Computation & Cognition: Assignment 4
Connectionism

Hao Chen s3990788

April 2025

1 Getting Started

Understanding how humans represent and reason about knowledge is an important area of research in cognitive science. An influential approach to this problem is connectionism, which differs from the idea of the traditional symbolic model, which assumes that knowledge is stored in explicit rules or symbols that are manipulated by logical operations. Connectionism argues that cognition is not generated on the basis of explicit rules but arises from the interactions between simple units in a structure similar to a neural network. In the Interactive Activation and Competition (IAC) model proposed by McClelland, either a person, a category, or an attribute is represented by a node in a network, and knowledge arises from patterns of connections between these nodes. These connections can be either excitatory, where the activation of one node promotes the activation of a related concept, or inhibitory, where the activation of one concept inhibits competing nodes. The whole network evolves dynamically under the constant propagation of activation and inhibition, eventually reaching a stable state, which is the system's 'interpretation' or 'decision' of the inputs. This mechanism provides a neurally plausible model for human reasoning, classification, and memory retrieval processes. In this report, we construct and implement a simplified IAC model based on a dataset of person instances and their associated attributes (e.g., gang, age, education level, marital status, occupation, etc.). By probing specific nodes in the network, we analyze how activation spreads throughout the system and how the activation values of other nodes (especially attribute nodes) evolve over time and eventually converge to a steady state, thus investigating possible mechanisms for knowledge representation and inference processes in connectionist models.

2 Network Structure

2.1 Data structure

The dataset used in this assignment is the same as described in the original paper. It contains 27 gang members, each associated with six types of attributes:

- Name (27 unique names ending with "-name")
- Gang affiliation (2 gangs)
- Age group (3 brackets)
- Education level (3 levels)
- Marital status (3 statuses)
- Occupation (3 occupations)

Each node in the network represents either an instance (i.e., a gang member) or a property (i.e., an attribute), and nodes are connected via either excitatory or inhibitory links.

The dataset was provided as a CSV file containing a 68×68 binary matrix. Each row and column represents a node in the network, and the entries indicate the presence of excitatory connections between nodes. Specifically, a value of 1 denotes an excitatory connection. We stored the matrix as a **pandas.DataFrame**, which offers a convenient and readable tabular structure. This allowed us to easily access nodes by name, extract specific subsets (e.g., instance nodes or property nodes), and eventually convert the data into a NumPy array for efficient matrix operations in the model.

2.2 Inhibitory connections

To represent inhibitory connections, we first used get_all_nodes_groups(df) method to extracted all mutually exclusive attribute groups from the dataset. For example, gang affiliation ("Jets", "Sharks"), age group ("20s", "30s", "40s"). For each instance in the dataset, we checked which attributes were present (i.e., had a value of 1). Then, for each attribute the instance possessed, we using the set_inhibitory_connections(df) method to identified all other mutually exclusive attributes in the same group and inserted inhibitory connections (with value -1) between the instance and those incompatible attributes. This effectively ensures that, for example, if "Art" is a member of "Jets", then "Art" will be connected with -1 to "Sharks". The process is symmetric: "Sharks" is also set to -1 with respect to "Art".

Finally, for each attribute group, we also inserted symmetric inhibitory links between all pairs of mutually exclusive attribute nodes. In other words, nodes such as "20s" and "30s" inhibit each other directly, regardless of which instance they belong to. All inhibitory connections were added directly to the connection matrix by replacing 0s, while existing excitatory connections (value 1) were preserved. This process results in a single unified matrix encoding both excitatory and inhibitory interactions.

2.3 The _init_() method of the IAC model

In the **__init__()** method of the model class, we defined several key parameters that control the dynamics of the network. These include:

- **D** – the decay rate, which controls how quickly activation values return toward the resting level.
- **R** – the resting activation level for each node.
- **E** and **I** – coefficients for excitatory and inhibitory input, respectively.
- **M** and **m** – the maximum and minimum allowed activation values.

