(https://google.com/racialequity?authuser=0)

Reducing Loss: An Iterative Approach

ated Time: 10 minutes

The previous module (https://developers.google.com/machine-learning/crash-course/descending-into-ml?authuser=0) introduced the concept of loss. Here, in this module, you'll learn how a machine learning model iteratively reduces loss.

Iterative learning might remind you of the "Hot and Cold" (http://www.howcast.com/videos/258352-how-to-play-hot-and-cold/) kid's game for finding a hidden object like a thimble. In this game, the "hidden object" is the best possible model. You'll start with a wild guess ("The value of w_1 is 0.") and wait for the system to tell you what the loss is. Then, you'll try another guess ("The value of w_1 is 0.5.") and see what the loss is. Aah, you're getting warmer. Actually, if you play this game right, you'll usually be getting warmer. The real trick to the game is trying to find the best possible model as efficiently as possible.

The following figure suggests the iterative trial-and-error process that machine learning algorithms use to train a model:

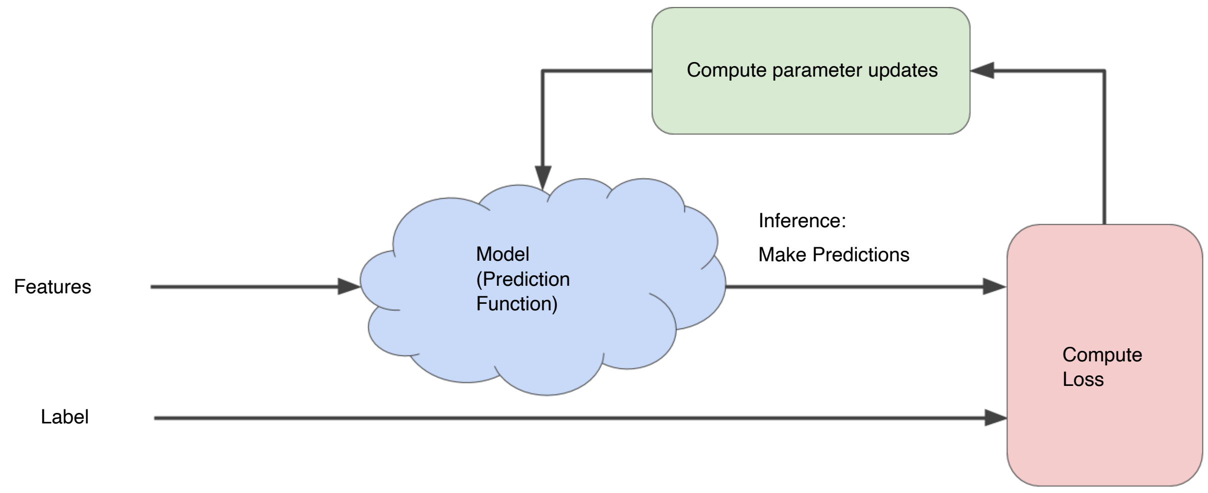

Figure 1. An iterative approach to training a model.

We'll use this same iterative approach throughout the Machine Learning Crash Course, detailing various complications, particularly within that stormy cloud labeled "Model (Prediction Function)." Iterative strategies are prevalent in machine learning, primarily because they scale so well to large data sets.

The "model" takes one or more features as input and returns one prediction (y') as output. To simplify, consider a model that takes one feature and returns one prediction:

$$y' = b + w_1 x_1$$

What initial values should we set for b and w_1? For linear regression problems, it turns out that the starting values aren't important. We could pick random values, but we'll just take the following trivial values instead:

- b = 0
- w_1 = 0

Suppose that the first feature value is 10. Plugging that feature value into the prediction function yields:

$$y' = 0 + 0 \cdot 10 = 0$$

The "Compute Loss" part of the diagram is the loss function (https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss?authuser=0)
that the model will use. Suppose we use the squared loss function. The loss function takes in two input values:

- y': The model's prediction for features *x*
- y: The correct label corresponding to features *x*.

At last, we've reached the "Compute parameter updates" part of the diagram. It is here that the machine learning system examines the value of the loss function and generates new values for b and w_1. For now, just assume that this mysterious box devises new values and then the machine learning system re-evaluates all those features against all those labels, yielding a new value for the loss function, which yields new parameter values. And the learning continues iterating until the algorithm discovers the model parameters with the lowest possible loss. Usually, you iterate until overall loss stops changing or at least changes extremely slowly. When that happens, we say that the model has **converged**.

›int: