Bayesian Logistic Classification

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

March 22, 2022

Abstract

Abstract goes here

1 Introduction

1.1 Motivation

1.2 Logistic classification

The logistic classifier is a binary classifier that takes a D-dimensional input $\boldsymbol{x}_n$ and outputs a class label $y_n = \{0, 1\}$, 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).

1.3 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:

$$\mathcal{P}(\boldsymbol{w}) = \log p(\boldsymbol{w}) = -\frac{1}{2\sigma_0^2}(\boldsymbol{w} - \boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w} - \boldsymbol{m}_0) + \text{const} \tag{4}$$

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. Instead, we use the Laplace approximation to find an approximation for the evidence.

1.4 Laplace approximation

The Laplace approximation is a method of finding a Gaussian distribution $q(\boldsymbol{z})$ that closely models a desired distribution $p(\boldsymbol{z})$ around a local maximum of $p(\boldsymbol{z})$. This allows a distribution that is difficult to normalise (such as the posterior) to be replaced with a distribution with a well-defined integral and normalising constant. For brevity, rewrite Equation 2 as:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \frac{f(\boldsymbol{w})}{K} \approx q(\boldsymbol{w}) \tag{6}$$

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

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 $\boldsymbol{w}_0$:

$$\begin{aligned} \log f(\boldsymbol{w}) &\approx \log f(\boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0}(\boldsymbol{w} - \boldsymbol{w}_0) \\ f(\boldsymbol{w}) &\approx f(\boldsymbol{w}_0) \exp\left(\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0}(\boldsymbol{w} - \boldsymbol{w}_0)\right) \end{aligned} \tag{7}$$

Equation 7 is of the form of an un-normalised Gaussian. Letting $\boldsymbol{S} = -\nabla^2 \log f(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_0}$ and normalising Equation 7 gives the result for $q(\boldsymbol{w})$:

$$\begin{aligned} p(\boldsymbol{w}|\tilde{\boldsymbol{X}}, \boldsymbol{y}) &\approx \frac{1}{(2\pi)^{\frac{M}{2}} \det \boldsymbol{S}^{-\frac{1}{2}}} \exp\left(\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{S}(\boldsymbol{w} - \boldsymbol{w}_0)\right) \\ &= \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_0, \boldsymbol{S}^{-1}) \end{aligned} \tag{8}$$

For this to hold, $\boldsymbol{S}$ must be positive definite, which is equivalent to saying $\boldsymbol{w}_0$ must be a maximum.

This method can also be used to find an approximate value of the normalising constant K. Substituting Equation 8 into the definition of K:

$$\begin{aligned} K = \int f(\boldsymbol{w})d\boldsymbol{w} &\approx f(\boldsymbol{w}_0) \int \exp\left(\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)^T \boldsymbol{S}(\boldsymbol{w} - \boldsymbol{w}_0)\right) d\boldsymbol{w} \\ &= \sqrt{\frac{(2\pi)^N}{\det \boldsymbol{S}}} f(\boldsymbol{w}_0) \end{aligned} \tag{9}$$

Using the results for the likelihood of the logistic model in the previous report and the log of **??**,

The value of $\boldsymbol{w}$ at the maximum, $\boldsymbol{w}_{\text{MAP}}$, can be found by setting the derivative of Equation 5 to zero. The mean of $q(\boldsymbol{w})$ will be set to $\boldsymbol{w}_{\text{MAP}}$.

Using Equation 8, the covariance matrix of $q(\boldsymbol{w})$ can be found:

$$\begin{aligned} \boldsymbol{S}_N^{-1} &= -\nabla^2 \log p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} \\ &= \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} \sigma(\boldsymbol{w}^T \tilde{\boldsymbol{x}}_n)\sigma(-\boldsymbol{w}^T \tilde{\boldsymbol{x}}_n)\tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^T \end{aligned} \tag{10}$$

Defining $\boldsymbol{\sigma} = \sigma(\tilde{\boldsymbol{X}}^T \boldsymbol{w})$, this can be written in vector form as:

$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T \boldsymbol{\sigma}(1 - \boldsymbol{\sigma}) \tag{11}$$

Hence, using Equation 8, the posterior distribution is:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_{\text{MAP}}, \boldsymbol{S}_N) \tag{12}$$

And using Equation 9, the normalizing constant is:

$$p(\boldsymbol{X}|\boldsymbol{y}) = \frac{(2\pi)^{\frac{N}{2}}}{\det \boldsymbol{S}_N^{-\frac{1}{2}}} p(\boldsymbol{X}|\boldsymbol{w} = \boldsymbol{w}_{\text{MAP}}, \boldsymbol{y})p(\boldsymbol{w} = \boldsymbol{w}_{\text{MAP}}) \tag{13}$$

1.5 Predictive distribution

The predictive distribution can also be approximated using the Laplace method:

$$\begin{aligned} p(y^* = 1|\boldsymbol{x}^*, \boldsymbol{y}, \boldsymbol{X}) &= \int p(y^* = 1|\boldsymbol{x}^*, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X})d\boldsymbol{w} \\ &\approx \int \sigma(\boldsymbol{w}^T\boldsymbol{x}^*)q(\boldsymbol{w})d\boldsymbol{w} \end{aligned} \tag{14}$$

