Logistic Classification

Full Technical Report

Sanchit Gandhi

March 25, 2020

Abstract

Logistic regression is a fundamental statistical learning technique which makes use of supervised machine learning methods for classification tasks. Parameters of linear regression equations are are optimised to give binary outputs (class labels) which minimise prediction error for a subset of the data. Predictive performance is then measured for the rest of the data which makes up the valuation set. This report investigates the use of a probabilistic linear classifier and compares it to an instance of non-linear logistic regression. The non-linear model was implemented through use of non-linear functions of the inputs to generate an feature-expanded input set, without a change to the inference. Average log-likelihoods and confusion matrices were used as metrics for model performance. The linear classifier did not overfit to the training data, but the limitation of linear decision boundaries for a clustered data set meant the predictive accuracy of the model was poor. Non-linear classifiers performed significantly better if the length-scale parameter of the feature functions was set correctly, generalising the training data well. However, the model underfit or overfit if the parameter was chosen outside the optimal range, and gave no indication of predictive uncertainty.

The task of logistic classification was extended by incorporating a prior over the parameters of the model in conjunction with the likelihood function. In doing so, a Laplace approximation allowed for a Gaussian estimate of the posterior distribution to be formed. The logarithm of this distribution was maximised to obtain the maximum a posterior (MAP) point estimate of the parameters. A second set of approximations were performed, giving an approximate Bayesian predictive distribution. This model was shown to have a marginal improvement over the ML estimate in terms of over fitting to the training data.

The optimal hyper-parameters of the Bayesian distribution were found by maximising the model evidence through a grid search approach. With these settings of the parameters, both the MAP and Bayesian solutions exhibited a much lower degree of over fitting, as quantified by comparing the final training and testing log-likelihoods per data point. They also returned higher accuracy hard class probability predictions for a given threshold.

Introduction

Given a two-dimensional dataset with binary class labels, we wish to partition the input space into decision regions for each class, with a measure of confidence (probability) for each decision. The task of logistic classification can be extended to higher dimensional inputs, such as audio or text, for applications such as speech recognition or email spam detection, highlighting the importance of such models. A linear logistic classifier was trained, and its predictive performance compared to that of a non-linear classifier. The non-linear classifier used the same linear regression equations, but with a feature-expanded input set through use of non-linear radial basis functions (RBFs). The novelty of this approach was that non-linear decision boundaries were obtained without a change to the inference.

There are several hurdles which must be overcome to develop a fully Bayesian approach to logistic classification. The first of which is the fact that the Bayesian posterior distribution and its corresponding predictive distribution are intractable. This issue is overcome through use of a Laplace approximation to give a Gaussian estimate of the posterior and predictive expression in the form of a sigmoid function.

The next challenge involves tuning the hyper-parameters of the RBF width and prior distribution. The normalising constant of the posterior distribution is the model evidence for a given training data set and model

parameters. The optimal parameter settings are those which maximise the model evidence. A visual approach to evaluating the model evidence for 100 combinations of these parameters was employed by means of a heat map, and interval reduction used to hone in on the optimal configuration.

Despite the added complexities, a fully Bayesian approach remains a superior method to point estimate models. With appropriately tuned hyper-parameters, issues of over fitting are avoided, and a predictive uncertainty is built into the model through the variance of the distribution. This was shown by comparing the class predictive probability contours, average training and test log-likelihoods, and confusion matrices for the ML, MAP and Bayesian models.

1 Model Theory

We first consider the logistic classification model and derive the gradients of the log-likelihood given a vector of binary labels $\mathbf{y}$ and a matrix of input features $\mathbf{X}$. The gradient of the log-likelihood can be written as:

$$\begin{aligned}\frac{\partial \mathcal{L}(\beta)}{\partial \beta} &= \sum_{i=1}^{N} \frac{\partial \mathcal{L}(\beta)}{\partial \sigma^{(n)}} \frac{\partial \sigma^{(n)}}{\partial \beta} \\ &= \sum_{i=1}^{N} \frac{\left(y^{(n)} - \sigma^{(n)}\right)}{\sigma^{(n)}(1-\sigma^{(n)})} \cdot \tilde{\mathbf{x}}^{(n)} \sigma^{(n)}(1-\sigma^{(n)}) \\ &= \sum_{i=1}^{N} \left(y^{(n)} - \sigma^{(n)}\right) \tilde{\mathbf{x}}^{(n)}. \end{aligned} \tag{1}$$

The scalar term in this expression gives the prediction error for feature $\tilde{\mathbf{x}}^{(n)}$. Gradient ascent is a means of solving the optimisation problem of finding the parameters β that minimise the prediction error for a set of training data. This is done through iteratively stepping up the objective $\mathcal{L}(\beta)$, as the gradients point up the hill, until a convergence criteria is reached or time runs out.

2 Gradient Ascent

A vectorised pseudocode implementation of gradient ascent to estimate the parameters β from the log-likelihood is given as:

```
Function estimate_parameters:

   Input:  feature matrix X ∈ R^{N×D}, labels y ∈ R^N, number of steps n
   Output: vector of coefficients b

   Code:
    Initialise: b ∈ R^{N×(D+1)}, count = 0
    for count < n:
        sigmoid value s = (1 + exp(-b · X))
        b = b + n ·(y - s)^T · X
        count = count + 1
    endfor
    return b
```

The learning rate parameter η is chosen as a trade-off between rate of convergence and achieving convergence itself. In general, increasing the learning rate increases the rate of convergence. However, there is an upper limit to learning rate, beyond which successive iterations move away from the optimum and can then diverge. There is no analytic rule for choosing the optimum learning rate. It is a parameter which must be tuned as the

process is run. One such way of selecting a learning rate is by plotting the log-likelihood as a function of the iteratively determined parameters β, and observing the graph to see whether the method converges and if so at what rate. Oscillatory behaviour is indicative of divergence from the maximum, due to too high a learning rate. The learning rate can then be adjusted to a value which guarantees convergence at a reasonable rate, by increasing if convergence is slow, or decreasing if convergence is not achieved.

3 Dataset Visualisation

Figure 1 is a visualisation of the dataset in the two-dimensional input space, showing each datapoint's class label.

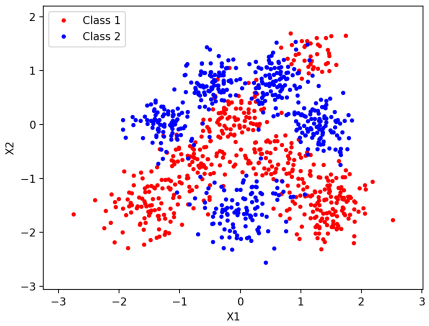


Figure 1: Visualisation of the data.

