The Maths of Machine Learning and Nerual Networks

- Alex Punnen
- 2019-2020

Vectors, Dot Products and the Perceptron

Q. Why are inputs and weights modelled as Vectors in Neural Networks ?

To understand the maths or modelling of the neural network, it is best to start from the beginning when things were simple. The earliest neural network - the Rosenblatt's perceptron was the first to introduce the concept of using vectors and the property of dot product to split hyperplanes of input feature vectors. These are the fundamentals that are still used today. There is a lot of terms above, and the rest of the article is basically to illustrate these concepts from the very basics.

Before we talk about why NN inputs and weights are modelled as vectors (and represented as matrices) let us first see what these mathematical concepts mean geometrically. This will help us in understanding the intuition of these when they are used in other contexts/ in higher dimensions.

Q. What does a vector mean?

A Vector is meaningless[1] unless you specify the context - Vector Space. Assume we are thinking about something like force vector, the context is a 2D or 3D Euclidean world

Source: 3Blue1Brown's video on Vectors

[1]Maths is abstract and meaningless unless you apply it to a context- this is a reason why you will get tripped if you try to get just a mathematical intuition about the neural network

The easiest way to understand it is in a geometric context, say 2D or 3D cartesian coordinates, and then extrapolate it. This is what we will try to do here.

Q. What is the connection between Matrices and Vectors?

Vectors are represented as matrices. Example here is a Euclidean Vector in three-dimensional Euclidean space (or R^3), represented as a column vector (usually) or row vector

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}$$

Q. What is a Dot product? and what does it signify ?

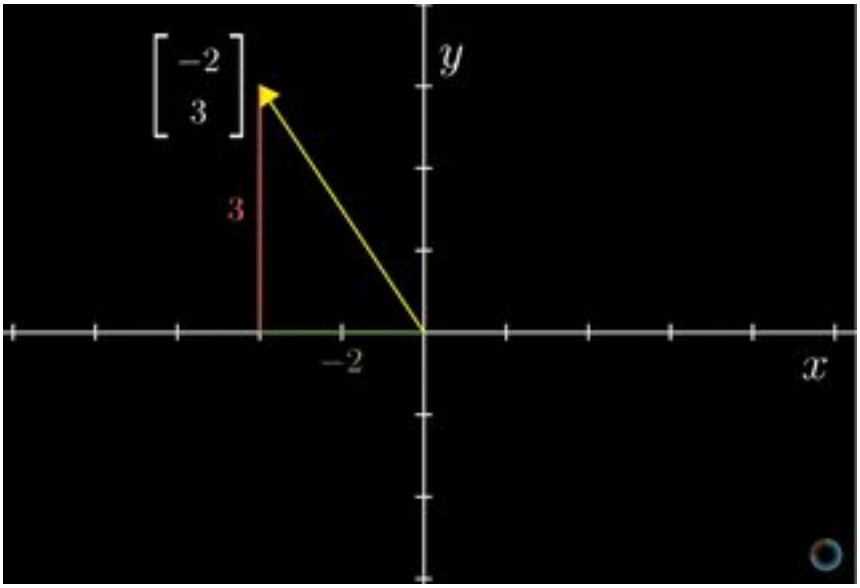


Figure 1: vector2D

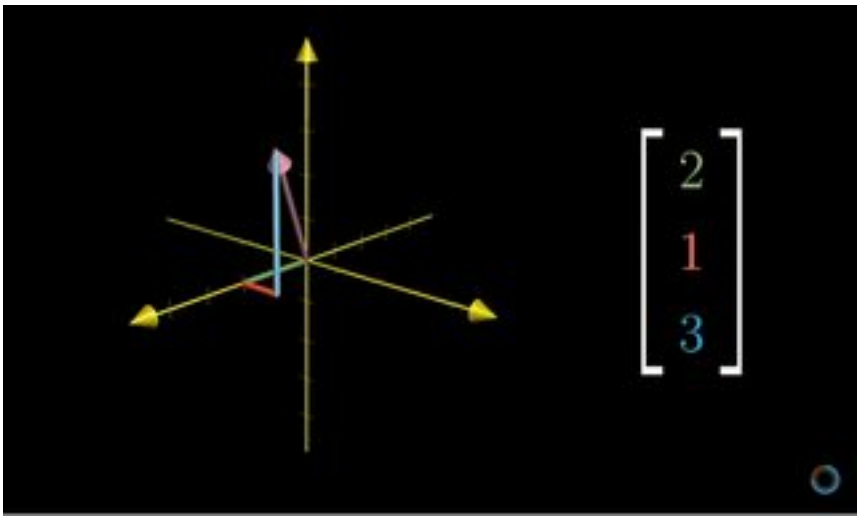


Figure 2: vector3D

First the dry definitions. **Algebraically,** the dot product is the sum of the products of the corresponding entries of the two sequences of numbers.

