An Analysis of Penalized Regression in High Dimensional Scenarios

Gabriel Ackall[1]*, Connor Shrader[2]*
Mentor: Dr. Seongtae Kim[3]
[1]Georgia Tech, Civil Engineering
[2]University of Central Florida, Mathematics
[3]NCA&T University, Mathematics and Statistics
*Authors contributed equally

September 19, 2021

Abstract

With the prevalence of big data in recent years, the importance of modeling high dimensional data and selecting influential features has increased greatly. High dimensional data is common in many fields such as genome decoding, rare disease identification, economic modeling, and environmental modeling. However, most traditional regression machine learning models are not designed to handle high dimensional data or conduct variable selection. In this paper, we investigate the use of penalized regression methods such as ridge, least absolute shrinkage and selection operation (lasso), elastic net (E-net), smoothly clipped absolute deviation (SCAD), and minimax concave penalty (MCP) compared to traditional machine learning models such as random forest, XGBoost, and support vector machines. We evaluate these models using factorial design methods for Monte Carlo simulations in 270 environments, with factors being the number of predictors, number of samples, signal to noise ratio, covariance matrix, and correlation strength. We also compare different models using empirical data to evaluate their viability in real-world scenarios. Since our models are regression models, we evaluate the models using the test mean squared error, variable selection accuracy, β-sensitivity, and β-specificity. From our investigation, our findings indicate that penalized regression models outperform more traditional machine learning algorithms in most high-dimensional situations or in situations with a low number of data observations. Machine learning models are not often compared to penalized regression methods and so our analysis helps to expand the scope of how penalized regression is used to help model data. Additionally, the analysis helps to create a greater understanding of the strengths and weaknesses of each model type and provide a reference for other researchers on which machine learning techniques they should use, depending on a range of factors and data environments.

Keywords: penalized regression, variable selection, classification, machine learning, large p small n problem, Monte Carlo simulations

1 Introduction

In the modern world, machine learning techniques such as random forest, gradient boosting, and support vector machines are often touted as versatile one-size-fits-all solutions when it comes to modeling big data [18]. This is due in part to tree based models such as XGBoost winning numerous machine learning competitions [18]. While this versatility is frequently the case, an increasingly common type of data set where there are more predictors than observations can pose challenges for these machine learning algorithms. In these situations, lesser known statistical modeling techniques that perform variable selection can potentially perform equivalently or even better than these machine learning techniques. However, there is a distinct lack of academia focusing on comparing these variable selection techniques with the more traditional machine learning techniques. This paper serves to help bridge that gap.

In these situations where there are more predictors, p, than observations, n, many traditional machine learning techniques either become infeasible to use or fail to give good predictions. The large number of predictors and small number of observations make it easy for such models to **overfit**, meaning that the models become fine tuned to the exact training data and instead of finding generalized patterns for a population of data, they find specific occurrences in the training data [17, 14]. Because of this, overfitted models are sensitive to new data which causes them to perform extremely well on the training data, but poorly on testing data or when deployed in the real world. Because a model's predictions in real world scenarios and on new data is the entire purpose of a model, it is very important to reduce overfitting so that predictive accuracy in these scenarios is maximized.

This paper investigates several methods to handle the large p, small n problem. First, We considered wrapper methods such as forward selection, backward selection, stepwise forward selection and stepwise backward selection using both Akaike information criterion (AIC) and Bayesian information criterion (BIC) as the stopping criteria for the models [1, 20]. These models fit several linear models using different subsets of predictors and selects the model that optimizes a specific metric. In addition, we studied penalized regression models such as ridge regression [16], least absolute shrinkage and selection operation (lasso) [21], elastic-net [25], smoothly clipped absolute deviation (SCAD) [12], and minimax concave penalty (MCP) [23]. These models simultaneously select important predictors and fit a linear model. Finally, we considered a few machine learning models: random forests (RF) [5], gradient boosting in the form of XGBoost [7], and support vector machine (SVM) models [8]. These models do not assume a linear relationship between a response and its predictors. To compare these different techniques, models were trained and evaluated using both Monte Carlo simulations and empirical genomic data.

