1678 Lecture 2: Function Approximation

Machine learning (which deep learning is a subset of) is about taking data and trying to approximate some function (that we don't know, and may not even be computable) that governs the data's behavior.

One way is binning (see the lecture slides for some pics).

Binning: A simple way to approximate a function

Binning outputs a binary one-hot vector (like this: only one item will be a 1, with the rest 0s)

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

saying which bin the actual thing would be in. If the "actual" function governing the data's behavior is smooth, we can get an upper bound on the amount of bins needed to get "close enough" (which is usually written ϵ)– sometimes, within machine precision.

Basis Functions

Binning is one example of a "basis function." You can also make models fitting data to a polynomial, or a number of nested periodic (think sine or cosine) functions, called a Fourier series.

As you make these basis functions– polynomials, Fourier, or neural networks– bigger and bigger, you could approximate **any** function if they have enough fectures, for **some** model weights. It's like a Taylor series– as you add more and more terms, you get closer and closer to, say, sine.

Higher-order basis functions grow exponentially with the input vector's dimensions, so they aren't useful for images and text. As GPT and Stable Diffusion show, though, neural networks are suitable for tasks like that.

Loss

| | |

| | |__

What a loss function is is a function that describes "how bad" our model weights are. We usually don't know what function we're trying to approximate (or else we'd just use it to predict things!), so we can approximate it with how badly our model would've performed on our data.

A common one is mean squared error, which is the average of (how far off our function is from each data point, squared)– exactly what it sounds like.