An Investigation of SLOE Estimation for Unbiased High-Dimensional Logistic Regression

Ameya Shere as12366@nyu.edu
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA

Shuvadeep Saha ss15592@nyu.edu
Department of Mathematics
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA

Editor: Dr. Rajesh Ranganath and Mark Goldstein

Abstract

Almost all modern software packages for logistic regression are predicated on classical large-sample theory. As (Sur and Candès, 2019) have shown, these models produce biased estimates of the MLE for data with a high aspect ratio (large number of features, but number of features is still less than number of rows). Recently, (Yadlowsky et al., 2021a) have built on this work by proposing a computationally efficient SLOE estimator that corrects for this bias. In this paper, we investigate the behavior of this SLOE estimator for different kinds of data, including Gaussian, non-Gaussian planar normalizing flow, heavy-tailed non-Gaussian, latent Gaussian, and latent non-Gaussian data. We simulate these datasets for different data sizes and evaluate the uncertainty quantification and model performance for SLOE, relative to the baseline Python statsmodels Logit model (Seabold and Perktold, 2010). We find that SLOE works best for Gaussian data with and without latent relationships, and for data from a non-Gaussian heavy-tailed distribution. Ultimately, our results indicate that the main improvement SLOE provides over classical large-sample logistic regression models is a more accurate understanding of uncertainty, rather than better performance on evaluation metrics or execution time. The code to run our experiments is publicly available on GitHub: `https://github.com/aashere/sloe-ml-project`.

Keywords: Logistic Regression, SLOE Estimator, Unbiased MLE

1. Introduction

Generally speaking, most software packages and libraries for logistic regression are built strictly on the basis of large-sample asymptotic theory of the maximum likelihood estimator (MLE). When the number of samples n and the number of covariates p are of comparable size, i.e., the number of features is a non-negligible fraction of the sample size, these standard approximations behave quite poorly even when the sample size is large. Recently, (Sur and Candès, 2019) showed that these issues in the high-dimensional regime can be mitigated by the use of a new approximation of the MLE's sampling distribution. It is to be noted that here we use the word high-dimensional strictly with respect to the aspect ratio of the dataset; that is, the ratio of the number of columns (p) to the number of samples (n), i.e., we define the aspect ratio as $\kappa := p/n > 0$. κ is held fixed as both n and p go to infinity.

A recent development was made by (Yadlowsky et al., 2021a) where the authors introduce a more practical SLOE (Signal Strength Leave-One-Out Estimator) estimation technique for such high-dimensional data, which has similar runtime but lesser requirements of computational power than what was proposed in the initial work by (Sur and Candès, 2019) and also (Zhao et al., 2022). The SLOE estimator is essentially a dimensionality correction applied to the MLE estimator.

The scope of this work is to investigate the range-of-applicability of the SLOE model, as applied to various simulated datasets such as Gaussian, non-Gaussian (generated using a normalizing planar flow), heavy-tailed non-Gaussian, latent Gaussian, and latent non-Gaussian.

2. Background

We intend to use the Python statsmodels Logit model as a baseline for our experiments (Seabold and Perktold, 2010). In order to verify that this model is indeed biased, we set up a study similar to the one conducted by (Sur and Candès, 2019).

We take high-dimensional data with $p = 800$ and $n = 4000$ which has the dimensionality constant $\kappa = 800/4000 = 0.2$. The matrix covariates are i.i.d. $\mathcal{N}(0, 1/n)$. We then calculate the probabilities (μ_i) using the sigmoid function, following which we sample Y from the Bernoulli distribution, $Y \sim \text{Bern}(\mu_i)$. The first $\frac{1}{8}$ of the coefficients are set to 10, the second $\frac{1}{8}$ are set to -10, and the rest to 0. The samples are scaled in such a way that the signal strength $\gamma^2 := \text{Var}(\beta^T X_i) = 5$. The scaling is mainly done such that the signal strength $\gamma = \sqrt{5} \approx 2.236$; that is, 95% of the observations are such that $-4.472 \leq \beta^T X_i \leq 4.472$ as shown in Figure 1.

Figure 2 shows that in the first quarter of the plot, the black line, which is the true β, does not pass through the midpoint of the red dots, which is the MLE $\hat{\beta}$. But for the rest of the plot with zero regression coefficients, the true model passes through the center of the red dots because their corresponding covariates have no effect on the MLE. This clearly shows that our baseline model is biased and requires correction.