Section 2 contains details about each model and details the implementation of these models for our study. Section **??** describes our simulation study design and results, while section **??** explains our empirical data analysis and results. Section **??** is a discussion of our results and Section **??** is the conclusion.

2 Methodology

2.1 Subset Selection Methods

Subset selection methods are wrapper methods that attempt to find a subset of the predictors $X_1, X_2, \ldots, X_p$ that are most correlated with the response variable Y. These algorithms usually fit models for many different subsets and choose the subset of predictors that results in the best model. Although subset selection techniques can be applied to many types of models, we will focus on subset selection with linear regression.

Two subset selection methods we consider are **forward selection** and **backward selection**. Forward selection begins by fitting a model with no predictors (only the intercept is non-zero) and iteratively adds predictors into the model. The predictor added at each step is chosen to best increase the model fit. Conversely, backward selection starts from the full (ordinary least squares) model with all p predictors and repeatedly removes predictors. Then, like best subset selection, the final model is chosen from the candidate models fitted at each step. Note that backward selection can only be used when $p \leq n$ since ordinary least squares cannot be used when $p > n$. Forward selection can always be used.

Although forward and backward selection will not always encounter the best possible model, these methods help optimize best subset selection and are quicker than best subset selection. Consequently, forward and backward selection can be used for larger values of p. However, despite these optimization, forward and backwards selection are still plagued by the fact that the number of combinations of predictors increases exponentially and thus they cannot be used for very large values of p.

The models produced by forward and backward selection can be improved by allowing predictors to be added and removed in the same algorithm. **Forward stepwise selection** begins with an empty model and iteratively improves the model by either adding a new predictor or removing an obsolete one. **Backward stepwise selection** works in the same way but starts with the full model. Like backward selection, backward stepwise selection can only be used when $p \geq n$. These techniques take longer to run than ordinary forward and backward selection, but they are more likely to find the best possible model.

When fitting a model using any of the subset selection methods, the performance metric used to select the best model is very important. At first, it may seem reasonable to choose a metric such as the residual sum of squares from Equation **??**. However, many metrics, including the residual sum of squares, only describe a model's performance on training data. This is problematic because including more predictors will always decrease the residual sum of squares on the training data. If $p \leq n$, then the model fitted with all p predictors is exactly the same model produced by ordinary least squares, which by definition minimizes the residual sum of squares! If $p > n$, then a model with the maximum possible number of predictors would be selected.

If we wish to produce a model that makes reliable predictions on test data, we must use a different performance metric. Two of the most common metrics used for this purpose are the **Akaike information criterion** (AIC) and the **Bayesian information criterion** (BIC) [1, 20]. For a given model, the Akaike information criterion can be computed by

$$\text{AIC} = 2p - 2\ln(L) \tag{1}$$

where p is the number of predictors for the model and L is the likelihood of the model. This likelihood represents how well a model fits to the training data. The purpose of the term $2p$ is to punish models with a large number of coefficients. The model with the lowest AIC is the one selected by the wrapper methods.

The Bayesian information criterion very similar to the Akaike information criterion; it is given by

$$\text{BIC} = p\ln(n) - 2\ln(L) \tag{2}$$

where again p is the number of predictors for the model and L is the likelihood of the model.

Note that if $n > 7$, then $\ln(n) > 2$ and so the penalty for BIC is larger than the penalty for AIC. Hence, a model selected using BIC will typically have fewer parameters than a model selected by AIC.

In addition to AIC and BIC, there are several other metrics that modify training error to estimate test error, such as C_p and adjusted R^2 [17]. However, this paper will focus on AIC and BIC.

2.2 Penalized Regression

In general, **penalized regression** works by fitting a model that punishes large coefficient estimates. By forcing coefficient values to shrink, the resulting model will have relatively low variance. All of the models discussed here can be used when there are more predictors than observations. Most, but not all, of these methods can also perform variable selection.

Almost all of the penalized regression methods in this paper solve an optimization problem of the form

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \Big[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \Big]^2 + \lambda \sum_{j=1}^{p} P(\beta_j) \right\} \tag{3}$$

where the first summation is the usual residual sum of squares, $\lambda \geq 0$ is a hyperparameter that controls the strength of the penalty, and $P(\beta)$ is a penalty function applied to each of the coefficients. The penalty is not applied to the intercept β_0. In general, $P(\beta)$ is an even function that is non-decreasing as $|\beta|$ increases. If $\lambda = 0$, then we have the usual ordinary least squares estimator. As λ increases, a stronger penalty is applied which will decrease the coefficient values. As λ approaches ∞, the model becomes an empty model where only the intercept term is non-zero.

