Data Science Fundamentals

Anhthy Ngo

December 22, 2020

Contents

1 Overview

There are two equally problematic cases which can arise when learning a classifier on a data set: underfitting and overfitting.

1. **Underfitting:** The classifier used to build prediction is very simple and is not able to learn complex patterns from the training data. In this case, both the training error and test error will be high, as the classifier does not account for the relevant information present in the training set.
2. **Overfitting:** The classifier fits and learns the noise or random fluctuations in the training data too well. This is problematic because this noise may not apply to new unseen data and negatively impacts the classifiers ability to generalize. Although the training error continues to decrease over time, test error will begin to increase as the classifier begins to make decisions based on patterns which exist only in the training set and not in the broader distribution.

Recall empirical risk minimization:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{l_{(s)}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer}$$

where the loss function is a continuous function which penalizes training error, and the regularizer is a continuous function which penalizes classifier complexity.

Examples of some loss functions include:

Table 1: Loss functions

	Function	Usage
Squared Loss	$(h(\mathbf{x}_i) - y_i)^2$	Linear Regression
Absolute Loss	$\lvert h(\mathbf{x}_i) - y_i \rvert$	Linear Regression
Hinge Loss	$\max[1 - h_{\mathbf{w}}(\mathbf{x}_i) y_i, 0]$	Standard SVM
Log Loss	$\log(1 + e^{-h_{\mathbf{w}}(\mathbf{x}_i) y_i})$	Logistic Regression
Exponential Loss	$e^{-h_{\mathbf{w}}(\mathbf{x}_i) y_i}$	Adaboost

2 Bias-variance tradeoff

We use the "bias-variance tradeoff" to describe the performance of a model. We often say that "high variance" is proportional to overfitting, and "high bias" is proportional to underfitting.

Let us define bias and variance. To use for formal terms for bias and variance, assume we have a point estimator $\hat{\theta}$ of some parameter or function θ. Then, the bias is defined as:

$$\textbf{Bias} = \mathbb{E}[\hat{\theta}] - \theta$$

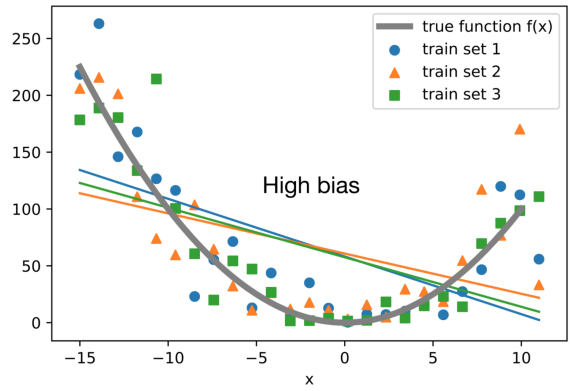
If Bias > 0, we say the estimator is positively biased, if Bias > 0, the estimator is negatively biased, and if Bias $= 0$, the estimator is unbiased.

Similarly, we define variance as:

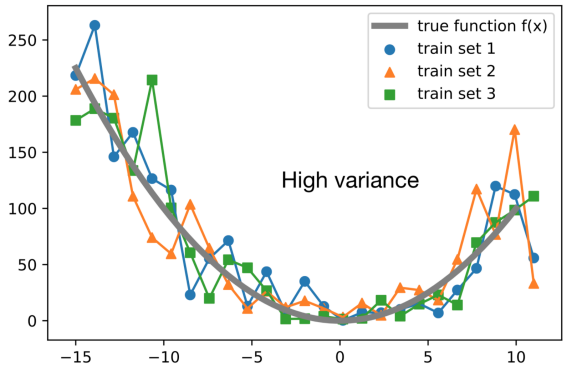
$$\begin{aligned}\textbf{Var}(\hat{\theta}) &= \mathbb{E}[\hat{\theta}^2] - (\mathbb{E}[\hat{\theta}])^2 \\ &= \mathbb{E}\big[(\mathbb{E}[\hat{\theta}] - \hat{\theta})^2\big]\end{aligned}$$

In Figure 1a, we can see different linear regression models fit to a training sets drawn from an unknown distribution. We can see the bias is large because the difference between the true value and predicted value on average (expectation of training sets) is large.

