Linear Regression

An ice cream shop recorded its sales on various days of the year to try to determine if there is a relationship between temperature and sales. The data points are plotted below, in blue. The red line is an example of a simple linear regression obtained by minimizing the squared errors from the points to the line.

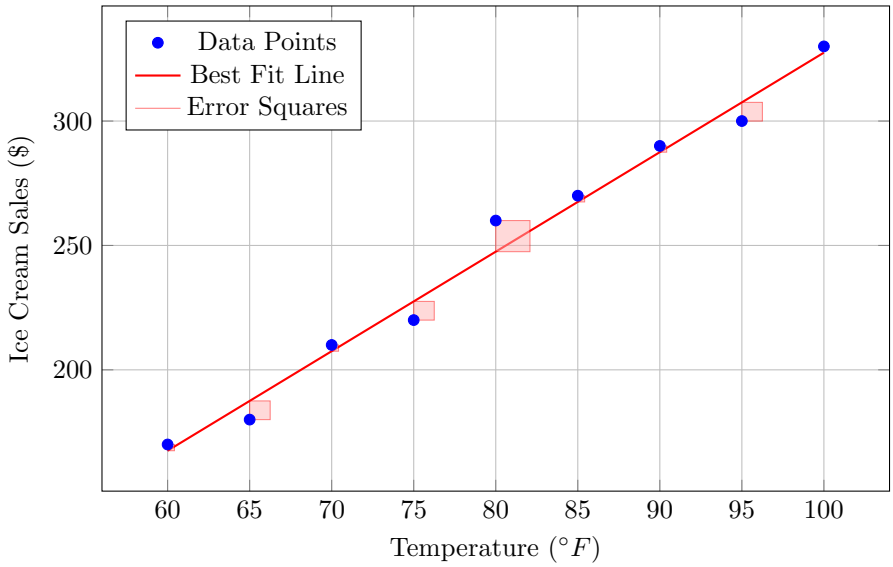

Figure 1: Temperature vs. Ice Cream Sales with Best Fit Line

1 Ordinary Least-Squares

Imagine we are given a data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i \in \{1, \ldots, n\}\}$ of size n, where each data point $\boldsymbol{x}_i$ is d-dimensional, to which we hope to fit a linear model. When $d = 1$, a **linear model** is a model in which each y_i is approximated by $mx_i + b$. Here, as in high school, m is the slope, while b is the y intercept.

In a regression model, Y is called the response—or dependent—variable, while the features are called the explanatory variables, or the predictors, and we are said to be "regressing Y on X."

The feature values can be encoded in a matrix X:

$$X = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & & & \\ x_{n1} & x_{n2} & \ldots & x_{nd} \end{bmatrix}$$

Each $\boldsymbol{x}_i$ in X is a row vector, i.e., an element of $\mathbb{R}^d$. When the number of features $d = 1$, the problem is called **simple regression**. Otherwise, it is called **multiple regression**.

The response variables $\boldsymbol{y} \in \mathbb{R}^n$ are given by the column vector:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

The goal is to find a set of weights $\boldsymbol{w} \in \mathbb{R}^d$, or parameters, or **coefficients** (e.g., m and b), so that the **estimates**,[1] or **predictions**,[2] $X\boldsymbol{w}$ approximate $\boldsymbol{y}$:

$$X\boldsymbol{w} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \dots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_1 x_{11} + w_2 x_{12} + \ldots + w_d x_{1d} \\ w_2 x_{21} + w_2 x_{12} + \ldots + w_d x_{1d} \\ \vdots \\ w_1 x_{n1} + w_2 x_{n2} + \ldots + w_d x_{nd} \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{y}$$

Observe that $X\boldsymbol{w} = \boldsymbol{y}$ is a system of n equations with d unknowns. A system of equations can be solved when it has the same number of equations and unknowns, i.e., when $n = d$. If $n < d$, i.e., if the matrix is wider than it is tall, then the system is **underdetermined**; it has infinitely many solutions. If $n > d$, i.e., if the matrix is taller than it is wide, which is usually the case in regression, the system is **overdetermined**, which means it has no solution. Consequently, rather than aim to solve this system of equations exactly, our goal is to solve for weights that minimize the difference between the predicted values $X\boldsymbol{w}$ and the observed values y. As usual, we square the errors, so as to make the ensuing optimization problem smooth.

Define the **residual** of a candidate solution $\boldsymbol{w}$ as the difference between $\boldsymbol{y}$ and $X\boldsymbol{w}$, i.e.,

$$\text{residual}(\boldsymbol{w}) = \boldsymbol{y} - X\boldsymbol{w}$$

The **least squares objective** is to minimize the square of this residual value: $\text{loss}(\boldsymbol{w}) \doteq \text{residual}^2(\boldsymbol{w})$.