When fitting penalized regression models, the choice of λ is very important. If λ is too small, then models may be overfit just like ordinary least squares; on the other hand, if λ is large, then the resulting models are highly biased and the resulting models may be underfit. In practice, the best value of λ can be selected using **cross validation** and **grid search**. This process involves splitting the training set into k disjoint subsets of equal size called **folds**. Then, k models are fitted for different values of λ, where one fold is used as a testing set and the other $(k-1)$ folds are used for training. The value of λ that results in the lowest test error can then be selected.

Ridge regression is a penalized regression model that uses the penalty function $P(\beta) = \beta^2$ [16]. One advantage of ridge regression is that it can handle highly correlated data

better than ordinary least squares. When predictors are correlated with each other, some algorithms have a hard time distinguishing which predictors are actually related to the response.

One drawback of ridge regression is that it cannot perform variable selection. It can shrink coefficients towards zero, but it cannot set coefficients to be exactly zero.

The **least absolute shrinkage and selection operation**, commonly referred to as **lasso**, is a shrinkage method with a very similar form to ridge regression [21, 17]. The penalty function for lasso is $P(\beta) = |\beta|$.

Unlike ridge regression, lasso regression can perform variable selection by setting coefficients to zero. This makes lasso regression favorable when most predictors are not related to the response variable. On the other hand, if the response truly does depend on all of the predictors, then lasso regression may incorrectly set some coefficients to zero. In addition, lasso regression does not have a closed-form solution, but computing the coefficients is still efficient.

Figure 1 provides a visual explanation for why ridge regression cannot perform variable selection while lasso regression can. Suppose that there are $p = 2$ predictors. The red circle in Figure 1a represents the circle $\beta_1^2 + \beta_2^2 \leq t$, which is the penalty boundary for ridge regression when $t = 1$. The red square in Figure 1b represents the boundary $|\beta_1| + |\beta_2| \leq t$ for lasso regression. The point in the center of the blue ellipses represents the values of β_1 and β_2 that ordinary least squares would estimate. Because this point is not within either of the red regions, ridge regression and lasso regression will give different coefficient estimates. The blue ellipses around this point represent contour curves for the residual sum of squares, which is a quadratic function of β_1 and β_2. The first intersection of these ellipses with the red regions represents the coefficient values selected by ridge and lasso regression, since it represents the point within the red region with the lowest residual sum of squares. Because the ridge regression boundary is round, the intersection in Figure 1a occurs when β_1 and β_2 are both positive. For lasso regression in Figure 1b, the intersection occurs at a corner where $\beta_1 = 0$. Thus, lasso regression has set β_1 to zero, while β_1 is non-zero for ridge regression.

Elastic-net regression uses both the ridge penalty and the lasso penalty at the same time [25]. Elastic-net solves the optimization problem

$$\hat{\beta}^{\text{E-net}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left[y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \right]^2 + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\} \tag{4}$$

where λ_1 and λ_2 are both tuning parameters.

An important limitation to note is that elastic net performs best when it close to either ridge or lasso regression, meaning that either λ_1 greatly exceeds λ_2 or λ_2 greatly exceeds λ_1 [25]. Additionally, because elastic net requires two tuning parameters, this makes it much more difficult to determine the best combination of tuning parameters to minimize error in the regression. However, this problem has been largely solved through by the LARS-EN algorithm developed by Zou et. al. which efficiently solves for the tuning parameters [25].

One major flaw of the lasso method is that the penalty punishes large coefficients, even if those coefficients should be large. One way to modify the lasso method is to use the **smoothly clipped absolute deviation** (SCAD) penalty [12]. The goal of this method is

to punish large coefficients less severely, which can help mitigate some of the bias introduced by the lasso method. The penalty function $P(\beta)$ for SCAD satisfies

$$\frac{dP}{d\beta} = \text{sign}(\beta)\left[I(|\beta| < \lambda) + \frac{\max(a\lambda - |\beta|, 0)}{(a-1)\lambda} I(|\beta| > \lambda)\right] \tag{5}$$

where $a > 2$ is a new hyperparameter and I is the indicator function ($I(Q)$ equals 1 if a statement Q is true, and equals 0 if Q is false). If the hyperparameter a is large, then SCAD behaves like lasso for larger coefficient values. Also, notice that λ is an argument for the penalty function P.

