Description of the Branch Model method

A. Gillgren[1]

[1] Chalmers University of Technology, Gothenburg, Sweden

E-mail: andreas.gillgren@chalmers.se

1. Branch Model method demonstration

1.1. Visualizing neural network mappings

As a first step in demonstrating the Branch Model method, we here show how interpretability can be facilitated by visualizing neural network mappings.

Consider a toy data set generated with the equation

$$y = A\sin(10x) \tag{1}$$

where y is an output parameter of interest, and where A and x are two independent input parameters randomly sampled from a uniform distribution between 0 and 1. A neural network can be trained on the data set to predict y from A and x. In other words, the neural network learns a functional mapping f, where $\hat{y} = f(x, A)$, as illustrated in Figure 1. Here, $\hat{y}$ represents the prediction of the output, in contrast to the actual output y.

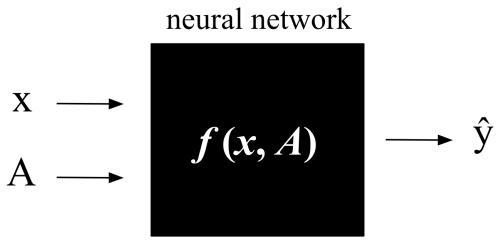


Figure 1: An illustration of a neural network designed to predict the parameter y from x and A. f represents the functional mapping of the network.

Assume that post training, the neural network is able to make accurate predictions. Also pretend that the true underlying Equation (1) is unknown, and that our goal is to investigate the relationship between the parameter of interest y and the inputs x and A. Usually, obtaining an understandable interpretation of the functional mapping of a neural network is a non-trivial task due to the many connecting nodes in a neural network (the black-box problem). Fortunately, in this case, having only two input parameters enables another method for interpreting the functional mapping: visualization. Post training, it is possible to parse data through the model and plot the predicted output $\hat{y}$ as a function of the inputs. For instance, $\hat{y}$ can be plotted versus x with A dictating the color, as in Figure 2.

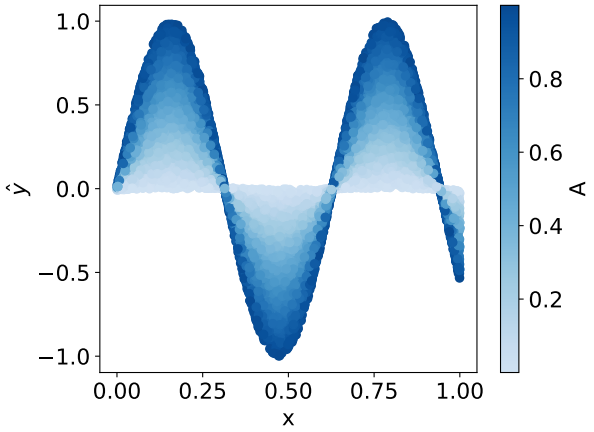


Figure 2: A visualization of the functional mapping learned by the neural network that predicts y from x and A, trained on a data set generated with Equation (1). The data points in this figure are obtained through parsing data through the neural network.

This visualization shows what the neural network has learned about the relationship between the input and output parameters. In particular, the neural network has learned a sine-relationship between $\hat{y}$ and x, where A affects the amplitude. This is in full agreement with the original Equation (1) that was used to generate the data. Note that this provides a global interpretation of what the model has learned across the full input domain, and not only for individual predictions. Note also that the trainable parameters of the network do not need to be analyzed to interpret the model, as all we need for the visualization are the predicted output $\hat{y}$, along with the inputs x and A. In essence, visualizations of neural networks facilitate a qualitative approach to investigating parameter dependencies. The advantage with neural networks compared to power scalings and other simpler models is that neural networks can highlight more complicated relationships.

Of course, with only two input parameters, it is not even necessary to use a neural network to investigate the relationship between y and the two inputs. We could simply plot the true output y versus x and A directly. However, the principle of visualizing neural networks serves as the cornerstone for the method we propose in the next section, which is why its introduction and emphasis are warranted here.

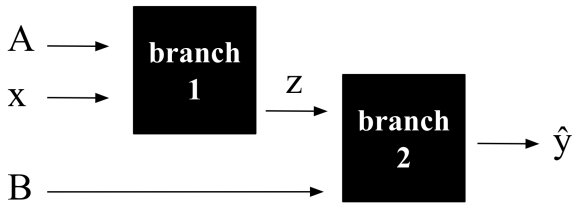


Figure 3: An example of a Branch Model. Each branch (boxes) contains dense neural network layers, and outputs one value that either is forwarded to another branch, or set as the output of the model. In this case, branch 1 calculates an intermediate value z from the input parameters A and x. This intermediate value is then forwarded to branch 2, which calculates the predicted output $\hat{y}$ from z and B.

