

The Mathematics of Deep Learning

UT Tyler Math Club

Alex Bearden

Department of Mathematics, The University of Texas at Tyler

cbearden@uttyler.edu

November 16, 2022

AI Balderdash


Five of the following are real mathematical terms (from [encyclopediaofmath.org](https://www.encyclopediaofmath.org/)), and five were generated using a neural network. Guess which are which!

- (real) ✓ ① Nephroid
- (fake) ✗ ② Axtric group
- ✗ ③ Constructive limit
- ✗ ④ Polthedo-prime varceset assusting and quasi-logarithmic residue
- ✗ ⑤ Non-Abel congruence
- ✓ ⑥ Cohn-Vossen transformation
- ✓ ⑦ B-Phi-structure
- ✓ ⑧ Antimatroid
- ✗ ⑨ Rebuture condition
- ✓ ⑩ Generatrix



Here's a further sampling from the AI-generated category:

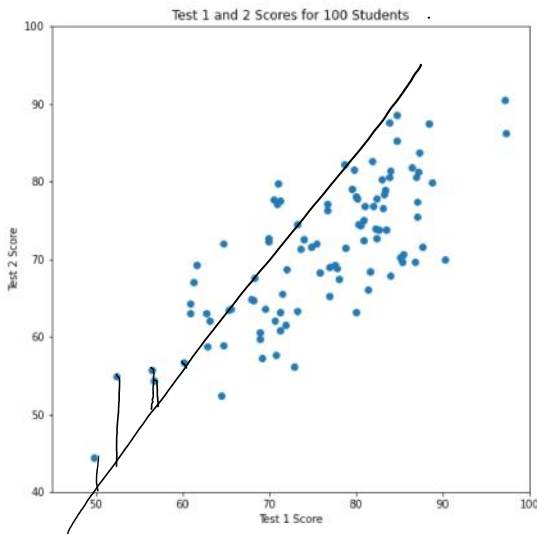
Sylthetial measure
Harbynskay square
Merten algebra
DOL-sequence
Brayamingor projector
Bikeanov theorem
Sortenkeveâdynami newarger problem
Commutants, method of
Banosi reparnections
Loxic group
Bloschteghans theorem
Finite-Hopgeostal category
Errsen errorplination transform
Abne-lays formula
Snav hoooten Ontle
Lie belivid-Floqued substatistic
Sumpay of a curve



The main goal of this talk is to explain some of the mathematics behind neural networks like the one that generated these phrases. Some of the mathematical concepts we'll encounter along the way are:

- ① Using derivatives for optimization;
- ② Probabilities;
- ③ The function $\sigma(x) = \frac{1}{1 + e^{-x}}$ and its basic properties;
- ④ Using logs to convert a product into a sum;
- ⑤ The Chain Rule;
- ⑥ Nightmarish proliferation of notation.

Suppose that the following graph shows Test 1 and Test 2 scores for 90 students in a course, and we wish to find the line that best captures the relationship between these points.



The most common measure of how well a line fits a collection of points is called the “mean squared error”—this is the average of the squares of the vertical distances from each point to the line. If the points given are

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n),$$

then mean squared error to a line $y = mx$ is

$$MSE = \frac{1}{n} ((y_1 - mx_1)^2 + (y_2 - mx_2)^2 + \dots + (y_n - mx_n)^2) .$$

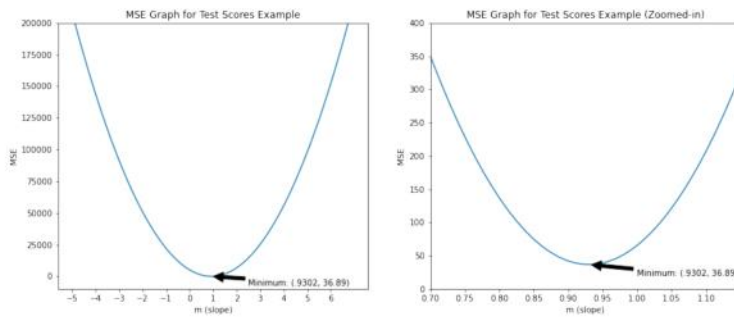
The goal then is to find the value of m that makes MSE as small as possible. This sounds like a job for...

