ML HW 03

Author	陳懷安
Student ID	0753736

1. Gaussian Process for Regression

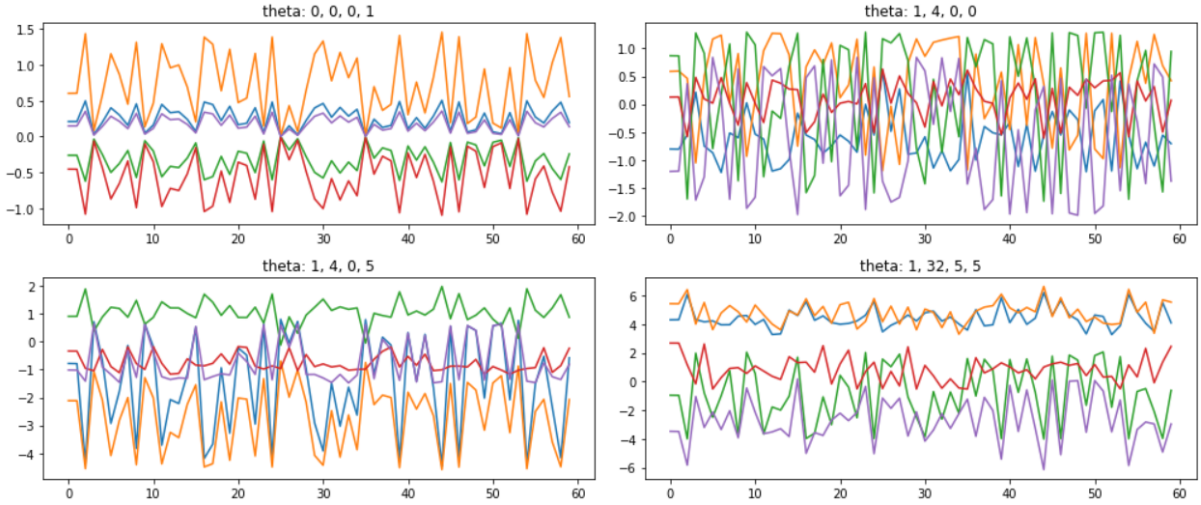
$$\mathrm{t}_n = y(\mathrm{X}_n) + \epsilon_n$$
$$\epsilon_n \sim N(0,\ \beta^{-1}),\quad \text{where } \beta = 1$$

Exponential-Quadratic Kernel Function

> Implement the Gaussian process with the exponential-quadratic kernel function.

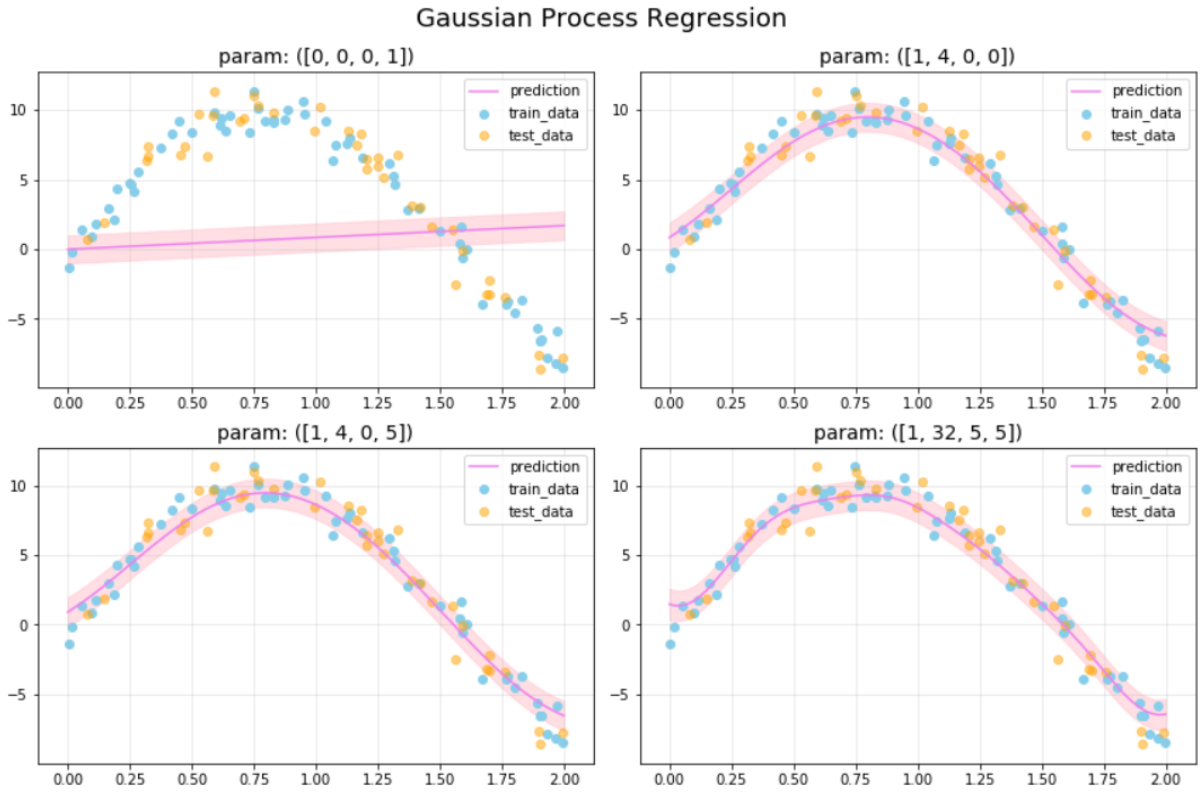
$$k(\mathrm{X}_n,\ \mathrm{X}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}||\mathrm{X}_n - \mathrm{X}_m||^2\right\} + \theta_2 + \theta_3 \mathrm{X}_n^T \mathrm{X}_m$$