1.2. Interpretable branch-based architecture

In this section, we present an alternative neural network architecture that achieves interpretability through visualization, even for cases with more than two input parameters.

Consider a new toy data set generated with the equation

$$y = A\sin(10x) + B \tag{2}$$

where similarly as the previous example, A, x and B are inputs that are randomly and independently sampled from a uniform distribution between 0 and 1. Assume again that we are unaware of the true Equation (2), and that our goal is to find the relationship between y and the three input parameters. A neural network could be trained to predict y from A, x, and B. However, since there are more than two inputs, it would be challenging to visualize the dependencies in the same manner as in Figure 2.

To solve this issue, we propose an approach based on splitting the neural network architecture into branches, such that each branch only has two input parameters and one output parameter. An example of such architecture is illustrated in Figure 3. We refer to this type of architecture as a *Branch Model*. By only allowing two parameters to be parsed through each branch, visualization is enabled, and thus interpretability is achieved for the full model. Additionally, since each branch essentially is a neural network, high expressive capacity is maintained.

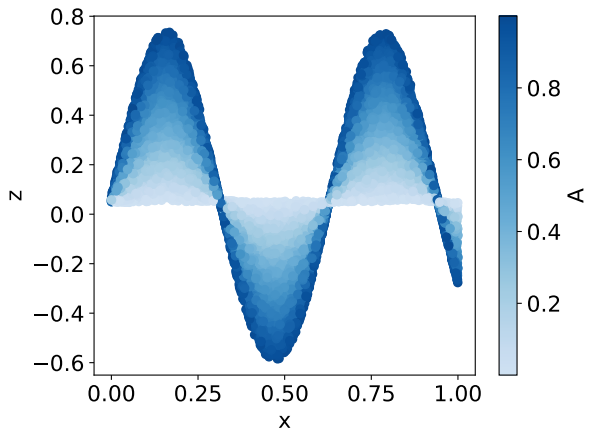
By training a Branch Model with the same architecture as illustrated in Figure 3, on a data set generated with Equation (2), a model that makes accurate predictions of y is achieved. To investigate the functional mapping that has been learned, data is parsed through the model, and the predictions of branch 1 and branch 2 are visualized in Figure 4a and Figure 4b respectively. In summary, these plots show that $\hat{y}$ is obtained through an addition-like operation between $z(x, A)$ and B, where the intermediate parameter z exhibits a sine-like dependency to x with A determining the amplitude. As in the first example, this is in full agreement with the underlying equation used to generate the data. In a broader context, through the joint analysis of Figure 4a and Figure 4b, we achieve a transparent qualitative description of the relationship between the input parameters (x, A and B) and the predicted output parameter $\hat{y}$. Note that for this example, it is necessary to analyze the intermediate parameter z to grasp the relationship between the predicted output $\hat{y}$ and the inputs x and A. We also emphasize that even though the branches in this model essentially are individual neural networks, they are trained together as one model. This means that prior knowledge of the intermediate parameter z is not required.

The Branch Model is inspired by NAMs [1], which are also based on the concept of splitting the neural network architecture into several parts and by interpreting through visualizations. However, the Branch Model addresses two of the limitations of Neural Additive Models (NAMs): 1) enforced summation of network outputs, and 2) the restriction to pairwise interactions, as each network in a NAM is limited to a maximum of two inputs. In the example presented in this section, branch 2 does indeed learn an addition-like operation between z and B, mirroring the structure of the true underlying equation. However, the Branch Model is not restricted to additive operations in the final step of the prediction like NAMs are. The Branch Model could have learned any relevant functional relationship. Moreover, while each individual network within the Branch Model is constrained to pairwise interactions, the overall model is not. For instance, in Figure 4, branch 1 can learn interactions between A and x, while branch 2 can learn interactions between z and B. Since z is calculated from A and x, the model can implicitly capture interactions among all three input parameters, thus extending beyond the pairwise limitation of NAMs.

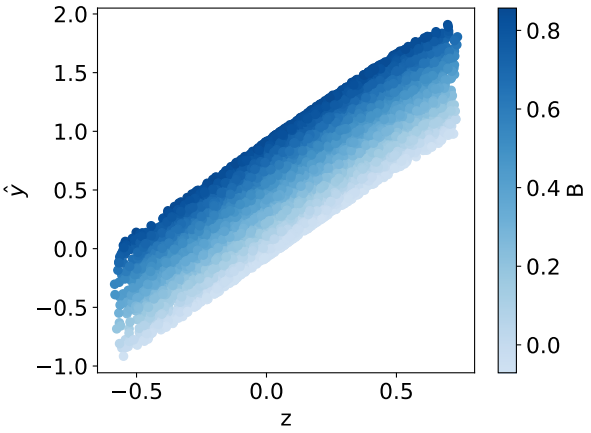
1.3. Determining which inputs belong to which branch

In the previous section, it is demonstrated how a Branch Model is able provide insight about the relationship between the predicted output $\hat{y}$ and the three input parameters A, x, and B in a data set generated by Equation (2). However, it is yet not explained how we determine, without prior knowledge of the data, that A and x is the appropriate choice of parameters to be parsed through branch 1 (see Figure 3).

