ROB313 Assignment 5 – Bayesian Inference and Publications

Objective

The objective of this assignment is to explore various Bayesian methods by first looking at the connection between Bayesian methods and the MAP solution, and how we can use iterative gradient descent to estimate the MAP. We experiment with the Laplace approximation as a means for calculating the marginal likelihood, and in subsequent parts of the assignment, we select a plausible proposal distribution as a way to simplify integrals over *dw*. When sampling from proposal distributions, we use Importance Sampling and Metropolis-Hastings MCMC sampling, and compare the time taken to converge and the accuracy of the results.

The final goal of the project is to read and understand a recent machine learning research paper and be capable of creating a comprehensive summary of the abstract and technical concepts.

Code Structure and Strategy

The code for this assignment was all written in one file, called *a5.py.* The code is split into 4 sections:

1. Common functions reused from other assignments
    a. *load_data*
    b. *make_x_matrix*: append a column of 1s to the front of any x data
    c. *sigmoid*
2. Part a, the computation of the log marginal-likelihoods using the Laplace Approximation
    a. *marginal_likelihoods*: the main function that finds the MAP solution for various prior variances and uses the MAP solution when computing the log marginal-likelihood with the Lapace approximation formula
    b. *log_likelihood, likelihood_grad, likelihood_hess:* compute $\log P(y|X,w)$, $\nabla \log P(y|X,w)$, $\nabla^2 \log P(y|X,w)$, respectively
    c. *log_prior, prior_grad, prior_hess:* compute $\log P(w)$, $\nabla \log P(w)$, $\nabla^2 \log P(w)$, respectively
3. Part b, define a proposal distribution and use importance sampling to select w values from this distribution
    a. *importance_sampling:* the main importance sampling algorithm, takes in a list of sample sizes and the MAP solution computed in part a, and calculates the best proposal variance/sample size combination on the validation set, and uses it on the testing set thereafter
    b. *proposal:* draws weights from a multi-variate Gaussian distribution using scipy.stats
    c. *prior/likelihood:* compute P(w) and P(y| X, w), respectively
    d. *visualization:* creates the proposal and posterior visualizations to help see the overlap in the distributions
4. Part c, using a Gaussian proposal distribution, use Metropolis-Hastings MCMC sampling to select w values, and visualize some of the flower results
    a. *metropolis_hastings:* the main Metropolis Hastings algorithm, functions very similarly to the importance_sampling function, but samples weights differently
    b. *hastings_sample:* the helper function that is used to draw each 100 weight samples for the Metropolis-Hastings algorithm

c. *Visualize_met_hast:* this function creates images of the histograms of the predictive posteriors of flowers of our choosing (defined by the input dictionary)

Q1. Bayesian Inference

a) Laplace Approximation

For the first portion of this assignment, we tested multiple variances for the Gaussian prior distribution P(w). The variances selected for the trial were $\sigma^2 = 0.5, 1, 2$. We used a gradient descent algorithm with a learning rate of $\eta = 0.0001$ and set a gradient threshold of $\epsilon = 10^{-6}$, at which point we were able to deduce that we were at the maximum a posteriori (MAP) solution. The log marginal-likelihood was then computed at this point, using $w = w^* = w_{MAP}$. Since the term in the log marginal-likelihood with $(w - w^*)$ will then go to 0, the resulting formula becomes:

$$\log\big(P(y|X)\big) = \log P(y|X, w) + \log P(w) + \frac{M}{2}\log(2\pi) - \frac{1}{2}\log|-H|$$

Where H is the hessian matrix of the MAP solution $H = \nabla^2 \log P(y|X, w) + \nabla^2 \log P(w)$.

Based on the variance values (σ^2) alone, we should expect a model with higher variance to have a higher model complexity. This is because when we have a larger prior variance for the weights, we have a larger set of weights we can choose from with a relatively high probability. Having a larger selection of w values means the model will take longer to converge to a MAP solution. In Table 1, we can see that the prior distribution is slightly too narrow when the variance is 0.5, but the log marginal-likelihood increases at a larger value of σ^2 because the model is slightly to complex.

