# Graph Neural Networks (GNN)

## • Graph Data

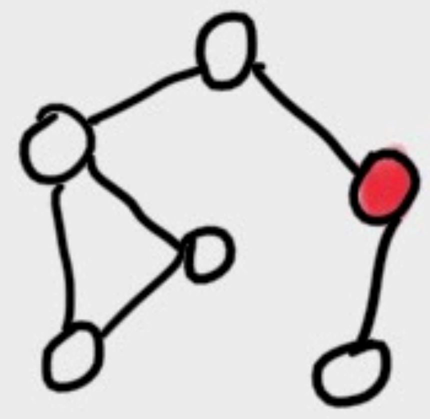
$G = (V, E)$

- Can be represented as an adjacency matrix
- Nodes & edges can have "features".

| | V1 | V2 | |
|---|---|---|---|
| V1 | 0 | 1 | |
| V2 | 1 | 0 | ... |
| | ⋮ | | |

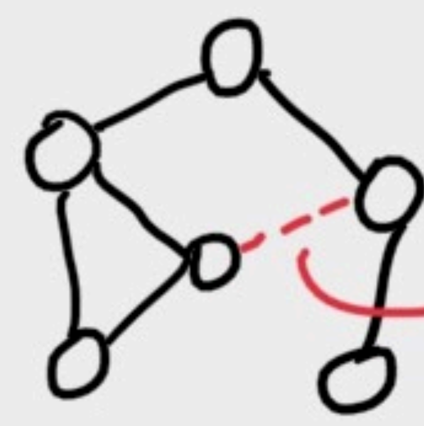
## • Common Graph Problems

- Node-level prediction:



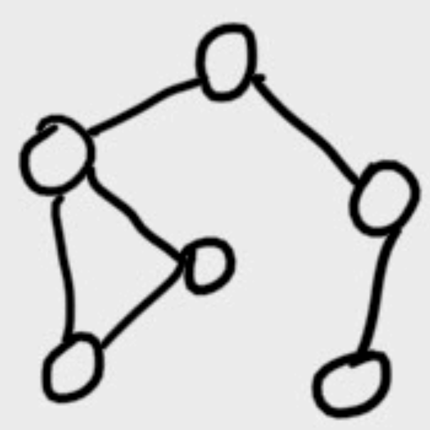
is this person single?
(unlabled node)

- Edge-level prediction
  (link prediction)



which product to buy next?

- Graph-level prediction



is this molecule a suitable drug?

