Elastic Net by Iteration: Linear Gradient and Coordinate Descent, and Logistic Coordinate Descent

Authors: Madeline Kwicklis, Diana Liang, and Chenhao Shangguan

Introduction

Advances in molecular genetics have made it possible to easily assay thousands of molecular features per sample, so biological traits can now be modeled with many potential covariates, which in turn requires variable selection and other regularization processes, particularly in the case that the number of features exceeds the number of samples. Since the number of covariates that may be falsely significant at the same level increases with the number of overall covariates, the need for high-dimensional statistical methods has increased. Elastic net provides both variable selection to limit falsely significant covariates and comparable penalized estimates. The standard for R is the package 'glmnet' which uses coordinate descent to perform elastic net for a variety of distributional families; and it's well-maintained. We focused on recapitulating their results using different iterative methods and testing them on analyzing the relationship between gene expression profiles from liver RNA samples of obesity clinic patients and whether those patients had non-alcoholic fatty liver disease.

Methods

Elastic net combines ridge and LASSO regression by using both methods of penalizing against larger estimates in the loss function for the distribution of interest:

$$loss = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2 + \frac{1}{2}(1-\alpha)\lambda||\beta||_{l2} + \alpha\lambda||\beta||_{l1} \quad (1)$$

$$L2\ Regularization\ ||\beta||_{l2} = \sum_{j=1}^{p}\beta_j{}^2 \qquad L1\ Regularization\ ||\beta||_{l1} = \sum_{j=1}^{p}\left|\beta_j\right|$$

Ridge regression uses L2 regularization as a penalty to shrink estimates toward zero while LASSO regression uses L1 regularization as a penalty to select certain parameters by setting others to zero. The γ parameter, a nonnegative number, controls the degree of the combined penalization, and the α parameter, a number between 0 and 1 inclusive, controls the degree each type of penalty is used where $\alpha = 0$ is only L2 and $\alpha = 1$ is only L1. The L1 penalty of LASSO regression can select out one out of a group of highly correlated covariates but may select too many, so the L2 penalty of ridge regression can be used to dilute this effect to balance variable selection and meaningful covariate estimates. Due to this combination of penalties in the loss function, specifically the L1 penalty, the loss function is not smooth enough for a second derivative, so we focused on iterative optimization techniques.

Linear Elastic Net Algorithms

Linear Gradient Descent

Our first algorithm minimizes the linear loss function of elastic net by gradient descent, which iteratively updates estimates by a fractional step size of the gradient of loss minimization. Given the current estimates, the gradient for each estimate would be:

$$\delta\beta = -\frac{1}{n}[X \times (y - \hat{y})] + (1-\alpha)\lambda\beta \pm \lambda\alpha$$

Each iteration updates all the estimates at the same time so that each step moves in multiple dimensions towards and eventually converges on the loss minimum. The run time would still increase with the number of covariates, time complexity $O(np)$, but vectorization improves on the original loops. In addition to the matrix of covariates and responses as well as λ and α parameters, the linear gradient descent also requires a starting point, step size, tolerance, and maximum number of iterations. The last

three control the balance of run time and precision of final estimates. The combination of step size and tolerance defines how close the final estimates must be to the true minimum if the step size is small enough or if the algorithm will not converge if it's too large. A maximum number of iterations sets a standard run time so that the user can understand complications of step size and tolerance without wasting time on a run with unideal inputs. This specific gradient descent uses the ridge regression solution as a default starting point since, after comparing with the ridge regression guided gradient descent, it greatly reduces run time and allows for larger sizes to converge at the same tolerance. We've also included a cut off point of $\lambda\alpha$ adopted from the coordinate descent algorithms below to aid in variable selection.