A classifier with a linear class boundary forms decision regions defined by a set of straight lines, as the decision boundary is orthogonal to the vector of parameters β. For the two class case of logistic regression, this decision boundary is a single straight line. There is little linear separation between each of the classes for the data shown in Figure 1. Therefore, it cannot be separated into two decision regions consisting of outputs of a single class by a single straight line, indicating that a classifier with linear class boundaries is unlikely to perform well.

4 Training the Model

The data was first randomly split into training and test sets with 800 and 200 datapoints respectively. The pseudocode from Section 2 was transformed into the following Python code:

```
def fit_w(X_tilde_train, y_train, n_steps, alpha):
    #Parameters initialised as random values drawn from a standard normal distribution
    w = np.random.randn(X_tilde_train.shape[ 1 ])

    #Classifier trained for fixed number of steps with learning rate alpha
    for i in range(n_steps):
        sigmoid_value = predict(X_tilde_train, w)
        w = w + alpha * np.dot(X_tilde_train.T, y_train - sigmoid_value)
    return w
```

The classifier was then trained using this code, with the learning rate parameter fixed to $\eta = 0.01$. The average log-likelihood on the training and test sets as the optimisation proceeds are shown in Figure 2.

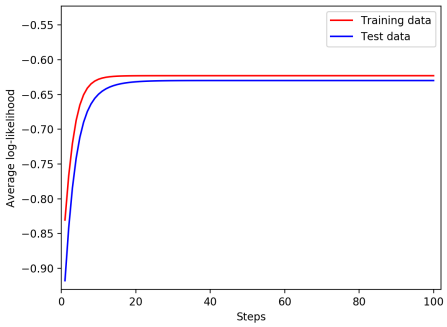


Figure 2: Learning curves showing the average log-likelihood of the training and test (right) datasets

Inspection of these plots confirms that an appropriate learning rate was chosen: the average log-likelihood of the training data increased with number of steps and converged to its maximum. Furthermore, no oscillatory behaviour (which would indicate divergence from the maximum) was present. Hence, the maximum-likelihood estimate of the parameters β was successfully found through gradient ascent.

Results for the average log-likelihoods of the training and test data vary similarly over the optimisation process, demonstrating that the maximum likelihood estimate generated by the model performs comparably for the two data sets. This suggests that the classifier has not overfit to the training data.

Figure 3 displays the visualisation of the contours of the class predictive probabilities on top of the data.

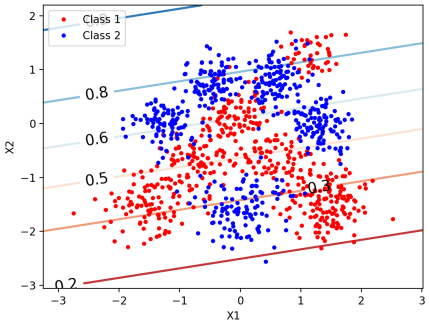


Figure 3: Visualisation of the contours of the class predictive probabilities

This figure further highlights the limitations of a linear classifier for this dataset. Predictive linear probability contours for the trained model run through regions containing data from both classes. Therefore, decision boundaries do not encapsulate the clustered nature of the data. Applying a threshold to the probabilistic predictions so that those greater than $\tau = 1/2$ are assigned a positive predicted class label $\hat{y} = 1$ and those equal or below are assigned to a negative predictive class label $\hat{y} = 0$, Figure 3 shows that a significant portion of data above and below the 0.5 contour is wrongly predicted as false positives and false negatives.

5 Model Performance

The final average training and test log-likelihoods are shown in Table 1. The values for each datasets are to within 1% of each other, indicating very similar levels of performance. These results confirm the deduction made from the average log-likelihood plots in Figure 2 that the model has not overfit to the training data. The 2x2 confusion matrices on the test set is shown in Table 2. By analysing this table, we conclude that the classifier incorrectly predicts the output class label $\hat{y}$ approximately 27% of the time. This is consistent with the predictive probability contours plot in Figure 4, which show roughly this proportion of data from each class lying to the wrong side of the $\tau = 0.5$ threshold. These results further highlight the shortcomings of a linear classifier applied to this dataset, as a significant proportion of predictions are made incorrectly.

Table 1: Average training and test log-likelihoods

Avg. Train ll	**Avg. Test ll**
-0.6229	-0.6299

Table 2: Confusion matrix on the test set

$p(\hat{y}\|y)$		$\hat{y}$	
		0	1
y	0	0.734	0.265
	1	0.273	0.726

6 Basis Function Feature Expansion

The inputs were expanded through a set of non-linear Gaussian radial basis functions (RBFs) centred on the training datapoints. The feature-expanded inputs took the form $\tilde{x}_1^{(n)} = 1$ (to handle the bias terms as before) and:

$$\tilde{x}_{m+1}^{(n)} = \exp\left(-\frac{1}{2l^2}\sum_{\mathrm{d}=1}^{2}\left(\mathrm{x}_{\mathrm{d}}^{(\mathrm{n})} - \mathrm{x}_{\mathrm{d}}^{(\mathrm{m})}\right)^2\right) \tag{2}$$

such that the $(m+1)^{\text{th}}$ feature is given by a radial basis function centred on the m^{th} training datapoint, with width l. The widths considered were $l = \{0.01, 0.1, 1\}$, with corresponding learning rate parameters of $\eta = \{0.0001, 0.001, 0.0001\}$. Figure 4 displays the visualisation of the contours of the resulting class predictive probabilities on top of the data for each choice of $l = \{0.01, 0.1, 1\}$.

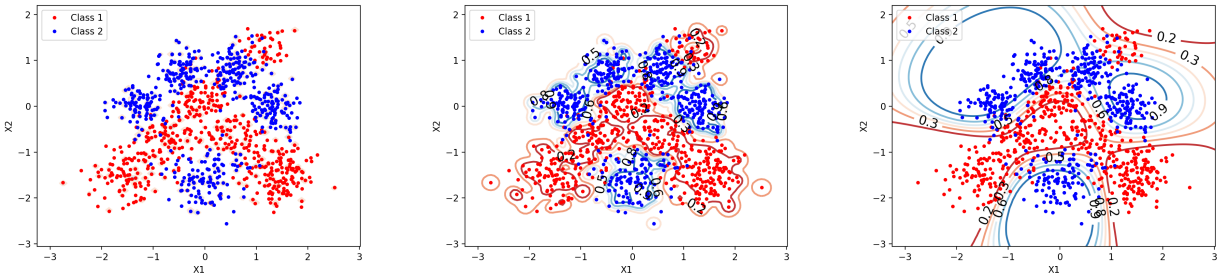


Figure 4: Visualisation of the contours of the class predictive probabilities for $l = 0.01$ (left), $l = 0.1$ (middle), $l = 1$ (right)

The probability contours for the class predictive probabilities are more easily seen for the $l = 0.01$ case by zooming in on a portion of the plot. The contours for a restricted range of inputs X_1 and X_2 are shown in Figure 5.

