Neuroscience 299: Computing with High-Dimensional Vectors
Assignment 5: Resonator networks
Due October 6, 1pm

Reminder: Please do *either* the writing assignment *or* the programming assignment. Expected length for the writing assignment is approximately 250-500 words, but there is no strict minimum or maximum.

Writing assignment:

Pick either option A or option B (you need **not** and should **not** answer both).

Option A: *New applications?* From the focus paper, you've seen examples of 1) searching a tree data structure and 2) visual scene analysis cast as a factorization problem. Can you come up with another factorization problem that the resonator network is good for?

In your response, please be sure to address the following:

1) The kind of data structure/representation that the problem is addressing
2) How the data structure is encoded with VSA operations
3) How to perform interesting tasks for the network.

For a general template of how to compose this, see p. 2318-2321 of Frady et al. (2020), "Resonator Networks, 1".

Option B: *Resonator Networks and Hopfield Networks*. Compare and contrast the Resonator network with the Hopfield (1982)[1] network.

- In what ways are the dynamics (update rules) of the resonator network like those of the Hopfield network?
- What is the logic behind the components of the equations?
- Can we think of both networks as descending an energy (Lyapunov) function? Why is this significant?
- How might a better understanding of performance of Hopfield networks help us analyze the properties of resonator networks?

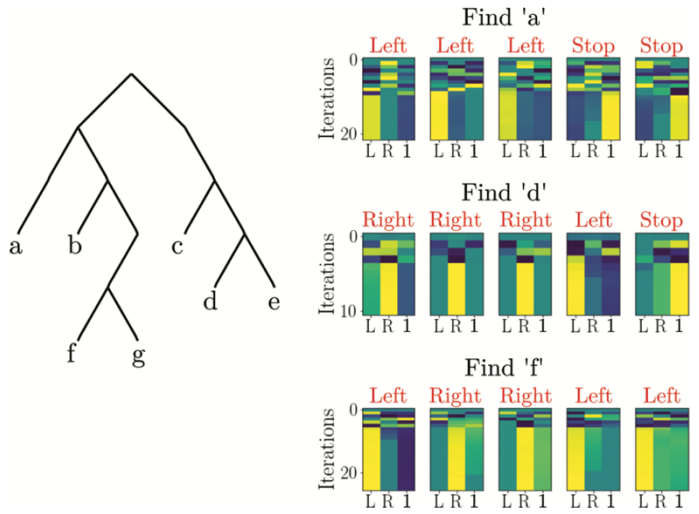
Note that the second prompt may require some prior knowledge about the literature surrounding Hopfield networks, but we will return to related issues in the "Information theory" module. You may also find Kent et al. (2020), "Resonator Networks, 2" helpful.

[1] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities.

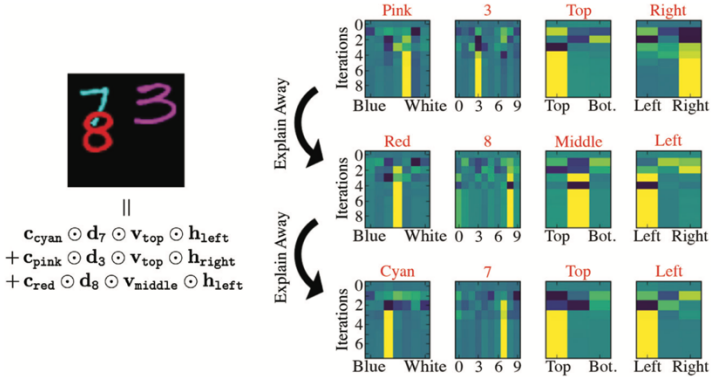
Programming assignment:

The main goal of this assignment is to implement a resonator network as described in "Resonator Networks, 1". **You should replicate <u>one</u> of the following figures, completing steps 1-8:**

- Figure 2: Tree search with a resonator network.



- Figure 4: Scene vector **s** is fed into a resonator network that decodes each object in the scene.



You may use any of the HD Computing/VSA Models we've introduced so far, but out of simplicity (and convenience of self-inverse binding) we recommend working with bipolar vectors (Multiply-Add-Permute).

To this end, please complete the following steps:

Step 1. Import your VSA functions (binding, bundling, permutation) – these can be re-used from the previous assignments.
Step 2. Generate "codebooks": random vectors for each of the elements that can be bound together.

- For Figure 2 (tree), the codebook consists of three vectors: “Left”, “Right”, and “1”. Note that the “1” vector whose indices are all +1 and thus not random.
    - You should also generate random vectors for the items in the tree (i.e. ‘a’, ‘b’, …)
- For Figure 4 (visual scene), there are four codebooks: for color, digit identity, vertical position, and horizontal position.

Step 3. Implement the data structure corresponding to your figure. You should use the same encoding strategy, but you can create a tree (visual scene) with different branches/elements (colors/numbers/locations).
Step 4. Implement a function performing the resonator network update step. That is, your function should take as inputs the previous state of the network and the codebook and output the next state. (The state is a set of vectors representing the estimates of each factor.)
Step 5. Create a for-loop that performs k updates of the network (k is a parameter that you can choose). Set the initial state of the network to be the superposition of elements in the codebook.
Step 6 (visual scene only). Implement the “explaining away” feature by subtracting
Step 7. Plot your results in a style that is similar to your chosen figure. Pick a dimensionality that is high enough for the network to converge.
Step 8. Experiment and comment on what dimensionality is needed for the resonator network to factorize correctly.

- What seems to be the minimum dimension (experimentally) for success? Explain and plot your findings.
- How does dimensionality affect convergence speed (i.e., the number of updates to obtain the correct factorization)?
- How do you interpret the behavior of the network when the dimensionality is too low to converge correctly?

General advice:

- You may find it helpful to *not* normalize the scene vector.
- The outer product of the codebooks (in the update rule) is a rather large matrix that might be slow to multiply. Consider performing two matrix multiplications instead (to save time and memory).
- With these two examples, 20,000 dimensional vectors should more than suffice. In fact, it’s very likely you need less (how much less is up for you to find out!).

Bonus questions*:*

- This assignment is more focused on a qualitative demonstration that the resonator network *can* work, rather than a quantitative assessment or theoretical analysis of performance. Can you think of quantitative metrics that can more rigorously define “good performance”?
- How does the number of items in the codebook affect the success of the resonator network?
- Change the update equation of your resonator network to work with Ordinary Least Squares (OLS) Weights. See Kent et al., “Resonator Networks, 2”, p. 2337-2338 for implementation details. Does this change make a difference to your network’s performance?
- Consider an asynchronous update to the factors. See Kent et al., “Resonator Networks, 2”, p. 2337 for implementation details. Do you notice any changes in performance?