Contents

1 Introduction

Multiple linear regression is widely used in many applications to model linear relationships between a set of predictors and a response. One of the more common and simpler methods of fitting such a model is known as the ordinary least squares method (OLS), which is found to be a Best Linear Unbiased Estimator (BLUE) by the Gauss-Markov theorem, that is, it gives the unbiased estimates of each regression coefficient with the lowest possible variance, provided that assumptions are satisfied. Furthermore, it has a main advantage of it being easily interpretable as a parametric model as opposed to non-parametric models.

The motivation for this project arises from one of these assumptions, that is, the data matrix, $\boldsymbol{X}$, is full rank, i.e. none of the predictors are (nearly) perfectly correlated with each other. In cases where this is not true, perfect multicollinearity is said to exist, and OLS is unable to estimate regression parameters. In such cases, there are several methods we can employ to address this issue, whether to drop one of the correlated variables, or to combine them, etc., based on the discretion of the data analyst. In this project, we investigate how two of the commonly used software for regression analysis, Python (`scikit_learn` and `statsmodel`) and R each handles perfect collinearity when fitting the model.

Furthermore, we embrace this opportunity to explore further on the topic of multicollinearity, that is, when two or more predictors are correlated to one another.[5] We first examine its impact on model prediction and interpretation. We then explore several known methods of detecting collinearity in the data, whether through hypothesis testing, or by using measures such as the variance inflation factor (VIF) and condition numbers/indices. In addition to aforementioned methods of addressing collinearity in the previous paragraph, we consider shrinkage methods such as ridge regression and lasso to alleviate the impact of overfitting (due to collinearity) on variance at the cost of negligible bias. We also briefly explore another potential pitfall in regression analysis, that is confounding variables, by examining examples of Simpson's paradox.

Here, we lay down the general setup for fitting a linear regression model. Given an $n \times p$ data matrix $\boldsymbol{X}$ (n samples and p predictors), corresponding to an $n \times 1$ column matrix $\boldsymbol{y}$ containing the observed response, we model the relationship between p predictors, X_i and the response variable, Y as

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon, \tag{1}$$

where ε represents the random error. We fit the model to the data collected for the expected response $E[Y]$ by estimating each β_i using the least squares method to obtain $\hat{\beta}_i$ so that we have

$$E[Y] = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i X_i. \tag{2}$$

Finally, it should be noted that there are different conventions to what collinearity and multicollinearity mean according to varying sources. In this report, we will use these two terms interchangably to refer to the correlation between two or more predictors in a regression model.

2 Objectives

There are three main objectives for this project:

1. Explore issues that arise from singularity.
2. Understand singular value decomposition (SVD).
3. Investigate how software deals with singularity when building linear models.

In practical settings, it is unlikely to obtain a set of perfectly correlated data due to inevitable noise. However, even near perfect collinearity leads to singularity when attempting to fit the regression model using software. We hence aim to explore how a rank-deficient data matrix can lead to singularity and the issues that arise from it. We also explore pitfalls caused by moderate to high multicollinearity in terms of interpretation and prediction. Subsequently, we look into commonly used methods to handle situations where regressors are highly correlated depending on purpose of analysis.

The next objective is to understand how to apply the singular value decomposition (SVD) in the computations of the least square estimates. The least squares method estimates the regression coefficients of a model with p predictors by fitting it to a dataset with n observations using the following equation

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{3}$$

where $\hat{\boldsymbol{\beta}}$ is a $p \times 1$ column matrix containing the least square estimates, $\boldsymbol{X}$ is a $n \times p$ data matrix, and $\boldsymbol{y}$ is a $n \times 1$ column matrix containing the observed response values. The derivation of this formula is included in the appendix (Section B.1). With the presence of perfect collinearity, the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is singular and hence $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ does not exist. However, the least squares solver of Python uses the Moore-Penrose inverse (or pseudoinverse) of $\boldsymbol{X}$, $\boldsymbol{X}^+$, which can be used to calculate the minimum norm approximation of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$, and can be computed using the SVD of $\boldsymbol{X}^T\boldsymbol{X}$ (see appendix Section B.2). A brief section in the appendix (Section A) is dedicated to the formulation and computation of the SVD.

Lastly, it is observed that there exists discrepancy between how different software handles highly correlated predictors when fitting a multiple linear regression model via OLS. In this project, we aim to understand the reason behind this difference in behaviour by referring to standard documentation and examining program output in Python and R. As an extension, methods for detecting and addressing multicollinearity using software are also explored.

3 Methodology

We narrowed our focus to three least squares solvers of interests: Python's `scikit-learn` library, Python's `statsmodel` library, and R's `lm` function. In each case, we randomly generated several sets of artificial data using three regressors where only one pair is correlated, each with 100 samples. The use of artificial data has the advantage where the true relationship between the response and predictors is known and self-determined, and that the degree of correlation and noise in the data can be manually specified. Note that the distribution used in data generation is irrelevant for our topic of study.

The data ranges from 0 to 100 and is uniformly distributed. The degree of collinearity was then varied between perfect and moderate to observe differences in output. The pseudoinverse of the data matrix was also computed and multiplied with the generated response to give the theoretical minimum norm solution for the coefficient estimates. Then, these results were compared with those given by the least squares solver function to confirm whether these functions use the pseudoinverse of the data matrix to estimate the least squares coefficients as suggested by documentation.

The true underlying relationship to which the artificial data was generated was set to be

$$Y = X_1 + 2X_2 + 3X_3, \tag{4}$$

where X_1 and X_2 are correlated with $X_2 = 2X_1$. We then vary the amount of uniformly distributed noise added to different generated sets: from no noise (perfect collinearity) to some number sampled from intervals [-25,25], [-50,50], and [-100,100]. The train-test ratio chosen for splitting the dataset into training and testing sets is fixed at 80-20. Lastly, to evaluate prediction performance, metrics such as the (adjusted) multiple coefficient of determination R^2 and mean squared error were used.

The scaling effect of `scikit_learn`'s `StandardScaler` on the ridge regression and LASSO models in the presence of collinearity was also given attention. This was done by referring to standard documentation as well as by testing it using artificially generated data. For data with and without high multicollinearity present respectively, two models were each fitted: one with predictors scaled by `StandardScaler` and one without. The output was then compared.

We have also performed regression analysis on California housing and applied methods to address collinearity using `statsmodel`.

3.1 Methods to detect collinearity

One simple way to detect collinearity is through a test for correlation, where we test the hypothesis

$$H_0 : \rho = 0 \qquad \text{vs.} \qquad H_1 : \rho \neq 0,$$

where ρ is the population correlation coefficient between a chosen pair of predictors. However, there are two main reasons to opt for an alternative method.

Firstly, as the number of predictors included in the model increases, the number of tests required to cover every combination of pairs increases. With large number of tests, the probability that we commit at least one error is high and hence we risk incorrectly identifying correlated pairs. More importantly, even if we find that every pair of predictors has no significant correlation, we cannot make the same conclusion about predictors amongst groups of other numbers.

3.1.1 Variance inflation factor (VIF)

A commonly used method for identifying highly correlated predictors is by using the variance inflation factor (VIF), given by

$$\text{VIF}_k = \frac{1}{1 - R_k^2}, \tag{5}$$

where R_k^2 is the multiple coefficient of determination of the k^{th} predictor regressed against the other predictors, assuming that it has zero mean. Intuitively, a higher value of R_k^2 indicates a good fit, hence a strong correlation between the k^{th} regressor and some of the other regressors. Note that uncorrelated regressors would simply have a coefficient close to zero, leading to a VIF of 1.[3]

The motivation behind the formulation of the VIF is to calculate the ratio between the variance of a coefficient estimate if it was uncorrelated with the other regressors (baseline case), and the actual variance. For example, a coefficient estimate having a VIF of 4 corresponds to its variance being 4 times of what it would have been if it was uncorrelated with every other predictor. In other words, VIF measures the factor by which the variance of the regression coefficients is inflated above what it would have been if $R_k^2 = 0$, hence its name.

There are different rule of thumbs for the thresholds beyond which one may consider the VIF value of a particular predictor to be high, ranging roughly from 4 to 10.[6] There is no definitively correct answer, and we have no choice but to make decisions based on our interpretation of the VIF value. Once predictors with high VIF's are identified, we can then examine the correlation matrix to identify which the predictors which each high VIF predictor is correlated to.

3.1.2 Condition numbers

Another metric for measuring the degreee of collinearity is the condition number (condition indices) of the data matrix $\boldsymbol{X}$, which indicates how sensitive the response is to changes in $\boldsymbol{X}$. Aside from using it to determine whether a model is highly collinear, the condition number of $\boldsymbol{X}^T\boldsymbol{X}$ is used to check for a potential source of large numerical error due to collinearity when computing $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ using software. In this case, it indicates the multiplicative factor to which round-off errors are amplified due to computer arithmetic being performed to finite precision.

The formula for the condition number of a $m \times n$ matrix $\boldsymbol{A}$ with rank $r \leq n$, denoted $\kappa(\boldsymbol{A})$, is given by

$$\kappa(\boldsymbol{A}) = \frac{\sigma_1}{\sigma_r}, \tag{6}$$

where σ_i are the singular values of $\boldsymbol{A}$ arranged in descending order. Note that definitions will differ for varying contexts, giving rise to different condition numbers. The point of discussion here is the condition number for inversion.

Similar to the case of VIF, there is no exact threshold for a condition number to be considered high, but a rule of thumb is any condition numbers exceeding 20 indicates the matrix to be ill-conditioned.

3.2 Methods to address collinearity

We present a few commonly-used methods to address moderate to perfect collinearity.

In the case of perfect collinearity, one of the correlated predictors is completely described by the other, hence our only choice is to remove one of them. This is however not recommended for non-perfect collinearity as we are at risk of committing omitted variable bias. In such cases, we may want to either combine some of the variables [3] or use shrinkage methods such as ridge regression or LASSO.

There are many possible ways of combining correlated variables, but in most cases the simple approach of adding the columns up (such as what R does) gives sufficiently improved results. Note that it is important that we consider the context when doing this, that is, whether combining the two variables make sense in context. An example is a model predicting the salary of employees based on predictors such as years of experience, gender, test scores for course A, and test scores for course B. Obviously, any employee who does well on one test is expected to do well in the other, and so we may consider adding the two test scores together to form a single column ”total test scores for course A and B”, then the results obtained may be interpreted accordingly.

For ridge regression and LASSO, otherwise known as L_2 and L_1 regularisation respectively, we find that it successfully decreases coefficient standard errors at the cost of increasing the bias (and hence decreasing R^2 score compared to the OLS model) by a neglibible amount. This is done by including a penalty term to the expression which we minimise to obtain the ridge and LASSO coefficient estimates, given by

$$\hat{\beta}_{\text{ridge}} = \arg\min_{\hat{\beta}} \left(\sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \|\hat{\beta}\|_2 \right) \tag{7}$$

and

$$\hat{\beta}_{\text{lasso}} = \arg\min_{\hat{\beta}} \left(\sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \|\hat{\beta}\|_1 \right) \tag{8}$$

respectively, where y_i is the i^{th} observed response, $\hat{y}_i$ is the i^{th} predicted response, λ is a parameter which determines the degree of regularisation. At $\lambda = 0$, we obtain the OLS model, as $\lambda \to \infty$, $\hat{\boldsymbol{\beta}} \to \mathbf{0}$, giving the null model. The penalty terms stabilises model coefficients in overfitted models by adding bias. As a result, by the bias-variance tradeoff, the variance of coefficient estimates decreases.

Before fitting the ridge or LASSO model, we standardise the predictors but applying a scaler as the ridge and LASSO coefficient estimates are not scale equivariant, contrary to OLS. Then we perform cross-validation on a range of values of λ to determine the optimal value of λ such that the test mean squared error is minimised. The range of values used is $\{0.01, 0.1, 1.0, 10.0, 100.0\}$.

4 Results and discussion

4.1 Python's `scikit-learn` library

After referring to documentation, we find that the least squares solver, `LinearRegression()` computes the coefficient estimates using the pseudoinverse of the data matrix, which we have verified. Under any degree of collinearity, we observed that model prediction is affected minimally, attributed by R^2 scores close to 1. However, the ridge regression or LASSO model fitted to data with perfect collinearity results in equal distributed coefficients between perfectly correlated predictors. After further investigation, we discovered that this was caused by `StandardScaler` from the same library which was used to scale predictors prior to fitting the model, which we show in Section B.4. Hence, this is unrelated to how `LinearRegression()` handles collinearity.

We also note that the solver does not give any warnings of high collinearity when displaying results and does not apply any methods to handle highly collinear data. The library is also not equipped with functions to calculate VIF of predictors. In this case, it is the analyst's own responsibility to check for collinearity using functions from the `statsmodel` library and decide on measures to address it manually.

4.2 Python's `statsmodel` library

As with `scikit_learn`'s `LinearRegression`, `statsmodel`'s `sm.OLS` computes the least squares coefficient by leveraging `NumPy`'s `numpy.linalg.pinv` function to calculate the pseudoinverse of the data matrix. It is hence no surprise that we obtain the exact same results for both least square solvers. One point of difference is that `statsmodel` warns us of high multicollinearity when we call `results.summary()`:

```
model_OLS=sm.OLS(y_train,X_train)
results_OLS=model_OLS.fit()
summarize(results_OLS)

# Output:
# ...[3] The smallest eigenvalue is 9.78e-26. This might indicate
    that there are strong multicollinearity problems or that the
   design matrix is singular.
```

4.3 R’s `lm` function

For R, the least squares solver `lm` combines perfectly collinear columns by adding them up. The first predictor replaced with the combined predictor while the other correlated variable(s) is dropped from the model. A portion of the output results is given below:

```
model <- lm(y~x1+x2+x3, data = df)
summary(model)

# Output:
# ...
# Coefficients: (1 not defined because of singularities)
# Estimate Std. Error    t value Pr(>|t|)
# (Intercept) -3.400e+01  3.716e-14 -9.150e+14   <2e-16 ***
# x1           5.000e+00  5.294e-16  9.444e+15   <2e-16 ***
# x2                  NA         NA         NA       NA
# x3           3.000e+00  5.587e-16  5.370e+15   <2e-16 ***
# ...
```

One important thing to note is that aside from the message ”1 not defined because of singularities”, R does not give any indications or warnings of high multicollinearity. It is all left to the data analyst to recognise this as a sign of collinearity and that measures were silently taken to eliminate the problem of singularity. We may avoid any careless situations by including an extra `singular.OK = FALSE` parameter in the function, which will give the following output when there is perfect collinearity.

```
model <- lm(y~x1+x2+x3, data = df, singular.ok = FALSE)

# Output:
# Error in lm.fit(x, y, offset = offset, singular.ok = singular.
   ok, ...) : singular fit encountered
```

However, for any degree of non-perfect collinearity, this warning will not be displayed, nor will it combine highly correlated predictors as with perfect correlated predictors. The data analyst must hence be aware of such a pitfall when interpreting the regression model and assess collinearity separately using the `vif` function from the `cars` package. We show an example performed on a generated dataset with noise in X_2 sampled from a continuous uniform distribution of $[-1, 1]$.

```
vif(model)

# Output:
#           x1           x2           x3
# 10218.852915 10220.421006     1.036177
```

4.4 Impact of multicollinearity

High multicollinearity mainly poses a major problem to model interpretation. Notably, it inflates the variance and covariances of least squares estimates and causes estimates and their variance to become sensitive to minor changes in data. As a result, we may obtain different results when performing variable selection after removing or adding samples from the dataset, or obtain incorrect signs of cofficients estimates [5]. Another curious effect is that the t-values of correlated predictors may be insignificant yet removing these predictors from the model results in a lower R^2 value. Although this seems contradictory by intuition, it is actually reasonable behaviour, as the prediction is contributed by both variables while the t-values indicate the contribution of one variable after accounting for the others.

When two predictors have perfect correlation, we have what is known as perfect collinearity. In such a case, at least one of the columns of the data matrix $\boldsymbol{X}$ is a linear combination of the other columns. This causes the matrix $\boldsymbol{X}^T\boldsymbol{X}$ to be singular (see appendix Section B.3), consequently causing $\hat{\boldsymbol{\beta}}$ unable to be computed (see Equation (3)). However, `statsmodel`'s and `scikit_learn`'s least square solvers compute the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ when estimating the regression coefficients, giving the least norm approximated solution (which is interpretable) when $\boldsymbol{X}$ is rank-deficient. However, the aforementioned impacts of high collinearity still apply hence it should be addressed.

It is important to note that collinearity does not, in any way, impact predictions or its precision. Hence, if the model is not expected to be interpreted, multicollinearity is no issue of concern and may be ignored. Moreover, even for more statistically-oriented analysts, there may also not be a need to address high collinearity provided that it does not inflate standard errors to a problematic extent [6], as there are many other factors which can suppress the inflation effect of standard errors, one of which is the sample size. Quoting the words of O'Brien, ”collinearity does not hurt as long as it does not bite”.[6]

4.5 Confounding variables

In addition to the pitfall of collinearity in performing regression analysis, confounding variables, or confounders, can also severely distort our results if not given attention [4]. The concept of confounders is illustrated using Simpson's paradox, where the relationship between individual predictors and the response disappears or reverses when all predictors are included in a single model [1]. The variables which cause such occurences are referred to as confounders.

We give the classic textbook example of the relationship between ice cream sales and shark attacks. We initially observe a positive correlation between these two variables, yet upon the inclusion of the confounder, temperature, we find that this correlation vanishes. One should avoid the confusion between confounders and correlated variables by noting that in addition to being correlated, confounders should also have a causal relationship to the variables involved in the confounding effect.

Although including confounders inherently introduces collinearity into the model, failing to account for them causes us to commit omitted variable bias in addition to causing the aforementioned confounding effects according to Simpson's paradox. (We note the difference between these confounding effects and the effect of collinearity on the coefficient signs: the latter may still occur even after including confounders in the model due to high collinearity causing coefficient estimates to become sensitive to changes in observed data). Hence, by weighing each consequence, we generally always include confounders as we have well-established methods for effectively addressing collinearity as well as for adjusting for confounding variables. Unfortunately, there is always a chance that confounders were not included in the study, causing many important issues difficult to be studied [2]. As this is not a problem exclusive to regression modelling, but observational studies in general, we highlight this further in Section 6.

5 Conclusion

After examining Python's and R's program output, we have observed differences in how they each handle the issue of multicollinearity. In particular, R's behaviour of combining correlated predictors while giving a warning upon detection of high multicollinearity aligns best with standard practice. It is also found that in the process of estimating the regression coefficients, R's `lm`, Python's `statsmodel`, and `scikit_learn` calculates the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ rather than $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ so that a result may be obtained even in the presence of singularity.

Additionally, we have observed a difference between the three least squares solvers' behaviour of flagging problematic multicollinearity, which indicates the general attitude of different fields that uses regression modelling. For example, `scikit_learn` aligns with machine learning engineers who are generally more concerned about making accurate predictions and hence would not be as concerned about the impacts of collinearity on model interpretability. This is in stark contrast to statisticians who bear greater interest in investigating the relationships between the regressors and the response. Hence, as opposed to `scikit_learn`, they would be more likely to use `statsmodel` or R, which in addition to flagging high multicollinearity, also affords a more extensive function library for the purpose of performing analysis on the model.

Taking these distinctions into account, it depends ultimately on the data analyst to make the final decision on which approach is best to handle anomalies in the model after performing analysis according to context and domain knowledge. Nevertheless, it is still crucial that one is made aware of these differences so as to select the appropriate tools depending on the individual's purpose and objective of regression modelling.

6 Future recommendations

The methods highlighted in Sections 3.1 and 3.2 are not exhaustive. One particular method of detecting collinearity for more potential exploration include the cos-max method [7], a recently proposed method which has the advantage of giving a coherent link between identifying which regressors are involved in a correlation and identifying which regressors are involved in each correlation. Furthermore, it is said to give more parsimonious collinearities compared other existing methods such as eigenvector analysis and variance decomposition. Additionally, other methods for addressing collinearity such as by using partial least squares regression or Bayesian regression, as well as their implementation in Python and R may also be explored in future research.

Expanding on the pitfalls in regression analysis, we also recommend a more in-depth consideration of confounding variables and causality.

Moreover, in Section 3.1.1, we stated that we examine correlation matrix to identify which predictors are correlated with each other after determining those with high VIF. The more discerning reader would realise that this comes with a limitation: only correlated pairs would be identified this way. Hence, to see beyond pairwise correlation, we recommend a further exploration on variance decomposition and principle component analysis (PCA) [4].

A Singular value decomposition

The SVD is a decomposition of any matrix into a sum of rank-1 matrices [8]. Before we give its formulation, we first state several definitions as well as results without proof.

A.1 Prerequisites

Definition A.1 (Symmetric matrices)**.** *An $n \times n$ square matrix $\boldsymbol{X}$ is symmetric if $\boldsymbol{A} = \boldsymbol{A}^T$.*

Definition A.2 (Orthogonal matrices)**.** *An $n \times n$ square matrix $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_n \end{pmatrix}$ is orthogonal if its columns $\boldsymbol{x}_i$ are orthonormal, that is, they are orthogonal to each other and each have a norm of 1, i.e. $\boldsymbol{x}_i \boldsymbol{x}_j^T = 0$ for $i \neq j$ and $\boldsymbol{x}_i \boldsymbol{x}_j^T = 1$ for $i = j$, where $1 \leq i, j \leq n$.*

Proposition A.1 (Spectral/eigen-decomposition)**.** *Any $n \times n$ symmetric matrix $\boldsymbol{A}$ can be written as*

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T = \sum_{i=1}^{n} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i^T,$$

where $\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \dots, \lambda_n\}$ is an $n \times n$ diagonal matrix with diagonals equal to the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{Q}$ is an orthogonal matrix whose columns are unit eigenvectors $\boldsymbol{q}_1, \dots, \boldsymbol{q}_n$ of $\boldsymbol{A}$.

Definition A.3 (Singular vectors and singular values)**.** *Let $\boldsymbol{A}$ be a $n \times p$ matrix, then σ is a singular value with corresponding left and right singular vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively if*

$$\boldsymbol{A}\boldsymbol{v} = \sigma \boldsymbol{u} \quad \text{and} \quad \boldsymbol{A}^T \boldsymbol{u} = \sigma \boldsymbol{v}.$$

A.2 Singular value decomposition

Proposition A.2 (Singular value decomposition)**.** *Let $\boldsymbol{A}$ be an $n \times p$ matrix with rank r, where $1 \leq r \leq \min(n, p)$. Then there exists an $n \times r$ matrix $\boldsymbol{U} = \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_r\}$, an $p \times r$ matrix $\boldsymbol{V} = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_r\}$, and an $r \times r$ matrix $\boldsymbol{\Sigma} = \mathrm{diag}\{\sigma_1, \dots, \sigma_r\}$, such that*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T,$$

where $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_r = \boldsymbol{V}\boldsymbol{V}^T$, and $\sigma_1 \geq \dots \geq \sigma_r > 0$.

The form given above is called the compact SVD. The non-compact form is given by

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ is an $n \times n$ orthogonal matrix, $\boldsymbol{V}$ is a $p \times p$ orthogonal matrix and Σ is a $n \times p$ diagonal matrix, leaving the remaining diagonals after the r^{th} entry zero.

Proposition A.3. *Let $\boldsymbol{A}$ be an $n \times p$ matrix with rank r, then*

$$\mathrm{rank}(\boldsymbol{A}^T\boldsymbol{A}) = \mathrm{rank}(\boldsymbol{A}) = r.$$

Proof. To prove this, we show that $\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{A}$ have the same null space and hence the same nullity. We do this by proving that $\mathrm{nullity}(\boldsymbol{A}) \subseteq \mathrm{nullity}(\boldsymbol{A}^T\boldsymbol{A})$ and $\mathrm{nullity}(\boldsymbol{A}^T\boldsymbol{A}) \subseteq \mathrm{nullity}(\boldsymbol{A})$ separately.

For the first case, let $\boldsymbol{x} \in \mathbb{R}^p$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}_n$. Then, $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T(\boldsymbol{0}_n) = \boldsymbol{0}_r$. So $\mathrm{nullity}(\boldsymbol{A}) \subseteq \mathrm{nullity}(\boldsymbol{A}^T\boldsymbol{A})$. For the second case, $\boldsymbol{x} \in \mathbb{R}^p$ such that $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. Left multiplying both sides by $\boldsymbol{x}^T$, we have

$$\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{A}\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}\|^2 = \boldsymbol{0},$$

which implies that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$, so $\mathrm{nullity}(\boldsymbol{A}^T\boldsymbol{A}) \subseteq \mathrm{nullity}(\boldsymbol{A})$.

Then since the dimension of the domain of $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}$ are both p, by the rank-nullity formula, they have the same rank. □

Proposition A.4. *Let $\boldsymbol{A}$ be an $n \times p$ matrix of rank r. Then the non-zero eigenvalues of both $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$ are $\sigma_1^2, \dots, \sigma_r^2$. Furthermore, the corresponding unit eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$ are given by the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively.*

Proof. We prove this result by first noting that $\boldsymbol{A}^T\boldsymbol{A}$ is a $p \times p$ symmetric matrix, so we can write its spectral decomposition as

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$$

where $\boldsymbol{V}$ is a $p \times r$ semi-orthogonal matrix containing the orthonormal eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$ (and hence its columns $\boldsymbol{v}_i$ are orthonormal) and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_r)$ is a diagonal matrix with its diagonals containing eigenvalues $\lambda_1 \geq \dots \geq \lambda_r > 0$. Note that we know that there are r eigenvalues by Proposition A.3.

Now, we let each $\sigma_i = \sqrt{\lambda_i}$ and $\boldsymbol{u}_i = \frac{1}{\sigma_i}\boldsymbol{A}\boldsymbol{v}_i$ for $i = 1, \dots, r$. We then show that the vectors $\boldsymbol{u}_i$ are orthonormal and subsequently, $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are left and right singular vectors corresponding to singular values σ_i as follows:

$$\boldsymbol{u}_i^T\boldsymbol{u}_j = \frac{1}{\sigma_i\sigma_j}\boldsymbol{v}_i^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}_j = \frac{1}{\sigma_i\sigma_j}\boldsymbol{v}_i^T(\lambda_j\boldsymbol{v}_j) = \frac{\sigma_j^2}{\sigma_i\sigma_j}\boldsymbol{v}_i^T\boldsymbol{v}_j = \frac{\sigma_j^2}{\sigma_i\sigma_j}\boldsymbol{v}_i \cdot \boldsymbol{v}_j.$$

Notice that if $i = j$, then the expression simplifies to just $\|\boldsymbol{v}_i\|^2 = 1$ since the vectors $\boldsymbol{v}_i$ are orthonormal. If $i \neq j$, then the dot product equals to zero, implying that the vectors $\boldsymbol{u}_i$

are orthonormal. Now, following the definition of singular vectors, we consider

$$\boldsymbol{A}^T\boldsymbol{u}_i = \frac{1}{\sigma_i}(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}_i) = \frac{\sigma_i^2}{\sigma_i}\boldsymbol{v}_i = \sigma_i\boldsymbol{v}_i,$$

so $\boldsymbol{v}_i$ are right singular vectors corresponding to singular values σ_i while $\boldsymbol{u}_i$ by our formulation are left singular vectors. Now all that is left is to construct $\boldsymbol{U} = \begin{pmatrix}\boldsymbol{u}_1 & \dots & \boldsymbol{u}_r & \dots & \boldsymbol{u}_n\end{pmatrix}$ and $\boldsymbol{\Sigma} = \mathrm{diag}\{\sigma_1, \dots, \sigma_r, 0, \dots, 0\}$. □

A.3 Computing the SVD

Let $\boldsymbol{A}$ be any $m \times n$ matrix. To compute the SVD of $\boldsymbol{A}$, we first solve for eigenvalues λ of $\boldsymbol{A}\boldsymbol{A}^T$ or $\boldsymbol{A}^T\boldsymbol{A}$ depending on which results in a matrix of smaller size. Then we take the singular values $\sigma = \sqrt{\lambda}$ and construct $\boldsymbol{\Sigma}$. Since in regression modelling, we often have more samples than predictors (which translates to more rows than columns), we choose $\boldsymbol{A}^T\boldsymbol{A}$. Now, we solve for right singular vectors by using its definition:

$$(\boldsymbol{A}^T\boldsymbol{A} - \lambda I)\boldsymbol{v} = \boldsymbol{0}.$$

Then, we convert the right singular vectors to unit vectors (so that $\boldsymbol{U}$ is orthogonal) and construct matrix $\boldsymbol{U}$. Compute $\boldsymbol{V}$ by using the definition of right singular vectors:

$$\sigma_i\boldsymbol{v}_i = \boldsymbol{A}^T\boldsymbol{u}_i.$$

B Referenced results

B.1 Estimating the least squares coefficients

Let the $n \times p$ data matrix containing n observations of p predictors be $\boldsymbol{X}$, and the corresponding $n \times 1$ column vector of observed responses be $\boldsymbol{y}$. We state the model representing the relationship between the $\boldsymbol{y}$ and $\boldsymbol{X}$ to be

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{\varepsilon}$ is a $n \times 1$ column vector containing the corresponding regression errors. The least squares method estimates $\boldsymbol{\beta}$ by minimising the sum of squared errors, i.e.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2 = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2. \tag{9}$$

We show the derivation of the closed form solution to this problem via vector calculus, where we find the derivative the sum of squared errors with respect to $\boldsymbol{\beta}$ and set it to zero:

$$\begin{aligned}
0 = \frac{d}{d\boldsymbol{\beta}}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 &= \frac{d}{d\boldsymbol{\beta}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \\
&= \frac{d}{d\boldsymbol{\beta}}\left(\boldsymbol{y}^T\boldsymbol{y} - (\boldsymbol{\beta}^T\boldsymbol{X}^T)\boldsymbol{y} - \boldsymbol{y}^T(\boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{X}\boldsymbol{\beta})\right) \\
&= \frac{d}{d\boldsymbol{\beta}}\left(\boldsymbol{y}\boldsymbol{y}^T - 2\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\right) \\
0 &= 0 - 2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\
\boldsymbol{X}^T\boldsymbol{y} &= \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} \\
\hat{\boldsymbol{\beta}} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}
\end{aligned}$$

So when $\boldsymbol{\varepsilon}$ is minimised, we have

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{10}$$

i.e. $\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ projects the observed response $\boldsymbol{y}$ onto the model space (or estimation space), that is, the set of all possible $\hat{\boldsymbol{y}}$ for which there exists a value of $\boldsymbol{\beta}$ such that $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$. More importantly, the projection is orthogonal, which has the geometric interpretation that the $\hat{\boldsymbol{y}}$ obtained by OLS is the closest vector in the model space to $\boldsymbol{y}$ [1]. This is consistent with the motivation of OLS.

In fact, we can derive the exact same formula from a geometrical standpoint by considering the orthogonal decomposition of $\boldsymbol{y}$ into a sum of one vector in the model space and one vector in the orthogonal subspace of the model space: the error space. Now, the OLS problem is rephrased as a matter of finding the orthogonal projection of $\boldsymbol{y}$ onto the model space so that the error component is minimised.

B.2 Minimum norm approximation using pseudoinverse

We can compute the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ using its SVD as follows.

Let $\boldsymbol{A} = \boldsymbol{X}^T\boldsymbol{X}$. If the SVD of $\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$, then the pseudoinverse of $\boldsymbol{A}$, $\boldsymbol{A}^+$, is given by

$$\begin{aligned}
\boldsymbol{A}^+ &= (\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T(\boldsymbol{U}\Sigma\boldsymbol{V}^T)^{-1}(\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T \\
&= (\boldsymbol{V}\Sigma\boldsymbol{U}^T\boldsymbol{U}\Sigma\boldsymbol{V}^T)^{-1}(\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T \\
&= (\boldsymbol{V}\Sigma^2\boldsymbol{V}^T)^{-1}(\boldsymbol{V}\Sigma\boldsymbol{U}^T) \\
&= (\boldsymbol{V}^T)^{-1}(\Sigma^{-1})^2(\boldsymbol{V})^{-1}\boldsymbol{V}\Sigma\boldsymbol{U}^T \\
&= \boldsymbol{V}\Sigma^{-1}\boldsymbol{U}^T.
\end{aligned}$$

Proposition B.1. *For $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, if the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has solutions, then $\boldsymbol{x}^* = \boldsymbol{A}^+\boldsymbol{b}$ is an exact solution and has the smallest possible norm, i.e. $\|\boldsymbol{x}^*\| \leq \|\boldsymbol{x}\|$ for all $\boldsymbol{x}$.*

Proof. Since $\boldsymbol{A}^+$ is a generalised inverse, $\boldsymbol{A}^+\boldsymbol{b}$ must be a solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Now, for any solution $\boldsymbol{x} \in \mathbb{R}^n$, consider its orthogonal decomposition via $\boldsymbol{A}^+\boldsymbol{A} \in \mathbb{R}^{n\times n}$:

$$\boldsymbol{x} = (A^+A)\boldsymbol{x} + (\boldsymbol{I} - A^+A)\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b} + (\boldsymbol{I} - A^+A)\boldsymbol{x}.$$

Then by the Pythagorean theorem, we have

$$\|\boldsymbol{x}\|^2 = \|\boldsymbol{A}^+\boldsymbol{b}\|^2 + \|(\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\boldsymbol{x}\|^2 \geq \|\boldsymbol{A}^+\boldsymbol{b}\|^2.$$

Hence $\|\boldsymbol{x}\| \geq \|\boldsymbol{A}^+\boldsymbol{b}\|$. □

B.3 Singularity due to perfect collinearity

Let $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{u} & k\boldsymbol{u} & \boldsymbol{v} \end{pmatrix}$ where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are arbitrarily linearly independent column vectors containing n observations of a particular feature with $\boldsymbol{u}^T = \begin{pmatrix} u_1 & \dots & u_n \end{pmatrix}$ and $\boldsymbol{v}^T = \begin{pmatrix} v_1 & \dots & v_n \end{pmatrix}$, with $k \in \mathbb{R}$ such that the column $k\boldsymbol{u}$ is a scalar multiple of the first column. Consider

$$\begin{aligned}
\boldsymbol{X}^T\boldsymbol{X} &= \begin{pmatrix} \boldsymbol{u}^T \\ k\boldsymbol{u}^T \\ \boldsymbol{v}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{u} & k\boldsymbol{u} & \boldsymbol{v} \end{pmatrix} \\
&= \begin{pmatrix} u_1 & \dots & u_n \\ ku_1 & \dots & ku_n \\ v_1 & \dots & v_n \end{pmatrix} \begin{pmatrix} u_1 & ku_1 & v_1 \\ \vdots & \vdots & \vdots \\ u_n & ku_n & v_n \end{pmatrix} \\
&= \sum_{i=1}^{n} \begin{pmatrix} u_i \\ ku_i \\ v_i \end{pmatrix} \begin{pmatrix} u_i & ku_i & v_i \end{pmatrix} \\
&= \sum_{i=1}^{n} \begin{pmatrix} u_i^2 & ku_i^2 & u_iv_i \\ ku_i^2 & k^2u_i^2 & ku_iv_i \\ u_iv_i & ku_iv_i & v_i^2 \end{pmatrix}.
\end{aligned}$$

The second row of $\boldsymbol{X}^T\boldsymbol{X}$ is k times the first row for every i. After summing up the rank-1 matrices, the second row of $\boldsymbol{X}^T\boldsymbol{X}$ will also be k times the first. So, $\boldsymbol{X}^T\boldsymbol{X}$ is not full rank, and hence is singular.

By writing matrix-matrix products in this form, each term in the sum is a rank-1 matrix. However, under no collinearity, the rows in every term will not necessarily be dependent on some other rows **by the same factor**. In the formulation above, the third row differs from the first row by a multiplicative factor of v_i/u_i (this is consistent with the case of collinear columns since $ku_i/u_i = k$), which is a constant under no collinearity. This fraction will differ for varying i, thus after adding each term up, there is no common factor between the two rows.

Alternatively for a simpler and less analytical approach, we look at each entry in the product matrix as multiplying rows with columns:

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} \boldsymbol{u}\cdot\boldsymbol{u} & \boldsymbol{u}\cdot(k\boldsymbol{u}) & \boldsymbol{u}\cdot\boldsymbol{v} \\ (k\boldsymbol{u})\cdot\boldsymbol{u} & (k\boldsymbol{u})\cdot(k\boldsymbol{u}) & (k\boldsymbol{u})\cdot\boldsymbol{v} \\ \boldsymbol{v}\cdot\boldsymbol{u} & \boldsymbol{v}\cdot(k\boldsymbol{u}) & \boldsymbol{v}\cdot\boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}\cdot\boldsymbol{u} & k\boldsymbol{u}\cdot\boldsymbol{u} & \boldsymbol{u}\cdot\boldsymbol{v} \\ k(\boldsymbol{u}\cdot\boldsymbol{u}) & k(k\boldsymbol{u}\cdot\boldsymbol{u}) & k(\boldsymbol{u}\cdot\boldsymbol{v}) \\ \boldsymbol{v}\cdot\boldsymbol{u} & k(\boldsymbol{v}\cdot\boldsymbol{u}) & \boldsymbol{v}\cdot\boldsymbol{v} \end{pmatrix}.$$

We observe that the second row is k times the first row as expected. The third row again is not collinear with the first two. Suppose they are, then we solve for some $\lambda \in \mathbb{R}$ such that

$$\boldsymbol{v} \cdot \boldsymbol{u} = \lambda(\boldsymbol{u} \cdot \boldsymbol{u}) \quad \text{and} \quad \boldsymbol{v} \cdot \boldsymbol{v} = \lambda(\boldsymbol{u} \cdot \boldsymbol{v}).$$

It then follows that

$$\begin{aligned}
\|\boldsymbol{v}\|^2 &= \boldsymbol{v} \cdot \boldsymbol{v} \\
&= \lambda(\boldsymbol{v} \cdot \boldsymbol{u}) \\
&= \lambda^2(\boldsymbol{u} \cdot \boldsymbol{u}) \\
&= \lambda^2\|\boldsymbol{u}\|^2 \\
\|\boldsymbol{v}\| &= \lambda\|\boldsymbol{u}\| \\
&= \frac{\boldsymbol{v} \cdot \boldsymbol{u}}{\|\boldsymbol{u}\|^2}\|\boldsymbol{u}\| \\
\boldsymbol{v} &= \frac{\boldsymbol{v} \cdot \boldsymbol{u}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}
\end{aligned}$$

i.e. $\boldsymbol{v}$ is its own orthogonal projection on vector $\boldsymbol{u}$, implying that they are collinear, which is a contradiction. Hence the third row is linearly independent of the first two rows.

B.4 Effect of `StandardScaler` on collinear columns in data matrix

From the documentation, `scikit_learn`'s `StandardScaler` calculates the Z-score of each entry by subtracting each entry by the mean of its column and dividing by the standard deviation of its column. Suppose that the true relationship between regressors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is $\boldsymbol{x}_2 = k\boldsymbol{x}_1$. Then we have

$$\bar{x_2} = k\bar{x_1} \qquad \text{and} \qquad s_2^2 = k^2 s_1^2.$$

So for collinear columns $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the entries of scaled columns $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are

$$u_{i1} = \frac{x_{i1} - \bar{x_1}}{s_1} \qquad \text{and} \qquad u_{i2} = \frac{x_{i2} - \bar{x_2}}{s_2} = \frac{kx_{i1} - k\bar{x_1}}{ks_1} = \frac{x_{i1} - \bar{x_1}}{s_1} = u_{i1},$$

i.e. after scaling predictors, the perfectly collinear columns will become identical, causing the least squares solver to assign equal weights to both predictors.

References

[1] Alan Agresti. *Foundations of Linear and Generalized Linear Models*. John Wiley and Sons Inc., 2015.

[2] Alan Agresti, Christine Franklin, and Bernhard Klingenberg. *Statistics : the art and science of learning from data*. Pearson Education Limited, 2018.

[3] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *Introduction to Statistical Learning with Applications in Python*. Springer, 7 2023.

[4] Ron Johnston, Kelvyn Jones, and David Manley. Confounding and collinearity in regression analysis: a cautionary tale and an alternative procedure, illustrated by studies of british voting behaviour. *Quality and Quantity*, 52:1957–1976, 7 2018.

[5] James McClave and Terry Sincich. *Statistics*. Pearson Education Limited, 13 edition, 2018.

[6] Robert M. O'Brien. A caution regarding rules of thumb for variance inflation factors. *Quality and Quantity*, 41:673–690, 10 2007.

[7] Zillur R. Shabuz and Paul H. Garthwaite. Examining collinearities. *Australian and New Zealand Journal of Statistics*, 66:367–388, 9 2024.

[8] Gilbert Strang. *Linear algebra and learning from data*. Wellesley-Cambridge Press, 2019.