Bayesian Logistic Classification

3F8: Inference Coursework
Theo Brown
Selwyn College, University of Cambridge

March 22, 2022

Abstract

Abstract goes here

1 Introduction

2 Model definition

2.1 Logistic classification

The logistic classifier is a binary classifier that takes N D-dimensional inputs $\{\boldsymbol{x}_n\}$ and outputs class labels $\{y_n\}$, $y_i = \{0, 1\}$, where the class labels are modelled as being independent and identically generated from a Bernoulli distribution:

$$\begin{aligned} p(y_n = 1|\tilde{\boldsymbol{x}}_n) &= \sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n) \\ p(y_n = 0|\tilde{\boldsymbol{x}}_n) &= 1 - \sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n) = \sigma(-\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n) \end{aligned} \tag{1}$$

with $\tilde{\boldsymbol{x}}_n = \left[1, \boldsymbol{x}_n^T\right]^T$, $\boldsymbol{w}$ as a vector of D+1 model weights, and $\sigma(x) = \frac{1}{1+e^{-x}}$ (the logistic function).

2.2 Bayesian logistic classification

A posterior distribution of the model weights is required to perform fully Bayesian classification:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{X}|\boldsymbol{w}, \boldsymbol{y})p(\boldsymbol{w})}{p(\boldsymbol{X}|\boldsymbol{y})} \tag{2}$$

From the definition of the logistic classifier, the log-likelihood of $\mathbf{w}$ is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}) = \log p(y|\tilde{\boldsymbol{X}}, \boldsymbol{w}) &= \log \prod_{n=1}^{N} \sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n)^{y_n}\sigma(-\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n)^{1-y_n} \\ &= \sum_{n=1}^{N} y_n \log\sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n) + (1-y_n)\log\sigma(-w^T\tilde{\boldsymbol{x}}_n) \end{aligned} \tag{3}$$

The prior distribution of the model weights is chosen to be Gaussian, $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{m}_0, \boldsymbol{S}_0)$. This gives the log-prior as:

$$\mathcal{P}(\boldsymbol{w}) = \log p(\boldsymbol{w}) = -\frac{1}{2}(\boldsymbol{w} - \boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w} - \boldsymbol{m}_0) + \text{const} \tag{4}$$

The log-posterior is:

$$\log p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{L}(\boldsymbol{w}) + \mathcal{P}(\boldsymbol{w}) + \text{const} \tag{5}$$

Calculating the model evidence $p(\boldsymbol{X}|\boldsymbol{y})$ exactly requires integrating the product of the likelihood and the prior, which is intractable. We present two methods to circumvent this problem: MAP estimation and the Laplace approximation. The performance of these methods is discussed in Section 4.

2.2.1 'Semi-Bayesian' classification: MAP estimation

One method of performing Bayes-driven logistic classification is to use the MAP estimate of the model weights, $\boldsymbol{w}_{\text{MAP}}$, which can be found by applying gradient ascent to the log-posterior. This method does not require the model evidence, but discards a lot of the information contained in the posterior and does not give a distribution of the model weights. As such, it is not 'true' Bayesian classification. The gradient of the log-posterior is calculated as follows:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{w}} \log p(\boldsymbol{w}|\tilde{\boldsymbol{X}}, \boldsymbol{y}) &= \frac{\partial}{\partial \boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) + \frac{\partial}{\partial \boldsymbol{w}} \mathcal{P}(\boldsymbol{w}) \\ &= \sum_{n=1}^{N} \left(y_n - \sigma(\boldsymbol{w}^T \tilde{\boldsymbol{x}}_n)\right) \tilde{\boldsymbol{x}}_n + \boldsymbol{S}_0^{-1}(\boldsymbol{w} - \boldsymbol{m}_0) \\ &= \tilde{\boldsymbol{X}}(\mathbf{y} - \sigma(\tilde{\boldsymbol{X}}^T \boldsymbol{w})) + \boldsymbol{S}_0^{-1}(\boldsymbol{w} - \boldsymbol{m}_0) \end{aligned} \tag{6}$$

where $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1 \dots \tilde{\boldsymbol{x}}_N]$. A standard gradient-based solver can be used with Equation 6 to find a value for $\boldsymbol{w}_{\text{MAP}}$, which can be used as a setting for the weights in Equation 1 to classify data points.

