

# 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  $\{\mathbf{x}_n\}$  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:

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

with  $\tilde{\mathbf{x}}_n = [1, \mathbf{x}_n^T]^T$ ,  $\mathbf{w}$  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:

$$p(\mathbf{w} | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{X} | \mathbf{w}, \mathbf{y}) p(\mathbf{w})}{p(\mathbf{X} | \mathbf{y})} \quad (2)$$

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

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

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

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

The log-posterior is:

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

Calculating the model evidence  $p(\mathbf{X} | \mathbf{y})$  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.

### 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,  $\mathbf{w}_{\text{MAP}}$ , 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:

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

where  $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1 \dots \tilde{\mathbf{x}}_N]$ . A standard gradient-based solver can be used with Equation 6 to find a value for  $\mathbf{w}_{\text{MAP}}$ , which can be used as a setting for the weights in Equation 1 to classify data points. The performance of the MAP-based classifier will be compared to a fully Bayesian classifier later in the report (??).

### 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  $q(\mathbf{w})$  that closely models the posterior  $p(\mathbf{w}|\tilde{\mathbf{X}}, \mathbf{y})$  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:

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

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

#### Approximate log-posterior

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

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

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

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

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

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

where  $N$  is the number of data points  $\mathbf{x}_n$ . For Equation 7 to hold,  $\mathbf{S}_N^{-1}$  must be positive definite, which is equivalent to saying  $\mathbf{w}_{\text{MAP}}$  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  $K$ :

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

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

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

### Covariance matrix

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

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

### Predictive distribution

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

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

Using the sifting property of the delta function:

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

Hence:

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

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

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

Where

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

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:

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

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

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

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

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

### Summary of Laplace Approximation

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

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

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

- Model evidence:

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

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

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

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

$$\begin{aligned} \mu_p &= \tilde{\mathbf{x}}_*^T \mathbf{w}_{\text{MAP}} \\ \sigma_p^2 &= \tilde{\mathbf{x}}_*^T \mathbf{S}_N \tilde{\mathbf{x}}_* \\ \lambda^2 &= \frac{\pi}{8} \end{aligned}$$

## 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)
```

Functions are defined to expand the data through a set of radial basis functions centred on the training points, and prepend a column of ones:

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

def expand_rbf(1, X, Z=X_train):
    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))

```

A numerical approximation of the MAP estimate for the weights is found:

```

from scipy.optimize import fmin_l_bfgs_b as minimise

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 * np.log(sigma)
                  + (1 - y) * np.log(1 - sigma))

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

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

def find_w_map(X, y, w0=None, prior_variance=1):
    if w0 is None:
        w0 = np.random.normal(size=X.shape[1])
    w_map, posterior_at_wmap, d = minimise(negative_log_posterior,
                                           w0,
                                           negative_posterior_gradient,
                                           args=[X, y, prior_variance])
    return w_map

expanded_training_set = expand_rbf(rbf_width, X_train)
w_map = find_w_map(expanded_training_set, y_train)

```

Note that `scipy.optimize.fmin_l_bfgs_b` is a minimisation function, so to maximise the posterior we have to work with the negative posterior and negative posterior gradient. Once  $w_{\text{MAP}}$  is found, the Laplace approximation for the model evidence and the predictive distribution can be found:

```

def S_N_inv(w, rbf_width, prior_variance):
    M = w.shape[0]
    X_tilde = expand_rbf(rbf_width, X_train)
    sigma = logistic(X_tilde @ w)
    return np.identity(M) / prior_variance + X_tilde @ X_tilde.T @ sigma @ (1 - sigma)

def log_evidence(w, X, y, rbf_width, prior_variance):
    d = np.linalg.det(S_N_inv(w, rbf_width, prior_variance))
    return (M/2)*np.log(2*np.pi) - 0.5*np.log(d) + log_likelihood(w, X, y) + log_prior(w, prior_variance)

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

```

## 4 Performance of the Laplace Approximation