The **core idea** of a Hypergraph Graph Neural Network (HGNN) is to extend graph neural networks (GNNs) to **hypergraphs**, where each hyperedge can connect multiple nodes. The core formulas for HGNN are derived from how node features are propagated across a hypergraph structure.

Core Idea Formulas

Let:

- V be the set of nodes.
- E be the set of hyperedges, where each hyperedge connects a subset of nodes, $e \subseteq V$.
- $H \in \mathbb{R}^{|V| \times |E|}$ be the incidence matrix of the hypergraph, where $H_{v,e} = 1$ if node v belongs to hyperedge e, otherwise $H_{v,e} = 0$.
- $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ be the node feature matrix, where d is the feature dimension of each node.

The key operation in HGNN is **feature propagation** through the hyperedges, which can be described as:

1. **Node-Hyperedge Projection:**
   The node features are projected to the hyperedges:

   $$\mathcal{Z}_e = H^T \mathbf{X}$$

   where $\mathbf{Z}_e \in \mathbb{R}^{|E| \times d}$ represents the feature embeddings of the hyperedges.

2. **Hyperedge-Nodes Aggregation:**
   The information from hyperedges is aggregated back to the nodes:

   $$\mathbf{Z}_v = H \mathbf{Z}_e = H H^T \mathbf{X}$$

   where $\mathbf{Z}_v \in \mathbb{R}^{|V| \times d}$ represents the updated node features.

3. **Normalization:**
   To avoid biased aggregation from high-degree nodes or hyperedges, we apply a normalization factor:

   $$\mathbf{H}_{\text{norm}} = D_v^{-1/2} H D_e^{-1} H^T D_v^{-1/2}$$

   where D_v and D_e are the diagonal degree matrices of the nodes and hyperedges, respectively. The degree of node v is the number of hyperedges it belongs to, and the degree of hyperedge e is the number of nodes it connects.

4. **Update Rule:**
   The final node update can be written as:

   $$\mathbf{X}^{(k+1)} = \sigma(\mathbf{H}_{\text{norm}} \mathbf{X}^{(k)} \mathbf{W}^{(k)})$$

   where $\mathbf{X}^{(k)}$ represents the node features at layer k, $\mathbf{W}^{(k)}$ is a learnable weight matrix, and σ is a non-linear activation function (e.g., ReLU).

Example

Consider a simple hypergraph with:

- 3 nodes $V = \{v_1, v_2, v_3\}$
- 2 hyperedges $E = \{e_1, e_2\}$, where $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$

The incidence matrix H is:

$$H = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Each row represents a node, and each column represents a hyperedge.

Let the initial node feature matrix $\mathbf{X}$ be:

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

1. Project node features to hyperedges:

$$\mathbf{Z}_e = H^T \mathbf{X} = \begin{pmatrix} x_1 + x_2 \\ x_2 + x_3 \end{pmatrix}$$

2. Aggregate back to nodes:

$$\mathbf{Z}_v = H \mathbf{Z}_e = \begin{pmatrix} x_1 + x_2 \\ x_1 + 2x_2 + x_3 \\ x_2 + x_3 \end{pmatrix}$$

3. Apply normalization (if needed) and update node features using the update rule.

This process continues across multiple layers, updating the node representations at each step.