An equivalent way to write the expression in Equation 5 is

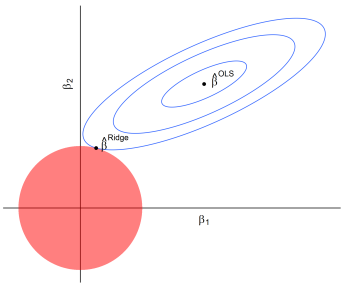
$$\frac{dP}{d\beta} = \begin{cases} 1, & |\beta| \leq \lambda \\ \frac{a\lambda - |\beta|}{(a-1)\lambda}, & \lambda < |\beta| < a\lambda \\ 0, & \alpha\lambda < |\beta| \end{cases} \tag{6}$$

This penalty function does not punish coefficients with large magnitude as heavily as the lasso method. In fact, if the magnitude of a coefficient is larger than $a\lambda$, then the penalty becomes constant since the derivative becomes zero.

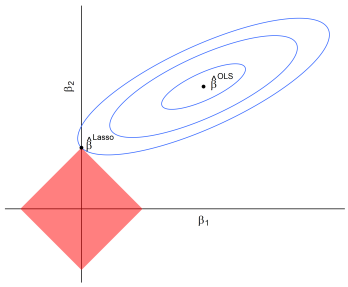
By integrating with respect to β and choosing $P(\beta) = 0$ [2], we see that

$$P(\beta) = \begin{cases} |\beta|, & |\beta| \leq \lambda \\ \frac{2a\lambda|\beta| - \beta^2 - \lambda^2}{2(a-1)\lambda}, & \lambda < |\beta| \leq a\lambda \\ \frac{\lambda(a+1)}{2}, & a\lambda < |\beta| \end{cases} \tag{7}$$

The **minimax concave penalty** (MCP) method is very similar to SCAD [23]. Both methods are used to avoid the high bias caused by the lasso method. MCP uses a penalty



(a) RSS contours and the ridge penalty boundary.



(b) RSS contours and the lasso penalty boundary.

Figure 1: RSS contours and penalty bounds for the ridge and lasso models when $p = 2$ and $t = 1$. The red regions represent the coefficient values allowed by ridge and lasso regression, respectively. The blue ellipses represent contours of the residual sum of squares, with $\hat{\beta}^{\text{OLS}}$ being the point where the residual sum of squares is minimized. The intersection of the ellipse with the red region in each plot represents the coefficient values selected by lasso and ridge.

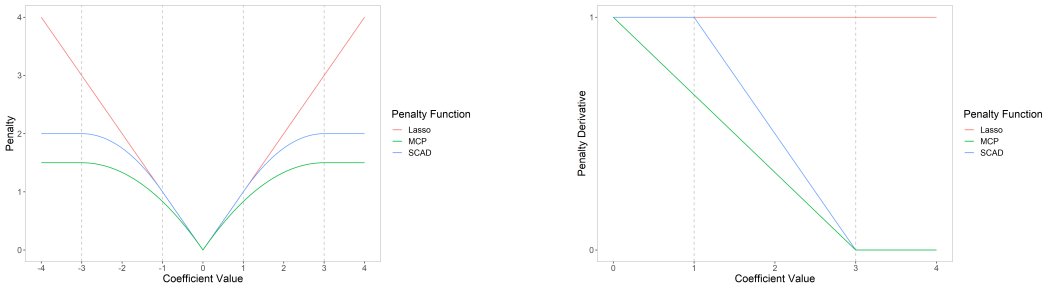
function that satisfies

$$\frac{dP}{d\beta} = \begin{cases} \text{sign}(\beta)\left(1 - \frac{|\beta|}{a\lambda}\right), & |\beta| \leq a\lambda \\ 0, & a\lambda < |\beta| \end{cases} \tag{8}$$

where, like SCAD, $a > 0$ is a hyperparameter. Integrating [2], we see that

$$P(\beta) = \begin{cases} |\beta| - \frac{\beta^2}{2a\lambda}, & |\beta| \leq a\lambda \\ \frac{1}{2}a\lambda, & a\lambda < |\beta| \end{cases} \tag{9}$$

