Chapter 3
Optimization and Regression

Abstract Linear regression is the simplest method to build a relationship between input and output features. While many relationships are non-linear in science and engineering, linear regression is fundamental to understanding more advanced regression methods. In particular, gradient descent will be discussed as a technique with a wide range of applications. Key to understanding linear regression are concepts of optimization. In this chapter, the fundamentals of linear regression will be introduced, including least squares optimization through gradient descent. Extensions of linear regression to tackle some nonlinear relationships will also be discussed, including piecewise linear regression, and moving least squares. The ease and strength of linear regression will be demonstrated through example problems in baseball and material hardness.

Keywords Linear regression · Least squares optimization · Coefficient of determination · Minimum · Gradient descent · Multivariable linear regression · Baseball · Indentation · Vickers hardness · Bacteria growth · Piecewise linear regression · Moving average · Moving least squares · Regularization · Cross-validation

3.1 Least Squares Optimization

Least squares optimization is a method for determining the best relationship between variables making up a set of data. For example, if data is collected for measuring the shoe size and height of people, the raw data could be plotted on a graph, with shoe size on one axis and height on the other axis. The raw data may be interesting, but it would be more useful if a mathematical relationship can be found between these variables. If the data points fall in approximately a row, then a straight line can be drawn through the points. To determine the best placement for the straight line, least

Supplementary Information: The online version of this chapter (https://doi.org/10.1007/978-3-030-87832-0_3) contains supplementary material, which is available to authorized users.

W. K. Liu et al., *Mechanistic Data Science for STEM Education and Applications*,
https://doi.org/10.1007/978-3-030-87832-0_3

squares optimization (or linear regression) can be used. The details of this method are shown in this chapter. If the points do not fall in approximately a straight line, then some form of nonlinear optimization must be used. This chapter will focus on piecewise linear regression, the moving average, and moving least squares optimization for nonlinear optimization.

The least squares method for optimization and regression was first published by Adrien-Marie Legendre (1805) and Carl Friedrich Gauss (1809) [1]. They were both studying the orbits of celestial bodies such as comets and minor planets about the sun based on observations. The term *regression* is a commonly-used word for least squares optimization. The term was coined by Sir Francis Galton from his work in genetics in the 1800's. Galton was initially studying the genetics of sweet peas, in particular comparing the weights of planted and harvested peas. He found that when he plotted the data for the planted and harvested weights, the slope was less than unity, meaning that the offspring of the largest and smallest peas did not demonstrate the same extremes, but "regressed" to the mean. His concept of regression to the mean based on the evaluation of data on graphs led to the use of the term regression to describe the mathematical relationship developed based on data [2].

3.1.1 Optimization

Optimization is the process of finding the minimum (or maximum) value of a set of data or a function. This can be accomplished by analyzing extensive amounts of data and selecting the minimum (or maximum) value, but this is generally not practical. Instead, optimization is generally performed mathematically. A cost function, $c(w)$, is written as a relation between variables of interest and the goal of the optimization is to find the minimum (or maximum) value of the function over the range of interest.

The minimum (or maximum) value of the function corresponds to the location where the tangent slope becomes zero. To find the tangent slope, the first derivative of cost function is computed using differential calculus

$$\frac{dc(w^*)}{dw} = 0 \tag{3.1}$$

Setting this derivative equal to zero leads to the value of w^* as the location of the minimum (or maximum) of the function.

In general, the location where the first derivative equals zero is a potential minima, maxima, or inflection point. The second derivative of the original function is used to distinguish between these three types of points

$$\frac{d^2c(w^*)}{dw^2} > 0, \text{convex (minimum)} \tag{3.2a}$$

$$\frac{d^2c(w^*)}{dw^2} < 0, \text{non} - \text{convex (maximum)} \tag{3.2b}$$

$$\frac{d^2c(w^*)}{dw^2} = 0, \text{(inflection point)} \tag{3.2c}$$

Example: Consider the following function

$$c(w) = 2w^2 + 3w + 4 \tag{3.3}$$

where $c(w)$ is a quadratic function of the independent variable w (see blue curve in Fig. 3.1). The minimum of this quadratic equation is the point where the slope tangent to the curve is horizontal. The slope is computed using differential calculus to find the first derivative as

$$\frac{dc}{dw} = 4w + 3 = 0 \tag{3.4}$$

Setting this equation for the first derivative equal to zero and solving for w provides the location of $w^* = -3/4$ as the location of zero slope (minimum point), as shown in Fig. 3.1.

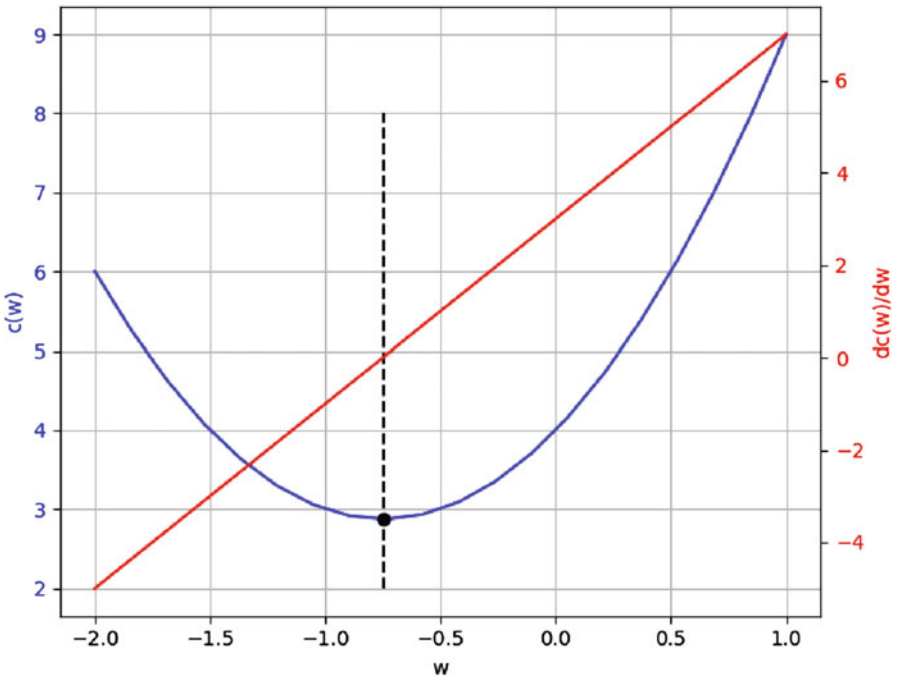


Fig. 3.1 A function (blue line) and its derivative (red line)

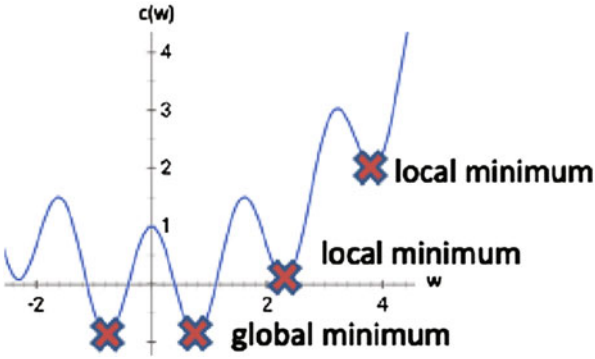


Fig. 3.2 Non-convex function with multiple local minima

For this function with the minimum located at $w^* = -3/4$, the second derivative is

$$\frac{d^2c(w^*)}{dw^2} = 4, \text{minimum} \tag{3.5}$$

Convex functions are preferred for optimization problems because they converge to a solution much easier. Non-convex functions can have multiple minima, maxima, and inflection points. The global minimum is defined as the absolute minimum across the span of interest. As shown in Fig. 3.2, finding a global minimum in a non-convex equation is challenging because a local minimum can be chosen erroneously instead of the global minimum.

3.1.2 Linear Regression

A straight line can easily be drawn through two data points and a simple linear expression can be written as

$$y = w_1 x + w_0 \tag{3.6}$$

where $w_1 = (y_2 - y_1)/(x_2 - x_1)$ is the slope of the line and w_0 is the location where the line crosses the y-axis. If there are more than two points and the points are not all aligned, it is obviously not possible to draw a straight line through all the data points. This common challenge often arises where the data points generally lie along a straight path, but it is not possible to fit a straight line through all the data points. One option is to try to draw a complicated curve through all the data points, but this option is generally not preferred due to the complex mathematics of such a curve. Instead, a straight line can be drawn through the data in such a way that it is close to as many points as possible. The process of determining the best fit of a straight line to the data is called linear regression (Fig. 3.3).

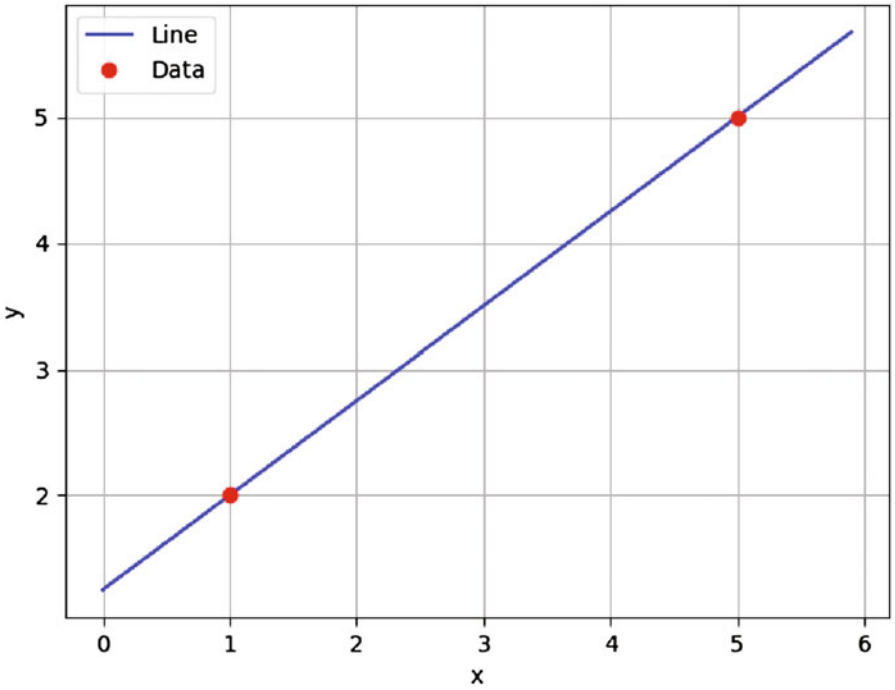
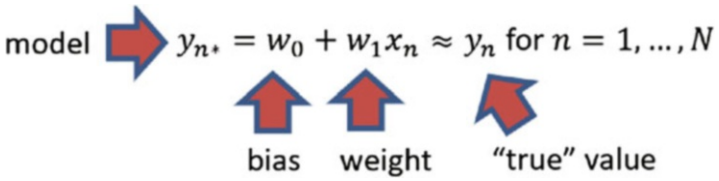


Fig. 3.3 Straight line through two data points

Linear regression is used to model a best-fit linear relationship between variables by fitting a linear equation to observed data. Consider a set of N data points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \tag{3.7}$$

A generic equation through these points can be written as



where the coefficients w_0, w_1 are called weights (the constant weight w_0 is often called the bias) and y_{n*} is the computed approximate value of the "true" value of y_n. The best fit approximation for y_{n*} is found through linear regression by determining the optimum values for the weights and bias based on the data. Note that when using regression for developing mechanistic data science models, it is important to assess the quality of the model through cross validation. The cross validation will be introduced in Sect. 3.3.