We determine which inputs belong to which branch by testing all possible setups and by evaluating the error



(a) Visualization of the mapping of branch 1 in Figure 3.



(b) Visualization of the mapping of branch 2 in Figure 3.

Figure 4: Visualizations of the two branches in Figure 3.

of each setup. Here, we refer to "setup" as how the different inputs are assigned to the different branches. The idea behind this approach is that the setup that result in the lowest error is able to best capture the true relationships in the data. Optimally, the best setup shows a similar level of error as a regular neural network, which may act as a benchmark model when evaluating setups. For cases with more input parameters, the search process occurs in subsequent steps, as illustrated in Figure 5.

To demonstrate the setup evaluation procedure in practice, we again consider the data set generated with Equation (2) and a Branch Model with the same architecture as in Figure 3. We test which pair of input parameters that produce the lowest error in the predicted output $\hat{y}$ when being parsed through branch 1 (while the third input is being passed directly to branch 2). The result of this test is shown in Figure 6. As expected, the most appropriate setup is to pair A and x to be reduced to z, and then to parse z together with B through branch 2. Otherwise, the consequence would be a series of parameter reductions that inadequately reflect the data, hence the larger error of the two other setups.

1.4. When input parameters are not separable

The method presented so far is designed to be useful for data sets in which the input parameters are easily separable, such that each parameter needs to be present in only one branch to accurately reflect the data. However, this is not always the case. For instance, consider a new toy data set generated with the equation

$$y = (a + b)\sin(c + b) \tag{3}$$

where again, a, b and c are independent input parameters, and where y is the output to be predicted. In this case, b affects both the amplitude and the argument of the sine-wave, and is therefore not easily separable. To achieve accurate predictions with a Branch Model, b must be parsed through two branches, as illustrated in Figure 7.

We can investigate the separability of input parameters using the following approach: when testing different setups of input parameters, see Figure 5, we initially restrict each input parameter to be parsed through only one branch. If none of these setups yield an error close to the benchmark error obtained with a regular neural network, we then consider all setups where one input parameter is allowed to be parsed through both branches. If yet none of these setups yield an error close to the benchmark error, it is possible to allow up to two input parameters to be parsed through both branches, and so on. Data sets with highly inseparable inputs are more difficult to interpret, and may limit the applicability of a Branch Model for such scenarios. This is however not only a challenge when considering a Branch Model, but when considering fitting models for interpretation purposes in general.

1.5. Branch Model summary

We here provide an overview summary of the key aspects of the Branch Model method to repeat and emphasize its motivation and purpose:

- Regular neural networks have a high expressive capacity but are black-boxes, meaning that they lack in interpretability.
- At the opposite end of the spectrum, simple models like power scalings provide interpretability but are limited in their expressive capacity.

