The Journal of Open Source Software

GENRE (GPU Elastic-Net REgression): A CUDA-Accelerated Package for Massively Parallel Linear Regression with Elastic-Net Regularization

Christopher Khan[1] **and Brett Byram**[1]

1 Vanderbilt University

DOI: 10.21105/joss.02644

Software

- Review
- Repository
- Archive

Editor: Stefan Pfenninger

Reviewers:

- @marouenbg
- @krystophny

Submitted: 10 July 2020
Published: 09 October 2020

Summary

GENRE (GPU Elastic-Net REgression) is a package that allows for many instances of linear regression with elastic-net regularization to be processed in parallel on a GPU by using the C programming language and NVIDIA's (NVIDIA Corporation, Santa Clara, CA, USA) Compute Unified Device Architecture (CUDA) parallel programming framework. Linear regression with elastic-net regularization (Zou & Hastie, 2005) is a widely utilized tool when performing model-based analyses. The basis of this method is that it allows for a combination of L1-regularization and L2-regularization to be applied to a given regression problem. Therefore, feature selection and coefficient shrinkage are performed while still allowing for the presence of groups of correlated features. The process of performing these model fits can be computationally expensive, and one of the fastest packages that is currently available is glmnet (Friedman, Hastie, & Tibshirani, 2010; Hastie & Qian, 2014; Qian, Hastie, Tibshirani, & Simon, 2013). This package provides highly efficient Fortran implementations of several different types of regression. In the case of its implementation of linear regression with elastic-net regularization, the objective function shown in (eq. 1) is minimized.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2N} \sum_{i=1}^{N} \left(\boldsymbol{y}_i - \sum_{j=1}^{P} \boldsymbol{X}_{ij} \boldsymbol{\beta}_j \right)^2 + \lambda \left(\alpha \left\| \boldsymbol{\beta} \right\|_1 + \frac{(1-\alpha) \left\| \boldsymbol{\beta} \right\|_2^2}{2} \right) \tag{1}$$

To minimize this objective function, cyclic coordinate descent is utilized as the optimization algorithm. This algorithm consists of minimizing the objective function with respect to one model coefficient at a time. Cycling through all of the coefficients results in one iteration, and this process continues until specified convergence criteria are satisfied. As previously stated, glmnet is highly efficient for single model fits, but performing thousands of these fits will still require significant computational time due to each one being executed in a serial fashion on a CPU. However, by using GENRE to perform massively parallel processing on a GPU, a significant speedup can potentially be achieved. This is due to the fact that modern GPUs consist of thousands of computational cores that can be utilized. Moreover, although the processing in GENRE is performed using the C programming language and CUDA, a MEX-interface is included to allow for this code to be called within the MATLAB (The MathWorks, Inc., Natick, MA, USA) programming language for convenience. This also means that with modification, the MEX-interface can be replaced with another interface if it is desired to call the C/CUDA code in another language, or the C/CUDA code can be utilized without an interface. Note that other packages have been developed that can utilize GPUs for linear regression with elastic-net regularization, such as H2O4GPU ("H2O4GPU," 2020). However, for this application, these packages typically focus on performing parallel computations on the GPU for one model fit at a time in order to achieve acceleration when compared to a serial

CPU implementation. For GENRE, the computations for a single model fit are not parallelized on the GPU. Instead, many model fits on the GPU are executed in parallel, where each model fit is performed by one computational thread.