Linear Coordinate Descent

Coordinate descent is an efficient algorithm that minimizes the loss function for one parameter at time, holding all others constant:

$$\beta_j^{(k)} = argmin_\theta \, Loss(\beta_0^{(k)}, \beta_1^{(k)}, ..., \beta_{j-1}^{(k)}, \theta, \beta_{j+1}^{(k-1)}, ..., \beta_p^{(k-1)})$$

Thus each cycle contains p+1 minimizations. In the case of linear elastic net, the parameter-wise minimization has a closed form solution when the sign of the parameter is known and nonzero: $\beta^{(k)}_j = X_j^T (Y - X_{(-j)}\tilde{\beta}_{(-j)})$, where $X_{(-j)}$ is the design matrix with the jth column removed and $\tilde{\beta}_{(-j)}$ is the current iteration of β without the jth element (see Appendix for derivation). Essentially, the solution is a function of the residual when the parameter being estimated is omitted from the model. Our implementation takes in all the inputs as gradient descent except for step size, though the design matrix variables must be scaled and centered. It is recommended that users not deviate from the default starting value of β. Because the derivative of the loss function does not exist at 0, both glmnet's algorithm and ours performs soft thresholding, where if the absolute value of the term shown in red falls below $\lambda\alpha$, the coefficient is set to zero. This is an efficient algorithm, with $O(np)$ complexity for each minimization for a naive analysis and thus $O(np^2)$ time for each cycle.

Ridge Regression Guided Gradient Descent

Ridge-regression guided gradient descent (RRGG) leverages the efficiency of a closed form solution to ridge regression with gradient descent. All the inputs for linear gradient descent apply with addition to an α step size. The mixture parameter α is incremented by the α step size. For each incrementation of α, gradient descent is performed using the solution corresponding to the previous value of α. If the gradient descent fails to converge, the size of the increment is halved to slowly converge on the specified value. To prevent complete stoppage of the algorithm, a minimum required step size for α can also be specified by the user. The complexity for solving the ridge regression is $O(np^3)$. Every iteration thereafter has the same complexity as gradient descent ($O(np)$). Theoretically, this algorithm can improve runtime over an unguided gradient descent when α is closer to 0 but seems to perform similarly to our gradient descent which also starts with ridge regression for $\alpha > 0$. Nonetheless, it is potentially a useful means of confirming that the gradient descent has reached a global, not merely local, minimum.

Logistic Coordinate Descent

Our logistic elastic net follows a similar algorithm to the linear coordinate descent algorithm but adjusted for binomial responses. This loss function includes a logit link that transforms the linear covariate function $X^T\beta$ into probabilities and is altered from a log-likelihood of a normally distributed

variable to the log-likelihood of a binomially distributed variable. With a change in the loss function and its derivatives, we utilized a quadratic approximation of the binomial log-likelihood function that includes a modified difference between a transformed response and the linear function of the covariates:

$$Approximated\ loss\ = \frac{1}{n}\sum_{i=1}^{n} w_i(z_i - x_i{}^T\beta)^2 + Constant$$

$$Weights\ w_i = logit(x_i{}^T\beta^{(k-1)})[1 - logit(x_i{}^T\beta^{(k-1)})]$$

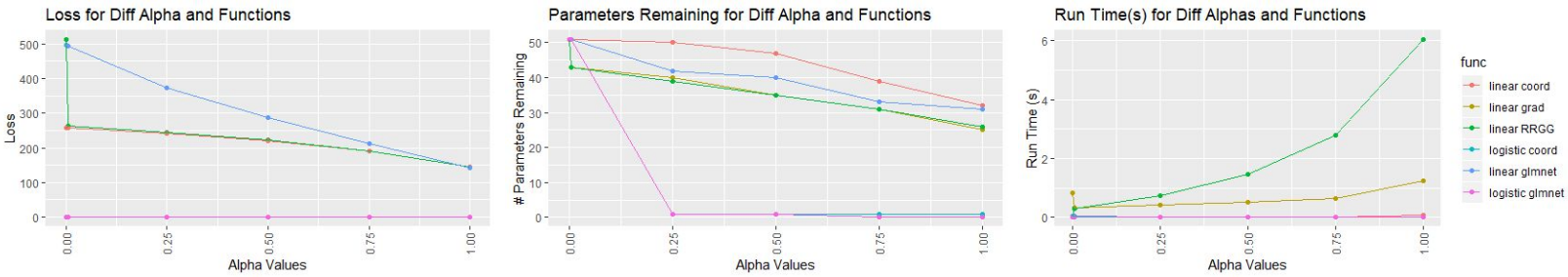
$$Pseudo-\ Response\ z_i = \frac{y_i - logit(x_i{}^T\beta^{(k-1)})}{logit(x_i{}^T\beta^{(k-1)})[1-logit(x_i{}^T\beta^{(k-1)})]} + x_i{}^T\beta^{(k-1)}$$