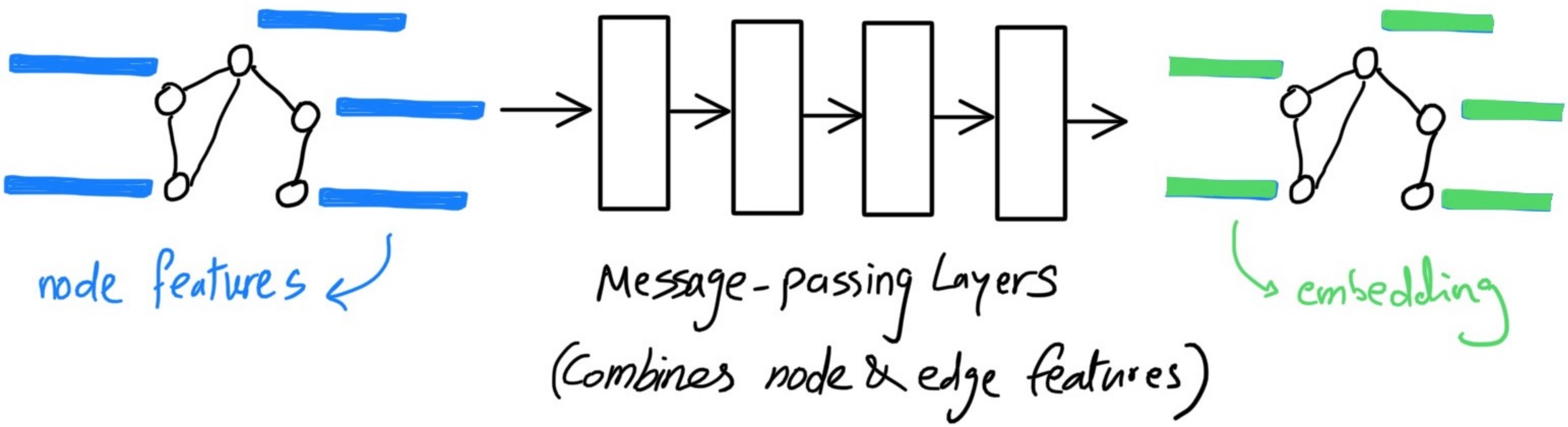
## • Challenges of Graph Data

- size and shape of a graph might change within dataset.
- isomorphism — flipping a graph produces the same one.
- Grid Structure — they are in a non-eucledean space.

- **Fundamental idea of GNN:**

  learning a suitable representation of graph data (Representation learning)

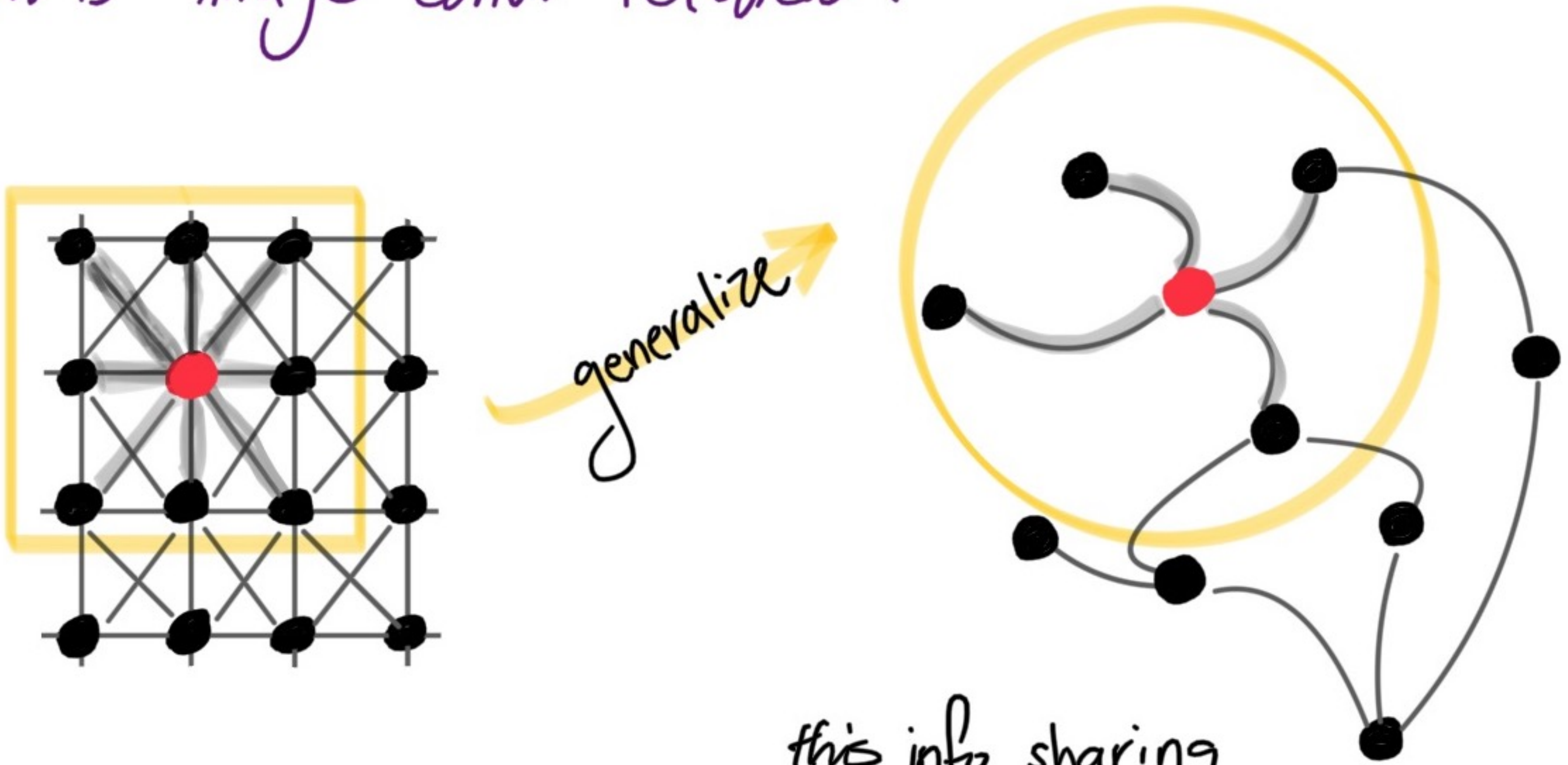
- **Message-passing Layers:**




gathering the current node info of neighbors → Combine them → update the node feature weights

A.K.A Graph Convolution

- How is image conv. related?

