3F8: Inference
Full Technical Report

Kyeong Min (Cindy) Yu

March 24, 2020

Abstract

In this report, extensions were made following the short report in developing an optimal classification model, especially investigating the Bayesian binary classifier. Theory explained in Bishop's book in using Laplace approximation to approximate the posterior distribution was implemented, and from results from this lab, proved to be a robust method. Lastly, the effect of tuning parameters according to the model evidence function described in Bishop's Book was investigated.

1. Introduction

In the short lab, the logistic classification algorithm was implemented to a simple data set, its results and performance was evaluated, with and without the use of radial basis function feature expansion. In this extended assignment, a Bayesian binary classifier was implemented, applying the Laplace approximation to the logistic classification model from before. Overfitting is an is an issue when using gradient ascent method to maximize the log-likelihood of the weights, and the effect is much bigger when there are more inputs, as seen from results in the previous lab. There are a few solutions to minimize overfitting- one of which is to use the Bayesian binary classier. The results were similarly evaluated as before using the average log-likelihood and confusion matrices. Further, the performance of selecting hyper-parameter values by optimizing the model evidence was investigated.

2. Exercise a)

The logistic classification problem is considered, in which inputs are expanded through a set of Radial Basis Functions centered on training data points. $N_{train} + 1$ is the number of parameters, where N_{train} is the number of training data points. The assumption is needed that the prior on β is given as a Gaussian with zero mean and variance ${\sigma_0}^2$:

$$p(\boldsymbol{\beta_m}) = \frac{1}{Z}\exp\left(-\frac{\boldsymbol{\beta_m^2}}{2{\sigma_0}^2}\right)$$

The **likelihood** was shown in the short coursework exercise:

$$p(\boldsymbol{y}|\widetilde{\boldsymbol{X}}, \boldsymbol{\beta}) = \prod_{n=1}^{N} p(y^{(n)}|\,\widetilde{\boldsymbol{x}}^{(n)}) = \prod_{n=1}^{N} \sigma\left(\boldsymbol{\beta^T}\widetilde{\boldsymbol{x}}^{(n)}\right)^{y^{(n)}} \left(1 - \sigma\left(\boldsymbol{\beta^T}\widetilde{\boldsymbol{x}}^{(n)}\right)\right)^{1-y^{(n)}}$$

The **posterior distribution** on β is proportional to the product of the prior and likelihood, given by Bayes rule:

$$p(\boldsymbol{\beta}|\boldsymbol{y}, \widetilde{\boldsymbol{X}}) \propto p(\boldsymbol{\beta})p(\boldsymbol{y}|\widetilde{\boldsymbol{X}, \boldsymbol{\beta}},)$$

The **predictive distribution** is given by

$$p(y_* = 1|\tilde{\boldsymbol{x}}_*, \boldsymbol{y}, \tilde{\boldsymbol{X}}) = \int p(y_* = 1|\tilde{\boldsymbol{x}}_*, \boldsymbol{\beta}) p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}}) d\boldsymbol{\beta}$$

According to Bishop's book the exact Bayesian inference for logistic regression is not intractable. The evaluation of the posterior distribution would require normalization of the product of a prior distribution and a likelihood function that itself comprises of a product of logistic sigmoid functions, one for every data point. Therefore, Laplace approximation is used to approximate the predictive distribution. We approximate the posterior distribution as a Gaussian, using the Laplace approximation. Then, having obtained the posterior distribution approximation, marginalization is needed in order to make predictions.

To find the approximate Gaussian, defined as $q(\boldsymbol{z})$, we need to find $f(\boldsymbol{z})$, the distribution of a single continuous variable. By conducting Taylor expansion of $\ln f(\boldsymbol{z})$ centered on the stationary point $\boldsymbol{z}_0$, the Gaussian approximation can be found for an M-dimensional space $\boldsymbol{z}$.

$$\ln f(\boldsymbol{z}) \approx \ln f(\boldsymbol{z}_{MAP}) - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_{MAP})^T \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{z}_{MAP})$$

Where $\boldsymbol{A}$ is the Hessian matrix defined by:

$$\boldsymbol{A} = -\nabla\nabla \ln f(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{z}_0}$$

