Computation & Cognition: Assignment 4
Connectionism

Adrien Joon-Ha Im (3984389)

April 2025

1 Introduction

In this report we will explore the connectionist model described by McClelland in his 1981 paper *"Retrieving General and Specific Information From Stored Knowledge of Specifics"* [1]. This work describes a foundational approach to the development of neural network models. It describes a model in which different instance and property nodes are connected in a network, and shows us how such a network can store and retrieve knowledge through interactions between these nodes.

The model consists of two types of nodes: instance nodes and property nodes which are interconnected by excitatory connections. These connections represent associations between individuals and their attributes. The model also includes inhibitory connections within cohorts, where nodes can be classified into similar categories. Examples of cohorts in the implemented model include for example age groups, or marital status of each individual. The individual nodes within each of these categories are inhibitory, and should not activate each other.

Before starting the assignment, I reviewed the paper to understand how the model works. I will be implementing an Interactive Activation and Competition (IAC) model that simulates the one described in McClelland's paper. I will be explaining how the dataset was prepared, how the network was set up, and the dynamics that exist within this model. I will then be exploring the way we can retrieve information and connections in the network by probing for individuals, but also groups of traits. We will be able to retrieve common properties for a certain group by probing it in the network, but also we will be able to find individuals that have a specific characteristic by probing the characteristic and looking at the activation levels of different individuals or groups.

2 Preparing Data and Setting Up Network

In order to develop my own implementation of the IAC model, I first downloaded the `dataset.csv` file that shows a matrix of the excitatory connections between the different nodes. The excitatory connections were given in the file as 1 when two nodes have an excitatory connection, and otherwise they were denoted as 0. The matrix also contained the list of node names, which included instance nodes (e.g., individual gang members) as well as property nodes (e.g., age group, education level, marital status, and occupation).

In order to store the connection information between the nodes, I used a NumPy array. This allowed me to store the matrix information in a computationally simple format, which allows for fast arithmetic computation when calculating the activation of each node.

In addition to the excitatory connections, I also created a second matrix for inhibitory connections. I grouped the different nodes that belong to the same category. For example, age, education, are mutually inhibitory nodes, and were grouped together into cohorts, called `inhibitory_groups` in my code. This is because nodes that are in the same groups cannot both be excited at the same time. For example a person cannot be both in their 20s and in their 30s simultaneously.

Then, I implemented the `IACModel` class, which takes the excitatory matrix, inhibitory matrix, as well as the parameters that are linked to node activation according to McClelland's model. These include resting activation, decay parameter, initial activation, and both inhibitory and excitatory weights. All of these values were initialized with the values given in McClelland's paper [1].

This setup created the foundation of my IAC model, which simulates the networks dynamics that were described in the paper. By using the same parameters and initializing them to the same values as in the previous study, I ensured that my implementation would replicate the expected behavior of the model.

3 System Dynamics

Before looking at how the network behaves as a whole, it is important to first start by looking at how an individual node is activated. In McClelland's IAC model, each node has a resting activation level R. In our model we used the same value of $resting\_activation = 0.1$, which is the value at which activation decays when there is no input.

The dynamics of the activation of the model can be resumed in four important equations, described on page 171 of McClelland [1]. These four equations were implemented in my code inside of the `update_activations()` function.

3.1 Decay Equation

When a node is excited and its activation rises, it must after a certain time eventually go back to its resting state. This is important so that the network stabilizes and does not stay in a constant state of activation. Decay also allows the network to reset after some time, so that new input and new probing can be done. The decay behavior can be described using the following activation update equation:

$$\text{decay}_i(t) = -D \cdot (a_i(t) - R) \tag{1}$$

where D is the decay parameter, $a_i(t)$ is the current activation at a specific time t, and R is the resting activation level.