Table 1, Log marginal-likelihood results at multiple variances

Variance	Number of iterations to converge	Log marginal likelihood
0.5	5469	74.824
1	9223	74.506
2	14169	74.814

Note that the values of log marginal-likelihood in Table 1 are not normalized and are for the entire training dataset.

b) Importance Sampling

To use importance sampling, we first had to assume the proposal distribution (q(w)) that $w^i \sim q(w)$ values would be drawn from. A multi-variate Gaussian distribution was selected, since we cannot assume the form of the posterior distribution as the prior P(w) is Gaussian, and the likelihood is categorical. We want the tails of the distribution are relatively heavy, so that we ensure the there is overlap with the posterior curve. We used the MAP solution found in part a) as the mean of the multi-variate Gaussian. We tested multiple combinations of the variances and sample sizes on the validation set, and then used the best result on the training set. Figures 1 and 2 below show visualizations for the proposal and posterior distribution for the optimal variance of $\sigma_p^2 = 1$. Note that the MAP solution calculated in part a) was assumed to be the mean of the proposal q(w) during validation (when the likelihood is calculated over just x_train) and during testing (when the likelihood is calculated over $x\_train \; + \; x\_valid$).

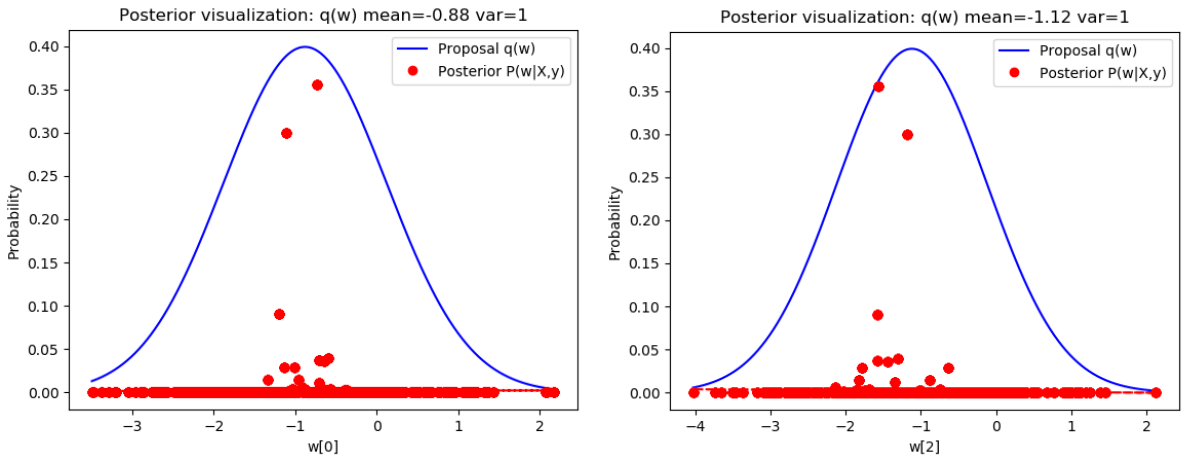


Figure 1 & 2, Examples of the proposal distribution overlap with the posterior, for elements 1 and 3 of the weight vector, respectively

Note that for figures 1 and 2, the probability of the posterior is dependent on the all of the components of the weight vector, whereas we are only plotting singular components of the w vector in each image. In the figures above, it was assumed that the posterior is equal to: $[\frac{r(w^i)}{\sum_{j=1}^{S} r(w^i)}]$, because of the following relationship:

$$P(y^*|x^*, X, y) = \int P(y^*|w^i, x^*)P(w|X, y)dw = \sum_{i=1}^{S} P(y^*|w^i, x^*)[\frac{r(w^i)}{\sum_{j=1}^{S} r(w^i)}]$$

Since we are performing binary classification of the flowers, we can predict the type of flowers by calculating the probability that the prediction is 1/true as follows:

$$P(y^* = 1|x^*, X, y) = \sum_{i=1}^{S} P(y^* = 1|w^i, x^*)[\frac{r(w^i)}{\sum_{j=1}^{S} r(w^i)}]$$

Where $r(w^i) = \frac{P(y|w^i, X)P(w^i)}{q(w^i)}$ and $P(y^* = 1|w^i, x^*)$ is the categorical likelihood ($\hat{f}(x, w^i) = sigmoid(X^T w)$ when $y^* = 1$). The probability of each y value being 1 was used to calculate the negative log-likelihood on the validation set.