3. Related Work

In an early work (Sur and Candès, 2019) have shown that the asymptotic behaviour of the MLE can be fully defined by solving a system of nonlinear equations in three variables, namely the standard deviation (σ), the bias (α), and an auxiliary parameter (λ), which is implicitly dependent on the aspect ratio κ and the signal strength γ^2. In order to set up the equations, it is required that the value of γ is known beforehand. (Sur and Candès, 2019) also suggest a ProbeFrontier method for estimating the value of γ in a heuristic way. This is necessary because the signal strength parameter is the key quantity required to set up the system of equations and unfortunately it is not directly observed because of the fact that γ is itself a function of β. The underlying idea is to search for a sub-sample size $n' < n$

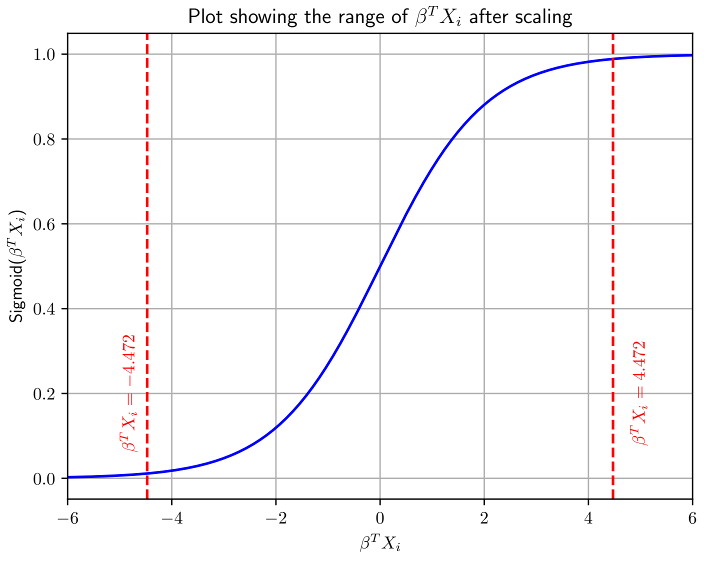

Figure 1: Plot showing the range of the observations after re-scaling $\beta^T X_i$. This is to ensure that the log-odds ratio does not grow so large that the output of the sigmoid function is 0 or 1 trivially (at the corners of the curve) (Sur and Candès, 2019).

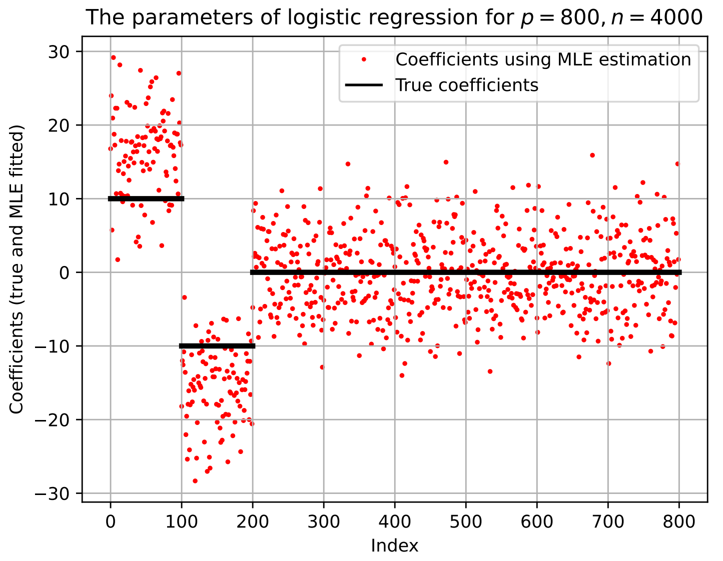

Figure 2: True coefficients β (in black) versus MLE estimates $\hat{\beta}$ (in red) from Python statsmodels Logit model. It is worthwhile to observe the bias in the MLE estimates i.e., the black line does not pass from the middle of the red dots. Clearly, the MLE overestimates the magnitude of the coefficients (above the true estimate)

such that the observations in the sub-sample are linearly separable. This alters the aspect ratio κ while keeping the signal strength the same as in the original sample.

In the above method, one would require to subsample the data repeatedly for various candidate values of aspect ratio, followed by another program ensuring that the linear separability condition of the subsample is met. Checking for separability each time is computationally expensive and time consuming which makes this method tricky and not readily usable. This has motivated (Yadlowsky et al., 2021a) to develop a more practical algorithm for the dimensionality correction. The estimating equations proposed by (Sur and Candès, 2019) are in terms of the original signal strength given as $\gamma^2 := \lim_{n \to \infty} Var(\beta^T X)$. These equations are reparameterized in terms of the corrupted signal strength given as $\eta^2 := \lim_{n \to \infty} Var(\hat{\beta}^T X)$ by (Yadlowsky et al., 2021a).

