3F8: Inference
Full Technical Report

Chia Jing Heng, jhc71

March 18, 2020

Abstract

This coursework aims to implement a Bayesian binary classifier. Building on the maximum likelihood (ML) classifier in the short lab, two other approaches are considered: maximum a posteriori (MAP) and full Bayesian approach. Laplace approximation is used to perform the full Bayesian approach. A grid-search method is then used to tune the hyper-parameters of the model to improve the performance of the full Bayesian approach. It is found that the MAP and full Bayesian approaches generalise and classify better than the ML approach. The MAP and Bayesian approaches have very similar performance.

1 Introduction

In the original coursework, we have performed Maximum Likelihood (ML) approach on logistic classification of the dataset. One problem with ML is that in high dimensions, there may be more than one solutions for the weights, β and it will also lead to overfitting. One way to overcome this is by introducing a prior on the weights to regulate the weights so the model will generalise better. This is the Maximum A Posteriori (MAP) approach.

Another alternative is by performing the full Bayesian approach on the predictive distribution. This is essentially averaging the prediction probability of the new data point over all possible weights using the posterior distribution of the weights. This is done using an integral over β. The problem, however, is that the integral is difficult to integrate. A solution to this is to perform an approximate Bayesian inference.

In this report, the theory of MAP inference and the full Bayesian approach using Laplace approximation is presented briefly. The code implementation are then presented, followed by results, discussions and comparisons between the ML, MAP and full Bayesian approaches. Hyper-parameters are then tuned to improve on the full Bayesian approach and a final comparison is made between the MAP and full Bayesian approach.

2 Exercise a. Theory

2.1 MAP

In the short lab, the objective function to be maximised for the ML problem is the log likelihood, which is given by

$$\begin{aligned}\mathcal{L}(\beta) &= \log \mathrm{p}(y|\tilde{X}, \beta) \\ &= \sum_{n=1}^{N} [y^{(n)} \log \sigma(\beta^{\mathrm{T}} \tilde{x}^{(n)}) + (1 - y^{(n)}) \log(1 - \sigma(\beta^{\mathrm{T}} \tilde{x}^{(n)}))]\end{aligned} \tag{1}$$

For the MAP problem, we assume a prior on the weights with the prior distribution:

$$\mathrm{p}(\beta_i) = \mathcal{N}(\beta_i|0, \sigma_0^2), \text{ for } i = 1, 2, \ldots, M. \tag{2}$$

The objective function to maximise is the log posterior, which is the log likelihood plus a prior term contributed by the prior distribution on the weights, assuming independent β_i's:

$$\begin{aligned}\mathcal{L}(\beta) &= \log \mathrm{p}(\beta|y, \tilde{X}) \\ &= \log \mathrm{p}(y|\tilde{X}, \beta) + \log \mathrm{p}(\beta) \\ &= \sum_{n=1}^{N} [y^{(n)} \log \sigma(\beta^{\mathrm{T}} \tilde{x}^{(n)}) + (1 - y^{(n)}) \log(1 - \sigma(\beta^{\mathrm{T}} \tilde{x}^{(n)}))] - \frac{1}{2\sigma_0^2} \beta^{\mathrm{T}} \beta\end{aligned} \tag{3}$$

Maximising the log posterior gives the MAP solution for β. The predictive probability for new data x_* is then given by $\sigma(\beta_{\mathrm{MAP}}^{\mathrm{T}} x_*)$.

2.2 Full Bayesian approach: Laplace approximation

Instead of just a point solution for the weights, a full Bayesian approach can be used. For the logistic classification problem, the predictive distribution is

$$\begin{aligned}\mathrm{p}(y_*|\tilde{x}_*, y, \tilde{X}) &= \int \mathrm{p}(y_*|\tilde{x}_*, \beta) \mathrm{p}(\beta|y, \tilde{X}) d\beta \\ &= \int \sigma(y_* \beta^{\mathrm{T}} \tilde{x}_*) \mathrm{p}(\beta|y, \tilde{X}) d\beta.\end{aligned} \tag{4}$$

The integral over β, however, is difficult and is often intractable. To go around this, an approximate Bayesian inference could be carried out instead. The posterior $\mathrm{p}(\beta|y, \tilde{X})$ is approximated with a simpler distribution, $\mathrm{q}(\beta)$.

