Neural Networks: The Hard Way

David E. Weirich

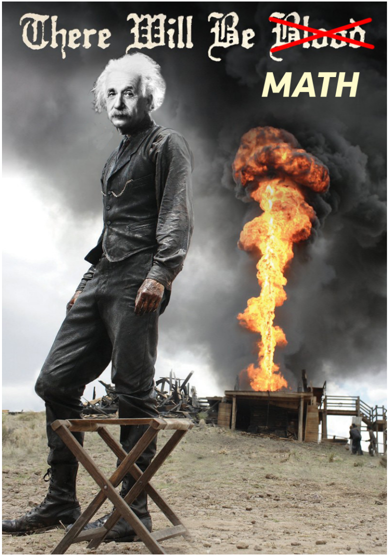
March 5, 2018

Talk Objectives

- Understand the mathematical basis for artifical neural networks.
- Use that knowledge to design and build a neural network without relying on preexisting ML frameworks.

Or more consicely...
Learn Neural Networks THE HARD WAY

Disclaimer



What are Neural Networks?

Q: What is a neural network?

A: *A neural network is this super confusing thing that is kind of like an artifical brain.*

Q: Why do we care?

A: *It can learn anything, do anything, and no one understands them.*

Why the "hard way"??? I prefer easy!

What do I mean by "the hard way"?
Doing things the hard way means teaching you how to teach yourself.

The Problem

We wish to approximate a function given a finite collection of known inputs and outputs.

$$||F - \widehat{F}|| < \varepsilon$$

Here F is the true function, and $\widehat{F}$ is our approximation.

Neurons

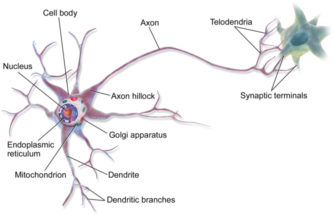


Figure: A neuron. These things are involved somehow? (Source: Wikipedia)

Pictures that look like this

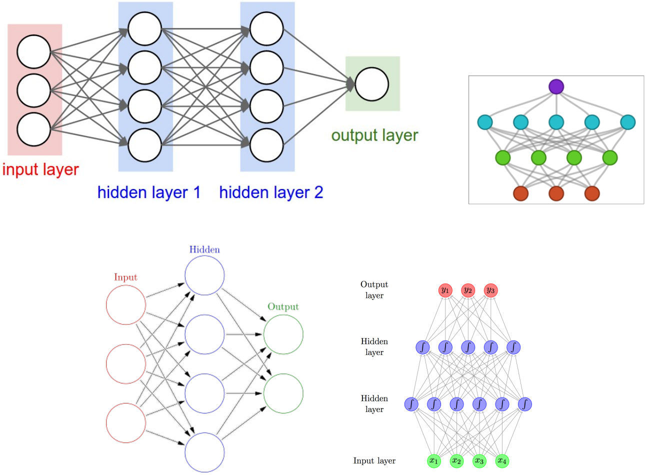


Figure: Some diagrams with circles and arrows.

Digits

$$8\ 9\ 7\ 1\ 6$$

Digits

Digits

The Σexy Way

Let's take a look at the math under the hood:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$\widehat{y} = \widehat{F}(x) = W_2\sigma(W_1\sigma(W_0x + b_0) + b_1) + b_2$$

where W_i is a weight matrix, b_i is a bias vector, and σ is an activation function.

Breaking that down

$$
\begin{aligned}
z_0 &:= W_0 x + b_0 \\
a_0 &:= \sigma(z_0) \\
z_1 &:= W_1 a_0 + b_1 \\
a_1 &:= \sigma(z_1) \\
&\vdots \\
\widehat{y} &:= W_n a_{n-1} + b_n
\end{aligned}
$$

Universal Approximation Theorem

Theorem 1 (Cybenko (1989), Hornik (1991))

Let σ be a nonconstant, bounded, and monotoncally increasing function. Given any $\varepsilon > 0$ and any continuous function F defined on a compact subdomain Ω of $\mathbb{R}^n$ there exists a constant N, real constants v_i and b_i, and real vectors w_i such that we may define

$$\widehat{F}(x) := \sum_{i=1}^{N} v_i \sigma(w_i^T \cdot x + b_i)$$

with

$$\left| F(x) - \widehat{F}(x) \right| < \varepsilon$$

for all x in Ω.

Okay, fine, I get it. But how do I set those weights?

For M observations (x_i, y_i), define a loss function

$$
\begin{aligned}
J(y_i, \widehat{y}) &:= \frac{1}{2M} \sum_{i=1}^{N} (y - \widehat{y})^2 \\
&= \frac{1}{2M} \sum_{i=1}^{N} \left(y - \widehat{F}(x_i) \right)^2 \\
&= \frac{1}{2M} \sum_{i=1}^{N} \left(y - \widehat{F}(x_i; W, b) \right)^2
\end{aligned}
$$

Now we just have to minimize the loss!

The Gradient

Recall from multivariable calculus that the *gradient* is a vector which points in the direction of steapest assent on a surface.

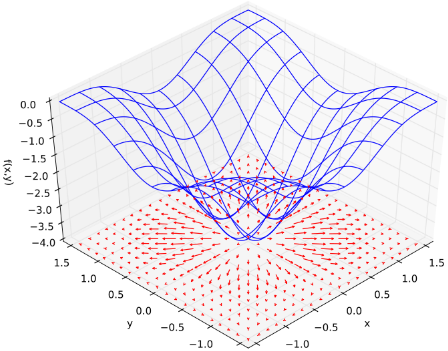


Figure: A function of two variables, with it's gradient.

The Game Plan

- We start by setting the weights and biases of the NN randomly.
- Calculate the gradient of the loss function J with respect to these parameters.
- Take a small step in the direction of the negative gradient.
- Repeat untill the error is acceptable.

WE’LL DO IT LIVE

WE’LL DO IT LIVE!

Thank you! :)