Linear Regression - Least Squares Coefficient Estimate and Other Approaches

October 5, 2020

In this document, we will discuss how to determine the coefficient values for variables. We describe the most common way to obtain the coefficient estimate, which is called *least squares coefficient estimate*. In order to achieve higher prediction accuracy and better model interpretability, we will introduce two ways to determine the coefficient values: 1) subset selection; and 2) shrinkage methods including ridge regression and Lasso.

1 Least Squares Coefficient Estimate

The general formula for linear regression can be written as $h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k$, where $x_1, x_2, ..., x_k$ are the values for k features, θ_0 is the constant, and $\theta_1, \theta_2 ..., \theta_k$ are coefficient values for k features. This section is to describe the algorithm and/or mathematical formula to obtain the values for $\theta_0, \theta_1, \theta_2 ..., \theta_k$ such that the *mean squared error* (MSE) is minimized. The detailed problem statement is shown as follows.

Suppose our training dataset has n data points such that $\{(x_1^{(1)}, x_2^{(1)}, ..., x_k^{(1)}, y^{(1)}), (x_1^{(2)}, x_2^{(2)}, ..., x_k^{(2)}, y^{(2)}), ..., (x_1^{(n)}, x_2^{(n)}, ..., x_k^{(n)}, y^{(n)})\}$, we would love to determine $\theta_0, \theta_1, \theta_2 ..., \theta_k$ so as to minimize the formula

$$RSS = J(\theta) = J(\theta_0, \theta_1, \theta_2 ..., \theta_k) = \sum_{i=1}^{n} [y^{(i)} - (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_k x_k^{(i)})]^2, \tag{1}$$

where $MSE = RSS/n$.

1.1 Simple Linear Regression

In this subsection, we focus on the linear regression with only one available feature. Therefore, the training dataset is $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(n)}, y^{(n)})\}$ and the objective in Eq. (1) can be written as $J(\theta_0, \theta_1) = \sum_{i=1}^{n} [y^{(i)} - (\theta_1 x^{(i)} + \theta_0)]^2$. We provide two ways to obtain the solution: one is a mathematical way to obtain the exact values for θ_0 and θ_1 through calculus and statistics, and the

other one is an algorithmic way to obtain the approximate values for θ_0 and θ_1 through the gradient descent algorithm.

1.1.1 Exact Values of θ_1 and θ_0

Through the calculus, we can obtain two following equations:
$\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_1} = \frac{\partial\{\frac{1}{n}\sum_{i=1}^{n}[y^{(i)}-\theta_1 x^{(i)}-\theta_0]^2\}}{\partial \theta_1} = 2\cdot\sum_{i=1}^{n} x^{(i)}\cdot(\theta_1 x^{(i)}+\theta_0-y^{(i)})$
$= 2\left\{\left(\sum_{i=1}^{n}\left[x^{(i)}\right]^2\right)\theta_1+\left(\sum_{i=1}^{n}\left[x^{(i)}\right]\right)\theta_0-\left(\sum_{i=1}^{n}\left[x^{(i)}\cdot y^{(i)}\right]\right)\right\}=0$
and $\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_0} = \frac{\partial\{\frac{1}{n}\sum_{i=1}^{n}[y^{(i)}-\theta_1 x^{(i)}-\theta_0]^2\}}{\partial \theta_0} = 2\cdot\sum_{i=1}^{n}(\theta_1 x^{(i)}+\theta_0-y^{(i)})$
$= 2\left\{\left(\sum_{i=1}^{n}\left[x^{(i)}\right]\right)\theta_1+n\theta_0-\left(\sum_{i=1}^{n}\left[y^{(i)}\right]\right)\right\}=0$
Therefore, we obtain a and b through the above two equations as follows.

$$\theta_1 = \frac{\sum_{i=1}^{n}(x^{(i)}-\bar{x})(y^{(i)}-\bar{y})}{\sum_{i=1}^{n}(x^{(i)}-\bar{x})^2} \tag{2}$$

and

$$\theta_0 = \bar{y}-\theta_1\bar{x}, \tag{3}$$

where $\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x^{(i)}$ and $\bar{y}=\frac{1}{n}\sum_{i=1}^{n}y^{(i)}$ are the average value of the training dataset.