Table 2, Validation results with importance sampling

Proposal Variance, σ_p^2	**Sample Size**	**Validation Negative LL**	**Validation Accuracy**
1	10	16.22	70.97 %
1	100	15.57	70.97 %
1	500	15.04	70.97 %
2	10	17.02	80.65 %
2	100	16.14	70.97 %
2	500	15.14	67.74 %
5	10	17.14	74.19 %
5	100	15.93	61.29 %
5	500	15.77	70.97 %

When selecting the proposal variance and sample size to be used on the test set, we chose the negative log-likelihood (NLL) as the primary metric. It is clear from the table above that the accuracy is a less precise metric since many of the combinations of variance and sample size have the same accuracy values. As discovered in previous assignments, the negative log-likelihood represents "how correct" the results are, whereas the accuracy just determines if the results are correct or not. Thus, if we used accuracy as the most important metric, we would get the optimal values to be $\sigma_p^2 = 2$ and sample size $= 10$, but these parameters also have one of the highest NLL values, which leads us to believe this high accuracy is a random chance based on weight sampling.

When the proposal variance $\sigma_p^2 = 1$ and sample size $= 500$ were run on the test set, we saw a test negative log-likelihood of **7.27**, and a test accuracy of **73.33 %.**

c) Metropolis-Hastings MCMC sampler

For this model, we created a Metropolis-Hastings MCMC sampling algorithm which had a burn-in of 1000 iterations and then sampled every 100th sample after the burn-in. The new sampling algorithm takes slightly longer than the importance sampling to run, however the overall results were slightly preferable. Multiple proposal distribution (Gaussian) variances σ_p^2 were tested, with a constant sample size of 100, and it was found that the best variance for the proposal distribution was 1, as was found with the importance sampling method.

When selecting the variance, we also want to consider the acceptance rate of new w values – we want an acceptance rate of approximately 50% throughout the Metropolis-Hastings sampling. If the variance on the proposal distribution is too low, we do not explore very many other weight vectors (i.e. reject too few samples). If the variance is too high, we end up rejecting too many samples because the probability of any new weight vector is low, and the distribution becomes a sparse subset of the target posterior.

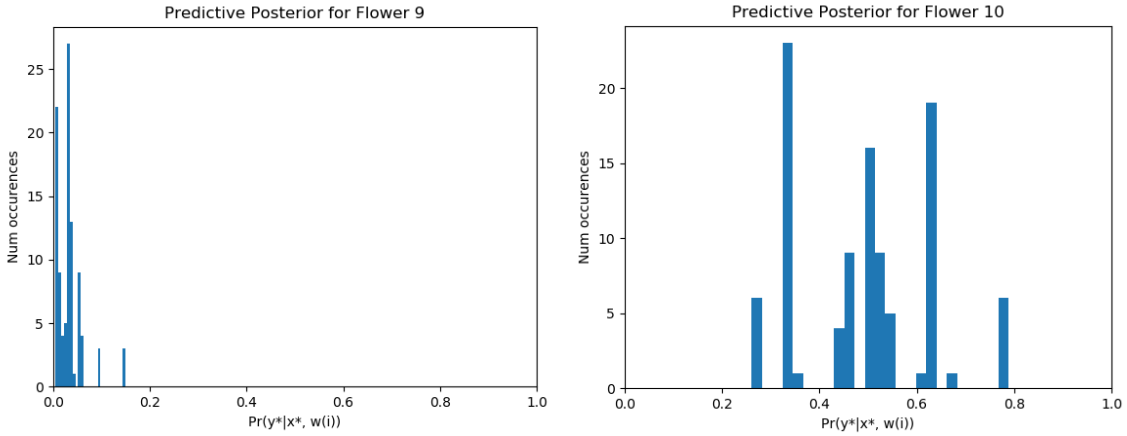
Table 3, Validation results with Metropolis-Hastings MCMC sampling

Proposal variance	Validation negative log-likelihood	Validation accuracy
1	15.11	67.74 %
2	15.38	67.74 %
5	16.08	64.52 %

We then tested the results on the *x_test* dataset with a proposal (q(w)) variance of 1 and received a test negative log-likelihood of **7.16** and a test accuracy of **73.33 %.** Note that this sampling method resulted in the same test accuracy as importance sampling, however the test log-likelihood is slightly lower for Metropolis-Hastings MCMC.

For each of the weights selected at random for dataset points 9 and 10, we visualized the 100 (since the sample size is always 100) predictive posterior results in the form of a histogram. The histograms can be seen in figures 3 and 4 and show that the classification value is a distribution itself, rather than simply being a singular value $\in [0,1]$ as it would be in the frequentist approach. For example, comparing the histograms of the predictive posteriors of flowers 9 and 10, we see with a frequentist approach that flower 9 would be classified as 0/false with no problem as it has little variance with the Bayesian methods, but where the Bayesian methods assist us are when classifying flowers like flower 10. Since there are various weights that make the flower > 0.5 and < 0.5 predictive probability, the Bayesian method allows us to analyze whether or not there are any

spikes that indicate the true value of the flower (i.e. at 0.35 in figure 4). For example, in figure 4, the density of the predictions seems to be higher > 0.5, but the true value is 0/false, so we need to consider the spike at ~ 0.35 to not be a random occurrence.



