Descending into ML: Linear Regression

**Estimated Time:** 6 minutes

It has [long been known](https://wikipedia.org/wiki/Dolbear's_law) that crickets chirp more frequently on hotter days than on cooler days. For decades, professional and amateur entomologists have cataloged data on chirps-per-minute and temperature. As a birthday gift, Aunt Ruth gives you her beloved cricket database and invites you to learn a model to predict this relationship for yourself.

A nice first step is to examine your data by plotting it:

Raw data of chirps/minute (x-axis) vs. temperature (y-axis). 0 25 50 75 100 125 150 175 Cricket Chirps Per Minute 5 10 15 20 25 30 Temperature in Celsius

**Figure 1. Chirps per Minute vs. Temperature Centigrade.**

Sure enough, the plot shows the temperature rising with the number of chirps. Is this relationship between chirps and temperature linear? Yes, you could draw a single straight line like the following to approximate this relationship:

Best line establishing relationship of chirps/minute (x-axis) vs. temperature (y-axis). 0 25 50 75 100 125 150 175 Cricket Chirps Per Minute 5 10 15 20 25 30 35 Temperature in Celsius

**Figure 2. A linear relationship.**

True, the line doesn't pass perfectly through every dot, but the line does clearly show the relationship between chirps and temperature for the data we have. Applying a little algebra, you could write down this relationship as follows:

y=mx+b

where:

* y is the temperature in Celsius—the value we're trying to predict.
* m is the slope of the line.
* x is the number of chirps per minute—the value of our input feature.
* b is the y-intercept.

By convention in machine learning, you'll write the equation for a model only slightly differently:

y′=b+w1x1

where:

* y′ is the predicted [label](https://developers.google.com/machine-learning/crash-course/framing/ml-terminology#labels) (a desired output).
* b is the bias (the y-intercept). In some machine-learning documentation, it is instead referred to as w0 .
* w1 is the weight of feature 1. Weight is the same concept as "slope" written with m above.
* x1 is a [feature](https://developers.google.com/machine-learning/crash-course/framing/ml-terminology#features) (a known input).

To **infer** (predict) the temperature y′ for a new chirps-per-minute value x1 , just plug the x1 value into this model.

The subscripts (for example, w1 and x1 ) foreshadow more sophisticated models that rely on *multiple* features. For example, a model that relies on three features would use the following equation:

y′=b+w1x1+w2x2+w3x3

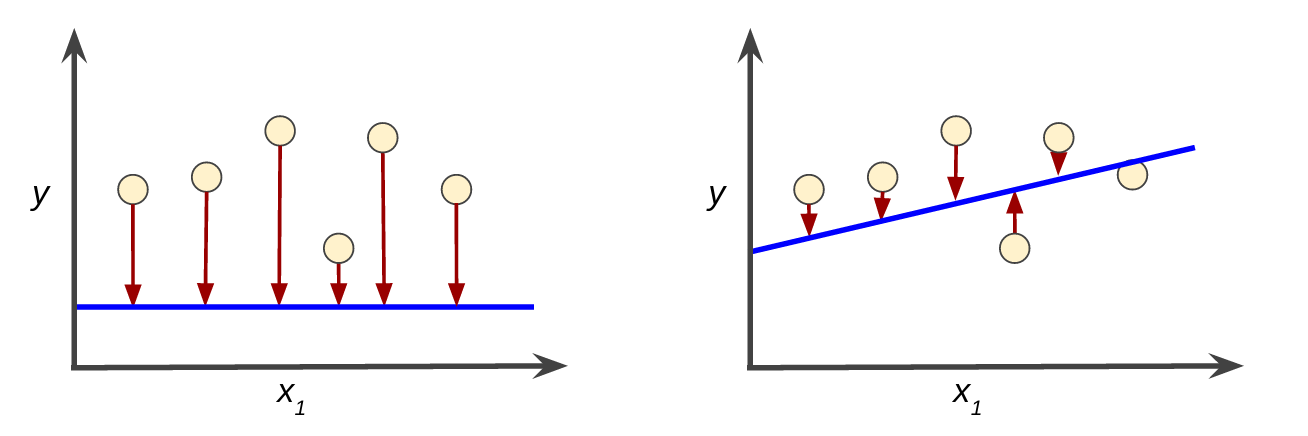
# **Descending into ML: Training and Loss**

**Estimated Time:** 6 minutes

**Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called **empirical risk minimization**.

Loss is the penalty for a bad prediction. That is, **loss** is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples. For example, Figure 3 shows a high loss model on the left and a low loss model on the right. Note the following about the figure:

* The red arrow represents loss.
* The blue line represents predictions.



**Figure 3. High loss in the left model; low loss in the right model.**

Notice that the red arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the blue line in the right plot is a much better predictive model than the blue line in the left plot.

You might be wondering whether you could create a mathematical function—a loss function—that would aggregate the individual losses in a meaningful fashion.

### **Squared loss: a popular loss function**

The linear regression models we'll examine here use a loss function called **squared loss** (also known as **L2 loss**). The squared loss for a single example is as follows:

= the square of the difference between the label and the prediction

= (observation - prediction(**x**))2

= (y - y')2

**Mean square error** (**MSE**) is the average squared loss per example. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

MSE=1N∑(x,y)∈D(y−prediction(x))2

where:

* (x,y) is an example in which
  + x is the set of features (for example, temperature, age, and mating success) that the model uses to make predictions.
  + y is the example's label (for example, chirps/minute).
* prediction(x) is a function of the weights and bias in combination with the set of features x.
* D is a data set containing many labeled examples, which are (x,y) pairs.
* N is the number of examples in D.

Although MSE is commonly-used in machine learning, it is neither the only practical loss function nor the best loss function for all circumstances.