Applying Bayesian Logistic Regression on IMDB 50K Dataset

Model Selection. There are only two classes (positive, negative) used for sentiment analysis of the 50K IMDB dataset. Hence, we applied Bayesian logistic regression to accurately predict review sentiment, which is often used for binary classification.

Data Pre-Processing. To perform Bayesian logistic regression, we must first decide on how to represent our reviews as input vectors. to be used to represent each review. We decided to transform the dataset through (1) TF-IDF vectorization and (2) dimensionality reduction. TF-IDF provides additional information on how important the word is for determining review sentiment across the training corpus. Applying the vectorizer helps account for artifacts such as stop words without additional pre-processing. Dimensionality reduction through single value decomposition (SVD) constrains the number of features in our dataset. Reducing the size of the input vectors from 90K to 5K speeds up training significantly and can be effectively applied to sparse data.

Training with Minibatch Gradient Ascent. To obtain the MAP estimate of the coefficients, we randomly sampled a minibatch of input data to calculate the gradient per training iteration:

Algorithm 1 Minibatch stochastic gradient ascent

```
Initialize $\beta[0] \leftarrow 0$
Initialize **loss** $\leftarrow []$
**while** numiters < maxiters **do**
    randomly sample $(x_b, y_b)$ from training dataset $X, y$
    calculate $g$ from minibatch:
        $$g = -\lambda\beta[t] + \frac{n}{B}\sum_{b=1}^{B}(y_b - \sigma(\beta^T x_b))x_b$$
    evaluate out of sample negative log-likelihood:
        $$L = -\sum_{b=1}^{B}(y_b \log(\sigma(\beta^T x_b)) + (1 - y_b)\log(1 - \sigma(\beta^T x_b))$$
    append $L$ to **loss**
    perform gradient step
        $$\beta[t+1] = \beta[t] + \frac{\rho}{t} g$$
**end while**
return $(\beta[N-1]$, **loss**)
```

Influence of Prior Precision on Loss and Accuracy. When applying stochastic gradient descent on the training data, we wanted to examine how much the prior would influence the model using our hyperparameter λ. Figure 1 demonstrates our evaluation of the model on λ values of 0.1, 0.5, 1, 3, and 5. As expected, we observe a far noisier stochastic gradient as we increase in the two plots.

However, the variance in the accuracy and loss over the training iterations decreases as the gradient ascent algorithm approaches convergence.

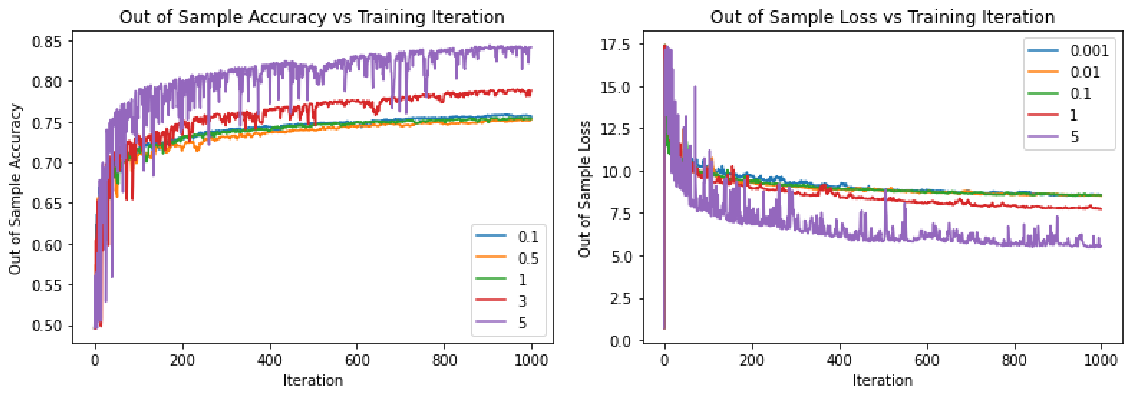

Figure 1. Out-of-sample metrics for different λ values across 1000 learning iterations.

While the prior's greater influence introduces greater volatility in the accuracy and loss curves for our regression model, we also observe a substantial increase in out-of-sample accuracy and decrease in out-of-sample loss. Hence, the improvements in accuracy are worth the price of the noisier gradient to achieve greater performance. We determined that $\lambda = 5$ achieved the best balance.

