Linear Regression: How to Make What's Old New Again

Kimberly Roye
Sara Jardine
Christian Smart

Galorath Federal

Abstract

With the booming popularity of machine learning techniques, modern data scientists may have you believe that using linear regression to analyze a dataset is outdated and no longer an effective method. Though many cost estimating applications are nonlinear, when there are linear relationships among features, ordinary least squares regression often performs better than the most powerful machine learning techniques. Scenarios for which linear regression should be chosen over more complicated algorithms are presented, as well as techniques such as regularization, gradient descent, and Bayesian methods.

Introduction

Machine learning techniques are growing in popularity across a multitude of industries. In the cost community, these algorithms are also being used to help cost analysts predict effort or cluster datapoints to facilitate deeper understanding of patterns within data. Linear regression is a machine learning technique that has largely been overshadowed by alternative techniques that use more computational power and require coding skills to implement. We will introduce some of these techniques, such as regularization, gradient descent, and Bayesian methods, that are proven to be useful in certain situations. However, we will also show that linear regression can still be the best method for explaining simple relationships between variables within certain datasets. Despite innovative machine learning methods, linear regression is still a powerful method that supersedes many fancier methods taking preference in modern day data analysis.

Linear Regression – A Quick Refresh

In cost and data analysis, we are often concerned about the relationships between a few factors and one primary target variable. The target variable is our dependent variable, and our goal is to understand how other variables within the available dataset impacts the increase or decrease of this variable. Most often, we are attempting to understand factors impacting cost; how many effort hours are required to develop a new component, how much will this new program cost

if additional functionality and lethality is incorporated into a vehicle, or how much will it cost to sustain this missile?

Historically, when estimating costs in defense and aerospace programs, the relationships that exist between cost and technical parameters such as weight and speed are typically non-linear (e.g., power equation). However, occasionally we witness a linear relationship between effort/cost and an independent variable.

Linear models use least squares to fit a line through a dataset. The goal is to fit the best line to the data to minimize the sum of the squares of the residuals. The residuals are the distance from the line to each data point. In the optimal situation, the data point will fit tightly to the line that is fit to the data. This results in small distances between the line and each data point.

The equation for the line is $\hat{Y} = \beta_1 X + \beta_0 + \varepsilon$ with β_1 being defined as the slope of the line, β_0 is the y-intercept and ε is an error term representing random sampling noise. Figure 1 provides an example of how the linear regression line is fit to the data points based on a given target, dependent, and independent variable.

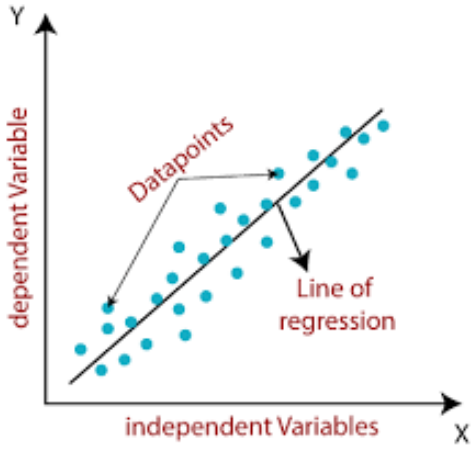

Figure 1: Linear Regression Example (Source: Javatpoint)

The data that is used in this paper to illustrate the use of linear regression and other methods is a dataset of software programs currently in the sustainment phase. After exploring the available data, we have discovered that the number of Software Changes and the Duration of the program are both influential in estimating the total hours (or effort) required annually to sustain a software program. This means we are no longer focusing on just one independent variable, but two.

When there is more than one independent variable in a linear model, it is now called a multiple linear regression model. The equation form for the model based on the software dataset is $\hat{Y} = \beta_1 X_1 + \beta_2 X_2 + \beta_0 + \varepsilon$ where β_1 and β_2 are the slopes of the two independent variables and β_0 is the Y-intercept.

Figure 2 presents the plot of the multiple regression line fit to the software data. For this resulting model, we have a multiple linear regression model with Software Changes (SC) and Duration being the independent variables estimating Total Hours.