Then, we wish to implement this to the logistic classification model from the short coursework. Taking the log of both sides of the posterior distribution equation and taking the prior distribution equation into account,

$$\begin{aligned}\ln p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}}) &= \ln p(\boldsymbol{\beta}) + \ln p(\boldsymbol{y}|\boldsymbol{\beta}, \tilde{\boldsymbol{X}}) + const \\ &= \sum_{n=1}^{N} \{ y_n log\sigma(\boldsymbol{\beta}^T \tilde{x}_n) + (1 - y_n) \log(1 - \sigma(\boldsymbol{\beta}^T \tilde{x}_n)) \} - \frac{\boldsymbol{\beta}^T \boldsymbol{\beta}}{2{\sigma_0}^2} + const\end{aligned}$$

The above MAP solution is obtained by maximizing the posterior distribution. The solution of $\boldsymbol{\beta}_{MAP}$ is the mean of the Gaussian approximation. Taking the derivative,

$$\frac{\partial}{\partial \boldsymbol{\beta}} \ln p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}}) = \frac{\partial}{\partial \boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) - \frac{\boldsymbol{\beta}}{{\sigma_0}^2}$$

Where $\mathcal{L}(\boldsymbol{\beta})$ above is the log-likelihood function from the short coursework.

Let us call the terms in the log in the equation above $\mathcal{L}^*(\boldsymbol{\beta})$, such that , $\mathcal{L}^*(\boldsymbol{\beta}) = \ln p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}})$, as this is our interest for minimization.
We know that

$$\frac{\partial}{\partial \boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) = \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \left(\boldsymbol{y} - \sigma(\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \boldsymbol{\beta}) \right)$$

So,

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\beta}} \mathcal{L}^*(\boldsymbol{\beta}) &= \frac{\partial}{\partial \boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) - \frac{\boldsymbol{\beta}}{{\sigma_0}^2} \\ &= \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \left(\boldsymbol{y} - \sigma(\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \boldsymbol{\beta}) \right) - \frac{\boldsymbol{\beta}}{{\sigma_0}^2}\end{aligned}$$

Taking derivatives again,

$$\frac{\partial^2}{\partial \boldsymbol{\beta}^{\boldsymbol{2}}} \mathcal{L}^*(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} \left\{ \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \left(\boldsymbol{y} - \sigma(\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \boldsymbol{\beta}) \right) - \frac{\boldsymbol{\beta}}{{\sigma_0}^2} \right\} = \tilde{\boldsymbol{X}}^{\boldsymbol{T}} \frac{\partial}{\partial \boldsymbol{\beta}} \sigma(\tilde{\boldsymbol{X}}^{\boldsymbol{T}} \boldsymbol{\beta}) - \frac{1}{{\sigma_0}^2} \boldsymbol{I}$$

The covariance matrix is given by the inverse of the matrix of second derivates of the negative log likelihood. The covariance matrix, $\boldsymbol{C}$ is obtained as follows:

$$\boldsymbol{C} = -\nabla\nabla \ln p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}}) = \frac{1}{{\sigma_0}^2} \boldsymbol{I} + \sum_{n=1}^{N} \sigma(\boldsymbol{\beta}^T \tilde{x}_n)(1 - \sigma(\boldsymbol{\beta}^T \tilde{x}_n)) \tilde{x}_n \tilde{x}_n^T$$

With the mean vector and the Covariance matrix, the **Gaussian approximation to the posterior distribution** is:

$$q(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\boldsymbol{\beta}_{MAP}, \boldsymbol{C})$$

To approximate the predictive distribution, we use section 4.5.2 of Bishop's book. The predictive distribution is obtained by marginalizing with respect to the posterior distribution, in which itself is approximated by the Gaussian distribution as shown above. Using results in the section in Bishop's book,

$$p(y_* = 1|\tilde{\boldsymbol{x}}_*, \boldsymbol{y}, \tilde{\boldsymbol{X}}) = \int p(y_*|\tilde{x}_*, \boldsymbol{\beta}) p(\boldsymbol{\beta}|\boldsymbol{y}, \tilde{\boldsymbol{X}}) d\boldsymbol{\beta} \approx \int \sigma(\boldsymbol{\beta}^T \tilde{x}_*) q(\boldsymbol{\beta}) d\boldsymbol{\beta} = \int \sigma(a) p(a) da$$
$$= \int \sigma(a) \mathcal{N}(a|\mu_a, \sigma_a^2) da$$

denoting $a = \boldsymbol{\beta}^T \tilde{\boldsymbol{x}}_*$, $\mu_a = \boldsymbol{\beta}_{MAP}^T \tilde{\boldsymbol{x}}_*$, and $\sigma_a = \tilde{\boldsymbol{x}}_*^T \boldsymbol{C} \tilde{\boldsymbol{x}}_*$.
Then, $\sigma(a)$ is estimated by the probit function, $\Phi(\lambda a)$, where $\lambda^2 = \pi/8$.