$\mathrm{q}(\beta)$ can be obtained by first expressing the posterior as $\frac{1}{\mathrm{Z}}\mathrm{f}(\beta)$, where Z is a normalisation constant for $\mathrm{f}(\beta)$. Applying Taylor expansion on $\log \mathrm{f}(\beta)$ about the maximum a posteriori solution of β, β_{MAP}, throwing away cubic and higher order terms, and finally taking the exponent, this gives an exponential form for $\mathrm{f}(\beta)$. This is then used to deduce $\mathrm{q}(\beta)$:

$$\mathrm{q}(\beta) = \mathcal{N}(\beta|\beta_{\mathrm{MAP}}, \mathrm{A}^{-1})$$

where A is the negative of the Hessian of $\log \mathrm{f}(\beta)$ evaluated at β_{MAP}. The matrix A can be easily derived from the log posterior and it takes the form:

$$\mathrm{A} = \frac{1}{\sigma_0^2}\mathrm{I} + \sum y_n(1 - y_n)\tilde{x}_n \tilde{x}_n^T. \tag{5}$$

Plugging this approximation into eq (4),

$$\begin{aligned}\mathrm{p}(y_*|\tilde{x}_*, y, \tilde{X}) &\approx \int \sigma(y_* \beta^{\mathrm{T}} \tilde{x}_*) \mathrm{q}(\beta) d\beta \\ &= \int \sigma(a) \mathrm{p}(a) da,\end{aligned} \tag{6}$$

where

$$a = \beta^{\mathrm{T}} \tilde{x}_*$$

and

$$\mathrm{p}(a) = \int \delta(a - \beta^{\mathrm{T}} \tilde{x}_*) \mathrm{q}(\beta) d\beta.$$

Since $\mathrm{q}(\beta)$ is a normal distribution. $\mathrm{p}(a)$ is also a normal distribution. Evaluating the mean and variance of a,

$$\mathrm{p}(a) = \mathcal{N}(a|\beta_{\mathrm{MAP}}^{\mathrm{T}} \tilde{x}_*, \tilde{x}_*^{\mathrm{T}} \mathrm{A}^{-1} \tilde{x}_*).$$

The integral over a in eq (6), however, cannot be evaluated analytically. This is overcome by recognising that a logistic function can be approximated by a probit function: $\sigma(a) \approx \Phi(\lambda a)$ when $\lambda^2 = \pi/8$. Figure 1 shows a logistic curve and a probit curve with $\lambda^2 = \pi/8$.

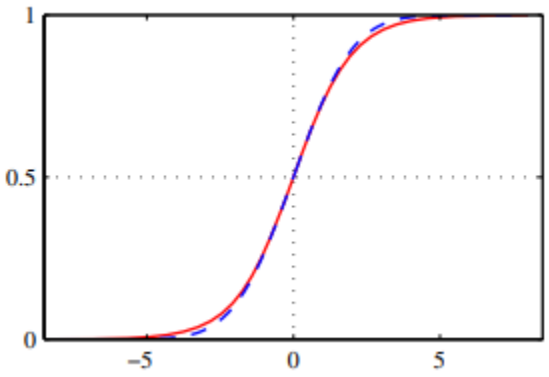

Figure 1: $\sigma(a)$ (shown in red) vs $\Phi(\lambda a), \lambda^2 = \pi/8$ (shown in dashed blue)

The advantage of this approximation is that the integral in eq (6), after approximated, can be expressed analytically as another probit function and this can be approximated by a logistic function:

$$\begin{aligned} \mathrm{p}(y_*|\tilde{x}_*, y, \tilde{X}) &\approx \int \Phi(\lambda a)\mathcal{N}(a|\mu_a, \sigma_a^2)da \\ &= \Phi\left(\frac{\mu_a}{(\lambda^{-2} + \sigma_a^2)^{\frac{1}{2}}}\right) \\ &\approx \sigma\left(\frac{\mu_a}{(1 + \lambda^2\sigma_a^2)^{\frac{1}{2}}}\right). \end{aligned}$$

In short, Laplace approximation can be applied to the logistic classification as follows:

$$\mathrm{p}(y_*|\tilde{x}_*, y, \tilde{X}) \approx \sigma\left(\frac{\mu_a}{(1 + \frac{\pi}{8}\sigma_a^2)^{\frac{1}{2}}}\right) \tag{7}$$

where $\mu_a = \beta_{\mathrm{MAP}}^{\mathrm{T}} x_*$ and $\sigma_a^2 = \tilde{x}_*^{\mathrm{T}} \mathrm{A}^{-1} \tilde{x}_*$. A can be evaluated with eq (5).