Bloschtegans theorem!!!

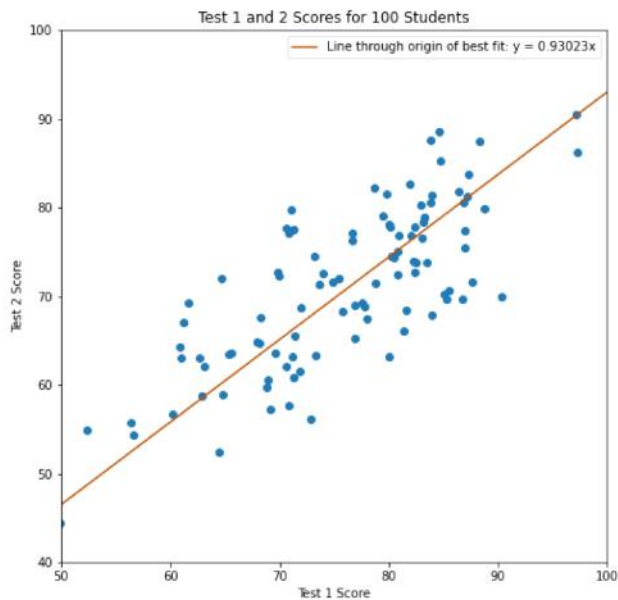
I mean, calculus. Setting the derivative of MSE (with respect to the variable m) equal to 0 and solving gives that the optimal m is:

$$m_{bestest} = \frac{x_1y_1 + x_2y_2 + \cdots + x_ny_n}{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Here's a graph of MSE against m for the student test scores example:

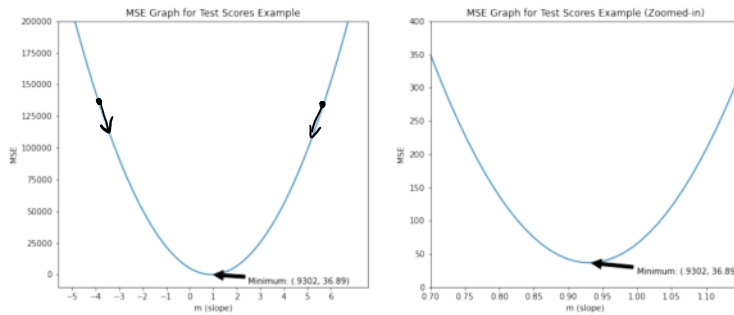


And here's the resulting line through the origin of best fit:



Having a closed form solution for $m_{bestest}$ is nice, but could become very costly to compute in more complicated situations. An alternate approach is start somewhere on the curve, compute the derivative, and let that tell you where to go.

More specifically, we could guess an initial m_0 , compute $MSE'(m_0)$, update our slope to $m_1 = m_0 - 0.1MSE'(m_0)$, and so on.

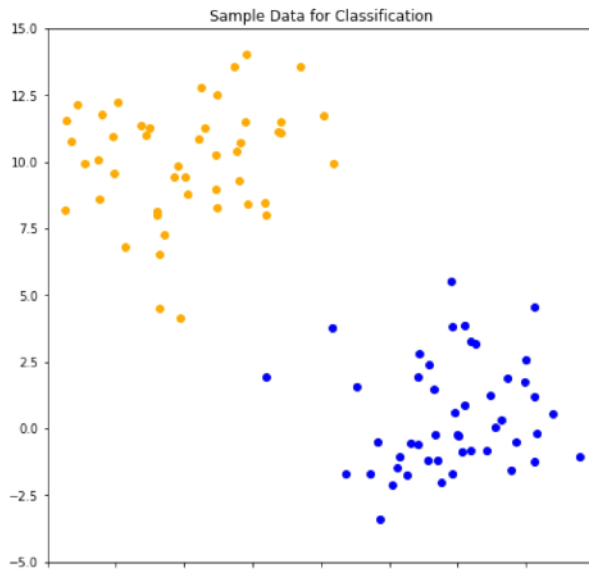


(See visualization.)

To recap what we did in terms we can generalize, we:

- 1 Started with labeled data: (x, y) . ^{label}
- 2 Chose a model: $y = mx$. ^(one parameter, m)
- 3 Chose a “cost function”: MSE .
- 4 Optimized to find the m that minimized the cost.

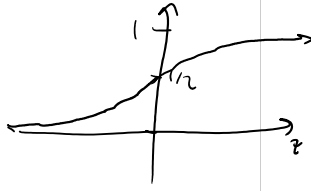
Moving on to a new problem, suppose we have a bunch of points labeled either blue or orange as in the following, and we wish to get a computer to “learn” what the difference is and be able to decide the color of any future dots we provide to it.



Going through the same process as above, we have:

- 1 Labeled data: (x, y, c) ^{label (0 or 1)}

- 2 Model:
$$c = \begin{cases} 1 & \text{if } \sigma(m_1 x + m_2 y + b) \geq \frac{1}{2} \\ 0 & \text{if } \sigma(m_1 x + m_2 y + b) < \frac{1}{2} \end{cases}$$

 $\sigma(z) = \frac{1}{1+e^{-z}}$
 (three parameters) 

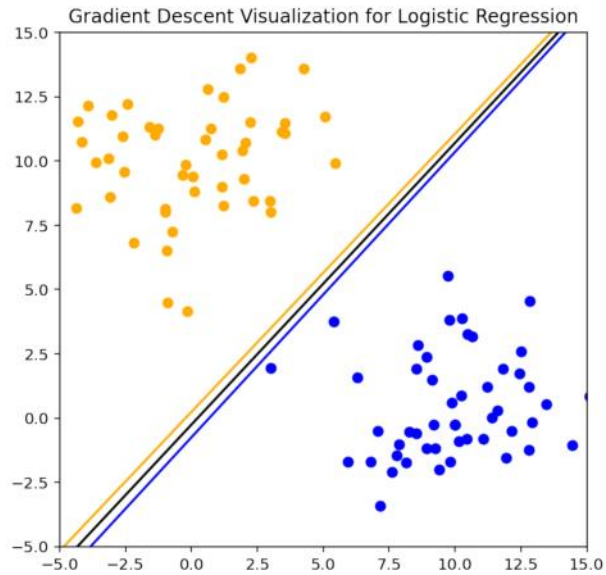
- 3 Cost function:
$$J(m_1, m_2, b) = \frac{1}{n} \sum_{i=1}^n \left(p_i \text{ if } c_i = 1 \right) \left(1 - p_i \text{ if } c_i = 0 \right) \dots \dots \left(p_n \text{ if } c_n = 1 \right) \left(1 - p_n \text{ if } c_n = 0 \right)$$

 $p_i = \sigma(m_1 x_i + m_2 y_i + b)$

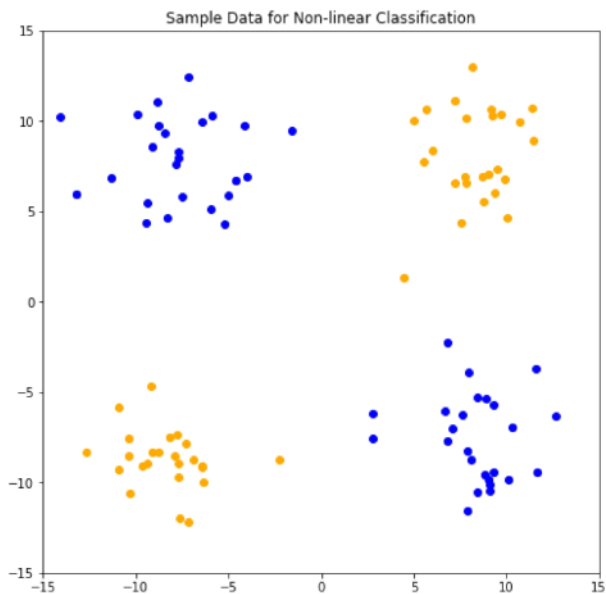
- 4 Optimize:
$$= \frac{1}{n} (c_1 \ln p_1 + (1-c_1) \ln(1-p_1)) + \dots + (c_n \ln p_n + (1-c_n) \ln(1-p_n))$$

 Here to use gradient descent.