Figure 2 below shows the penalty functions (and their derivatives) for lasso, SCAD, and MCP as a function of a coefficient value β. We see that lasso applies a much stronger penalty to large coefficients than SCAD or MCP. Also, note that SCAD starts with a derivative equal to that of the lasso for small values of β; on the other hand, the derivative of the penalty function for MCP starts decreasing immediately.



(a) Penalty functions for lasso, SCAD, and MCP.

(b) Derivatives of the penalty functions for lasso, SCAD, and MCP.

Figure 2: Penalty functions for lasso, SCAD, and MCP, as well as their derivatives. These plots use $\lambda = 1$ and $a = 3$. The dashed vertical lines are the knots for SCAD and MCP.

Note that of the penalized regression methods discussed above, only ridge regression has a general closed-form solution. Although lasso, elastic-net, SCAD and MCP do not have closed-form solutions, their solutions can be efficiently approximated [13, 3]. In some special cases, however, a closed-form solution exists.

Let $\mathbf{X}$ be a $n \times p$ matrix where each row contains the predictor values for one observation. To simplify things, we will assume that our data is centralized so that the coefficient β_0 is 0; that way, we do not need to include an extra entry of 1 in each row of $\mathbf{X}$. In an **orthonormal design**, we assume that $\mathbf{X}$ is orthonormal, meaning that the magnitude of each column of $\mathbf{X}$ is one and every pair of columns of $\mathbf{X}$ is orthogonal. In this special case, the matrix $(\mathbf{X}^\top\mathbf{X})^{-1}$ is the $p \times p$ identity matrix. As a result, each coefficient is independent of the other coefficients; in other words, we can compute the coefficient estimate for each predictor without needing to use the values for the other predictors. For example, in an orthonormal design, the general solution to ordinary least squares regression given in Equation **??** simplifies to

$$\hat{\boldsymbol{\beta}}^{\text{OLS}} = \mathbf{X}^\top\mathbf{y} \tag{10}$$

To compute the coefficient estimates for lasso, SCAD, and MCP in an orthonormal design, we first compute the vector $\mathbf{z} = \mathbf{X}^\top\mathbf{y}$ [12]. Each entry of $\mathbf{z}$ corresponds to one

predictor. We then apply a **thresholding function** to each entry of $\mathbf{z}$ to get the coefficient estimate for that predictor. For lasso, elastic-net, SCAD, and MCP, a closed-form for this thresholding function exists [21, 12, 25, 23]. The input to the thresholding function is the actual coefficient value, and the output is the estimated value for that coefficient in an orthonormal design. Figure 3 shows the threshold functions for lasso, SCAD and MCP when $\lambda = 1$. For SCAD and MCP, we used the value $a = 3$. For reference, the identity line is included, which can be considered as the threshold function for ordinary least squares.

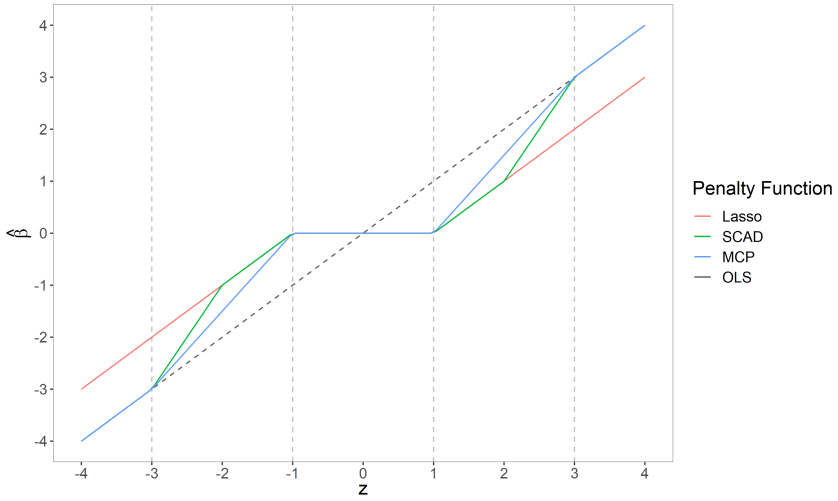


Figure 3: Thresholding function for lasso, SCAD, and MCP when $\lambda = 1$ and $a = 3$ in an orthonormal design. The input (z) represents an entry from the vector $\mathbf{z} = \mathbf{X}^\top \mathbf{y}$; the output ($\hat{\beta}$) is the coefficient estimate. The dashed vertical lines are the knots for SCAD and MCP.

We see that when $|z|$ is small, the thresholding function for lasso, SCAD, and MCP are all zero. This matches with the assumption that these methods should set small coefficient estimates to zero. As $|z|$ increases, the estimated value for that coefficient gets larger. Notice that ordinary least squares doesn't decrease the estimated coefficient value for any choice of z; this is because of the fact that ordinary least squares is unbiased, whereas the other methods are biased.

In this figure, we also see that the threshold function for lasso is parallel to the identity line for large values of β. This means that even if a predictor is very influential on the response, lasso will predict a coefficient estimate that is less than the true coefficient value. This makes lasso a very biased model. This issue was one of the main motivations for SCAD and MCP. Unlike lasso, SCAD and MCP converge to the identity line as β increases, which makes these methods less likely to decrease large coefficient estimates.

Another feature of SCAD and MCP is their oracle-like properties [12, 23]. The term **oracle** in this context was first proposed by Donohe and Johnstone in [9]. Suppose that a linear model could be fitted with the aid of an oracle. This oracle could tell you the subset of predictors that are truly related to the response so that a model can be fitted using only the important predictors. Although such a model is not possible in practice, this oracle procedure serves an ideal that can be worked towards.

2.3 Non-linear models

We next discuss several non-linear methods for regression: random forests, gradient boosting, and support vector machines.

Random Forest models (RF) are tree-based and solve the issue of high variance from individual decision trees by aggregating the predictions of many independent decision trees [5]. Each tree is fit using a subset of the observations and a subset of the predictors so that the trees are relatively independent from one another.

To fit each decision tree within a random forest model, we first select a random sample of the n observations with replacement, in a process called **bootstrapping** [11]. The number of observations chosen for each tree is a hyperparameter that can be changed. After selecting a set of observations, a random sample of predictors are chosen out of the p predictors without replacement. Again, this helps decorrelate the trees. The number of predictors used in each tree is also a hyperparameter that can be changed. A decision tree is then generated using the available observations and predictors.

A random forest model aggregates all of the individual decision trees into a single model. The number of trees generated, B, is a hyperparameter that can be changed; usually, a random forest model will contain at least several hundred trees. To make predictions with a random forest, a test observation is passed into each decision tree and the predictions from each tree are aggregated. Figure 4 demonstrates how a prediction is made using a random forest. For regression, the results are normally aggregated using the mean; for classification, the prediction chosen most often by the trees is usually used as the final prediction.

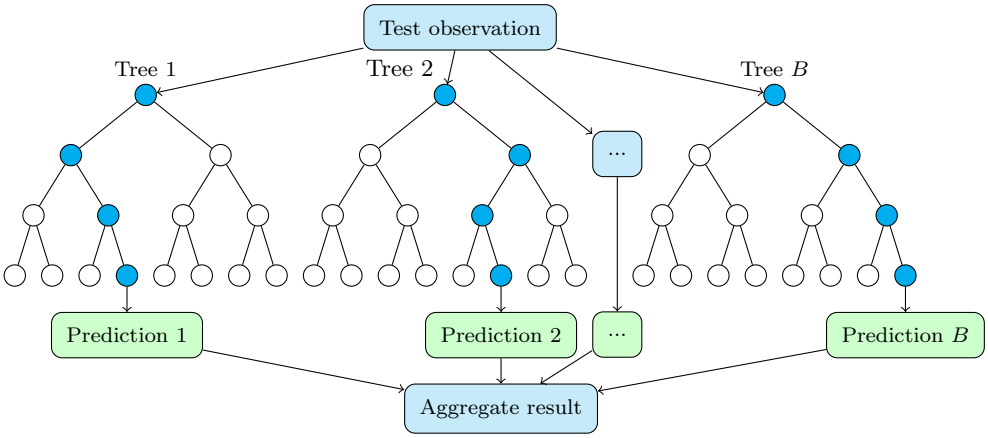


Figure 4: Visualization of how predictions are made using a random forest model. An observation is input into each decision tree. Predictions from each tree are then aggregated into a single result that is used as the final prediction.

