`PYENSMALLEN`: HIGH-PERFORMANCE OPTIMIZATION FOR STATISTICAL COMPUTING IN PYTHON

APOORVA LAL

1. Summary

Modern statistical applications increasingly involve large datasets with millions of observations, making computational efficiency a critical concern. Many popular Python libraries for statistical modeling (such as SciPy (Virtanen et al. 2020) and statsmodels (Seabold and Perktold 2010)) were not designed with these scales in mind, resulting in excessive computation times for large problems. This also limits users' ability to perform uncertainty quantification via the nonparametric bootstrap. This creates a significant barrier for researchers working with big data, often forcing compromises in model complexity or dataset size.

`pyensmallen` seeks to solve this problem by providing Python bindings to the highly optimized header-only ensmallen (Bhardwaj et al. 2018) C++ library, which leverages high-performance linear algebra through Armadillo (Sanderson and Curtin 2016). This enables access to ensmallen's state-of-the-art optimization algorithms, with a focus on methods commonly used in statistical estimation:

- L-BFGS for smooth objective optimization in maximum likelihood estimation
- ADAM (and variants) for neural network-style optimization SGD (optionally with momentum)
- Frank-Wolfe algorithms for constrained optimization with lp-ball or simplex constraints
- Generalized Method of Moments (GMM) estimation using ensmallen optimizers and JAX-powered automatic differentiation (Bradbury et al. 2018)

The library is designed for researchers and practitioners who need to train models on large datasets where existing solutions become prohibitively slow. Our implementation scales efficiently with both dataset size and dimensionality, enabling analyses that would otherwise be computationally infeasible. Our benchmarks demonstrate that `pyensmallen` consistently outperforms both SciPy and statsmodels across a range of regression models and dataset sizes, with the performance advantage becoming more pronounced as data size increases:

- For linear regression with 10 million observations, `pyensmallen` is 5-11x faster than SciPy and 3-4x faster than statsmodels
- For logistic regression with high-dimensional data, `pyensmallen` achieves 11-15x speedup over SciPy and 2-4.5x faster than statsmodels
- For Poisson regression with large datasets, `pyensmallen` is up to 13x faster than SciPy and 30x faster than statsmodels

Date: 31 March 2025.

Importantly, this speed advantage does not come at the cost of accuracy - all libraries achieve essentially identical parameter estimates since the loss functions are all convex, confirming that `pyensmallen` delivers the same statistical results much more efficiently.

The performance benefits enable several practical advantages:

- **Practical nonparametric-bootstrap**: The speed improvements make bootstrap resampling for inference viable even with large datasets. This allows users to construct confidence intervals around most statistical functionals
- **Model Selection Benefits**: Researchers can iterate through more model specifications and hyperparameter choices in the same time budget
- **Reliable convergence**: Unlike some competitors that occasionally fail to converge on challenging problems (particularly with Poisson regression), `pyensmallen` shows robust convergence across all test cases

2. Benchmarks

2.1. **runtime.** All benchmarks were conducted using synthetic datasets with controlled properties to ensure fair comparison. We tested each library on identical data across various sizes (from 1,000 to 10,000,000 observations) and dimensionalities (k=5 and k=20) over five iterations and measure mean runtime and RMSE. The complete benchmark methodology and code are available in the repository's `paper` directory, allowing for full reproducibility of our results.

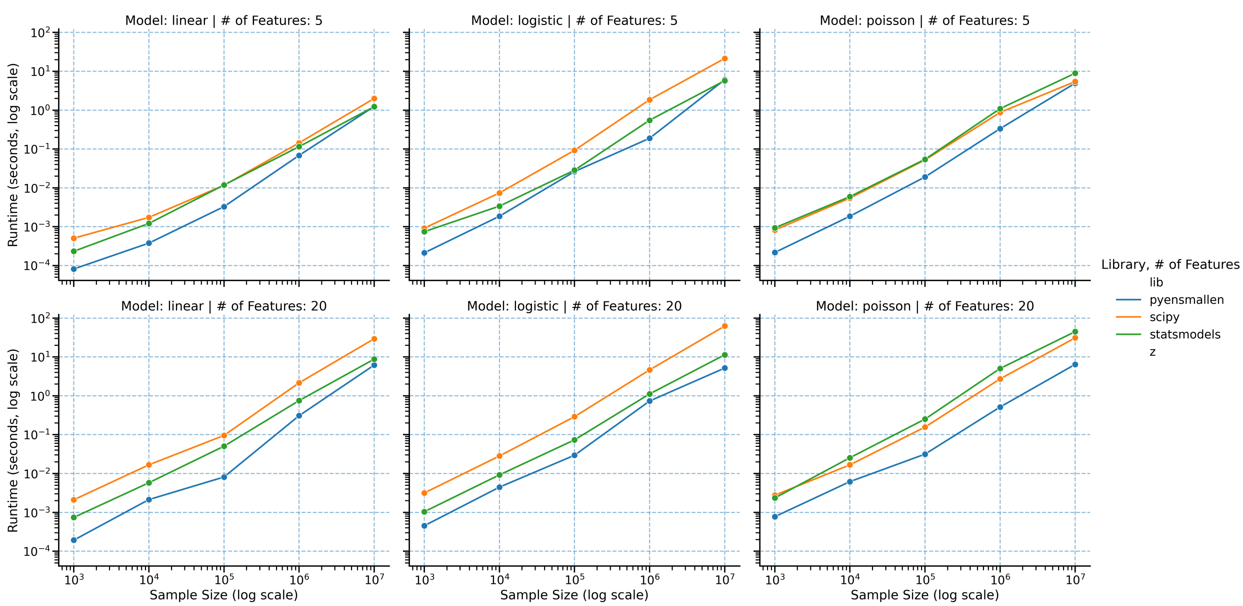

Figure 1. Library Performance Comparison across Regression Models

Figure 1: Performance comparison of different libraries across linear, logistic, and Poisson regression models. `pyensmallen` consistently delivers superior performance, especially as dataset size increases. This is especially surprising for linear regression, where statsmodels uses the closed form solution $\hat{\beta} = (X'X)^{-1}X'y$, while `pyensmallen` and scipy minimize

square loss using L-BFGS. These figures show execution times for different models as a function of dataset size. The slope of each line indicates how efficiently each library scales. The consistently lower position of the `pyensmallen` line demonstrates its performance advantage, which grows with larger datasets.

References

Bhardwaj, Shikhar, Ryan R Curtin, Marcus Edel, Yannis Mentekidis, and Conrad Sanderson. 2018. "Ensmallen: A Flexible c++ Library for Efficient Function Optimization." *Workshop on Systems for ML and Open Source Software at NeurIPS*. https://ensmallen.org/.

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. 2018. "JAX: Composable Transformations of Python+NumPy Programs." http://github.com/google/jax.

Sanderson, Conrad, and Ryan Curtin. 2016. "The Design and Implementation of the Armadillo c++ Linear Algebra Library." In *Mathematical Software–ICMS 2016: 5th International Conference, Berlin, Germany, July 11-14, 2016, Proceedings*, 57–67. Springer.

Seabold, Skipper, and Josef Perktold. 2010. "Statsmodels: Econometric and Statistical Modeling with Python."

Virtanen, Pauli, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17 (3): 261–72.