3.1.3 Method of Least Squares Optimization for Linear Regression

The optimum values for the coefficients w_0, w_1 are determined by minimizing the total error between the computed approximate value, y_{n*} and the "true" value, y_n. Each data point, x_n, is multiplied by the weight, w_1, and the bias, w_0, is added to it as the approximation, y_{n*}, to the "true" value, y_n. The approximation error at data point n in this linear regression model can be evaluated by subtracting the true value y_n and squaring the difference. The total error is found by repeating this for all data points

$$total\ error = \sum\nolimits_{n=1}^{N} (y_{n*} - y_n)^2 = \sum\nolimits_{\mathrm{n}=1}^{\mathrm{N}} (w_0 + w_1 x_n - y_n)^2 \tag{3.8}$$

The best fit line will be the one which minimizes the total error and is determined by performing a least squares optimization. This optimization begins with a cost function, $c(w_0, w_1)$, which is the average of the total error for all data points

$$cost\ function = c(w_0, w_1) = \frac{1}{N} \sum_{n=1}^{N} (w_0 + w_1 x_n - y_n)^2 \tag{3.9}$$

The weights and bias for a best fit line are determined by finding the minimum (or maximum) of a function or a set of data. In this case, the goal is to minimize the total error. The minimum value can be determined experimentally by collecting an extensive amount of data and selecting the lowest overall value, but that is generally not practical. Instead, the minimum value is usually determined mathematically using a functional relationship between variables of interest.

3.1.4 Coefficient of Determination (r^2) to Describe Goodness of Fit

Goodness of fit describes how closely the data points, y_i, are to the line drawn through them. If the line goes through the points like Fig. 3.3, the fit is perfect. If the points do not lie directly on the line but are generally evenly clustered along the length of the line like Fig. 3.4, the fit shows a linear relationship (adequate precision of the data is generally dependent on the application). Conversely, if the points do not evenly cluster along the length of the line, there may be no correlation between the variables or a nonlinear correlation between the variables.

A common way to quantify goodness of fit is by the coefficient of determination, r^2

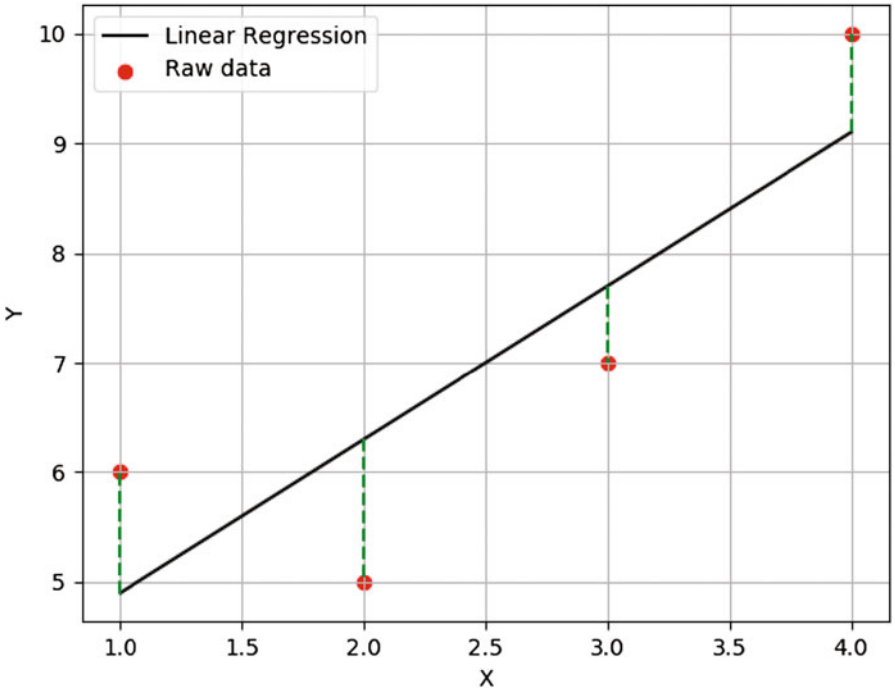


Fig. 3.4 Linear regression through non-colinear points

$$r^2 = \frac{regression\ sum\ of\ squares}{total\ sum\ of\ squares} = \frac{\sum_i \left(y_i^* - \overline{y}\right)^2}{\sum_i \left(y_i - \overline{y}\right)^2} \tag{3.10}$$

where $\overline{y}$ is the average of the data points y_i. A good fit of the regression to the data will result in an r^2 value closer to one.

3.1.5 Multidimensional Derivatives: Computing Gradients to Find Slope or Rate of Change

As shown above, computing the slope or the rate of change is important for optimization problems. *Gradient* is the slope or rate of change in a particular direction. For one-dimensional problems, determining the rate of change is trivial, but for problems involving multiple variables, determining the slope is more challenging since the slope can be different in each direction.

The rate of change is the amount one variable changes when one or more other variables change. Consider the mountain in Fig. 3.5 below. If a skier follows the red-dotted path, a short amount of forward motion will result in a big vertical drop (and more speed) as the skier goes from red dot to red dot. On the other hand, if a

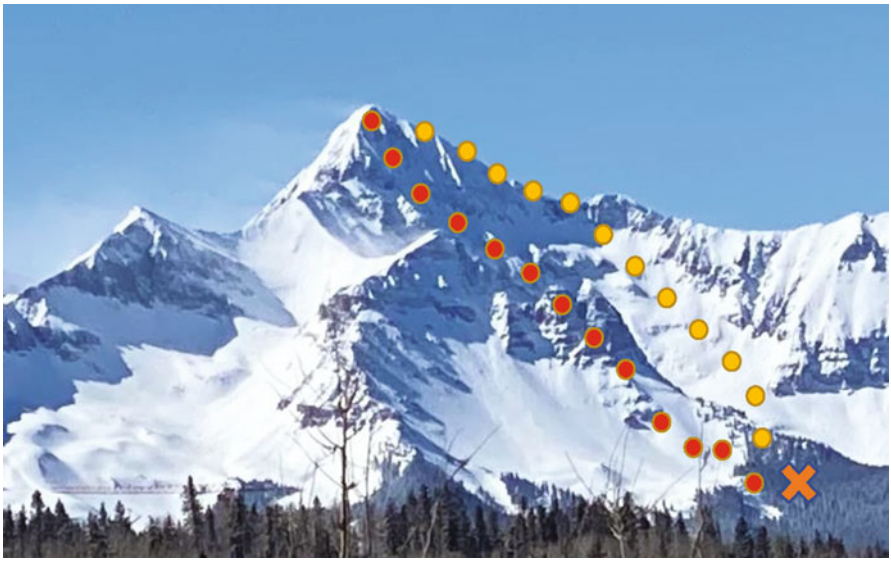


Fig. 3.5 Ski mountain with two possible paths. The red-dotted path is steeper than the yellow-dotted path. (Photo courtesy of Rebecca F. Boniol.) A video is available in the E-book, Supplementary Video 3.1

skier follows the yellow-dotted path, the vertical change with respect to the forward motion is less going from yellow dot to yellow dot. It can be seen that because of this, the route from the top of the mountain to the orange X is shorter and more direct when following the red-dotted path.

The rate of change, or slope, can be determined on an average sense (vertical change from the top of the mountain to the orange X relative to the horizontal change in position) or instantaneously (different slope at every red and yellow dot). If measured data are used, the instantaneous rate of change is estimated by dividing the change in vertical height by the change in horizontal distance. If an equation is available, the instantaneous rate of change can be computed mathematically by taking the derivative of the equation for the hill in the direction of travel. More generally, a coordinate system is defined and partial derivatives with respect to each of the coordinates are taken. This set of partial derivatives forms a vector and is the mathematical definition of the gradient. Once the expression for the gradient is computed, the gradient in any specific direction can be computed.

The gradient of a cost function is needed to perform a multivariate optimization. For the ski mountain in Fig. 3.5, the optimization values can be visualized, with the maximum corresponding to the top of the mountain and the minimum corresponding to the bottom of the mountain.

Optimization can be performed mathematically if a functional relationship is available. Using the mathematical function, the global minima for higher dimension functions requires defining the gradient of the function. The gradient is the derivative

of the function in multiple directions. Consider a vector $\boldsymbol{w}$ containing all the independent variables or features $w_0 \ldots w_S$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_S \end{bmatrix} \tag{3.11}$$

and a scalar function $c(\boldsymbol{w})$ made of these variables. The gradient of the scalar function $c(\boldsymbol{w})$ is a vector composed of the partial derivatives[1] with respect to each variable or feature:

$$\frac{\partial c(\boldsymbol{w})}{\partial \boldsymbol{w}} = \nabla c(\boldsymbol{w}) = \begin{bmatrix} \dfrac{\partial c(\boldsymbol{w})}{\partial w_0} \\ \dfrac{\partial c(\boldsymbol{w})}{\partial w_1} \\ \vdots \\ \dfrac{\partial c(\boldsymbol{w})}{\partial w_S} \end{bmatrix} \tag{3.12}$$

where the symbol ∇ is shorthand notation for the gradient and the symbol ∂ denotes the partial derivative.

Example: Consider the following function with two variables, w_0 and w_1

$$c(\boldsymbol{w}) = (w_0)^2 + 2(w_1)^2 + 1 \tag{3.13}$$

The independent variable, $\boldsymbol{w}$, and gradient of the function, $c(\boldsymbol{w})$, can be written in matrix form as

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{3.14a}$$

$$\nabla c(\boldsymbol{w}) = \begin{bmatrix} \dfrac{\partial c(\boldsymbol{w})}{\partial w_0} \\ \dfrac{\partial c(\boldsymbol{w})}{\partial w_1} \end{bmatrix} = \begin{bmatrix} 2w_0 \\ 4w_1 \end{bmatrix} \tag{3.14b}$$

[1] Partial derivatives are derivatives taken with respect to one variable while holding all other variables constant.

The point w^* where the gradient equals zero is found by setting the gradient equal to zero

$$\begin{bmatrix} 2w_0 \\ 4w_1 \end{bmatrix} = 0 \rightarrow \boldsymbol{w}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.15}$$

3.1.6 *Gradient Descent (Advanced Topic: Necessary for Data Science)*

Computing the minimum or maximum of a function by setting the gradient equal to zero works well for basic functions but is not efficient for higher order functions. *Gradient descent* is a more efficient way to determine the minimum of a higher order function.

The minimum of a cost function, $c(w)$, can be determined through an explicit update algorithm. The process begins by writing an explicit update equation for the independent variable, w, as

$$w^{k+1} = w^k - \alpha \frac{dc\left(w^k\right)}{dw} \tag{3.16}$$

where w^{k+1} is the value of w to be computed at the next step, w^k is the current value of w, and $\frac{dc\left(w^k\right)}{dw}$ is the derivative of the cost function evaluated at the current time step. The parameter, α, is a user-defined learning rate. This is illustrated using the function plotted in Fig. 3.6. The gradient descent algorithm is shown in the following five steps:

1. Select an arbitrary starting point w^0
2. Find the derivative of the cost function $c(w)$ at w^0
3. Descend to the next point through the gradient descent equation $w^1 = w^0 - \alpha \frac{dc\left(w^0\right)}{dw}$
4. Repeat the process for the next point $w^2 = w^1 - \alpha \frac{dc\left(w^1\right)}{dw}$
5. Continue doing so until the minimum is reached (i.e. negligible change in w) (Fig. 3.7)

   Example: Consider a function $g(w) = \frac{1}{50}\left((w)^4 + (w)^2 + 10w\right)$

1. Start at $w^0 = 2$ and use $\alpha = 1$
2. Take the derivative $\frac{dg(w)}{dw} = \frac{1}{50}\left(4(w)^3 + 2w + 10\right)$ and evaluate at the current position of w, $\frac{dg(2)}{dw}$
3. Use the gradient descent formula to calculate w at the next step.

