Minimize model errors with cost functions

4 minutes

The learning process repeatedly alters a model until it can make high-quality estimates. To determine how well a model is performing, it uses mathematics in the form of a cost function, also known as an objective function. To understand what objective function is, let's break it down a little.

Error, cost, and loss

In supervised learning, error, cost, and loss all refer to the number of mistakes that a model makes in predicting one or more labels.

These three terms are used somewhat loosely and often interchangeably in machine learning, which can cause some confusion when reading from multiple sources. For the sake of simplicity, we'll use them interchangeably here. Cost is calculated using mathematics, it isn't a qualitative judgment. For example, if a model predicts that a daily temperature will be 40°F, but the actual value is 35°F, we might say it has an error of 5°F.

Minimizing cost is our goal

Given that cost indicates how badly a model works, our goal is to have zero cost. In other words, we want to train the model to make no mistakes at all. This idea is often impossible though, so instead we set a slightly more nebulous goal of training the model to have the minimum cost possible.

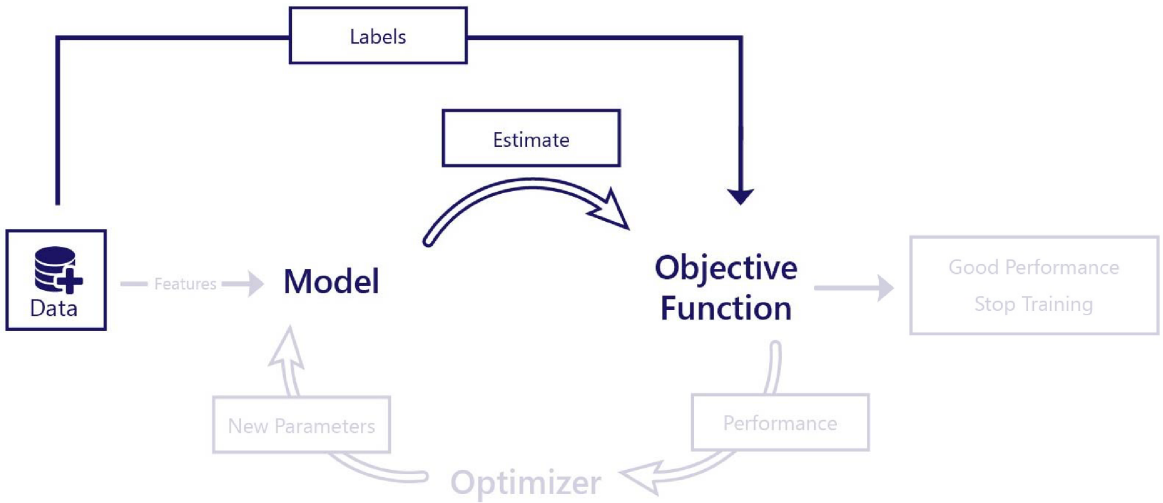
Because of this goal, how we calculate cost dictates what the model will try to learn. In the above example, we defined cost as the error in estimating temperature.

What is a cost function?

In supervised learning, a cost function is a small piece of code that calculates cost from a model's prediction and the expected label—the correct answer. For example, in our previous exercise we calculated cost by calculating the prediction errors, squaring them, and summing them.

Once the cost function has calculated cost, we know whether the model is performing well or not. If it's performing well, we might choose to stop training. If not, we can pass cost

information to the optimizer, which uses this information to select new parameters for the model.



During training, different cost functions can change how long training takes, or how well it works. For example, if the cost function always states errors are small, the optimizer will only make small changes to the model. As another example, if the cost function returns very large values when certain mistakes are made, the optimizer will make changes to the model so that it doesn't make these kinds of mistakes.

There isn't a one-size-fits-all cost function—which is best depends on what we're trying to achieve. Meaning we often need to experiment with cost functions to get a result we're happy with. In the next exercise, we'll do this experiment.

Next unit: Exercise - Optimize a model using cost functions

Continue >