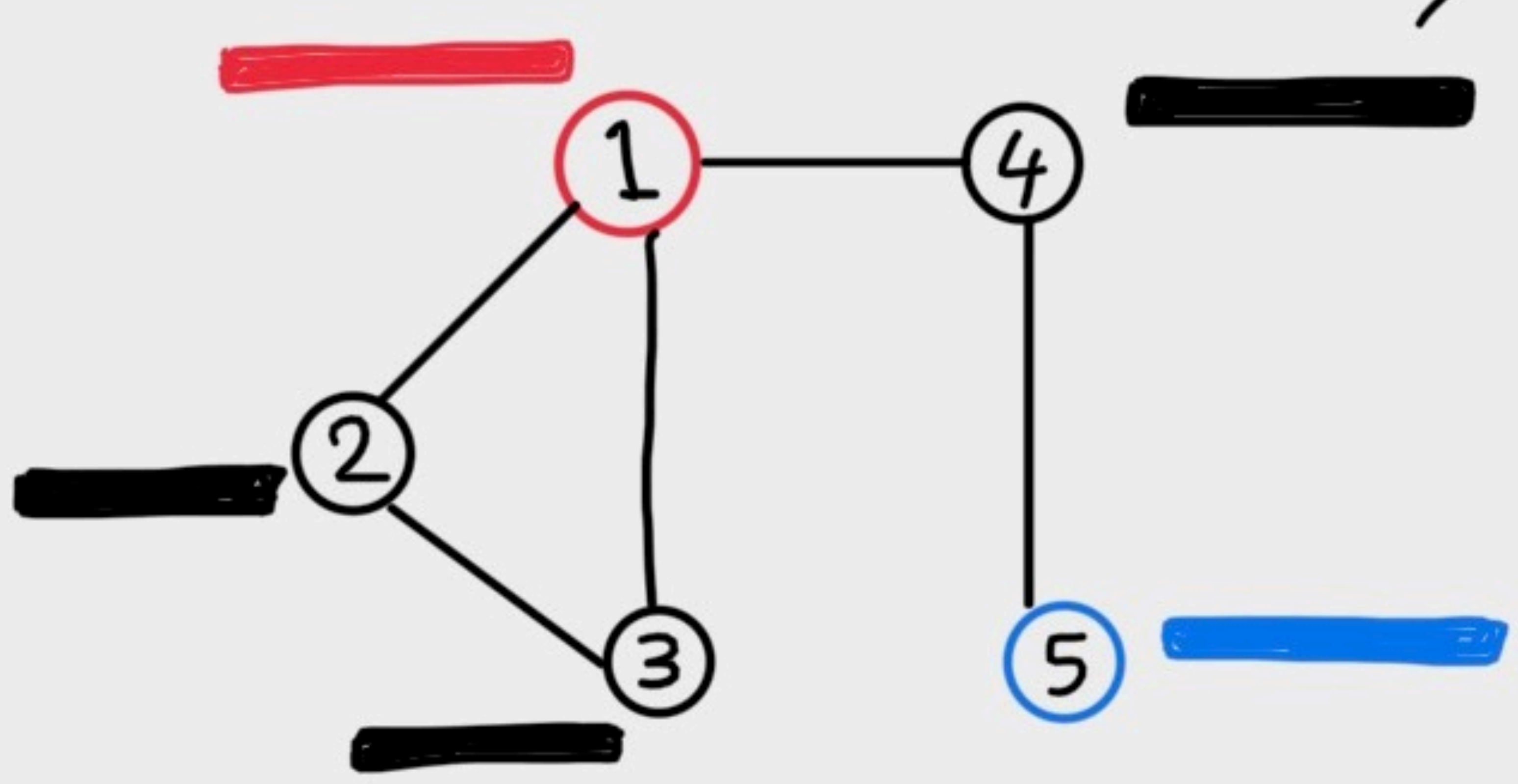



this info sharing