This idea of fitting multiple models with bootstrapping and aggregating the results can be used with other models besides decision trees. In general, this process is called **bagging** (a combination of the words "bootstrap" and "aggregating") [4].

Boosting is the technique of sequentially improving a weak learner until it becomes a strong learner [19]. A **gradient boosting machine** (GBM) is a boosting technique that

uses gradient descent to minimize error in a model and correct the shortcomings of the previous model [15]. Boosting can be used on different types of models, but decision trees are the most common to use. Unlike random forest models, where each tree is independent of one another, the trees in a boosting model are grown sequentially. Each tree is fitted to correct the mistakes made by the previous tree. The details of how errors are corrected depends on the algorithm selected. With regression, each tree can be fitted by using the residuals from the previous tree as the training data [17]. Like random forest models, predictions are made by aggregating the results from each individual tree. See figure 5 for a diagram of how a boosting model is fitted.

Gradient boosting machines that use decision trees can be used for both regression and classification. When using a gradient boosting model with decision trees, **relative variable importance** and **pruning** can be used as a sort of pseudo-variable selection method to lower complexity and prevent over-fitting. However, this does not yield the same interpretability or bias trade off benefits that true variable selection yields.

Gradient boosting models often suffer from slow computation speeds due to the large number of sequential models that need to be trained. **Extreme Gradient Boosting** (XGBoost) is a faster version of gradient boosting that utilizes parallel computing as well as different optimization techniques to speed up computation [6]. For these reasons, XGBoost is often preferred over standard gradient boosting models and is very commonly used in many machine learning applications. In this study, we will use the XGBoost algorithm for gradient boosting.

There are a few different hyperparameters that can control how an XGBoost model is fit. The learning rate controls how quickly the gradient boosting model learns. If this learning rate is high, then the model learns quickly, but it may not learn as efficiently. If the learning rate is low, then the model takes longer but typically makes better predictions. The maximum tree depth determines how high each tree can be. Using smaller tree sizes may oversimplify the model, but it could also mitigate overfitting.

The final non-linear model that we considered is the **support vector machine** (SVM)

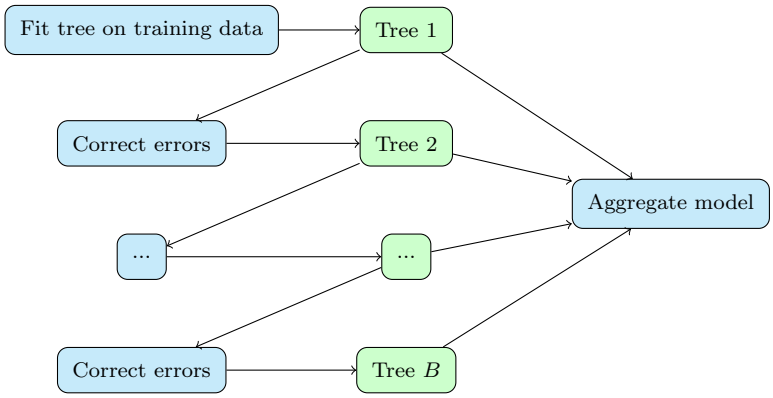


Figure 5: How a boosting model with decision trees is fitted. Each tree is fitted to correct the errors of the previous tree. Predictions are made by combining the results from each decision tree.

[8]. Support vector machines are versatile models that can be used for both classification and regression. Originally, a support vector machine was designed as a binary classifier that separates the two response classes with a hyperplane in $(p-1)$-dimensional space. The hyperplane chosen by a support vector machine is chosen to maximize the distance between the hyperplane and any of the observation points. The observations closest to this separating hyperplane are called **support vectors**. Predictions are made by determining which side of the hyperplane an observation lies on. Observations that lie above the hyperplane are assigned to one class, and observations on the other side are predicted to be the other class. Figure 6 demonstrates a simple support vector machine for classification.