- linear kernel: θ = {0, 0, 0, 1}
- squared exponential kernel: θ = {1, 4, 0, 0}
- exponential-quadratic kernel: θ = {1, 4, 0, 5}
- exponential-quadratic kernel: θ = {1, 32, 5, 5}



Prediction Result

> Plot the prediction result.



可以簡單的從上面 4 個圖形看出，在 `linear kernel` 的情況下，配飾出一條回歸線，`error` 必然最高；一旦加入了指數項後，圖形的配適情形在三組係數組合之間就幾乎沒有差異。

RMSE

> Calculate the corresponding root-mean-square errors for both training and test sets with respect to the four kernels.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}(m(x_n) - t_n)^2}$$

	param	train_RMSE	test_RMSE
0	[0, 0, 0, 1]	6.6576	6.7485
1	[1, 4, 0, 0]	1.0522	1.2988
2	[1, 4, 0, 5]	1.0288	1.2861
3	[1, 32, 5, 5]	0.9640	1.2585

`kernel` 由簡單到複雜，在 `linear kernel` 的情況下 `RMSE` 最高，符合預期。隨著 `kernel` 的複雜化， `RMSE` 逐漸下降， `test` 資料集的型情也是如此，故判斷尚未出現 `overfitting` 的現象。

Automatic Relevance Determination

> Try to tune the hyperparameters by trial and error to find the best combination for the
> dataset or apply the `automatic relevance determination` (ARD).

$$k(\mathrm{X}_n,\ \mathrm{X}_m) = \eta_{0i}\exp\left\{-\frac{\eta_{1i}}{2}||\mathrm{X}_n - \mathrm{X}_m||^2\right\} + \eta_{2i} + \eta_{3i}\mathrm{X}_n^T\mathrm{X}_m$$

各參數的 `∂C_N / ∂θ_i` 如下：

$$\begin{aligned}
\frac{\partial C_N}{\partial \eta_{0i}} &= \exp\left\{-\frac{\eta_{1i}}{2}||\mathrm{X}_n - \mathrm{X}_m||^2\right\} \\
\frac{\partial C_N}{\partial \eta_{1i}} &= \eta_{0i}\left(-\frac{1}{2}||\mathrm{X}_n - \mathrm{X}_m||^2\right)\exp\left\{-\frac{\eta_{1i}}{2}||\mathrm{X}_n - \mathrm{X}_m||^2\right\} \\
\frac{\partial C_N}{\partial \eta_{2i}} &= 1 \\
\frac{\partial C_N}{\partial \eta_{3i}} &= \mathrm{X}_n^T\mathrm{X}_m
\end{aligned}$$