The equation shows that without input in the network, the activation level of a node gradually goes down to resting activation. Also, if the activation is less than R, the decay will allow it to gradually rise the activation back to R. In other words, each node decays towards the resting value R at a rate of D. The decay equation was implemented within the `update_activations()` method in my `IACModel` class as follows:

```
decay = -self.decay_parameter * (self.activations[i] - self.resting_activation)
```

This follows the equation listed in McClelland's paper [1]. A simulation of the decay of a single node over 100 time steps can be seen in Figure 1 below.

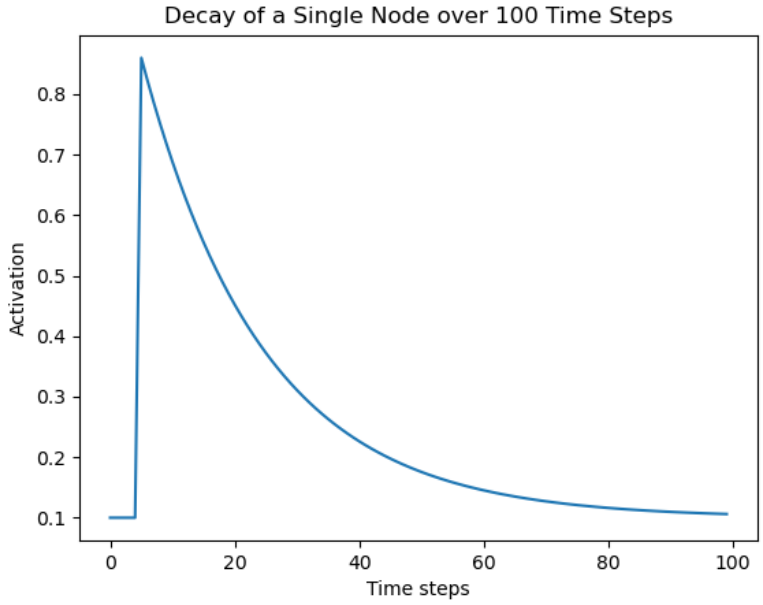

Figure 1: Activation decay of a node over 100 time steps after excitation. This plot shows the activation level of a single node over 100 time steps. Initially, the node has a resting activation of ($R = 0.1$). At time $t = 5$, the node is excited with an external input which increases activation to 0.9. Afterwards, the activation decays to the resting activation state according to the following rule: $\text{decay} = -D \cdot (A - R)$, where D is the decay parameter, A is the activation of the node at a specific time, and R is the resting activation value. This plot shows the activation decay process of a single node in the IAC Model.

3.2 Net Input Equation

Each node receives inputs from three different sources, as shown by the net input equation.

$$\text{input}_i(t) = p_i(t) + E \cdot \sum \text{excitors} - I \cdot \sum \text{inhibitors} \tag{2}$$

where $p_i(t)$ is the probe input to node i, and E and I are the respective weights for excitatory and inhibitory inputs. This equation shows us that to calculate the net input of each node, we must take the sum of the probe input (for example when probing a specific node) with the product of the excitatory weight and active connected excitors. From this sum, we subtract the sum of the inhibitory weight and the activation of the active connected inhibitors. This is how I implemented this equation in my model:

```
    for i in range (self.num_nodes):
            excitatory_input = np.dot(self.excitatory_matrix[i], self.activations)
            inhibitory_input = np.dot(self.inhibitory_matrix[i], self.activations)
            total_input = (external_input[i] + self.excitatory_weight*excitatory_input
                - self.inhibitory_weight*inhibitory_input)
```

Here, `external_input[i]` is the probe input, that is used to stimulate an external excitation when probing. By using `np.dot`, I took the sum of the weighted activations, as it is shown in the equation in McClelland (1981). This net input equation gives us the resulting signal arriving at a certain node from the network and the external input.

3.3 Input Effect Equation

The effect of input on a node depends on the sign of the input. In order to calculate the input effect equation, we use Equation 2. The equation for input effect is the following:

$$\text{effect}_i(t) = \begin{cases} (M - a_i(t)) \cdot \text{input}_i(t), & \text{if input} \geq 0 \\ (a_i(t) - m) \cdot \text{input}_i(t), & \text{if input} < 0 \end{cases} \tag{3}$$

where $M = 1.0$ is the maximum activation and $m = -0.2$ is the minimum activation. This equation allows us to scale the input value so that the effect depends on how close the activation is to its max or min value.