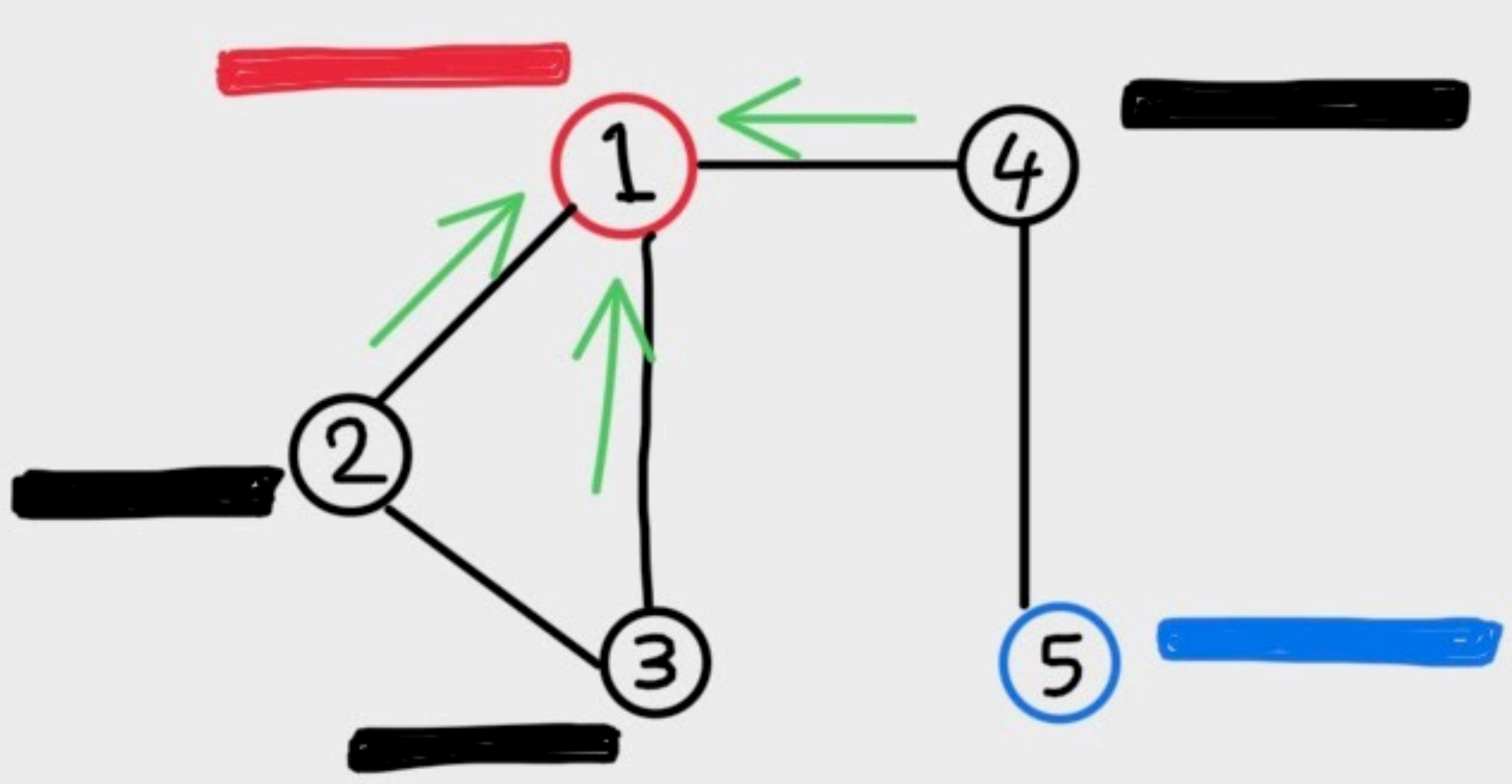
is what we call message passing.