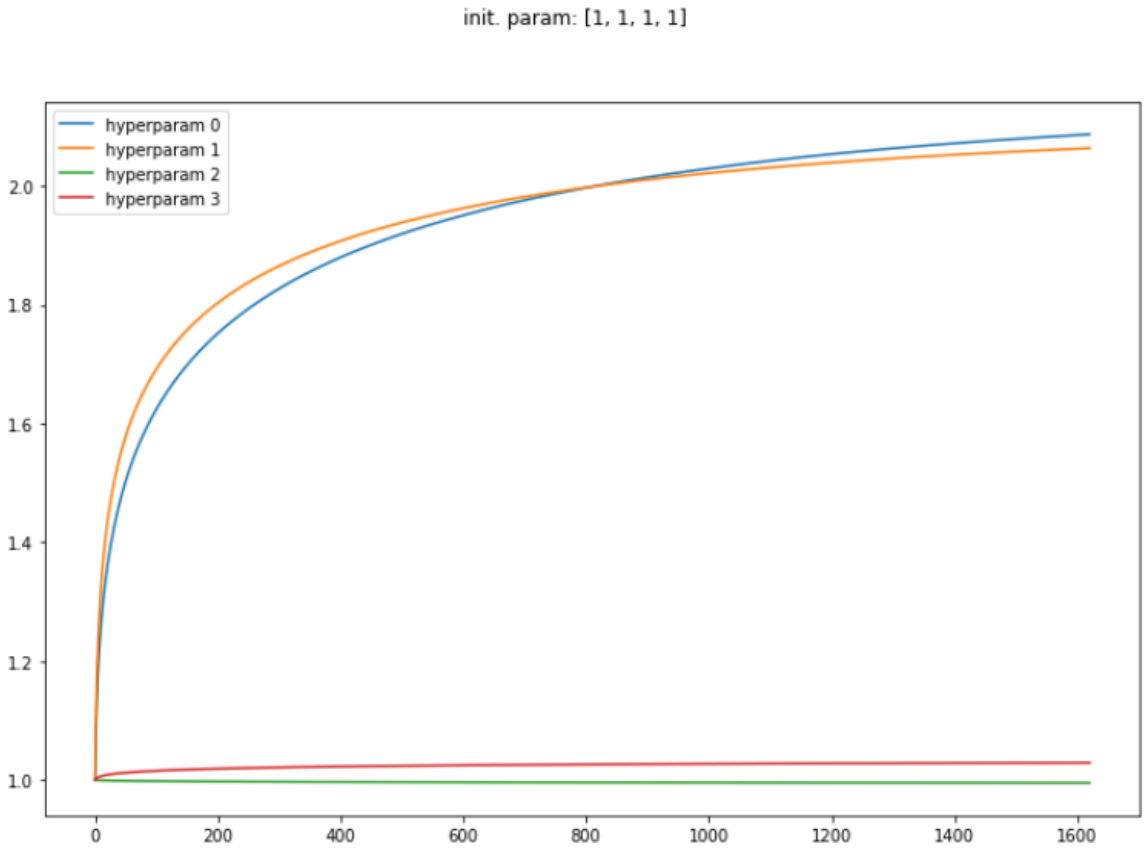
超參數的調整上，我們可以利用以下的求導過程得到更新的參數設定：

$$\ln p(t|\theta) = -\frac{1}{2}|C_N| - \frac{1}{2}t^T C_N^{-1} t - \frac{N}{2}\ln(2\pi)$$
$$\frac{\partial}{\partial \theta_i}\ln p(t|\theta) = -\frac{1}{2}\mathrm{Tr}\left(\mathrm{C_N^{-1}}\frac{\partial \mathrm{C_N}}{\partial \theta_\mathrm{i}}\right) + \frac{1}{2}\mathrm{t^T C_N^{-1}}\frac{\partial \mathrm{C_N}}{\partial \theta_\mathrm{i}}\mathrm{C_N^{-1}t}$$

