# 3F8: Inference
# Full Technical Report

os416

March 28, 2023

**Abstract**

Maximising the likelihood, whilst simple, is prone to overfitting, hence we seek a solution to this. Using Bayesian inference we fit a prior on the weights to reduce overfitting.

Python will be used to implement the logistical classifier, optimize it and visualise the results.

We explore the performance of the MAP and the Laplace approximation for the posterior. We find performance is similar: the MAP solution is more prone to overfitting and unable to indicate uncertainty, whilst being far easier to compute

## 1 Introduction

Using MAP method of obtaining classification boundaries is unable to indicate uncertainty in the classifier due to the nature of choosing a single set of parameters, $\beta_{MAP}$. If we where to instead integrate over a distribution of $\beta$ obtained from a given prior on $\beta$ and our data, we will be able to both classify data and obtain the corresponding uncertainty in the bounds. This proves to be difficult due to the intractable (Non-linear) posterior on $\beta$, introduced by the function, which is used to classify the data. Both evaluating the posterior and calculating the predictive distribution are intractable.

We will use the Laplace Approximation in order to obtain a Gaussian approximation of the posterior. This will allow us to easily integrate over $\beta$ and normalise, circumventing the direct intractable nature of the Bayesian logistic regression.

## 2 Mathematical Theory

The aim is to find a Gaussian approximation centered on the mode of $p(\mathbf{z})$.

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

where $Z = \int f(\mathbf{z})\, d\mathbf{z}$

We find the mode $\boldsymbol{z_0}$ by evaluating $\frac{df(\mathbf{z})}{d\mathbf{z}}|_{\mathbf{z}=\mathbf{z_0}} = 0$.

Now we must find the Covariance matrix. To do this we take the 2nd order expansion of the log of $f(\mathbf{z})$. This yields equation 2.

$$ln(f(\mathbf{z})) \approx ln(f(\mathbf{z_0}) - \frac{1}{2}(\mathbf{z} - \mathbf{z_0})^T A(\mathbf{z} - \mathbf{z_0}) \tag{2}$$

Where A is the hessian of $-ln(f(\mathbf{z}))$. From equation 2, we take the exponential, resulting in equation 3.

$$f(\mathbf{z}) \approx f(\mathbf{z_0})exp(-\frac{1}{2}(\mathbf{z} - \mathbf{z_0})^T A(\mathbf{z} - \mathbf{z_0})) \tag{3}$$

Normalising equation 3 yields equation 4,the Gaussian approximation for $p(\mathbf{z})$.

$$p(\mathbf{z}) = \frac{f(\mathbf{z})}{Z} \approx q(\mathbf{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z_0},\, \boldsymbol{A}^{-1}) \tag{4}$$

Hence $Z = f(\mathbf{z_0})(2\pi)^{\frac{d}{2}}|\mathbf{A}|^{-\frac{1}{2}}$

Now we apply the same theory above to Bayesian logistic regression. Equation 5 is equivalent to equation 1.

$$p(\beta|\mathcal{D}) = \frac{p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\tilde{X}})p(\boldsymbol{\beta})}{Z} \tag{5}$$

where $p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\tilde{X}}) = \prod_{i=1}^{N} \sigma(y_n \boldsymbol{\beta^T} \boldsymbol{\tilde{x_n}})$ and the prior is given by $p(\beta_m) = \frac{1}{C} exp(-\frac{\beta_m^2}{2\sigma_0^2})$ for m = 0 ... N, C is the normalising constant.

In order to obtain the mode, we take the negative log of the posterior.

$$-\mathcal{L}(\boldsymbol{\beta}) = -\sum_{i=1}^{N} ln(\sigma(y_n \boldsymbol{\beta^T} \boldsymbol{\tilde{x_n}})) + \frac{\boldsymbol{\beta}^T \boldsymbol{\beta}}{2\sigma_0^2} \tag{6}$$

We then differentiate this and set it equal to 0 to find the mode $\boldsymbol{\beta_{MAP}}$.

$$-\frac{d\mathcal{L}(\boldsymbol{\beta})}{d\boldsymbol{\beta}} = -(\mathbf{y} - \sigma(\boldsymbol{\beta}^T \mathbf{X}))^T \mathbf{X} + \frac{\boldsymbol{\beta}}{\sigma_0^2} \tag{7}$$