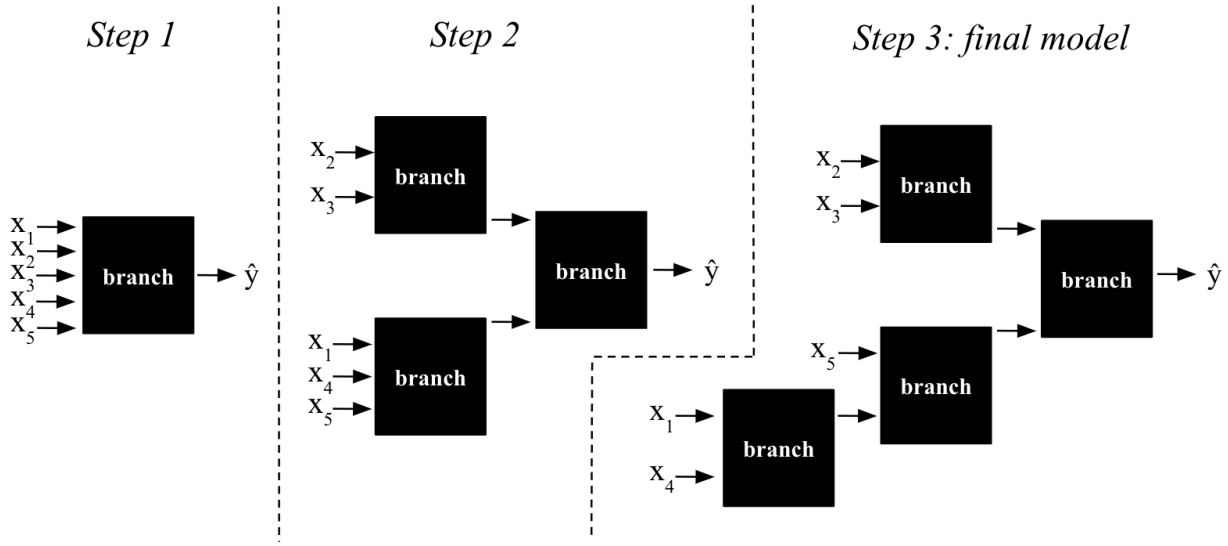


Figure 5: For cases with more than 3 input parameters, the search for the appropriate branch setup occurs in steps. In this example, there are five input parameters: x_1, x_2, x_3, x_4 and x_5. As a first step, a regular neural network is used to obtain a benchmark value for the error. In step two, the input parameters are split into two branches. For this example, we find that the split $\{x_2, x_3\}$ $\{x_1, x_4, x_5\}$ leads to the lowest error after testing all possible splits. In step 3, another branch is added to the bottom branch from the previous step, such that only two parameters are parsed through each branch in the final model. Note that in step 3, x_2 and x_3 are locked as inputs to the upper branch since we know from step 2 that this is appropriate. In other words, step 3 focuses solely on exploring all possible ways to parse x_1, x_4 and x_5 through the two lower branches, and finding which setup leads to the lowest error.

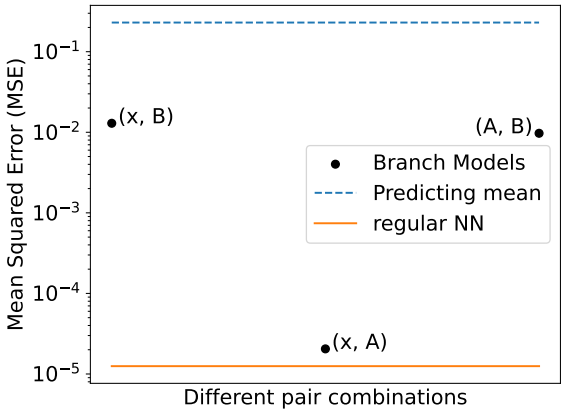


Figure 6: The mean squared error (MSE) of different Branch Model setups (black dots). The letters at each dot indicate which input parameters are parsed through branch 1 (see Figure 3). For reference, the MSE is also shown for a regular neural network (shortened as NN, solid line) and for a model that always predicts the mean value of the output in the database (dashed line).

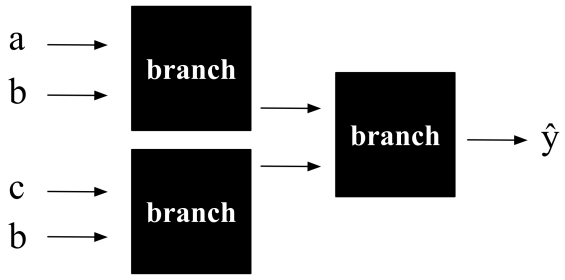


Figure 7: Branch Model architecture when predicting y from a, b and c, trained on a data set generated with Equation (3). Since b is not separable from the other input parameters, it needs to be parsed through two branches to accurately reflect the data.

- We introduce a neural network architecture that we refer to as a Branch Model, consisting of neural network branches. The primary objective of the Branch Model is to offer both interpretability and high expressive capacity.
- Each branch in a Branch Model has two input parameters and one output parameter. By plotting the output of each branch versus its two inputs, we

can interpret the relationship between the inputs and the output. Thus, interpretability is achieved through visualization.

- The validity of a specific Branch Model setup is assessed by comparing the error of the model predictions with the error of a regular neural network, which acts as a benchmark model.
- In addition to showing the relationships between the input parameters and the output parameter, the final Branch Model structure unveils the hierarchical sequencing through which input parameters are successively paired and reduced until reaching the output parameter.
- As the Branch Model is transparent, it is straightforward to detect extrapolation in the multi-dimensional input space. For instance, Figure 4a shows the combinations of x and z where there are no samples in the training data, which indicates areas where predictions would likely be unreliable due to extrapolation.
- For similar reasons as outlined in the previous point, overfitting is easier to detect with a Branch Model compared to a black-box neural network. If the model has captured irregular, overly complicated patterns that appears to perfectly fit all the data points, including noise, this will be indicated in the visualizations.
- We expect Branch Models to be useful for cases with relatively few ($\approx< 10$) input parameter. This is because more inputs necessitate the incorporation of more branches, which makes the interpretation process more difficult since all branches need to be interpreted together to obtain a full interpretation of the model. Additionally, we expect Branch Models to be useful when input parameters are relatively separable.

References

[1] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey Hinton. Neural additive models: Interpretable machine learning with neural nets, 2020.