Within the initialization process, we store the dataset matrix as a DataFrame and extract the full list of node names. We then compute the total number of nodes, convert the matrix into a NumPy array with both excitatory and inhibitory connections (via the **set_inhibitory_connections()** function), and initialize the activation vector to zero. Additionally, we create a name-to-index mapping to efficiently locate nodes during activation updates.

3 Dynamics of the IAC Model

The dynamics of the IAC model refer to how the states of its nodes change over time. In the IAC model, each node has an activation value that evolves across timesteps. By observing and comparing these activation levels, we can determine whether a node has been stimulated by input or how strongly it is activated relative to others in the network.

A higher activation value indicates that the node is currently receiving stronger support—either through direct external input (such as a probe) or via activation spreading from connected nodes. If a node's activation is significantly higher than others in the same attribute group (e.g., "Jets" vs. "Sharks"), it suggests that the model has selected this node as the most compatible or likely candidate based on the current input. This behavior reflects a central feature of IAC models: competitive selection and constraint satisfaction through a balance of excitation and inhibition.

A node's activation is influenced by multiple factors: external input, excitatory and inhibitory input from connected nodes, and a natural decay toward a resting activation level. These components together govern the flow of activation through the network, enabling it to settle into a stable configuration that reflects both the structure of the input and the internal relationships among nodes. In the following sections, we describe in detail how each of these factors contributes to the change in activation values over time.

3.1 Net input

The IAC model computes the *net input* to each node as the sum of three components: external input, excitatory input, and inhibitory input. This is formalized in the following equation:

$$\text{input}_i = \text{probe}_i + E \cdot \sum_j e_{ij}^i - I \cdot \sum_j i_{ij}^i \tag{1}$$

- $p_i(t)$ represents the external input to node i at time t, such as a probe stimulus. In our implementation, external input is only provided at $t = 0$ in the form of (**{"Jets": 0.2}**).
- $E \sum_j e_{ij}(t)$ is the total excitatory input received from other nodes, scaled by the excitation strength E. In our code, this is implemented by identifying all nodes that are excitatorily connected to node i—that is, all positions with value 1 in the connection matrix. The activations of those connected nodes are summed and then multiplied by E before being added to the net input.
- $I \sum_j i_{ij}(t)$ is the total inhibitory input from other nodes, scaled by the inhibition strength I.We similarly identify all positions with value -1 in the connection matrix to find nodes that inhibit node i. The activations of those nodes are summed, multiplied by I, and subtracted from the net input. This subtraction reflects the suppressive nature of inhibitory connections, which act to reduce the activation of a node.

By combining these three components, the IAC model calculates the overall degree of excitation or inhibition that a node experiences at each moment, referred to as the **net input**.

3.2 Effect

Once the net input to a node is computed, the model calculates its effect on the node's activation. The direction and strength of this effect depend on whether the net input is excitatory (positive) or inhibitory (negative).

$$\text{effect}_i(t) = \begin{cases} (M - a_i(t)) \cdot \text{net\_input}_i(t), & \text{if net\_input}_i \geq 0 \\ (a_i(t) - m) \cdot \text{net\_input}_i(t), & \text{if net\_input}_i < 0 \end{cases} \tag{2}$$

When the net input $input_i(t)$ is greater than or equal to zero, the node is considered to be in an excitatory state. In this case, its activation increases according to the difference between the current activation and the upper bound M. This ensures that as the activation level gets closer to the maximum value, the rate of increase slows down, preventing overshooting.

When the net input $input_i(t)$ is less than zero, the node is inhibitory. The activation value then decreases based on how close it is to the lower bound m.

This model not only distinguishes between excitatory and inhibitory influences based on the sign of the net input but also keeps each node's activation bounded within a fixed range $[m, M]$. In our implementation, we use $M = 1.0$ and $m = -0.2$, consistent with the original IAC model proposed by McClelland.