Figures 3 & 4, Predictive posteriors ($P(y^* = 1|x^*, w^i)$ for flowers 9 and 10

Note that since this is a binary classification problem, the predictive posterior is again just the plot of $\hat{f}(x^i, w) = \frac{1}{1+e^{-X^T w}}$.

Q2. Machine Learning Publication

The machine learning research paper selected for analysis is *Fairness Constraints: Mechanisms for Fair Classification* (Zafar, et al. 2017), which aims to eliminate disparate treatment[1] and disparate impact[2] simultaneously. Due to several undesired forms of unexpected outcomes in machine learning algorithms, including indirect discrimination[3] and reverse discrimination[4]. This article becomes increasingly more relevant as we utilize AI and machine learning algorithms to help with important decision-making processes, for example immigration or the hiring process at a company.

The research proposes 2 concepts that could provide a solution to indirect and reverse discrimination, both of which are formulated as minimization problems. First, there is the minimization of the loss function (i.e. maximization of the accuracy) with fairness constraints (the p%-rule[5]). Secondly, they create a minimization of the fairness metric, with an accuracy constraint. To determine the amount of unfairness in the decisions, this methodology uses the covariance between the user's sensitive attributes and the signed distance to the decision boundary[6]. When testing the model, they consider binary and non-binary sensitive attributes, as well as > 1 sensitive

[1] Using sensitive data (i.e. gender, race, age, etc.) parameters when training machine learning models.
[2] Discrimination in decisions made by models that are trained with or without sensitive data.
[3] Without using previous sensitive data, machine learning models are still trained to discriminate against certain groups based on historical data that they receive.
[4] The need to use sensitive data to avoid having any discrimination against groups within society.
[5] The percent of users with a sensitive attribute that pass the decision boundary must have the ratio p:100 with the users which also pass the same decision boundary but do not have the sensitive attribute.
[6] Separates the users in the training set based on their class labels.

attribute which has a correlation to the class labels, which has never been attempted prior to this research.

For both methods, they immediately remove disparate treatment by extracting sensitive features from the dataset and use $\boldsymbol{x_i}$ to be the set of points without sensitive data, and $\boldsymbol{z_i}$ to be the sensitive portion of the data. The basis for comparison of the fairness models is a logistic regression model with the MLE **w** vector as the weights, and a class conditional probability with the predictions being the sigmoid of $\boldsymbol{X^T w}$.

In the table below that summarizes the 2 proposed methods, $\bar{z}$ is the mean of the sensitive attributes, N is the number of training points, $\boldsymbol{d_\theta(x)}$ is the distance from the decision boundary (only dependant on **x** data points, not on sensitive features **z**), and **c** is the maximum covariance matrix. Note that the covariance matrix, c, goes to **0** as the p% rule goes to 100%.

Table 4, Comparison of the fairness and accuracy constraints minimization problems

	Fairness constraints	Accuracy constraints
Minimization	$\underset{\boldsymbol{w} \in \mathbb{R}^M}{argmin}\ L(\boldsymbol{w})$	$\underset{\boldsymbol{w} \in \mathbb{R}^M}{argmin} \left\lvert \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{z_i} - \bar{\boldsymbol{z}})\boldsymbol{d_w}(\boldsymbol{x_i}) \right\rvert$
Constraint	$\left\lvert \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{z_i} - \bar{\boldsymbol{z}})\boldsymbol{d_w}(\boldsymbol{x_i}) \right\rvert < \boldsymbol{c}$	$L(\boldsymbol{w}) \leq (1+\gamma)L(\boldsymbol{w}^*)$

When experimenting on real-world data, they used 5-fold cross-validation, and compared their classifiers to the previously mentioned regularized logistic regression model (which uses sensitive attributes – disparate treatment). The model with fairness constraints ended up achieving fairness by moving the set of users without sensitive attributes to the negative (false) class and moving the other group of users to the positive (true) class. Whereas in the model with accuracy constraints, it was observed that it does not remove the group of users from the positive class label, but rather slowly increases the number of users with sensitive features in the positive class because of the constraints on the accuracy.

References

M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. 2017. "Fairness Constraints: Mechanisms for Fair Classification." *20th International Conference on Artificial Intelligence and Statistics* . Fort Lauderdale, Florida: Journal of Machine Learning Research.