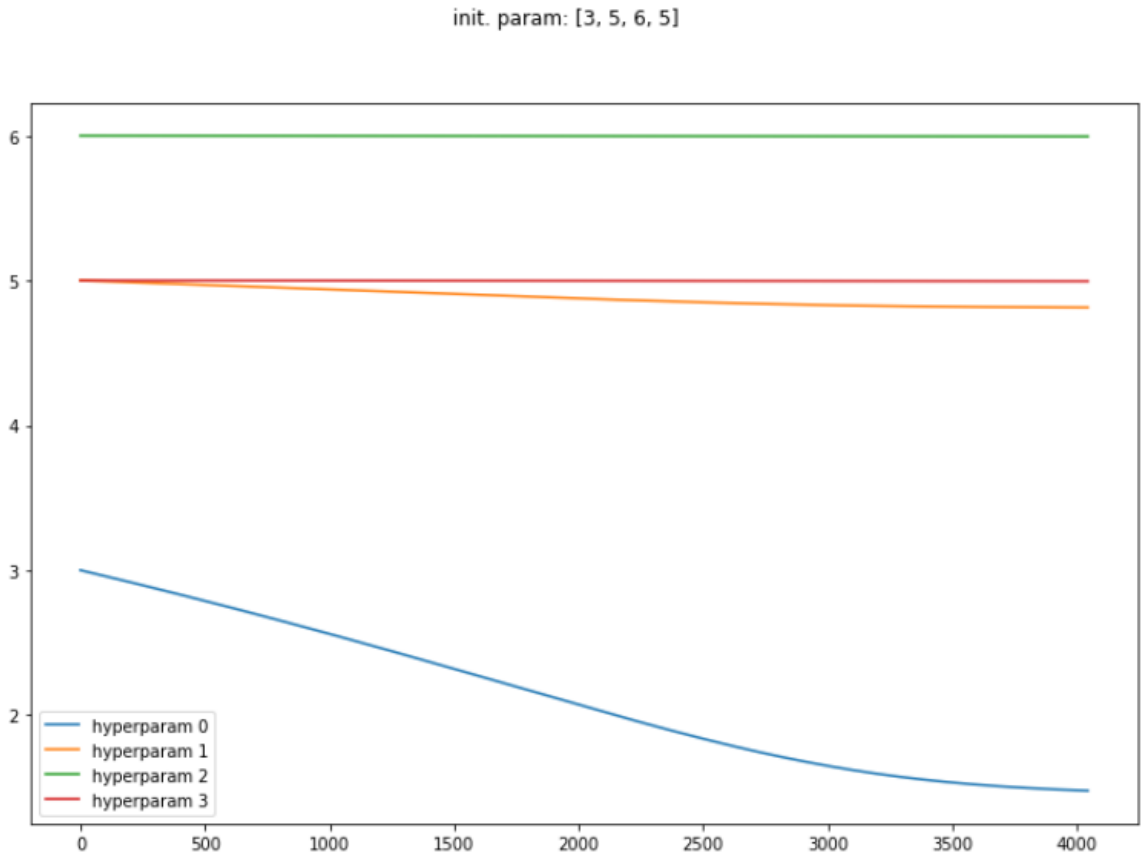
之後超參數將會以以下形式進行更新：

$$\eta_{ji}^{(new)} = \eta_{ji}^{(old)} + learning\_rate \times \frac{\partial}{\partial \theta_i}\ln p(t|\theta)$$

此處將各超參數起始值設定為 `[1, 1, 1, 1]` ，learning rate 設定為 `0.00001` 。調參曲線如下圖所示：



將各超參數起始值設定為 `[3, 5, 6, 5]` ，learning rate 設定為 `0.00001` 。調參曲線如下圖所示：



Discussion

> Explain your findings and make some discussion.