2.2.2 'True' Bayesian classification: the Laplace approximation

The Laplace approximation allows us to perform 'true' Bayesian classification. In this method, approximations to the posterior distribution and the model evidence can be found by finding a Gaussian distribution $q(\boldsymbol{w})$ that closely models the posterior $p(\boldsymbol{w}|\tilde{\boldsymbol{X}}, \boldsymbol{y})$ around a local maximum. The normalising constant of a Gaussian is well defined, so the calculation of the approximate model evidence is simple. For brevity, rewrite Equation 2 as:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{f(\boldsymbol{w})}{K} \approx q(\boldsymbol{w})$$

where $f(\boldsymbol{w}) = p(\boldsymbol{X}|\boldsymbol{w}, \boldsymbol{y})p(\boldsymbol{w})$ and $K = p(\boldsymbol{X}|\boldsymbol{y}) = \int f(\boldsymbol{w}) d\boldsymbol{w}$.

Approximate log-posterior
To find $q(\boldsymbol{w})$, we start with the truncated Taylor expansion of $\log f(\boldsymbol{w})$ around a local maximum $\boldsymbol{w}_0$:

$$\log f(\boldsymbol{w}) \approx \log f(\boldsymbol{w}_0) + \nabla \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0)$$

At a maximum of $f(\boldsymbol{w})$, $\nabla \log f(\boldsymbol{w}) = 0$ as the logarithm is a monotonic function. Hence, close to $\boldsymbol{w}_0$:

$$\begin{aligned}\log f(\boldsymbol{w}) &\approx \log f(\boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0) \\ f(\boldsymbol{w}) &\approx f(\boldsymbol{w}_0) \exp\left(\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0} (\boldsymbol{w} - \boldsymbol{w}_0)\right)\end{aligned}$$

This is of the form of an un-normalised Gaussian centred on the maximum at $\boldsymbol{w}_0$. Set $\boldsymbol{w}_0 = \boldsymbol{w}_{\text{MAP}}$, and let $\boldsymbol{S}_N^{-1} = -\nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}}$, then normalise to obtain the approximate posterior distribution:

$$\begin{aligned}p(\boldsymbol{w}|\tilde{\boldsymbol{X}}, \boldsymbol{y}) &\approx \frac{1}{(2\pi)^{\frac{N}{2}} \det \boldsymbol{S}_N^{-\frac{1}{2}}} \exp\left(\frac{1}{2}(\boldsymbol{w} - \ \boldsymbol{w}_{\text{MAP}})^T \boldsymbol{S}_N^{-1}(\boldsymbol{w} - \boldsymbol{w}_{\text{MAP}})\right) \\ &= \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_{\text{MAP}}, \boldsymbol{S}_N)\end{aligned} \tag{7}$$

where N is the number of data points $\boldsymbol{x}_n$. For Equation 7 to hold, $\boldsymbol{S}_N^{-1}$ must be positive definite, which is equivalent to saying $\boldsymbol{w}_{\text{MAP}}$ must be a maximum (which is true by definition).

Approximate log-evidence
Now we can obtain an approximate value of the normalising constant K, by substituting Equation 7 into the definition of K:

$$\begin{aligned}K = \int f(\boldsymbol{w}) d\boldsymbol{w} &\approx f(\boldsymbol{w}_0) \int \exp\left(\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{S}_N^{-1}(\boldsymbol{w} - \boldsymbol{w}_0)\right) d\boldsymbol{w} \\ &= \sqrt{\frac{(2\pi)^N}{\det \boldsymbol{S}_N}} f(\boldsymbol{w}_0)\end{aligned}$$

