The following work is a direct application of the proofs established in the paper "The Representation Theory of Neural Networks" written by Marco Antonio Armenta and Pierre-Marc Jodoin: https://arxiv.org/abs/2007.12213 .

1 Introduction

Neural networks are hard to study because of the nature of their structure. By using the mathematical theory of quiver representations, we can have an algebraic interpretation of those neural networks which then can be studied with algebraic methods. In particular, this interpretation allow us to compute the values of the neurons (hidden neurons and the outputs) of the neural network given a data sample x. This study gives a brief overview on how to compute those values by firstly mapping the neural network to the quiver space representation (i.e the moduli space M) and then using another mapping to get the outputs from the moduli space (Figure 1)

2 From neural network to the quiver space representation

A neural network can be described as an oriented weighted graph W and a non-linear activation function f. The edges of the graph consist of the neurons of the neural network, the vertices are the connections between the neurons and the weights of the graph are the parameters of the neural network (Figure 2).

If we think of each layer of the neural network as a matrix (where the coefficients of that matrix are the parameters) and given a data sample x like so:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} ; \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & a_{d3} & \dots & a_{dn} \end{bmatrix} ; \begin{bmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{d1} & b_{d2} & b_{d3} & \dots & b_{dn} \end{bmatrix}$$

then the mapping from the neural network to the quiver space representation is obtained as followed:

- for the first layer, we simply multiply all the elements of the j^{th} column of the matrix corresponding to that layer by the i^{th} element of the input vector x:

$$\begin{bmatrix} a_{11}x_1 & a_{12}x_2 & a_{13}x_3 & \dots & a_{1n}x_d \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & \dots & a_{2n}x_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d1}x_1 & a_{d2}x_2 & a_{d3}x_3 & \dots & a_{dn}x_d \end{bmatrix}$$

- for the n^{th} layer that is not the first layer, we first introduce a new vector which ultimately will become our new x. That new vector is constructed with the n-1 matrix/layer like followed:

$$X = \begin{bmatrix} \frac{f(a_{11}x_1+a_{21}x_1+\cdots+a_{d1}x_1)}{a_{11}x_1+a_{21}x_1+\cdots+a_{d1}x_1} \\ \frac{f(a_{12}x_2+a_{22}x_2+\cdots+a_{d2}x_2)}{a_{12}x_2+a_{22}x_2+\cdots+a_{d2}x_2} \\ \vdots \\ \frac{f(a_{1n}x_d+a_{2n}x_d+\cdots+a_{dn}x_d)}{a_{1n}x_d+a_{2n}x_d+\cdots+a_{dn}x_d} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_d \end{bmatrix}$$

We then proceed to embed the n^{th} matrix as previously but instead of using the input vector we now use the newly defined vector X:

$$\begin{bmatrix} b_{11}F_1 & b_{12}F_2 & b_{13}F_3 & \dots & b_{1n}F_d \\ b_{21}F_1 & b_{22}F_2 & b_{23}F_3 & \dots & b_{2n}F_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{d1}F_1 & b_{d2}F_2 & b_{d3}F_3 & \dots & b_{dn}F_d \end{bmatrix}$$

At the end, we obtain a set of matrices: the quiver matrices.

3 Computing the values of the neurons

Once we have the quiver matrices, it's fairly simple to compute the values of the neurons of a given layer. Indeed, we just need to propagate a vector of only 1 (ones) all the way through the desire layer and forget all the layers after it. Obviously, if we want the output of the neural network, we just need to propagate the vector on the entire set of matrices:

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11}x_1 & a_{12}x_2 & a_{13}x_3 & \dots & a_{1n}x_d \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & \dots & a_{2n}x_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d1}x_1 & a_{d2}x_2 & a_{d3}x_3 & \dots & a_{dn}x_d \end{bmatrix} \cdot \begin{bmatrix} b_{11}F_1 & b_{12}F_2 & b_{13}F_3 & \dots & b_{1n}F_d \\ b_{21}F_1 & b_{22}F_2 & b_{23}F_3 & \dots & b_{2n}F_d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{d1}F_1 & b_{d2}F_2 & b_{d3}F_3 & \dots & b_{dn}F_d \end{bmatrix}$$

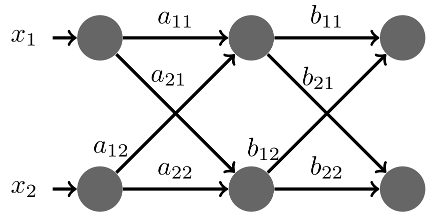
Note that the values obtain after propagating the vector of only ones are the values before applying the activation function.

4 The code

The "QuiverRepresentation" class requires 6 inputs: the data sample, the dimension of the feature vector, the parameters of the (trained) neural network, the total number of layers in the model, the layer for which we want to compute the values of the neurons, a list consisting of the activation function (relu, sigmoid, tanh or softmax) applied at each layer.

$$\begin{tikzcd}
\mathbb{R}^d \arrow[rr, "\Psi(W,f)"] \arrow[dr, "\varphi(W,f)"'] & & \mathbb{R}^k \\
& \mathrm{M} \arrow[ur, "\hat{\Psi}"'] &
\end{tikzcd}$$

Figure 1: A neural network is a mapping $\Psi(W, f)$ from a feature space of dimension d to an output space of dimension k where W is the graph structure of the neural network and f the non-linear activation function. The quiver space representation of a neural network consists of a mapping $\varphi(W, f)$ to a space M of high dimension where everything is linear. The dimension of that space is: (number of vertices - number of hidden neurons). The mapping $\hat{\Psi}$ allo0w us to retrieve the outputs of the neural network from the space M.



$$\boxed{\begin{array}{l}\text{Outputs of the hidden layer}\\ f_1(x) = f(a_{11}x_1 + a_{12}x_2)\\ f_2(x) = f(a_{21}x_1 + a_{22}x_2)\end{array}}$$

$$\boxed{\begin{array}{l}\text{Outputs of the neural network}\\ f_1(x)b_{11} + f_2(x)b_{12}\\ f_1(x)b_{21} + f_2(x)b_{22}\end{array}}$$

Figure 2: The graph structure of a multilayer perceptron and the outputs of each layer given a non-linear activation function f.