As $\text{residual}(\boldsymbol{w})$ is an n-dimensional column vector, we square it as follows:

$$\text{loss}(\boldsymbol{w}) = \text{residual}^2(\boldsymbol{w}) = (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{y} - X\boldsymbol{w})_i^2$$

In other words, we are interested in minimizing the sum of the squared residuals.

As usual, to minimize an objective, we take its derivative and set it equal to zero.

First, let's simplify the loss:

$$\begin{aligned} &\text{loss}(\boldsymbol{w}) && (1) \\ &= (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) && (2) \\ &= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T X\boldsymbol{w} - (X\boldsymbol{w})^T\boldsymbol{y} + (X\boldsymbol{w})^T X\boldsymbol{w} && (3) \\ &= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T X\boldsymbol{w} - \boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{w}^T X^T X\boldsymbol{w} && (4) \\ &= \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{w}^T X^T\boldsymbol{y} + \boldsymbol{w}^T X^T X\boldsymbol{w} && (5) \end{aligned}$$

In Equation 3, we rely on the following fact: $(AB)^T = B^T A^T$. Equation 5 follows from the fact that $(\boldsymbol{y}^T X\boldsymbol{w})^T = \boldsymbol{w}^T X^T\boldsymbol{y}$, and that these values are both scalars, so they are equal to their transpose.

[1] statistical nomenclature

[2] machine learning nomenclature

Now let's take the derivative:

$$\nabla_{\boldsymbol{w}} \text{loss}(\boldsymbol{w}) \tag{6}$$
$$= \nabla_{\boldsymbol{w}} \boldsymbol{y}^T \boldsymbol{y} - 2\nabla_{\boldsymbol{w}} \boldsymbol{w}^T X^T \boldsymbol{y} + \nabla_{\boldsymbol{w}} \boldsymbol{w}^T X^T X \boldsymbol{w} \tag{7}$$
$$= -2X^T y + 2X^T X w \tag{8}$$

Equation 8 relies on the following fact: $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T A \boldsymbol{x} = 2A\boldsymbol{x}$.

Setting this derivative equal to zero yields a **closed-form** (i.e., analytical) solution to linear regression:

$$-2X^T y + 2X^T X w = 0 \tag{9}$$
$$X^T X w = X^T y \tag{10}$$
$$w = \underbrace{(X^T X)^{-1} X^T}_{\text{pseudo-inverse}} y \tag{11}$$

The matrix $X^T X$ might not be invertible. This matrix is only invertible when the columns of X are linearly independent, so that they span $\mathbb{R}^d$. The pseudo-inverse $(X^T X)^{-1} X^T$, however, always exists, and can be calculated via a **singular value decomposition**.

We have established that $\boldsymbol{w}$ satisfies the **first-order optimality condition**, namely $\nabla_{\boldsymbol{w}} \text{loss}(\boldsymbol{w}) = 0$. To conclude that $\boldsymbol{w}$ is a minimum, we also need to establish that it satisfies the second-order optimality condition. For $\boldsymbol{w}$ to be a minimum, the **second-order optimality condition** requires that $\nabla^2_{\boldsymbol{w}} f(\boldsymbol{w}) \geq 0$. Indeed, $\nabla^2_{\boldsymbol{w}} \text{loss}(\boldsymbol{w}) = 2X^T X \geq 0$.

Alternative Derivation We describe an alternative shorter, geometric, and arguably more intuitive way, to derive the least-squares estimators. The only way the residual $\boldsymbol{y} - X\boldsymbol{w}$ can be zero is if $\boldsymbol{y}$ is a linear combination of the columns of X: i.e., if $\boldsymbol{y}$ lies in the column span of X. As most of the time it does not, our stated goal of minimizing the value of the residual (squared) can instead be understood as minimizing the distance from the residual to the column span of X. To minimize this distance, the residual should be **projected** onto the column span of X; that is, it should be **orthogonal** to each column of X: for all $j \in \{1, \ldots, n\}$, i.e., $(\boldsymbol{y} - X\boldsymbol{w}) \cdot X_j = 0$. Equivalently, but expressed more compactly in matrix notation, $(\boldsymbol{y} - X\boldsymbol{w})^T X = 0$. This requirement implies:

$$X^T(\boldsymbol{y} - X\boldsymbol{w}) = 0 \tag{12}$$
$$X^T X \boldsymbol{w} = X^T \boldsymbol{y} \tag{13}$$
$$\boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{y} \tag{14}$$

