSE: 0.110931, validation RMSE: 0.109193
M=3, training RMSE: 0.105295, validation RMSE: 0.404435
M=4, training RMSE: 0.102881, validation RMSE: 802.7516
```



```
M=1, training RMSE: 0.122544, validation RMSE: 0.106402
M=2, training RMSE: 0.112432, validation RMSE: 0.135585
M=3, training RMSE: 0.102694, validation RMSE: 0.139145
M=4, training RMSE: 0.099619, validation RMSE: 0.886167
```



```
M=1, training RMSE: 0.123066, validation RMSE: 0.104566
M=2, training RMSE: 0.113704, validation RMSE: 0.105898
M=3, training RMSE: 0.108538, validation RMSE: 16.033955
M=4, training RMSE: 0.105984, validation RMSE: 618.134650
```

```
For M = 1, average of validation error = 0.121231
For M = 2, average of validation error = 0.130673
For M = 3, average of validation error = 4.185091
For M = 4, average of validation error = 355.951837
```

可以看到選用了不同的data set做training與validation的動作，validation 的error 即有大於或小於training error 的現象(M=1時)，印證第一題-(a) Explain 部分的結果。

Validation 的error 做平均之後，發現M=1 時，有最小的error，代表以此組data set 而言，M=1 應為最適合的model。延續第一題的結果討論，若切不同的data set來做training也可能導致不同結果。可能原因為data set 太小，導致data間的變異影響較大。

3. Maximum a posterior approach

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

Maximizing likelihood is equivalent to minimizing the sum-of-squares error function.

$$\Rightarrow w_{ML} = w_{LS}$$

$$MAP \propto ML \cdot prior$$

$$\begin{aligned} w_{MAP} &= \arg\max_{w} p(w|x,t,\alpha,\beta) \\ &= \arg\min_{w} \left\{ \frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\alpha}{2} w^T w \right\} \end{aligned} \tag{2}$$

The key difference between MAP and ML is that we adding a regularization term to control overfitting. Where we can minimize regularized sum-of-squares error function with regularization parameter, $\lambda = \frac{\alpha}{\beta}$. In 3.1.4 Regularized least squares, we can obtain

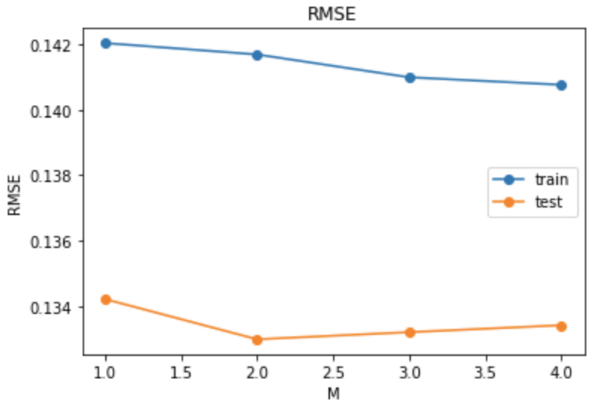
$$w = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t$$

.

(b) Use Maximum a posterior approach method to retest the model in 2 you designed. You could choose Gaussian distribution as a prior. **Code Result, Explain**

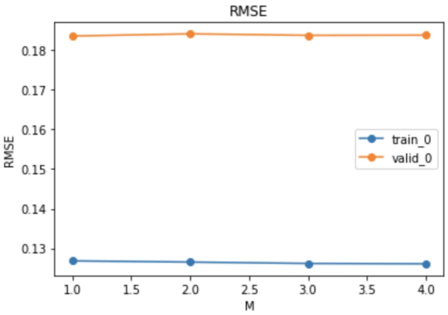
當我們加入regularization term$\lambda = 0.1$ 到model 後,training error keep相同表現隨著M增加而降低，但發現testing error 儘管在M=4時也不再有明顯overfitting了，regularization term 確實能有效控制overfitting.

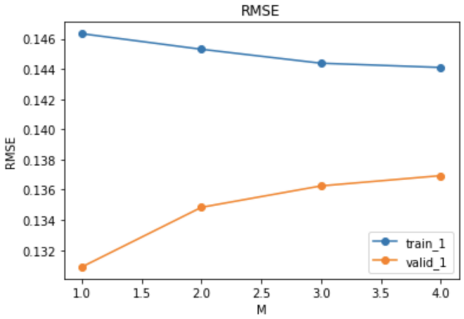
此時，testing error最小發生在M=2，加入regularization term也確實能夠提升model complexity來更準確執行regression.



```
M=1, training RMSE: 0.142037, testing RMSE: 0.134219
M=2, training RMSE: 0.141688, testing RMSE: 0.132992
M=3, training RMSE: 0.140991, testing RMSE: 0.133210
M=4, training RMSE: 0.140763, testing RMSE: 0.133422
```

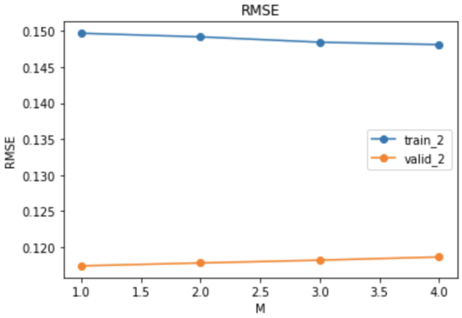
另外，以此model如2-2-c題，在training data部分實作N-fold validation來挑選hyperparameter M，可以觀察到隨著M上升，平均的error並無顯著增加，但在此實作中仍爲M=1 的error 些微低於其他model。

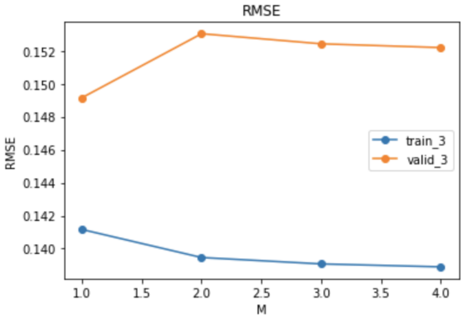




```
M=1, training RMSE: 0.126847, validation RMSE: 0.183500
M=2, training RMSE: 0.126542, validation RMSE: 0.184048
M=3, training RMSE: 0.126194, validation RMSE: 0.183674
M=4, training RMSE: 0.126107, validation RMSE: 0.183720
```

```
M=1, training RMSE: 0.146333, validation RMSE: 0.130888
M=2, training RMSE: 0.145300, validation RMSE: 0.134836
M=3, training RMSE: 0.144378, validation RMSE: 0.136253
M=4, training RMSE: 0.144098, validation RMSE: 0.136931
```





```
M=1, training RMSE: 0.149710, validation RMSE: 0.117372
M=2, training RMSE: 0.149208, validation RMSE: 0.117774
M=3, training RMSE: 0.148465, validation RMSE: 0.118162
M=4, training RMSE: 0.148134, validation RMSE: 0.118604
```

```
M=1, training RMSE: 0.141163, validation RMSE: 0.149155
M=2, training RMSE: 0.139462, validation RMSE: 0.153036
M=3, training RMSE: 0.139075, validation RMSE: 0.152423
M=4, training RMSE: 0.138897, validation RMSE: 0.152195
```

```
For M = 1, average of validation error = 0.145229
For M = 2, average of validation error = 0.147424
For M = 3, average of validation error = 0.147628
For M = 4, average of validation error = 0.147863
```

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

Maximum likelihood 與maximum a posterior 兩種方法的結果做比較，MAP 加入regularization term後，隨著model 的order增加能夠有效控制不會over-fitting，與(a)小題的推導論述相符。