From Figure 1b, we can see several unpruned decision tree models fit. If we took the expectation over the training sets, the average hypothesis would fit the true function perfectly (given the noise is unbiased and has an expected value of 0). We have high variance through because on average, a prediction differs a lot from the expectation value of the prediction.



(a) High bias



(b) High variance

Figure 1: Depiction of high bias in regression setting (a) and high variance in decision tree setting (b)

Linear models often have high bias but low variance. Nonlinear machine learning algorithms often have a low bias but a high variance.

For example, in practice:

1. **Linear Models:**

$$\textbf{Linear Cost function} = \min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer}$$

   In linear models, we use regularization to parametrically control the model complexity, more specifically we penalize complex models. In doing so, we decrease variance with the expense of increasing bias. Regularization negates or minimizes the effect of predictor columns with large outliers by penalizing their regression coefficients. There are two main types of regularizers that are commonly used:: Ridge and Lasso.

2. **K-Nearest Neighbors**: increasing the value of k which increases the number of neighbors that contribute to the prediction will increase the bias and decrease the variance of the model. Conversely, decreasing k will increase variance and decrease bias. At very large k's, the boundary line becomes very smooth so there isn't much variance, but the lack of a match to the boundary line is a sign of high bias. At very small k's, the jaggedness of the boundary and islands are signs of variance. The locations of the islands and the exact curves of the boundaries will change radically as new data is gathered.

3. **Support Vector Machine:**

$$\textbf{SVM cost function} = \frac{1}{2}||w||^2 + \frac{C}{n} \sum_{i=1}^{n} (1 - y_i[w^T x_i + b])_+$$

   This algorithm has low bias and high variance, but the trade-off can be changed by increasing the C parameter that influences the number of violations of the margin allowed in the training data. A large C gives lower bias and higher variance because of high penalties for misclassification which forces the algorithm to explain the input data stricter and potentially overfit. A smaller C, gives higher bias and lower variance because the cost of misclassification is low, allowing for more mistakes for the sake of wider "cushion".

4. **Decision Tree:** This algorithm has low bias and high variance. The tree makes almost no assumptions about target function but it is highly susceptible to variance in data. In decision trees, pruning of tree is a method to reduce variance. Pruning reduces size of trees by removing sections of the tree that provide little power to classify instances (e.g. information gain). Pruning reduces complexity of final classifier, and hence improves predictive accuracy by the reduction of overfitting.

2.1 Detecting high bias and variance

If a classifier is under-performing, (e.g. if the test or training error is too high), there are several ways to improve performance. We can detect the root of the problem by looking at two overarching situations:

1. **Situation 1:** The training error is below the desired threshold (ϵ), but the test error is significantly higher
   Symptoms: Training error is much lower than test error, training error is much lower than ϵ.
   Remedies: Adding more training data, reduce model complexity – complex models are prone to high variance, bagging
2. **Situation 2:** The test error is remarkably close to training error, but both are above the desired tolerance of ϵ
   Symptoms: Training error is higher than ϵ
   Remedies: Use more complex model (e.g. kernelize, use non-linear models), add features, boosting

2.2 How do decrease variance of our models?

1. **Bootstrap Sampling**: Given set $\mathcal{D}$ containing n training examples, create $\mathcal{D}^{'}$ by drawing n examples at random with replacement from $\mathcal{D}$.

   Bagging: The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance.

   (a) Create k bootstrap examples $\mathcal{D}_1, \ldots, \mathcal{D}_k$
   (b) Train distinct classifier $\mathcal{H}_i$ on each $\mathcal{D}_i$
   (c) Classify new instance by majority vote/average

   If each single classifier is unstable – that is, it has high variance, the aggregated classifier $\bar{\mathcal{H}}$ has a smaller variance than a single original classifier. Random forests use bagging and by creating many of these trees, in effect a "forest", and then averaging them the variance of the final model can be greatly reduced over that of a single tree. Bagging, however, has little effect on bias.