Linear Regression, Continued

2 Polynomial Regression

It is straightforward to incorporate a y intercept into our matrix notation. The trick to doing so is to append an extra dimension to the parameter vector $\boldsymbol{w} \in \mathbb{R}^d$ and to likewise append an extra feature/column to X,

whose value is always 1, so that w_{d+1} represents the intercept:

$$X\boldsymbol{w} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \dots & \vdots & \\ x_{n1} & x_{n2} & \dots & x_{nd} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ w_{d+1} \end{bmatrix} = \begin{bmatrix} w_1x_{11} + w_2x_{12} + \ldots + w_dx_{1d} + w_{d+1} \\ w_2x_{21} + w_2x_{12} + \ldots + w_dx_{1d} + w_{d+1} \\ \vdots \\ w_1x_{n1} + w_2x_{n2} + \ldots + w_dx_{nd} + w_{d+1} \end{bmatrix}$$

In a simple regression of y on $X = \begin{bmatrix} \boldsymbol{x} \end{bmatrix}$, i.e., only one feature, the aforementioned trick is akin to appending $\boldsymbol{x}^0 = \mathbf{1}$ to X, resulting in $\begin{bmatrix} \boldsymbol{x} & 1 \end{bmatrix}$, or $\begin{bmatrix} 1 & \boldsymbol{x} \end{bmatrix}$. More generally, it is equally possible to append $\boldsymbol{x}$ raised to any other power p to X as well: e.g., $\begin{bmatrix} 1 & \boldsymbol{x} & \boldsymbol{x}^2 & \boldsymbol{x}^3 & \dots & \boldsymbol{x}^p \end{bmatrix}$. Therefore, **polynomial regression** reduces to linear regression! In theory at least; in practice, the set of possible combinations of powers of features is massive. Assuming just two features, say $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and $p = 3$ yields the following (long) list of possibilities:[3]

$$1 \;\; \boldsymbol{x}_1 \;\; \boldsymbol{x}_1^2 \;\; \boldsymbol{x}_1^3 \;\; \boldsymbol{x}_2 \;\; \boldsymbol{x}_2^2 \;\; \boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1\boldsymbol{x}_2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2 \;\; \boldsymbol{x}_1\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2 \;\; \boldsymbol{x}_1\boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2^3$$

Feature selection is an important and difficult problem in machine learning. The deep learning revolution can in large part be attributed to its success at automatically identifying pertinent features.

3 Regularized Regression

Linear regression is a machine learning model with a strong bias, namely the decision to fit a *line* (as opposed to a curve) to data. Even with this inherent bias, linear regression models can have high variance. Another way to limit variance is to limit the range of the model parameters $\boldsymbol{w} \in \mathbb{R}^d$.

The p-**norm** of a vector $\boldsymbol{w} \in \mathbb{R}^d$, denoted $\|\boldsymbol{w}\|_p$, for some $p \geq 1$, is a way to gauge $\boldsymbol{w}$'s size.[4]

The most popular norm is the 2-norm, also called the **Euclidean norm**:

$$\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$$

Other common choices include the 1-norm:

$$\|\boldsymbol{w}\|_1 = \sum_{i=1}^{d} |w_i|$$

and the ∞-norm:

$$\|\boldsymbol{w}\|_\infty = \max_{i \in \{1,\ldots,d\}} |w_i|$$

Nomenclature An optimization problem with decision variables $\boldsymbol{z} \in \mathbb{R}^d$ is called **constrained** when its solution must lie lie in some smaller subset, say C, of $\mathbb{R}^d$. This smaller subset is called the **feasible set**. (In an **unconstrained** optimization problem, the solution can be found anywhere in $\mathbb{R}^d$.) Figure 2 depicts the feasible sets when the p-norm of a vector in $\mathbb{R}^d$ is constrained to be less than or equal to 1, for $p \in \{1, 2, 1000, \infty\}$.

[3] I probably missed some!

[4] A norm is a function from $\mathbb{R}^d \to \mathbb{R}$ that satisfies the following three properties:

1. $\|\boldsymbol{x}\| > 0$, for all $x \neq 0$
2. $\|\alpha\boldsymbol{x}\| = \alpha\|\boldsymbol{x}\|$, for all $\alpha \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^d$
3. $\|x + y\| \leq \|x\| + \|y\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$

The final property is called the **triangle inequality**, because the sum of the lengths of two sides of a triangle is at least that of the third.