Taking logs and substituting in the definitions of $\boldsymbol{w}_0$, K, and $f(\boldsymbol{w})$ gives the approximate log-evidence:

$$\log p(\boldsymbol{X}|\boldsymbol{y}) \approx \frac{N}{2}\log 2\pi - \frac{1}{2}\log\det \boldsymbol{S}_N + \mathcal{L}(\boldsymbol{w}_{\text{MAP}}) + \mathcal{P}(\boldsymbol{w}_{\text{MAP}}) \tag{8}$$

Covariance matrix
Differentiating Equation 6, the covariance matrix of $q(\boldsymbol{w})$ can be found:

$$\begin{aligned}
\boldsymbol{S}_N^{-1} &= -\nabla^2 \log p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} \\
&= -\nabla^2\mathcal{P}(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} - \nabla^2\mathcal{L}(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} \\
&= \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N}\sigma(\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n)\sigma(-\boldsymbol{w}^T\tilde{\boldsymbol{x}}_n)\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{x}}_n^T
\end{aligned} \tag{9}$$

Predictive distribution
The predictive distribution can also be approximated using the Laplace approximation for the posterior:

$$\begin{aligned}
p(y^*=1|\boldsymbol{x}^*,\boldsymbol{y},\boldsymbol{X}) &= \int p(y^*=1|\boldsymbol{x}^*,\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X})d\boldsymbol{w} \\
&\approx \int \sigma(\boldsymbol{w}^T\boldsymbol{x}^*)q(\boldsymbol{w})d\boldsymbol{w}
\end{aligned}$$

Using the sifting property of the delta function:

$$\sigma(\boldsymbol{w}^T\boldsymbol{x}) = \int \delta(a-\boldsymbol{w}^T\boldsymbol{x})\sigma(a)da$$

Hence:

$$\begin{aligned}
p(y^*=1|\boldsymbol{x}^*,\boldsymbol{y},\boldsymbol{X}) &\approx \int\int \delta(a-\boldsymbol{w}^T\boldsymbol{x}^*)\sigma(a)q(\boldsymbol{w})d\boldsymbol{w}da \\
&= \int \sigma(a)\int \delta(a-\boldsymbol{x}^{*T}\boldsymbol{w})q(\boldsymbol{w})d\boldsymbol{w}da
\end{aligned}$$

The inner integral applies a linear constraint to $q(\boldsymbol{w})$, as the argument of the delta function is 0 unless $a = \boldsymbol{x}^{*T}\boldsymbol{w}$. Hence, the approximate predictive distribution is:

$$\begin{aligned}
p(y^*=1|\boldsymbol{x}^*,\boldsymbol{y},\boldsymbol{X}) &\approx \int \sigma(a)\mathcal{N}(a;\boldsymbol{x}^{*T}\boldsymbol{w}_{\text{MAP}},\boldsymbol{x}^{*T}\boldsymbol{S}_N\boldsymbol{x}^{*})da \\
&= \int \sigma(a)\mathcal{N}(a;\mu_p,\sigma_p^2)da
\end{aligned} \tag{10}$$

$$\begin{aligned}
\mu_p &= \boldsymbol{x}^{*T}\boldsymbol{w}_{\text{MAP}} \\
\sigma_p^2 &= \boldsymbol{x}^{*T}\boldsymbol{S}_N\boldsymbol{x}^{*}
\end{aligned}$$

This integral cannot be expressed analytically, so another approximation is required. The logistic function can be approximated well by a probit function scaled such that the gradient of the two functions at the origin are equal. It can be shown that this gives:

$$\sigma(x) \approx \Phi^{-1}\left(\sqrt{\frac{\pi}{8}}x\right) = \Phi^{-1}(\lambda x) \tag{11}$$

Substituting into Equation 10 and evaluating using properties of the probit function gives:

$$\begin{aligned}
p(y^*=1|\boldsymbol{x}^*,\boldsymbol{y},\boldsymbol{X}) &\approx \int \Phi^{-1}(\lambda x)\mathcal{N}(a;\mu_p,\sigma_p)da \\
&= \Phi^{-1}\left(\frac{\mu_p}{\sqrt{\lambda^{-2}+\sigma_p^2}}\right)
\end{aligned}$$