2. **Max-Depth of Decision Tree**: A decision tree can have high variance because, if you imagine a very large tree (max-depth is high), it can basically adjust its predictions to every single input. If you make a small change to one of the predictors, a decision tree might give you a completely different answer. Setting max-depth lower (higher) will decrease (increase) variance.

3. **Increase Dataset Size** The law of large numbers states that as the size of a sample is increased, the more accurate of an estimate the sample mean will be of the population mean. This is perhaps the simplest approach to reduce the model variance.

4. **Dimensionality reduction** and **feature selection** can decrease variance by simplifying models.

2.3 Bias-variance decomposition for Squared Loss

We can decompose the loss function such as squared loss (MSE) into three terms: **bias, variance**, and **noise**. We ignore the noise term for simplicity.

The MSE of an estimator $\hat{\theta}$ with respect to an unknown parameter θ is defined as:

$$\text{MSE}(\hat{\theta}) = \text{E}_\theta \left[(\hat{\theta} - \theta)^2 \right]$$

The squared error loss decomposition into bias and variance starts with some algebraic manipulation and using the quadratic formula $(a+b)^2 = a^2 + b^2 + 2ab$. Also note that $\mathbb{E}_\theta[\hat{\theta}] - \theta$ is a constant.

$$\begin{aligned}
\text{MSE}(\hat{\theta}) &= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \theta \Big)^2 \right] \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] + \mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \right] \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big)^2 + 2 \Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big) \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big) + \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \right] \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big)^2 \right] + \mathbb{E}_\theta \left[2 \Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big) \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big) \right] + \mathbb{E}_\theta \left[\Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \right] \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big)^2 \right] + 2 \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big) \mathbb{E}_\theta \Big[\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big] + \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big)^2 \right] + 2 \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big) \Big(\mathbb{E}_\theta[\hat{\theta}] - \mathbb{E}_\theta[\hat{\theta}] \Big) + \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \\
&= \mathbb{E}_\theta \left[\Big(\hat{\theta} - \mathbb{E}_\theta[\hat{\theta}] \Big)^2 \right] + \Big(\mathbb{E}_\theta[\hat{\theta}] - \theta \Big)^2 \\
&= \mathbf{Var}_\theta(\hat{\theta}) + \mathbf{Bias}(\hat{\theta}, \theta)^2
\end{aligned}$$

3 Regularization

The basic idea of regularization is penalize the complexity of a model, or equivalently to encourage the model to be as simple as possible. We do this with the tuning parameter λ, which penalizes large coefficients.

Regularization reduces overfitting. Why? Regularization targets the biggest coefficients which might be overstating the case for their respective variables. The penalty generally prevents the model from placing "too much" importance of any one variable because the large coefficients are penalized more harshly than smaller ones. Imagine: $275,238x_1^2 - 543,845x_1x_2 + 385,832x_2^2$ is best fit for well-spaced points all with $|y| < 1$. A small change in x willl result in a big change in y. This is why we need to penalize large weights. The high weights signal sensitivity, in other words, the model was defined to fit one state and may not fit to another state due to the models high sensitivity. By reducing weights, overfitting is avoided and model generalization is improved.

Table 2: Regularizers

	Regularizer	$r(w)$
Lasso Regression	ℓ_1-Regularization	$\|w\|_1$
Ridge Regression	ℓ_2-Regularization	$w^T w = \|w\|_2^2$

How do we choose λ? We want to choose the λ that minimizes our cost function (i.e., MSE). To choose λ we use validation data, separate from the training and test data, to evaluate the error of the model for different values of λ and select the best value (K-fold cross-validation).

Cross-validation procedure:

1. Partition training data $\{1, \ldots, n\}$ into K subsets (i.e., folds) of roughly equal size (in practice $K \in [5, 10]$)
$$F = (F_1, F_2, \ldots, F_K)$$
2. For each $k = 1, \ldots, K$:
   - Train $(x_i, y_i), i \notin F_k$, and validate on $(x_i, y_i), i \in F_k$
   - For each value of the tuning parameter $\lambda = \in \{\lambda_1, \ldots, \lambda_m\}$, compute the estimate $\hat{f}_\lambda^{-k}$ on the training set and record the total error on the validation set:
$$e_k(\lambda) = \sum_{i \in F_K} \left(y_i - \hat{f}_\lambda^{-k}(x_i)\right)^2$$

3. For each tuning parameter value λ, compute the average error over all folds,

$$\mathrm{CV}(\lambda) = \frac{1}{n}\sum_{k=1}^{K} e_k(\lambda) = \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in F_K}\left(y_i - \hat{f}_\lambda^{-k}(x_i)\right)^2$$

4. Having done this, we get a cross-validation error curve $\mathrm{CV}(\theta)$, which we can use to choose the value of λ that minimizes this curve

$$\lambda^* = \underset{\lambda\in\{\lambda_1,\ldots,\lambda_m\}}{\arg\min}\ \mathrm{CV}(\lambda)$$

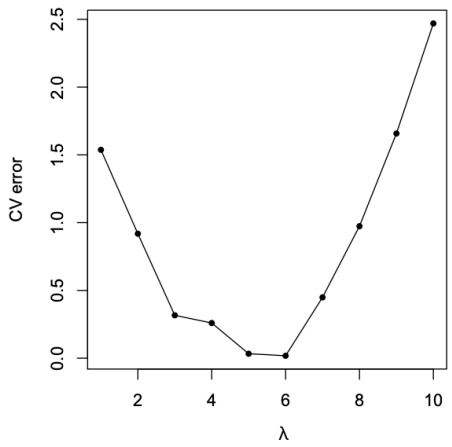


Figure 2: Cross validation errors for $\mathrm{CV}(\lambda)$

5. Compute the chosen model $\hat{f}_{\lambda^*}^{-k}$ on the entire training set

6. Apply $\hat{f}_{\lambda^*}^{-k}$ to the test set to assess the test error

Remark: What if our data set is too small to set aside a test set? Then we let all available data be our training set, apply K-fold cross-validation, still choose λ^* as the minimizer of the CV error, and then refit the model with λ^* on the entire training set.

3.1 Ridge Regression

Ridge regression (ℓ_2 regularization) tries to penalize coefficients together by shrinking the estimated coefficients towards zero simultaneously. Ridge regression introduces some bias, but it can greatly reduce the variance, resulting in a better MSE.

$$\begin{aligned}\hat{\beta}^{\text{ridge}} &= \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \\ &= \arg\min_{\beta} \|y - \mathbf{X}^T\beta\|_2^2 + \lambda\|\beta\|_2^2\end{aligned}$$

Here $\lambda \geq 0$ is a tuning parameter, which controls the strength of the penalty term. Note that:

- When $\lambda \to 0$, then $\hat{\beta}^{\text{ridge}} \to \hat{\beta}^{\text{OLS}}$
- When $\lambda \to \infty$, then $\hat{\beta}^{\text{ridge}} \to 0$
- For λ in between, we are balancing two ideas: fitting a linear model of y on X, and shrinking the coefficients.

The general trend is:

- As $\lambda \to \infty$, variance $\to 0$, but bias increases
- As $\lambda \to 0$, bias $\to 0$, but variance increases

Solving the Ridge Solution:

1. First we rewrite and FOIL $\hat{\beta}^{\text{ridge}}$

$$\begin{aligned}\hat{\beta}^{\text{ridge}} &= \arg\min_{\beta}\left[\|y - \mathbf{X}^T\beta\|_2^2 + \lambda\|\beta\|_2^2\right] \\ &= \arg\min_{\beta}\left[(y - \mathbf{X}^T\beta)^T(y - \mathbf{X}^T\beta) + \lambda\beta^T\beta\right] \\ &= \arg\min_{\beta}\left[y^Ty - 2\beta^T\mathbf{X}^Ty + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda\beta^T\beta\right]\end{aligned}$$

2. Now we solve $\hat{\beta}^{\text{ridge}}$ by taking the derivative and setting it to 0.

$$\begin{aligned}\nabla\hat{\beta}^{\text{ridge}} &= 0 \\ -2\mathbf{X}^Ty + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta &= 0 \\ -\mathbf{X}^Ty + \mathbf{X}^T\mathbf{X}\beta + \lambda\beta &= 0 \\ \mathbf{X}^T\mathbf{X}\beta + \lambda\beta &= \mathbf{X}^Ty \\ (\mathbf{X}^T\mathbf{X} + \lambda I)\beta &= \mathbf{X}^Ty \\ \hat{\beta}^{\text{ridge}} &= (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^Ty\end{aligned}$$

The ridge solution is unique. Why? Let $\mathbf{X}$ be a $n \times d$ matrix. In ordinary least squares, when $d > n$, $\mathbf{X}^T\mathbf{X}$ is not invertible. When $\lambda = 0$, (ordinary least squares), the condition that $d > n$ causes the sample covariance matrix $\mathbf{X}^T\mathbf{X}$ to be rank deficient and thus cannot be inverted to yield a unique solution. This is why there can be an infinitude of solutions to the ordinary least squares problem when $d > n$. However, when $\lambda > 0$ (i.e., when ridge regression is used), the addition of λI to the sample covariance matrix ensures positive definiteness and linear independence. In other words, the matrix becomes invertible, and the solution becomes unique.

Positive Definite A matrix is positive definite if $z^T\mathbf{A}z > 0$ for all vectors $z \neq 0$. If $\mathbf{A}$ is positive definite, then all of its eigenvalues are positive, so 0 is not an eigenvalue of $\mathbf{A}$. Therefore, the system of equations $\mathbf{A}x = 0$ has no non-trivial solution, and so A is invertible.

Proof If $\lambda > 0$, then

$$z^T(\mathbf{X}^T\mathbf{X} + \lambda I)z = z^T\mathbf{X}^T\mathbf{X}z + \lambda z^T z = \|\mathbf{X}z\| + \lambda\|z\| \geq \lambda\|z\| \geq 0$$

because $\lambda > 0$; and also $\lambda\|z\|$ is 0 only if $z = 0$. So $\mathbf{X}^T\mathbf{X} + \lambda I$ is positive definite, and therefore invertible.

Ridge Regression in Practice

1. In contrast to ordinary least squares, Ridge regression is highly affected by the scale of the predictors. Therefore it is better to standardize the predictors before applying Ridge regression.
2. Ridge regression will never set coefficients to exactly zero and therefore cannot perform variable selection in the linear model. While this doesn't seem to hurt its prediction ability, it is not desirable for the purposes of interpretation (especially if the number of variables p is large)

3.2 Lasso Regression

Lasso (least absolute shrinkage and selection operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accu-

racy and interpretability of the statistical model it produces.

$$\begin{aligned}\hat{\beta}^{\text{lasso}} &= \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= \arg\min_{\beta} \underbrace{\|y - \mathbf{X}^T\beta\|_2^2}_{\text{Loss}} + \underbrace{\lambda\|\beta\|_1}_{\text{Penalty}}\end{aligned}$$

Here $\lambda \geq 0$ is a tuning parameter, which controls the strength of the penalty term. Note that:

- When $\lambda \to 0$, then $\hat{\beta}^{\text{lasso}} \to \hat{\beta}^{\text{OLS}}$
- When $\lambda \to \infty$, then $\hat{\beta}^{\text{lasso}} \to 0$
- For λ in between, we are balancing two ideas: fitting a linear model of y on X, and shrinking the coefficients. But the nature of the ℓ_1 penalty causes some coefficients to be shrunken to zero exactly. As λ increases, more coefficients are set to zero.

The general trend is:

- As $\lambda \to \infty$, variance $\to 0$, but bias increases
- As $\lambda \to 0$, bias $\to 0$, but variance increases

Lasso is advantageous because it performs **feature selection** and is **interpretable**. Lasso is useful for model explainability because there can be much fewer features to work with.

3.3 Comparing Ridge and LASSO

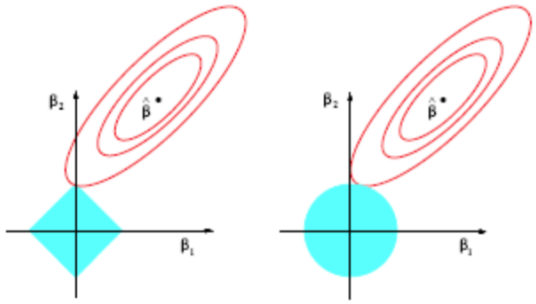


Figure 3: Estimation picture for the lasso (left) and ridge regression (right). Shown are the contours of the error (MSE) and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the MSE error function.

Figure 3 displays the comparison of the Lasso (left) and the Ridge Regression (right) constraints. The red ellipses each represent different MSEs from adjusting β_1 and β_2. The MSE is minimized at the black point and it increases quadratically as we move away from it. The solid blue isosurfaces are the feasible regions which are minimized at the origin (where $\beta_1 = \beta_2 = 0$) and also increases quadratically as we move away from origin. Heuristically, for each method, we are looking for the intersection of the red ellipses and the blue region as the objective is to minimize the error function while maintaining the feasibility.

With the decrease of t, both methods get smaller β_1 and β_2. Here comes the difference between Lasso and Ridge: the tangent point for Lasso cases is more likely to be achieved on the axis (some parameter be reduced to zero), while the tangent point for Ridge cases is more likely to be got on a non-axis point (non-zero points). Observe that the ℓ_1 ball has sharp corners: $\hat{\beta}^{\text{lasso}}$ is therefore very likely to be at one of these corners which is a point where many coordinates are set to 0. Hence, the ℓ_1 penalization tends to produce sparse solutions.

Ridge vs Lasso in practice

- Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (ergo: when only a few predictors actually influence the response)

- Ridge works well if there are many large parameters of about the same value (ergo: when most predictors impact the response).
- When features have high collinearity, Ridge regression often performs better than Lasso.

4 Multicollinearity

Recall the formula for the estimated coefficients in a multiple linear regression:

$$\hat{\beta}^{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$$

Issues arise when $\mathbf{X}^T\mathbf{X}$ isn't invertible. There are several equivalent conditions for any square matrix $\mathbf{U}$ to be singular or non-invertible:

1. The determinant det $\mathbf{U}$ (or $|\mathbf{U}|$) is 0
2. At least one eigenvalue of $\mathbf{U}$ is 0 (This is because the determinant of a matrix is the product of its eigenvalues)
3. $\mathbf{U}$ is **rank deficient**, meaning that one or more of its columns (or rows) is equal to a linear combination of the other rows

Since we're not concerned with any old square matrix, but specifically with $\mathbf{X}^T\mathbf{X}$, we have an additional equivalent condition:

1. $\mathbf{X}$ is **column-rank** deficient, meaning one or more of its columns is equal to a linear combination of the others

The last condition explains multicollinearity: we have p different predictor variables, but some are linear combinations of the others, and thus no new information is gained. In other words, there is a near-linear relationship (correlation) among predictors. Some examples of correlation include using the same information twice (weight in lb and weight in kg) and the dummy variable trap. If the degree of correlation between predictors is high enough, it can cause problems when you fit the model and interpret the results.

4.1 Why is multicollinearity a problem?

The idea of regression analysis is that you can change the value of one predictor and not the others. However, when predictors are correlated, it indicates that changes in one predictor are

associated with shifts in another predictor. The stronger the correlation, the more difficult it is to change one predictor without changing another. Thus, it becomes difficult for the model to estimate the relationship between each predictor and the response independently because the predictors tend to change in unison.