The approximation is derived from a second-order Taylor expansion of the binomial log-likelihood $l(\beta|y) = \sum_{i=1}^{n} y_i\, log(x_i\beta) - (1-y_i)log(1+exp(x_i\beta)) + C$ [1]. In this case, the expansion occurs around the previous coordinate descent cycle's estimate of β, $\beta^{(k-1)}$. [1] This approximation with the appropriate weights fits neatly into the coordinate descent format of the linear case where the logistic loss is minimized one parameter at a time and soft thresholding pushes any estimates below $\lambda\alpha$ to zero. Inputs of the logistic coordinate descent inputs are comparable to that of the linear, and the time complexity for this efficient method is again $O(np)$ for each parameter and $O(np^2)$ for each cycle.

Cross Validation (CV) λ Selection

Every method so far requires an input of both λ and α parameters, so we included a cross validation function that outputted the λ with the lowest MSE or loss given α. Using the inputs expected of the particular function and the number of folds to divide the data, we used CV to ensure that the chosen λ is less likely to be affected by overfitting. The output of λ with the lowest MSE or loss, then, reduces the number of variables to consider in the actual analysis.

Algorithm Analysis



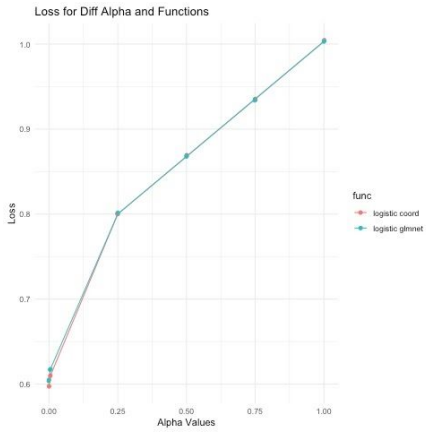


Figure 1: Loss, # Parameters Remaining, and Run Time for Different α and Functions of Elastic Net ($\lambda = 1$)

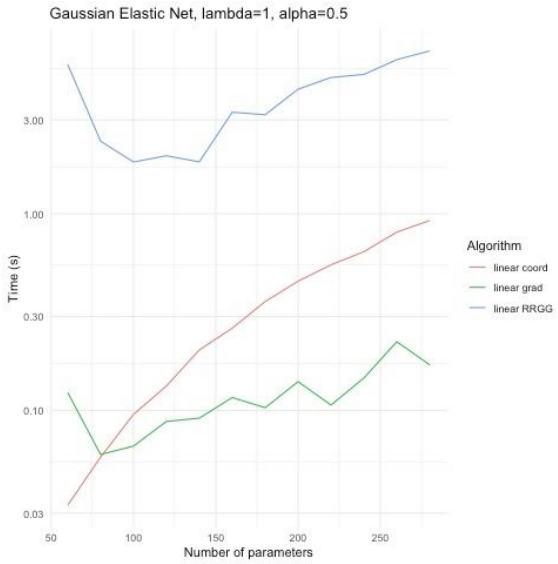


Figure 2: Run Time (s) for Linear Algorithms for Different Number of Parameters ($\lambda = 1,\ \alpha = 0.5$)