Finally, $\boldsymbol{A}$ is given by the negative hessian of $\mathcal{L}(\boldsymbol{\beta})$ evaluated at $\boldsymbol{\beta_{MAP}}$. $\boldsymbol{A} = -\bigtriangledown \bigtriangledown \mathcal{L}(\boldsymbol{\beta})$

$$\boldsymbol{A} = \boldsymbol{X}^T (\sigma(\boldsymbol{\beta}^T \mathbf{X})^T (1 - \sigma(\boldsymbol{\beta}^T \mathbf{X}))\boldsymbol{I}_{nxn})\boldsymbol{X} + \frac{1}{\sigma_0^2}\boldsymbol{I}_{mxm} \tag{8}$$

Note $\boldsymbol{I}_{mxm}$ is a m x m Identity matrix.
We now seek the predictive distribution as shown in equation 9.

$$p(y_*|\boldsymbol{x_*}, \boldsymbol{\mathcal{D}}, \beta) = \int p(y_*|\boldsymbol{x_*}, \beta)p(\beta|\boldsymbol{\mathcal{D}})\, d\boldsymbol{\beta} \tag{9}$$

This integral is far easier to compute if we can approximate the likelihood, $p(\boldsymbol{y_*}|\boldsymbol{x_*}, \boldsymbol{\beta})$, from a logistic function to a probit function, as shown below.

$$p(y_*|\boldsymbol{x_*}, \boldsymbol{\mathcal{D}}, \beta) \approx \int \Phi(y_*|\boldsymbol{x_*}, \beta)\mathcal{N}(\beta|\boldsymbol{\beta_{MAP}},\, \boldsymbol{A^{-1}})\, d\boldsymbol{\beta} = \Phi\left(\frac{\boldsymbol{\beta_{MAP}^T x_*}}{\sqrt{1 + \boldsymbol{x_*^T A^{-1} x_*}}}\right) \tag{10}$$

Lastly we calculate model evidence. This is a key characteristic used to compare Bayesian models. The model evidence is given by equation 11. $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i)$ is equivalent to the likelihood of a particular model $\mathcal{M}_i$.

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}_i)p(\boldsymbol{\theta}|\mathcal{M}_i)\, d\boldsymbol{\theta} \tag{11}$$

Noting that this is equivalent to the normalising constant Z from equation 4, due to $Z = \int f(\mathbf{z})\, d\mathbf{z}$, we acquire equation 12.

$$ln(p(\mathcal{D}|\mathcal{M}_i)) \approx ln(p(\mathcal{D}|\boldsymbol{\theta_{MAP}}, \mathcal{M}_i) + \frac{M}{2} ln(2\pi) - \frac{1}{2} ln|\boldsymbol{A}| \tag{12}$$

Equation 12 can now be approximated to equation 13 when the hessian is full rank and the prior is broad.

$$ln(p(\mathcal{D}|\mathcal{M}_i)) \approx ln(p(\mathcal{D}|\boldsymbol{\theta_{MAP}}, \mathcal{M}_i) - \frac{1}{2} M ln(N) \tag{13}$$

## 3 Model Implementation

```
# define the posterior. Take the negative so we can find the minimum
def neg_log_postierior(X_tilde, y, w, sigma_0_sqrd):
  return -log_likelihood(X_tilde, y, w) + 1/(2 * sigma_0_sqrd) * np.matmul(w.T, w)

# define the gradient, which will be used to find the minimum
def grad_neg_log_postierior(X_tilde, y, w, sigma_0_sqrd):
  sigmoid_value = predict(X_tilde, w)
  dL_dw = np.matmul((y - sigmoid_value).T, X_tilde)
  return -dL_dw + 1/(sigma_0_sqrd) * w

def lapl_apprx(X_tilde, y, sigma_0_sqrd):

  M = X_tilde.shape[1]  # Note we have M-dimensions but M is the number of columns
  N = X_tilde.shape[0]

  # We start with random weights and then use scipy.optimize in order to find
  # the value of w when grad=0
  w0 = np.random.randn(M)
  w_map, f, d = scipy.optimize.fmin_l_bfgs_b(func=lambda
                                             w: neg_log_postierior(X_tilde, y, w,
                                                                   sigma_0_sqrd),
                                             x0=w0, fprime=lambda
                                             w: grad_neg_log_postierior(X_tilde, y, w,
                                                                        sigma_0_sqrd))

  # Calculate the Hessian using Bishop's equation 4.143.
  sigmoid_value = predict(X_tilde, w_map)
  A = ( X_tilde.T @ np.diag(sigmoid_value*(1-sigmoid_value))
       @ X_tilde + np.diag(1/sigma_0_sqrd * np.ones(M)) )

  # Calculation for model evidence and the approximation
  log_Z = ( log_likelihood(X_tilde, y, w_map) - 0.5*M*np.log(sigma_0_sqrd)
          - 0.5 * np.dot(w_map, w_map)/sigma_0_sqrd
         - 0.5 * np.linalg.slogdet(A)[1] )
  log_Z_approx = log_likelihood(X_tilde, y, w_map) - 0.5*M*np.log(N)
  return w_map, A, log_Z


# Obtains the predicted distribution using the probit approximation of the logistic function.
def predict_laplace(X_tilde, w_map, A):
  A_inv = np.linalg.inv(A)
  pred_dist = []
  for x_tilde in X_tilde:
    pred_dist.append(scipy.stats.norm.cdf(w_map.T @ x_tilde /
                                          np.sqrt(1 + x_tilde.T @ A_inv @ x_tilde)))

  return np.array(pred_dist)
```