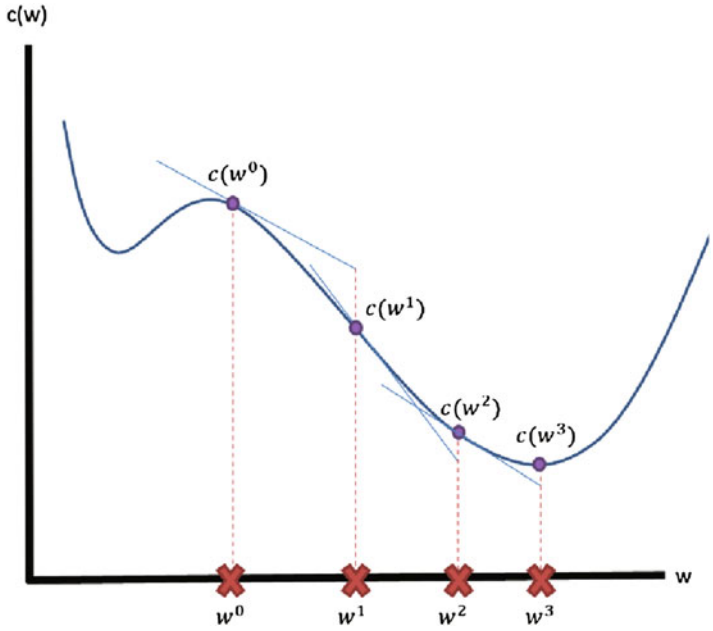


Fig. 3.6 Gradient descent methodology. A video is available in the E-book, Supplementary Video 3.2

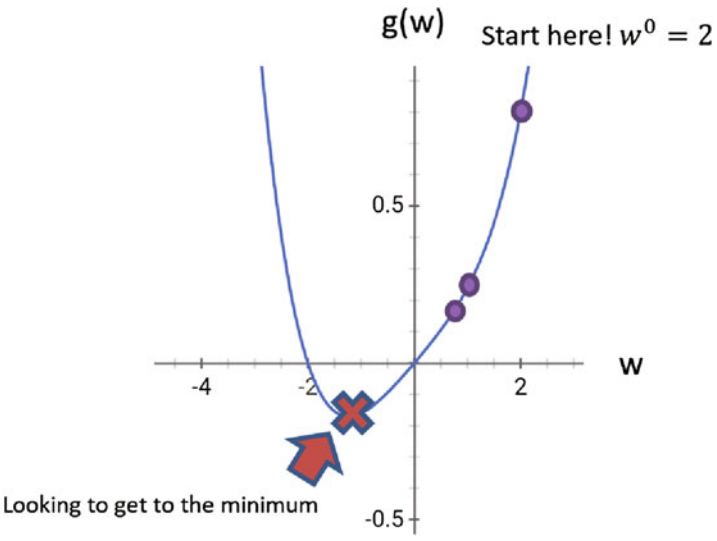


Fig. 3.7 Gradient descent example for $g(w) = \frac{1}{50}\left((w)^4 + (w)^2 + 10w\right)$

$$w^1 = w^0 - \alpha \frac{dg(w^0)}{dw} = 2 - 1\frac{dg(2)}{dw} = 1.08$$

4. Repeat the process for the next step, $w^2 = w^1 - \alpha \frac{dg(w^1)}{dw} = 1.08 - 1\frac{dg(1.08)}{dw} = .736$
5. Continue until the minimum is reached at $g(w) = -0.170$

When using gradient descent method for higher dimensions, the explicit update formula is written as

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \alpha \nabla c\left(\boldsymbol{w}^k\right) \tag{3.17}$$

where the univariate derivative has been replaced with the gradient and the independent variable $\boldsymbol{w}$ is now a vector. The gradient descent steps in multiple dimensions are

1. Start at an arbitrary vector $\boldsymbol{w}^0$
2. Find the gradient of function c ($\boldsymbol{w}$) at $\boldsymbol{w}^0$
3. Descend to the next point using the gradient descent equation $\boldsymbol{w}^1 = \boldsymbol{w}^0 - \alpha \nabla c(\boldsymbol{w}^0)$
4. Repeat the process for $\boldsymbol{w}^2 = \boldsymbol{w}^1 - \alpha \nabla c(\boldsymbol{w}^1)$
5. Continue until minimum is reached (negligible change in $\boldsymbol{w}$).

3.1.7 *Example: "Moneyball": Data Science for Optimizing a Baseball Team Roster*

Baseball is a game in which tradition is strong and data and statistics carry great weight. This allows baseball fans to compare the careers of Ty Cobb in 1911 to Pete Rose in 1968 (or anyone else for that matter). Historically, the worth of a player was largely dictated by their batting average (how many hits compared to how many time batting) and runs batted in (how many runners already on base were able to score when the batter hit the ball). However, through the use of data science, a new trend emerged (Fig. 3.8).

The game of baseball is played with 9 players from one team in the field playing defense (Fig. 3.8). A pitcher throws a baseball toward home plate where a batter standing next to home plate tries to hit the ball out into the field and then run to first base. If the batter hits the ball and makes it to first base before the ball is caught or picked up and thrown to first base, then the batter is awarded a hit and allowed to stay on the base, becoming a base runner. If the ball is caught in the air, picked up and thrown to first base before the batter arrives, or the batter is tagged when running to first base then the batter is out. The runner can advance to second, third, and home bases as other batters get hits. When the runner reaches home base, the team is awarded a run. The batting team continues to bat until they make three outs, at which point they go out to the field and the team in the field goes to bat.

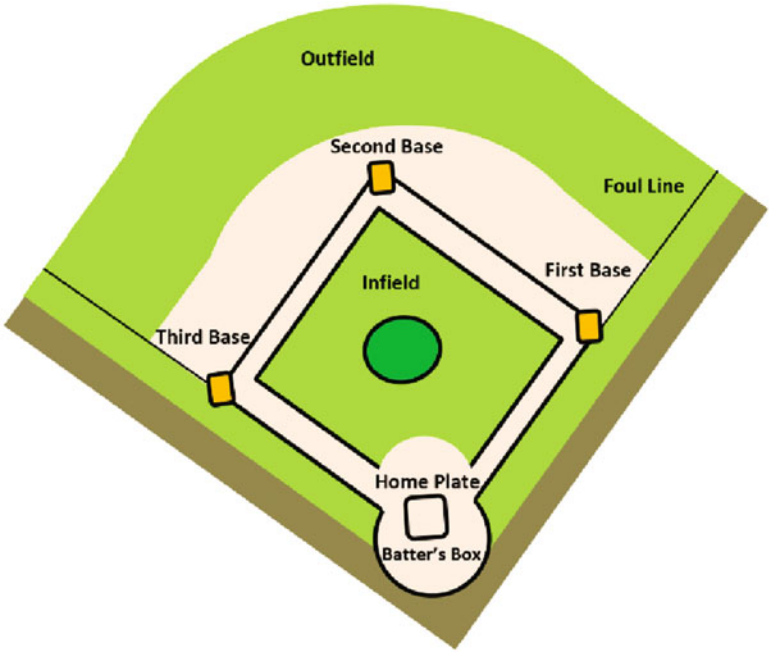


Fig. 3.8 Baseball field (https://entertainment.howstuffworks.com/baseball2.htm)

As mentioned previously, batting average (BA) and runs batted in (RBI) have traditionally been a very important statistic for baseball teams to evaluate the worth of a player. Players with high BA's and high RBI's were paid very large salaries by the richest teams (usually large market teams) and the small market teams had trouble competing.

In 2002, Billy Beane, the general manager of the Oakland Athletics, utilized data science to build a competitive team. Although Major League Baseball (MLB) generates around $10 billion in annual revenue, the smaller market MLB teams have much lower budgets with which to recruit and sign players. In 2002, Oakland A's general manager, Billy Beane, found himself in a tough situation because of this. The Oakland A's were a small market team without a large budget for player salaries. Beane, and his capable data science assistant, Paul DePodesta, analyzed baseball data from previous seasons and determined that they needed to win 95 games to make the playoffs. To achieve this goal, they estimated they needed to score 133 more runs than their opponents. The question they had to answer was "what data should they focus on".

To build a competitive team, Beane and DePodesta looked at a combination of a player's on-base percentage (OBP), which is the percentage a batter reaches base, and the slugging percentage (SLG), which is a measure of how many bases a batter is able to reach for a hit. In formulaic terms: $SLG = (1B + 2B * 2 + 3B * 3 + HR * 4)/AB$, where $1B$, $2B$, *and* $3B$ are first, second, and third base, respectively, HR is a

"home run", and *AB* is an "at bat". Through these two measures, it is possible to assess how often a player is getting on base in any possible way (and thus in a position to score) and how far they go each time they hit the ball.

It is possible to show through linear regression that SLG and OBP provide a good correlation with runs scored (RS). Using a moneyball baseball dataset available from Kaggle (https://www.kaggle.com/wduckett/moneyball-mlb-stats-19622012/data), a regression analysis was performed to compare the number or runs scores as a function of the batting average, and then as a function of the on base percentage and slugging percentage. A sampling of the moneyball data used for the analysis is shown below (Table 3.1).

A linear regression analysis was first performed on the RS vs. BA. The results, which are plotted in the Fig. 3.10, showed that the correlation between the RS and BA was only 0.69. BA was deemed a marginally useful statistic because it does not account for players hitting singles versus home runs and does not account for players getting on base by walks or being hit by a pitch.

By contrast, a linear regression between the RS and OBP shows a correlation of 0.82. OBP accounts for all the ways a player can get on base, and as such, provides a more meaningful measure of the number or runs scored than does the batting average.

Finally, a multivariate linear regression was performed with the RS vs the OBP and the SLG. The results of this linear regression showed a correlation of 0.93, meaning that OBP combined with SLG provided a better indicator or run scoring performance than BA or the OBP by itself. It should be noted that the linear combination of OBP and SLG is called On-base Plus Slugging (OPS), and is a commonly used baseball statistic in the game today ($OPS = OBP + SLG$). With this measure of OPS, the amount of time a player reaches base is accounted for as well as how many bases they are able to reach when they do get on base.

Using these data science techniques, Beane and DePodesta and the Oakland A's were able to win 103 games in 2002 (including a record-setting 20-game win streak), finish in first place, and make the playoffs. Today, OPS and OBP and SLG are some of the most closely watched baseball statistics by baseball insiders and fans alike.

3.1.7.1 Moneyball Regression Analysis Steps

Step 1: Multimodal Data Generation and Collection

Baseball statistics are readily available. One such database is in Kaggle (a sample of data is shown in Table 3.1).

Step 2: Feature Engineering

Various features are present in baseball sports analytics. However, we will restrict to team averaged stats for a few indicators. These include runs scored (RS), wins (W),

Table 3.1 A sample of baseball statistical data from Kaggle (https://www.kaggle.com/wduckett/moneyball-mlb-stats-19622012/data)

Team	League	Year	RS	RA	W	OBP	SLG	BA	Playoffs	Rank season	Rank playoff	G	OOBP	OSLG
ARI	NL	2012	734	688	81	0.328	0.418	0.259	0			162	0.317	0.415
ATL	NL	2012	700	600	94	0.32	0.389	0.247	1	4	5	162	0.306	0.378
BAL	AL	2012	712	705	93	0.311	0.417	0.247	1	5	4	162	0.315	0.403
BOS	AL	2012	734	806	69	0.315	0.415	0.26	0			162	0.331	0.428
CHC	NL	2012	613	759	61	0.302	0.378	0.24	0			162	0.335	0.424
CHW	AL	2012	748	676	85	0.318	0.422	0.255	0			162	0.319	0.405

on base percentage (OBP), slugging percentage (SLG), and on base plus slugging (OPS). RS corresponds to how much a team scores. W is the number of games a team wins in a season. OBP corresponds to how frequently a batter reaches base. SLG corresponds to the total number of bases a player records per at-bat. OPS is the sum of OBP and SLG.

Step 3: Dimension Reduction

Reduce the dimension of the problem by only considering the aforementioned stats/features.

Step 4: Reduced Order Modeling

Reduce the order of the model by assuming it is linear for the purposes of this demonstration.

Step 5: Regression and Classification

Use regression to determine model parameters and decide whether a linear hypothesis is adequate and offers any insight.

Module 6: System and Design

Use OBP and SLG to predict performance (RS) and, therefore, potential recruitment.

Returning to the baseball example, it is possible to find a relationship between OBP and RS (Fig. 3.9). Use the cost function

$$c(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(w_0 + w_1 x_n - y_n)^2 \tag{3.18a}$$

$$c(\boldsymbol{w}) = \frac{1}{N}\left[(w_0 + w_1 0.327 - 691)^2 + (w_0 + w_1 0.341 - 818)^2 + \ldots\right] \tag{3.18b}$$

and the gradient descent equation

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \alpha \nabla c\left(\boldsymbol{w}^{k-1}\right) \tag{3.19}$$

to find the optimal weights $\boldsymbol{w}$ for the model.

Figure 3.10 shows the regression model results for RS vs. BA, OBP, SLG, and OPS. There is a good correlation between the runs scored and the on base

Fig. 3.9 Sample data for runs scored (RS) and on base percentage (OBP)

	y_n	x_n
n	RS	OBP
	691	0.327
	818	0.341
	729	0.324
	687	0.319
	772	0.334
	777	0.336
	798	0.334
	735	0.324
	897	0.35

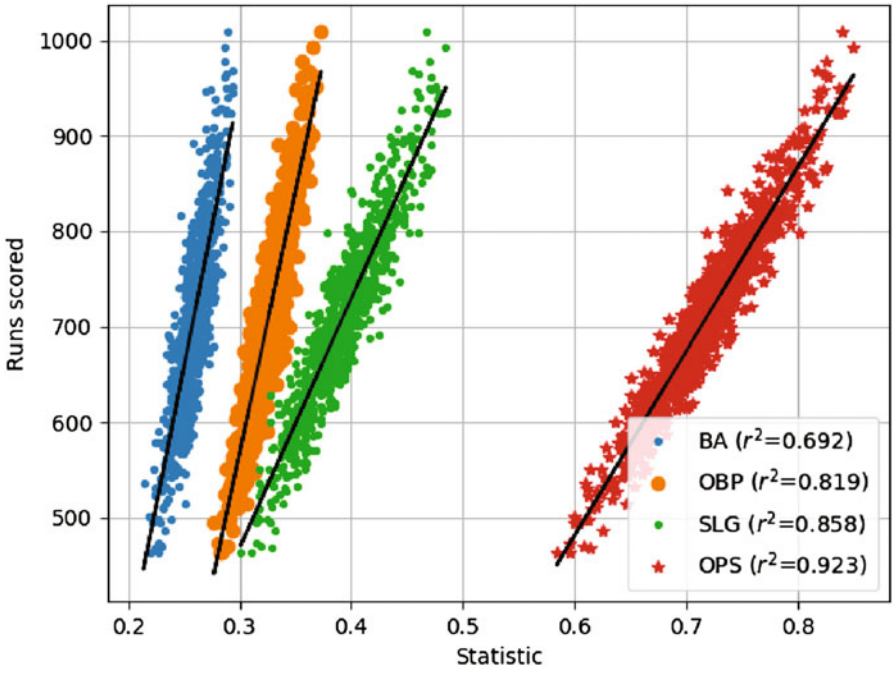


Fig. 3.10 Moneyball analysis results for runs scored (RS) vs. four different statistic—batting average (BA), on base percentage (OBP), slugging percentage (SLG), and on base plus slugging (OPS)

percentage. The correlation can be quantified by the r^2 value, with r^2 closer to 1.0 indicating a better the correlation of the linear regression to the data. Figure 3.10 also shows the regression for RS and SLG and RS and BA. Historically, BA was used by baseball scouts to recruit potential players. Billy Beane was correct in using OBP

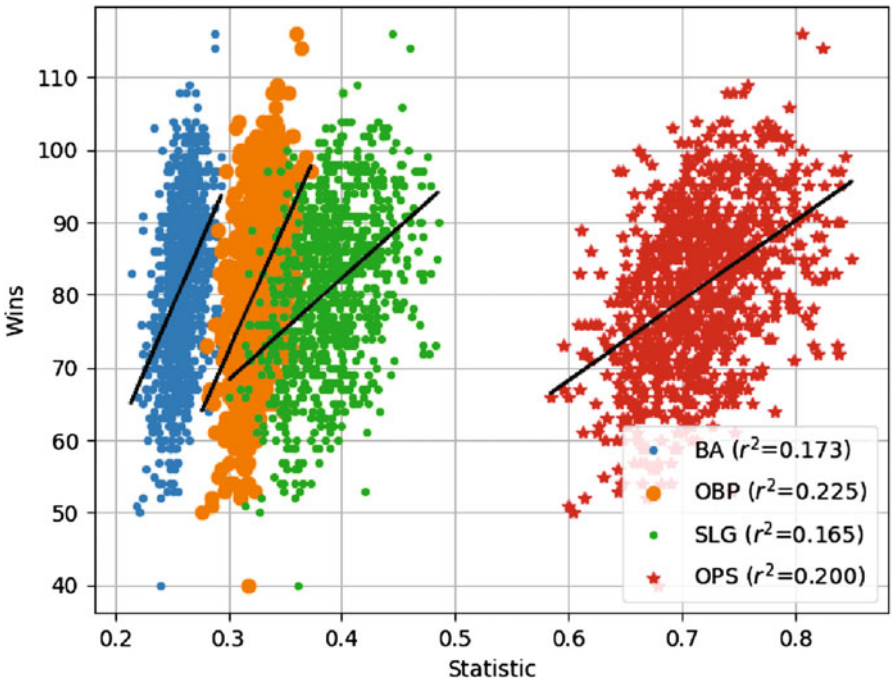


Fig. 3.11 Regression analysis results for wins (W) vs. four different statistic—batting average (BA), on base percentage (OBP), slugging percentage (SLG), and on base plus slugging (OPS)

and SLG (or a combination of them) instead of BA to do his recruitment as these statistics have a better correlation with RS.

One other logical question is why not perform the analysis based on wins (W) instead of just runs scores (RS) since that is the ultimate metric. As can be seen in Fig. 3.11, the correlation between W and any of these statistics is not good. These are only offensive statistics and do not account for pitching and defense, which are other important parts of winning baseball games.

The regression analysis performed thus far has been performed using one statistical variable at a time. It is possible to perform linear regression using multiple variables, such as performing a linear regression of RS versus both OBP and SLG. Using two variables for linear regression will result in a planar fit through the data instead of a straight line (Fig. 3.12).

Note that the dependent variable y_n (runs scored) depends on a vector of independent variables $\boldsymbol{x}_n$ (baseball statistics such as OBP and SLG)

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n) \tag{3.20}$$

For a multivariate linear regression, the model equation is

$$w_0 + w_1 x_{1,n} + w_2 x_{2,n} + \ldots + w_S x_{S,n} \approx y_n \text{ for } n = 1, \ldots, N \tag{3.21}$$

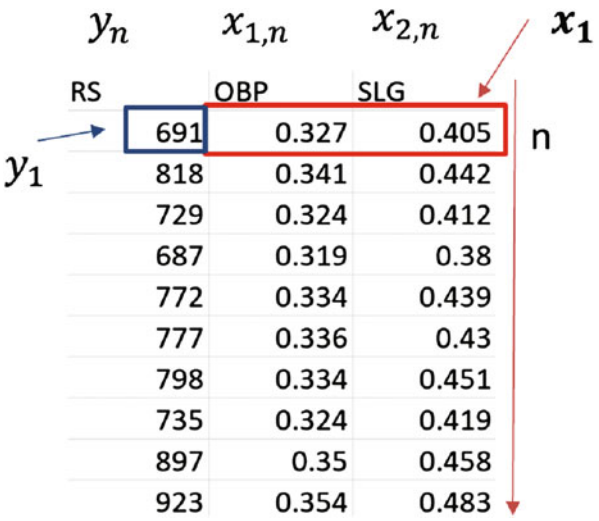


Fig. 3.12 Sample data for runs scores (RS), on base percentage (OBP) and slugging percentage (SLG)

Note that for this specific problem of performing a linear regression of RS with OBP and SLG, the above equation reduces to

$$w_0 + w_1 OBP_n + w_2 SLG_n \approx RS_n \text{ for } n = 1, \ldots, N \tag{3.22}$$

However, for generality, the arbitrary form is still used. Use the two following vectors to compact the equation:

$$\widehat{\boldsymbol{x}}_n = \begin{bmatrix} 1 \\ x_{1,n} \\ x_{2,n} \\ x_{3,n} \\ \vdots \\ x_{S,n} \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_S \end{bmatrix} \tag{3.23}$$

The model can be written in matrix notation as

$$\widehat{\boldsymbol{x}}_n{}^T \boldsymbol{w} \approx y_n \text{ for } n = 1, \ldots, N \tag{3.24}$$

After summing the squared differences and dividing by the number of points, the following cost function is obtained

$$c(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \left(\widehat{\boldsymbol{x}}_n{}^T \boldsymbol{w} - y_n \right)^2 \tag{3.25}$$

The gradient of the cost function is

$$\nabla c = \frac{2}{N}\sum_{n=1}^{N}\left(\widehat{\boldsymbol{x}}_n{}^T\boldsymbol{w} - y_n\right) * \widehat{\boldsymbol{x}}_n \tag{3.26}$$

Using the gradient descent method to find the minimum weights leads to

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \alpha\nabla c\left(\boldsymbol{w}^{k-1}\right) \tag{3.27a}$$

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \alpha\frac{2}{N}\sum_{n=1}^{N}\left(\widehat{\boldsymbol{x}}_n{}^T\boldsymbol{w}^{k-1} - y_n\right) * \widehat{\boldsymbol{x}}_n \tag{3.27b}$$

Applying the gradient descent method to determine the weights and bias results in

$$\boldsymbol{RS} = -803 + 2729 * \boldsymbol{OBP} + 1587 * \boldsymbol{SLG} \tag{3.28}$$

with $r^2 = 0.93$ (Fig. 3.13).

It is interesting to note that this result is essentially identical to the linear regression result between RS and OPS shown in Fig. 3.10. The OPS variable is the On Base Plus Slugging, and as the name implies, it is equal to the sum of the OBP and the SLG. As such, the results are the same when the analysis is performed either way.

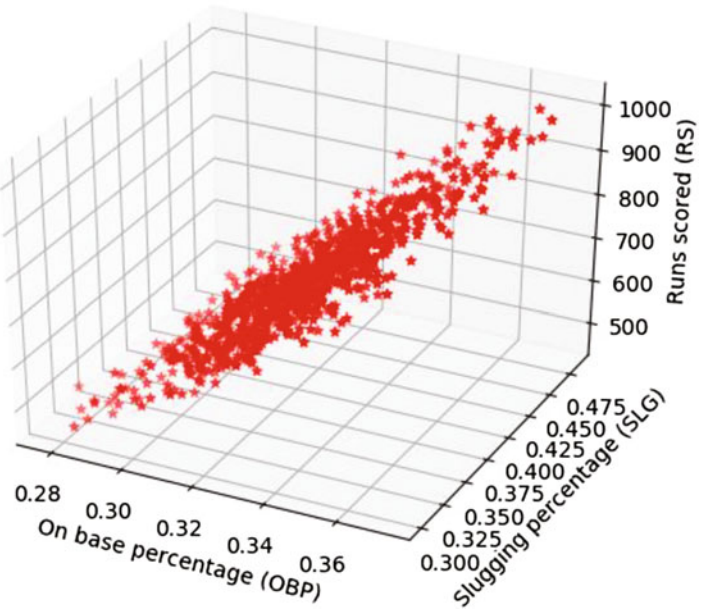


Fig. 3.13 Moneyball analysis results in 3D plot format for RS vs OBP and SLG

As a result of these various regression models, it can be seen that Billy Beane's hypothesis of evaluating players based on their OBP and SLG is more accurate in terms of their offensive potential to score runs. Using these data science techniques, Beane's Oakland A's were able to win 103 games in 2002 (including a record-setting 20-game win streak), finish in first place, and make the playoffs. Today, OPS and OBP and SLG are some of the most closely watched baseball statistics by baseball insiders and fans alike.

