Pyglmnet: Python implementation of elastic-net regularized generalized linear models

Mainak Jas[1, 2], Titipat Achakulvisut[3], Aid Idrizović[4], Daniel Acuna[5], Matthew Antalek[6], Vinicius Marques[4], Tommy Odland[7], Ravi Prakash Garg[6], Mayank Agrawal[8], Yu Umegaki[9], Peter Foley[10], Hugo Fernandes[11], Drew Harris[12], Beibin Li[13], Olivier Pieters[14, 20], Scott Otterson[15], Giovanni De Toni[16], Chris Rodgers[17], Eva Dyer[18], Matti Hamalainen[1, 2], Konrad Kording[3], and Pavan Ramkumar[19]

1 Massachusetts General Hospital **2** Harvard Medical School **3** University of Pennsylvania **4** Loyola University **5** University of Syracuse **6** Northwestern University **7** Sonat Consulting **8** Princeton University **9** NTT DATA Mathematical Systems Inc **10** 605 **11** Rockets of Awesome **12** Epoch Capital **13** University of Washington **14** IDLab-AIRO – Ghent University – imec **15** Clean Power Research **16** University of Trento **17** Columbia University **18** Georgia Tech **19** System1 Biosciences Inc **20** Research Institute for Agriculture, Fisheries and Food

DOI: 10.21105/joss.01959

Software

- Review
- Repository
- Archive

Editor: Ariel Rokem

Reviewers:

- @professoralkmin
- @ryEllison

Submitted: 26 November 2019
Published: 16 December 2019

Summary

Generalized linear models (GLMs) are well-established tools for regression and classification and are widely applied across the sciences, economics, business, and finance. Owing to their convex loss, they are easy and efficient to fit. Moreover, they are relatively easy to interpret because of their well-defined noise distributions and point-wise nonlinearities.

Mathematically, a GLM is estimated as follows:

$$\min_{\beta_0, \beta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \beta_0 + \beta^T x_i) + \lambda \mathcal{P}(\beta)$$

where $\mathcal{L}(y_i, \beta_0 + \beta^T x_i)$ is the negative log-likelihood of an observation (x_i, y_i), and $\lambda \mathcal{P}(\cdot)$ is the penalty that regularizes the solution, with λ being a constant that controls the amount of regularization.

Modern datasets can contain a number of predictor variables, and data analysis is often exploratory. Under these conditions it is critically important to regularize the model to avoid overfitting the data. Regularization works by adding penalty terms that penalize the model parameters in a variety of ways. This can be used to incorporate prior knowledge about the parameters in a structured form. In pyglmnet, we offer users the ability to combine different types of regularization with different noise distributions in the GLMs.

Despite the attractiveness of regularized GLMs, the available tools in the Python data science eco-system are highly fragmented. Specifically:

- statsmodels provides a wide range of link functions but no regularization.
- scikit-learn provides elastic net regularization but only limited noise distribution options.
- lightning provides elastic net and group lasso regularization, but only for linear and logistic regression.

Pyglmnet is a response to this fragmentation. The table below compares pyglmnet with existing libraries as of this writing.

	pyglmnet	scikit-learn	statsmodels	lightning	py-glm	Matlab	glmnet in R
Distributions							
Gaussian	x	x	x	x	x	x	x
Binomial	x	x	x	x	x	x	x
Poisson	x		x		x	x	x
Poisson (softplus)	x						
Probit	x						
Gamma	x		x			x	
Regularization							
L2	x	x		x			
Lasso	x	x		x			x
Group lasso	x			x			x
Tikhonov	x						

Pyglmnet implements the algorithm described in Friedman, J., Hastie, T., & Tibshirani, R. (2010) and the accompanying popular R package glmnet. As opposed to python-glmnet or glmnet_python, which are wrappers around this package, pyglmnet is written in pure Python and runs on Python 3.5+. The implementation is compatible with the existing data science ecosystem. Pyglmnet's API is designed to be compatible with scikit-learn (Buitinck et al., 2013). Thus, it is possible to do:

```
glm.fit(X, y)
glm.predict(X)
```

As a result of this compatibility, `scikit-learn` tools for building pipelines, cross-validation and grid search can be employed by pyglmnet users. Pyglmnet has already been used in published work (Benjamin et al., 2017; Bertrán et al., 2018; Höfling, Berens, & Zeck, 2019; Rybakken, Baas, & Dunn, 2019). It contains unit tests and includes documentation in the form of tutorials, docstrings and examples that are run through continuous integration.

Acknowledgements

`Pyglmnet` development is partly supported by NIH NINDS R01-NS104585 and the Special Research Fund (B.O.F.) of Ghent University.

References

Benjamin, A. S., Fernandes, H. L., Tomlinson, T., Ramkumar, P., VerSteeg, C., Chowdhury, R., Miller, L., et al. (2017). Modern machine learning outperforms GLMs at predicting spikes. *bioRxiv*, 111450. doi:10.1101/111450

Bertrán, M. A., Martínez, N. L., Wang, Y., Dunson, D., Sapiro, G., & Ringach, D. (2018). Active learning of cortical connectivity from two-photon imaging data. *PloS one*, *13*(5), e0196527. doi:10.1371/journal.pone.0196527

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., et al. (2013). API design for machine learning software: Experiences from the scikit-learn project. In *ECML pkdd workshop: Languages for data mining and machine learning* (pp. 108–122).

Höfling, L., Berens, P., & Zeck, G. (2019). Probing and predicting ganglion cell responses to smooth electrical stimulation in healthy and blind mouse retina. *bioRxiv*, 609826. doi:10.1101/609826

Rybakken, E., Baas, N., & Dunn, B. (2019). Decoding of neural data using cohomological feature extraction. *Neural computation*, *31*(1), 68–93. doi:10.1162/neco_a_01150