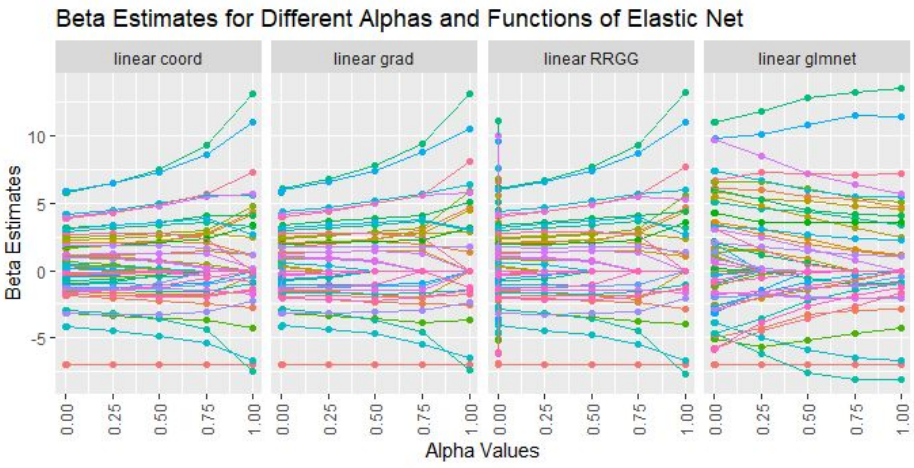


Figure 3: $\hat{\beta}$ Behavior for Different α and Functions of Elastic Net ($\lambda = 1$)

We assessed the accuracy and speed of our functions by simulating test cases for the Gaussian model. We generated a 50-sample design matrix of 50 random normal variables and a linear combination of those vectors for the response with normally distributed coefficients. We then recorded time and loss across different values of α for each of the functions. A similar analysis was done for the number of parameters by padding with non-predictive covariates. Simulations were repeated for the logistic case; responses were randomly generated as Bernoulli-distributed with probabilities calculated as the logit transformation of the linear combination of predictors.

Time-wise, linear gradient descent and linear RRGG descent competed as we edited the former function. With vectorization, the two were running at similar times with a faster RRGG at smaller α and vice versa. We included the ridge regression solution as starting point for regular gradient descent, and it overshadowed RRGG in efficiency. Gradient descent was running faster than RRGG except for when $\alpha = 0$ (**Figure 1**). Linear coordinate descent, though, was faster than both of the gradient descent algorithms at any point in the edits. For $\lambda = 1$ and $\alpha = 0.5$, we included extra unassociated parameters to test our algorithms over increasing numbers of parameters. Linear coordinate was faster than our other linear algorithms for the first 60 parameters, but gradient descent became faster for greater numbers of parameters (**Figure 2**). Since R performs more efficiently with vectorization than loops, our vectorized gradient descent was likely faster due to its vectorization than the coordinate descent, which largely relied on loops due to its iterative nature. Gradient descent also tends to penalize harsher with L1 regularization to select less parameters.

We can see that gradient descent and RRGG have a nearly identical pattern of variable shrinking and selection, as expected, and selected fewer covariates than the other two linear algorithms (**Figure 3**). The estimate patterns of our linear functions were in agreement with each other but not with glmnet. Though glmnet documentation[1] suggests that the loss function for the Gaussian case is also (1), the values of loss corresponding to glmnet are greater than those of our linear functions (**Figure 1**). The variable shrinking and selection patterns of the logistic coordinate descent and logistic glmnet are quite similar (see Appendix for logistic coordinate runtimes), and both calculate the intercept separately from the other parameters' calculations.

Genetics Data Example

We applied our implementation of logistic elastic net to a human microarray dataset from a longitudinal study of non-alcoholic steatohepatitis (NASH), an inflammatory condition of the liver that results from non-alcoholic fatty liver disease (NAFLD)[2]. In this longitudinal study, patients of an obesity clinic received liver biopsies, and microarray data was generated from the collected samples. After 1 year of follow-up, consenting patients were re-biopsied to determine if they had developed NASH (n=79; 60 with microarray data). Using the microarray data at baseline, we built a predictive model of whether the patients developed NASH using logistic elastic net on the microarray data. Part of the dataset will be used for training the model and the rest for prediction.