Using the sifting property of the delta function:

$$\sigma(\boldsymbol{w}^T\boldsymbol{x}) = \int \delta(a - \boldsymbol{w}^T\boldsymbol{x})\sigma(a)da \tag{15}$$

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:

$$\begin{aligned} p(y^* = 1|\boldsymbol{x}^*, \boldsymbol{y}, \boldsymbol{X}) &\approx \int \sigma(a)\mathcal{N}(a; \boldsymbol{x^*}^T\boldsymbol{w}_{\text{MAP}}, \boldsymbol{x^*}^T\boldsymbol{S}_N\boldsymbol{x^*})da \\ &= \int \sigma(a)\mathcal{N}(a; \mu_p, \sigma_p^2)da \end{aligned} \tag{16}$$

Where

$$\mu_p = \boldsymbol{x^*}^T\boldsymbol{w}_{\text{MAP}} \tag{17}$$

$$\sigma_p^2 = \boldsymbol{x^*}^T\boldsymbol{S}_N\boldsymbol{x^*} \tag{18}$$

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:

$$\sigma(x) \approx \Phi^{-1}\left(\sqrt{\frac{\pi}{8}}x\right) = \Phi^{-1}(\lambda x) \tag{19}$$

Substituting into Equation 16 and evaluating using properties of the probit function gives:

$$\begin{aligned} p(y^* = 1|\boldsymbol{x}^*, \boldsymbol{y}, \boldsymbol{X}) &\approx \int \Phi^{-1}(\lambda x)\mathcal{N}(a; \mu_p, \sigma_p)da \\ &= \Phi^{-1}\left(\frac{\mu_p}{\sqrt{\lambda^{-2} + \sigma_p^2}}\right) \end{aligned} \tag{20}$$

Using Equation 19, this can be converted back into a logistic function:

$$p(y^* = 1|\boldsymbol{x}^*, \boldsymbol{y}, \boldsymbol{X}) \approx \sigma\left(\frac{\mu_p}{\sqrt{1 + \sigma_p^2\lambda^2}}\right) \tag{21}$$

1.5.1 Finding the MAP estimate

The Laplace approximation requires the location of a maximum in the distribution to be approximated, which in this case will be the MAP estimate of $\boldsymbol{w}$. This can be found by applying gradient ascent to the posterior. Firstly, the gradient of the log-posterior is calculated:

$$\begin{aligned} \frac{\partial}{\partial\boldsymbol{w}}\log p(\boldsymbol{w}|\tilde{\boldsymbol{X}}, \boldsymbol{y}) &= \frac{\partial}{\partial\boldsymbol{w}}\log p(\tilde{\boldsymbol{X}}|\boldsymbol{w}, \boldsymbol{y}) + \frac{\partial}{\partial\boldsymbol{w}}\log p(\boldsymbol{w}) \\ &= \tilde{\boldsymbol{X}}(\mathbf{y} - \sigma(\tilde{\boldsymbol{X}}^T\boldsymbol{w})) + \frac{1}{\sigma_0^2}\boldsymbol{w} \end{aligned} \tag{22}$$

This can then be used with a standard gradient-based solver to find a value for $\boldsymbol{w}_{\text{MAP}}$.

1.5.2 Results of the Laplace approximation

Applying the Laplace approximation gives following results:

- Approximate posterior distribution of $\boldsymbol{w}$:

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}; \boldsymbol{w}_{\text{MAP}}, \boldsymbol{S}_N) \tag{23}$$

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

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) \approx \sigma\left(\frac{\mu_p}{\sqrt{1+\sigma_p^2\lambda^2}}\right) \tag{24}$$

- Approximate model evidence:

$$p(\boldsymbol{X}|\boldsymbol{y}) = \sqrt{\frac{(2\pi)^N}{\det \boldsymbol{S}_N^{-1}}} \exp\left(\mathcal{L}(\boldsymbol{w}_{\text{MAP}}) + \mathcal{S}(\boldsymbol{w}_{\text{MAP}})\right) \tag{25}$$

where

$$\boldsymbol{S}_N = \left(\frac{1}{\sigma_0^2}\boldsymbol{I} + \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T\boldsymbol{\sigma}(1-\boldsymbol{\sigma})\right)^{-1} \tag{26}$$

$$\boldsymbol{\sigma} = \sigma(\tilde{\boldsymbol{X}}^T\boldsymbol{w}) \tag{27}$$

$$\mu_p = \tilde{\boldsymbol{x}}_*^T\boldsymbol{w}_{\text{MAP}} \tag{28}$$

$$\sigma_p^2 = \tilde{\boldsymbol{x}}_*^T\boldsymbol{S}_N\tilde{\boldsymbol{x}}_* \tag{29}$$

$$\lambda^2 = \frac{\pi}{8} \tag{30}$$

$$\mathcal{L}(\boldsymbol{w}_{\text{MAP}}) = \log p(\boldsymbol{X}|\boldsymbol{w}, \boldsymbol{y})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} \tag{31}$$

$$\mathcal{P}(\boldsymbol{w}_{\text{MAP}}) = \log p(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_{\text{MAP}}} \tag{32}$$

and N is the number of data points $\boldsymbol{x}_n$.

1.6 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(l, 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 $\boldsymbol{w}_{\text{MAP}}$ 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_

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))
```

2 Performance of the Laplace Approximation