Here's the result on the data above:

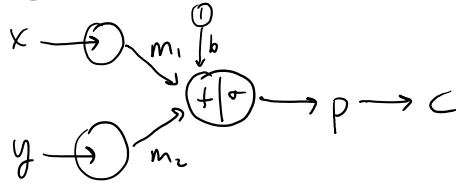


In the previous demonstration, the decision boundary was always a line. What if our data looks like this though? A logistic regression would be hopeless...

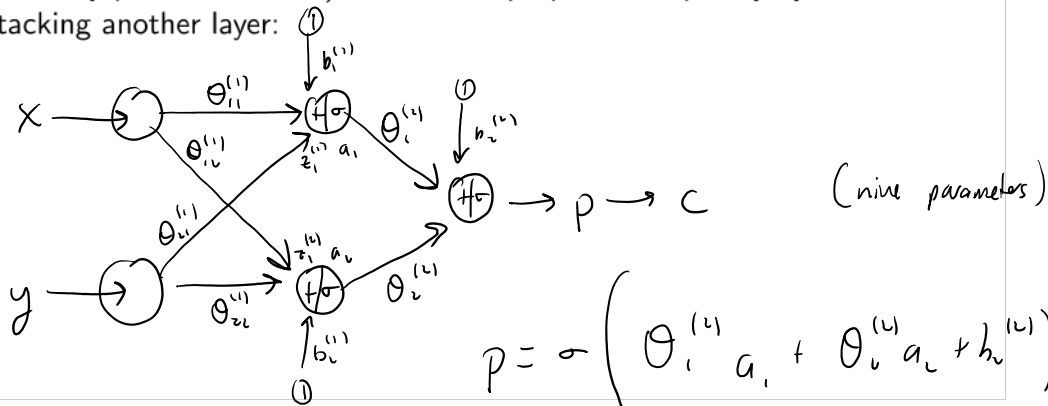


Neural networks!

The logistic regression model we used above can be graphed like this:



Taking inspiration from neurons in the brain (or perhaps just driven by pure masochism), we can ramp up the complexity by stacking another layer:



where $a_i = \sigma(\theta_{i1}^{(1)} x + \theta_{i2}^{(1)} y + b_i^{(1)})$, etc.

Neural networks are generally unwieldy, but nowadays people have tamed them pretty well. In general, they could have any number of layers and neurons per layer. There are also several important specialty architectures that are used for computer vision, natural language processing, etc.

As for training a neural network, the simplest effective algorithm is gradient descent, where the derivatives are computed via a process called *backpropagation*, which relies on the Chain Rule from calculus.

If f , g , and h are all differentiable functions, then the derivative of $f \circ g \circ h$ is given by

$$f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x) = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}$$

Wouldn't it be fun to work out the derivatives $\frac{\partial J}{\partial \theta_1^{(2)}}$ and $\frac{\partial J}{\partial \theta_{11}^{(1)}}$?!

(For one input.)

$$\begin{aligned} \frac{\partial J}{\partial \theta_1^{(2)}} &= \frac{\partial J}{\partial p} \cdot \frac{\partial p}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial \theta_1^{(2)}} \\ &= \frac{p-c}{p(1-p)} \cdot p(1-p) \cdot a_1 \\ &= (p-c) \cdot a_1 \end{aligned}$$

$$\begin{aligned} \frac{\partial J}{\partial \theta_{11}^{(1)}} &= \frac{\partial J}{\partial p} \cdot \frac{\partial p}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a_1} \cdot \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \cdot \frac{\partial z_1^{(1)}}{\partial \theta_{11}^{(1)}} \\ &= \frac{p-c}{p(1-p)} \cdot p(1-p) \cdot \theta_1^{(2)} \cdot a_1 \cdot (1-a_1) \cdot x \\ &= (p-c) \theta_1^{(2)} a_1 (1-a_1) x \end{aligned}$$

(using $J = c \ln p + (1-c) \ln(1-p)$, $p = \sigma(z^{(2)})$, $z^{(2)} = \theta_1^{(2)} a_1$, $a_1 = \sigma(z_1^{(1)})$, $z_1^{(1)} = \theta_{11}^{(1)} x + \theta_{12}^{(1)} y + b_1^{(1)}$).

Thank you!