Weights in Machine Learning

Originally published at https://blog.metaphysic.ai/weights-in-machine-learning/ January 12, 2023

In a neural network, weights are the learned traits that determine the strength of a connection (or *signal*) between any two of the neurons that make up the content of the network.

A neural network's core function is to make predictions based on learned data (i.e., the data that it was trained on). In order to do this, it has to perform a kind of repetitive triage on novel data (i.e., data it was *not* trained on), and then treat it according to the rationale and standards that it was taught by the original data.

The format or meaning of the weight will vary according to the type of network. In a language model, the weight's function may be to boost the emphasis on a word or language token; in a predictive analytics model, to favor a statistic or number, or some other outcome that can be expressed numerically; and in a multi-modal generative model such as Stable Diffusion (which synthesizes 'fake' photos by processing associations between language and images that it learned from studying real photos), to favor (or reject) an association between a semantic term and a visual feature.

Gold Weights

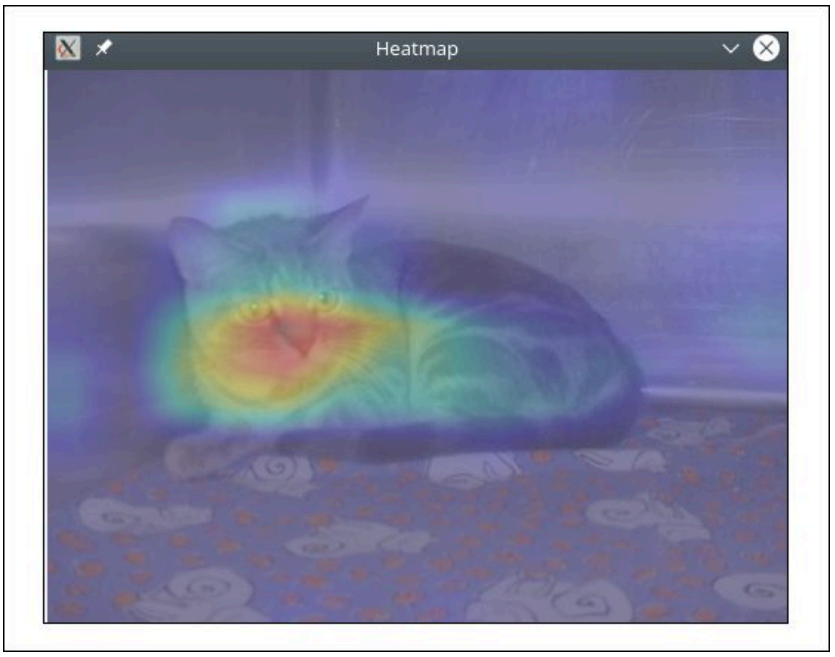
From this point of view, the neurons themselves can be considered base building units that are defined by the weights in much the same way that cement and architectural plans define a house (as opposed to the pile of bricks from which it was constructed). In machine learning research, the weights are *everything* – the ultimate 'gold' that emerges after weeks or even months of training a system.

By contrast, the surrounding architecture that's used to train the weights is valuable only if you have the resources to exploit it. Though some of the most powerful image synthesis systems are freely available in open source repositories, it's like getting the keys to a luxury car in a country where gas is $1000 a gallon and the minimum length of a car journey is 1000 miles. Therefore the prospect of re-training a formidable generative network from scratch is, in general, out of reach to amateurs and enthusiasts.

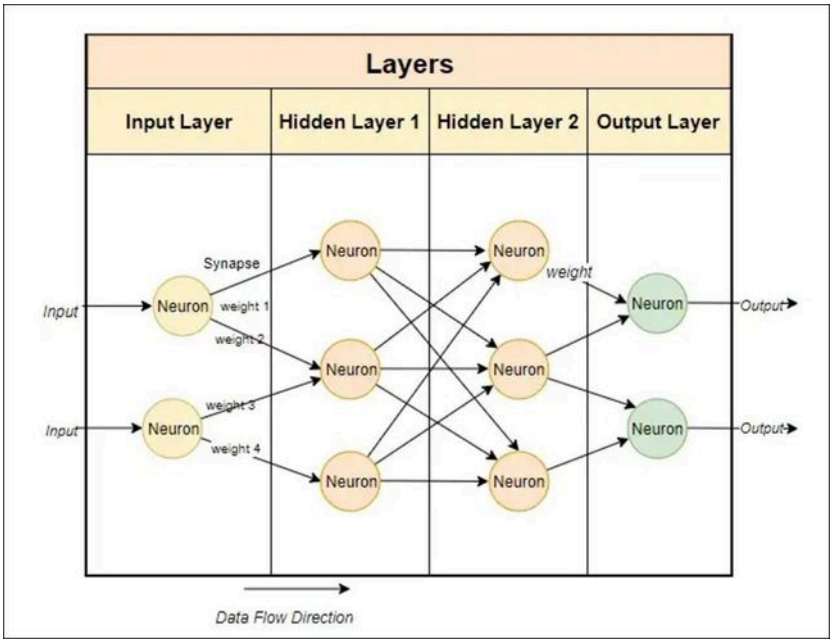
Hidden Layers

Neural networks consist of a complex array of neurons (also known as nodes), each of which contains an input, a weight, and a bias value. The weight itself is a specific interpretive connection between two neurons, while the bias vector is an additional weight that, unlike the other weights, is not linked to an input, and not linked to any of the other layers in the network.

A 'simple' weight decides how quickly an activation function will trigger (i.e., an action which will eventually lead to a prediction, such as a number, or an authentically deepfaked face), whereas a bias weight will delay the activation.
As such, the weight *itself* is biased, because it has very strong 'opinions' about the data that it's associated with, whereas the bias value, which lacks this bond, can regulate the activation function more objectively, so long as it was set up well, and accords with the logic of the dataset and the objective.



The weights are applied in *hidden layers* of the neural network. Examples of non-hidden layers would be the input layer (where data is entered into the network) and the output or *final* layer (where transformed data manifests and becomes available to the user).



Low weight values will let the data pass through essentially unaltered, but the higher the weight value, the more the weights will transform the data. In this sense, a transformation could be a number of potential operations, depending on the type of network and on the objective; but all such operations constitute a kind of value judgement on what should be done with the data, whether it should be prioritized and allowed to enable activations, and (together with the bias value), the *extent* to which the data should considered in the calculations.

Setting Up Weights and Biases

When a network is trained from zero, with no *a priori* knowledge of the potential features that it may eventually begin to discern in the training data, the weights are initially randomized, and gradually conditioned on the data until they begin to enable more and more accurate predictions.
During the training process, a neuron will calculate the *weighted sum* of the input data. This means that it will take the value of the data and perform calculations that conform the data to the schema of the network. After this, the bias value is added to the weighted sum that's been calculated.

At this point, the bias is acting as a normalization mechanism, further conforming the data to the objectives set up within the training architecture – and the bias value has a wider view of the overall objectives than the weights, which are instead concerned with more localized transformative properties.

Weights are calculated by a range of tertiary mechanisms, primarily through loss functions (such as Structural Similarity Index, Peak signal-to-noise ratio and Mean Absolute Error) – algorithms which govern the way in which these aforementioned mathematical operations are carried out, and by what principles.