Statement of Need

The core motivation for developing GENRE was that many of the available packages for performing linear regression with elastic-net regularization focus on achieving high performance in terms of computational time or resource consumption for single model fits. However, they often do not address the case in which there is a need to perform many model fits in parallel. For example, the research project that laid the foundation for GENRE involved performing ultrasound image reconstruction using an algorithm called Aperture Domain Model Image REconstruction (ADMIRE) (Byram, Dei, Tierney, & Dumont, 2015; Byram & Jakovljevic, 2014; Dei & Byram, 2017). This algorithm is computationally expensive due to the fact that in one stage, it requires thousands of instances of linear regression with elastic-net regularization to be performed in order to fit models of ultrasound data. When this algorithm was implemented on a CPU, it typically required an amount of time that was on the scale of minutes to reconstruct one ultrasound image. The primary bottleneck was performing all of the required model fits due to the fact that a custom C implementation of cyclic coordinate descent was used to compute each fit serially. However, a GPU implementation of the algorithm was developed, and this implementation provided a speedup of over two orders of magnitude, which allowed for multiple ultrasound images to be reconstructed per second. For example, on a computer containing dual Intel (Intel Corporation, Santa Clara, CA) Xeon Silver 4114 CPUs @ 2.20 GHz with 10 cores each along with an NVIDIA GeForce GTX 1080 Ti GPU and an NVIDIA GeForce RTX 2080 Ti GPU, the CPU implementation of ADMIRE had an average processing time of 94.326 ± 0.437 seconds for one frame of ultrasound channel data while the GPU implementation had an average processing time of 0.436 ± 0.001 seconds. The average processing time was obtained for each case by taking the average of 10 runs for the same dataset, and timing was performed using MATLAB's built-in timing capabilities. The 2080 Ti GPU was used to perform GPU processing, and the number of processing threads was set to 1 for the CPU implementation. The main contributor to this speedup was the fact that the model fits were performed in parallel on the GPU. For this particular case, 152,832 model fits were performed. Note that double precision was used for the CPU implementation while single precision was utilized for the GPU implementation due to the fact there is typically a performance penalty when using double precision on a GPU. Moreover, for the CPU implementation, MATLAB was used, and a MEX-file was used to call the C implementation of cyclic coordinate descent for the model fitting stage. In addition, note that one additional optimization when performing the model fits on the GPU in the case of ADMIRE is that groups of model fits can use the same model matrix, which allows for improved coalesced memory access and GPU memory bandwidth use. This particular optimization is not used by GENRE.

Aside from this application, there are a number of other applications that can potentially benefit from having the ability to perform model fits in a massively parallel fashion, which is why the code was developed into a package. For example, linear regression with elastic-net regularization has been applied to the field of genomics in order to develop predictive models that utilize genetic markers (Ogutu, Schulz-Streeck, & Piepho, 2012; Waldmann, Mészáros, Gredler, Fuerst, & Sölkner, 2013). In addition, like ADMIRE, there are a variety of other signal processing applications. For example, this regression method has been used to create models of functional magnetic resonance imaging data in order to predict the mental states of subjects and provide insight into neural activity (Carroll, Cecchi, Rish, Garg, & Rao, 2009). Moreover, another signal processing example is that linear regression models with elastic-net regularization have been used in combination with hidden Markov random field segmentation to perform computed tomography estimation for the purposes of magnetic resonance imaging-based attenuation correction for positron emission tomography/magnetic resonance imaging

(Chen et al., 2014). Now, through the use of GENRE, it is possible to reduce the amount of processing time that is required in each of the aforementioned examples by computing the models in parallel for each case.

Example Benchmark Comparing GENRE with glmnet

GENRE has the potential to provide significant speedup due to the fact that many model fits can be performed in parallel on a GPU. Therefore, an example benchmark was performed where we compared GENRE with glmnet, which is written in Fortran and performs the model fits in a serial fashion on a CPU. In this benchmark, 20,000 model matrices were randomly generated within MATLAB. Each model matrix consisted of 50 observations and 200 predictors (50x200), and an intercept term was included for all of the models. Note that to add an intercept term in GENRE, a column of ones was appended at the beginning of each model matrix to make the predictor dimension 201 (adding a column of ones is not required for glmnet). For each model matrix, the model coefficients were randomly generated, and the matrix multiplication of the model matrix and the coefficients was performed to obtain the observation vector. Therefore, this provided 20,000 observation vectors with each containing 50 observations. Once the data was generated, both GENRE and glmnet were used to perform the model fits and return the computed model coefficients. An α value of 0.5 and a λ value of 0.001 were used for all of the model fits. The tolerance convergence criterion for both packages was set to 1E-4. It was also specified for each package to standardize the model matrices, which means that the unstandardized model coefficients were returned. Note that the column of ones for each model matrix corresponding to the intercept term is not standardized in the case of GENRE. GENRE allows for the user to select either single precision or double precision for performing the model fits on the GPU, so processing was done for both cases. The MATLAB version of the glmnet software package includes a compiled executable MEX-file that allows for Fortran code to be called, and it uses double precision for the calculations. In addition, due to the fact that all of the model matrices have a small number of observations (50) in this case, GENRE is also able to use shared memory in addition to global memory when performing the model fits. Shared memory has lower latency than global memory, so utilizing it can provide performance benefits. Therefore, processing was performed both with and without using shared memory.