if $\vec{a} = \langle a_1, a_2, a_3 \rangle$ and $\vec{b} = \langle b_1, b_2, b_3 \rangle$, then $\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$

Geometrically, it is the product of the Euclidean magnitudes of the two vectors and the cosine of the angle between them

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \ \left\|\vec{b}\right\| \cos\theta$$

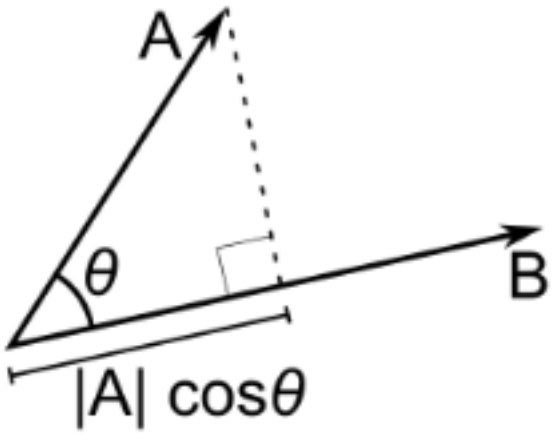


Figure 3: dotproduct

These definitions are equivalent when using Cartesian coordinates. Here is a simple proof that follows from trigonometry - http://tutorial.math.lamar.edu/Classes/CalcII/DotProduct.aspx

Related Link https://sergedesmedt.github.io/MathOfNeuralNetworks/VectorMath.html#learn_vector_math_diff

Dot Product and Vector Alignment

If two vectors are in the same direction the dot product is positive and if they are in the opposite direction the dot product is negative.

So you could use the dot product as a way to find out if two vectors are aligned or not. That is for any two distinct sets of input feature vectors in a vector space (say we are classifying if a leaf is healthy or not based on certain features of the leaf), we can have a weight vector, whose dot product with one input

feature vector of the set of input vectors of a certain class (say leaf is healthy) is positive and with the other set is negative. In essence, we are using the weight vectors to split the hyper-plane into two distinctive sets.

The first Artificial Neuron - Perceptron

The initial neural network - the Rosenblatt's perceptron was doing this and could only do this - that is finding a solution if and only if the input set was linearly separable. (that constraint led to an AI winter and frosted the hopes/hype generated by the Perceptron when it was proved that it could not solve for XNOR not linearly separable)

Here is how the Rosenblatt's perceptron is modelled

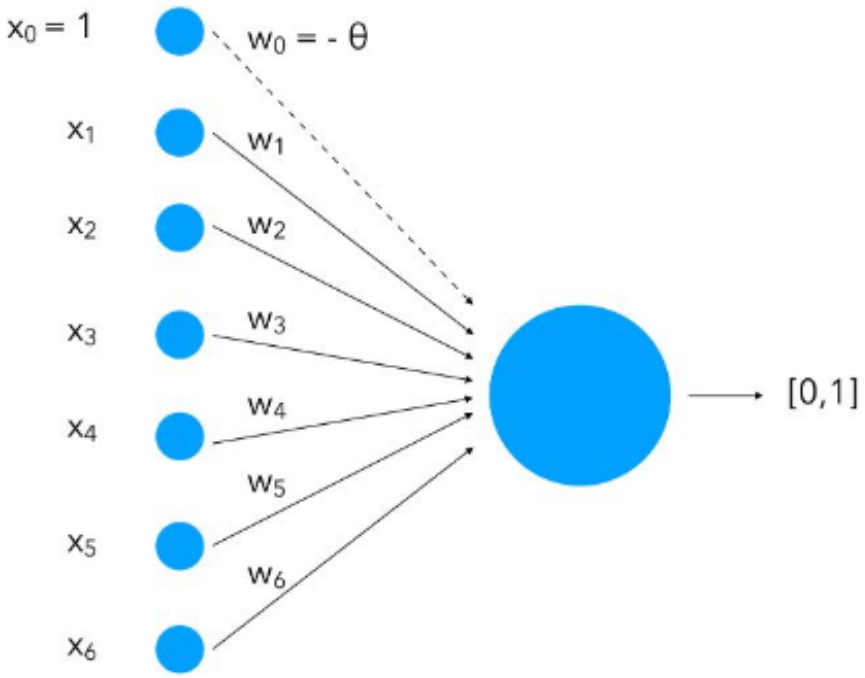


Image source https://maelfabien.github.io/deeplearning/Perceptron/#the-classic-model