The **predictive distribution approximation** is therefore,

$$p(y_* = 1|\tilde{\boldsymbol{x}}_*, \boldsymbol{y}, \tilde{\boldsymbol{X}}) \approx \sigma(\kappa(\sigma_a^2)\mu_a)$$

where $\kappa(\sigma_a^2) = \left(1 + \frac{\pi\sigma_a^2}{8}\right)^{-\frac{1}{2}}$.

Using section 4.4.1 in Bishop's book, we can obtain the approximation to the model evidence, which plays a central role in Bayesian model compassion. Before however, we have to approximate the normalization constant. Z:

$$Z = \int f(\boldsymbol{z}) d\boldsymbol{z} \approx f(\boldsymbol{z}_0) \int \exp\{-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_0)^T \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{z}_0)\} d\boldsymbol{z} = f(\boldsymbol{z}_0) \frac{(2\pi)^{M/2}}{|\boldsymbol{A}|^{1/2}}$$

Where $\boldsymbol{A}$ is the negative Hessian matrix described before.

From Bayes Theorem, the **model evidence** is given by

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

Where $\mathcal{D}$ is a dataset, θ_i are the parameters.
Identifying $f(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})$ and $Z = p(\mathcal{D})$, and using the approximation of Z above,

$$\ln p(\mathcal{D}) \approx \ln p(D|\boldsymbol{\theta}_{MAP}) + \ln p(\boldsymbol{\theta}_{MAP}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |\boldsymbol{A}|$$

Where M is the number of parameters in $\boldsymbol{\theta}$, and N is the number of data points.
Hence, because $\mathcal{L}^*(\boldsymbol{\beta}) = \mathcal{L}(\boldsymbol{\beta}) - \frac{\boldsymbol{\beta}^T \boldsymbol{\beta}}{2{\sigma_0}^2}$

$$\ln p(\mathcal{D}) = \mathcal{L}(\boldsymbol{\beta}) - \frac{\boldsymbol{\beta}^T \boldsymbol{\beta}}{2{\sigma_0}^2} - \frac{1}{2} \ln |\boldsymbol{A}|$$

3. Exercise b)

In this exercise, python codes for computing the Laplace approximation, predictive distribution and model evidence were written.

```
s0 = 1
#define func first
def func(beta, *args):
    X =args[0]
    y =args[1]
    r = -(y.size*compute_average_ll(X, y, beta) - (np.dot(beta.T,beta)/(2*s0*s0)))
    return r

X_tilde = get_x_tilde(X_train)
beta = np.random.randn(X_tilde.shape[1])
beta_map, value, d = optm.fmin_l_bfgs_b(func, beta, args=(X_tilde, y_train))

def Lap_approx(X, y, s0=1):
    X = np.concatenate((X, np.ones((X.shape[0],1))),1)
    beta = np.random.randn(X.shape[1])
    beta_map, value, d = optm.fmin_l_bfgs_b(func, beta, args=(X, y))

    s1=s0*s0*np.identity(beta_map.size)
    exp1 = logistic(np.dot(X, beta))
    exp2 = np.multiply(exp1, 1-exp1)

    A = np.zeros(s1.shape)
    for i in range(y.size):
        A += exp2[i]*np.dot(np.array(X[i])[:,None], np.array(X[i])[None,:])
    cov_inv = 1/(s0*s0*np.identity(beta_map.size)) + A
    cov=np.linalg.inv(cov_inv)

    return beta_map, cov
```

Figure 1: Python code for Laplace approximation

First we define the function we wish to minimize in the optimization function, which is $\mathcal{L}^*(\boldsymbol{\beta}) = \mathcal{L}(\boldsymbol{\beta}) - \frac{\boldsymbol{\beta}^T\boldsymbol{\beta}}{2{\sigma_0}^2}$. Using the *optm.fmin_l_bfgs_b* function that performs gradient based minimization, the MAP solution is found. Further, approximation of the covariance matrix is carried out. The summation term in the Hessian expression is computed using a for loop, and once the Hessian is found, the Covariance matrix is found simply by taking the inverse of the Hessian matrix. The function returns the covariance matrix and the MAP solution.

```
def bayesian_prediction(x):
    x_expand = evaluate_basis_functions(l, x, X_train)
    x_tilde = np.concatenate((x_expand, np.ones((x_expand.shape[0],1))),1)
    mu_a = np.dot(x_tilde, beta_map)
    var_a = np.diagonal(np.dot(x_tilde, np.dot(cov, x_tilde.T)))
    return logistic(np.dot(np.reciprocal(np.sqrt(1+np.pi*var_a/8)), mu_a))
```