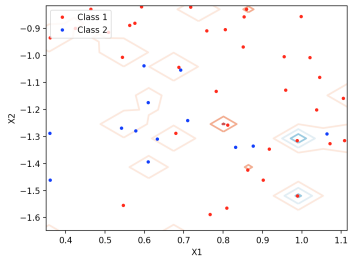


Figure 5: Further visualisation of the contours of the class predictive probabilities for $l = 0.01$

7 RBF Model Performance

The final training and test log-likelihoods per datapoint obtained for each setting of $l = \{0.01, 0.1, 1\}$ are shown in Tables 3, 4 and 5. These results indicate that the width of the RBF plays a central role in the performance of the predictive model.

Avg. Train ll	Avg. Test ll
-0.802	-0.701

Table 3: Results for $l = 0.01$

Avg. Train ll	Avg. Test ll
-0.194	-0.300

Table 4: Results for $l = 0.1$

Avg. Train ll	Avg. Test ll
-0.244	-0.287

Table 5: Results for $l = 1$

The average log-likelihoods for the training and test datasets were significantly lower for the $l = 0.01$ case compared to the other two settings, suggesting that this is a sub-optimal choice of parameter. This is consistent with the probability contours for the class predictive probabilities. Figure 5 reveals that a setting of $l = 0.01$ results in a predictive probability distribution sharply centred about each datapoint of Class 1. This is due to the fact the width of the RBF is too small, meaning the basis functions poorly leverage information between datapoints. Hence, the resulting class predictive probabilities fail to encapsulate the wider distribution of data, and so the classifier is said to have overfit to the training data. A classification model with such narrow contours may be of use in predicting class labels when datapoints are in very close proximity.

The results of Table 4 indicate that the classifier with width parameter $l = 0.1$ still overfits to the training data, as the average log-likelihood for the training data is approximately a factor of 1.5 greater than the test data. This difference demonstrates that the classifier performs significantly better on the training set compared to valuation set. This notion is confirmed by inspecting the class predictive probability contours for $l = 0.1$ in Figure 4, which reveals a set of contours which closely follow the positions of individual datapoints. Supposing the data was generated through a process which involved the addition of random noise, by fitting closely to each datapoint the model has trained itself to the detail of the noise. This impairs its performance on newly seen data, such as that from the test set. Overfitting can be avoided through use of a regulariser, which penalises the magnitude of the parameters and so discourages the model from using extreme weights, or by increasing the RBF width.

The average log-likelihoods were of a similar order for the training and test sets for a RBF width $l = 1$. The contour plot also shows smooth class predictive probabilities distributions which vary from 0 to 1 at the boundaries between regions containing data belonging to different classes. From these observations, we can conclude that the classifier has not overfit to the training data. If anything, the model has slightly underfit: sharper probability transitions at the boundaries would yield improved predictive results. Hence, an appropriate choice of parameter l would lie somewhere between 0.1 and 1.

The 2×2 confusion matrices for the three models trained with $l = \{0.01, 0.1, 1\}$ are show in Tables 6, 7 and 8.

Table 6: Conf. matrix $l = 0.01$.

$p(\hat{y}\|y)$		$\hat{y}$ = 0	$\hat{y}$ = 1
y	0	0.916	0.083
	1	0.891	0.108

Table 7: Conf. matrix $l = 0.1$.

$p(\hat{y}\|y)$		$\hat{y}$ = 0	$\hat{y}$ = 1
y	0	0.913	0.086
	1	0.114	0.885

Table 8: Conf. matrix $l = 1$.

$p(\hat{y}\|y)$		$\hat{y}$ = 0	$\hat{y}$ = 1
y	0	0.904	0.095
	1	0.094	0.905

After analysing these matrices, we can conclude that a width of $l = 0.01$ is a poor choice of parameter: the output prediction is $\hat{y} = 1$ nearly 90% of the time, regardless of the true class label y, giving a high proportion of false negatives. On the other-hand, parameter choices of $l = 0.1$ and 1 result in correct prediction of the class label in approximately 90% of cases, highlighting their superior performance.

Comparing these results to those obtained using the original inputs (documented in Table 2), the number of correctly predicted labels is much higher for the $l = 0.1$ and $l = 1$ cases. It is therefore evident that the use of RBFs to expand the inputs greatly improves the accuracy of the logistic classifier if the function width is chosen correctly, as the model is able to generate non-linear decision boundaries which are better suited to the clustered nature of the dataset compared to linear ones.

8 Bayesian Model Theory

Outline of Model

The model used was the logistic classifier with a feature-expanded input set, as outlined in Equation (2). For a training data set $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the number of model parameters $N = (|\mathcal{D}| + 1)$. Setting out to find a Gaussian form for the posterior distribution, an uncorrelated Gaussian prior over the model parameters was assumed, with zero mean and variance σ_0^2:

$$p(\boldsymbol{\beta}) = \frac{1}{(2\pi\sigma_0^2)^{N/2}} \exp\left(-\frac{1}{2\sigma_0^2}\boldsymbol{\beta}^\top\boldsymbol{\beta}\right). \tag{3}$$

The posterior distribution over $\boldsymbol{\beta}$ is given by Bayes' rule:

$$p(\boldsymbol{\beta}|\mathcal{D}) \propto p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{X})p(\boldsymbol{\beta}). \tag{4}$$

Taking the log of both sides, and substituting for the prior distribution from (4), and log-likelihood of parameters $\mathcal{L}(\boldsymbol{\beta})$:

$$\begin{aligned}\ln p(\boldsymbol{\beta}|\mathcal{D}) &= \mathcal{L}(\boldsymbol{\beta}) + \ln p(\boldsymbol{\beta}) + \text{const.} \\ &= \mathcal{L}(\boldsymbol{\beta}) - \frac{1}{2\sigma_0^2}\boldsymbol{\beta}^\top\boldsymbol{\beta} + \text{const.}\end{aligned} \tag{5}$$

The maximum a posteriori (MAP) estimate is the set of parameters $\boldsymbol{\beta}_{\text{MAP}}$ which maximises the posterior distribution. It is found by computing the gradients of the log of the posterior, denoted by $\mathcal{L}^*(\boldsymbol{\beta})$. Using the result for the gradients of the log-likelihood of the parameters $\frac{\partial}{\partial\boldsymbol{\beta}}\mathcal{L}(\boldsymbol{\beta})$ derived in Equation (1):

$$\frac{\partial\mathcal{L}^*(\beta)}{\partial\beta} = \sum_{n=1}^{N}\left(y^{(n)} - \sigma^{(n)}\right)\boldsymbol{\phi}^{(n)} - \frac{1}{\sigma_0^2}\boldsymbol{\beta} \tag{6}$$

where $\sigma^{(n)} = \sigma(\boldsymbol{\beta}^\top\boldsymbol{\phi}^{(n)})$. The MAP solution is determined in the following exercise through use of a Python library function which performs gradient based minimisation.