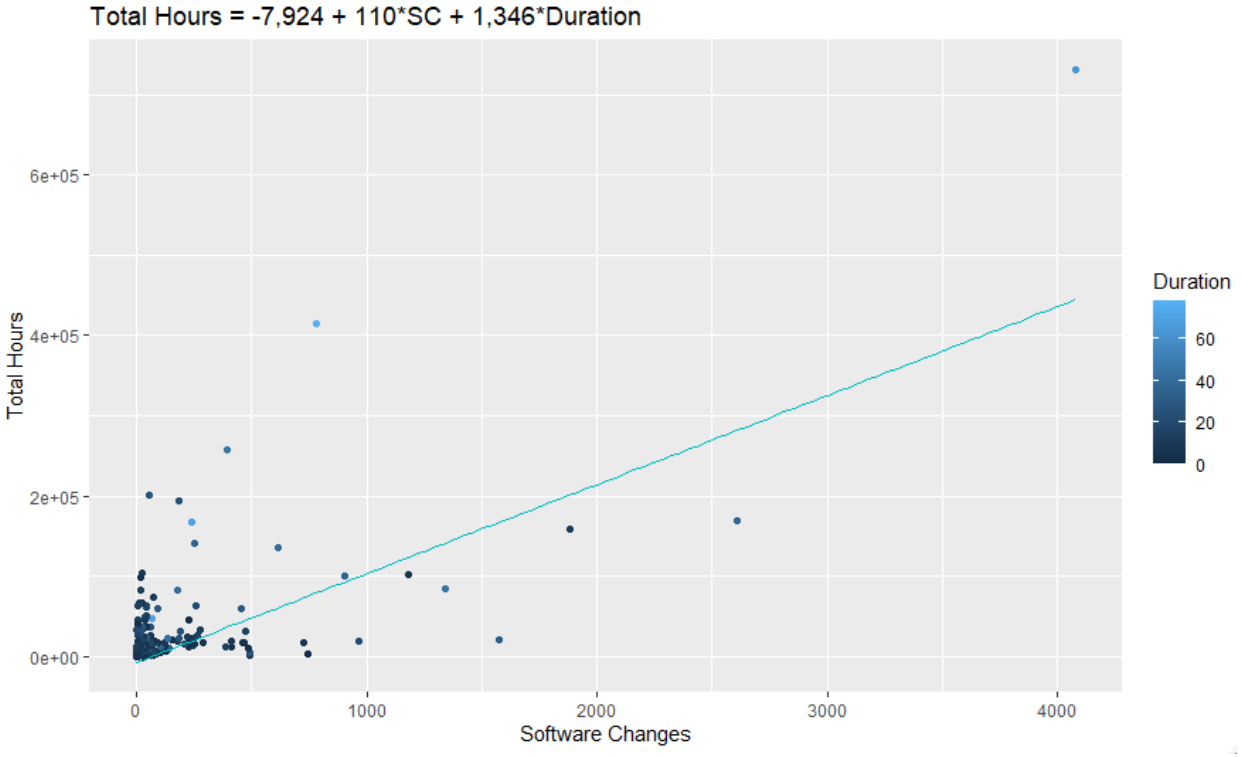

Figure 2: Plot of Multiple Linear Regression Model for Software Hours

The model was trained with 221 datapoints from the original dataset of 316 datapoints and the same model applied to the remaining 95 datapoints that were withheld to test the model. This is performed to determine how well the model predicts Total Hours based on new data that has not been included in the training or learning process. The goodness-of-fit metrics for the training and testing metrics are presented in Table 1.

Table 1: Multiple Linear Regression Goodness-of- Fit Statistics

Metric	Training	Test
R^2_{adj}	75%	75%
Root Mean Squared Error (RMSE)	26,928	48,089

The $\boldsymbol{R^2_{adj}}$ tells us how much of the variability in Total Hours is explained by Software Changes and Duration. The Root Mean Square Error (RMSE) is the square root of the average of the squares of the residuals and is a measure of the spread in the residuals. When visually observing the plot in Figure 2, it is not surprising that the RMSE is large for this dataset since we observe several points that fall far from the line fit to the data. Ideally, we would want to see all points falling closer to the line but based on the nature of the imperfect data in our study, a high RMSE seems reasonable.

Methods to Improve Linear Models

Suppose we are worried that the model we built is performing much better on the training data than on unseen data. The goal with a linear model is, as previously mentioned, to minimize the sum of the squares of the residuals while finding the patterns in the data that will help to predict the target variable for new data. If the model constructed does not predict Total Hours well for new software programs entering the sustainment phase, then the model is deemed not useful, and we are back to square one. In this case, alternative techniques should be considered to determine a more applicable model.

Regularization is a technique that reduces error in a model by fitting a function on the training dataset while helping to avoid overfitting the data. The method performs this by adding a penalty to the least squares method to reduce the slope of the line and make predictions less sensitive to the parameters of the model. Another technique is gradient descent, which is an iterative method that attempts to find values for the intercept and slope of a line to fit the data. A third technique is Bayesian methods, which can also be applied to linear regression methods. When incorporating Bayesian methods, we create the linear regression model using probability distributions rather than point estimates. This yields a $\boldsymbol{Y}$ that is assumed to be drawn from a probability distribution. A more detailed discussion of each of the three methods to improve linear models is as follows.

Regularization

Regularization is often referred to as the *shrinkage* method since the penalty term that is added helps to constrain the slope parameters to zero. First, we will define the cost function. As mentioned, linear regression (or Ordinary Least Squares) seeks to find the best line that minimizes the sum of the residuals. The cost function quantifies the error between predicted and expected values (the residuals) and presents it in the form of a single real number. The cost function can either be minimized or maximized. In regularization, the cost function will be minimized to find the values of the model parameters for which the cost function returns the smallest number possible. The **loss function** formula is defined as:

$$Mean\ Squared\ Error = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_i - \beta_0)^2$$

where *n* is the sample size. With each type of regularization, a penalty is added to the loss function to estimate the parameters.

With the incorporation of a penalty, bias is introduced into the model. Bias is the difference between the average prediction of the model and the actual value which we are trying to predict. Models with high bias underfit the data. However, with the addition of a minimal amount of bias, the variance is reduced. Variance is the variability of model prediction for a given data point and tells us the spread of the data. When there is high variance, the model tends to overfit the data. Figure 3 provides a visual for understanding the goal of balancing bias and variance to ensure strong models are created to capture patterns in the data and are not too specific to the training data used to train the algorithm.

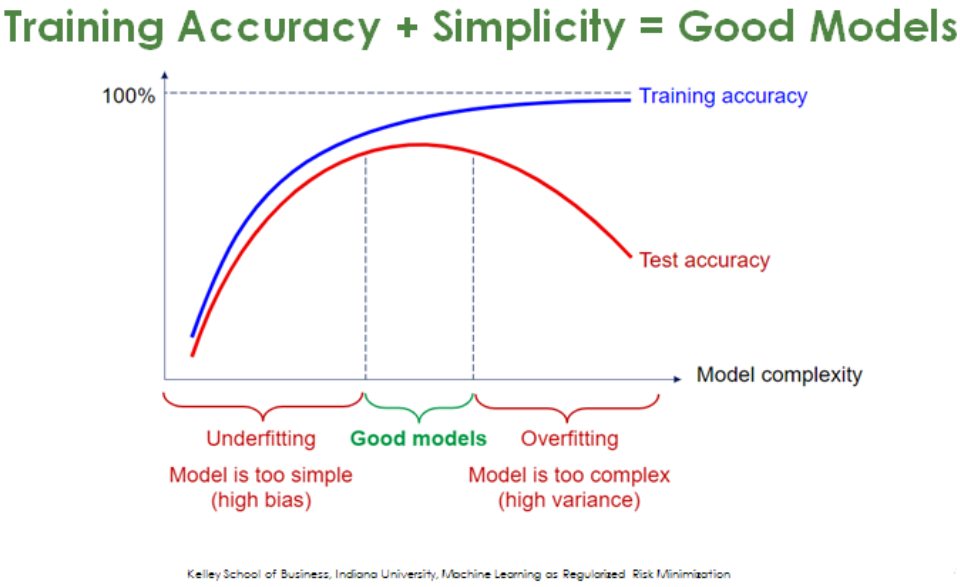

Figure 3: Understanding Underfitting and Overfitting

Three regularization methods will be covered: Ridge, Lasso, and Elastic-Net regression. A simple explanation of each method is included in the following sections.