1. Regression coefficients are very sensitive to minor changes in model (when variables are added/dropped).
2. Multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical power of your regression model. Thus, the variances and the standard errors of the regression coefficient estimates will increase. You might not be able to trust the p-values to identify independent variables that are statistically significant.

4.2 How do we detect multicollinearity?

To detect collinearity (among two predictors) we normally check the following:

1. **High Correlation Coefficients:** If the correlation > 0.8 then severe multicollinearity may be present.
2. **Volatile Coefficients:** Coefficient estimates vary excessively from model to model.
3. **Conflicting Significance:** The t-tests for each of the individual slopes are non-significant ($P > 0.05$), but the overall F-test for testing all of the slopes are simultaneously 0 is significant ($P < 0.05$).

Looking at correlations only among pairs of predictors, however, is limiting and a multicollinear relationship involving three or more variables might be totally invisible on a pairs plot. This is why we use **variance inflation factors (VIF)** to help detect multicollinearity.

VIF quantifies how much variance is inflated. Recall that we learned previously that the standard errors, and hence the variances, of the estimated coefficients are inflated when multicollinearity exists. A VIF exists for each of the predictors in a multiple regression model. For example, the variance inflation factor for the estimated regression coefficient β_j, denoted VIF_j, is just the factor by which the variance of β_j is "inflated" by the existence of correlation among the predictor variables in the model.

In particular, the variance inflation factor for the jth predictor is:

$$\text{VIF}_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the R^2-value obtained by regressing the jth predictor on the remaining predictors.

A VIF of 1 means that there is no correlation among the jth predictor and the remaining predictor variables, and hence the variance of β_j is not inflated at all. The general rule of thumb is that VIFs exceeding 4 warrant further investigation, while VIFs exceeding 10 are signs of serious multicollinearity requiring correction.

4.3 Fixing Multicollinearity

There are a few things we can do to remediate multicollinearity:

1. **Do Nothing:** Leave the model as is, despite multicollinearity. The presence of multicollinearity doesn't affect the efficiency of extrapolating the fitted model to new data provided that the predictor variables follow the same pattern of multicollinearity in the new data as in the data on which the regression model is based.
2. **Drop Redundant Variable:** Since not all of the p predictors are actually contributing information, a natural way of dealing with multicollinearity is to drop some variables from the model.
3. **Increase Sample Size:** Increasing the sample size improves the precision of an estimator and reduces the adverse effects of multicollinearity. Usually adding data though is not feasible.
4. **Perform Ridge Regression:** The real problem with collinearity is that when it happens, there isn't a unique solution for $\hat{\beta}^{\text{OLS}}$. This causes the variance of $\hat{\beta}^{\text{OLS}}$ to be infinite. With Ridge regression there is a unique solution.

References

[1] CS4870 Machine Learning Lecture 11: Model Selection
https://www.cs.cornell.edu/courses/cs4780/2018sp/lectures/lecturenote11.html

[2] Overfitting and Underfitting Blog
https://medium.com/@itbodhi/overfitting-and-underfitting-in-machine-learning-models-76cb60dbdaf6

[3] Bias-Variance Decomposition Blog
http://rasbt.github.io/mlxtend/user_guide/evaluate/bias_variance_decomp/

[4] Scott Fortmann. Bias-Variance Trade-Off.
http://scott.fortmann-roe.com/docs/BiasVariance.html

[5] Ryan Tibshirani. Model selection and validation 1: Cross-validation. March 26, 2013.
https://www.stat.cmu.edu/~ryantibs/datamining/lectures/18-val1.pdf

[6] Jim Forst. Multicollinearity in Regression Analysis: Problems, Detection, and Solutions.
https://statisticsbyjim.com/regression/multicollinearity-in-regression-\analysis/

[7] Larry Wasserman. STAT401. Lecture 17: Multicollinearity
http://www.stat.cmu.edu/~larry/=stat401/lecture-17.pdf

[8] STAT 462. Lecture 10.7: Detecting Multicollinearity Using Variance Inflation Factors
https://online.stat.psu.edu/stat462/node/77/