Bayesian Logistic Regression

ROB313: Introduction to Learning from Data

Yizhi (Jojo) Zhou, 1003002396

April 13, 2019

Objectives

In this project, Bayesian logistic regression is used to predict the binary results of one class in the Iris dataset, with a Bernoulli likelihood. The following report briefly describes the implementation of the different appriximation methods that improves the efficiency of Bayesian logistic regression, studies the influence on different parameters/models proposed before the training, as well as the performance of Bayesian compared to the frequentist approach. The report is then ended by a brief report on a summary paper from P. Domigos.

Solution Structure and Strategies

The implemention is composed of the following 4 parts, with more details in the Jupyter Notebook[1]:

- **Probability Functions** This section includes all the log of likelihood P(y—w, x), prior distribution P(w), and the log of posterior distribution which is the sum of these two, as well as their gradients and hessians which will all be later used, for example, when finding the Maximum A Posterior of w, and when calculating the importance weights of each sample of w.

$$\Pr(y|\mathbf{w},\mathbf{x}) = \left[\widehat{f}(\mathbf{x};\mathbf{w})\right]^y \left[1-\widehat{f}(\mathbf{x};\mathbf{w})\right]^{1-y},$$

$$\Pr(\mathbf{w}) = \prod_{i=0}^{D} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{w_i^2}{2\sigma^2}\Big) = \mathcal{N}\big(\mathbf{w}|\mathbf{0},\sigma^2\mathbf{I}\big),$$

  where $\widehat{f}(\mathbf{x};\mathbf{w}) = \Pr(y{=}1|\mathbf{w},\mathbf{x})$ gives the class conditional probability of class 1 by mapping $\mathbb{R}^D \to [0,1]$, and $\mathbf{w} = \{w_0, w_1, \ldots, w_D\} \in \mathbb{R}^{D+1}$. Also, $\widehat{f}$ is a logistic sigmoid acting on a linear model as follows

$$\widehat{f}(\mathbf{x};\mathbf{w}) = \mathsf{sigmoid}\Big(w_0 + \sum_{i=1}^{D} w_i x_i\Big),$$

- **Bayesian with Laplace Approximation**
  - *Gradient Ascent.* The Maximum A Posterior (w_{MAP}) is obtained by performing a Gradient Ascent to maximize the log of posterior probablity. It has an iteration cap and is early-stopped as soon as the gradient's largest value is smaller than the errorbound.
  - *Bayesian Logistic Regression.* This is the main body of Question 1, implemented in a class. It initialzes the prior distribution with different σ^2 values and computes the w_{MAP} corresponding to that distribution. The models with different σ^2's are then compared by their marginal likelihood. The results are shown in the next section.

- **Importance Sampling** Question 2 is implemented in one class, using the probability functions from before. Instead of computinbg one "optimal" w, however, this model directly uses predictions generated from all the w's drawn from the proposal distribution, weighted by their importance

[1]The Jupyter Notebook is attached in the submitted folder.

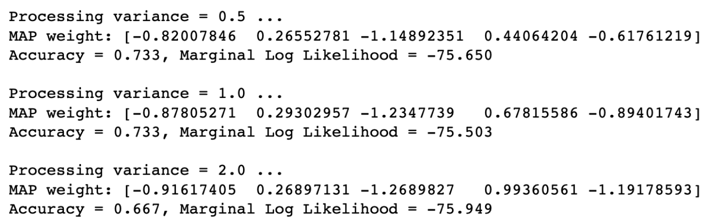
(proportional to the likelihood, prior, and inversely proportional to the proposal, at each w). The accuracy of prediction, as well as the sampling process is visualized and discussed.

- **Metropolis-Hastings MCMC Sampling** This class follows the same structure with the previous one, except for its different method of sampling. Instead of randomly generating a number of samples from the proposal distribution, it tries one random step each time from the proposal distribution, but rejests it if criteria are not met. After 1000 burn-in steps, the algorithm then takes 1 sample in every 100 steps, using which to perform the prediction as in Importance Sampling.