Ridge Regression

As mentioned, regularization methods are considered when we suspect the selected model may be overfitting the data. Ridge regression attempts to fit a new line to the data while introducing a small amount of bias. The purpose of including this bias is to drop the variance and provide better long-term predictions.

Ridge regression minimizes the loss function with an added function. Ridge regression seeks to minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_i - \beta_0)^2 + \lambda * Slope^2$$

This method is an extension of OLS but adds a penalty to the method. Lambda, denoted by λ, determines the severity of this penalty. The value of λ can range from 0 to positive infinity. If λ=0, the ridge regression penalty is also equal to zero. This means the ridge regression line will be the same as the OLS base case regression line since both are minimizing the sum of squared residuals. The user must determine the value of λ that optimizes the regression results. This should be done using cross-validation to determine which regression equation results in the lowest variance.

Cross-validation is a statistical resampling technique used to evaluate models. It uses a random, limited sample to estimate how the model is expected to perform on data that has not been used to train the model. This technique is an iterative process. To determine the optimal value of λ, k-fold cross-validation is used and follows these steps:

Step 1: The data is randomly sorted

Step 2: The dataset is split into *k* groups

Step 3: A sample of the data is extracted as a hold out or test dataset

Step 4: The remaining datapoints are used as the training dataset

Step 5: The original OLS model is fit to the training dataset and evaluated on the test dataset

Step 6: An evaluation score is calculated and retained, and the model is discarded

Step 7: This process occurs iteratively until all the k groups have been used

Step 8: The model with the lowest error will be selected and the λ value from this model is considered optimal

Once the optimal λ value is chosen, the model that minimizes the loss function is calculated.

Ridge regression can be applied to continuous and discrete variables. This method is best used when there is a large quantity of variables within the dataset, specifically if there are more variables than data points. Ridge regression was

applied to the software sustainment dataset using the R programming application.

The optimal lambda based on cross validation is 0.001. Using this lambda in the algorithm for Ridge Regression, we obtained the optimum model of:

$$Total\ Hours = -7{,}924.07 + 110.99 * SC + 1{,}345.49 * Duration$$

This model is very close to the version created using OLS. Table 2 presents the fit statistics for the Ridge regression model. Though we see the RMSE is the same as the OLS model, the R^2_{adj} is significantly less for the test set than the training set. This means the model does not fit as well on new data as it does on the data within the training dataset.

Table 2: Ridge Regression Goodness-of-Fit Statistics

Metric	Training	Test
R^2_{adj}	75%	36%
Root Mean Squared Error (RMSE)	26,928	48,089

Lasso Regression

Lasso regression is like Ridge regression, but with some important differences. Lasso regression seeks to minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_i - \beta_0)^2 + \lambda * abs(Slope)$$

Instead of adding the penalty to include $\lambda * Slope^2$ as with the Ridge method, the penalty multiplies λ by the absolute value of the slope. However, like Ridge, λ can range from 0 to positive infinity, and the optimal value is determined by cross-validation. An important difference between Lasso and Ridge aside from the loss function is that as we increase the value of λ, the slope gets smaller and moves towards zero. Ridge regression can only shrink the slope asymptotically near zero while Lasso can shrink it to zero.

Lasso seeks to discard useless variables from equation, so the models produced by Lasso will at times be simpler and easier to interpret.