The term η^2 is called the corrupted signal strength because one can think of it as a corrupt estimate $\hat{\beta}$ of the true signal β. In order to estimate η, we also have to estimate the predictor covariance which is the key challenge here owing to the dependence between η and the covariance. (Yadlowsky et al., 2021a) also suggest that this dependence can be mitigated by a leave-one-out (LOO) estimator. Later, the authors propose a further refinement of the LOO estimator by introducing the Signal Strength Leave-One-Out (SLOE) estimator which is the main objective of their study. This estimator gives us unbiased results for high-dimensional (higher-aspect-ratio) datasets as well as smaller-aspect-ratio datasets. In the results presented by (Yadlowsky et al., 2021a), the authors validate the SLOE model using Gaussian and also non-Gaussian genomics data, but no inference is made about the model performance for other non-Gaussian distributions or Gaussian and/or non-Gaussian latent models. In our present work we investigate this SLOE model for various simulated datasets.

4. Experimental Setup

We use the Python libraries PyTorch, NumPy, pandas, scikit-learn, and matplotlib to generate data and run our experiments (Paszke et al., 2019; Harris et al., 2020; Wes McKinney, 2010; Pedregosa et al., 2011; Hunter, 2007). The code to run our experiments is publicly available on GitHub: `https://github.com/aashere/sloe-ml-project`.

4.1 Simulated Data

We simulate 5 different kinds of data to test our model: Gaussian data, non-Gaussian planar flow data, heavy-tailed non-Gaussian data, latent Gaussian data, and latent non-Gaussian planar flow data. In each case, the data matrix is of dimension $n \times p$, where n is the number of samples, and p is the number of columns. Note that $\kappa = \frac{p}{n}$. After generating the data matrix, it is standardized using scikit-learn's RobustScaler.

4.1.1 Gaussian Data

We generate a jointly Gaussian data matrix X by drawing n samples from multivariate normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$, where Σ the $p \times p$ covariance matrix of n p-dimensional $\mathcal{N}(0,1)$ random variables. We use this covariance matrix instead of I_p so that the samples will have nonzero covariance with each other.

4.1.2 Non-Gaussian Normalizing Planar Flow Data

We generate non-Gaussian data by first following the procedure to generate jointly Gaussian data, which results in data matrix X. Then, we use a planar flow, a type of normalizing flow, to transform the distribution of these samples.

In general, normalizing flows provide a method for transforming simple distributions into more complex distributions. The simple distribution is passed through a sequence of invertible transformations called simple flows, and after each transformation, the result is normalized using the determinant of the Jacobian matrix so that the transformed function is still a probability density.

In our case, we use only one transformation, and we choose a planar flow. Given the scaling vector $u \in \mathbb{R}^p$, tangent vector $w \in \mathbb{R}^p$, shift $b \in \mathbb{R}$, and non-linearity $h : \mathbb{R} \to \mathbb{R}$, a planar flow f_{pf} on $\mathbb{R}^p$ is defined by

$$f_{pf}(z) = z + uh(w^\top z + b)$$

(Kong and Chaudhuri, 2020)

For our data, we set $u = 1.25$, $w = v$, where v is a randomly generated p-dimensional vector, and $b = 0.01$. For all samples $x \in X$, we apply f_{pf} to transform X into non-Gaussian dataset X_{pf}. The difference in distributions of one column of X and X_{pf} can be seen in Figure 3. We observe that this column has a Gaussian distribution in X, but after the planar flow has been applied to all the samples in X, the same column has a bimodal distribution in X_{pf}. Thus, the data in X_{pf} comes from a more complex non-Gaussian distribution.

