Transforming to Yoked Neural Networks to Improve ANN Structure

Anonymous Author(s)
Affiliation
Address
email

1 Appendix

Due to the limit on paper length, this section will provide a comprehensive explanation of the YNN learning process. As outlined in the original paper, we will delve into the learning through both the forward process and the backward process.

1.1 Forward Process

If we choose to yoke the nodes of the level i, the forward process involves two steps. Firstly, based on the real values of the nodes in the level $i-1$, we compute the meta value of each node in level i using:

$$\widehat{N}^i = f(N^{i-1}) * W^{(i-1)i}. \tag{1}$$

The process is illustrated in Figure 1, where the green nodes correspond to the meta value of the nodes.

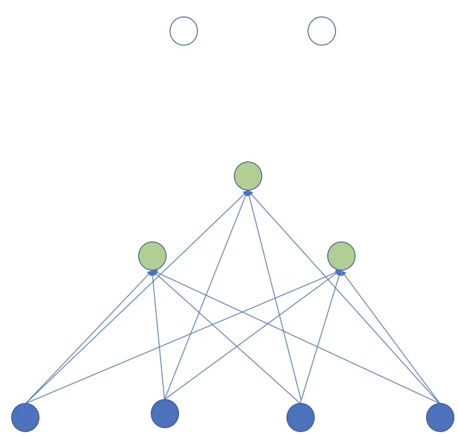

Figure 1: Fig 1.

Secondly, if the nodes of the level are yoked together, the real value of each node can be determined using its meta value along with the real values of the other yoked nodes. In this case, the real values of the nodes act as variables. Since these variables rely on each other, they form a system of equations as shown below.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

$$
\begin{cases}
w_{01}^i + \sum\limits_{j \neq 1} f(n_j^i) * w_{j1}^i + f(\widehat{n}_1^i) * w_{11}^i = n_1^i \\
w_{02}^i + \sum\limits_{j \neq 2} f(n_j^i) * w_{j2}^i + f(\widehat{n}_2^i) * w_{22}^i = n_2^i \\
\dots \\
w_{0m}^i + \sum\limits_{j \neq m} f(n_j^i) * w_{jm}^i + f(\widehat{n}_m^i) * w_{mm}^i = n_m^i
\end{cases}
$$

The yoking process mentioned above is depicted in Figure 2. It is worthwhile noting that , according to the system of equations, the self-spin weight in the bidirectional complete graph corresponds to the meta value of each node. As a result, the meta values of the nodes can be extracted, as demonstrated in Figure 3 and Figure 4.

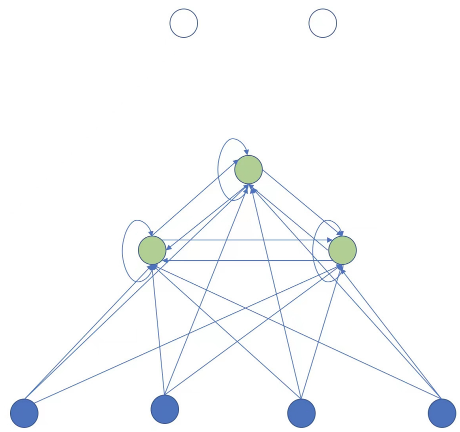

Figure 2: Fig 2.

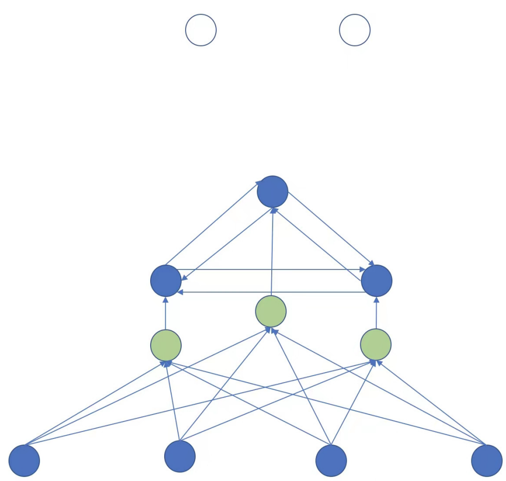

Figure 3: Fig 3.