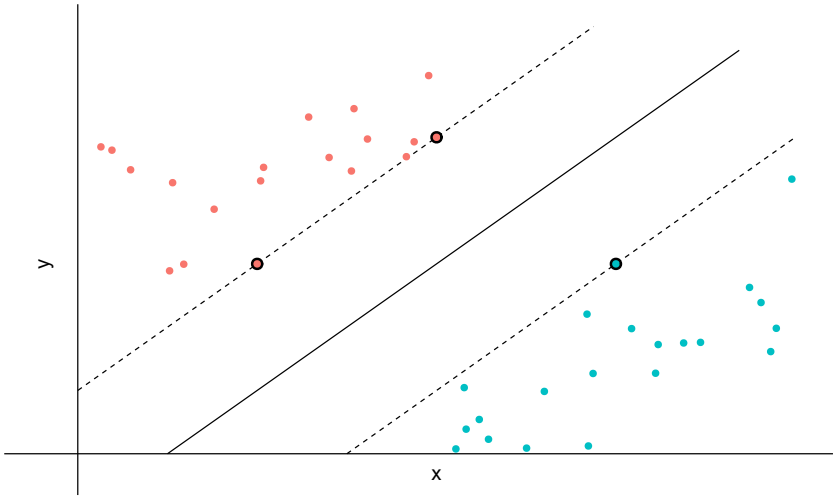


Figure 6: Visualization of a simple support vector machine for binary classification. The solid black line is the hyperplane that maximizes the margin between the two classes. The points with a black outline are the support vectors. They are the points that define the hyperplane.

Note that in most data sets, the classes cannot be split perfectly into two sides. Furthermore, this model has very high variance as changing the points near the hyperplane can significantly alter the hyperplane. Finally, this model cannot handle cases where the true boundary is non-linear. Luckily, support vector machines in practice can handle such issues. Typically, support vector machines are allowed to misclassify some of the training data, which address the cases where the data cannot be split perfectly by a hyperplane. Also, the predictor space can be enlarged to handle non-linear decision boundaries; this is typically done by using **kernels**. For example, using a radial kernel can create decision boundaries that enclose regions of the p-dimensional space.

Support vector machines can also be generalized to handle classification when there are more than two classes. They can also be used for regression [10]. We will be using support vector machines for regression in our study.

Like the other non-linear methods, there are many different hyperparameters that can be tuned. The two hyperparameters that we considered are ϵ and the cost function. The hyperparameter ϵ defines how tolerant the algorithm is of small errors. If ϵ is large, then the support vector machine will tolerate larger errors; if ϵ is small, then even small errors

will be punished. The cost hyperparameter C determines how strongly the model punishes incorrect predictions. If C is large, then the model punishes incorrect predictions more, making it fit more tightly to the training set. If C is smaller, then the model is likely to allow more incorrect predictions in the training data.

References

[1] Akaike, H. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

[2] Breheny. Adaptive lasso, mcp, and scad. URL: https://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-29.pdf, 2016.

[3] Breheny, P. and Huang, J. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, 5(1):232–253, 2011.

[4] Breiman, L. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] Breiman, L. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[7] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., and Li, Y. *xgboost: Extreme Gradient Boosting*, 2021. R package version 1.4.1.1.

[8] Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] Donoho, D. L. and Johnstone, J. M. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[10] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., Vapnik, V., et al. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.

[11] Efron, B. and Tibshirani, R. J. *An introduction to the bootstrap*. CRC press, 1994.

[12] Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[13] Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[14] Friedman, J., Hastie, T., Tibshirani, R., et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[15] Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[16] Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[17] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An introduction to statistical learning*, volume 112. Springer, 2013.

[18] Nielsen, D. Tree boosting with xgboost-why does xgboost win "every" machine learning competition? 2016.

[19] Schapire, R. E. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[20] Schwarz, G. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[21] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[22] Zeng, Y. and Breheny, P. The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r. *ArXiv e-prints*, 2017.

[23] Zhang, C.-H. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[24] Zou, H. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[25] Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.