從第二題的配適結果來看，在 `kernel` 的設定上，越複雜越能完美配適 `training data`。雖然目前的參數設定尚未產生明顯的 `overfitting` 現象，但經過嘗試，若將四個參數測定為 `[100, 100, 100, 100]`，則會發生 `train RMSE` 持續下降的同時，`test RMSE` 反向上升，因此 `超參數` 的調整有其必要性。

故在接下來的小節採用 ARD 的方法進行 `超參數` 的優化，這個階段的優化在數次嘗試後發現，`起始值` 的設定有很大的影響，但大致上可以看到第一個參數在數次優化的過程中是傾向越大越好的狀態，其餘參數的變動則不多。

---

2. SVM

$$y(\mathrm{x}) = \sum_{n=1}^{N} \alpha_n t_n k(\mathrm{x}\mathrm{x}_n) = \mathrm{w}^T \mathrm{x} + b$$

$$\mathrm{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathrm{x}_n)$$

optimize `w`, `b`, `{ξ}`

$$\frac{\partial L}{\partial W} = 0 \Rightarrow w = \sum_{n=1}^{N} a_n t_n \phi(\mathrm{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n$$

Analyze two methods

> Analyze the difference between two decision approaches. Decide which one you want to
> use and explain why you choose this approach.

`one-versus-rest`：

這個方法會將每一個要識別的類別視為 `+1`，其餘則為 `-1`。因此在這`三類`的分類問題會產生 `3` 個 SVM，之後再依據將新資料放入個個判別，值最大者就歸屬到該類。

不過也就是這樣一對多的判別，就很容易發生 `資料不平衡` 的情形，還需要其他額外的處理來應付這類問題。

`one-versus-one`：

此方法會從 `3` 個類別中抽出 `2` 個來進行訓練，因此在本題最後會有三個 SVM 模型，之後放入新資料經過各個 SVM 類後依據 `投票` 準則判斷最終歸屬。

此法較不會產生 `資料不平衡` 的問題，但伴隨而來的就是 `運算時間較長`，記`憶體占用較大`。

本題由於類別較少，對於運算時間不太在意，同時也能避免資料不平衡的問題的條件下，應會優先選擇 `ovo` 的方式來進行。

PCA

> to use the principal component analysis (PCA) to reduce the dimension of the data to 2 dimension.

通過 `PCA` 取了兩個主成分後，再利用 `scikit-learn` 內置的標準化函數整理數據。

Linear Kernel

> Use the dataset to build a SVM with `linear kernel` to do multi-class classification.

linear kernel:

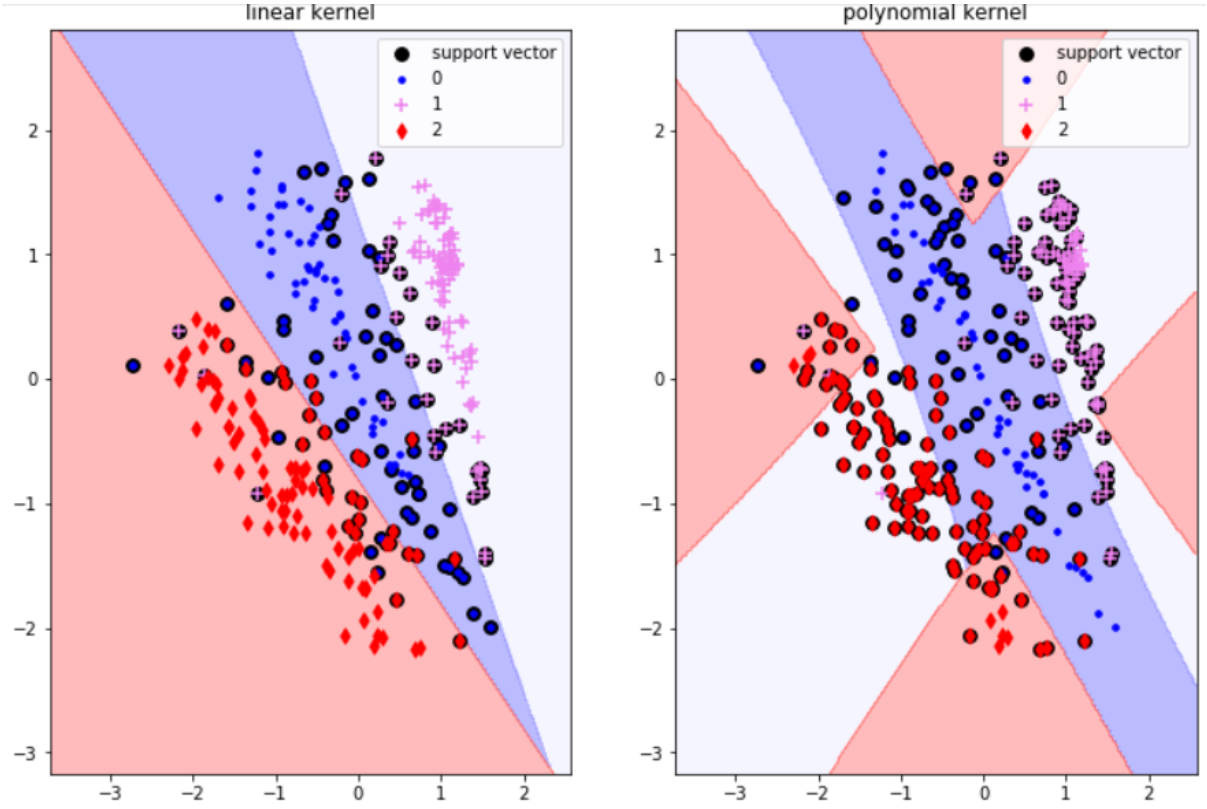
$$k(\mathrm{x}_i, \mathrm{x}_j) = \phi(\mathrm{x}_i)^T \phi(\mathrm{x}_j) = \mathrm{x}_i^T \mathrm{x}_j$$