Bayesian Logistic Regression

Bayesian likelihoods with different variances of the prior distribution

With each prior distribution having a variance of $\sigma^2 = 0.5$, 1, or 2, respectively, the marginal likelihood of its posterior distribution which is approximated by the Laplace model, is presented in Figure 1.



```
Processing variance = 0.5 ...
MAP weight: [-0.82007846  0.26552781 -1.14892351  0.44064204 -0.61761219]
Accuracy = 0.733, Marginal Log Likelihood = -75.650

Processing variance = 1.0 ...
MAP weight: [-0.87805271  0.29302957 -1.2347739   0.67815586 -0.89401743]
Accuracy = 0.733, Marginal Log Likelihood = -75.503

Processing variance = 2.0 ...
MAP weight: [-0.91617405  0.26897131 -1.2689827   0.99360561 -1.19178593]
Accuracy = 0.667, Marginal Log Likelihood = -75.949
```

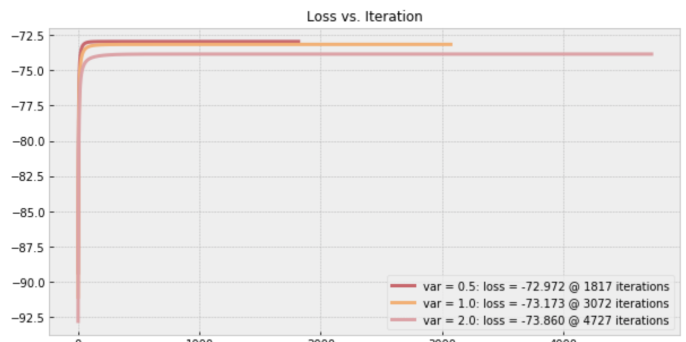
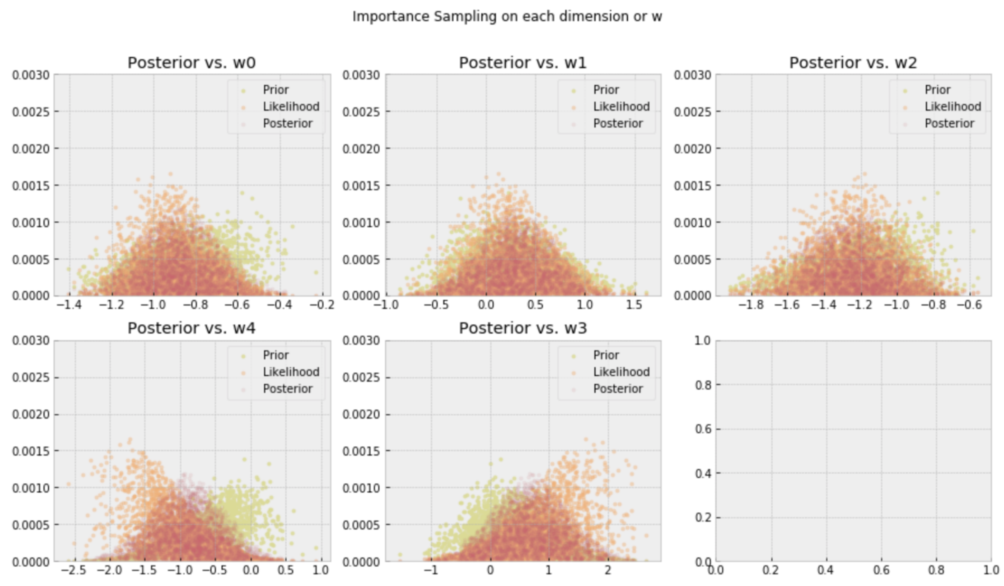
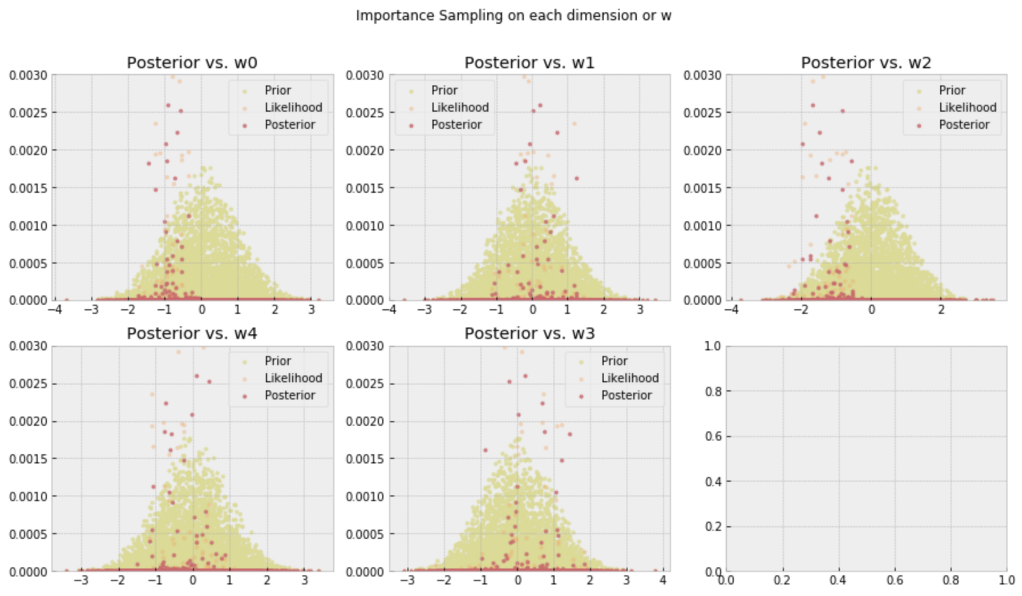