In my implementation, the logic was applied as follows:

```
if total_input >= 0:
                effect = (self.max_activation - self.activations[i]) * total_input
            else:
                effect = (self.activations[i]-self.min_activation) * total_input
```

3.4 Activation Update

The final step of the process consists of combining the effect equation (Equation 3) with the decay equation (Equation 1) in order to have the current activation. We compute the new activation by adding the input effect and the decay to the current activation. This can be summarized in the equation below:

$$a_i(t+1) = a_i(t) + \text{effect}_i(t) + \text{decay}_i(t) \tag{4}$$

This equation was implemented with the following code:

```
new_activations[i] = self.activations[i] + effect + decay
new_activations[i] = np.clip(new_activations[i], self.min_activation, self.max_activation)
```

After calculating the new activation at time $t + 1$, we must clip the value so that they stay within the predefined active range of $[-0.2, 1.0]$. This active range ensures that the activation values remain within the bounds, so that it does not become unrealistically excited or inhibited.

These four equations define together the behavior of the IAC model. We can see that the activation level is based on decay, inhibition and excitation, as well as external input. This implementation of the model will then allow us to probe different nodes in order to retrieve knowledge in the network as will be explained in the following section.

4 Probing the Network and Retrieving Information

To probe a node, we need to apply an external input so that a node becomes active. This activation of a node then spreads to the connected nodes through the excitatory links defined in the database. Nodes within the same cohorts will inhibit each other. Eventually, the activation levels of each nodes will show which of them are most strongly liked to the probed node. To test the model, I probed the network by giving an external input at different nodes. This allowed me to see how the external input propagates through the network and to see which nodes are activated as a result.

4.1 Probing the 'Jets' Node

First, I probed the 'Jets' node by applying an external input on its node. The simulation was run for 500 time steps to allow the network to stabilize. As shown in Figure 2, the nodes that have the highest activations as a result are "Jets", "20s", "JH", "Single" and "Bookie". This suggests some of the most common traits shared by members of the Jets gang.