Polynomial Kernel

> Use the dataset to build a SVM with `polynomial kernel` to do multi-class classification.

polynomial kernel:

$$k(\mathrm{x}_i, \mathrm{x}_j) = \phi(\mathrm{x}_i)^T \phi(\mathrm{x}_j) = (\mathrm{x}_i^T \mathrm{x}_j)^2$$
$$\phi(\mathrm{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]$$
$$\mathrm{x} = [x_1, x_2]$$



從上圖可以看出：

- `linear kernel` 之下，界線為兩條直線。
- `polynomial kernel` 之下，界線較具有複雜的變化性。

Discussion

`linear kernel` 的優點是模型較為簡單，較不易發生 overfitting 的狀況，同時又較具有可解釋性，但缺點就是如果資料呈現非線性的分離情形，在最終分類結果表現就會較差。

`polynomial kernel` 的優點就是可以進行更複雜的分類，更可以將維度況張到更高次方，進行更細緻的分類。但相對的就可能發生 overfitting 的問題，因此在參數的調整上要比較小心。

> 兩種 kernel 並無何者絕對優於另一個的說法，我們需要觀察資料的分布、資料的型態，加以判斷該選用什麼樣的 `參數設定` 。

---

3. Gaissian Mixture Model

K-Mean Model

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}||x_n - \mu_k||^2$$

GMM

$$p(z_k = 1) = \pi_k$$

$$\sum_{k=1}^{K}\pi_k = 1$$

$$p(\mathrm{x}) = \sum_{Z} p(\mathrm{z})p(\mathrm{x}|\mathrm{z}) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\mathrm{x}|\mu_k, \Sigma_k)$$

藉由 `K Mean` 的算法，反覆計算收斂到用 `k 個平均數(顏色)` 來代表整張圖片，接著將計算出來的 `mean` ，導入 `GMM` 的計算，通過 `EM 演算法` 計算出最可能的值。

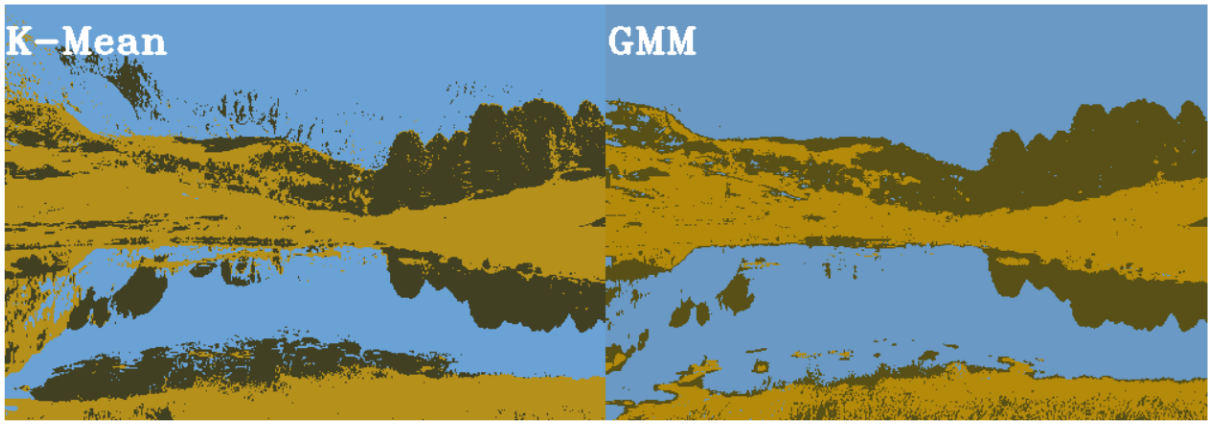
以下將展示在設定 K 依序為 3, 5, 7, 10 時，圖片會呈現什麼樣的面貌。

