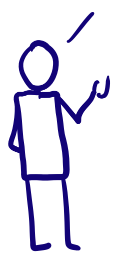
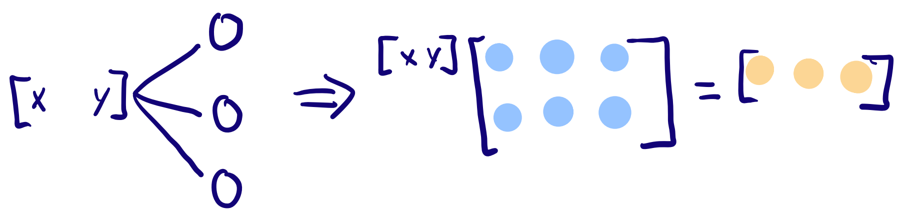
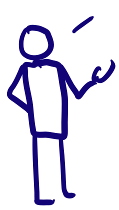
Feature engineering in neural networks



You remember that neural networks do dimensional, nonlinear transforms on the inputs right?

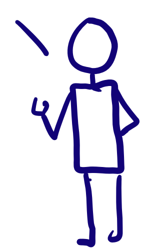


$$(1\times 2) @ (2\times 3) = [1\times 3]$$



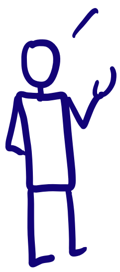
Let's explore what kind of transformations happen. What are those 6 new numbers? and how do they relate to [x y]?

The key to this analysis is just working out the algebra.

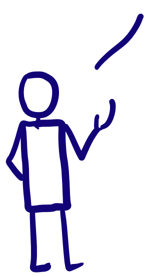


$$\begin{bmatrix} x & y \end{bmatrix} \underbrace{\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}}_{\text{weights}} + \underbrace{\begin{bmatrix} g & h & i \end{bmatrix}}_{\text{biases}} = \overbrace{\begin{bmatrix} ax + dy + g, & bx + ey + h, & cx + fy + i \end{bmatrix}}^{\text{elementwise nonlinear activation}}$$

(1×3) output

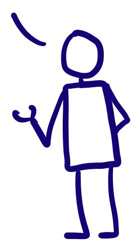


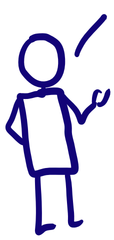
these new elements are just linear combinations of the original features (plus a constant and nonlinear activation).



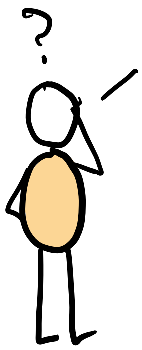
Note you don't see new features like xy, x^2, y^2, $\sqrt{x}$, etc... This formulation of the neural network does not do that.

So, if those are important features in your model you should include it!



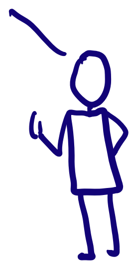


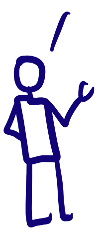
Neural networks might work anyway, but they always work better with the right features that capture the relevant nonlinearities.



Um. Professor, how can it work anyway? You keep saying its not magic...

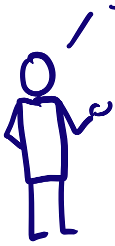
That's right, its not magic! Let's take another look at that activation function tanh.





tanh hides its real form, which is not convenient to write. You can see it is a sum and ratio of exponentials.

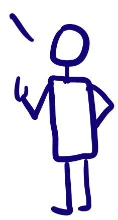
$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

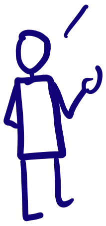


If we now look into e^{x}, we can express it as a Taylor's series:

$$e^{x} = 1 + x + \frac{x^{2}}{2!} + \frac{x^{3}}{3!} + \cdots$$

So, buried inside each e^{x} is a lot of nonlinearities! and pseudo-combinations.

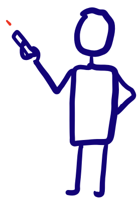




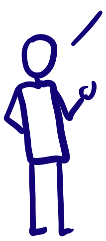
For example, consider e^{a+b}
we can write out the first 3 terms

$$\approx 1 + (a+b) + \frac{1}{2}(a+b)^2 + \ldots\ldots$$

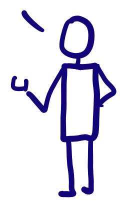
$$\approx 1 + (a+b) + \frac{1}{2}(a^2 + 2ab + b^2) + \ldots\ldots$$



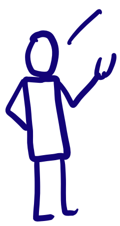
And look at that,
you see <u>some</u> squared +
cross terms!



Now, those aren't exactly in the form that makes them the same as having them in the features, and they exist in sums and ratios in the final output of tanh, but it is often just enough to usually work with enough neurons.



This is also the case for the sigmoid activation function.



Relu works for a different reason. The net result of Relu is a piecewise linear function. Any function can be approximated this way. You may need a lot of neurons to approximate non linear data this way! Anyway, its all math, All the way down!