# Simple Message Passing Example

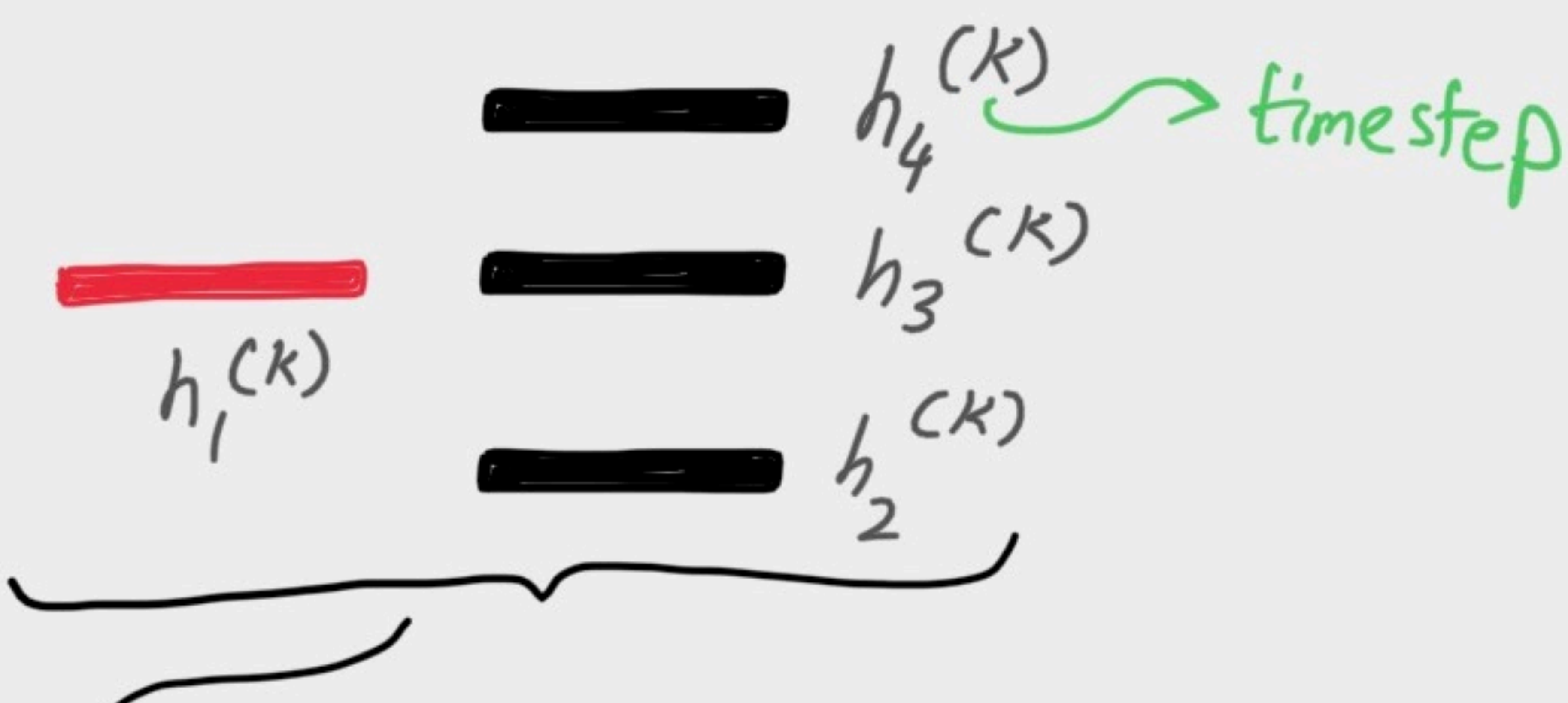
a.k.a node state




Our focus in on node 1.