The computer that was used for the benchmarks contained dual Intel Xeon Silver 4114 CPUs @ 2.20 GHz with 10 cores each along with an NVIDIA GeForce GTX 1080 Ti GPU and an NVIDIA GeForce RTX 2080 Ti GPU. The 2080 Ti GPU was used to perform GPU processing. For each case, the average of 10 runs was taken, and timing was performed using MATLAB's built-in timing capabilities. Note that GENRE has a data organization step that loads the data for the model fits from files and organizes it into the format that is used by the GPU. For this benchmark, this step was not counted in the timing due to the fact that it was assumed that all of the data was already loaded into MATLAB on the host system for both GENRE and glmnet. The GPU times include the time it takes to transfer data for the model fits from the host system to the GPU, standardize the model matrices, perform the model fits, unstandardize the model coefficients, transfer the computed model coefficients back from the GPU to the host system, and store the coefficients into a MATLAB cell structure. The CPU time includes the time it takes to standardize the model matrices, perform the model fits, unstandardize the model coefficients, and store the coefficients into a MATLAB cell structure. The benchmark results are shown in Table 1 below. Note that DP, SP, and SMEM correspond to double precision, single precision, and shared memory respectively. In addition, note that the input data for the model fits was of type `double` for this benchmark. Therefore, in the case of GENRE, some of the inputs would need to be converted to type `single` before they are passed to the GPU when using single precision for the computations. Moreover, GENRE also converts the datatype of the computed model coefficients to the datatype of the original input data. This means that for the single precision cases, the computed model coefficients would need to be converted to be type `double` after they are passed back to the host system

from the GPU. For purposes of benchmarking the single precision cases, the time to perform the type conversions of the inputs to type `single` was not included, and the returned model coefficients were just kept as type `single`. This is due to the fact that including these times would increase the benchmark times for the single precision cases in this scenario, and if it were a different scenario, the double precision cases could be impacted instead of the single precision cases. For example, if the type of the original input data was `single` and double precision was used for the calculations, then these datatype conversions would have to be made for the double precision cases, but they would not have to be made for the single precision cases.

Table 1: Benchmark times (seconds).

GENRE DP	GENRE DP SMEM	GENRE SP	GENRE SP SMEM	glmnet
1.339 $\pm$ 0.036	1.090 $\pm$ 0.028	1.035 $\pm$ 0.046	0.885 $\pm$ 0.029	9.809 $\pm$ 0.196

As shown in Table 1, GENRE provides an order of magnitude speedup when compared to glmnet, and the best performance was achieved by using single precision with shared memory. For glmnet, the benchmark result that is shown was obtained by using the naive algorithm option for the package because this option was faster than the covariance algorithm option. For example, the benchmark result that was obtained when using the covariance algorithm option was 32.271 $\pm$ 0.176 seconds. In addition, it is important to note that in these benchmarks, most of the time for GENRE was spent transferring the model matrices from the host system to the GPU. However, there are cases when once the model matrices have been used in one call, they can be reused in subsequent calls. For example, a user might want to reuse the same model matrices except just change the α value or the λ value that is used in elastic-net regularization, or they might want to just change the observation vectors that the model matrices are fit to. By default, each time GENRE is called, the *clear mex* command is executed, and the GENRE MEX-files are setup so that all allocated memory on the GPU is freed when this command is called. However, in a case where the model matrices can be reused after they are transferred once, the *clear mex* command can be removed. Essentially, every time one of the MEX-files for GENRE is called for the first time, all of the data for the model fits will be transferred to the GPU. However, if the *clear mex* command is removed, then for subsequent calls, all of the data for the model fits will still be transferred except for the model matrices, which will be kept on the GPU from the first call. By not having to transfer the model matrices again, performance can be significantly increased. To demonstrate this, the same benchmark from above was repeated, but for each case this time, GENRE was called once before performing the 10 runs. This is to replicate the case where the model matrices are reused in subsequent calls. The benchmark results are shown in Table 2 below. Note that the benchmark for glmnet was not repeated.