Laplace Approximation

A fully Bayesian approach to logistic regression in which a predictive distribution for $\mathbf{y}^*$ is obtained given a new data point $\mathbf{x}^*$ is not possible (Bishop, 2006). The normalising constant of the posterior distribution is found by integrating over the product of the prior distribution and likelihood function, which itself is comprised of a product of logistic sigmoid functions. Since the posterior distribution is no longer Gaussian, the integral over the parameters $\boldsymbol{\beta}$ is intractable. The integral used to generate the predictive distribution is similarly intractable.

The Laplace approximation is a means of approximating a probability density function by a Gaussian distribution centred on the mode of the original distribution. Considering the multivariate distribution $p(\mathbf{x})$ over the M-dimensional space $\mathbf{x}$ defined by:

$$p(\mathbf{x}) = \frac{1}{Z}f(\mathbf{x}) \tag{7}$$

where $Z = \int f(\mathbf{x})\mathrm{d}\mathbf{x}$ is the normalisation constant. The mode of $p(\mathbf{x})$ is the point $\mathbf{x}_0$ at which $\nabla p(\mathbf{x}) = 0$. The covariance matrix is found by through the Taylor expansion of $\ln f(\mathbf{x})$ centred on the mode x_0. Since this is a stationary point, the $\nabla f(\mathbf{x_0})$ term is zero:

$$\ln f(\mathbf{x}) \simeq \ln f\left(\mathbf{x}_0\right) - \frac{1}{2}\left(\mathbf{x} - \mathbf{x}_0\right)^{\mathrm{T}}\mathbf{A}\left(\mathbf{x} - \mathbf{x}_0\right) \tag{8}$$

where the Hessian matrix is defined as

$$\mathbf{A} = -\nabla\nabla\ln f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}.$$

Exponentiating both sides yields:

$$f(\mathbf{x}) \simeq f(\mathbf{x}_0) \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}} \mathbf{A} (\mathbf{x}-\mathbf{x}_0)\right\} \tag{9}$$

which is distribution of the form of a multivariate Gaussian with covariance matrix $\mathbf{A}^{-1}$. The normalised distribution $q(\mathbf{x})$ is obtained through use of the standard normalisation of a multivariate Gaussian, giving the final form of the Laplace approximation:

$$q(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\mathbf{x}_0, \mathbf{A}^{-1}\right). \tag{10}$$

The normalisation constant Z of the distribution $f(\mathbf{x})$ can then be approximated as:

$$\begin{aligned} Z &= \int f(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &\simeq f(\mathbf{x_0}) \int \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}} \mathbf{A} (\mathbf{x}-\mathbf{x}_0)\right\} \mathrm{d}\mathbf{x} \\ &= f(\mathbf{x_0}) \frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}} \end{aligned} \tag{11}$$

where the last line follows from the fact the integrand is Gaussian with normalising constant $\frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}}$. Applying the Laplace approximation to the posterior distribution $p(\boldsymbol{\beta}|\mathcal{D})$, the mode is by definition the MAP estimate $\boldsymbol{\beta}_{\mathrm{MAP}}$, which gives the mean of the Gaussian. The covariance matrix $\mathbf{S}_N$ is found by evaluating the inverse of the Hessian matrix of the negative log posterior:

$$\begin{aligned} \mathbf{S}_N^{-1} &= -\nabla\nabla \ln p(\boldsymbol{\beta}|\mathcal{D}) \\ &= -\nabla \left(\sum_{n=1}^{N} \left(y^{(n)} - \sigma^{(n)}\right) \boldsymbol{\phi}^{(n)} - \frac{1}{\sigma_0^2}\boldsymbol{\beta} \right) \\ &\overset{(a)}{=} \frac{1}{\sigma_0^2}\mathbf{I} - \sum_{n=1}^{N} \frac{\partial\left(\left(y^{(n)} - \sigma^{(n)}\right)\boldsymbol{\phi}^{(n)}\right)}{\sigma^{(n)}} \frac{\partial \sigma^{(n)}}{\partial \boldsymbol{\beta}} \\ &= \frac{1}{\sigma_0^2}\mathbf{I} - \sum_{n=1}^{N} (-\boldsymbol{\phi}^{(n)}) \cdot \sigma^{(n)}(1-\sigma^{(n)}) \boldsymbol{\phi}^{(n)\mathrm{T}} \\ &= \frac{1}{\sigma_0^2}\mathbf{I} + \sum_{n=1}^{N} \sigma^{(n)}(1-\sigma^{(n)}) \boldsymbol{\phi}^{(n)} \boldsymbol{\phi}^{(n)\mathrm{T}} \end{aligned} \tag{12}$$

where (a) follows from the chain-rule. The posterior distribution is thus approximated as:

$$q(\boldsymbol{\beta}|\mathcal{D}) = \mathcal{N}\left(\boldsymbol{\beta}|\boldsymbol{\beta}_{\mathrm{MAP}}, \mathbf{S}_N\right). \tag{13}$$

The predictive distribution is found by marginalising over the posterior distribution:

$$\begin{aligned} p\left(\mathbf{y}^* = 1|\mathbf{x}^*, \mathcal{D}\right) &= \int p\left(\mathbf{y}^* = 1|\mathbf{x}^*, \boldsymbol{\beta}, \mathcal{D}\right) p(\boldsymbol{\beta}|\mathcal{D}) \mathrm{d}\boldsymbol{\beta} \\ &= \int \sigma\left(\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\phi}^*\right) p(\boldsymbol{\beta}|\mathcal{D}) \mathrm{d}\boldsymbol{\beta}. \end{aligned} \tag{14}$$

Approximating the prior distribution as $q(\boldsymbol{\beta}|\mathcal{D})$ and letting $a = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\phi}^*$:

$$\begin{aligned} p\left(\mathbf{y}^* = 1|\mathbf{x}^*, \mathcal{D}\right) &\simeq \int \sigma\left(\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\phi}^*\right) q(\boldsymbol{\beta}|\mathcal{D}) \mathrm{d}\boldsymbol{\beta} \\ &= \int \sigma(a) p(a) \mathrm{d}a \end{aligned} \tag{15}$$

where

$$p(a) = \int \delta\left(a - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\phi}^*\right) q(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta}. \tag{16}$$

Using the property that a marginal distribution of a Gaussian is also Gaussian, the mean and covariance of the distribution $p(a)$ are derived as follows:

$$\begin{aligned} \mu_a = \mathbb{E}[a] &= \int p(a) a \mathrm{d}a \\ &= \int q(\boldsymbol{\beta}) \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\phi}^* \mathrm{d}\boldsymbol{\beta} = \boldsymbol{\beta}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}^* \end{aligned} \tag{17}$$

similarly

$$\begin{aligned} \sigma_a^2 = \mathrm{var}[a] &= \int p(a) \left\{a^2 - \mathbb{E}[a]^2\right\} \mathrm{d}a \\ &= \int q(\boldsymbol{\beta}) \left\{ \left(\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\phi}^*\right)^2 - \left(\boldsymbol{\beta}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}^*\right)^2 \right\} \mathrm{d}\boldsymbol{\beta} \\ &= \boldsymbol{\phi}^{*\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}^* \end{aligned} \tag{18}$$

giving the Gaussian form $p(a) = \mathcal{N}\left(a|\mu_a, \sigma_a^2\right)$. The logistic sigmoid function $\sigma(a)$ is then approximated by a probit function $\Phi(\lambda a)$, with the gradients chosen to be equal at the origin, such that $\lambda^2 = \pi/8$. When convolved with the Gaussian $p(a)$, the resulting distribution is another probit function, which can then be approximated by a corresponding logistic sigmoid function:

$$\begin{aligned} \int \Phi(\lambda a) \mathcal{N}\left(a|\mu_a, \sigma_a^2\right) \mathrm{d}a &= \Phi\left(\frac{\mu_a}{\left(\lambda^{-2} + \sigma_a^2\right)^{1/2}}\right) \\ &\simeq \sigma\left(\frac{\mu_a}{\left(1 + \pi\sigma_a^2/8\right)^{1/2}}\right) \end{aligned} \tag{19}$$

which is the final form of the predictive distribution for $\mathbf{y}^* = 1$. The complementary probability for $\mathbf{y}^* = 0$ is given by $p\left(\mathbf{y}^* = 0|\mathbf{x}^*, \mathcal{D}\right) = 1 - p\left(\mathbf{y}^* = 1|\mathbf{x}^*, \mathcal{D}\right)$.

The normalising constant $p(\mathbf{X})$ in the expression for the posterior distribution of parameters $p(\boldsymbol{\beta})$ is found by applying the result of Equation (11):

$$\begin{aligned} p(\mathbf{X}) &\simeq f(\boldsymbol{\beta}_{\mathrm{MAP}}|\mathcal{D}) \frac{(2\pi)^{N/2}}{|\mathbf{S}_N^{-1}|^{1/2}} \\ \ln p(\mathbf{X}) &\simeq \ln f(\boldsymbol{\beta}_{\mathrm{MAP}}|\mathcal{D}) + \frac{N}{2} \ln 2\pi - \frac{1}{2} \ln |\mathbf{S}_N^{-1}|. \end{aligned} \tag{20}$$

From Equation (4), $\ln f(\boldsymbol{\beta}|\mathcal{D})$ can be written as:

$$\begin{aligned} \ln f(\boldsymbol{\beta}|\mathcal{D}) &= \ln p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{X}) + \ln p(\boldsymbol{\beta}) \\ &= \mathcal{L}\left(\boldsymbol{\beta}_{\mathrm{MAP}}\right) - \frac{N}{2} \ln(2\pi\sigma_0^2) - \frac{1}{2\sigma_0^2} \boldsymbol{\beta}_{\mathrm{MAP}} \boldsymbol{\beta}_{\mathrm{MAP}}^{\top}. \end{aligned} \tag{21}$$

Substituting this into the expression for the model evidence and noting $|\mathbf{S}_N^{-1}| = |\mathbf{S}_N|^{-1}$:

$$\ln p(\mathbf{X}) \simeq \mathcal{L}\left(\boldsymbol{\beta}_{\mathrm{MAP}}\right) - N \ln \sigma_0 - \frac{1}{2\sigma_0^2} \boldsymbol{\beta}_{\mathrm{MAP}} \boldsymbol{\beta}_{\mathrm{MAP}}^{\top} + \frac{1}{2} \ln |\mathbf{S}_N| \tag{22}$$

where $\mathcal{L}(\boldsymbol{\beta}_{\mathrm{MAP}})$ is the log-likelihood of the MAP estimate of the parameters.

9 Bayesian Model Implementation

The MAP solution was obtained through use of the SciPy library function `scipy.optimize.fmin_l_bfgs_b`. This function takes an objective function and its first derivative as arguments, and performs gradient based minimisation, returning an estimate of the position and value of the minimum. Since the MAP estimate maximises the log of the posterior $\mathcal{L}^*(\boldsymbol{\beta})$, the SciPy function was implemented with objective function $-\mathcal{L}^*(\boldsymbol{\beta})$ and first derivative $-\frac{\partial \mathcal{L}^*(\beta)}{\partial \beta}$, determined from the functions `post` and `post_grad` respectively. The function `covariance` returns the covariance matrix of the Laplace approximation, calculated as the inverse of the Hessian.

```
def post(w, *args):
    prior_var, X, y =
    args[0],args[1],args[2]
    sigmoid = predict(X, w)
    L = np.dot(y, np.log(sigmoid)) +
        np.dot((1 - y), np.log(1 - sigmoid))
    if prior_var != 0:
        L -= 1 / (2 * prior_var) * w.T @ w
    #minimising the negative of L*
    return -L
```

```
def post_grad(w, *args):
    prior_var, X = args[0],args[1]
    sigmoid = predict(X, w)
    dL = np.dot(X_tilde_train.T, y_train -
    sigmoid)
    if prior_var != 0:
        dL -= 1/prior_var * w
    return -dL
```

```
def map(prior_var, X, y, n_steps):
    #initialse starting weights
    w = np.random.randn(X.shape[1])
    w_map =
    scipy.optimize.fmin_l_bfgs_b(func=-post,
    x0=w, fprime=-post_grad,
    args=(prior_var, X, y), maxiter=n_steps)
    return w_map[0]
```

```
def covariance(w, prior_var, X):
    sigmoid = predict(X, w)
    scalar = sigmoid - sigmoid * sigmoid
    hess = 1 / prior_var *
    np.identity(X.shape[1])
    for x, y in zip(X, scalar):
        hess += y * np.outer(x, x)
    S_n = np.linalg.inv(hess)
    return S_n
```

Equipped with the two terms which define the Laplace approximation of $q(\boldsymbol{\beta}|\mathcal{D})$, the Bayesian predictive distribution for the data was found by calculating the required mean and covariance of $p(a)$, and substituting into the approximate form of the distribution derived in Equation (19).

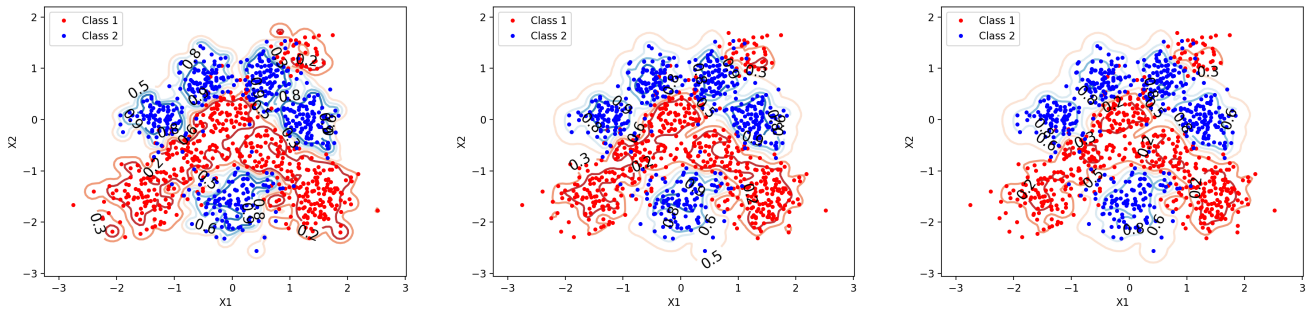
```
def bayesian_predictive(X, w_map, S_n):
    mu = X @ w_map
    var = np.diag(X @ (S_n @ X.T))
    den = 1 + 1/8 * np.pi * var * var
    k_prob = mu * den ** -0.5
    return logistic(k_prob)
```

The model evidence defined in Equation (22) involves calculation of the determinant of the covariance matrix $\mathbf{S}_N$. A robust means of calculating this determinant was implemented through use of the NumPy library function `numpy.linalg.slogdet`, which calculates the logarithm of the determinant. In doing so, overflow or underflow issues caused by a very small or very large determinant are avoided.

```
def model_evidence(w_map, S_n, X, y, prior_var, N):
    # log-likelihood - prior term - map term
    p_x = - post(w_map, 0, X, y) - N / 2 * np.log(prior_var)
          - 1 / (2 * prior_var) * w_map.T @ w_map
    (sign, logdet) = np.linalg.slogdet(S_n)
    # cov term
    p_x -= 0.5 * np.abs(logdet)
    return p_x
```

10 Bayesian Predictive Visualisation

The class predictive probability contours for the ML, MAP and Bayesian cases are shown in Figure 6. For all three cases, a RBF width of $l = 0.1$ was used and the models were trained using 800 training data points. The MAP and Bayesian approaches used a prior on the parameters with variance $\sigma_0^2 = 1$.



(a) ML (b) MAP (c) Bayesian

Figure 6: Visualisation of the contours of the class predictive probabilities

Without a regulariser which penalises large magnitudes of parameter weights, or prior which enforces the parameters to be small, there is no provision to stop the ML model overfitting to the training data. For the case shown in Figure 6, the class predictive probability plot reveals a set of contours which closely follow the positions of individual data points. Supposing the data was generated through a process which involved the addition of random noise, by fitting closely to each data point the model has trained itself to the detail of the noise. This impairs its performance on newly seen data, such as that from the test set.

Like the ML estimate, the MAP method returns a point estimate of the parameters. However, the MAP estimate is formed through use of the likelihood function as well as a prior distribution over the parameters. In doing so, the likelihood is weighted by the prior in the MAP estimate. By selecting a prior distribution which favours smaller parameters $\boldsymbol{\beta}$, the prior has a 'regularising' effect on the likelihood, reducing the degree of over fitting compared to the ML case. The result is a set of predictive probability contours which fit spurious patterns in the data less closely.

The MAP estimate produces predictive contours which are constant along the decision border. A Bayesian approach yields a predictive distribution with increased uncertainty in regions where there is less data, the uncertainty being linked to the variance of the distribution in the corresponding region. This is explained by Equation (14), which shows that the predictive probability for a new data point is the predictive distribution weighted by the posterior distribution over the parameters, integrated over the entire space.

The significance of the prior distribution is suppressed as the number of training data points N is increased, with the Bayesian solution tending towards the MAP estimate as N tends to infinity. Considering the form of the Hessian matrix in Equation (12), and noting that from their definitions, $0 < \sigma^{(n)} < 1$ and $0 < \boldsymbol{\phi}^{(n)}$ for all n, the Hessian is a positive definite matrix. As N is increased, the entries of the Hessian matrix grow larger in magnitude. As a result, the covariance matrix $\mathbf{S}_N$, defined as the inverse of the Hessian, becomes smaller. The variance of the distribution $p(a)$ is given by $\sigma_a^2 = \boldsymbol{\phi}^{*\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}^*$, which by the effect of a smaller $\mathbf{S}_N$, also decreases, tending to zero as N tends to infinity. For the limiting case, the normal distribution $p(a)$ tends to a delta function, centred on $\mu_a = \boldsymbol{\beta}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}^*$. By the sifting property of the delta function, Equation (15) reduces to:

$$\begin{aligned} p\left(\mathbf{y}^* = 1 | \mathbf{x}^*, \mathcal{D}\right) &\simeq \int \sigma(a) p(a) \mathrm{d}a \\ &= \int \sigma(a) \delta\left(a - \boldsymbol{\beta}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}\right) \mathrm{d}a \\ &= \sigma(\boldsymbol{\beta}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}) \end{aligned} \tag{23}$$

which is the MAP estimate of parameters. As the training set used to generate the plots in Figure 6 was relatively large (N=800), the prior has little influence on the shape of the posterior density, giving Bayesian class predictive probability contours similar to that of the MAP case.