Effect of Learning Rate and Batch Size on Optimization. To determine the optimal learning rate ρ and batch size B for training our model, we evaluated the out-of-sample accuracy and loss across different values of the hyperparameters.

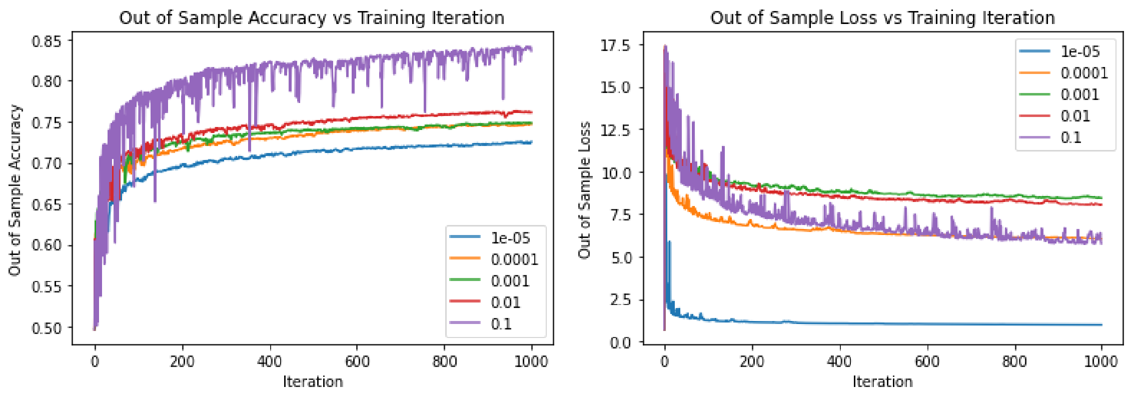

Figure 2. Out-of-sample metrics for different ρ values across 1000 learning iterations.

When testing across different learning rates, we encountered a peculiar trend in the out-of-sample losses. At $\rho = 10^{-5}$, the loss curves in Figure 2 are substantially lower than the alternative learning rates. Yet, the more optimal loss curve does not translate to the greatest accuracy. A possible explanation is that at a lower learning rate, the training has encountered a different local minimum which would not have been possible to reach without a small learning rate. In terms of accuracy, however, we observe improved performance similar to the results shown in Figure 1.

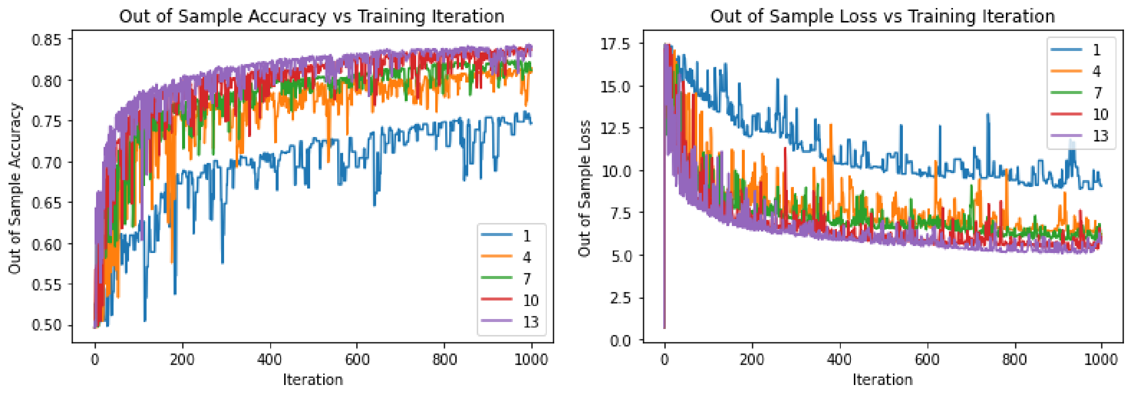

Figure 3. Out-of-sample metrics for different batch sizes B across 1000 learning iterations.

As predicted, we can observe a much noisier gradient when $B = 1$. Due to the unbiased nature of the stochastic gradient, however, the single sample batch still allows the model to converge on the loss. The main trade-off to consider between the batch sizes is between accuracy and computational complexity. An increase allows us to achieve better accuracy and a less noisy gradient at the expense of runtime as shown in Figure 3. In the case of the IMDB dataset, however, there seems to be little runtime cost compared to the considerable accuracy improvement. Hence, we decided to opt for greater batch sizes.

