Architectures of Intelligence
Assignment 7: Learning in Nengo

Matthijs Prinsen *(S4003365)*
Marinus van den Ende *(s5460484)*
Group 75

January 12, 2025

1 Exploring Weights

Why do we get a 10x10 matrix? What do the numbers represent?

The learning_connection weights between the pre_learning and post_learning ensembles determine the value by which the signal is multiplied before being summed into the new neuron's signal. There is a 10x10 matrix because we are dealing with two sets of 10 neurons. The matrix weights show how much each neuron of pre_learning influences each neuron in post_learning.

Can you see a relation between the values of both ensembles for different runs? Explain why there is a relation or not.

There does not seem to be a clear one-to-one relationship between the ensembles over different runs. However, there is a slight trend where part of the post_learning output often follows the pre_learning input. Since the weights are randomly determined, it is inevitable that some runs capture part of the signal better. The small number of neurons increases the error.

Explain why post_learning does not represent the same value as pre_learning, even though the neurons are directly connected.

While the neurons are directly connected through the identity matrix, the post_learning ensemble does not represent the same value as pre_learning. Although the ensembles are connected one-to-one, ensuring a direct mapping between the neurons of the two ensembles, the underlying encoding and decoding processes in pre_learning and post_learning are not aligned.

Additionally, the neurons in the pre_learning and post_learning ensembles are initialized with randomly generated activation curves. These random curves cause the output to be chaotic when signals from pre_learning are passed through them. This encoding/decoding mismatch, combined with the randomness in the activation curves, is why there is a discrepancy between the values of pre_learning and post_learning.

What would be the requirement for the above to work?

For both ensembles to produce the same values, their seeds must be set to the same value. If the seed is the same, both ensembles will have the same generated weights and, consequently, identical encoders and decoders. This alignment ensures consistent values between the ensembles across any run.

2 Learning Weights

What do you think the connection weights will be in this case?

The connection weights are initially set to zero. Specifically, they are set to an array of zeros with a length equal to the number of neurons. This is because we want the weights to start at zero and then either increase or decrease during training.

What happens if you set the learning rate to 0.01 or 0.1? Can you think of an explanation?

The learning rate determines the magnitude of weight adjustments after each error calculation. Nengo's PES learning rule employs a variant of gradient descent, so the learning rate should be low enough to avoid getting stuck in local minima but not so high that it causes overshooting of the global minimum. A learning rate of 1e-4 is optimal because it ensures steady progress toward the goal with each increment. Setting the learning rate higher (e.g., to 0.01 or 0.1) results in significantly larger weight updates, causing overshooting in nearly every iteration.

We now developed a model that learns the connection weights between two ensembles. However, to do so, we had to specify the error: basically reintroducing pre-calculated connection weights. Think about how this would work when doing, for example, alphabet-arithmetic. Where does the error signal come from in that case? What are therefore the requirements for using supervised learning?

Supervised learning models always require correctly labeled input-output pairs. To train such a model, the output is compared to the expected result, and an error value is calculated based on the difference. This error guides the model in adjusting its weights to improve performance.

For the Alphabet-Arithmetic model, the learning system needs to have the correct solutions for all inputs. These solutions could come from pre-calculated values, lookup tables, or hardcoded pairs. The model uses these solutions to calculate the error, determining how far the predicted output deviates from the correct answer.

This reliance on pre-known correct outputs highlights a key limitation of supervised learning: it struggles in scenarios where the correct output is unknown or uncertain.