1.1.2 Gradient Descent Algorithm

Although we can obtain exactly value of θ_1 and θ_0 through calculus, we may also obtain θ_1 and θ_0 approximately through gradient descent algorithm shown in Algorithm 1.

Algorithm 1 GradientDescent(ε, α)
/* Input:
* ε is the preset error and α is the learning rate.
* Output:
* θ_1 and θ_0
*/
1: $\theta_1=\theta_0=0$
2: do
3: Set $\theta_0^{'} := \theta_0-\alpha\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_0} = \theta_0-\alpha\cdot\left[\sum_{i=1}^{n}(\theta_1 x^{(i)}+\theta_0-y^{(i)})\right]$
4: Set $\theta_1^{'} := \theta_1-\alpha\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_1} = \theta_1-\alpha\cdot\left[\sum_{i=1}^{n}(\theta_1 x^{(i)}+\theta_0-y^{(i)})\cdot x^{(i)}\right]$
5: Set $\Delta J := J(\theta_0,\theta_1)-J(\theta_0^{'},\theta_1^{'})$
6: Set $\theta_0 := \theta_0^{'}$ and $\theta_1 := \theta_1^{'}$
7:while $\Delta J \geq \varepsilon$

Note that each iteration at Line 5 should make $J(\theta_0,\theta_1)\geq J(\theta_0^{'},\theta_1^{'})$.

In order to obtain θ_1 and θ_0 close to the exact values, we set $\varepsilon\leq 0.001$. Now, we consider how to determine the learning rate α. If the learning rate is very small, then the convergence of the algorithm (a.k.a the number of iterations of

loops) will be very slow. If the learning rate is large, $J(\theta_0, \theta_1)$ may not always decrease for the iterations. Furthermore, the large learning rate may cause the algorithm diverge. As for the practice, we start with $\alpha = 0.001$ and increase three times ($\alpha := 3\alpha$) each time until we reach an acceptable α. Finally, we choose αwhich is a slightly smaller than the acceptable α.

1.2 Linear Regression with Multiple Variables

Now, we extend the last subsection into multiple variable cases based on the discussion in the above subsection. We will see that the mathematical way will be very complicated by introducing the concepts of matrix, however, the key concepts in this extension will be used in other regression approaches for example *support vector machine* (SVM). We will also introduce the extension of the algorithmic way to obtain the coefficient values approximately.

1.2.1 Matrix Representation of Multiple Variables

As for the linear regression $h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k$, we introduce the new variable $x_0 = 1$ so that we can rewrite $h_\theta(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_k x_k = \theta^T X$ where vectors $\theta = (\theta_0, \theta_1, ..., \theta_k)^T$ and $X = (x_0, x_1, ..., x_k)^T$. Now we determine $\frac{\partial J(\theta)}{\partial \theta_j} = -2\sum_{i=1}^{n}[y^{(i)} - (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_k x_k^{(i)})]x_j^{(i)} = -2\sum_{i=1}^{n}[y^{(i)} x_j^{(i)} - \theta^T X^{(i)} x_j^{(i)}] = 0$for $0 \leq j \leq k$. We define the matrix as

$$\Phi = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & ... & x_k^{(1)} \\ x_0^{(2)} & x_1^{(2)} & ... & x_k^{(2)} \\ : & : & : & : \\ : & : & : & : \\ x_0^{(n)} & x_1^{(n)} & ... & x_k^{(n)} \end{bmatrix}_{n \times (k+1)}$$

Then the partial derivative equation can be written as $\Phi^T y - (\Phi^T \Phi)\theta = 0$ where $y = (y^{(1)}, y^{(2)}, ..., y^{(n)})^T$. If $|\Phi^T \Phi| \neq 0$, then we obtain $\theta = (\Phi^T \Phi)^{-1} \Phi^T y$. Although we have the formula to obtain the exact values for coefficient, it may not be computable due to the time complexity of the matrix multiplication. Therefore, we introduce the numerical way through gradient descent algorithm for multiple variables.