2.3 Model Evidence

Since the full Bayesian approach can be interpreted as averaging the predictive probability over all possible β's, it does not make sense to have a log likelihood. The equivalent is therefore the model evidence, $\mathrm{p}(\mathcal{D}) = \int \mathrm{p}(\mathcal{D}|\beta)\mathrm{p}(\beta)d\beta$, where $\mathcal{D}$ represents the dataset.

From Bayes' Theorem, we know that

$$\mathrm{p}(\beta|\mathcal{D}) = \frac{\mathrm{p}(\beta)\mathrm{p}(\mathcal{D}|\beta)}{\mathrm{p}(\mathcal{D})}.$$

The model evidence is then the normalisation constant for the posterior. Recognising $\mathrm{p}(\beta)\mathrm{p}(\mathcal{D}|\beta)$ as $\mathrm{f}(\beta)$ and $\mathrm{p}(\mathcal{D})$ as Z from section 2.2, and using the Taylor approximation of $\mathrm{f}(\beta)$, we have

$$\begin{aligned} \mathrm{p}(\mathcal{D}) &= \mathrm{Z} \\ &= \int \mathrm{f}(\beta)d\beta \\ &\approx \mathrm{f}(\beta_{\mathrm{MAP}})\frac{(2\pi)^{M}/2}{|\mathrm{A}|^{1/2}}. \end{aligned}$$

The log of model evidence is then

$$\log \mathrm{p}(\mathcal{D}) \approx \log \mathrm{p}(\mathcal{D}|\beta_{\mathrm{MAP}}) + \log \mathrm{p}(\beta_{\mathrm{MAP}}) + \frac{M}{2}\log 2\pi - \frac{1}{2}\log|\mathrm{A}| \tag{8}$$

, where M is the dimension of the data input, x.

3 Exercise b. Code Implementation

3.1 Gradient

As explained in section 2.1, the objective function for the MAP problem is the log posterior. The gradient is thus

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta_i} = \sum_{n=1}^{N}(y^{(n)} - \sigma(\beta^T \tilde{x}^{(n)}))\tilde{x}_i^{(n)} - \frac{1}{\sigma_0^2}\beta_i.$$

3.2 MAP

To find the MAP solution, `scipy.optimize.fmin_l_bfgs_b` is used. Since this `scipy` function minimises a function, the objective function is thus the negative of log posterior. The essential code implemented is as follows:

```
#the objective function to MINIMISE
def cost(w, X_tilde, y, var_w):
    #var_w = sigma_0 squared
    output_prob = predict(X_tilde, w)
    try:
        lp = np.dot(y, np.log(output_prob)) +
            np.dot((1 - y), np.log(1.0 - output_prob)) - 0.5*np.dot(w, w)/var_w #log posterior
    except: #to prevent RuntimeWarning due to log(0.)
        X_dot_w = np.dot(X_tilde, w)
        ll = 0
        for i in range(len(output_prob)):
            if output_prob[i] == 0.:
                ll += y[i]*X_dot_w[i]
            elif output_prob[i] == 1.:
                ll += (1 - y[i])*(-X_dot_w[i])
            else:
                ll += y[i]*np.log(output_prob[i]) + (1 - y[i])*np.log(1.0 - output_prob[i])
        lp = ll - 0.5*np.dot(w, w)/var_w #log posterior
    return -lp

def fit_w_map(X_tilde_train, y_train, var_w):
    w = np.random.randn(X_tilde_train.shape[1])
    w_map = scipy.optimize.fmin_l_bfgs_b(cost, w,
        args = (X_tilde_train, y_train, var_w), approx_grad=True)[0]
    return w_map
```