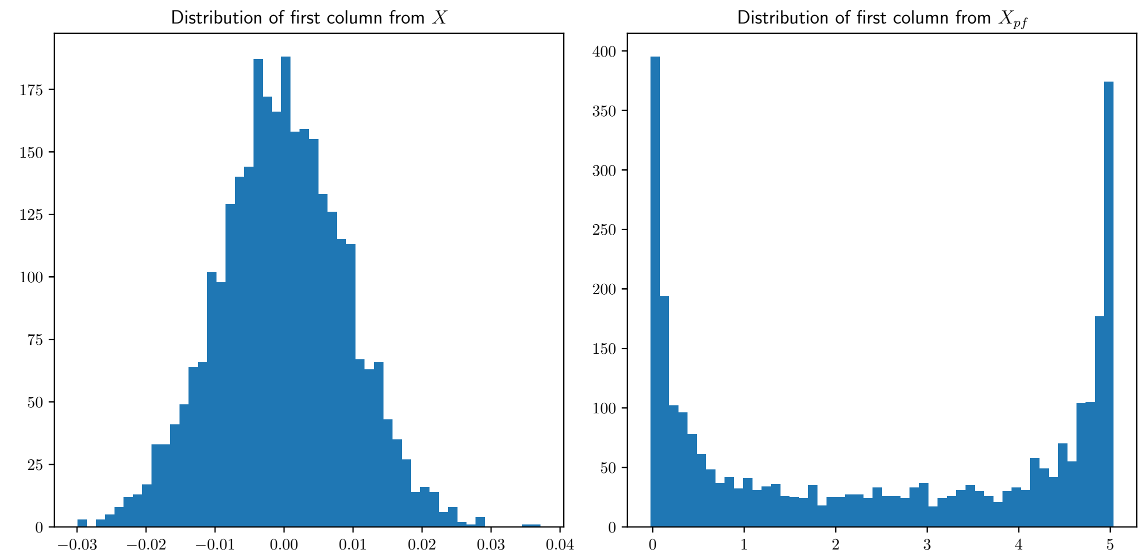

Figure 3: Distribution of the first column of the data matrix before and after the planar flow is applied. In this example $n = 3000$ and $p = 600$.

4.1.3 Heavy-Tailed Non-Gaussian Data

We generate heavy-tailed data by sampling from a Student's T distribution with 3 degrees of freedom. For all $1 \le i \le n$

$$x_i \sim \text{StudentT}(3)$$

4.1.4 Latent Gaussian Data

We generate data matrix X_l, whose samples all have relationships with a common latent variable. This data generating process is as follows. We first randomly generate 4 sets of parameters $\{(\mu_k, \Sigma_k)\}_{k=1}^4$, where $\mu_k \in \mathbb{R}^p$ and $\Sigma_k \in \mathbb{R}^{p \times p}$. To generate one sample x_i, we draw latent variable z_i from a $K = 4$ categorical distribution with equal class probabilities. We select the set of parameters corresponding to this z_i, and we generate x_i as a Gaussian random variable with these parameters. This process can be summarized as follows:

$$z_i \sim \text{Categorical}(1, ..., 4)$$
$$x_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

4.1.5 Latent Non-Gaussian Normalizing Planar Flow Data

For this dataset, we repeat the same data generating process as for the latent Gaussian dataset, but after generating the data, we transform all the samples using the same planar flow used in the non-Gaussian normalizing planar flow dataset.

4.1.6 Generating Labels

The data labels are generated according to the same procedure outlined in (Yadlowsky et al., 2021a):

$$\beta_j = \begin{cases} \frac{2\gamma}{\sqrt{p}} & 0 \le j < \frac{p}{8} \\ \frac{-2\gamma}{\sqrt{p}} & \frac{p}{8} \le j < \frac{p}{4} \\ 0 & j \ge \frac{p}{4} \end{cases}$$
$$y_i = \text{round}\Big(\frac{1}{1 + \exp(-\beta^T X_i)}\Big)$$

4.2 Experiments

We simulate all our datasets for all combinations of simulation parameters:

$$\kappa = \{0.1, 0.2\}$$
$$n = \{500, 1000, 3000\}$$

We split the data 80-20 into training and testing sets using scikit-learn's train_test_split function. The training set is used to fit the model, and the testing set is used for evaluation. We use the Python statsmodels Logit model as our baseline model. We use the Python SLOE code implemented by (Yadlowsky et al., 2021a) to test the SLOE model (Yadlowsky et al., 2021b).

There are two main goals of our experiments. First, we want to quantify the uncertainty in the SLOE model. Second, we want to evaluate SLOE model performance. To achieve these goals, we record the following for each model:

1. A histogram of the p-values for the null coefficients (the coefficients that we know to be 0)
2. A plot of prediction intervals for 8 samples from the testing set
3. F1-score for the model
4. Time taken to fit the model

5. Results and Discussion

5.1 p-values Under the Null Hypothesis

For the Gaussian data, we observe that for the baseline model, the distribution of p-values for the null coefficients is skewed to the left, whereas the distribution of p-values for the SLOE model is more uniform. These results can be seen in Figure 4, and they are consistent with the expected result for Gaussian data as per (Yadlowsky et al., 2021a).

