GLMNET

Problems

- You have more variables than observations e.g. genomic data
- You have multi-collinearity but don’t know which ones to remove
- You want to discover possible linear models
- You want to rank variables based on effect size, not arbitrary numbers

Terms…

- **Probability: P(data|parameters)** - Used to predict from a model
- **Likelihood: L(parameters|data)** - used to find the best model from data.
- Likelihood is the product of the probabilities of an independent set of events.
- In Maximum Likelihood Estimation, we maximize the log-likelihood $\ell(\theta)$
- Optimization techniques are used to quickly find the MLE parameters
- **Overfitting Concern:** Complex models or limited data can lead to parameter estimates that fit the training data too well, sacrificing generalizability

Penalized Maximum Likelihood

- An extension of the standard maximum likelihood estimation (MLE) framework
- **Likelihood Function:** In MLE, we maximize the log-likelihood $\ell(\theta)$ for parameters θ given data y
- **Addressing overfitting:** We add a penalty term $P(\theta)$ to the log-likelihood, which imposes a cost on extreme or overly complex parameter values. We can also scale up the penalty by multiplying it with a λ term
- So our objective becomes tp maximize $\ell(\theta) - \lambda P(\theta)$
- θ is our parameter set. It is what we want to fit

The Penalty term

L_2 Penalty

The L_2 penalty adds a term to the model's loss (or cost) function that is proportional to the sum of the squares of the model's weights. In regression, this is called "ridge regularization"

In least squares regression, the Sum of Squared Errors (SSE) is defined as:

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

Conceptually, when you add an (L2) penalty (ridge regularization), the loss function could become:

$$J(\mathbf{w}) = \text{SSE} + \lambda \sum_{j=1}^{p} w_j^2,$$

where: - w are the weights of the model, lambda is the regularization parameter controlling the strength of the penalty.

Usually we use the ML function instead of the traditional SSE

Benefits:

- The penalty term increases the loss for any models with large weights. This encourages the optimization algorithm to keep the weights small.
- The final model after optimisation will have weights that are less sensitive to the noise in the training data, which typically leads to better generalization on unseen data.

L1 Penalty

Lasso regression (Least Absolute Shrinkage and Selection Operator) is a technique that performs both variable selection and regularization, which enhances the prediction accuracy and interpretability of statistical models.

Objective Function

using MSE as an example, Lasso regression adds an (L1) penalty term to the loss function:

$$J(\mathbf{w}) = \mathrm{MSE} + \lambda \sum_{j=1}^{p} |w_j|,$$

where: |w_j| is the (L1) norm of the coefficients.

Effects of Lasso

- **Feature Selection:**
  The (L1) penalty has the property of forcing some coefficients to be exactly zero when lambda is sufficiently large. This effectively selects a subset of the original features, leading to a simpler model.
- **Model Complexity:**
  A larger lambda enforces stronger regularization, leading to a model with fewer non-zero coefficients, while a smaller lambda keeps the model closer to standard linear regression.

Practical Considerations

Why do we care?

Ridge

- helps us find the best set of regression coefficients without overfitting.
- Every variable will have a coefficient, but the least important variables will have very low coefficients close to Zero
- can be used to rank our variables

Lasso

- aggressively shrinks unimportant coefficients to Zero and eliminates them from the model entirely
- If you have loads of variables without theoretical basis for removing some, Lasso will demote them for you
- if you have multicollinearity, lasso will eliminate the redundant variables

What if both Lasso and Ridge are attractive?

Elastic Net

- It is essentially a hybrid approach that blends the properties of both lasso (L1) and ridge (L2) regularizations.
- It incorporates both penalties into the loss function, allowing you to balance between feature selection (lasso) and coefficient shrinkage (ridge)

$$J(\mathbf{w}) = \text{MSE} + \lambda \left(\alpha \sum_{j=1}^{p} |w_j| + \frac{1 - \alpha}{2} \sum_{j=1}^{p} w_j^2 \right)$$

- It's essentially the ridge and lasso penalties added together, plus an alpha parameter that determines which of the two penalties is stronger
- You end up with some variables eliminated, some shrunk towards zero.

R implementation

- Glmnet is a package that fits generalized linear and similar models via penalized maximum likelihood.
- You can essentially have an ElasticNet model and tune lambda and alpha and train your models. An alpha of 1 equals Lasso regression, Alpha of 0 equals Ridge regression, anything in the middle is a proportional mixture of the two
- Personally, i use the CARET package to fine tune my hyperparameters

Steps

- Data partitioning
- Setup the training parameter space and cross validation settings
- Now train the model
- Predict
- Draw the confmat
- Model
  - The coef_values are your regression coefficients. Some are reduced to 0 and eliminated from the model by Lasso. You can change alpha to 0 and retrain the model to try out ridge regression and see results.