Final Results and Best Accuracy. Using the optimal hyperparameters that we determined based on previous evaluation, we achieved a final loss of 3.880 and an accuracy of 0.881. Our results are comparable in accuracy to the scikit-learn SGD Logistic Classifier which achieved an accuracy of 0.8849, although our implementation does not compute as efficiently. The final out-of-sample accuracy and loss curves from Figure 4 demonstrate overall greater performance as the model continues to train over the 20,000 iterations which equate to 7.5 epochs.

There are ways to improve the final result through techniques such as (1) applying cross-validation to tune the hyperparameters, (2) shuffling the data per epoch, and (3) requiring the minibatches to cover all datapoints for each epoch instead of using random sampling. However, the model was still able to provide insight into the effect of the prior and the hyperparameters on the results.

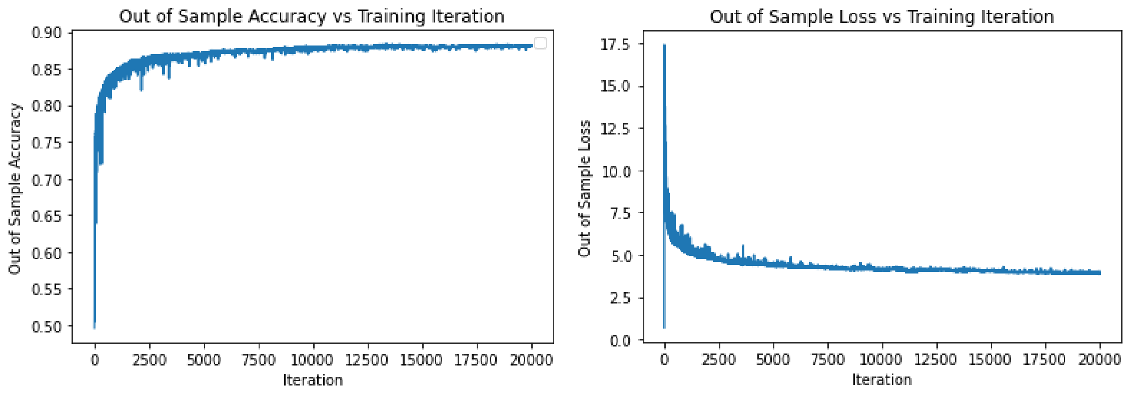

Figure 4. Out-of-sample accuracy and loss ($B = 15$, $\lambda = 5$, $\rho = 0.1$) across 20,000 learning iterations.

Further Analysis and Interpretation of the Posterior Coefficients. Due to applying principal component analysis, it difficult to interpret the exact features that have contributed to the regression model. Unlike with standard count or tf-idf vectorizers, we cannot simply match a feature to a corresponding word. Hence, we instead examined the mean and variance of the coefficients as shown in Figure 5.

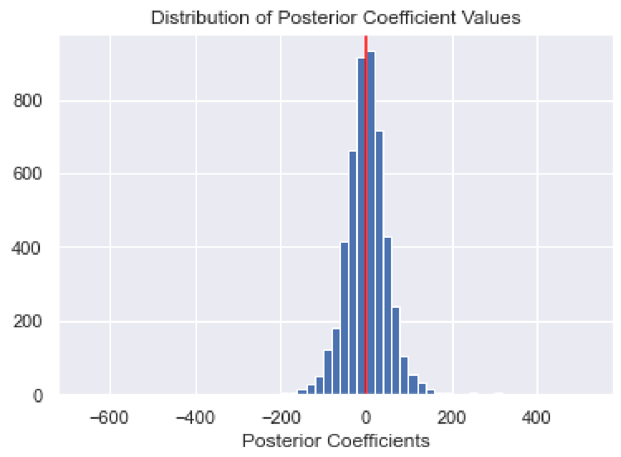

Figure 5. Histogram of posterior coefficient (β) distribution.

The distribution of the coefficients resembles a Gaussian distribution centered at mean 0.763 with standard deviation 59.30. The mean and small standard deviation shows that most of the coefficients are close to 0, which indicates that only a small proportion of features contribute significantly to the predicted sentiment. Upon examining the histogram further, we observe that there are outliers that hold significant values since our distribution ranges from a minimum of -661.1 to 499.6. Perhaps these features encapsulate words that hold high sentiment value such as "horrible" or "fantastic".