Samuel Looper
1003044181
2019-04-02

ROB313 Assignment #5

1. Assignment Objectives

The purpose of this assignment is twofold. Firstly, the performance of several variations of a Bayesian inference model for binary classification are evaluated. The prior distribution variation is parametrized, leading to an analysis on model complexity and approximate log marginal likelihood. Then, the accuracy of the model is evaluated for a chosen proposal distribution in importance sampling estimation. The last analysis of this segment of the assignment is for a Metropolis-Hastings MCMC sampler for classification. The latter part of the assignment is a report on an academic paper discussing the practical aspects and implications of machine learning technology. In this part of the report, the paper forming the basis of this discussion is *Concrete Problems in AI Safety* by Dario Amodei.

2. Algorithm Overview

For the first section of the assignment, the algorithms used in the 3 parts of the section was built in a modular fashion, which allowed for the code to run each question and build onto itself. The main loop of the code is sectioned off and contains the main function calls to generate the required data and figures. Outside, several functions were built to perform the specific required calculations. These can handle several different types of input subject to some restrictions.

For part 1a), the algorithm performs stochastic gradient descent to find the maximum a posteriori weight estimate. Next, those weights are used to generate an estimate of the class labels, subject to a sigmoid output function. This MAP estimate is then used to calculate the marginal likelihood using the Laplace Approximation using the equations given in the assignment for the relevant prior and posterior probabilities, as well as the hessian matrices.

In part 1b) the algorithm performs a double cross-validation over both sample size and variance for a proposed distribution. A series of helper functions abstract away the complex calculations from the main loop, with the only assumptions being made for the shape of the proposal and the mathematical operations for importance sampling. Weights are generated, then each point is estimated by summing over all estimates for a single weight on a weighted average with respect to r(*) values. To accomplish this the algorithm simply used nested for-loops. This same algorithm was repeated to calculate test results.

In part 1c), the algorithm was extended to perform a Metropolis-Hastings Monte Carlo Markov Chain estimation. The actual algorithm structure was very similar to part 1b), but the weights are generated via dependent probability distributions. Once again, the burn in, and

thinning are all accomplished via deterministic for loops to avoid infinite runtime errors. Probabilities are evaluated via simple logic according to the Metropolis- Hastings algorithm.

3. Results

3.1 Prior Variance Parameterization

Using the algorithm described in part 2, the variance of the Gaussian weight prior was analyzed with respect to marginal likelihood to reach conclusions about the model complexity. To achieve this, the variance was parameterized, and the algorithm calculated successive marginal likelihoods for each variance value. The results are summarized in the table below

Prior Variance	**Marginal Likelihood**	**MAP Weight Log Likelihood**
0.5	-72.976165	-66.300922
1	-72.759343	-66.758158
2	-74.567939	-67.471149

Table 1: Results from Question 1

As the variance increases, so too does the model complexity. A wider range of weight values are considered as more and more values have a significant weight prior probability associated with them due to the variance increasing. Thus, as we modified the variance, the model gets more and more complex. The tradeoff with variance and complexity is akin to Occam's razor, in the Bayesian sense: the simplest solution that accurately models the system will be the best. "Underfit" your model, and the variance is not large enough to assign significant probabilities to values that should be considered. "Overfit" your model, and too many values are being considered, diluting the importance of the actual range of correct values. In the end, we see that the middle of the 3 cases under consideration has the best performance in terms of marginal likelihood, as one model is "overfitted", while the other is "underfitted" in the bayesian sense. This means that the distributions used in the model consider an unnecessary large amount of weight values by assigning significant probabilities.