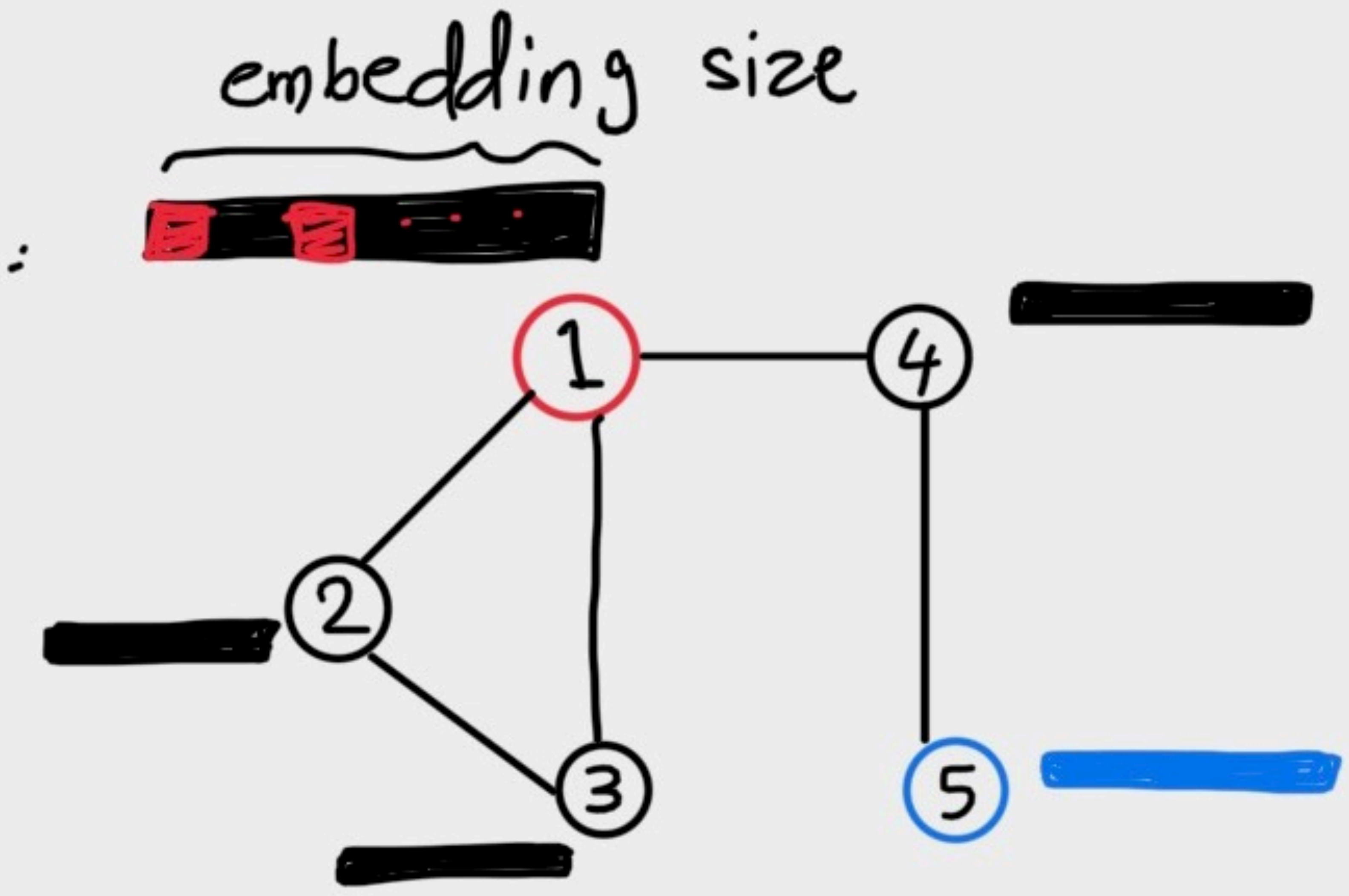



gather state from neighbors & self.



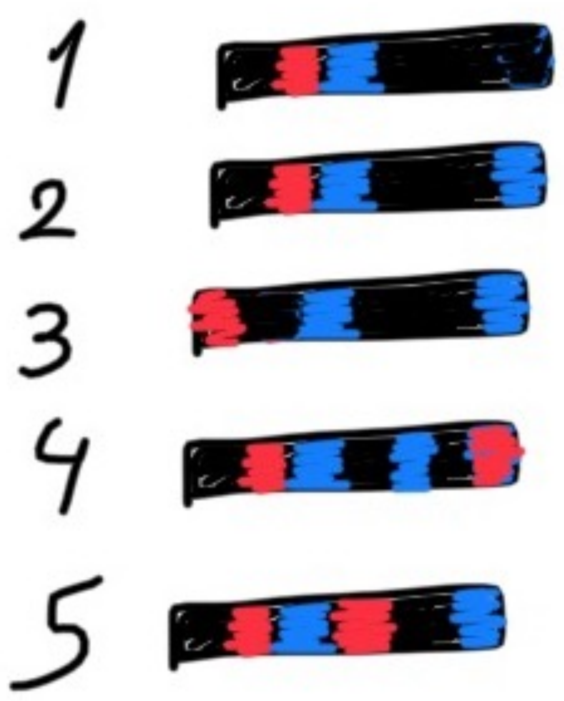

Then, we aggregate the states to get an updated state for node 1.


The updated graph:




We then do the same for other nodes too.

