Chpt5 Mechanics of learning

Augest 11, 2021

Cover:

1. Understanding how algorithms can learn from data
2. Reframing learning as parameter estimation, using differentiation and gradient descent
3. How PyTorch supports learning with autograd

A learning algorithm is presented with input data that is paired with desired outputs. Once learning has occured, that algorithm will be capable of producing correct outputs when it is fed new data that is similar enough to the input data it was trained on. With deep learning, this process works even when the input data and the desired output are *far* from each other.
The process always involves a function with a number of unknown parameters whose are estimated from data: in short, a *model*.
In this book, we're interested in models that are not engineered for solving a specific narrow task, but can be automatically adapted to specialize themselves for any one of many similar tasks using input and output pairs-in other words, general models trained on data relevant to the specific task at hand.

This chapter is about how to automate generic funciton-fitting.

5.2 Learning is just parameter estimation

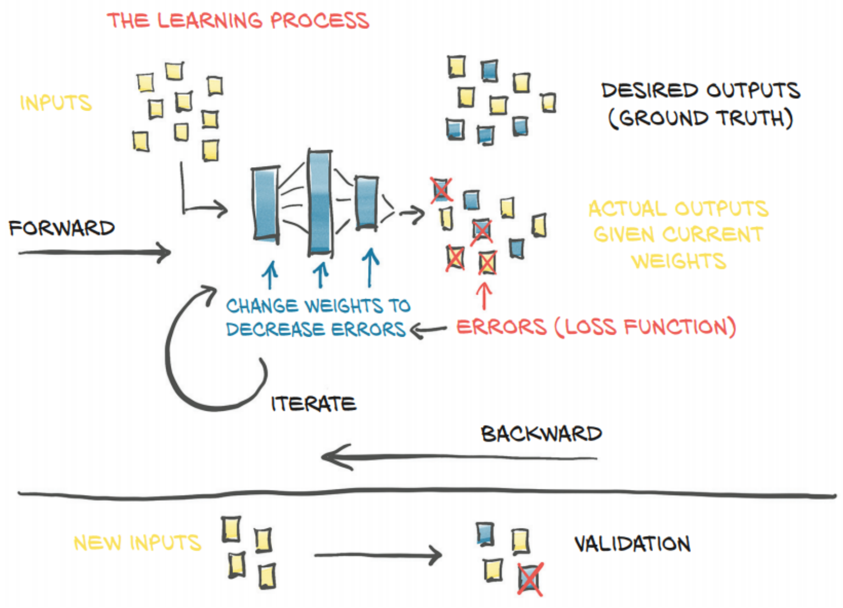
THE LEARNING PROCESS
INPUTS
DESIRED OUTPUTS (GROUND TRUTH)
FORWARD
ACTUAL OUTPUTS GIVEN CURRENT WEIGHTS
CHANGE WEIGHTS TO DECREASE ERRORS
ERRORS (LOSS FUNCTION)
ITERATE
BACKWARD
NEW INPUTS
VALIDATION

Figure 5.2 Our mental model of the learning process

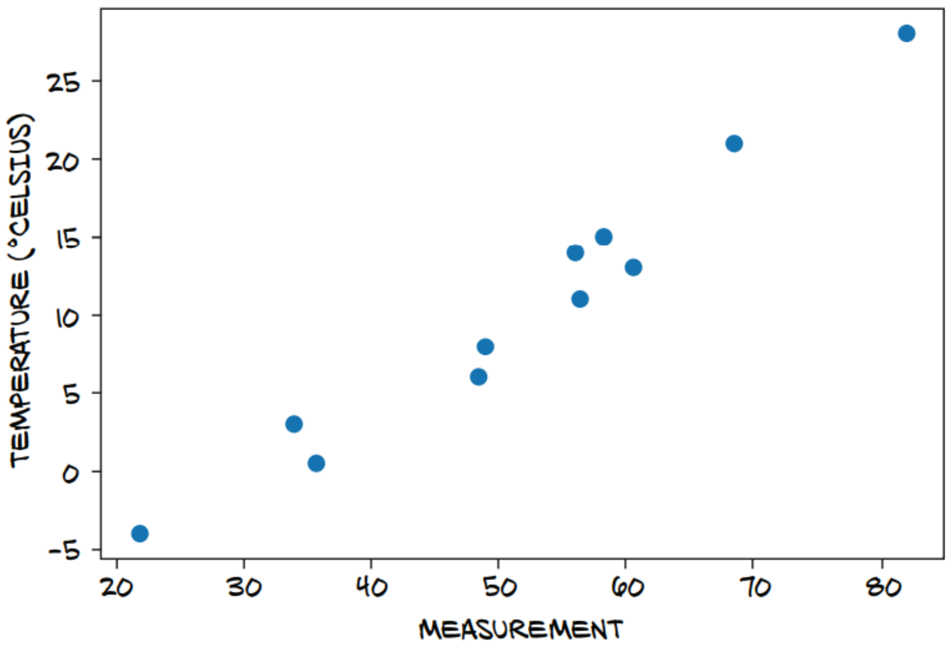

Figure 5.3 Our unknown data just might follow a linear model.

5.2.4 Chooseing a linear model as a first try

We assume the simplest possible model for converting between the two sets of measurements. The two may be linearly related-that is, multiplying t_u by a factor and adding a constant, we may get the temperature in Celsius

$t\_c = w * t\_u + b$

We choose to name w and b after *weight* and *bias*. Now we need to estimate w and b, the parameter in our model, based on the data we have. We'll go through this simple example using PyTorch and realize that training a neural network will essentially involve changing the model for a slightly more elaborate one, with a few more parameters. Our optimization process should therefore aim at finding w and b so that the loss function is at a minimum.

5.3 Less loss is what we want

The loss function in our case, that would be different between the predicted temperature t_p output by our model and the actual measurements: $t\_p - t\_c$.

Because the steepness of the growth also monotonically increase away from the minimum, both of them are said to be *convex*. Since our model is linear, the loss as a function of w and b is also convex.

5.4 Down along the gradient

We'll optimize the loss function with respect to the parameters using the *gradient descent* algorithm. The gradient descent idea is to compute the rate of change of the loss with respect to each parameter, and modify each parameter in the direction of decreasing loss.

5.5 PyTorh's autograd: Backpropagating al