3.3 Decay

After computing the effect of the net input on each node, the model also applies a decay mechanism. This reflects the tendency of each node's activation to gradually return to a resting level R when it is no longer receiving input. The purpose of this decay is to prevent a node from remaining highly activated indefinitely without ongoing support, ensuring the network remains dynamically stable.

The update rule incorporating decay is defined as:

$$\frac{da_i(t)}{dt} = \text{effect}_i(t) - D \cdot (a_i(t) - R) \tag{3}$$

- $a_i(t)$ is the current activation of node i at time t.
- $\text{effect}_i(t)$ is the contribution from the net input, computed as described in the previous section.
- D is the decay rate, and R is the resting activation level.

The term $D \cdot (a_i(t) - R)$ calculates how far the node's activation deviates from its resting level. This deviation is then subtracted from the effect, which prevents the activation from growing without bound and helps guide it back to a natural baseline. The closer the activation is to R, the smaller the decay effect becomes.

In our implementation, we set $D = 0.05$ and $R = 0.1$, meaning that each node will slowly return to this resting state.

This decay mechanism is crucial for stability: without it, nodes that are once activated might remain so permanently, preventing the network from adapting to new inputs. With decay, the model can more effectively respond to changing information and eventually converge to a stable and interpretable state.

3.3.1 A Simple Demonstration

To better show how decay works, I wrote a code where the activation is is equals to R (0.1) at t=0, and at t=5 we set the activation to 0.9 to simulate receiving a stimulus from the outside or from another node and then show how that node slowly falls back down from 0.9 to resting level.

We ran this simulation for 100 timesteps. The resulting curve (Figure 1) clearly shows the expected exponential decay behavior as the activation returns to R.

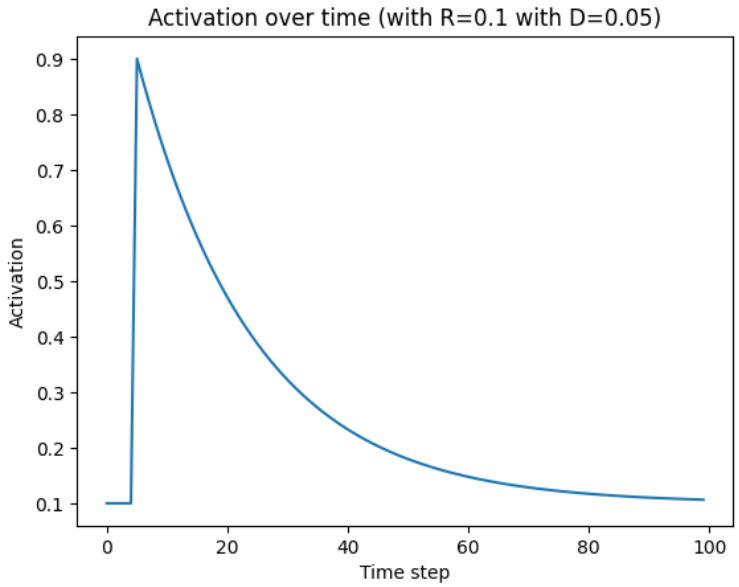

Figure 1: Decay of a node's activation from 0.9 toward resting level $R = 0.1$

4 Implementation of the Equations

We implemented all four core equations of the IAC model in the **update_activations()** method of our class. Below, we describe how each part of the computation corresponds to a specific equation in the model.

- **External input:** We initialize a vector **p** of zeros and insert non-zero values for the nodes that receive external stimulation (probe). This corresponds to $p_i(t)$ (1). In our setup, this input is only applied at $t = 0$.
- **Excitatory and inhibitory input:** To compute excitation and inhibition, we create boolean masks from the connection matrix:
  - exc_mask = (self.connections == 1).astype(float)
  - exc_input = self.E * (exc_mask @ self.activations)
  - inh_mask = (self.connections == -1).astype(float)
  - inh_input = self.I * (inh_mask @ self.activations)

  1 represents excitatory connections, and **-1** represents inhibitory connections. We then use matrix multiplication to sum the activation values of connected nodes, corresponding to the two summation terms in equation (1).
- **Effect of net input:** Using **np.where()**, we apply the effect(2) based on whether the net input is positive or negative.
- **Decay and final update:** We subtract the decay term $D(a_i(t) - R)$ from the computed effect and use it to update the activations. This implements Equation (3)

Overall, this method faithfully implements all the update rules described in McClelland's 1981 IAC model.