show RGB table and photo

- `K = 3`

	K-Means	b	g	r
0	0	214	162	106
1	1	27	144	181
2	2	34	64	65

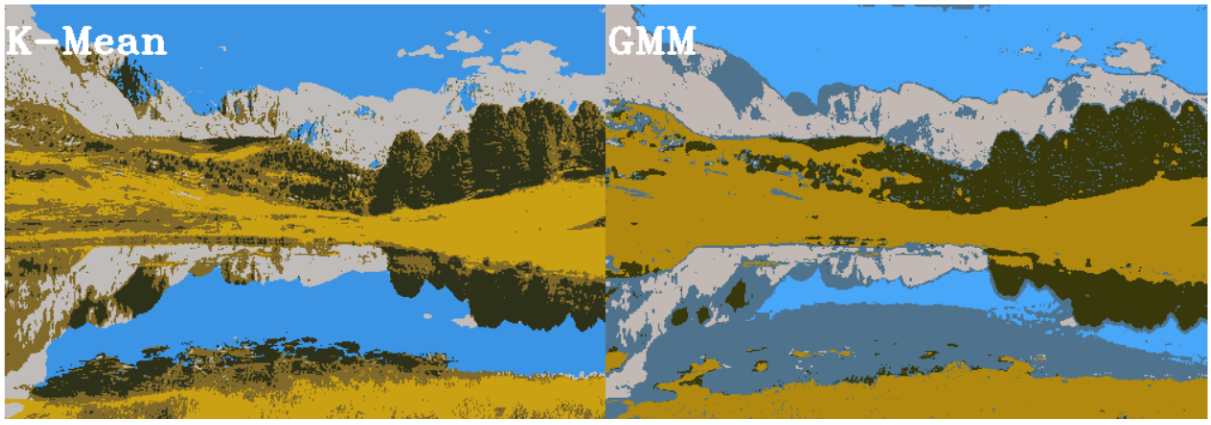
	GMM	b	g	r
0	0	196	153	105
1	1	11	138	180
2	2	22	80	88



- `K = 5`

	K-Means	b	g	r
0	0	228	148	60
1	1	19	161	202
2	2	185	188	192
3	3	41	106	129
4	4	26	49	46

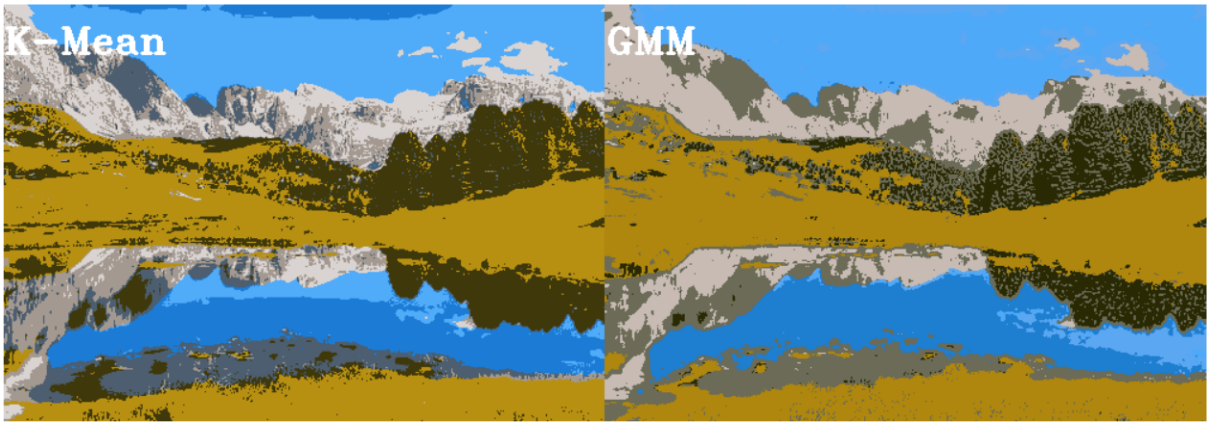
	GMM	b	g	r
0	0	249	167	73
1	1	14	137	177
2	2	180	185	195
3	3	141	115	80
4	4	11	56	56