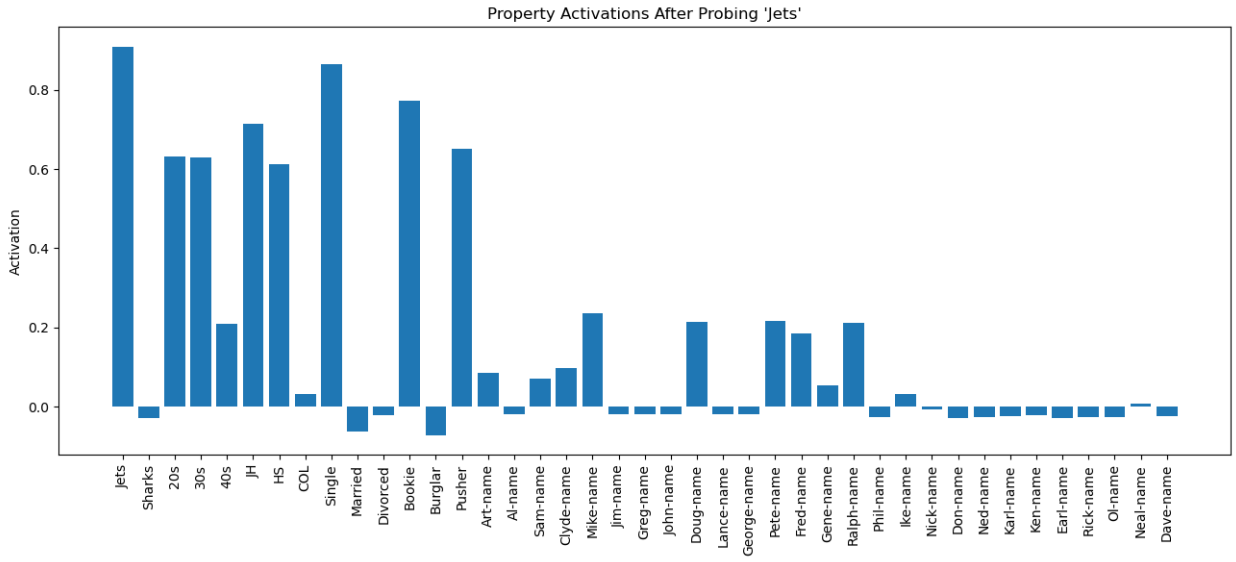

Figure 2: Activation of property nodes after probing the `Jets` node. Property nodes that represent common attributes of individuals in Jets show higher activation. These include, "Single", "20s" and "Bookie". This figures shows that it is possible to probe a network on a node in order to get knowledge about the instances.

4.2 Probing the 'George-name' Node

In my second experiment, I probed the network over 500 time steps with the property node "George-name" to test the activations of the network when probing for a specific individual. When a node that represents a person's name is probed, it should activate nodes that correspond to the properties that this individual possesses. When I probed George-name, the network retrieved the traits that describe him. The activation plot, Figure 3 shows his attributes quite clearly. We can deduce from the plot that George is likely to be a member of the "Jets", in his 20s, has a junior high education level, is single, and working as a bookie. These traits have high activation levels because of the excitatory connections between "George-name" and other attribute nodes, but also because of the inhibitory groups that suppress other nodes that are competing in the same category.

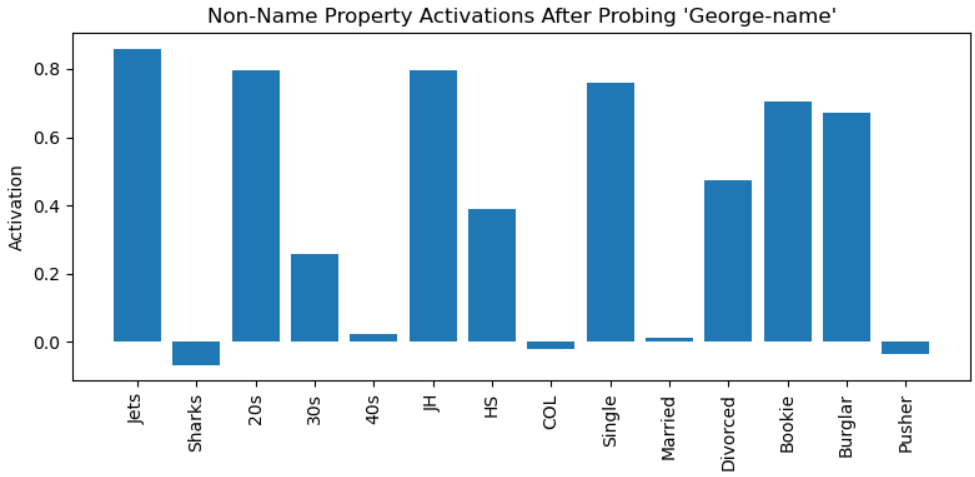

Figure 3: Activation of non-name property nodes after probing `George-name`. The network retrieves specific knowledge about George, including his gang affiliation, age, education level, marital status, and occupation. We can deduce that he is a member of the Jets, in his 20s, has a junior high education, is single, and works as a bookie.

4.3 Probing the '20s' and 'JH' Nodes

Finally, I probed the network with both the "20s" and "JH" nodes to see how the model retrieves individuals who share two different traits. We can see in Figure 4 that the instance nodes that correspond to individuals in their 20s with junior high education could be Jim, John, Lance and George, as all of them have very high activation levels. This demonstrates that we can also identify individuals in the network by probing for two different traits. The IAC model is very flexible in the way it retrieves specific nodes, as we are not only able to probe individuals or characteristics, but also a combination of them.