5 Simulations and Results

In all experiments, we applied an external input of 0.2 to the probe nodes. This value is consistent with the original IAC model settings described by McClelland (1981), ensuring comparability of our results.

5.1 Probe: "Jets"

From figure 2, after probing the node **"Jets"**, its activation rises to approximately 0.92, indicating a strong external input response. The mutually inhibitory node "Sharks" is suppressed to nearly the lower bound $m = -0.2$, confirming that the inhibitory group mechanism is functioning correctly. Several name property nodes associated with Jets members (e.g., "Art-name", "Sam-name") are also activated, with activation levels around 0.15. This illustrates the model's spreading activation behavior, in which excitatory connections from the probe activate linked instance nodes. In contrast, name nodes of "Sharks" members show negative activation, demonstrating competition between groups.

In addition, several property nodes that frequently co-occur with Jets members—such as **"20s"**, **"JH"**, and **"Single"**—also become highly activated (approximately 0.8). Although the exact values differ from those reported in McClelland's original results, our model successfully identifies the most relevant attributes associated with the probe, indicating that the core mechanisms of spreading and selection are working as intended.

Interestingly, in McClelland's paper, the three nodes in the **Occupation** group received equal activation values. In our simulation, however, **"Bookie"** reaches about 0.08 while **"Burglar"** and **"Pusher"** are suppressed to around -0.03. This suggests that our model selects **"Bookie"** as the most compatible occupation and suppresses the others. The cause of this discrepancy remains unclear and may relate to differences in the dataset or network connectivity.

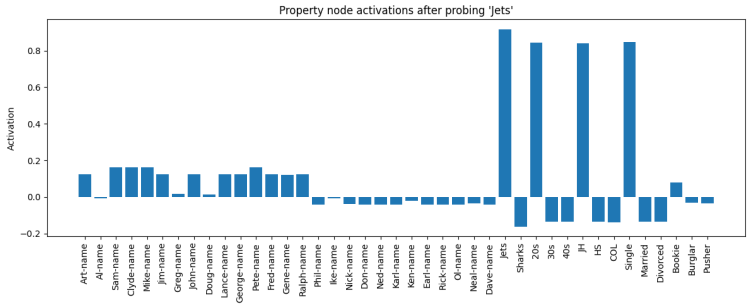

Figure 2: Property node activations after probing 'Jets' after 500 times

5.2 Probe: "George-name"

Since this probe is not included in the original McClelland report, we compare the results to the ground-truth data from the dataset. According to the database, **"George"** is connected to the following properties: "Jets", "20s", "JH", **"Divorced"**, and **"Burglar"**.

In the resulting plot 3, we observe that the nodes "Jets", "20s", "JH", **"Single"**, and **"Bookie"** are activated, while most other non-name property nodes are suppressed. This means that the properties **"Divorced"** and **"Burglar"**, which are supposed to be part of George's profile, have been incorrectly suppressed.

One possible explanation is the competitive dynamics within mutually exclusive attribute groups. For example, in the **marital** group, although **"Divorced"** is George's correct status, the node **"Single"** might have received stronger indirect support from other active nodes (e.g., from **"Jets"** members who are mostly single). Due to the mutual inhibition within this group, this can lead to **"Divorced"** being suppressed even when it should be active.

A similar explanation applies to the **occupation** group, where **"Bookie"** may have received stronger activation via indirect pathways, such as shared co-occurrence patterns among other active instance nodes. Since our network is highly interconnected and uses global spreading and inhibition, even correct nodes can lose in local competition if competing nodes are indirectly favored.