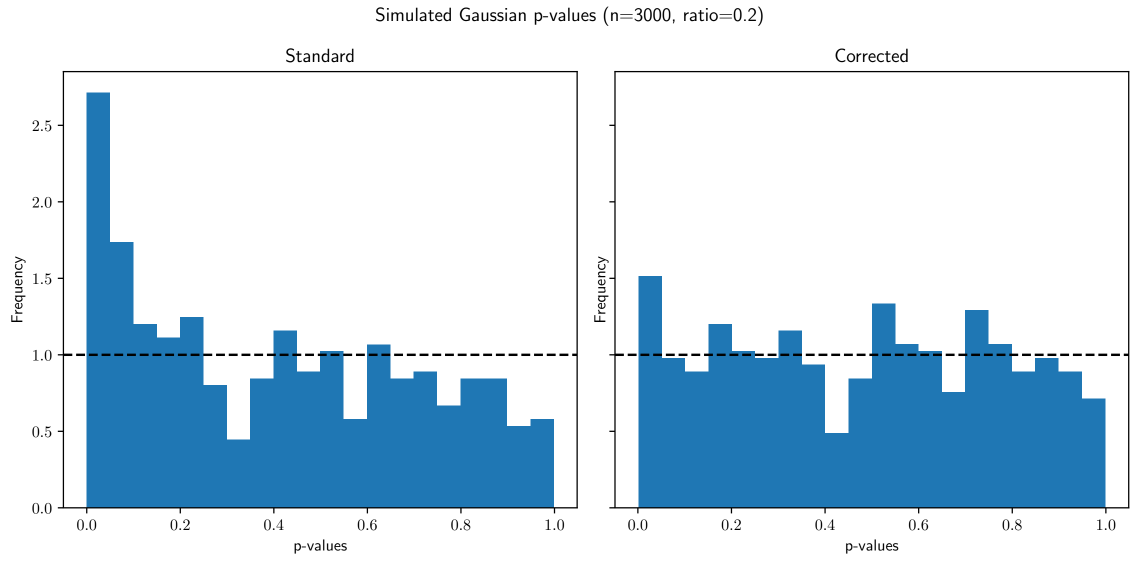

Figure 4: Distribution of p-values under the null hypothesis for the baseline model (left) and the SLOE model (right), fitted to Gaussian data. In this example, $n = 3000$ and $p = 600$.

The results for the non-Gaussian planar flow data can be seen in Figure 5. We observe that the SLOE model's p-values are skewed to the left, which indicates that the null p-values for the SLOE model are not much more trustworthy than the baseline null p-values.

For both the heavy-tailed non-Gaussian data and the latent Gaussian data, we observe similar results as for the Gaussian data, which means that the p-values of the SLOE model's coefficients are more accurate than the baseline p-values for these kinds of data.

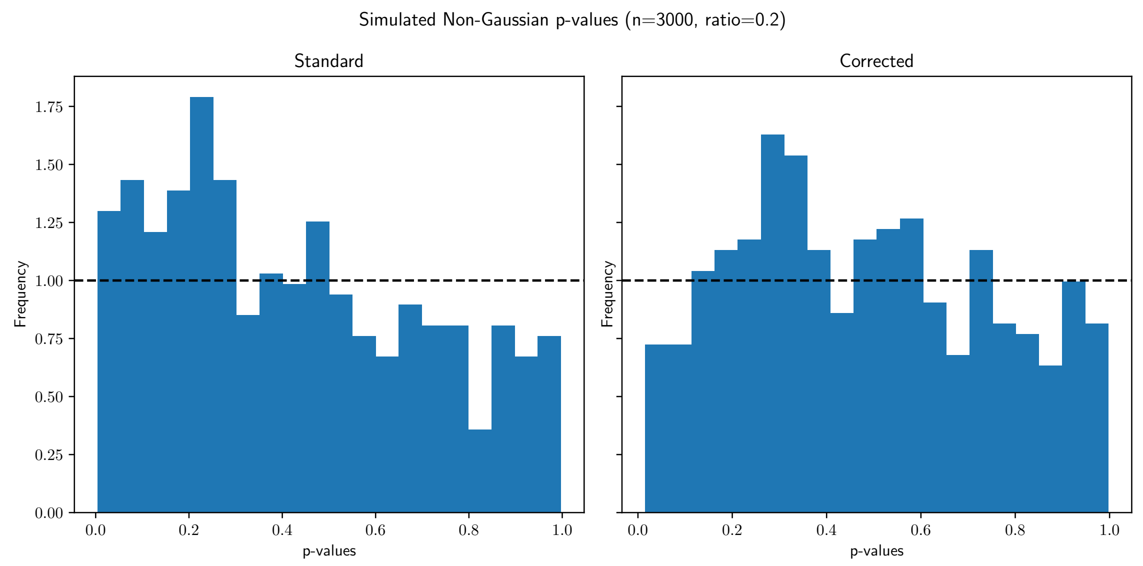

Figure 5: Distribution of p-values under the null hypothesis for the baseline model (left) and the SLOE model (right), fitted to non-Gaussian normalizing planar flow data. In this example, $n = 3000$ and $p = 600$.