3.3 Full Bayesian approach: Laplace approximation

To implement the full Bayesian approach using the Laplace approximation, the MAP solution for β is first obtained as described in section 3.2. The matrix A is then computed using eq (5). Finally, the predictive probability distribution is evaluated using Laplace approximation as in eq (7). The code is as follows:

```
def compute_A(X_tilde, w_map, var_w):
    output_prob = predict(X_tilde, w_map) #logistic

    A = np.identity(X_tilde.shape[1])/var_w
    for i in range(X_tilde.shape[0]):
        #print("la: {}".format(np.outer(X_tilde[i], X_tilde[i][0])))
        A += output_prob[i]*(1 - output_prob[i])*np.outer(X_tilde[i], X_tilde[i])
    return A

def calc_lap_pred_distn(X_tilde, w_map, var_w):
    mu_a = np.dot(X_tilde, w_map) #(N x 1)

    A = compute_A(X_tilde, w_map, var_w)
    A_inv = np.linalg.inv(A)
    var_a = np.dot(np.dot(X_tilde, A_inv), np.transpose(X_tilde)) #(N x N)
    var_a = np.diag(var_a)

    #Approximate logistic in integral with probit
    #This gives another probit and is approximated with a logistic
    k_var_a = (1 + np.pi*var_a/8)**-0.5

    pred_distn = logistic(k_var_a * mu_a)
    return pred_distn
```

3.4 Model Evidence

The log of model evidence is calculated using eq(8) and the code implemented for this is as follows:

```
def model_evidence(X_tilde, y, w_map, var_w):

    M = X_tilde.shape[1]
    N = X_tilde.shape[0]
    A = compute_A(X_tilde, w_map, var_w)

    ll_ev = N*compute_average_ll(X_tilde, y, w_map) #log likelihood

    #cholesky decomposition to prevent numerical errors
    chol = np.linalg.cholesky(A)
    log_det_A_inv = 2*np.sum(np.log(np.diag(chol)))
    prior_ev = -0.5*M*np.log(var_w) - 0.5*np.dot(w_map, w_map)/var_w - 0.5*log_det_A_inv

    model_ev = ll_ev + prior_ev
    return model_ev
```

4 Exercise c: Visualisation

The code is implemented on the dataset from the coursework exercise with the feature expansion on the inputs using radial basis functions. The width of the rbf, l is set at 0.1.

Maximum likelihood classification is first applied with learning step $\alpha = 0.004$ for 700 steps. The results are compared against those for MAP and full Bayesian approach (Laplace approximation). For MAP and Laplace approximation, the prior on the weights take the form in eq(2). σ_0^2 is set at 1.0 and l is set at 0.1 for the base case.

Figure 2 shows the data with predictive probability contours plotted. As expected, the ML solution tends to overfit. This is evident especially at the decision boundaries. The MAP solution reduces the overfitting and the full Bayesian approach improves on it further. Note the decision boundaries are softer for both MAP and the full Bayesian approach than for the ML solution. Note also that the classifier is less certain about the decision boundaries for the full Bayesian approach than for the MAP solution. This means the classifier generalises better with the full Bayesian approach than with the MAP solution and the ML solution.