11 Bayesian Model Performance

Tables 9 gives the final log-likelihoods per data point for the MAP and Bayesian predictive models.

Table 9: Final log-likelihoods per data point

	Avg. Train ll	**Avg. Test ll**
MAP	-0.212	-0.317
Bayesian	-0.268	-0.349

As explained previously, the Bayesian approach is formed as the predictive distribution weighted by the posterior distribution of the parameters, integrated over the entire space. In doing so, regions with less data have greater predictive uncertainty. This is contrary to the MAP approach, which is simply a point estimate of the parameter settings. The final log-likelihoods per data point are therefore lower for the Bayesian case, with uncertainties in the parameter settings leading to increased uncertainty in predictions.

A threshold to the probabilistic predictions was set such that those greater than $\tau = 0.5$ were assigned a positive predictive class label of $\hat{y} = 1$, and those less than or equal to the threshold a negative predictive class label $\hat{y} = 0$. Tables 10(a) and 10(b) show the 2 x 2 confusion matrices for the MAP and Bayesian methods for the training data set.

Table 10: Confusion matrices for the training data set

(a) MAP

$p(\hat{y}\|y)$		$\hat{y}$ = 0	$\hat{y}$ = 1
y	0	0.916	0.084
	1	0.144	0.856

(b) Bayesian

$p(\hat{y}\|y)$		$\hat{y}$ = 0	$\hat{y}$ = 1
y	0	0.916	0.084
	1	0.144	0.856

Since the Laplace approximated posterior distribution was obtained using $\boldsymbol{\beta}_{MAP}$ as the mode of the distribution, the MAP and Bayesian models have the same mean. They therefore share the same probability contours for $p(y^{(n)} = 1|\tilde{\mathbf{x}}^{(n)}) = 0.5$, and hence have identical decision boundaries. As a result, the hard predicted class probabilities are the same for each case. The proportion of false negatives is rather high, the position of the hard decision boundary lying to the wrong side of a substantial portion of the data with true label $y = 1$, namely through over fitting to the training data set.

12 Hyperparameter Optimisation

A grid search method was used to find the optimal parameters of the Bayesian predictive model. The optimal hyper-parameters (l^*, σ_0^*) are those which maximise the model evidence:

$$(l^*, \sigma_0^*) = \arg\max_{(l,\sigma_0)} \ln p(\mathbf{X}). \tag{24}$$

An initial grid of size 10 x 10 grid was created by selecting 10 values each of σ_0 and l, and the model evidence evaluated at each point on the grid. Interval reduction was then used to further tune the parameters, by repeating the grid search method with a a smaller range of grid axis values known to contain the maximum.

The parameters were first chosen to vary logarithmically, due to the fact they both appear as exponentiated terms in the expression for the posterior distribution. Previous analysis of the ML logistic classification model with feature expanded inputs using RBF widths of $l = \{0.01, 0.1, 1\}$ showed a setting of $l = 0.1$ exhibited a certain degree of over fitting, whilst $l = 1$ slightly under fit the data. Thus, the optimal setting of the hyper-parameter was known to lie somewhere in between. The initial RBF width values were therefore chosen to take values in the range $0.1 < l < 1$.

As σ_0 is increased, the prior becomes more diffuse. Beyond a certain point, the reduction of the regularising effect on the likelihood causes the posterior distribution to become more spread, giving a lower posterior normalising constant, and hence lower model evidence. The range of σ_0 for the gird search approach was found by experimentation, namely through noting that the model evidence quickly became small with values of σ_0 greater than ≈ 1.2 for the range of l tested. The prior standard deviation was therefore chosen to lie in the range $0.5 < \sigma_0 < 1.2$.

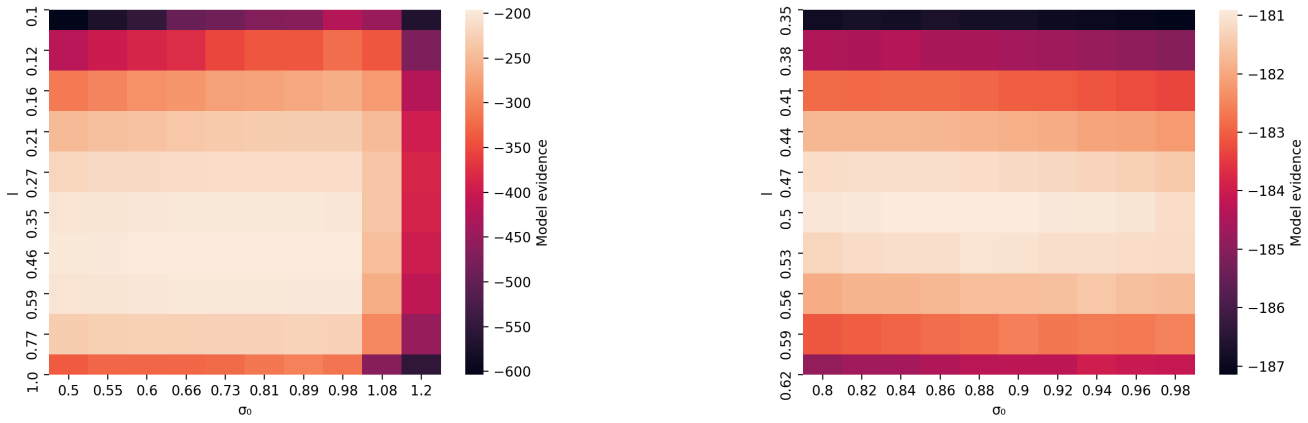
The following Python code outlines how the model evidence was evaluated for each point on the heat map:

```
l = np.logspace(-1, 0, num=10)
sigma = np.logspace(np.log10(0.5), np.log10(1.2), num=10)

def heat_map(X_train, y_train, l, prior_var, n_train, n_steps):
    me = []
    for i in range(len(l)):
        # feature expand inputs
        X_tilde_train = get_x_tilde(evaluate_basis_functions(l[i], X_train, X_train))
        me_l = []
        for j in range(len(prior_var)):
            # evaluate map estimate, covariance and model evidence for each (l, prior_var)
            w_map = map(prior_var[j], X_tilde_train, y_train, n_steps)
            S_n = covariance(w_map, prior_var[j], X_tilde_train)
            me_l.append(model_evidence(w_map,S_n,X_tilde_train,y_train,prior_var[j],
            n_train))
        me.append(me_l)
    return me

me = heat_map(X_train, y_train, l, sigma**2, n_train, n_steps)
```

The heat map was generated through use of the Seaborn library function `seaborn.heatmap`, and is shown in Figure 7(a). The maximum model evidence was found to occur for parameter settings of $(l, \sigma_0) = (0.46, 0.89)$. Interval reduction was then employed to tune the arguments further, by noting that the maximum model evidence must occur for $0.35 < l < 0.59$ and $0.81 < \sigma_0 < 0.98$. The grid search was then repeated with these ranges, the result of which is shown in Figure 7(b).