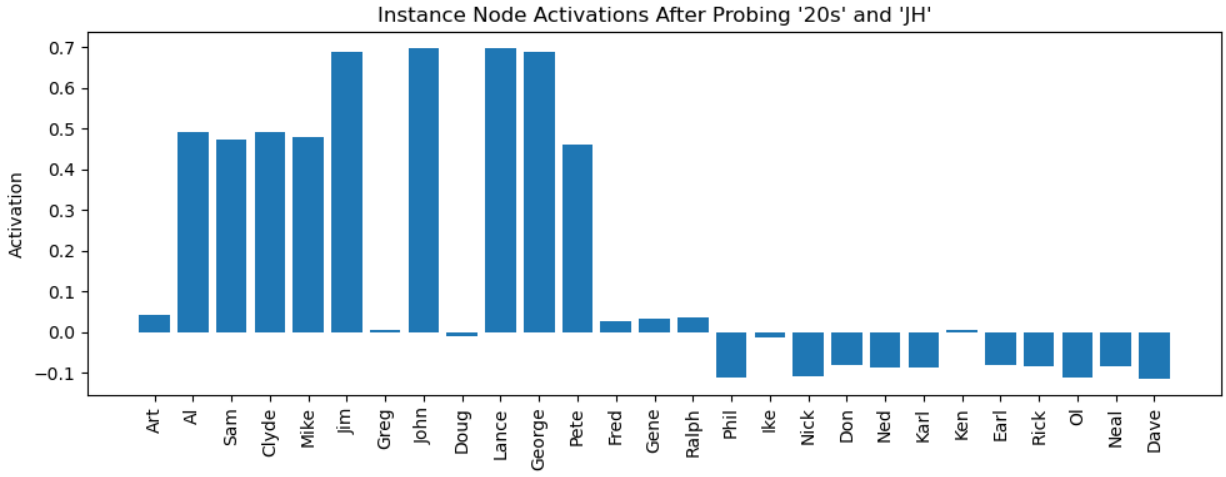

Figure 4: Activation of instance nodes after probing "20s" and "JH". Individuals who share both properties show increased activation.

4.4 Comparison with McClelland (1981)

To validate the results of my model, I compared my activation levels for my first and third experiments with the results obtained by McClelland.

The comparison of my results and McClelland's results when probing "Jets" can be seen in Figure 5. We can see that most of the nodes have similar activation levels, and follow a similar pattern. However,

the "Burglar" node is the one that shows a significant difference, as my IAC model calculated a negative activation while McClelland's model calculated a positive value for activation.

The difference in activations between the IAC model and McClelland's model could be due to a variety of reasons. McClelland (1981) ran the simulations until the activations stabilized, but it is not always clear how many time steps were used for his experiments. In our implementation, we have constantly used 500 time steps. Another possible reason for the difference could be the way that the values are clipped in the range $[-0.2, 1.0]$. In the IAC model, this was implemented with a NumPy function.

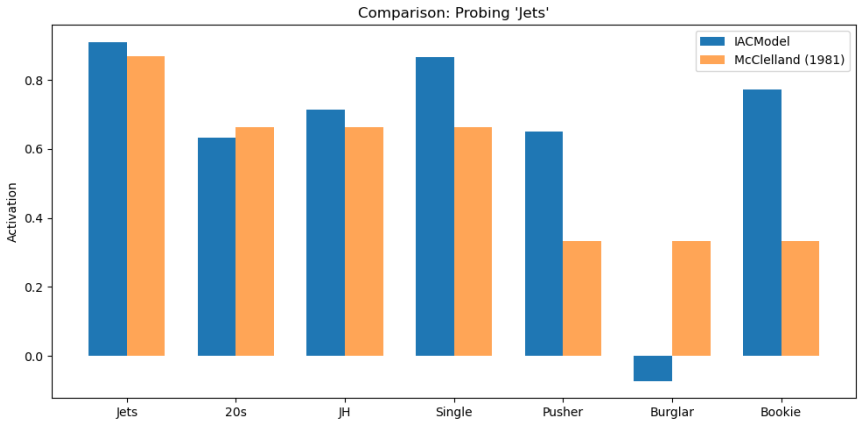

Figure 5: Activation comparison for selected property nodes after probing "Jets", comparing results form the IAC model and McClelland (1981). Apart from the node "Burglar", the general activation levels of each of these selected nodes match well. The nodes that had a relatively stronger activation in McClelland, also showed high activation in the IACModel, and those with relatively lower activation also showed similar behavior.