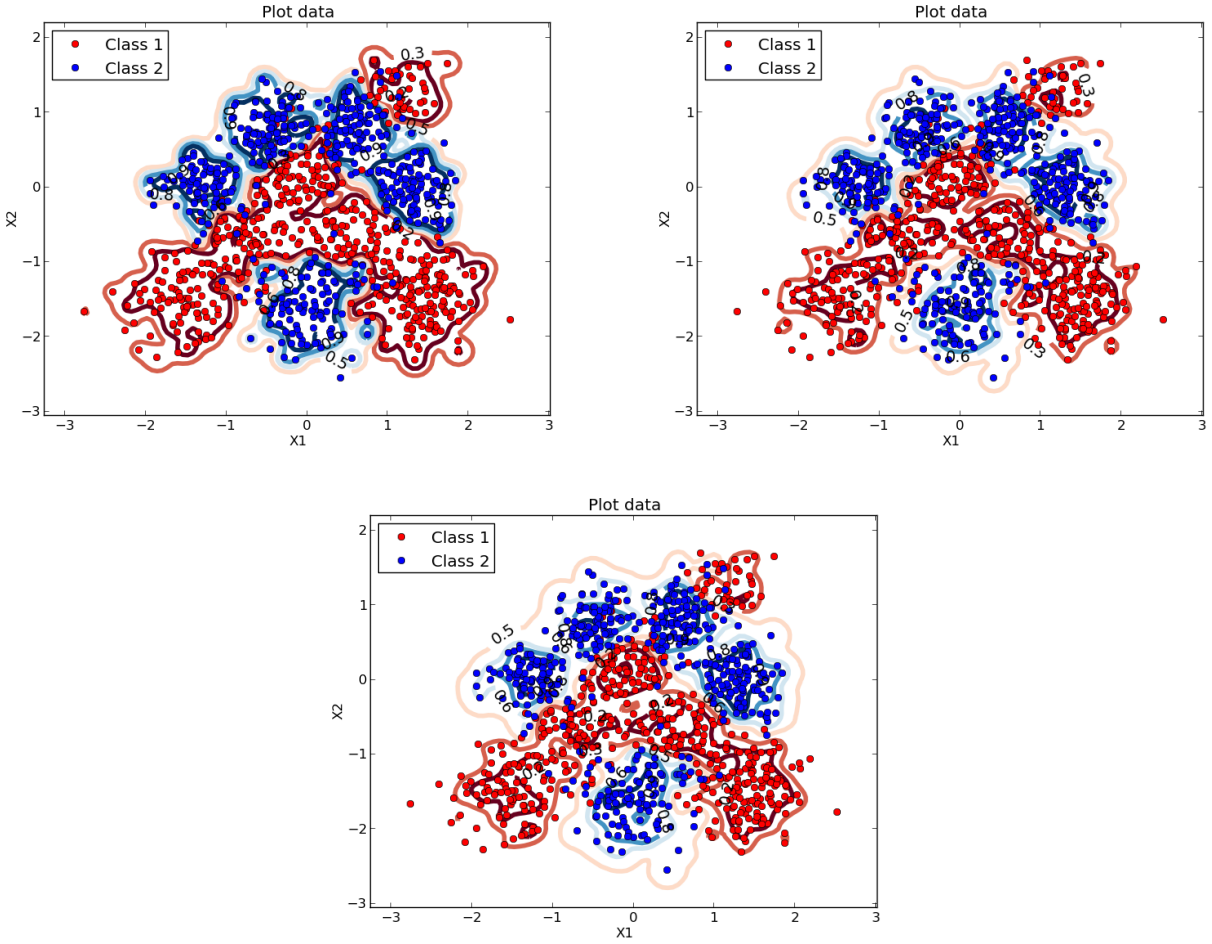

Figure 2: Plots showing data and contour lines for the predictive distribution ($l = 0.1$) generated by the ML solution (top left), the MAP solution (top right) and the Laplace approximation (bottom).

5 Exercise d: LL's and confusion matrices

The average training and test log-likelihoods are evaluated for the three approaches. For the full Bayesian approach, log model evidence is used instead. The results are tabulated in table 1.

The log-likelihoods for the MAP and full Bayesian approach are generally lower than that for the ML approach. This is because the classifier is less overfitting for MAP and full Bayesian. The log-likelihoods for the Laplace approximation (model evidence) are also lower than that for the MAP solution because the Laplace approximation averages over all possible weights and generalises better.

	Avg. Train ll	**Avg. Test ll**
ML	-0.1372	-0.2876
MAP	-0.2204	-0.2965
Laplace	-0.3952	-0.7489

Table 1: Average training and test log-likelihoods for ML, MAP and Laplace approximation (model evidence).

The confusion matrices for three approaches are shown in table 2. It can be seen that there is improvement in the classification for MAP and Laplace over the ML solution. There is especially significant improvement in the proportion of true positives detected. The performance for both MAP and Laplace approach are identical. This shows that it might not be worth the extra computation to perform the full Bayesian approach and just use the MAP solution. However, the hyper-parameters σ_0^2 and l can be tuned to further investigate the difference in performance between the MAP and the full Bayesian approach.

		$\hat{y}$	
		0	1
y	0	0.9158	0.0842
	1	0.1238	0.8762

		$\hat{y}$	
		0	1
y	0	0.9191	0.0808
	1	0.0594	0.9406

		$\hat{y}$	
		0	1
y	0	0.9191	0.0808
	1	0.0594	0.9406

Table 2: Confusion matrix for ML (left), MAP (middle) and Laplace approximation (right) with $\sigma_0^2 = 1.0$ and $l = 0.1$ for MAP and Laplace approximation.