3.2 Importance Sampling

Next, the iterative importance sampling algorithm was used in an analysis of the performance of such an algorithm with respect to weight sample size and the proposal distribution. Importance Sampling is used to calculate the predictive posterior for a single given point according to the following equation:

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

Thus, it uses a label estimate vector **y**, and a matrix of input states **X**, and a set of sampled weights **w,** to calculate an approximation of the posterior probability distribution based on a weighted average across the sampled weights.

$$r(w) = \frac{P(y|w, X)\, P(w)}{q(w)}$$

Weights are sampled with respect to a known proposal distribution (q(w)). The choice of proposal distribution is a key factor in the performance of the algorithm. Well designed proposal distributions have heavy tails, approximate the posterior (with the help of Laplace Approximation), and have key variables parameterized by cross-validation. For our analysis we chose a multivariate Gaussian centered at the MAP estimate from the previous section. Based on our analysis of the previous sections we concluded a Gaussian was an successful approximation of the posterior, and chose the MAP as its mean to best fit the range of values considered in the posterior.

The variance and sample size were both parameterized to find optimal values with respect to the negative log likelihood of the validation set estimate. The range of sample sizes considered were [10, 30, 100, 300, 1000], and the range of variances considered were [1,2,5,10]. The optimal parameters were used to compute the testing accuracy and log likelihood. The results are summarized in the table below. It is worth noting that the optimal sample size and variance did tend to vary when the code is re-run, likely due to the stochasticity of the importance sampling method. However, the performance metrics were constant to within 5%.

Optimal Parameter Results			
Sample Size	300	**Testing Log Likelihood**	-10.880925
Proposal Variance	1	**Testing Accuracy**	0.800000
Validation Log Likelihood	-14.63641		

Table 2: Results of the cross-validation for importance sampling proposal distributions

While the justification of the proposal distribution choice may be convincing, it is still important to test this intuition by confirming whether the posterior approximations made at the various sample weights is properly modelled by the proposal distribution. To evaluate this, a new set of sample weights was generated, and at every weight the r-value was calculated. These values were then plotted. Since a 5 dimension plot would not be feasible, a series of univariate gaussians were plotted for the proposal distribution, and r-value samples were plotted with respect to the values at a given index of the weight vector. The 5 plots, for each of the 5

indices, are shown below as the final results for this analysis. As evidence by the plots, the peaks occur at the right mean value, and are generally contained within the gaussian curve. Since most r_values hover around zero, many many more samples would be necessary to fill this distribution and truly confirm whether the proposal distribution is the best fit, although with the data at hand that looks to be the case.

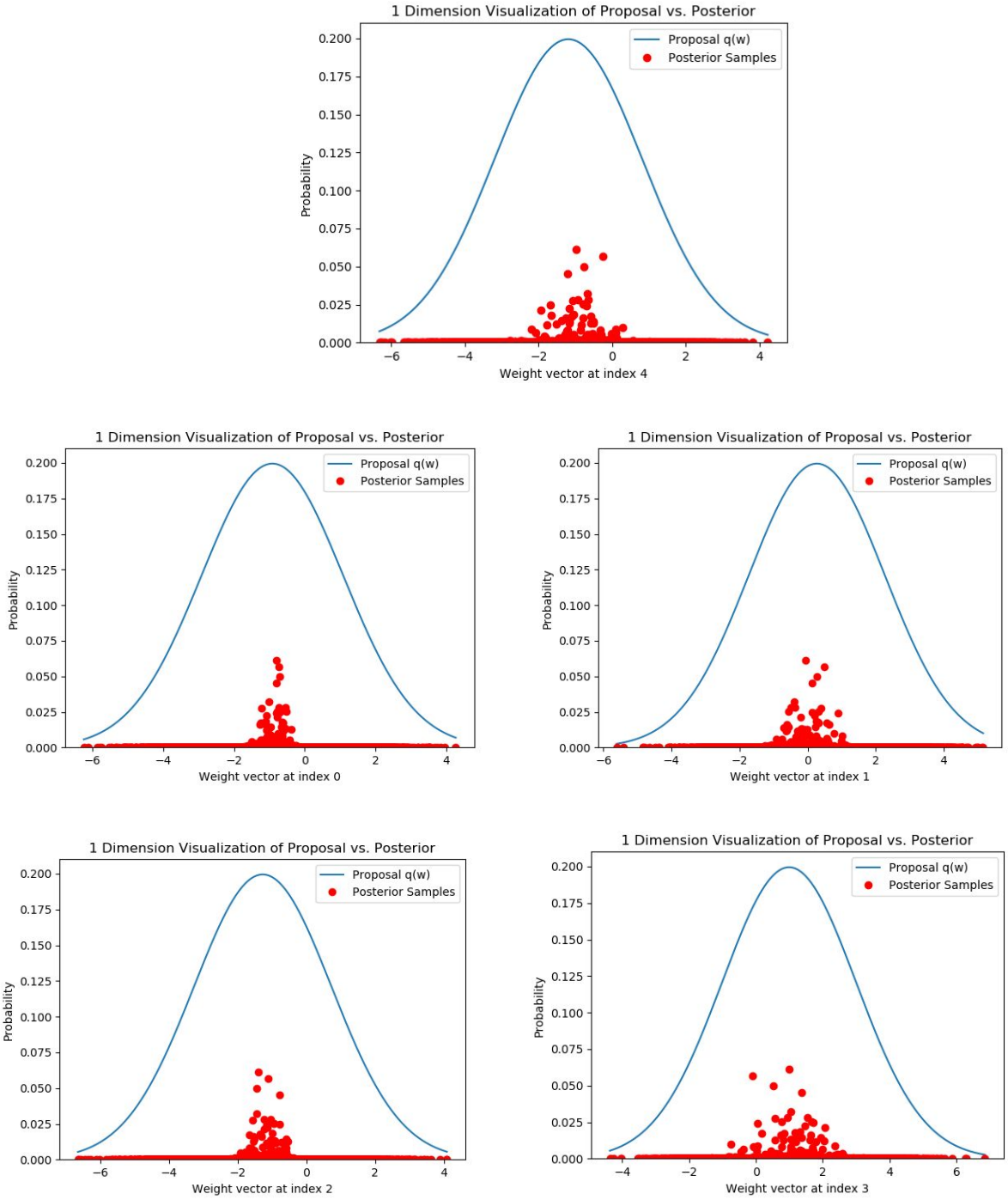

Fig 1-5: Plots of the sample posterior estimates and proposal distributions in 1 dimension

3.3 Metropolis-Hastings MCMC

For the last analysis of this question, a Metropolis-Hasting Monte Carlo Markov Chain was developed, optimal parameters established by cross-validation, the test log likelihood and accuracy noted, and a full analysis performed on two single state variables in the test set.

Firstly, the multivariate Gaussian was used to sample new weights in the weight generation process, similarly to question 2. Once again, the variance was established by cross validation, minimizing the negative log likelihood of the validation set. The results are summarized in the table below. Worth noting that the variance had a much more pronounced effect in the performance of the system with this method, and that in general the system performed slightly better with respect to log likelihood than previous algorithms.

Optimal Parameter Results			
Sample Size	100	**Validation Accuracy**	0.806452
Proposal Variance	8	**Testing Log Likelihood**	-8.097017
Validation Log Likelihood	-14.549997	**Testing Accuracy**	0.800000

Table 3: Results of the cross-validation for Metropolis Hasting Monte Carlo Markov Chain

Next, the results on specific states in the testing set were analyzed by constructing a histogram of posterior prediction probabilities for flowers 9 and 10 in the iris test set. The results are summarized below.