3.1.8 Example: Indentation for Material Hardness and Strength

Stress (σ) is the distribution of force in a material with a load applied to it. Strain (ε) is the relative displacement of a material that results from an applied load. For a uniform rod of material being pulled in tension, the stress is equal to the applied force divided by the cross-sectional area and the strain is equal to the change in length divided by the original length. A plot of the stress versus strain demonstrates some useful material relationships for material performance when loaded. To understand the relevance of stress in material deformation picture this scenario. Two people of equal weight step on your foot. One person is wearing sneakers while the other is wearing heels. Which case would hurt most? Getting stepped on with heels will hurt more since the force (equal in both cases) is concentrated of a smaller area (the heel piece), therefore having a higher stress.

An example of a stress vs. strain curve for a metal is shown in Fig. 3.14. At first the sample is not loaded a. Then comes the initial linear portion, known as Young's modulus, is the elastic, or recoverable, part of the curve b – when the load is removed, both the stress and strain return to zero in this region. The yield strength

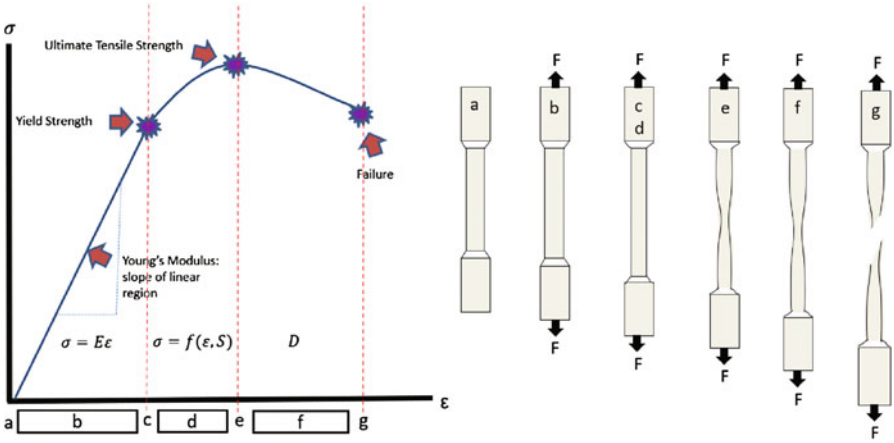


Fig. 3.14 Typical stress vs. strain curve for a metal

c is at the upper end of the linear portion of the curve and defines the onset of permanent (or plastic) deformation (nonlinear function between stress and strain/state variables S) d-g – when loaded beyond this point and then unloaded some permanent shape change will occur. In engineering metals the yield strength is calculated by offsetting the Young's modulus line 0.2% percent. The peak of the curve is the ultimate tensile strength e, which is the stress when the peak load is applied. After the ultimate tensile strength is the region where neck (or visible deformation) f occurs (at which point some damage D occurs), and finally, the fracture point g. Note that the parts of the curve after the yield point are generally not linear but can be approximated as piecewise linear.

Indentation is an experimental method to measure the hardness of a material, or its the resistance to plastic deformation. This test is done using an indenter of a pre-determined shape, such as hemispherical or diamond-shaped. The indenter is pressed into the surface of a material with a specified force. One such hardness test is the Vickers Hardness (HV) test, which uses a diamond-shaped indenter. After the indentation mark is made, the average diagonal distance is measured and the Vickers Hardness is computed as

$$HV = \frac{F}{A} = \frac{F}{\frac{d^2}{2\sin(68)}} = 1.8544\frac{F}{d^2} \tag{3.29}$$

where F is applied force, A is the surface area of the indentation, and d is the average diagonal dimension of the diamond shaped indentation. As shown in Fig. 2.10, the Vickers indenter is diamond-shaped, with the faces making a 68° angle from the indentation axis. It has been shown that for some materials, HV and ultimate tensile strength (TS) are well correlated.

3.1.9 Example: Vickers Hardness for Metallic Glasses and Ceramics

Vickers hardness measurements were reported for different material by Zhang et al. [3]. Some of the representative values for metallic glasses are shown in Fig. 3.15. The Vickers hardness vs. ultimate tensile strength for metallic glasses and ceramics is shown in Fig. 3.16. Inspection of the data shows that the measurements for the metallic glasses are generally oriented in a linear pattern, but the measurements for the ceramic materials are much more scattered.

Regression analysis using least squares optimization is performed on the data for the metallic glasses and the ceramics. Results show that a linear relationship works very well for metallic glasses ($r^2 = 0.949$), but not for ceramics ($r^2 = 0.0002$) for this data set (see Fig. 3.16). The difference in indentation results between these two types of materials is to be expected. Indentation is measuring the amount of force for local

Material	Ultimate Tensile Strength (Gpa)	Vickers Hardness (HV)
Zr52.5Ni14.6Al10Cu17.9Ti5	1.8	5.15
(Co0.942Fe0.058)69Nb3B22.4Si5.6	3.32	9.82
Zr55Cu30Al10Ni5	1.8	5
Pd40Ni40P20	1.78	5.6
Fe41Co7Cr15Mo14C15B6Y2	3.5	13.45
Fe74Ni9Cr4Si3B10	2.93	9.16
Fe66Ni7Zr6Cr8Si3B10	2	8.31
Fe63Ni7Zr6Cr8W3Si3B10	2.73	9.1
Zr53Cu30Ni9Al8	2.05	5.22
(Zr53Cu30Ni9Al8)99.75Si0.25	2.05	5.54
(Zr53Cu30Ni9Al8)99.5Si0.5	1.82	5.64
(Zr53Cu30Ni9Al8)99.25Si0.75	2.1	5.67
(Zr53Cu30Ni9Al8)99Si	1.75	5.82
Zr41.2Ti13.8Cu12.5Ni10Be22.5	1.95	5.95
Zr-400°C×5min	1.97	6.1

Fig. 3.15 Representative metallic glass material data from Zhang et al. [3]

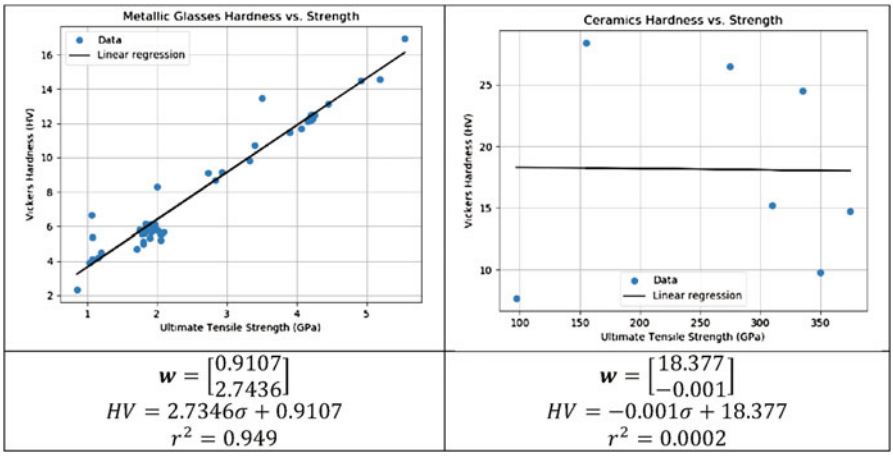


Fig. 3.16 Vickers Hardness (HV) vs ultimate tensile strength for metallic glass and ceramic materials

plastic flow on the surface. Metallic materials will demonstrate this type of deformation, but ceramics are generally brittle and will experience surface cracking and fragmentation instead of plastic deformation.

3.2 Nonlinear Regression

A simple straight line relationship often does not exist between two variables. In these cases, it is necessary to employ some form of regression capable of building a nonlinear relationship, such as piecewise linear regression analysis, moving average analysis, or a moving least squares regression analysis.

3.2.1 Piecewise Linear Regression

Piecewise linear regression is one of the most basic nonlinear regression techniques since it consists of subdividing a set of nonlinear data into a series of segments that are approximately linear. Once that is done, a linear regression can be done on these sections one by one. This can be illustrated by planning a route on a map. As shown in Fig. 3.17, if one were to plot a route from Chicago to Los Angeles, there is not a straight route to follow. Instead, there are roads filled with curves that go west for a long way, then roads that traverse in a west southwest direction for a long way, and finally roads that go in a southwest orientation for the remainder of the route. If a

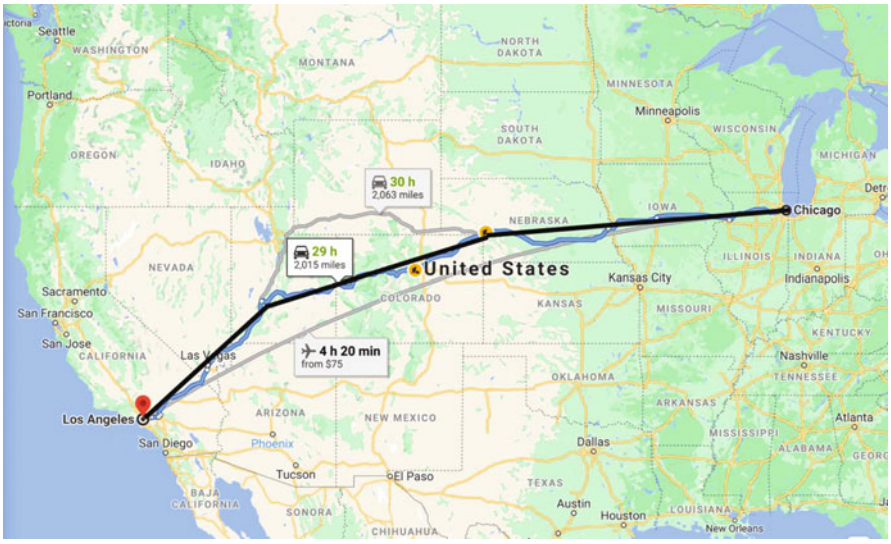


Fig. 3.17 Map from Chicago to Los Angeles with piecewise linear route overlaid

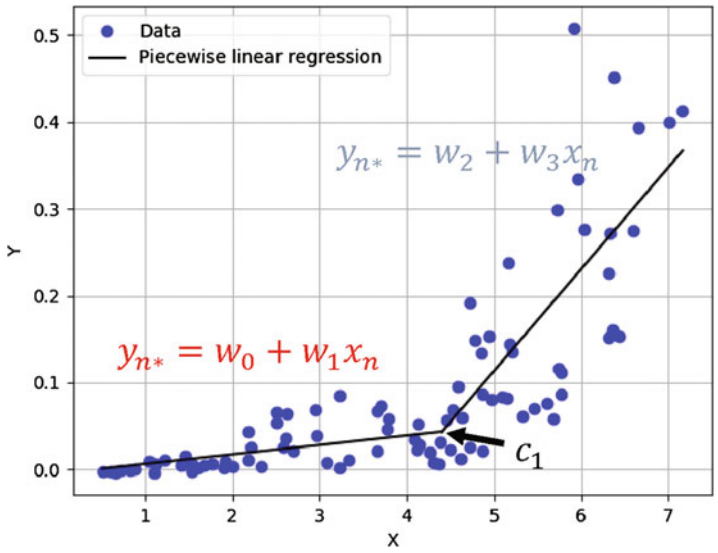


Fig. 3.18 Piecewise linear regression through data

person wanted to estimate the distance, the route could be broken into several linear segments (three in this case) to quickly estimate travel distance and compare routes.