6 Exercise e: Hyper-parameter tuning, Grid search and code implementation

The base case for Laplace approximation is $\sigma_0^2 = 1.0$ and $l = 0.1$. These hyper-parameters can be tuned to improve the performance of the classifier. To do this, a grid-search method is used. 10 different values for both σ_0^2 and l are chosen, giving 100 different combinations of hyper=parameters. The model evidence for each hyper-parameter combination is calculated and they are plotted into a heat map.

For my implementation, I have chosen the ranges $0.01 < \sigma_0^2 < 5.0$ and $0.01 < l < 0.9$. The code implemented are shown below and the resulted heat map is plotted as in figure 3.

```
#GRID SEARCH
var_w_arr = np.linspace(0.01, 5.0, 10)
l_arr = np.linspace(0.01, 0.9, 10)

#initialise grid with zeros
s = (len(var_w_arr), len(l_arr))
model_ev_arr = np.zeros(s)

#update grid
for i in range(len(var_w_arr)):
    for j in range(len(l_arr)):

        X_tilde_train = get_x_tilde(evaluate_basis_functions(l_arr[j], X_train, X_train))

        w_map = fit_w_map(X_tilde_train, y_train, var_w_arr[i], compute_ll = False)
        model_ev = model_evidence(X_tilde_train, y_train, w_map, var_w_arr[i])

        model_ev_average = model_ev/(X_tilde_train.shape[0])
```

```
        model_ev_arr[i][j] = model_ev_average

#PLOT HEATMAP
def plot_heatmap(var_w_arr, l_arr, model_ev_arr, cbarlabel=''):

    fig, ax = plt.subplots()
    im = ax.imshow(model_ev_arr, cmap='inferno', interpolation='nearest')
    cbar = ax.figure.colorbar(im, ax=ax)
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")

    # We want to show all ticks...
    l_arr1 = [round(l_arr[i], 4) for i in range(len(l_arr))]
    var_w_arr1 = [round(var_w_arr[i], 4) for i in range(len(var_w_arr))]
    ax.set_xlabel('l (width of rbf gaussian)')
    ax.set_ylabel('prior var')
    ax.set_xticks(np.arange(len(l_arr1)))
    ax.set_yticks(np.arange(len(var_w_arr1)))
    # ... and label them with the respective list entries
    ax.set_xticklabels(l_arr1)
    ax.set_yticklabels(var_w_arr1)

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

    ax.set_title("Average training model evidence")
    fig.tight_layout()
    plt.show()
```

Figure 3: Heat map plot of the the approximation of the model evidence obtained in the grid search ($0.01 < \sigma_0^2 < 5.0$, $0.01 < l < 0.9$)

From the heatmap, a smaller range is chosen for both σ_0^2 and l to further narrow down the optimal hyper-parameter combination. 15 values are chosen from the range $0.1 < \sigma_0^2 < 4.0$ and $0.1 < l < 0.5$ respectively for the second round of grid search. The resulting heatmap is in figure 4. From the heatmap, the hyper-parameter $(\sigma_0^2, l) = (0.3786, 0.3857)$ is chosen as it maximises the model evidence.

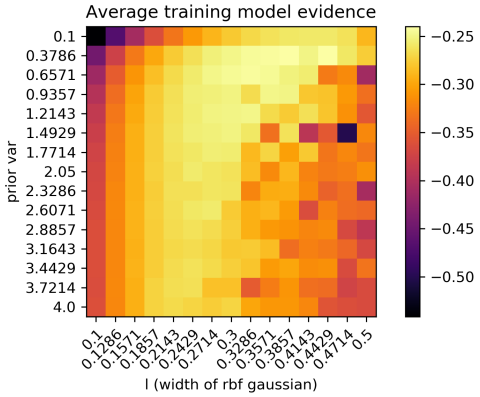

Figure 4: Heat map plot of the the approximation of the model evidence obtained in the grid search ($0.1 < \sigma_0^2 < 4.0$, $0.1 < l < 0.5$)

7 Exercise f: Visualisation, LL's and confusion matrices after tuning

With the hyper-parameter values chosen as $(\sigma_0^2, l) = (0.3786, 0.3857)$, the MAP approach and the full Bayesian approach are applied to the dataset. Figure 5 shows the contours for both approaches. Tables 3 and 4 shows the log-likelihoods and confusion matrices for both approaches.

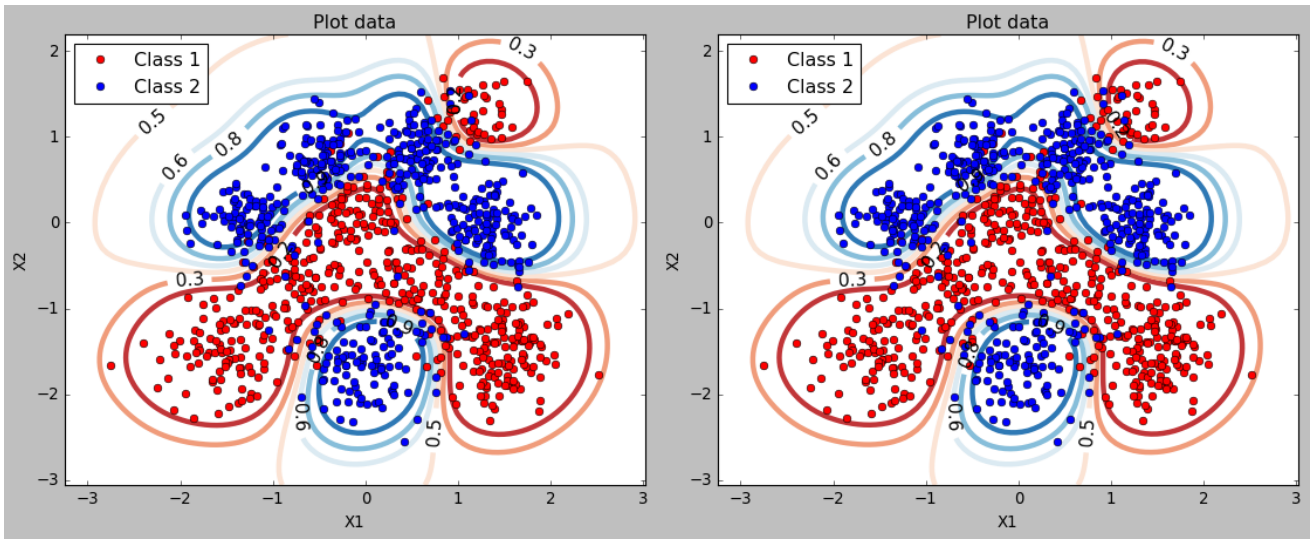

Figure 5: Visualisation of the contours of the class predictive probabilities for MAP (left) and Laplace approximation (right) after hyper-parameter tuning by maximising the model evidence.

The contours in figure 5 for Laplace approximation show an improvement in generalisation as compared to those in figure 2. The same goes to the model evidence in table 3 as compared to that in table 1. When compared to MAP with the same hyper-parameters, however, the performance are still very similar for both approaches. This can be seen from the contours as they are almost identical. The confusion matrices (table 4) for both approaches are identical and do not show much improvement from that in table 2. This shows that it might not be worth the extra computation to perform the full Bayesian approach using Laplace approximation and adequate to just use the MAP solution for logistic classification of this dataset.

	Avg. Train ll	**Avg. Test ll**
MAP	-0.1903	-0.2106
Laplace	-0.2573	-0.4032

Table 3: Average training and test log-likelihoods for MAP and Laplace approximation after hyper-parameter tuning by maximising the model evidence.

		$\hat{y}$	
		0	1
y	0	0.9278	0.0072
	1	0.0874	0.9126

		$\hat{y}$	
		0	1
y	0	0.9278	0.0072
	1	0.0874	0.9126

Table 4: Confusion matrix for MAP (left) and Laplace approximation (right) after hyper-parameter tuning by maximising the model evidence.

8 Conclusions

The MAP and full Bayesian approaches generalise better and classify the data with higher accuracy than the ML approach. The MAP and full Bayesian approaches give identical confusion matrices and similar contours, although the full Bayesian approach give less overfitting contours. After tuning the hyper-parameters σ_0^2 and l, the contours for both approaches improve. However, both approaches have almost identical contours and the confusion matrices (identical for both) show no significant improvement from before tuning. This suggests that MAP approach is adequate for logistic classification of this dataset and there is no need to waste extra computation cost to perform the full Bayesian approach using Laplace approximation.