Figure 2: Python code for Bayesian prediction

The mean and covariance expressions established are computed, and the prediction is carried out using the logistic function.

```
def model_evidence(s, l, X=X_train, y= y_train):
    K = evaluate_basis_functions(l, X_train, X_train)
    beta, cov = Lap_approx(K, y_train, s)
    K_1 = np.concatenate((K, np.ones((K.shape[0],1))),1)
    H = -np.linalg.inv(cov)
    likelihood = compute_average_ll_Laplace(K_1, y_train, beta)*len(y_train)
    sign, logdet = np.linalg.slogdet(H)
    r = likelihood - np.dot(w.T,w)/(s*s*2) - logdet/2
    return r
```

Figure 3: Python code for approximation of model evidence

The model evidence is computed as follows. After having expanded inputs through radial basis functions, and using the Laplace approximation function to determine the Hessian, we compute

the terms: the log likelihood and the log determinant Hessian, needed for the expression for model evidence found in part a).

4. Exercise c)

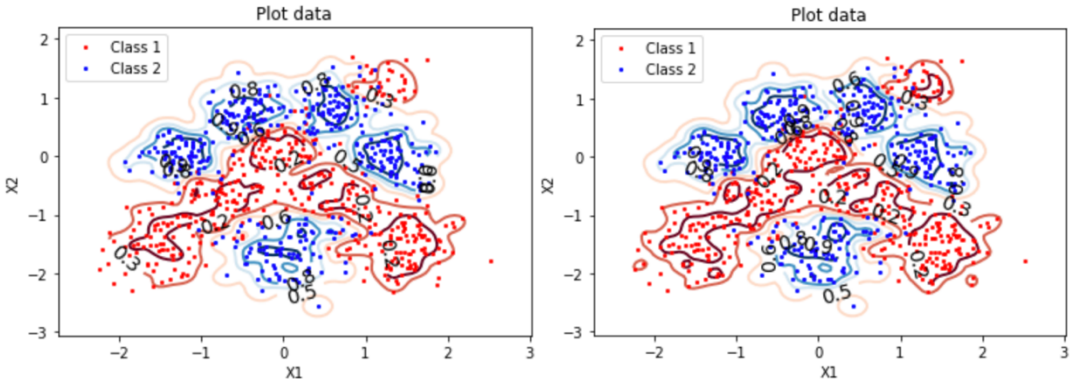

Figure 4: Plots showing data and contour lines for the predictive distribution generated by the Laplace approximation (left) and MAP solution (right)

Parameters are fixed to $\sigma_0^2 = 1$ and $l = 0.1$. Using the functions defined in part b), the predictions were visualized using probability contours as we have previously done in the short lab. We compare this to the visualized prediction of the MAP solution. Here, $\boldsymbol{\beta_{MAP}}$ is used as the parameter for the predictive distribution instead.
Comparing the two plots, the p=0.5 contour line are almost identical. However, the contour lines for larger/smaller probabilities (ex. p=0.2 and p=0.9) are smaller for the plot generated by Laplace approximation. The MAP solution gives a slightly better result, but showed increased overfitting.

5. Exercise d)

Include results in tables 1,2,3 and 4 and explain the results obtained and any findings

Avg. Train ll	Avg Test ll
-0.6823	-0.7146

Table 1: Log-likelihoods for MAP solution

Avg. Train ll	Avg Test ll
-0.7830	-0.7977

Table 2: Log-likelihoods for Laplace approximation

		$\hat{y}$	
		0	1
y	0	0.6733	0.3267
	1	0.2323	0.7677

Table 3: Conf. matrix for MAP solution.

		$\hat{y}$	
		0	1
y	0	0.6733	0.3267
	1	0.2323	0.7677

Table 4: Conf. matrix for Laplace approximation

The confusion matrices for the two solutions are identical, because the two have identical p=0.5 contours. This result is consistent with the results from the comparison from contour plots which gave identical p=0.5 contours.

From Tables 1 and 2, it can be concluded that the MAP solution gives a better result than Bayes, as it gave a higher log-likelihood value. This is because the approximating the predictive distribution using Bayes increases the uncertainty as it integrates over all solutions, unlike the MAP solution.