In the second experiment, we can also see a similar pattern in the activations, with a notable difference. The instance nodes of individuals showed a much higher activation level in the IAC model than in McClelland's experiments. Similarly, the reasons for this discrepancy could be multiple.

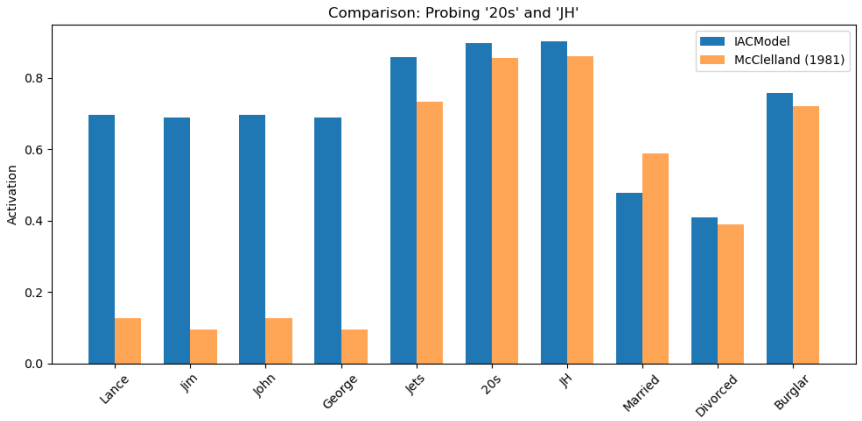

Figure 6: Activation comparison for selected nodes after probing both "20s" and "JH". The IACModel matches the activation levels for all categories, except for the instance name nodes. The nodes with the highest levels of activations match in both cases, but it must be noted that the exact values are different.

In conclusion, while there are some differences between the values computed by my IAC model and McClelland's values, we can conclude that the overall pattern is similar. It suggests that the implementation of the IAC model is representative of the key mechanisms explained by McClelland (1981).

5 Evaluation

This assignment was overall relatively challenging, but it was rewarding. Despite being short, fully comprehending McClelland's paper [1] proved to be conceptually challenging at the beginning. It was however very rewarding once I understood and got a deeper insight into the IAC model.

Implementing the data preprocessing in code was relatively straightforward, though it required careful attention to ensure the matrices were properly sized and aligned. While the equations in the paper were relatively straightforward, understanding how they relate to each other, and how to implement them into code was also another challenge of this assignment. A minor mistake at the beginning of implementation in the sign of the inhibitory matrix proved to have great consequences on the overall activation levels. A mistake made in calculation caused the inhibitory values to cancel out the excitation values. Once this problem was discovered, I was able to generate much more coherent activation values when probing.

6 Conclusion

This assignment showed how we can apply the Interactive Activation and Competition (IAC) model proposed by McClelland. It showed how a simple network can store and retrieve knowledge [1]. We have seen that the model relies on the excitatory links between different nodes, and how we can compute activation levels through excitation, inhibition, and decay. We were able to retrieve knowledge from this model by probing not only individuals, but also by probing their attributes, and observed how the activation of individuals increased when they shared the probed attribute or attributes. Even though my specific implementation of McClelland's model did not exactly match his results, we were able to observe a very similar pattern. Probing names such as "George" activated the correct traits and probing for specific traits such as "Jets" also activated the correct attributes. In conclusion, this assignment deepened my understanding of connectionism and how this framework is foundational for more complex neural networks. I was also able to get more insight on the basic building blocks of such models, and how they are governed by relatively simple mathematics.

References

[1] J. L. McClelland, "Retrieving general and specific information from stored knowledge of specifics," *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 3, 1981.