Finally, for the latent non-Gaussian planar flow data, we observe in Figure 6 that the SLOE model null p-values are skewed to the right. It is not clear what this means, other than that this kind of data violates the assumptions of the SLOE model.

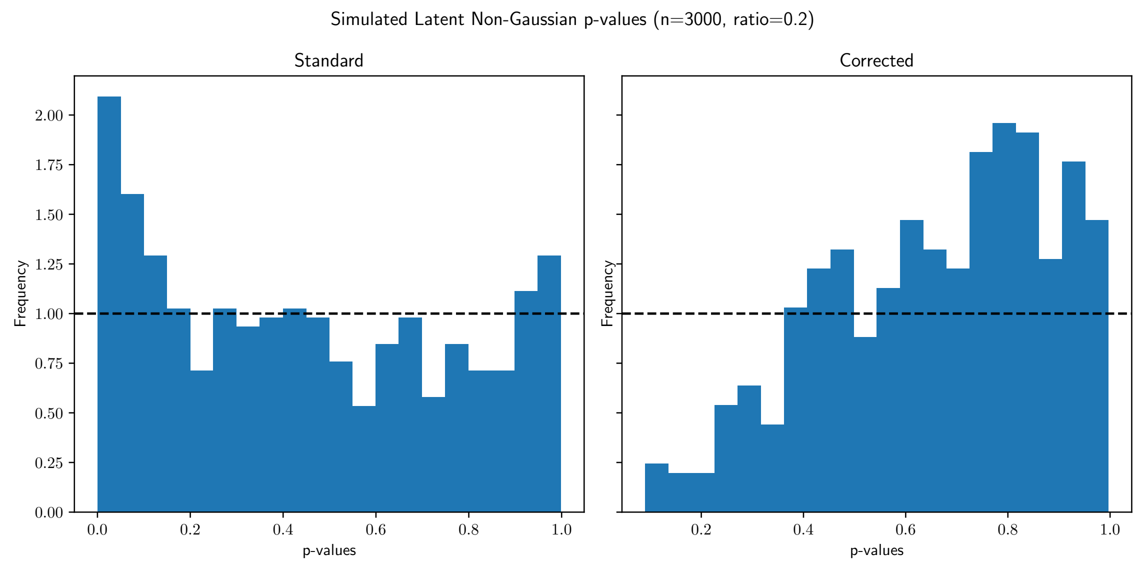

Figure 6: Distribution of p-values under the null hypothesis for the baseline model (left) and the SLOE model (right), fitted to latent non-Gaussian normalizing planar flow data. In this example, $n = 3000$ and $p = 600$.

5.2 Prediction Intervals

For the Gaussian data, we observe in Figure 7 that many of the baseline prediction intervals are narrower than the SLOE prediction intervals. This result matches the results from (Yadlowsky et al., 2021a), because it indicates that the prediction intervals for biased models are often overconfident, and the SLOE model corrects for this with wider intervals. We observe a similar result for all the datasets used in this experiment, which suggests that regardless of the type of data, baseline biased logistic regression models tend to have overly high confidence that their predictions are right.