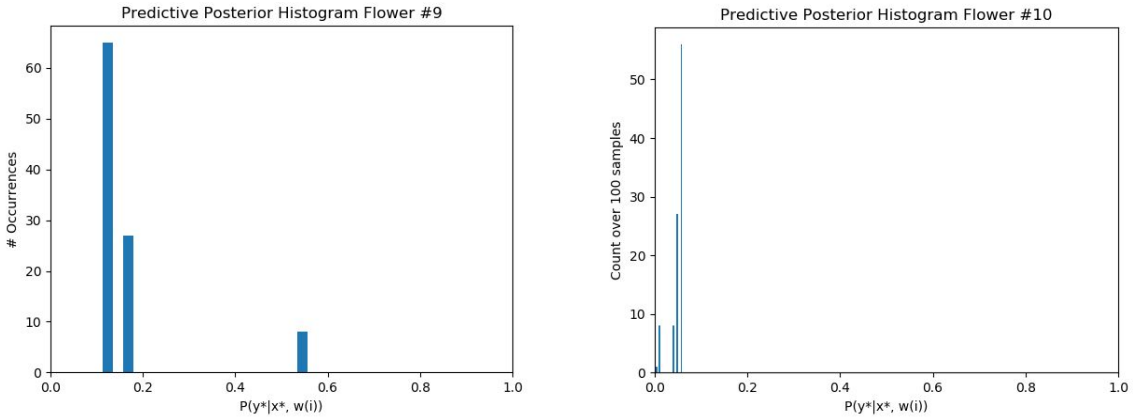

Fig 6-7: Predictive posterior histograms for flowers #9 & #10 from iris test set

The histograms represent the individually evaluated predictive posteriors used in the Metropolis-Hastings sum for the test set predictions. Thus we have a sample size of 100 posteriors, over which an approximate integral, or weighted average, was calculated to come to a final result. Whereas a frequentist would only be considered with the final result, the above visualization of the posterior probability distribution clearly shows that two states can have the same result, but different distributions in a bayesian approach. Indeed, both would probably indicate the flower is not in class 2 (*iris virginica*), but whereas the second has a high degree of certainty, the first has more uncertainty associated with it. The distributions give rich and nuanced information about this very uncertainty which can be useful in assessing a model's performance or qualifying a final result, which is impossible to gather from a frequentist approach to the problem.

4. Machine Learning Paper Review

For this part of the assignment I will be reviewing Concrete Problems in AI Safety by Dario Amodei et al. This paper revolves around machine learning accidents, defined as unintended adverse events that can have harmful consequences to an environment, which emerge from machine learning systems when they are poorly implemented. The paper is very practically focused; accidents are framed exclusively on existing real world machine learning systems. I think this was an excellent choice of framing, as it makes for a more novel and productive discussion avoiding the speculation around general AI and other future systems. However, they do not specifically state what machine learning paradigms and techniques their analysis applies to (reinforcement learning and supervised learning) until several pages into their analysis, making it harder to relate their high level analysis to the technical details of any given type of machine learning system.

Nonetheless, the paper does a thorough and well organized analysis of machine learning accidents in the following pages. Firstly they categorize accidents relative to their origin. Poor objective function errors are discussed with respect to two subcategories of error: negative side effects and reward hacking. The former type of error occurs when a machine learning system affects its environment outside of the designer's intent. The paper then outlines several techniques to reduce these kinds of accidents, which mostly revolve around regularizing the variance of the environment state in an objective function. The latter type of error occurs when a model is incentivized to find solutions that are technically correct but do not achieve the designers intent. These accidents are very difficult to solve, but the paper outlines some rules of thumb and techniques to reduce reward hacking. Next, the paper analyzes issues caused by poor scalability of the algorithm, which occurs when the objective function cannot be calculated at a desired frequency, leading to sparse data for the machine learning system. The paper focused on techniques in the realm of semi-supervised learning to tackle these types of accident. The next type of error was unsafe exploration, which happens in reinforcement learning when exploration or training actions have significant negative consequences. Finally,

the last type of error is due to systems not being robust to distributional shift, or differences between training data and the real time system input.

The methodic categorization of machine learning systems made the paper well organized and easy to read. Each category was supported by several real world examples, enhancing the understanding of the paper. Finally, the techniques to tackle accidents were similarly well organized, and explained in such a way that I could understand the issues and practical considerations in topics we have not had the theoretical background for. Indeed, it was a shame that most of the analysis centered around reinforcement learning; I would have wished more could be related to supervised learning so my technical understand of that machine learning paradigm could give me better context for this analysis. All in all, this paper was extremely focused on the practical applications of machine learning systems, made ample use of real world examples, and well organized into categories. For these reasons I find the paper very informative and well written.