Chpt6 Neural network to fit data

Augest 18, 2021

Cover:

1. Nonlinear activation functions as the key difference compared with linear models.
2. Working with Pytorch's nn module.
3. Solving a linear-fit problem with a nn.

6.1 Artificial neurons

NN: mathemactical entities capable of representing complicated functions through a composition of simpler functions. At its core, it's nothing but a linear transformation of the input followed by the application of a fixed nonlinear function (*activation function*). $output = f(w * x + b)$. In general, x and o can be similiar scalars, or vector-valued.

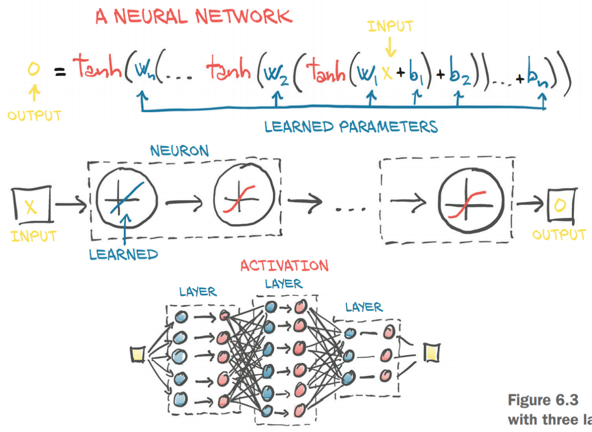

Figure 6.3 A neural network with three layers

Remeber that w here is a **matrix**, and x is **vector**. Using a vector allows w to hold an entire layer of neurons, not a just single weight.

6.1.2 Understanding the error function

NN does not have that same property of a convex error surface, even when using the same error-squared loss function. There's no single right answer for each parameter we're attempting to approximate. The result in nn training looking very much like parameter estimation from a mechanical perspective. A big part of reason nn have non-convex error surfaces is due to the **activation function**. The ablility of neurons to apporximate a very wide range of useful functions depends on the combination of the linear and nonlinear behavior inherent to each neuron.

6.1.3 All we need is activation

The activation function plays two important roles:

1. In the inner parts of the model, it allows the output function to have different slopes at different values. By trickily composing these differently sloped parts for many outputs, neural networks can approximately arbitrary functions.

2. At the last layer of network, it has the role of concentrating the outputs of the preceding linear operation into a given range.

Activation functions:

1. Are nonlinear. The nonlinearity allows the overall network to approximate more complex functions.

2. Are differentiable, so that gradients can be computed throught them.

Deep neural networks give us the ability to approximate highly nonlinear phenomena without having an explicit model for them.

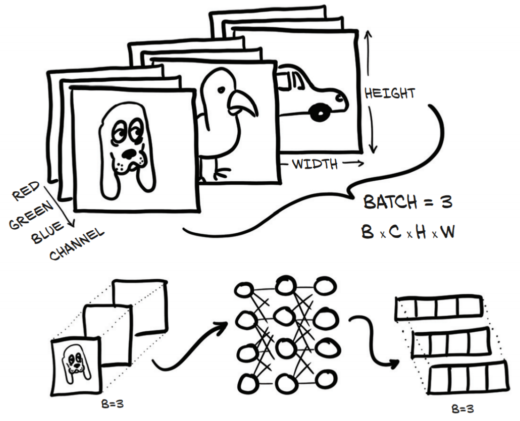

Figure 6.7 Three RGB images batched together and fed into a neural network. The output is a batch of three vectors of size 4.

Showing a similar situation with batched image data. Our input is B*C*H*W with a batch size of 3 (say, images of a dog, a bird, and then a car), three channel dimensions (red, green, and blue), and an unspecified number of pixels for height and width. As we can see, the output is a tensor of size B*Nout, where the Nout is the number of features: four in this case.

Another benefit of batching is that some advanced models use statistical information from the entire batch, and those statistics get better with larger batch sizes.

6.6 Summary

1. NN can be automatically adapted to specialize themselves on the problem at hand.

2. NN allows easy access to the analytical derivatives of the loss with respect to any parameter in the model.

3. To recognize overfitting, it's essential to maintain the training set of data points separate from the validation set.