A global piecewise linear regression equation can be developed as a series of line segments with continuity at the common points. Consider the following set of data with two straight lines in Fig. 3.18 fit through different parts of the data

$$\begin{aligned} y_{n*} &= w_0 + w_1 x_n \quad x < c_1 \\ y_{n*} &= w_2 + w_3 x_n \quad x > c_1 \end{aligned}$$

If the equation in red applies to the line left of point c_1 and the blue equation applies to the line right of c_1, at the common point $x = c_1$

$$w_0 + w_1 c_1 = w_2 + w_3 c_1 \tag{3.30}$$

which leads to

$$\begin{aligned} y_{n*} &= w_0 + w_1 x_n \quad x < c_1 \\ y_{n*} &= w_0 + (w_1 - w_3) c_1 + w_3 x_n \quad x > c_1 \end{aligned} \tag{3.31}$$

Using these equations, a global cost function can be written as

$$c(\boldsymbol{w}) = \frac{1}{N_1} \sum_{n=1}^{N_1} (y_{n*} - y_n)^2 + \frac{1}{N_2 - N_1} \sum_{n=N_1+1}^{N_2} (y_{n*} - y_n)^2 \tag{3.32}$$

which can be used for a least squares optimization for linear regression as discussed earlier in this chapter.

3.2.2 *Moving Average*

A moving average provides a good method to smooth out data and mute the effects of spikes in the data. This is a popular method for analyzing trends with stock prices in order to smooth out the effects of day to day movement of the stock price. As shown in Fig. 3.19, the price of the S&P 500 stock index goes up and down on a daily basis, but the overall trend is upward for the time period shown.

As such, if a financial analyst wanted to evaluate the long term performance of the stock, a moving average provides a good tool for doing so. In addition, evaluating different moving averages can also provide insight into stock trends. In Fig. 3.19, the 50 and 200 day moving averages smooth out the data to show the overall trend. The 200-day can also act as a "floor" or lower limit—buying opportunities exist when the price drops down to that level or below.

A basic form of a moving average is a *simple moving average*, which is computed by summing a quantity of interest over a range and dividing by the number of samples

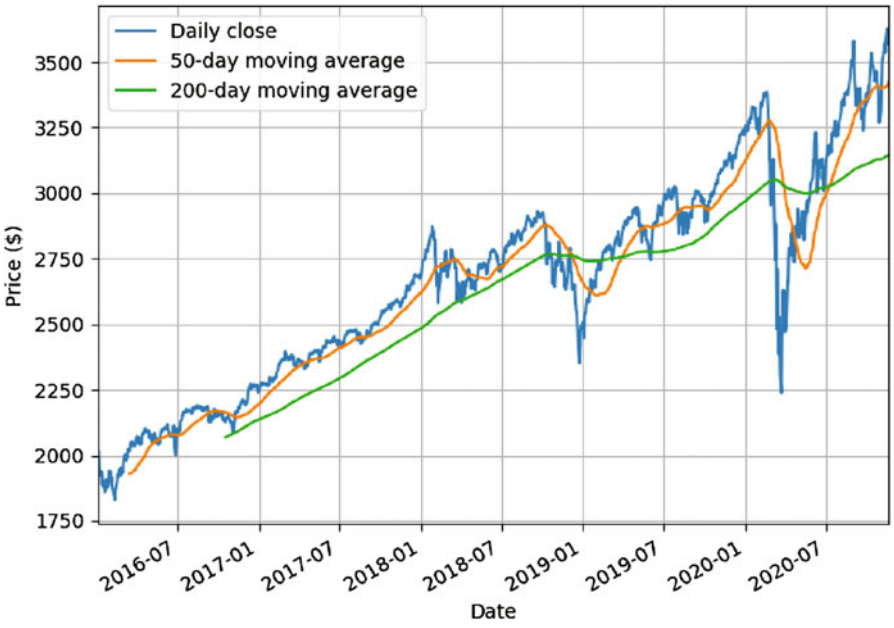


Fig. 3.19 S&P 500 stock index price with two rolling averages (50 and 200 days) overlaid. The spread between these moving averages provides insight into stock trends

```
start = '2016-01-01'
df2 = web.DataReader('^GSPC', 'yahoo',start)
df2.to_csv('gspc.csv')
df2['Close'].plot()
df2['Close'].rolling(50).mean().plot()
df2['Close'].rolling(200).mean().plot()
plt.legend(['Daily close','50-day moving average','200-day
moving average'])
plt.ylabel('Price ($)')
plt.show()
plt.grid()
```

Fig. 3.20 Sample Python code for rolling average in Fig. 3.19

$$SMA_c = \frac{\sum(P_c + P_{c-1} + ..P_{c-k})}{k} \tag{3.33}$$

where P_c is the stock price at time c and k is the number of points used for the averaging. It should be noted that performing a moving average on one variable of a plot results in the graph giving an appearance of being off of the original data. To prevent this effect, the averaged value can be plotted at the average of the independent variable instead of at the extent (Fig. 3.20).

3.2.3 Moving Least Squares (MLS) Regression

Moving least squares (MLS) regression is a technique to perform regression analysis on data that does not necessarily demonstrate a linear relationship between the variables. The technique is similar to the least squares linear regression already shown, but the inclusion of a weight function allows it to be performed point by point with the data weighted to the point being evaluated. In other words, the regression is "bent" to the data by only using a few points at a time. The weight function results in a localized point-by-point least square fit instead of a global least squares fit as shown previously. Common weight functions include bell-shaped curves such as cubic splines and truncated Gaussian functions.

An MLS curve fit through the Apple stock data is shown in Fig. 3.21. The original data (plotted with the blue curve) consisted of 252 data points. An MLS approximation was performed using only 45 evenly spaced points. The weight function used was a cubic spline with a coverage radius of 3 (approximately 3 points on each side of the point of interest were involved in each calculation). The results show that the MLS approximation with this set of parameters is able to accurately capture the trends of the data but results in smoothing the data similar to the moving average calculation shown earlier.

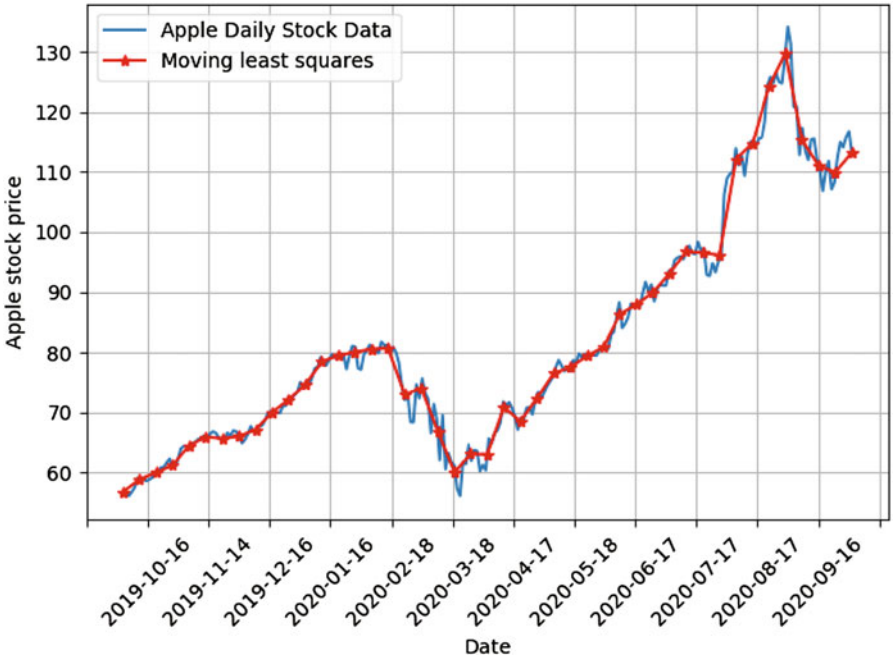


Fig. 3.21 Apple stock price with moving least squares (MLS) approximation overlaid

3.2.4 Example: Bacteria Growth

Recall the discussion on bacteria from Chap. 2. The growth of bacteria is an example of a nonlinear relationship. If food is left out at room temperature and not refrigerated, bacteria will begin to grow. As shown by the blue circles in Fig. 3.22, the number of bacteria will increase slowly at first (lag phase), but after some period of time the rate of increase will become much more rapid (exponential phase). Later on, the rate of increase will become much slower again (stationary phase). A simple linear regression shown by the red line in Fig. 3.22 obviously provides a poor fit to the data and would not provide a useful predictive tool. However, if the data can be fit using piecewise linear regression as shown by the green lines. As shown in Fig. 3.22, the lag and stationary phases can be fit using a single line for each, but the exponential phase requires at least two line segments due to the nonlinear nature of the bacteria growth during this phase.

Figure 3.23 shows the results for the application of a MLS approximation to the bacteria growth data. The original data contained 45 data points, but the MLS approximation was done using only 15 data points.

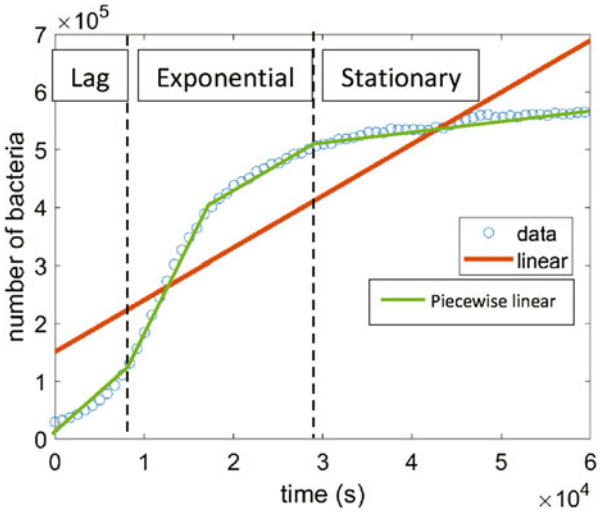


Fig. 3.22 Bacteria growth model

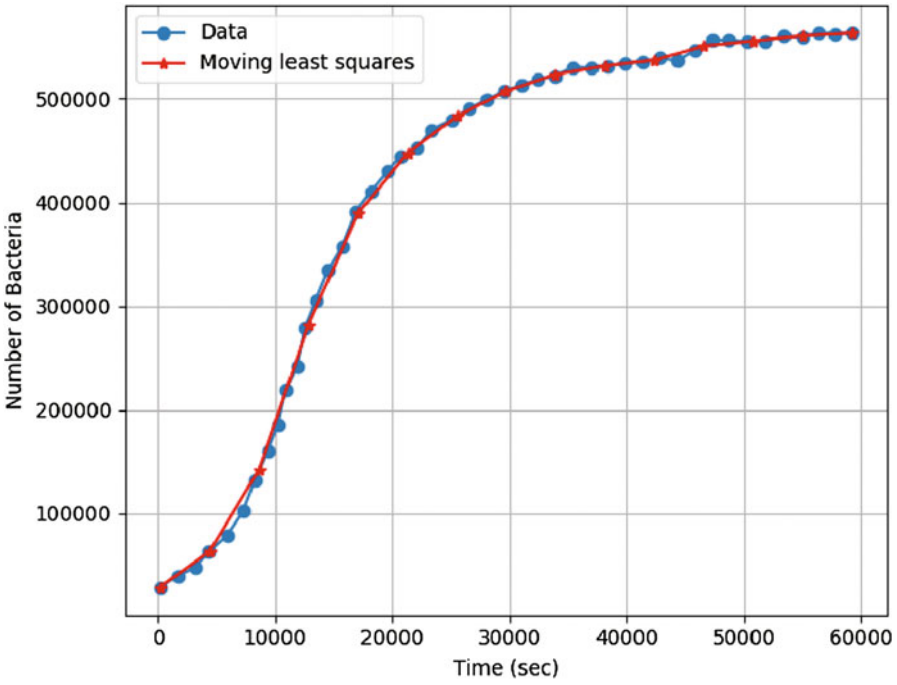


Fig. 3.23 Moving least squares (MLS) approximation of bacteria growth data

3.3 Regularization and Cross-Validation (Advanced Topic)

Nonlinear regression methods require complicated, higher order regression models to achieve accuracy. These methods may lose generality for the regression models. In order to find a good balance between model complexity and accuracy, regularization is introduced into the regression model. The regularized loss function introduces and extra term and is given by

$$\boldsymbol{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_{*n} - y_n)^2 + \lambda\|\boldsymbol{w}\|_p^p \tag{3.34}$$

where λ is a predefined regularization parameter with nonnegative value, a positive number,y_n is the original data, y_{*n} is the regression model, and N is the number of data points.