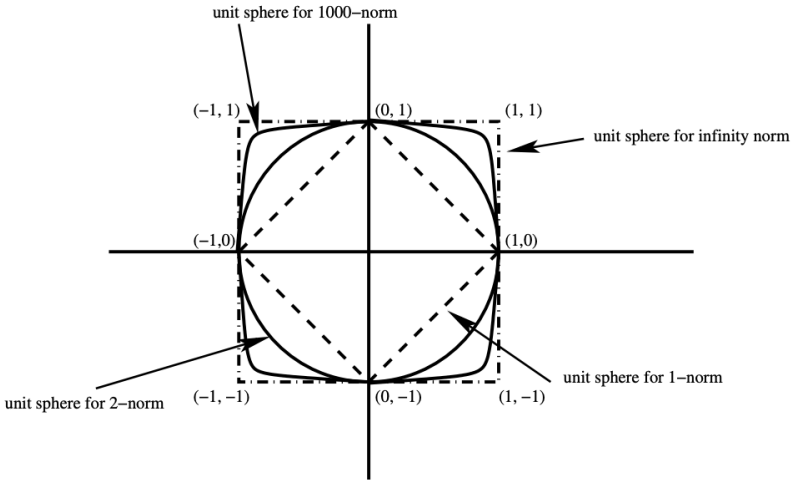

Figure 2: The feasible sets when the p-norm of a vector in $\mathbb{R}^d$ is constrained to be less than or equal to 1, for $p \in \{1, 2, 1000, \infty\}$. Image source.

Recall the ordinary least squares (OLS) objective:

$$\text{loss}(\boldsymbol{w}) = (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) = \sum_{i=1}^{n}(\boldsymbol{y} - X\boldsymbol{w})_i^2$$

In other words,

$$\text{loss}(\boldsymbol{w}) = \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2$$

Observe that OLS is an unconstrained optimization problem, as $\boldsymbol{w}$ can take on any value in $\mathbb{R}^d$. **Regularized regression** is a constrained optimization problem, with the same objective, but a limited domain for $\boldsymbol{w}$.

Ridge regression uses the 2-norm as a regularizer: for some $\beta > 0$,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 \tag{15}$$

$$\text{subject to} \quad \|\boldsymbol{w}\|_2 \leq \beta \tag{16}$$

LASSO uses the 1-norm: for some $\beta > 0$,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 \tag{17}$$

$$\text{subject to} \quad \|\boldsymbol{w}\|_1 \leq \beta \tag{18}$$

The choice of β is a choice of how much bias to include in the model.

Like OLS, ridge regression has a closed-form solution. This solution can be found by first reformulating the ridge regression constrained optimization problem in terms of its Lagrangian,[5] and then following the same steps as last time to solve OLS. The closed-form solution of ridge regression generalizes that of OLS:

$$\boldsymbol{w} = (X^TX + \lambda I)^{-1}X^T\boldsymbol{y}$$

[5]TODO !!!

On the other hand, LASSO does not have a closed-form solution. The difficulty arises from the fact that the absolute value function is not differentiable at zero. As a result, LASSO is generally solved using a generalization of gradient descent called *subgradient* descent, as the subgradient exists everywhere. The subgradient, however, is not unique, so this algorithm is usually slower to converge than gradient descent.

While both ridge regression and LASSO impose bias, and thereby limit variance, their behavior is notably different. Take, for example, the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. While $\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, only $(1, 0)$ has 1-norm 1; $\|(1/\sqrt{2}, 1/\sqrt{2})\|_1 = \sqrt{2}$. As a result, LASSO has a tendency to select features, by completing zeroing out less important features—not just assigning them small coefficients.

Figure 3 is a depiction of ridge regression and LASSO. The OLS minimum is the dark black dot. As this point lies outside the feasible set, the optimal point for each problem lies on the boundary of the feasible set. But only for LASSO does it fall on a corner; the coefficient in the y direction is completely zeroed out.

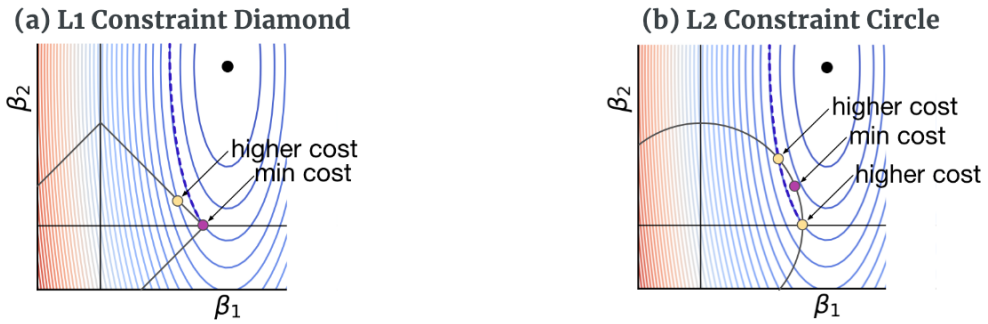

Figure 3.2. L1 and L2 constraint regions get different coefficient locations, on the diamond and circle, for the same loss function. Keep in mind that there are an infinite number of contour lines and there is a contour line exactly meeting the L2 purple dot.

Figure 3: Image source.