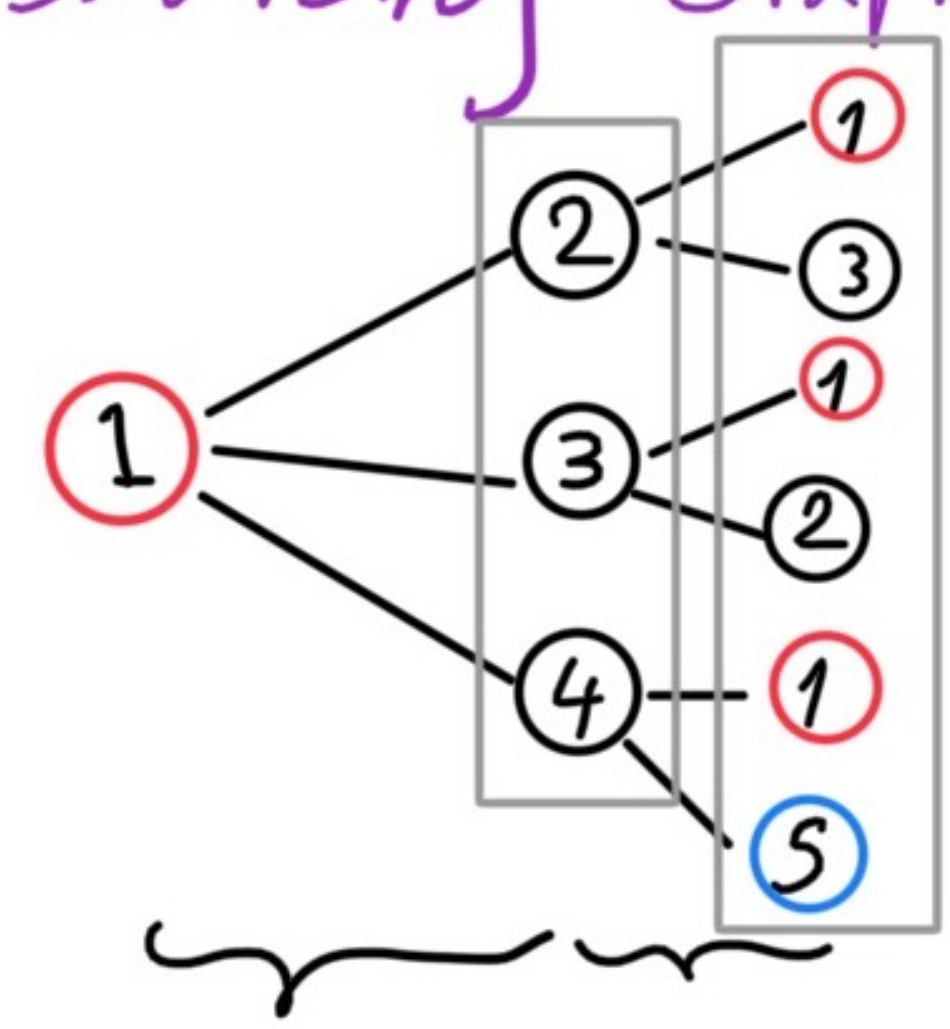
* after n message passing steps, all the nodes know some info about other nodes.




structural and feature-based information → we use this to make predictions

* This local feature aggregation can be compared to learnable CNN kernels.

## Visualizing Graph computation:




two neighborhood hops to let 1 know about 5.

this is actually the number of message-passing layers.

## Question: what happens if we do too many message passings?

oversmoothing. → eventually makes all the node states too similar / indistinguishable.

## ⋆ Math

$$h_u^{(k+1)} = \text{UPDATE}^{\overset{\text{timestep}}{(k)}}\left(h_u^{(k)}, \text{AGGREGATE}\left(\underbrace{\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}}_{\text{direct neighbors of node } u}\right)\right)$$

UPDATE
- mean
- max
- neural network
- recurrent neural network

AGGREGATE
- mean
- max
- normalized sum
- neural network

they must be differentiable.

## ⋆ Variations of GNN

self-loop

① GCN $\quad h_v^{(k)} = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$

sum of normalized neighbor embeddings

## ② MLP as AGGREGATOR
(from DeepSet paper)

aggregated message

$$m_{\mathcal{N}(u)} = \underline{\text{MLP}_\theta}\left(\sum_{v \in \mathcal{N}(u)} \text{MLP}_\phi(h_v)\right)$$

send states through a MLP

So, there are learnable weights for finding best way to aggregate the neighbors.

## ③ Graph Attention Networks

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} h_v$$

attention weights

$$\alpha_{u,v} = \frac{\exp(a^T [W h_u \oplus W h_v])}{\sum_{v' \in \mathcal{N}(u)} \exp(a^T [W h_u \oplus W h_{v'}])}$$

when aggregating the neighbors, the importance of them is considered.

④ Gated Graph Neural Networks (GGNN)

$$h_u^{(k)} = \underbrace{\mathrm{GRU}}_{\text{recurrent update of the state}}\left(h_u^{(k-1)}, m_{\mathcal{N}(u)}^{(k)}\right)$$