1.2.2 Gradient Descent Algorithm for Multiple Variables

Note that $x_0^{(i)} = 1$ at Line 3 for all data points $1 \leq i \leq n$ and each iteration at Line 4 should make $J(\theta_0, \theta_1, ..., \theta_k) \geq J(\theta_0^{'}, \theta_1^{'}, ..., \theta_k^{'})$.

The setting of ε and α can be the same as the setting for simple linear regression. Thus, we omit the discussion of those two parameters and focus on the special requirement for multiple variables. Usually, different features may have different ranges of data. For example, the house size may be from hundreds square feet to thousands square feet. On the other hand, the number

Algorithm 2 GradientDescent(ε, α)
/* Input:
* ε is the preset error and α is the learning rate.
* Output:
* θ_l for $0 \leq l \leq k$
*/
1: $\theta_l = 0$ for $0 \leq l \leq k$
2: do
3: Set $\theta_l^{'} := \theta_l - \alpha \frac{\partial J(\theta_0, \theta_1, ..., \theta_k)}{\partial \theta_l} = \theta_l - \alpha \cdot \frac{1}{n} \cdot \left[\sum_{i=1}^{n} (\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \cdots + \theta_k x_k^{(i)} - y^{(i)}) x_l^{(i)}\right]$ for $0 \leq l \leq k$
4: Set $\Delta J := J(\theta_0, \theta_1, ..., \theta_k) - J(\theta_0^{'}, \theta_1^{'}, ..., \theta_k^{'})$
5: Set $\theta_l := \theta_l^{'}$ for $0 \leq l \leq k$
6:while $\Delta J \geq \varepsilon$

of bedrooms in the house may be from 0 to 5. When we use the original data to predict the house price through linear regression, we can easily observe that the house size will have greater impacts than the number of bedrooms in our algorithm based on the calculation at Line 3. In order to avoid the different ranges among features, we introduce one popular way to polish original data into approximately -1 to 1 range. As for the j^{th} data of feature i, we obtain the new data $\widetilde{x}_i^{(j)} = \frac{x_i^{(j)} - u_i}{\sigma_i}$ where $u_i = \frac{\sum_{j=1}^{n} x_i^{(j)}}{n}$ is the mean value of feature i and $\sigma_i = \sqrt{\frac{\sum_{i=1}^{n} (x_i^{(j)} - u_i)^2}{n}}$ is the standard deviation of feature i.

2 Linear Model Selection and Regularization

We have seen in the above subsections that one typically fits linear model using least squares. However, we may want to use another fitting procedure instead of least squares. As we will see, alternative fitting procedure can yield better *prediction accuracy* and *model interpretability*.

1. *Prediction Accuracy*: Provided that the true relationship between the response and the predictors is approximately linear, the least squares estimates will have low bias. If $n >> p$ — that is, if n, the number of observations, is much larger than p, the number of variables—then the least squares estimates tend to also have low variance, and hence will perform well on test observations. However, if n is not much larger than p, then there can be a lot of variability in the least squares fit, resulting in overfitting and consequently poor predictions on future observations not used in model training. And if $p > n$, then there is no longer a unique least squares coefficient estimate: the variance is infinite so the method cannot be used at all. By constraining or shrinking the estimated coefficients, we can often substantially reduce the variance at the cost of a negligible increase in bias. This can lead to substantial improvements in

the accuracy with which we can predict the response for observations not used in model training.

2. *Model Interpretability*: It is often the case that some or many of the variables used in a multiple regression model are in fact not associated with the response. Including such irrelevant variables leads to unnecessary complexity in the resulting model. By removing these variables—that is, by setting the corresponding coefficient estimates to zero—we can obtain a model that is more easily interpreted. Now least squares is extremely unlikely to yield any coefficient estimates that are exactly zero.

There are many alternatives, both classical and modern, to using least squares estimate. Here, we discuss two important classes of methods.

1. *Subset Selection.* This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.

2. *Shrinkage.* This approach involves fitting a model involving all p predictors. However, the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as *regularization*) has the effect of reducing variance. Depending on what type of shrinkage is performed, some of the coefficients may be estimated to be exactly zero. Hence, shrinkage methods can also perform feature selection.