1.2 Backward Process

In our YNN model, the backward process of the yoked nodes aligns with the forward process. Firstly, if we obtain the gradient of the meta value for the nodes in level $i+1$, we can proceed to determine the gradient of the real values in the current level. This is due to the fact that the real values in the

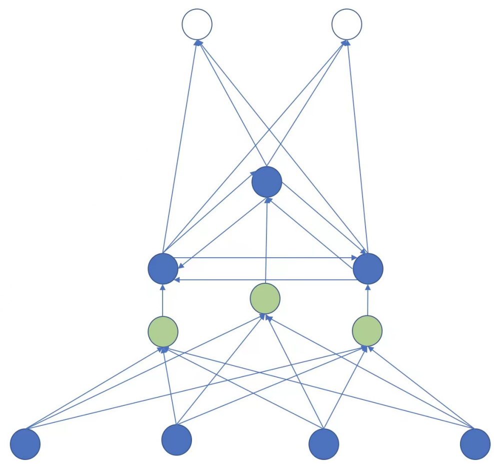

Figure 4: Fig 4.

current level determine the meta value of the next value, as indicated by Equation 1. Therefore, the gradient of the real values in the current level is computed as follows:

$$d(N^i) = d(\widehat{N}^{i+1}) * W^{i(i+1)T} * f^{-1}(N^i)\ . \tag{2}$$

In the backward process, we initially compute the gradient of the output as the gradient of the meta value for the nodes in the last level. Since the node is activated by the function f, we multiply its inverse function as part of the calculation.

Secondly, in the forward process, the real values are calculated from their corresponding meta values based on the system of equations. We have divided the process into two steps, as depicted in Figure 3. According to the system of equations, the diagonal of the adjacency matrix represents the weights for the meta value. To facilitate the peeling process, we introduce the operator

$$C^i = W^i - diag(W^i) + eye(W^i), \tag{3}$$

as well as operator

$$diag(W^i)\ . \tag{4}$$

For operator C_i, we begin by peeling its diagonal matrix and subsequently adding an eye matrix. This is necessary because the gradient of the node is not only influenced by other yoked nodes but also by the passed-down gradient of the node itself that needs to be back to the lower levels. The combine peeling steps can then be calculated as

$$d(\widehat{N}^i) = d(N^i) * C^{iT} * f^{-1}(N^i) * diag(W^i) * f^{-1}(\widehat{N}^i)\ . \tag{5}$$

. By following this approach, we can obtain the gradient of the meta value using the gradient of the real value. Subsequently, the backward process can proceed to lower levels. As the node is activated by the function f in the system of equations, we multiply its inverse function at this stage.

1.3 The Gradient of The Weights

Finally, we consider the gradient of the weights which is derived from the gradient of the meta values and the gradient of the real values.

Regarding the weights between levels, in the forward process, they are determined by Equation 1. Then their gradient can be calculated as

$$d(W^{i(i+1)})^T = d(\widehat{N}^{i+1})^T * f(N^i)\ . \tag{6}$$

Furthermore, in level i, the weights are determined by the system of equations. For each equation, the relevant diagonal weights act on its meta value. Accordingly, if we have m nodes in level i, the gradient of the weight connected to the jth node can be calculated using the introduced operator

$$D^i_j = (n^i_1, ..., \widehat{n}^i_j, ..., n^i_m)\ . \tag{7}$$

The gradient of the weight corresponding to node j in level i can be calculated by

$$d(W^i(:,j))^T = d(n_j^i) * f(D_j^{i'}) \,. \tag{8}$$

1.4 The Essence of YNN

The fundamental concept of YNN lies in the information flow within a list of cliques. Each clique can be considered a neural module, and a collection of cliques forms a list of neural modules, as depicted in Figure 5. Within a clique, information may circulate in cycles. Due to the interdependencies among the nodes, a solution to the system of equations represents a balanced state for the neural module. The meta values of the nodes serve as constraints for achieving this balance within the neural module.

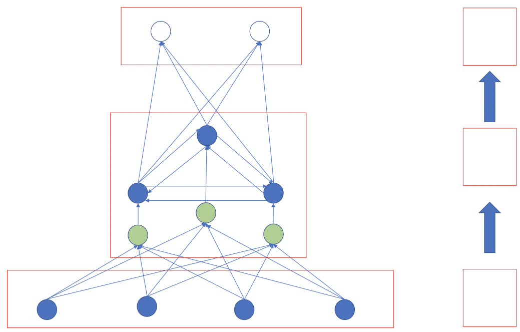

Figure 5: Fig 5.