## 4 Initial Model

Figure 1 shows the contours of both the MAP solution and the Laplace approximation with $l = 0.1$ and $\sigma_0^2 = 1$. We observe a relative large levels of similarity between the solutions, both effectively separate the classes. Though they don't enclose the outliers that effectively, with the MAP solution showing higher levels of overfitting.

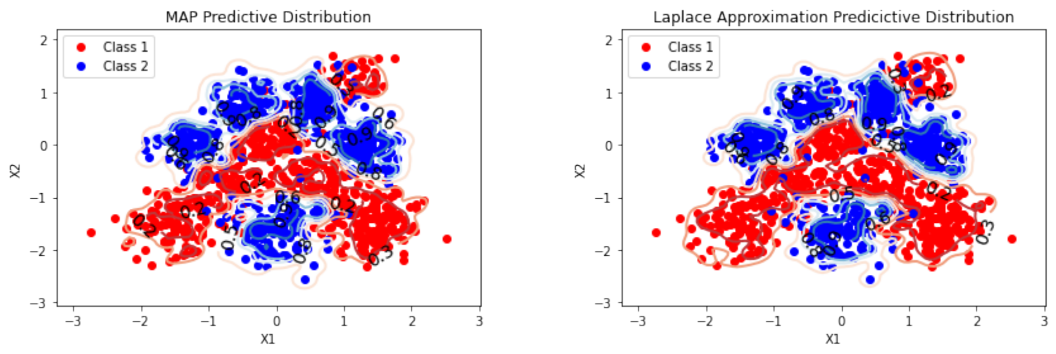

Figure 1: Plots showing data and contour lines for the predictive distribution generated by the MAP solution (left) and the Laplace approximation (right).

## 5 Initial Model Performance

Table 1 and 2 both show that the models are over-fitted due to the sizeable difference between the Avg. log likelihood in the training data, which is higher, compared to the test data.

| **Avg. Train ll** | **Avg. Test ll** |
|---|---|
| -0.21897 | -0.30293 |

Table 1: Log-likelihoods for MAP solution.

| **Avg. Train ll** | **Avg. Test ll** |
|---|---|
| -0.19960 | -0.27584 |

Table 2: Log-likelihoods for Laplace approximation.

Despite the edge the Laplace solution has on the MAP, we can see that this is insignificant as they have the exact same confusion matrix. This tells us that whilst the models may assign slightly different values, they are extremely close to each other and hence still assign to the same classes. This indicates just how similarly the models perform.

| | | $\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| $y$ | 0 | 0.94 | 0.06 |
| | 1 | 0.17 | 0.83 |

Table 3: Conf. matrix for for MAP solution.

| | | $\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| $y$ | 0 | 0.94 | 0.06 |
| | 1 | 0.17 | 0.83 |

Table 4: Conf. matrix for Laplace approximation.