2.1 Subset Selection

In this subsection, we consider some methods for selecting subsets of predictors. We will describe the measurements for selecting the subset of features. We then introduce the best subset and stepwise model selection procedures.

2.1.1 Measurements for Subset Selection

Usually, the training set *mean squared error* (MSE) is generally an underestimate of the test MSE. (Recall that MSE = RSS/n.) This is because when we fit a model to the training data using least squares, we specifically estimate the regression coefficients such that the training RSS (but not the test RSS) is as small as possible. In particular, the training error will decrease as more variables are included in the model, but the test error may not. Therefore, we have to carefully design the measurements for subset selection.

We can directly estimate the test error, using either a validation approach or a cross-validation approach. Please check the cross-validation documents for detail. Here, we discuss how to estimate the test error directly by making an *adjustment* to the training error to account for the bias due to overfitting. Here, we consider four approaches: C_p, *Akaike information criterion* (AIC), *Bayesian information criterion* (BIC), and *adjusted* R^2.

1. C_p. For a fitted least squares model containing d features, the C_p estimate of test MSE is computed using the equation $C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$, where $\hat{\sigma}^2$ is an estimate of the variance of the error associated with each response measurement for linear regression in Eq. (1). Typically, $\hat{\sigma}^2$ is an estimated using the full model containing all features. Essentially, the C_p statistic adds a penalty of $2d\hat{\sigma}^2$ to the training RSS in order to adjust for the fact that the training error tends to underestimate the test error. Clearly, the penalty increases as the number of features in the model increases; this is intended to adjust for the corresponding decrease in training RSS. If $\hat{\sigma}^2$ is an unbiased estimate σ^2, then C_p is an unbiased estimate of test MSE. As a consequence, the C_p statistic tends to take on a small value for models with a low test error, so when determining which of a set of models is best, we choose the model with the lowest C_p value.

2. AIC. The AIC criterion is defined for a large class of models fit by maximum likelihood. In the case of the linear regression with Gaussian errors, maximum likelihood and least squares are the same thing. In this case, AIC is given by $AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$, where, for simplicity, we have omitted an additional constant. Hence for least squares models, C_p and AIC are proportional to each other.

3. BIC. BIC is derived from a Bayesian point of view, but ends up looking similar to C_p (and AIC) as well. For the least squares model with d features, the BIC is, up to irrelevant constants, given by $BIC = \frac{1}{n\hat{\sigma}^2}(RSS + ln(n) \cdot d\hat{\sigma}^2)$. Like C_p, the BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value. Notice that BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $ln(n) \cdot d\hat{\sigma}^2$ term, where n is the number of observations. Since $ln(n) > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than C_p .

4. *adjusted* R^2. The usual R^2 is defined as $R^2 = 1 - \frac{RSS}{TSS}$, where $TSS = \sum_{i=1}^{n}(y^{(i)} - \bar{y})^2$ is the t*otal sum of squares.* Since RSS always decreases as more variables are added to the model, the R^2 always increases as more variables are added. For a least model with d variables, the adjust R^2 statistic is calculated as *adjusted* $R^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$. Unlike C_p, AIC, and BIC, for which a small value indicates a model with a low test error, a large value of adjusted R^2 indicates a model with a small test error. Maximizing the adjusted R^2 is equivalent to minimizing $\frac{RSS}{n-d-1}$. While RSS always decreases as the number of variables in the model increases, $\frac{RSS}{n-d-1}$ may increase or decrease, due to the presence of d in the denominator. The intuition behind the adjusted R^2 is that once all of the correct variables have been included in the model, adding additional noise variables will lead to only a very small decrease in RSS. Since adding noise variables leads to an increase in d, such variables will lead to an increase in $\frac{RSS}{n-d-1}$,

and consequently a decrease in the adjusted R^2. Therefore, in theory, the model with the largest adjusted R^2 will have only correct variables and no noise variables. Unlike the R^2 statistic, the adjusted R^2 statistic pays a price for the inclusion of unnecessary variables in the model.

2.1.2 Best Subset Selection

To perform *best subset selection*, we fit a separate least square coefficient estimate for each possible combination of the p features. We then look at all of the resulting models, with the goal of identifying the one that is best. The whole algorithm can be described as follow.

Algorithm 3 Best Subset Selection
/* Input:
* a dataset with p features
* Output:
* a subset of features with size $k \leq p$
*/
1: Let M_0 denote the *null model*, which contains no features. // This model simply predicts the sample mean for each observation.
2: For $k$from 1 to p
3: Fit all $C_p^k = \begin{pmatrix} p \\ k \end{pmatrix}$ models that contains exactly k features.
4: Pick the best among those $C_p^k = \begin{pmatrix} p \\ k \end{pmatrix}$ models, and call it M_k. //Note that "best" is defined as having the smallest RSS or MSE.
5: End For
6: Select a single model among $M_0, M_1, ..., M_p$ with minimum cross-validated test error, or minimum C_p, or minimum AIC, or minimum BIC, or maximum adjusted R^2.

Since the algorithm will consider all combinations of features, the time complexity for Algorithm 3 is $O(2^p)$ which is not computable when p is large. Therefore, we will present computationally efficient alternatives to best subset selection in the next subsection.

2.1.3 Stepwise Selection

For computational reasons, best subset selection cannot be applied with very large p. Best subset selection may also suffer from statistical problems when p is large. The larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data. Thus an enormous search space can lead to overfitting and high variance of the coefficient estimates. For both of these reasons, stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

The first alternative is forward stepwise selection. Forward stepwise selection begins with a model containing no features, and then adds features to the model, one-at-a-time, until all of the features are in the model. In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model. The detail of the algorithm is shown in Algorithm 4.

Algorithm 4 Forward Stepwise Selection
/* Input:
* a dataset with p features
* Output:
* a subset of features with size $k \leq p$
*/
1: Let M_0 denote the *null model*, which contains no features. // This model simply predicts the sample mean for each observation.
2: For $k$from 0 to $p-1$
3: Consider all $p-k$ models that augment the features in M_k with one additional feature.
4: Choose the best among those $p-k$ models, and call it M_{k+1}. //Note that "best" is defined as having the smallest RSS or MSE.
5: End For
6: Select a single model among $M_0, M_1, ..., M_p$ with minimum cross-validated test error, or minimum C_p, or minimum AIC, or minimum BIC, or maximum adjusted R^2.

In Algorithm 4, we test $\sum_{k=0}^{p-1}(p-k) = 1 + \frac{p(p+1)}{2}$ number of models, which tremendously decrease from 2^p models in Algorithm 3. Furthermore, forward stepwise selection can be applied even in the high-dimensional setting where $n < p$; however, in this case, it is possible to construct sub-models $M_0, M_1, ..., M_{n-1}$ only, since each sub-model is fit using least squares, which will not yield a unique solution if $p \geq n$.

The second alternative is backward stepwise selection. Backward stepwise selection begins with the full least squares model containing all p features, and then iteratively removes the least useful feature one-at-a-time. Details are given in Algorithm 5.

Like forward stepwise selection, the backward selection approach searches through only $\sum_{k=0}^{p-1}(p-k) = 1 + \frac{p(p+1)}{2}$ models, and so can be applied in settings where p is too large to apply best subset selection. However, the backward selection requires that the number of samples n is larger than the number of features p.

The best subset, forward stepwise, and backward stepwise selection approaches generally give similar but not identical models. As another alternative, hybrid versions of forward and backward stepwise selection are available, in which variables are added to the model sequentially, in analogy to forward selection. However, after adding each new variable, the method may also remove any variables that no longer provide an improvement in the model fit. Such an approach attempts to more closely mimic best subset selection while retaining

Algorithm 5 Backward Stepwise Selection
/* Input:
* a dataset with p features
* Output:
* a subset of features with size $k \leq p$
*/
1: Let M_p denote the *full model*, which contains all features.
2: For $k$from p to 1
3: Consider all k models that contain all but one of the features in M_k for a total $k-1$ features.
4: Choose the best among those k models, and call it M_{k-1}. //Note that "best" is defined as having the smallest RSS or MSE.
5: End For
6: Select a single model among $M_0, M_1, ..., M_p$ with minimum cross-validated test error, or minimum C_p, or minimum AIC, or minimum BIC, or maximum adjusted R^2.

the computational advantages of forward and backward stepwise selection.

2.2 Shrinkage Methods

The subset selection methods described in the above subsection involve using least squares to fit a linear model that contains a subset of the features. As an alternative, we can fit a model containing all p predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero. It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance. The two best-known techniques for shrinking the regression coefficients towards zero are *ridge regression* and *the lasso*.

2.2.1 Ridge Regression

Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates $\theta_0, \theta_1, ..., \theta_k$ are the values that minimize

$$\sum_{i=1}^{n}[y^{(i)} - (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_k x_k^{(i)})]^2 + \lambda \sum_{j=1}^{k} \theta_j^2, \tag{4}$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately. Eq. (4) trades off two different criteria. As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small. However, the second term, $\lambda \sum_{j=1}^{k} \theta_j^2$, called a *shrinkage penalty*, is small when $\theta_1, \theta_2, ..., \theta_k$ are close to zero, and so it has the effect of shrinking the estimate of θ_j toward zero. The tuning parameter λ serves to control the relative impact of these two

terms on the regression coefficient estimates. When $\lambda = 0$, the penalty term has no effect, and ridge regression will produce the least squares estimates. However, as $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero. Unlike least squares, which generates only one set of coefficient estimates, ridge regression will produce a different set of coefficient estimates, $\{\theta_1, \theta_2, ..., \theta_k\}_\lambda$, for each value of λ. Selecting a good value for λ is critical and can be determined through cross-validation.

Note that in Eq. (4), the shrinkage penalty is applied to $\theta_1, \theta_2, ..., \theta_k$, but not to the intercept θ_0 . We want to shrink the estimated association of each variable with the response; however, we do not want to shrink the intercept, which is simply a measure of the mean value of the response when $x_1^{(i)} = x_2^{(i)} = ... = x_k^{(i)} = 0$. If we assume that the variables have been centered to have mean zero before ridge regression is performed, then the estimated intercept will take the form $\theta_0 = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}$.

Here, we discuss the reason that ridge regression improves over least squares. Ridge regression's advantage over least squares is rooted in the bias-variance trade-off. As λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. In general, in situations where the relationship between the results and the features is close to linear, the least squares estimates will have low bias but may have high variance. This means that a small change in the training data can cause a large change in the least squares coefficient estimates. In particular, when the number of variables p is almost as large as the number of observations n, the least squares estimates will be extremely variable. And if $p > n$, then the least squares estimates do not even have a unique solution, whereas ridge regression can still perform well by trading off a small increase in bias for a large decrease in variance. Hence, ridge regression works best in situations where the least squares estimates have high variance.

2.2.2 The Lasso

Ridge regression does have one obvious disadvantage. Unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all features in the final model. The penalty $\lambda \sum_{j=1}^{k} \theta_j^2$ in Eq. (4) will shrink all of the coefficients towards zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$). This may not be a problem for prediction accuracy, but it can create a challenge in model interpretation in settings in which the number of variables is quite large.

The lasso is a relatively recent alternative to ridge regression that overcomes the above disadvantage. The lasso estimates $\theta_0, \theta_1, ..., \theta_k$ are the values that minimize

$$\sum_{i=1}^{n}[y^{(i)} - (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_k x_k^{(i)})]^2 + \lambda \sum_{j=1}^{k} |\theta_j| \,, \tag{5}$$

Comparing Eq. (5) to Eq. (4), we see that the lasso and ridge regression

have similar formulations. The only difference us that the θ_j^2 term in the ridge regression penalty has been replaced by $|\theta_j|$ in the lasso penalty.

As with ridge regression, the lasso shrinks the coefficient estimates towards zero. However, in the case of the lasso, the penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large. Hence, much like subset selection, the lasso performs *feature selection*. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression. We say that the lasso yields *sparse* models—that is, models that involve only a subset of features.