Inputs are x_1 to x_n , weights are some values that are learned w_1 to w_n. There is also a bias (b) which in above is -θ If we take the bias term out, the equation is

$$f(x) = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

If we take a dummy input x_0 as 1, then we can add the bias as a weight w_0 and then this bias can also fit cleanly to the sigma rule

$y = 1$ **if** $\sum_i w_i x_i \geq 0$ else $y = 0$

This is the dot product of weight and input vector w.x

Note that dot product of two matrices (representing vectors), can be written as that transpose of one multiplied by another

$$\sigma(w^T x + b) = \begin{cases} 1, & \text{if } w^T x + b \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

All three equations are the same.

From https://sergedesmedt.github.io/MathOfNeuralNetworks/RosenblattPerceptronArticle.html

> So, the equation $\mathbf{w} \cdot \mathbf{x} > \mathbf{b}$ defines all the points on one side of the hyperplane, and $\mathbf{w} \cdot \mathbf{x} <= \mathbf{b}$ all the points on the other side of the hyperplane and on the hyperplane itself. This happens to be the very definition of "linear separability" Thus, the perceptron allows us to separate our feature space in two convex half-spaces

If we can calculate the weights then we can have a weight vector, which splits the input feature vectors to two regions by a hyperplane.

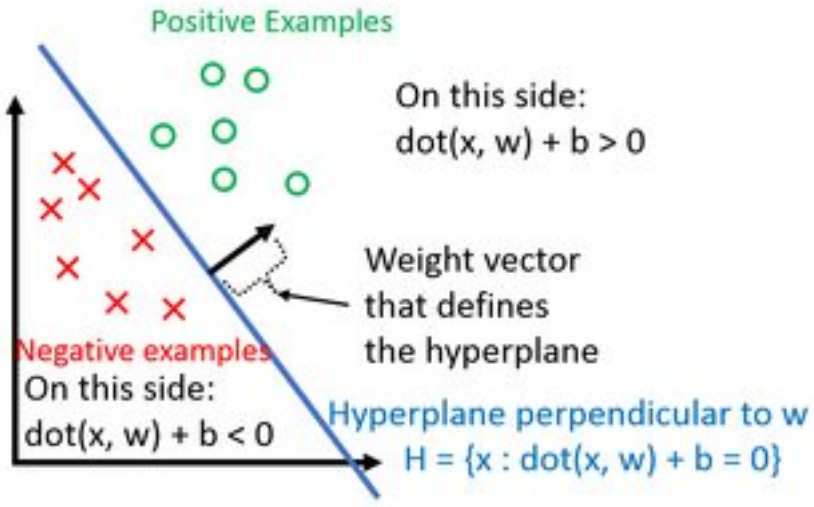
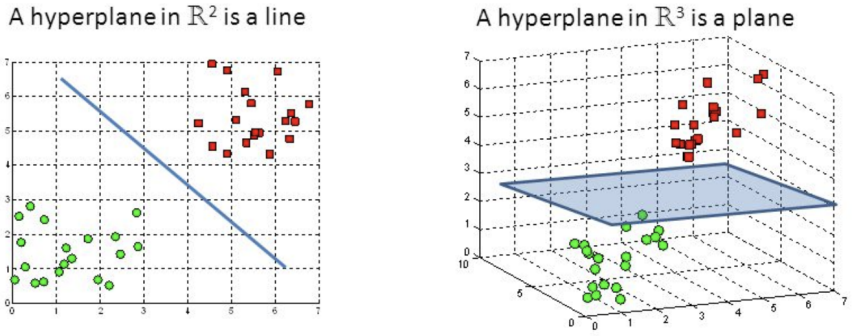


Image source https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/images/perceptron/perceptron_img1.png

In simple terms, it means that an unknown feature vector of an input set belonging to say Dogs and Cats, when done a Dot product with a trained weight

vector, will fall into either the Dog space of the hyperplane, or the Cat space of the hyperplane. This is how neural networks do classifications.



(*Hyperplane*)