Lasso regression was applied to the software sustainment dataset. The optimal lambda based on cross validation is 0.001, as with Ridge regression. Using this lambda in the algorithm for Lasso regression, we obtain the optimum model of:

$$Total\ Hours = -7{,}924.07 + 111 * SC + 1{,}345.49 * Duration$$

This model is identical to the model produced with Ridge regression. Table 3 presents the fit statistics for the Lasso regression model. Again, the R^2_{adj} is

significantly less for the test set than the training set. There is no improvement realized using Ridge or Lasso regression.

Table 3: Lasso Regression Goodness-of-Fit Statistics

Metric	Training	Test
R^2_{adj}	75%	36%
Root Mean Squared Error (RMSE)	26,928	48,089

Elastic-Net Regression

Elastic-Net regression is a hybrid approach that combines the components of Ridge and Lasso regression methods and includes two lambdas in the loss function. Elastic-Net seeks to minimize:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_i - \beta_0)^2 + \lambda * Slope^2 + \lambda * abs(Slope)$$

For this method, cross-validation is used on different combinations of $\lambda_1$and λ_2 to find the best values. When both values of lambda are zero, the resulting model will be the same as the base case OLS model.

This hybrid approach groups and shrinks the parameters associated with the correlated variables or removes them if they are highly correlated. Based on the characteristics of Lasso regression, Elastic-Net tends to favor a more simplified model.

Elastic-net regression was applied to the software sustainment dataset. The optimal lambda based on cross validation is 0.008. Using this lambda in the algorithm for Elastic-net regression, we obtain the optimum model of:

$$Total\ Hours = -7{,}924.07 + 111 * SC + 1{,}345.49 * Duration$$

This model is identical to the model produced with the other two regularization techniques. Table 4 presents the fit statistics for the Elastic-net regression model. Again, the R^2_{adj} is significantly less for the test set than the training set. There is no improvement realized using any of the regularization techniques.

Table 4: Elastic-Net Regression Goodness-of-Fit Statistics

Metric	Training	Test
R^2_{adj}	75%	36%
Root Mean Squared Error (RMSE)	26,930	48,056

In summary, regularization methods, which adds a penalty term to constrain the slope parameters to zero did not improve the original OLS model regression results. In our sustainment software program dataset, linear regression prevails as the best method to determine a model that predicts hours as a function of software changes and duration.

Gradient Descent

An additional method that can be used to optimize a line fit to data is gradient descent. Gradient descent finds the optimal values for both the intercept and slope(s) of a linear regression line.

Recall from the discussion of regularization, the loss function was defined as:

$$Mean\ Squared\ Error = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_i - \beta_0)^2$$

The goal of gradient descent is also to minimize the loss function.

This method works by progressing through the following steps:

1. Take the derivative of the loss function with respect to the intercept
    a. The derivative is used to determine where the sum of the squared residuals is the lowest
2. Begin with an initial guess or a base line fit to the data
3. Gradient descent finds the minimum value by taking steps from the initial guess until it reaches the best value
4. The algorithm determines the step size by multiplying the slope by the learning rate
    a. The learning rate, alpha, must be chosen. The learning rate is a hyperparameter that needs to be tuned and can be done so by using a variety of methods. This process will not be discussed at length in this paper, but the authors suggest analysts explore this topic in detail. Some rules of thumb to keep in mind when selecting the learning rate is that if the rate is too small, it will take a long time for the algorithm to find a minimum value of the loss function. Choosing a learning rate that is too large is also problematic as the minimum value of the loss function might be skipped over and an optimal solution never reached. For this paper, we chose a fixed learning rate of 0.00000000025.
    b. Step Size = Slope (from the starting model) * Learning Rate
5. Calculate the new intercept
    a. New Intercept = Old Intercept (from the starting model) – Step Size

6. Plug the new intercept into the derivative formula and continues to repeat the steps above until the Step Size is close to zero
7. When the Step Size converges to zero, the slope should also be very small and be approaching zero

Gradient descent was applied to the software sustainment dataset. Alpha was set to 0.00000000025. We obtained the optimum model of:

$$Total\ Hours = 0.04 + 46.07 * SC + 1.23 * Duration$$

This model differs from the models produced using regularization significantly. Table 5 presents the fit statistics for the gradient descent model. Again, the R^2_{adj} is significantly less for the test set than the training set. There is no improvement realized using this technique compared to the linear model.

Table 5: Gradient Descent Goodness-of-Fit Statistics

Metric	Training	Test
R^2_{adj}	70%	21%
Root Mean Squared Error (RMSE)	42,995	59,239

Bayesian Methods

Historically, parametric methods have been based on frequentist techniques. Frequentist statistics is the classical method that uses a sample of data as inputs. If you have taken Statistics 101 in college, most, if not all the class was oriented towards this approach. For example, traditional linear and nonlinear regression analysis is a frequentist approach. The challenge with frequentist statistics is that it requires a large amount of data. Statisticians have conducted numerous studies using random data and have concluded that you need 50 data points for a regression analysis with 10 additional data points for every independent variable you want to include. For example, if you want to include three independent variables in your analysis, you need 80 data points. The number of highly specialized systems used in the Department of Defense and NASA means that we typically have nowhere near that much data. For example, the Missile Defense Agency has only developed a handful of different kill vehicles, and NASA has only developed a few crewed launch vehicles. When looking at truly applicable data, the sample size shrinks even further – when considering launch vehicles, the primary systems that NASA has completed have been those for the Apollo and Shuttle programs. The Apollo program began in the 1960s, and the Shuttle program began in the 1970s. Thus, there are no directly applicable historical data points within the last 40 years. Considering the changes that have taken place in the realm of technology since then, there really is no applicable historical data at all for these systems.

For small data sets like these, Bayesian methods can help provide more accurate estimates. Bayesian methods leverage all your experience, making them less subject to being overwhelmed by noise. This prior experience can be subjective or objective. The objective data could involve the use of similar data.

This approach has proven to be successful in a multitude of applications. Bayesian techniques were used in World War II to help crack the Enigma code used by the Germans, thus helping to shorten the war. John Nash's equilibrium for games with incomplete or imperfect information is a form of Bayesian analysis (John Nash's life was portrayed in the film, *A Beautiful Mind*). Actuaries have used Bayesian methods for over 100 years to set insurance premiums. Bayesian voice recognition researchers applied their skills as leaders of the portfolio and technical trading team for the Medallion Fund, a $5 billion hedge fund which has averaged annual returns of 35% after fees since 1989.

For the application of Bayesian regression, we will write the linear equation in mean deviation form:

$$\boldsymbol{Cost = \alpha_{\bar{x}} + \beta(X - \bar{X}) + \varepsilon}$$

where Cost is in 2022$ million. This form makes it easier to establish prior inputs, since it is easier to think of an average value for prior cost than it is for the intercept of the least-squares equation.

What is needed to apply Bayesian regression is two estimates. They could both be based on objective data. Another possibility is that you would have some prior idea about the cost or some element of the cost and that could be combined with objective data using Bayes' Theorem.

Consider as an example, the problem of estimating a spacecraft data processing unit (DPU). A linear cost estimating relationship (CER) based on 14 data points has the form:

$$Cost = 8.33 + 0.42*(Power - 19.4)$$

where Cost is the effort required in 2022$ to develop and build the data processing unit and Power is the average beginning of life instrument power in watts. Ten data points is a very small sample. To avoid being fooled by randomness, a typical rule of thumb for regression equation development is at least 50 data points.

However, in this case, you also believe that the DPU you are estimating is very similar to one that flew on a previous mission which we will call Mission X. You believe that if the instrument power requirements were the same that the cost would also be the same. However, the average power for Mission X's instruments was 69 watts, while for the system you are estimating, the same parameter is 30 watts. The actual cost for the Mission X DPU was $24 million. Your CER predicts a cost equal to $12.8 million. The CER is superior in some respects because it accounts for the difference in power, while the analogy does not. Is there a way to combine both pieces of information to get a better estimate?