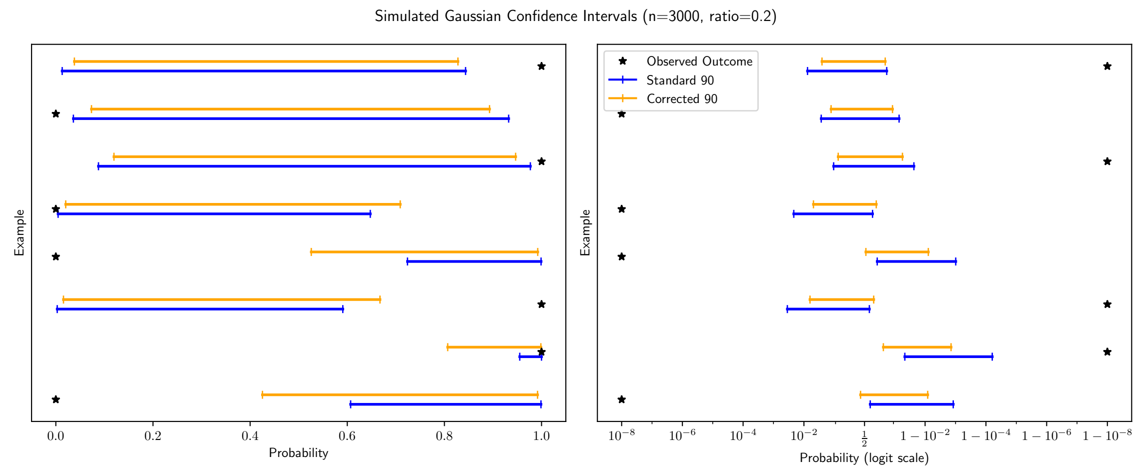

Figure 7: Prediction intervals for the baseline (blue) and SLOE model (orange) for 8 held-out Gaussian test data points. The same plot is shown in linear scale (left), to emphasize interval width, and logit scale (right), to emphasize interval centers. In this example, $n = 3000$ and $p = 600$.

Interestingly, we note that while in most of the datasets, the center of the intervals for the baseline and SLOE models are about the same, for the heavy-tailed non-Gaussian data, we observe that not only are the SLOE intervals wider, a few are also centered in a different place (Figure 8). Looking at the $\hat{\alpha}$ values for the SLOE model across all the datasets for $n = 3000$ and $\kappa = 0.2$, we observe that the heavy-tailed data has the highest $\hat{\alpha}$, at around 1.9. According to (Yadlowsky et al., 2021a), the $\hat{\alpha}$ value is the bias inflation factor; in other words, this number means that for heavy-tailed data, the logits in the baseline model have been inflated by an estimated 90% as compared to the SLOE model. It makes sense that such a high inflation would lead to prediction intervals that are not only too narrow, but also centered at the wrong probability.

In fact, a further look at the $\hat{\alpha}$ values in Table 1 reveals that the heavy-tailed data has the highest $\hat{\alpha}$ value when $n = 1000$ and $n = 3000$, which may suggest that for larger data sizes, heavy-tailed data suffers from the worst bias in the baseline model compared to the other kinds of simulated data we have tried.

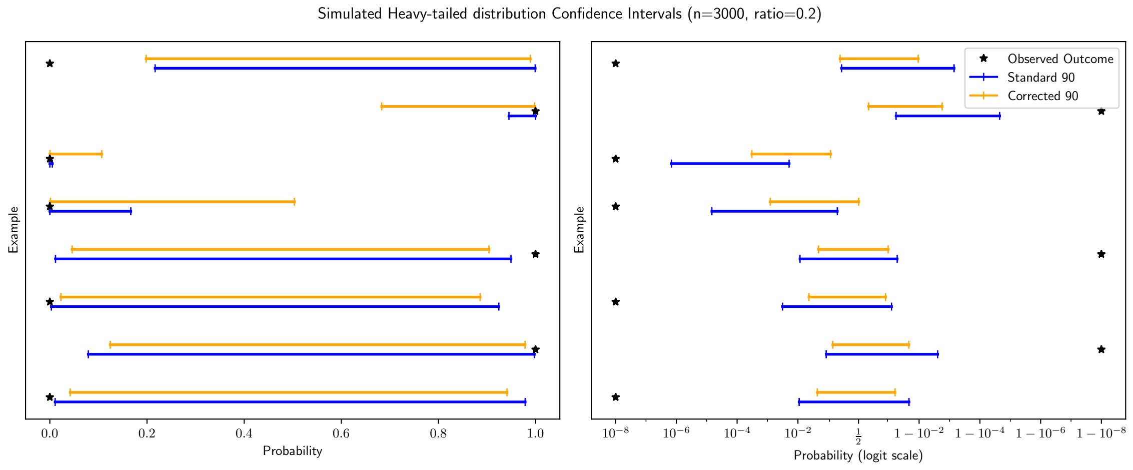

Figure 8: Prediction intervals for the baseline (blue) and SLOE model (orange) for 8 held-out heavy-tailed non-Gaussian test data points. The same plot is shown in linear scale (left), to emphasize interval width, and logit scale (right), to emphasize interval centers. In this example, $n = 3000$ and $p = 600$.

n	κ	Data Type	Max $\hat{\alpha}$
500	0.1	latent_non_gaussian	2.23
500	0.2	latent_gaussian	3.81
1000	0.1	heavy	1.25
1000	0.2	heavy	2.07
3000	0.1	heavy	1.25
3000	0.2	heavy	1.91

Table 1: The highest $\hat{\alpha}$ for each set of simulation parameters, along with the type of simulated data associated with that value. Values have been rounded to 2 decimal places.

5.3 F1-score

For most of the experiments, the baseline model and the SLOE model have identical F1-scores when evaluated using the testing data. Table 2 shows the experiments where there is a difference in F1-score. We observe that specifically when the data size is large ($n = 3000$) and when latent variables are involved, the SLOE model has a marginally better F1-score.