## 6 Model Tuning

```
#I found the most suitable range was 0.1 to 2 for the variance and 0.1 to 1 for l.
sigma_0_sqrds = np.linspace(0.1,2,19)
ls = np.linspace(0.1, 1, 19)
modelEV = []
```

```
# We iterate through different values for l and the variance and calculate
# the corresponding model evidence.
for l in tqdm(ls):
  for sigma_0_sqrd in sigma_0_sqrds:
    X_tilde_train = get_x_tilde(evaluate_basis_functions(l, X_train, X_train))
    X_tilde_test = get_x_tilde(evaluate_basis_functions(l, X_test, X_train))

    w_map, A , log_Z = lapl_apprx(X_tilde_train, y_train, sigma_0_sqrd)
    modelEV.append(log_Z)
modelEV = np.array(modelEV).reshape(19,19)
```

The result from the code above is then plotted on a heat map seen in figure 2. I opted for a larger grid in order to obtain a greater level of accuracy. From the heat map we are able to obtain the optimal hyper-parameters for the classifier, $l = 0.55$ and $\sigma_0^2 = 0.73$.

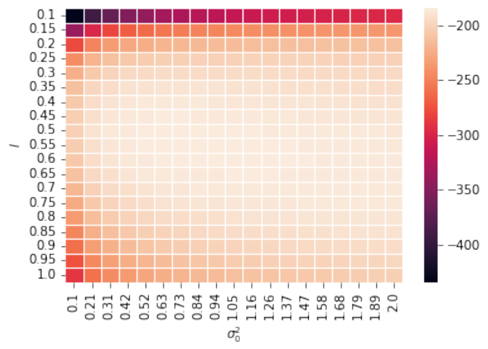

Figure 2: Heat map plot of the the approximation of the model evidence obtained in the grid search.

## 7 Tuned Model Performance

Figure 3 shows the resulting contours of the Laplace approximation with the optimal hyper-parameters chosen in the previous section. As we can see the contours are far better at enclosing the separate clusters and including the points further out, which indicates a decrease in overfitting. This is numerically shown in table 5, where the difference between the training and test Avg. log likelihood is far lower.

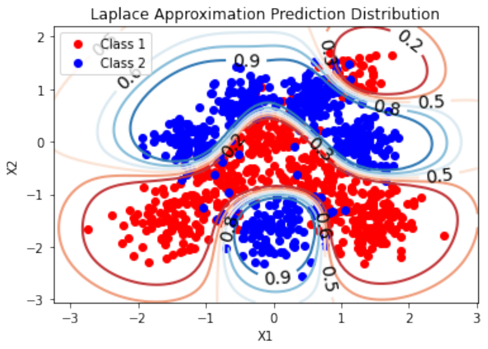

Figure 3: Visualisation of the contours of the class predictive probabilities for Laplace approximation after hyper-parameter tuning by maximising the model evidence.

| **Avg. Train ll** | **Avg. Test ll** |
|---|---|
| -0.20275 | -0.22286 |

Table 5: Average training and test log-likelihoods for Laplace approximation after hyper-parameter tuning by maximising the model evidence.

|  |  | $\hat{y}$ 0 | 1 |
|---|---|---|---|
| $y$ | 0 | 0.91 | 0.09 |
|  | 1 | 0.08 | 0.92 |

Table 6: Confusion matrix for Laplace approximation after hyper-parameter tuning by maximising the model evidence.

Lastly table 6 indicates no significant change for the models ability to identify class 0, but a large increase in performance when identifying points belonging to class 1.

## 8 Conclusions

At first with the original hyper-parameters it was clear that the MAP was a sufficient approximation as it performed similarly to the Laplace approximation, whilst being far easier to compute.

After tuning the hyper-parameters the performance of the Laplace approximation was significantly improved and far less prone to overfitting than before. Although when plotting the MAP solution with the tuned parameters, the confusion matrix was once again the exact same as the Laplace solution and the Avg. test and training log likelihoods were -0.20959 and -0.23145 respectively, once again indicating that the MAP was a sufficient approximation, with only a small amount of additional overfitting.

The reason for the MAP solution performing similarly to the Laplace solution is due to the small variance and hence a tight spread which can be approximated to a point impulse, the MAP solution. This also indicates that the approximation of the model evidence will perform poorly as that approximation assumes a broad prior.
This would be convenient for a given data set due to the Laplace solution requiring the hessian, which is difficult to compute accurately; they are rarely full-rank and often diverge when estimating the determinant.

Overall the MAP solution is far less demanding, although is slightly more susceptible to overfitting and it won't be able to indicate the uncertainty in data classification.