Figure 1: Upper: Results from Laplace approximation. Lower: Gradient Ascent in finding w_{MAP}

As can be seen from the marginal likelihoods, the model's performance isn't in a linear relationship with the variance; instead, it decreases when the variance goes up to 1 and increases again when the variance keeps going to 2. This reflects the concept of **bias-variance tradeoff**, where model complexity increases with the variance where the model is considered "overfitted", and decreases with bias where it is "underfitted". The goal is to find a sweet spot in between underfitting and overfitting, where neither the variance or bias is minimized but the model performance is optimized. In this case, the model with σ^2 =2 is the most complex, also reflected from the fact that it took the most iterations (under the same other parameters such as learning rate) to reach to the w_{MAP}. Meanwhile, σ^2 = **1 returns the best result amoung all**, indicating that it is closer to the optimal model complexity regarding the bias-variance tradeoff; as a result, its accuracy on the test set is also higher than the overfitted model with σ^2 =2.



(a) Laplace appriximation with $\sigma^2 = 1$



(b) Direct Gaussian with $\mu = 0, \sigma^2 = 1$

Figure 2: Visiualization of posterior, prior, and likelihood, with different proposal distributions

Importance sampling and its performance

In face, as long as the proposal distribution isn't too far away from the actual underlying one, the prediction always converge with a large enough amount of samples chosen, as the importance weights eventually correct the bias introduced by sampling from the wrong distribution. The difference is in the amount needed, which is significantly larger for bad proposals and small for good ones. In this case, **the Laplace model computed with a prior distribution of $\sigma^2 = 1$ is chosen as the proposal distribution**, since it is the closest to the actual underlying distribution without extra computation. Figure 2 shows the visualization on each dimension with the chosen proposal distribution, as well as a dummy proposal of simply a zero-mean Gaussian model with variance of 1. It is clear that the samples drawn are completely different, while the accuracies are essentially the same, both being 73.3% on the test set with 3000 samples. It has been tested and shown that, however, when only 3 samles are chosen, the Laplace approximation as the proposal leads to a 66.7% accuracy while the dummy one led to a 46.7%, which agrees with the expectation.