5.4 Execution Time

Finally, for execution time, our results are shown in Table 3. Note that only experiments where SLOE takes less time to fit are shown. We see that SLOE is faster for large data sizes, and when the data comes from a more complicated (non-Gaussian) distribution.

n	κ	Data Type	Baseline F1-score	SLOE F1-score
1000	0.1	non_gaussian	0.578	0.5714
3000	0.1	latent_gaussian	0.5697	0.5727
3000	0.1	latent_non_gaussian	0.5413	0.5405
3000	0.2	latent_gaussian	0.4832	0.4824
3000	0.2	latent_non_gaussian	0.5163	0.4984

Table 2: F1-Scores on testing data for baseline and SLOE models. Values have been rounded to 4 decimal places. Only data where the F1-scores are different are shown.

n	κ	Data Type	Baseline Time to Fit (s)	SLOE Time to Fit (s)
3000	0.2	non_gaussian	4.4205	4.1317
3000	0.2	latent_gaussian	4.794	3.9953
3000	0.2	latent_non_gaussian	3.645	3.436
3000	0.2	heavy	4.621	3.9952

Table 3: Time taken to fit the baseline and SLOE models. Values have been rounded to 4 decimal places. Only data where SLOE is faster are shown.

6. Conclusion

Our experiment results indicate that the uncertainty in the SLOE model is most accurate for Gaussian data with or without latent variables and data from a heavy-tailed distribution. We find that non-Gaussian planar flow data, whether or not it contains latent relationships, often either causes poor model uncertainty quantification or violates the assumptions of the SLOE model. Caution should be used when interpreting the results of the SLOE model for these kind of data.

We also find that for larger data sizes and data with latent relationships, the SLOE model yields a better F1-score and takes less time to fit. However, the scores are the same for most other experiments, and the baseline model takes less time to fit than the SLOE model most of the time. These results indicate that the real benefit of SLOE is not necessarily the prediction itself or the speed of the model, but rather the uncertainty associated with that prediction (i.e., p-values, prediction intervals).

In future work, researchers can build upon these experiments by considering more ways to test how the SLOE model performs in practice. In this paper, we have used non-Gaussian data generated using a planar flow transformation, and sampled from a heavy-tailed distribution. However, "non-Gaussian" is a very broad class of distributions, and as we have seen, some non-Gaussian distributions (heavy-tailed) work better with the SLOE model. Future research could explore how SLOE performs for different types of non-Gaussian distributions.

One limitation of our study is that the execution time results are not entirely reliable. The performance of the Python statsmodels Logit model has been tuned and optimized over a much longer period of time than that of the SLOE code provided by (Yadlowsky et al., 2021b). Thus, it may not be fair to compare the execution time of the two. Future research could perform code optimizations for the SLOE module in order to make the comparison between the two models more meaningful.

Finally, during the course of our research, we tried to test the model with a real Heart Disease dataset from the UCI Machine Learning Repository (Janosi et al., 1988). However, we found that the SLOE model often had convergence issues or other optimization issues, so results were very limited. It would be interesting to see in a future study if these issues could be resolved, and if so, how the model performs on real data from different domains (e.g., healthcare/genomics, financial, socioeconomic, etc.).

Acknowledgments

We would like to acknowledge Dr. Rajesh Ranganath and Mark Goldstein for their support and feedback during this project. We are also grateful to Helene Sajer for helping us with gathering data and relevant papers during the initial stages of the project.

References

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

A. Janosi, W. Steinbrunn, M. Pfisterer, and R. Detrano. UCI machine learning repository, 1988. URL `https://archive.ics.uci.edu/ml/datasets/heart+disease`.

Z. Kong and K. Chaudhuri. The expressive power of a class of normalizing flow models. 2020. URL `https://arxiv.org/pdf/2006.00392.pdf`. Arxiv preprint.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

P. Sur and E. Candès. A modern maximum-likelihood theory for high-dimensional logistic regression. *Proceedings of the National Academy of Sciences (PNAS)*, 116(29): 14516–14525, 2019.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

S. Yadlowsky, T. Yun, C. McLean, and A. D'Amour. Sloe: A faster method for statistical inference in high-dimensional logistic regression. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021a.

S. Yadlowsky, T. Yun, C. McLean, and A. D'Amour. sloe-logistic. `https://github.com/google-research/sloe-logistic`, 2021b.

Q. Zhao, P. Sur, and E. Candès. The asymptotic distribution of the mle in high- dimensional logistic models: Arbitrary covariance. *Bernoulli*, 28(3):1835–1861, 2022.