6. Exercise e)

A Grid Search approach for the optimization process was used: a 10 by 10 size grid was created representing all possible combinations of 10 different values for σ_0 and l. Then, the approximation for the model evidence for each point in this grid was computed. The parameters, σ_0 and l were varied logarithmically so that it produced a more uniform variation. Inclinations of $\sqrt{10}$ were used here, therefore, the ten points chosen were as following, both ranging from 0.01 to $10\sqrt{10}$.

```
def grid_search(s_grid, l_grid):
    result = []
    for row, s in enumerate(s_grid):
        for col, l in enumerate(l_grid):
            result.append([s,l, model_evidence(s,l)])
    return result

l_grid = [0.01,0.0316,0.1,0.316,1,3.16,10,31.6,100,316]
s_grid = [0.01,0.0316,0.1,0.316,1,3.16,10,31.6,100,316]

grid = grid_search(s_grid, l_grid)
```

Figure 5. Python code for Grid Search

Parameters chosen were:

$$\sigma_0 = [0.01, 0.0316, 0.1, 0.316, 1, 3.16, 10, 31.6, 100, 316]$$
$$l = [0.01, 0.0316, 0.1, 0.316, 1, 3.16, 10, 31.6, 100, 316]$$

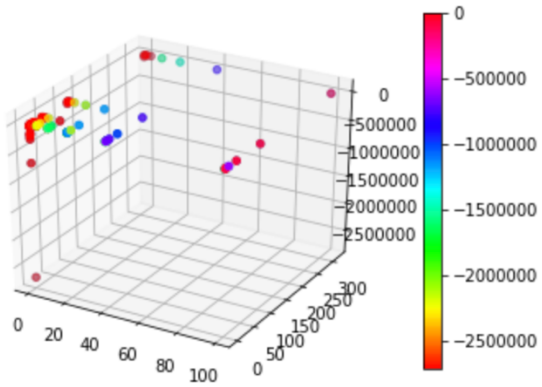

Figure 6. Heat map plot of the approximation of the model evidence obtained in the grid search

The best σ_0 and l were 100 and 1 respectively, found by correspondence of the maximum model evidence calculated, was 3227.

7. Exercise f)

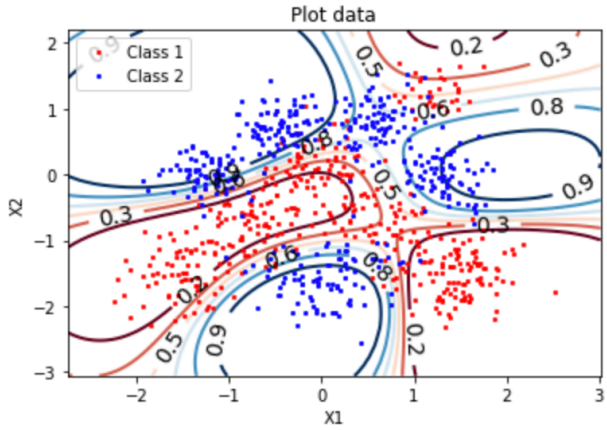

Figure 7: Visualization of the contours of the class predictive probabilities for Laplace approximation after hyper-parameter tuning by maximizing the model evidence.

Avg. Train ll	Avg Test ll
-0.7766	-0.7903

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

		$\hat{y}$	
		0	1
y	0	0.6733	0.3267
	1	0.2323	0.7677

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

The results from Figure 7, Tables 5 and 6 show that maximizing the model evidence does not necessarily mean that the performance is improved. Comparing the confusion matrix and average training and test log-likelihood values with that from tables 2 and 4, we can see no significant difference in selecting parameters according to the model evidence. It can also be noted that the confusion matrix stays the same. Comparing the contour plots in figures 4 and 7, it can be observed that using a large σ_0 as used here, the performance is actually quite poor. This suggests that the parameters found using model evidence is not optimal.

8. Conclusions

The Bayesian classifier model proved to be a fairly robust method, reducing overfitting compared to the ML approach explored in the short lab. However, in practice Bayesian classification is more computationally complex and expensive, and it further has a higher uncertainty, as the predictive model used in the Bayesian approach integrates over all solutions. Lastly, tuning the parameters using the model evidence, parameter values of $\sigma_0 = 100$ and l=1 were predicted to be the best set. However, visualizing the result using these parameters, it was found that simply tuning parameter in accordance to maximum model evidence proved to be ineffective, as the chosen σ_0 was thought to have been too large.