- `K = 7`

	K-Means	b	g	r
0	0	144	153	164
1	1	11	56	63
2	2	249	171	80
3	3	111	94	76
4	4	18	144	183
5	5	213	124	28
6	6	210	212	217

	GMM	b	g	r
0	0	180	187	199
1	1	3	41	41
2	2	249	169	78
3	3	87	107	107
4	4	14	135	175
5	5	207	126	30
6	6	242	167	91

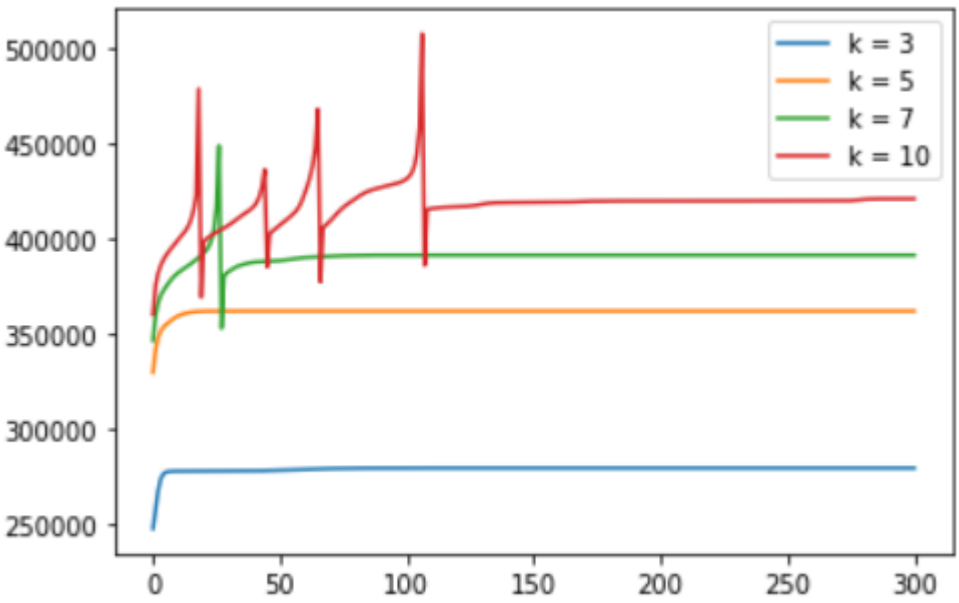


- `K = 10`

	K-Means	b	g	r
0	0	6	26	24
1	1	121	93	68
2	2	17	142	184
3	3	143	152	162
4	4	216	125	28
5	5	16	70	80
6	6	249	171	80
7	7	20	107	139
8	8	209	211	217
9	9	21	188	225

	GMM	b	g	r
0	0	253	180	85
1	1	174	104	35
2	2	33	145	194
3	3	104	115	116
4	4	225	132	18
5	5	235	181	113
6	6	244	148	61
7	7	10	64	69
8	8	182	188	200
9	9	8	137	177

Log-Likelihood plot



上圖為在各 `K值` 的設定之下， `GMM` 中 `log-likelihood` 的變化。

Discussion

從輸出的圖片我們可以看到，整體而言， `K Mean` 的畫面相對於 `GMM` 更能夠呈現近似原始圖片的色彩分布。不過有趣的是，在 `K值` 偏低時， `GMM` 能夠刻畫出樹葉的細節以及山體的輪廓上的特徵，而 `K Mean` 在這些地方則表現的相對「`模糊`」。