Expression had been assayed by the Affymetrix GeneChip HuGene 2.0 ST array[2] and intensity values had been normalized with the Robust Multichip Average algorithm (citation) across all 231 available samples. We constrained our analysis to the 10% highest intensity probes. The 60 participants with follow-up were all female, average age of 43, with a total of 18 cases. The probe intensities were residualized from age to circumvent the predictive value of the probes mediated by age, and scaled and centered. Logistic elastic net was then applied to the data; 5-fold cross-validation on 30 training samples

determined the choice of λ ($\lambda = 0.135$) based on minimum loss, and the log-odds of developing NASH were calculated for the remaining 30 test cases. A model closer to ridge regression ($\alpha = 0.05$) was used.

Since the predictive log-odds of developing NASH were higher for patients who later did develop NASH, it seemed plausible that thresholding the scores would be able to accurately distinguish between cases and controls. Indeed, we found that the log-odds were reasonably discriminatory, with an Area Under the Curve (AUC) of 0.735 (**Figure 4**).

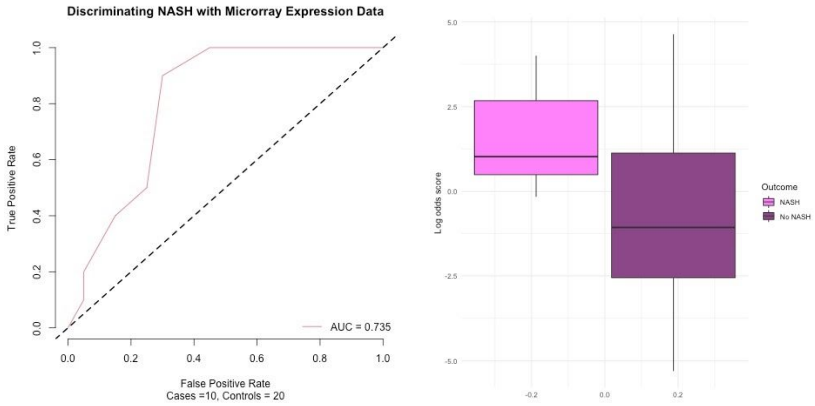


Figure 4: AUC and Log-Odds Predicting NASH based on Logistic Coordinate Descent Elastic Net Model ($\lambda = 0.135$, $\alpha = 0.05$)

Conclusion

The provided code includes these 4 functions to solve elastic net using iterative methods: linear gradient descent, linear ridge regression guided gradient descent (RRGG), linear coordinate descent, and logistic coordinate descent. We introduced linear RRGG descent to decrease run time for linear gradient descent, but, with updates to the linear gradient descent, unguided gradient descent had faster runtimes than RRGG descent for all cases except with α extremely close to 0. Linear and logistic coordinate descent ran much faster for cases with fewer parameters, at speeds comparable to their glmnet equivalents. Yet, analysis of our linear functions showed a discrepancy between our functions and those of the glmnet. For the linear case, estimates were consistent between our functions yet inconsistent with glmnet.

Next, we analyzed microarray data using our logistic coordinate descent algorithm to determine which gene expressions were most associated with developing NASH. We used cross-validation to determine the λ which minimized the CV-loss function. We then applied the logistic elastic net function to half of the dataset, predicted the remaining half, and found that the predicted log-odds of developing NASH were higher for those who were eventually diagnosed with it, demonstrating some predictive potential for this microarray dataset, though our assessment were limited by small sample size and imbalanced case-control ratios. Future work should include validation in a larger dataset.

Without knowing the explicit loss function for glmnet, we cannot compare our results, since the different minimization criteria could explain most differences. For now, our functions appear to corroborate each other and can analyze data meaningfully. Because of the iterative nature of the algorithms, vectorization was often not possible, and the runtime of our functions could therefore most likely be improved through reimplementation in a lower-level language such as C++ or Fortran.

Citations

1. Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2009). Regularization Paths For Generalized Linear Models via Coordinate Descent.
   URL: https://web.stanford.edu/~hastie/Papers/glmnet.pdf
2. Lefebvre, P., Lalloyer, F., Baugé, E., Pawlak, M., Gheeraert, C., et. al. (2017). Interspecies NASH disease activity whole-genome profiling identifies a fibrogenic role of PPARα-regulated dermatopontin. *JCI insight*, *2*(13), e92264.
   https://doi.org/10.1172/jci.insight.92264