Metropolis-Hastings MCMC sampler and its performance

The implementation of MCMC sampler is already discussed in the second section of the report. The variance doesn't actually contribute to the performance of prediction - whatever the varience is set to be, the prediction accuracy is 73.3% over the test set. The difference that it makes is the rejection rate – in this case, $\sigma_p^2 = 1$ leads to a rejection rate of about 90%, while $\sigma_p^2 = 0.05$ leads to that of about 65%, showing the relationship that **a smaller variance leads to a lower, more desirable rejection rate**. With the chosen $\sigma_p^2 = 0.05$, Figure 3 shows the predictive posterior class-conditional probability samples for the 9th, and the 10th datapoints in the test set as histograms. Though both datapoints are correctly predicted to be "False", it is obvious that the 9th one is way more certain to be "False" (i.e. the probabilities of being "True" have an average close to 0 and a very small variance), while the 10th one is a lot less certain (i.e. the average probability is close to 0.5 and has a large variance). If it were a frequentist approach, which only cares about the most likely paramater rather than a whole list of 100 samples, it could still result in a similar prediction (in terms of accuracy) as well as its confidence indicates in how close the sigmoid value is to 0.5, but it couldn't ever show the distribution of predictions resulting from different parameters - it would never know how the performance/results would change with even small changes of that optimized parameter. For example, the 9th datapoint in this case has featured that would lead to a similar, correct prediction under slight changes of the parameter, while the 10th datapoint varies a lot under such changes. As frequentist approaches essentially doesn't deal with the variance in the uncertainty, they loses such idea of confidence level and are therefore prone to misinterpretations of the values.

```
Accuracy = 0.733 with MCMC sampling
Rejection rate = 0.660 at variance 0.05
```

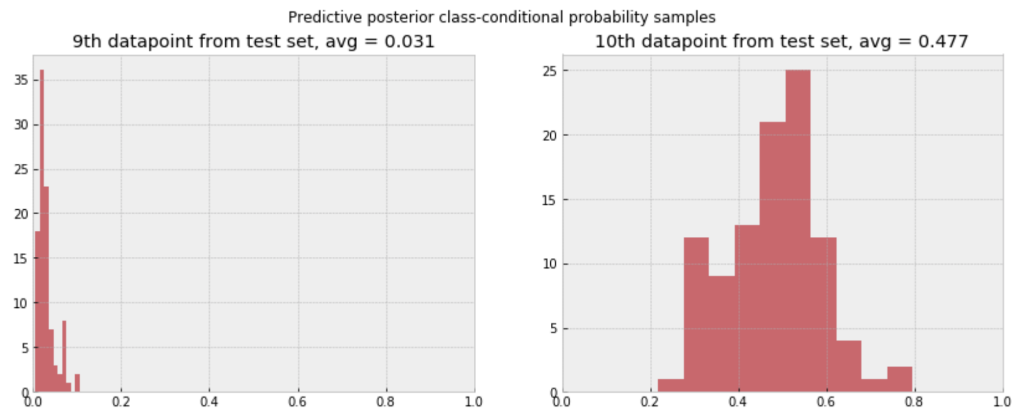


Figure 3: Distribution of predictions on two datapoints

Brief Literature Review

Paper Source

Domingos, P. *A few useful things to know about machine learning.* Communications of the ACM, 55(10), pp. 78–87, 2017. `https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf`

Thoughts

The paper summarizes some "folk wisdom" of machine learning, which, for a beginner like myself, is a great way to have some pitfalls and priorities in mind for my future studies and practices. Some of the topics already resonate with my experiences from the previous assignments in this course. For example, the warning that the training set and test set must never be mixed is also reflected in the Mauna_Loa dataset, where the range of test set's features are disjoint from that of the train set; as a result, when using the train set itself to see the performance, it falsely gives a high accuracy opposite to what the test set generates, especially when using nonparametric methods like kNN. This dataset has also verified the importance of feature engineering, the 7th point in the article. When using the Generalized Linear Model as a representation, the basis functions is the key to successful fitting; the sinusoidal mapping of features gave a high accuracy while a Gaussian fitting was dreadful.

The representation models and optimization techniques we've learned so far are limited, so some other parts of this article are yet to be experienced and then understood. For example, though the Mnist Small dataset has more than 700 dimensions, the impact of high dimensionality hasn't been obvious to us in either the time efficiency or the understanding of how different dimensions contribute to the prediction, both in terms of runtime and accuracy. I greatly appreciate the table listing the different representations, evaluations, and optimizations in the beginning of the paper - It would be great if the article could explain more about their different usages and connect them better with the rest lessons. As mentioned in the 8th point in the article, being smart of designing or choosing the features is the hardest part of machine learning and usually takes the longest time. Other than simply warning us, however, it would be even more helpful if a chart of pros and cons or studies done on each model(method) could be summarized as well.