The answer is yes. One method would be to assume that the analogy imparts useful information about an adjustment to the a-value of the CER, but not to the b-value. If the b-value 0.42 applies, then the implied a-value can be found by solving the equation

$$A+0.42*(69-19.4) = 24$$

which yields A = 3.17.

To apply Bayes' Theorem to combine the two a-values, we need a standard deviation for the implied a-value from the analogy. One way to think about this is estimate your confidence and express it in those terms. For example, if you are highly confident in your estimate of the intercept parameter you may decide that means you are 90% confident that the actual value will be within 5% of your estimate. For a normal distribution with mean μ and standard deviation σ, the upper limit of a symmetric two-tailed 90% confidence interval is 20% higher than the mean, that is,

$$\mu+1.645\sigma=1.20\mu$$

from which it follows that

$$\sigma= (0.20)/(1.645)\ \mu\approx 0.12\mu$$

Thus, the coefficient of variation, which is the ratio of the standard deviation to the mean, is 12% in this case. See Table 6 for a list of suggested values for the coefficient of variation based on the true mean being within 20% of the estimate with the stated confidence level.

Table 6: Coefficient of Variations for the Confidence That the True Mean is Within 20% of the Estimated Mean

Confidence Level	Coefficient of Variation
90%	12%
80%	16%
70%	19%
50%	30%
30%	52%
10%	159%

Suppose for this estimate that I am 70% confident that the true a-value is within 20% of the implied a-value 3.17. Then with a coefficient of variation equal to 12%, the standard deviation is equal to 3.17*0.19 = 0.60.

To combine the intercepts, we use a weighted average of the two intercepts. The weights are determined by the inverses of the variances (Smart 2014). The variance about the intercept from the regression is 2.17 and the variance based on subjective judgment is 0.36. It is straightforward to calculate that the implied a-value from the analogy has weight equal to 86% while the regression-based intercept gets weight equal to 14%. The combined intercept is thus 3.89.

We have no information from the analogy about the slope, so we only use the slope from the regression (you can think of the slope coefficient from the analogy having infinite variance, which means it gets a zero weight in the weighted average). The combined CER is thus 3.89+0.42*(Power – 19.4). The estimate applied to the input of 30 watts is $8.3 million. This is one way to combine objective and subjective data in a rigorous and proven framework.

When Linear Regression Works Best

Although we have introduced alternative techniques to potentially improve simple linear regression, there are still characteristics of a dataset that gravitates toward linear regression as the best method to derive the most statistically significant regression equation.

Characteristics of a linear dataset typically occurs when there is a relatively limited range in either the dependent variable, the independent variable, or both the dependent and independent variables. This can occur when estimating at the system and subsystem level, when all the data points are very similar, such as ships within a specific class, robotic spacecraft with a similar application (e.g., Martian missions), or wheeled and tracked vehicles. A limited range of data is also likely to occur when estimating at the component level, since the various data

points tend to be approximately the same size. Examples include batteries and data processing units. A good rule of thumb is that when the cost and the independent variable data points being modeled are all within an order of magnitude of one another, the relationship between cost and the independent variables is likely to be linear. When the spread is several orders of magnitude, the relationship is likely to be nonlinear.

Conclusion

Linear regression remains one of the most powerful algorithms in machine learning. With the emergence of new machine learning techniques and modern data science techniques within powerful tools like R and Python, it may be easy to overlook a simple linear relationship that may be the best fit for predicting a relationship in a dataset. When a linear relationship is determined, seeking ways to improve the regression statistics for a stronger relationship should be explored such as regularization, gradient decent, and Bayesian methods. When testing if these methods improve your regression line, sometimes the conclusion is there is no improvement and OLS remains the best regression method. When evaluating relationships in a dataset, oftentimes, the simplest model is not only preferred but represents the best relationship.

References

"Coding Deep Learning for Beginners – Linear Regression (Part 2): Cost Function", Krzyk, Kamil, August 2018.

"Understanding the Bias-Variance Tradeoff", Singh, Seema, May 2018.

"Bayesian Parametrics: How to Develop a CER with Limited Data and Even Without Data," Smart, Christian, June 2014.