Using Equation 11, this can be converted back into a logistic function:

$$p(y^*=1|\boldsymbol{x}^*,\boldsymbol{y},\boldsymbol{X}) \approx \sigma\left(\frac{\mu_p}{\sqrt{1+\sigma_p^2\lambda^2}}\right) \tag{12}$$

Summary of Laplace Approximation

- Posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_{\text{MAP}}, \boldsymbol{S}_N) \tag{7 restated}$$

$$\boldsymbol{S}_N = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} \sigma(\boldsymbol{w}^T \tilde{\boldsymbol{x}}_n)\sigma(-\boldsymbol{w}^T \tilde{\boldsymbol{x}}_n)\tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^T \tag{9 restated}$$

- Model evidence:

$$\log p(\boldsymbol{X}|\boldsymbol{y}) \approx \frac{N}{2}\log 2\pi - \frac{1}{2}\log\det \boldsymbol{S}_N + \mathcal{L}(\boldsymbol{w}_{\text{MAP}}) + \mathcal{P}(\boldsymbol{w}_{\text{MAP}}) \tag{8 restated}$$

- Predictive distribution for new points $\boldsymbol{x}_*$:

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) \approx \sigma\left(\frac{\mu_p}{\sqrt{1 + \sigma_p^2 \lambda^2}}\right) \tag{12 restated}$$

$$p(y_* = 0|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) = 1 - p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})$$

$$\mu_p = \tilde{\boldsymbol{x}}_*^T \boldsymbol{w}_{\text{MAP}}$$
$$\sigma_p^2 = \tilde{\boldsymbol{x}}_*^T \boldsymbol{S}_N \tilde{\boldsymbol{x}}_*$$
$$\lambda^2 = \frac{\pi}{8}$$

3 Implementation in Python

Firstly, the data is loaded and split into training and test sets:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

X_data, y_data = shuffle(np.loadtxt('data/X.txt'),
                         np.loadtxt('data/y.txt'))

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, train_size=800)
```

Some utility functions are defined, include the function that expands the inputs through radial basis functions (RBFs) to allow for non-linear decision boundaries:

```
def log(x, epsilon=1e-5):
    """Log function that avoids divide by zero errors"""
    return np.log(x + epsilon)

def prepend_ones(M):
    return np.column_stack((np.ones(M.shape[0]), M))

def expand_inputs(width, data, centers=X_train):
    """Expand data through a set of RBFs with given centres and widths"""
    l = width
    Z = centers
    X = data
    X2 = np.sum(X**2, 1)
    Z2 = np.sum(Z**2, 1)
```

```
    ones_Z = np.ones(Z.shape[ 0 ])
    ones_X = np.ones(X.shape[ 0 ])
    r2 = np.outer(X2, ones_Z) - 2 * np.dot(X, Z.T) + np.outer(ones_X, Z2)
    return prepend_ones(np.exp(-0.5 / l**2 * r2))
```

Next, the central equations of the model are defined. In this implementation, we consider a uniform prior on the weights with mean 0 and covariance $\boldsymbol{I}\sigma_0^2$:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; 0, \boldsymbol{I}\sigma_0^2) \tag{13}$$
$$\mathcal{P}(\boldsymbol{w}) \propto -\frac{1}{2\sigma_0^2}\boldsymbol{w}^T\boldsymbol{w}$$

```
def logistic(x):
    return 1 / (1 + np.exp(-x))

def log_prior(w, variance):
    return -1 / (2 * variance) * (w.T @ w)

def log_likelihood(w, X, y):
    sigma = logistic(X @ w)
    return np.sum(y * log(sigma)
                  + (1 - y) * log(1 - sigma))

def log_likelihood_gradient(w, X, y):
    return (y - logistic(X @ w)) @ X

def log_posterior(w, X, y, prior_variance):
    return log_likelihood(w, X, y) + log_prior(w, prior_variance)

def posterior_gradient(w, X, y, prior_variance):
    return (y - logistic(X @ w)) @ X - w / prior_variance
```

We use gradient descent to find the MAP and maximum likelihood estimates of the weights:

```
from scipy.optimize import fmin_l_bfgs_b as minimise

def find_w_map(X, y, prior_variance, w0):
    w_map, *d = minimise(lambda w, X, y, prior_variance:
                            -log_posterior(w, X, y, prior_variance),
                         w0,
                         lambda w, X, y, prior_variance:
                            -posterior_gradient(w, X, y, prior_variance),
                         args=[X, y, prior_variance])
    return w_map

def find_w_ml(X, y, w0):
    w_ml, *d = minimise(lambda w, X, y: -log_likelihood(w, X, y),
                        w0,
                        lambda w, X, y: -log_likelihood_gradient(w, X, y),
                        args=[X, y])
    return w_ml
```

Note that `scipy.optimize.fmin_l_bfgs_b` is a minimisation function, so to maximise $f(x)$ we have to work with $-f(x)$ and $-\nabla f(x)$.

Once $\boldsymbol{w}_{\text{MAP}}$ is found, the Laplace approximation for the model evidence and the predictive distribution can be found. First we find the Hessian matrix S_N using Equation 9:

```
def calculate_hessian(weights, rbf_width, prior_variance):
    # The Hessian will be MxM, where M is the number of features (= D+1)
    M = weights.shape[0]
```

```
    X_tilde = expand_inputs(rbf_width, X_train)

    # Contribution from prior
    h = np.identity(M) / prior_variance

    # Contribution from data
    for x in X_tilde:
        sigma = logistic(weights @ x)
        h += sigma * (1 - sigma) * np.outer(x, x)

    return h
```

Then the predictive function (Equation 12) can be implemented:

```
def laplace_prediction(inputs, weights, rbf_width, prior_variance):
    X_tilde = expand_inputs(rbf_width, X_train)
    sigma = logistic(X_tilde @ weights)
    S_N = np.linalg.inv(calculate_hessian(weights, rbf_width, prior_variance))
    predictive_mean = inputs @ weights
    predictive_variance = np.array([x.T @ S_N @ x for x in inputs])
    return logistic(predictive_mean / np.sqrt(1 + predictive_variance*np.pi/8))
```

The approximate log-evidence is calculated using Equation 8. The log-determinant is calculated using `np.linalg.slogdet`, which helps avoid the numerical errors encountered when calculating the determinant directly:

```
def log_evidence(w, X, y, rbf_width, prior_variance):
    h = calculate_hessian(w, rbf_width, prior_variance)
    sign, logdet = np.linalg.slogdet(h)
    if np.exp(logdet) == 0:
        # If the determinant is 0, the approximation does not work
        return -np.inf
    return ((y.size/2)*log(2*np.pi)
            - 0.5*logdet
            + log_likelihood(w, X, y)
            + log_prior(w, prior_variance))
```

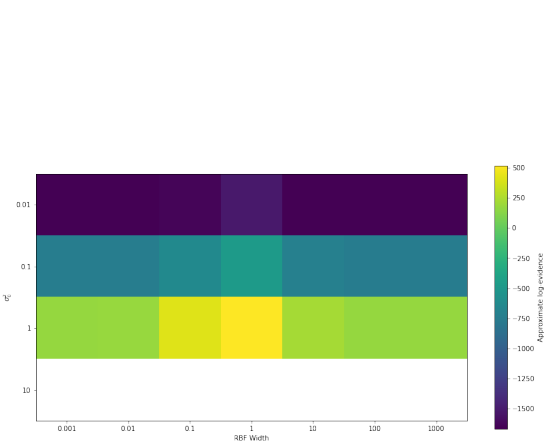
4 Performance of the Laplace Approximation

5 Optimising the hyperparameters

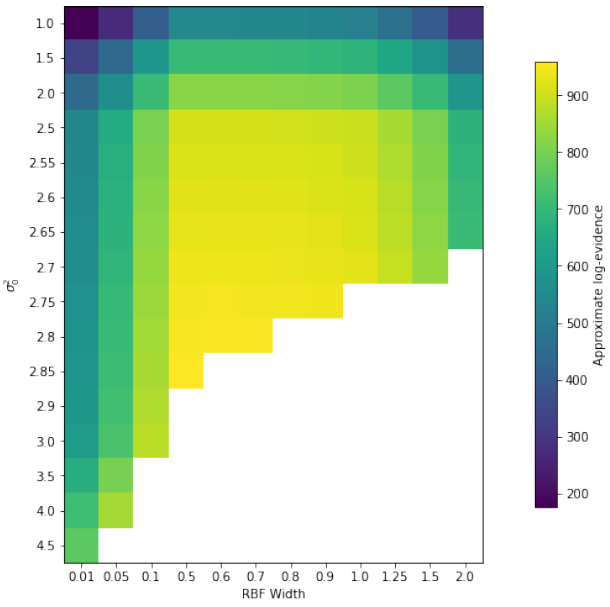
The classifier we have implemented has two hyperparameters: `rbf_width` and `prior_variance`. The optimal setting for these parameters can be found by maximising the model evidence. We perform a grid search to find the maximum model evidence:

```
def grid_search(rbf_widths, prior_variances):
    grid = np.empty((len(rbf_widths), len(prior_variances)))
    for i, rbf_width in enumerate(rbf_widths):
        for j, prior_variance in enumerate(prior_variances):
            expanded_X_train = expand_inputs(rbf_width, X_train)
            w_map = find_w_map(expanded_X_train, y_train, prior_variance)
            grid[i][j] = log_evidence(w_map, expanded_X_train, y_train,
                                      rbf_width, prior_variance)
```

Initially, a coarse grid is used to find the appropriate range to optimise over (Figure 1a). The most promising range is identified to be $0.1 < \sigma_0^2 < 10$, $0.01 <$ RBF width < 10. It is also observed that for large prior variances ($\sigma_0^2 \geq 10$) the evidence cannot be computed - shown as white on the heatmap - as the covariance matrix $\boldsymbol{S}_N$ is no longer full rank. A finer grid search is performed within the optimal



(a) Coarse heatmap



(b) Fine heatmap

Figure 1: Heatmaps of the model evidence for different settings of the hyperparameters

region (Figure 1b). The gradient of the evidence around the maximum is quite shallow, so there are a number of settings of the hyperparameters that could give good results.

We choose a pair of values that are close to the center of the peak: $\sigma_0^2 = 2.8$, RBF width $= 0.7$. The Laplace approximation of the predictive distribution for this setting is shown in Figure 2.

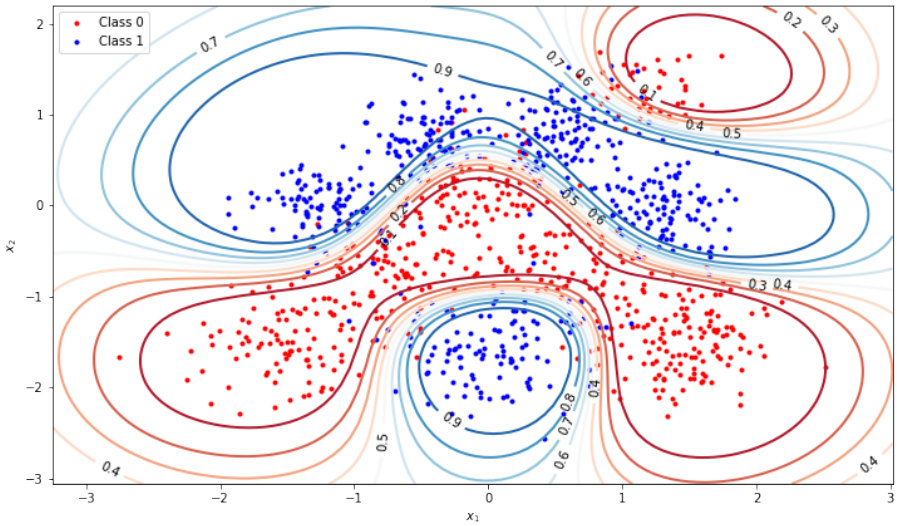


Figure 2: Laplace approximation of the predictive distribution for $\sigma_0^2 = 2.8$, RBF width $= 0.7$

6 Conclusion