Table 2: Benchmark times (seconds) for subsequent calls reusing the model matrices.

GENRE DP	GENRE DP SMEM	GENRE SP	GENRE SP SMEM	glmnet
0.305 $\pm$ 0.000	0.080 $\pm$ 0.000	0.195 $\pm$ 0.001	0.055 $\pm$ 0.001	9.809 $\pm$ 0.196

As shown in Table 2, when the model matrices can be reused and do not have to be transferred again, GENRE provides a speedup of over two orders of magnitude when compared with glmnet, and using single precision with shared memory provides the best performance. This type of performance gain would most likely be difficult to achieve even when using a multi-CPU implementation of cyclic coordinate descent on a single host system. In addition, it is important to note that this benchmark was just to illustrate an example of when using GENRE provides performance benefits, but whether or not performance benefits are achieved depends on the problem. For example, in GENRE, one computational thread on the GPU

is used to perform each model fit. Therefore, when many model fits are performed on a GPU, the parallelism of the GPU can be utilized. However, if only one model fit needs to be performed, then using a serial CPU implementation such as glmnet will most likely provide better performance than GENRE due to factors such as CPU cores having higher clock rates and more resources per core than GPU cores.

Acknowledgements

This work was supported by NIH grants R01EB020040 and S10OD016216-01 and NAVSEA grant N0002419C4302.

References

Byram, B., Dei, K., Tierney, J., & Dumont, D. (2015). A model and regularization scheme for ultrasonic beamforming clutter reduction. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *62*(11), 1913–1927. doi:10.1109/tuffc.2015.007004

Byram, B., & Jakovljevic, M. (2014). Ultrasonic multipath and beamforming clutter reduction: A chirp model approach. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *61*(3), 428–440. doi:10.1109/tuffc.2014.2928

Carroll, M., Cecchi, G., Rish, I., Garg, R., & Rao, A. (2009). Prediction and interpretation of distributed neural activity with sparse models. *NeuroImage*, *44*(1), 112–122. doi:10.1016/j.neuroimage.2008.08.020

Chen, Y., Juttukonda, M., Lee, Y., Su, Y., Espinoza, F., Lin, W., Shen, D., et al. (2014). MRI based attenuation correction for pet/mri via mrf segmentation and sparse regression estimated ct. *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*. doi:10.1109/isbi.2014.6868131

Dei, K., & Byram, B. (2017). The impact of model-based clutter suppression on cluttered, aberrated wavefronts. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, *64*(10), 1450–1464. doi:10.1109/tuffc.2017.2729944

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1). doi:10.18637/jss.v033.i01

H2O4GPU. (2020). Retrieved from https://github.com/h2oai/h2o4gpu

Hastie, T., & Qian, J. (2014). Glmnet vignette. Retrieved from http://www.web.stanford.edu/~hastie/Papers/Glmnet_Vignette.pdf

Ogutu, J. O., Schulz-Streeck, T., & Piepho, H. P. (2012). Genomic selection using regularized linear regression models: Ridge regression, lasso, elastic net and their extensions. *BMC Proceedings*, *6*(S2). doi:10.1186/1753-6561-6-s2-s10

Qian, J., Hastie, T., Tibshirani, R., & Simon, N. (2013). Glmnet for matlab. Retrieved from http://www.stanford.edu/~hastie/glmnet_matlab/

Waldmann, P., Mészáros, G., Gredler, B., Fuerst, C., & Sölkner, J. (2013). Evaluation of the lasso and the elastic net in genome-wide association studies. *Frontiers in Genetics*, *4*. doi:10.3389/fgene.2013.00270

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x