This equation shows that besides the first MSE term, a p-norm regularization term is added. This term acts as a "penalty" for having $\boldsymbol{w}$ be too large. In theory, the regularization term seeks to balance on the seesaw of simplicity and accuracy. The λ parameter is analogous to the fulcrum location of the seesaw. A larger λ implies more simplicity in the model, and smaller λ implies more accuracy in the model. If the

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Load team data
df = pd.read_csv('baseball.csv',sep=',').fillna(0)
df['OPS'] = df.OBP + df.SLG

df2002 = df.loc[df.Year < 2002]


# Linear regression for Runs scored

slBA, intBA, r_valBA, p_valBA, ste_errBA =
stats.linregress(df2002.BA,df2002.RS)
rsqBA = r_valBA**2
slOBP, intOBP, r_valOBP, p_valOBP, ste_errOBP =
stats.linregress(df2002.OBP,df2002.RS)
rsqOBP = r_valOBP**2
slSLG, intSLG, r_valSLG, p_valSLG, ste_errSLG =
stats.linregress(df2002.SLG,df2002.RS)
rsqSLG = r_valSLG**2
slOPS, intOPS, r_valOPS, p_valOPS, ste_errOPS =
stats.linregress(df2002.OPS,df2002.RS)
```

Fig. 3.24 Linear regression Python code for baseball example

```
rsqOPS = r_valOPS**2

plt.plot(df2002.BA,df2002.RS,'.',label='BA ($r^2$=%.3f)'
%rsqBA)
plt.plot(df2002.OBP,df2002.RS,'o',label='OBP ($r^2$=%.3f)'
%rsqOBP)
plt.plot(df2002.SLG,df2002.RS,'.',label='SLG ($r^2$=%.3f)'
%rsqSLG)
plt.plot(df2002.OPS,df2002.RS,'*',label='OPS ($r^2$=%.3f)'
%rsqOPS)
plt.xlabel('Statistic')
plt.ylabel('Runs scored')
plt.legend(loc='lower right')
plt.grid()



yBA = slBA*df2002.BA + intBA
plt.plot(df2002.BA,yBA,'k-')

yOBP = slOBP*df2002.OBP + intOBP
plt.plot(df2002.OBP,yOBP, 'k-')

ySLG = slSLG*df2002.SLG + intSLG
plt.plot(df2002.SLG,ySLG, 'k-')

yOPS = slOPS*df2002.OPS + intOPS
plt.plot(df2002.OPS,yOPS, 'k-')

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df2002.OBP,df2002.SLG,df2002.RS,marker='*',color='r'
)
ax.set_xlabel('On base percentage (OBP)')
ax.set_ylabel('Slugging percentage (SLG)')
ax.set_zlabel('Runs scored (RS)')

x = df2002.OBP
y = df2002.SLG
x,y = np.meshgrid(x,y)
z = -803 + 2729*x + 1587*y

# Linear regression for Wins

slWBA, intWBA, r_valWBA, p_valWBA, ste_errWBA =
stats.linregress(df2002.BA,df2002.W)
```

Fig. 3.24 (continued)

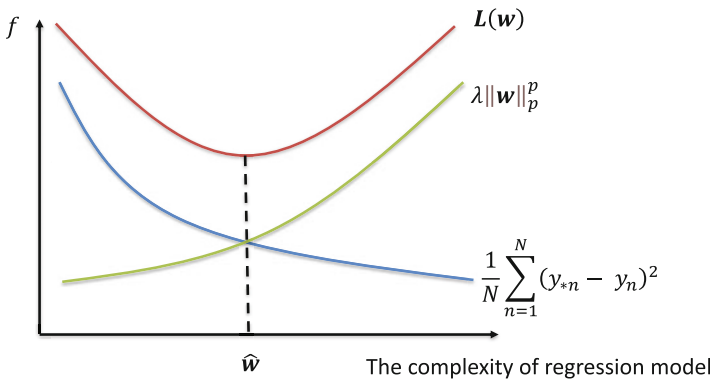


Fig. 3.25 How the regularization term balances model complicity and accuracy

regularization parameter is $\lambda = 0$, the problem is a standard least square regression problem, (i.e., regression without regularization). The choice of λ will be introduced in Sect. 3.3.4. A diagram of how the regularization term balances model complicity and accuracy is shown in Fig. 3.25. With the increase of the model complexity, the MSE term typically decreases while the regularization term increases. The sum of them can achieve a minimum when an appropriate model complexity is selected. Two commonly used regularization approaches are the L1 and L2 norm regularized regression methods.

3.3.1 Review of the Lp-Norm

The Lp-norm is a measure of a vector size, which is defined as the p-th root of the sum of the p-th-powers of the absolute values of the vector components

$$\|\boldsymbol{w}\|_p = \left(\sum\nolimits_{i=1}^{N} |w_j|^p\right)^{1/p} \tag{3.35}$$

Example:

$$\text{If } p = 1, \|\boldsymbol{w}\|_{p=1} = \sum\nolimits_{i=1}^{N} \mid w_j \mid = |w_1| + |w_2| + \ldots + \mid w_N \mid \tag{3.36}$$

$$\text{If } p = 2, \|\boldsymbol{w}\|_{p=2} = \left(\sum\nolimits_{i=1}^{N} |w_j|^2\right)^{1/2} = \sqrt{|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2} \tag{3.37}$$

For the limit case when $p = \infty$,

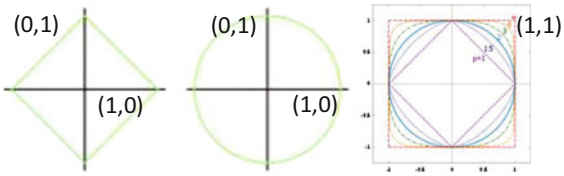


Fig. 3.26 Geometric interpretation of L1, L2 and ∞ norm (https://en.wikipedia.org/wiki/Lp_space)

$$\|w\|_\infty = \max\left(\left|w_j\right|\right) = \max\left(|w_1|, |w_2|, \ldots, |w_N|\right) \tag{3.38}$$

The shape of Lp-norm is illustrated for a vector containing only two components, i.e. $N = 2$. For comparison, it is required that $\|\boldsymbol{w}\|_p = 1$.

$$\|\boldsymbol{w}\|_{p=1} = 1 \; \|\boldsymbol{w}\|_{p=2} = 1 \; \|\boldsymbol{w}\|_{p=\infty} = 1$$

The contour indicates for the possible points (w_1, w_2) that satisfy $\|\boldsymbol{w}\|_p = 1$. In general, as p approaches infinity, the contour approaches a square, that is $\|\boldsymbol{w}\|_\infty$ (Fig. 3.26).

3.3.2 *L1-Norm Regularized Regression*

The L1-norm can be used to relieve overfitting by adding a regularization term to the regression equation. A two-dimensional polynomial example is used to demonstrate the recovery of the true function with independently and identically distributed Gaussian noise for each term

$$y_* = 1(x + \epsilon_1)^5 - 4(x + \epsilon_2)^2 - 5(x + \epsilon_3) \tag{3.39}$$

where y_* is the simulated data, and the noise ϵ_i~*Normal* (0, 0.05) for all $i = 1; 2;$ 3. To simulate noise, 81 linearly spaced x-coordinates are spaced between interval [−2,2]. The noise data are used to define uncertainties and reflect inaccuracies in measurements.

At the beginning of this section, the general form of regularized regression function is given in Eq. (3.34). For $p = 1$, the expression of L1-norm regularized regression is

$$\boldsymbol{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(y_{*n} - y_n)^2 + \lambda\|\boldsymbol{w}\|_{p=1}^1 \tag{3.40}$$

A comparison showing the advantages of regularized regression is shown for two regressive methods, which are tested and recover the correct function in Eq. (3.39). Firstly, for non-regularized regression, the mean square error (MSE) is used to find weights $w_0, \ldots, w_5$ in polynomial $y_n = w_0 + w_1 x_n + w_2 {x_n}^2 + \ldots + w_5 {x_n}^5$. Secondly,

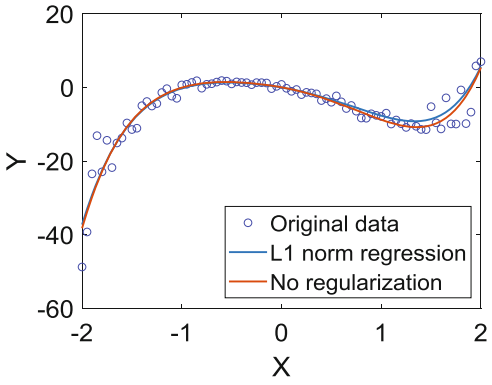


Fig. 3.27 Comparisons between L1-norm regularization regression and non-regularized regression

regularization term with L1-norm is added to find weights of L1-norm regularized regression. The results are shown in Fig. 3.27.

The blue dots represent the original data based on Eq. 3.39, the blue line is L1-norm regularized regression result, and the red line is non-regularized regression result. The regression model obtained by the regularization is

$$y = 0.9671x^5 - 3.9175x^2 - 4.6723x \text{ with } \lambda = 0.074 \tag{3.41}$$

The regression without regularization is

$$y = 1.0498x^5 + 0.2190x^4 - 0.1003x^3 - 4.5166x^2 - 4.8456x - 0.0018 \tag{3.42}$$

Results show that both the models fit the data very well. However, the predicted functions are very different. The main difference is that L1 regularization can eliminate some high order terms in the regression model (w_0, w_3, w_4 equal to zero). This shows sparsity and thus can be used for feature selection (or model selection), e.g., the order of x. Also, note that in this problem, an appropriate value of λ is 0.074. One approach of choosing appropriate λ is through the K-fold cross-validation, details will be introduced in Sect. 3.3.4.

3.3.3 L2-Norm Regularized Regression

Similarly, L2-norm can be used instead of L1-norm for the regularization term. The L2-norm regularized regression is defined as

$$\boldsymbol{L}(\boldsymbol{\omega}) = \frac{1}{N}\sum_{n=1}^{N}(y_{*n} - y_n)^2 + \lambda\|\boldsymbol{\omega}\|_{p=2}^2 \tag{3.43}$$

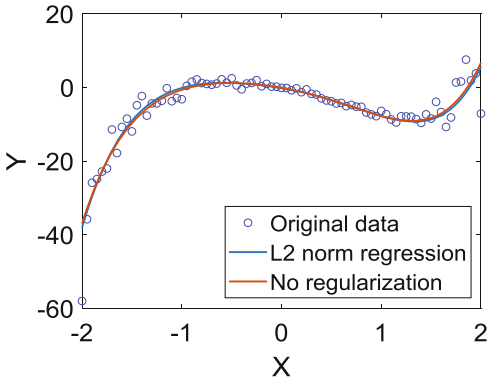


Fig. 3.28 Comparisons between L2-norm regularization regression and non-regularized regression

L2-norm uses the concept of "sum of squares", and thus has useful properties such as convexity, smoothness and differentiability. L2-norm regularized regression also has an analytical solution because of these properties. Using the same polynomial model y_{*n} for regression, the results are depicted in Fig. 3.28.

Similarly, the blue dots are generated as the original data based on Eq. (3.39), the blue line is L2-norm regularized regression result, and the red line is non-regularized regression result. The regression model obtained by the regularization is

$$y = 1.0119x^5 - 0.0800x^4 + 0.0495x^3 - 3.7368x^2 - 5.3544x - 0.0707 \text{ with } \lambda = 1 \quad (3.44)$$

The regression without regularization is the same as Eq. (3.42)

$$y = 1.0498x^5 + 0.2190x^4 - 0.1003x^3 - 4.5166x^2 - 4.8456x - 0.0018$$

Both models fit the data well, and the weights of L2-norm regularized regression are nonzero, but smaller than non-regularized regression weights (different from L1-norm regularized regression). That is, where the L1-regularized regression may squeeze sufficiently small coefficients to zero, it may omit the intricate details. The L2-norm seeks to capture those details, and thus L2-norm regression can preserve details and detect sophisticated patterns in data.

Comparing L1-norm and L2-norm regressions, it is found that L1-norm regression has a better performance in selecting key features of model. However, the L2-norm can preserve details and detect sophisticated patterns in data.

3.3.4 K-Fold Cross-Validation

Cross-validation, sometimes called rotation estimation or out-of-sample testing, is an important model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

Table 3.2 Divided data sets

Data set number	Training data	Test data
Set 1	Point 9 to 81	Point 1 to 8
Set 2	Point 1 to 8 and 17 to 81	Point 9 to 16
Set 3	Point 1 to 16 and 25 to 81	Point 17 to 24
...	...	...
Set 10	Point 1 to 72	Point 73 to 81

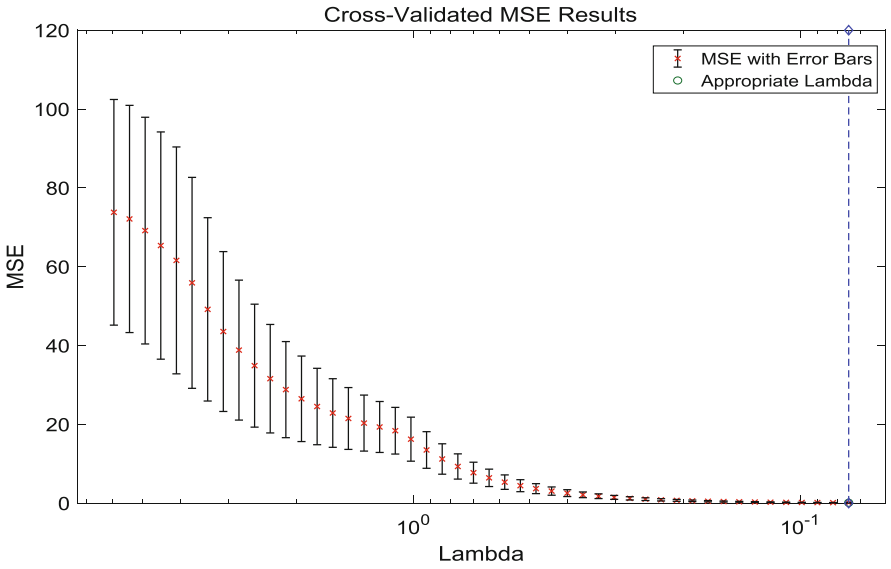


Fig. 3.29 Comparison result of different λ

One commonly used cross-validation method is K-fold cross-validation, in which the original sample is randomly partitioned into K equally sized subsamples. Consider the example in Sect. 3.3.2. Eighty-one (81) equally spaced x-coordinates are spaced between interval [−2,2] through Eq. (3.39). If K = 10-fold cross-validation is used, each data set has 8 data points (9 for the last data set). Of the K = 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data [4].[2] The cross-validation process is then repeated ten times, with each of the ten subsamples used exactly once as the validation data. The MSE for ten data sets can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. K = 10-fold cross-validation is commonly used [5], but in general K remains an unfixed parameter [6]. The divided data sets are in Table 3.2.

[2]Newbie question: confused about train, validation and test data! Archived from the original on 14 March 2015. Retrieved 14 November 2013.

The corresponding MSE of each set can be calculated. Thus, for those K = 10 sets, an error bar is computed with the corresponding mean value and standard deviation. It is used to evaluate the goodness of fit for the regression model. The error bar results of different regularization parameter λ are used to find an appropriate λ. For example, if 50 different λs are evaluated from 0.01 to 10, the comparison result is shown in Fig. 3.29. The appropriate λ value with the minimum MSE is 0.0745.