(a) Initial grid search

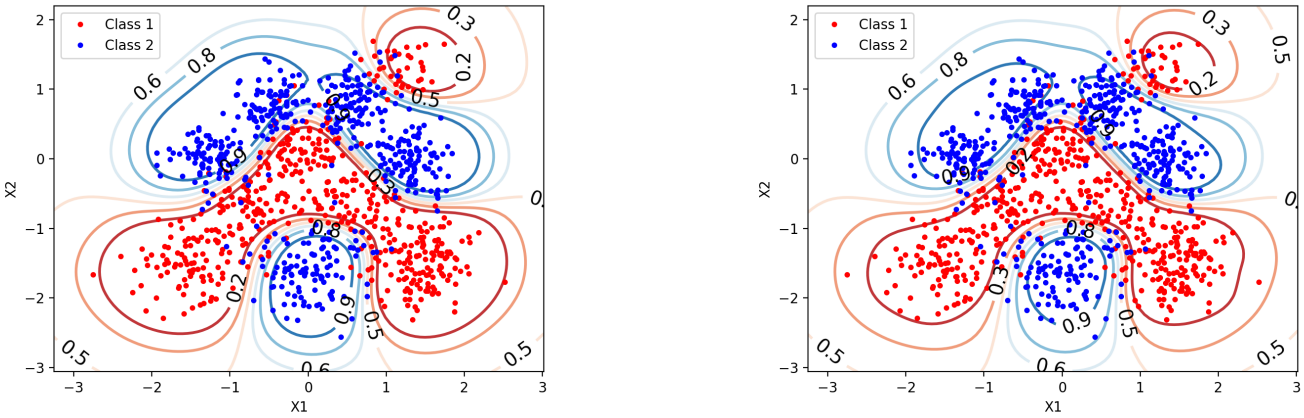
(b) Interval reduced range

Figure 7: Heat map plot of the the approximation of the model evidence

The optimal tuned hyper-parameters which maximised the model evidence were found to be $(l^*, \sigma_0^*) = (0.50, 0.86)$

13 Performance Under Optimal Hyperparameters

The class predictive probabilities were then plot using the optimal parameter settings and are shown in Figure 8.



(a) MAP

(b) Bayesian

Figure 8: Visualisation of the contours of the class predictive probabilities following hyper-parameter tuning by maximising the model evidence

Compared to the contours of the class predictive probabilities with the un-tuned hyper-parameters, both the MAP and Bayesian models exhibit a much lower degree of over fitting. As the RBF width was increased from 0.1 to 0.5, the basis functions better leverage information between data points, giving a set of wider-spread contours. With a smaller variance, the prior distribution also has a greater 'regularising' effect on the likelihood, favouring smaller parameters $\boldsymbol{\beta}$. The combination of these two factors results in less fitting to individual data points, giving class predictive probabilities that better encapsulate the wider distribution of data. More evident than before

is the effect of uncertainty in the Bayesian case where less data is observed. Compared to the MAP estimate model, which produces contours constant along the decision border, the Bayesian predictive probability contours are more spread in regions containing less data.

The notion of reduced over fitting is further highlighted by examining the final training and test log-likelihoods per data point following training on 800 data points, as shown in Table 11.

Table 11: Final log-likelihoods per data point following hyper-parameter tuning by maximising the model evidence

	Avg. Train ll	**Avg. Test ll**
MAP	-0.178	-0.206
Bayesian	-0.197	-0.221

Following hyper-parameter tuning, the average training log-likelihoods for both models improved in the order of $\approx$ 20%. A more substantial change was observed for the average testing log-likelihoods, which increased from -0.317 to -0.206 and -0.349 to -0.221 for the MAP and Bayesian cases respectively. With much closer final log-likelihoods between the training and test data sets, the predictive models obtained using the optimal hyper-parameter settings conclusively over fit to the training data to a lower degree than the initial settings of $l = 0.1$ and $\sigma_0^2 = 1$. Table 12 shows the 2 x 2 confusion matrix for the training data set. As discussed previously, the hard class predictions using a threshold of $\tau = 0.5$ were the same for both the MAP and Bayesian methods.

Table 12: Confusion matrix for the training data set following hyper-parameter tuning by maximising the model evidence

$p(\hat{y}\|y)$		$\hat{y}$	
		0	1
y	0	0.959	0.041
	1	0.107	0.893

Using the optimal hyper-parameters resulted in solutions which outperformed the previous models for both the proportion of false negatives and positives. The greater predictive accuracy is again due to the lower degree of over fitting, giving decision boundaries which better reflect the global distribution of the data set.

Conclusion

In implementing a maximum likelihood logistic classifier to tackle the task of classification of a two-dimensional dataset, the following conclusions were made:

1. Gradient ascent is an iterative means by which the parameters that minimise the prediction error for a classifier can be determined.
2. The learning rate must carefully be selected by examining average log-likelihood plots to ensure the optimisation process converges at an appropriate rate.
3. A classifier with linear class boundaries was not best suited to the dataset, as linear decision regions poorly describe the grouped distribution of the data.
4. Radial basis functions provided a means of expanding the feature set to give non-linear predictive probability contours without changing the inference. The width of these functions had to be chosen appropriately to achieve a model which generalises well. Too small a width resulted in poor leverage of information between datapoints, and so overfit to the training data. Whereas too large a width failed to capture local features of the distribution of data, hence underfitting.

A Bayesian approach to logistic classification overcomes many of the major shortcomings associated with the maximum likelihood method, although doing so brought about limitations of its own:

5. The Laplace approximation is an effective means of tackling the issue of the Bayesian model and its corresponding predictive distribution taking intractable forms. However, this requires calculation and inversion of the Hessian of the original distribution, a computationally expensive process.

6. Use of a prior distribution which favours smaller parameters prevents the predictive model over fitting to the training data. The parameters of the prior must be tuned through an optimisation process which maximises the model evidence in order to deliver the best results, thus adding another computationally demanding procedure to the program code.

7. The variance of the Bayesian distribution, associated with the density of the data in a given region, results in an uncertainty being incorporated into the class predictions. This uncertainty is lost with the MAP estimate, the contours remaining constant along the decision boundary.

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.