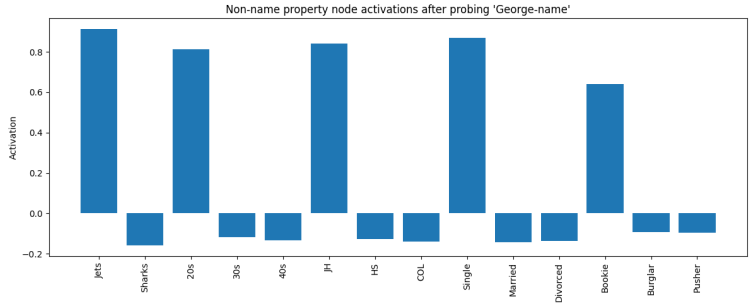

Figure 3: Non-name property node activations after probing 'George-name' after 500 times

5.3 Probe: "20s" and "JH"

In this simulation, we probed the network with the attribute nodes **"20s"** and **"JH"**, corresponding to age and education level, respectively. According to the dataset of the paper, these two properties should activate four specific instances—**Jim**, **John**, **Lanceand George**—since they are the only ones in the dataset who possess both of these attributes ("20s" and "JH").

Our results are largely consistent with the findings in the paper. The instance nodes **Jim**, **John**, **Lanceand George** show significantly higher activation values (all above 0.6) 4, indicating that the model correctly identified individuals possessing both probed properties. This demonstrates that the network is capable of combining two cues—age and education level—and retrieving the relevant individuals who share these attributes.

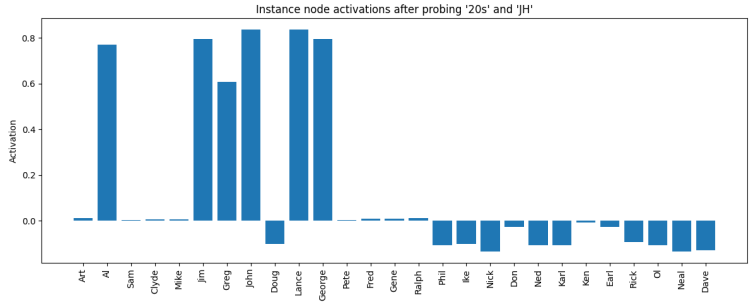

Figure 4: Instance node activations after probing '20s' and 'JH' after 500 times

5.4 Summary

Overall, our results show that the IAC model is capable of performing integrative inference by combining multiple input cues and dynamically balancing excitatory and inhibitory influences. The network successfully activates relevant nodes and suppresses incompatible alternatives. The observed results closely align with those reported in McClelland's original work, with only minor discrepancies caused by occasional over-activation or over-inhibition in certain nodes.

6 Evaluation

In the early stages of implementation, distinguishing between different types of property nodes was relatively straightforward. By identifying all node names ending with "`-name`", we were able to separate instance nodes from name property nodes, and then identify the remaining non-name property nodes. Furthermore, mutually exclusive property nodes (e.g., "Jets" and "Sharks") could be grouped into inhibitory sets, since only one node in each group is active at a time while others remain inactive. This enabled us to identify inhibitory connections in a systematic way.

However, designing and implementing the correct inhibitory connections such that the model updates activation values accurately under both spreading and suppression mechanisms turned out to be a major challenge. As shown in Figure 4, for example, the node `Al-name` receives an activation value similar to nodes

that actually match the input probe, despite not having the probed attribute "20s". This suggests that inhibition was not fully effective in limiting activation to only the most relevant instances.

Additionally, implementing net input updates required several iterations. Initially, our connection matrix only contained binary values (1 for excitation, 0 for no connection). We attempted to calculate inhibitory input separately using predefined inhibitory groups, but this approach proved difficult to maintain and verify. As a solution, we modified the connection matrix to include -1 entries to represent inhibitory links directly. This allowed us to compute the total net input—including both excitation and inhibition—uniformly from the connection matrix, simplifying the update mechanism and improving interpretability.