```
%% L1 and L2 norm regression example

%% Generation of data
clc
clear
x0 = -2:0.05:2; % 81 linearly spaced x-coordinates are spaced between
interval [-2,2]
n = length(x0); % The total number of data points (81)
x1 = x0+randn(1,n)*0.05; % x+epsilon1
x2 = x0+randn(1,n)*0.05; % x+epsilon2
x3 = x0+randn(1,n)*0.05; % x+epsilon3
x = [x1.^5;x0.^4;x0.^3;x2.^2;x3;ones(1,n)]';
weights = [1;0;0;-4;-5;0]; % Weights
y = x*weights; % Simulated data y = x^5 - 4*x^2 - 5*x

%% Regressions
[b_lasso,fitinfo] = lasso(x,y,'CV',10); % L1-norm regularized regression
lam = fitinfo.Index1SE; % Index of appropriate Lambda
b_lasso_opt = b_lasso(:,lam) % Weights for L1-norm regularized regression

lambda = 1; % Set lambda equals to 1 for L2-norm regularized regression
(You can also find an appropriate Lambda yourself)
b_ridge = (x'*x+lambda*eye(size(x, 2)))^-1*x'*y % Weights for L2-norm
regularized regression (Has analytical solution)

b_ols = polyfit(x1',y,5) % Weights for non-regularized regression
(Ordinary Least Squares)

xplot = [x0.^5;x0.^4;x0.^3;x0.^2;x0;ones(1,n)]';
y_lasso = xplot*b_lasso_opt; % L1 norm regression result
y_ols = xplot*b_ols'; % Non-regularized regression result
y_ridge = xplot*b_ridge; % L2 norm regression result

%% Plots
plot(x0,y,'bo') % Plot of origin data
hold on
plot(x0,y_lasso,'LineWidth',1) % Plot of L1 norm regression
hold on
plot(x0,y_ridge,'LineWidth',1) % Plot of L2 norm regression
hold on
plot(x0,y_ols,'LineWidth',1) % Plot of non-regularized regression
ylabel('Y','fontsize',20)
xlabel('X','fontsize',20)
legendset = legend('Original data','L1 norm regression','L2 norm
regression','Noregularization','location','southeast');
set(gca,'FontSize',20);
lassoPlot(b_lasso,fitinfo,'PlotType','CV'); % Cross-validated MSE
legend('show') % Show legend
```

Fig. 3.30 Matlab code for regularization regression

The steps to find the appropriate value of λ using K-fold cross-validation are summarized below:

1. For each regularization parameter, divide the original data set into K equal folds (parts).
2. Use one part as the test set and the rest as the training set.
3. Train the model and calculate the mean square error (MSE) with the test set.
4. Repeat steps 2 and 3 K times, and each time use a different section as the test set.
5. Compute the average and standard deviation of the set of MSE including K MSEs. Take the average accuracy as the final model accuracy.
6. Compare MSE for different λ values to find the appropriate regularization parameter.

The Matlab code for this section is given in Fig. 3.30.

3.4 Equations for Moving Least Squares (MLS) Approximation (Advanced Topic)

Consider an approximation function written as

$$y_{n*}(\mathbf{x}) = \mathbf{p}(\mathbf{x})^T \mathbf{a}(\mathbf{x})$$

where y_{n*} is an approximation to be computed, $\mathbf{p}(\mathbf{x})$ is a basis vector and $\mathbf{a}(\mathbf{x})$ is a vector of unknown coefficients. For a polynomial basis vector

$$\boldsymbol{p}(\mathbf{x}_n)^T = \left[1,\ x_n,\ {x_n}^2 \ldots,\ {x_n}^{(d-1)},\ {x_n}^{\mathrm{d}}\right]$$

$$\mathbf{a}(x) = \begin{bmatrix} a_0(x) \\ a_1(x) \\ a_2(x) \\ \vdots \\ a_d(x) \end{bmatrix}$$

Note that the coefficients $\mathbf{a}(\mathbf{x})$ are not constant as they are for linear regression, and vary with the position.

(continued)

The cost function for MLS is

$$c(\mathbf{a}(\mathbf{x})) = \sum_{I=1}^{N} w(\mathbf{x} - \mathbf{x_I})\left(\mathbf{p}(\mathbf{x}_I)^T \mathbf{a}(\mathbf{x}) - y_n\right)^2$$

where $w(x - x_I)$ is a weight function and y_n is the discrete data being used for the regression analysis. The minimum of the cost function can be determined by taking the derivative as

$$\frac{\partial \mathrm{c}}{\partial \mathbf{a}(\mathrm{x})} = \mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{y} = 0$$

where

$$\mathbf{A}(\mathbf{x}) = \sum_{I=1}^{N} w(\mathbf{x} - \mathbf{x}_I)\mathbf{p}(\mathbf{x}_I)\mathbf{p}^T(\mathbf{x}_I)$$

$$\mathbf{B}(\mathbf{x}) = [w(\mathbf{x} - \mathbf{x_1})\mathbf{p}(\mathbf{x_1}), w(\mathbf{x} - \mathbf{x_2})\mathbf{p}(\mathbf{x_2}), \ldots, w(\mathbf{x} - \mathbf{x_N})\mathbf{p}(\mathbf{x_N})]$$

and $\mathbf{y}$ is the vector of raw datapoints.
The coefficients $\mathbf{a}(\mathbf{x})$ can be solved as

$$\mathbf{a}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{y}$$

which can be used to solve a reduced order nonlinear MLS approximation to the original data

$$y_{\mathrm{n}*} = \sum_{\mathrm{I}=1}^{\mathrm{N}} \phi_I(\mathbf{x}) y_I$$

where y_{n*} is the reduced order MLS approximation, and $\phi_I(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{A}^{-1}(\mathbf{x})\mathbf{B_I}(\mathbf{x})$ is the MLS approximation function.

References

1. Wolberg EJ (2010) The method of least squares. In: Designing quantitative experiments. Springer, Berlin, Heidelberg
2. Stanton JM (2001) Galton, Pearons, and the peas: a brief history of linear regression for statistics instructors. J Stat Educ 9:3

3. Zhang P, Li SX, Zhang ZF (2011) General relationship between strength and hardness. Mater Sci Eng A 529:62–73
4. Galkin A (2011) What is the difference between test set and validation set? Retrieved 10 Oct 2018
5. McLachlan GJ, Do K-A, Ambroise C (2004) Analyzing microarray gene expression data. Wiley
6. Wikipedia. Cross-validation. https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation