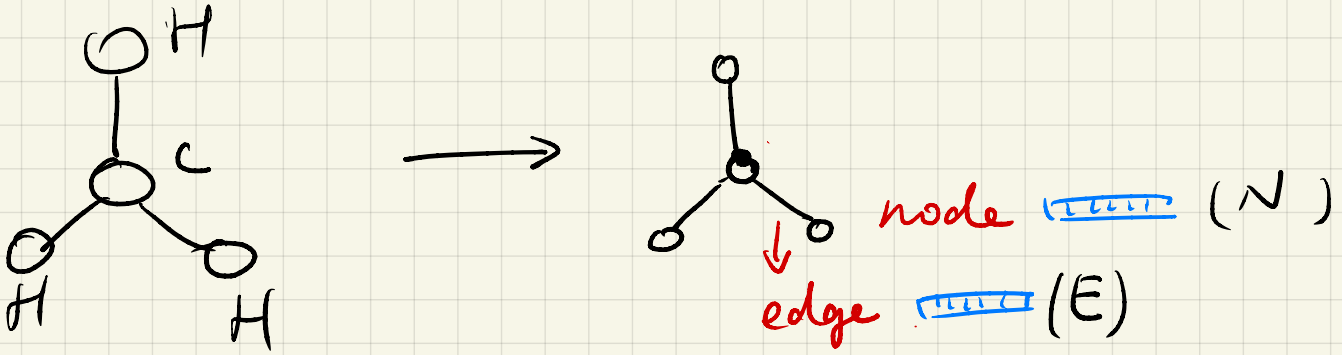
PASM 2026 Lecture 2

Graph neural networks

Undirected graphs are widely used as a natural way for atomistic system representation.

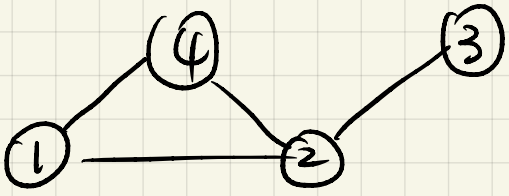
Atomic system → Graph (G)



A graph is $G(N, E)$.

To solve a domain problem, it is important to choose a proper graph representation.

Adjacency matrix is used to represent graph connectivity.

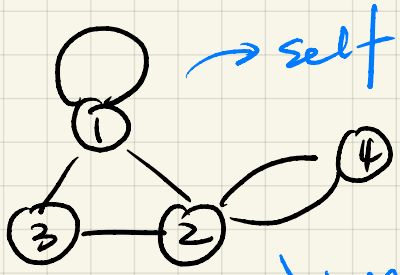


$A_{ij} = 1$, there is an edge between two nodes

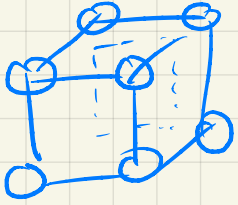
$A_{ij} = 0$, otherwise.

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

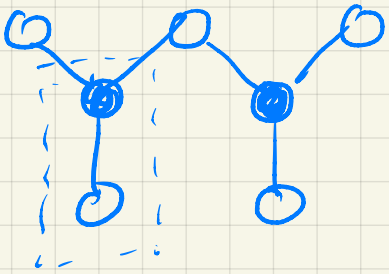
symmetric & sparse.



→ self edge →



↳ multiedge →



For solids, self edges and multiedges are commonly presented.

Now we have a representation, let's build a NN by designing a graph convolution:

A challenge: graph is permutation invariant!

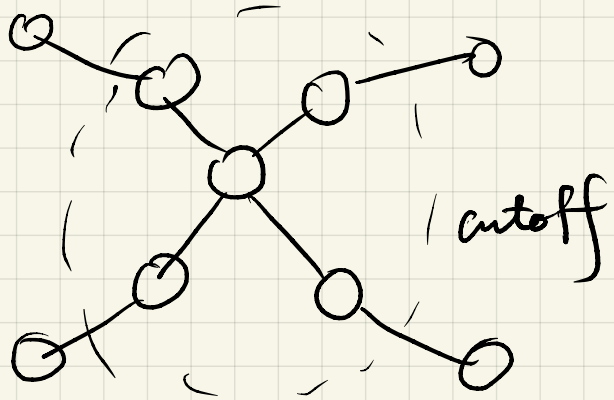
Idea: transform information at the neighboring nodes and combine them to generate new node information

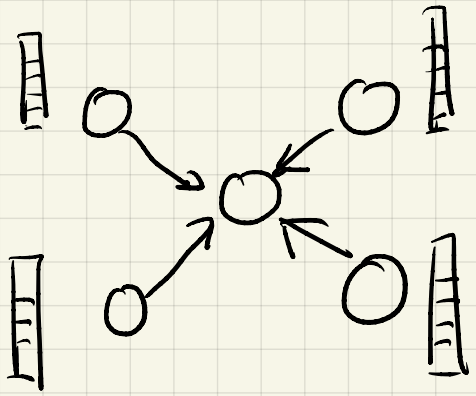
- transform "message" h_i from neighbors:

$$W_i h_i$$

↓ transformation matrix

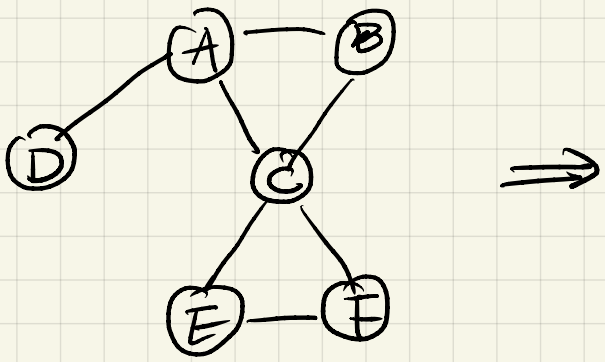
- combine the messages: $AGG(\sum_i W_i h_i)$



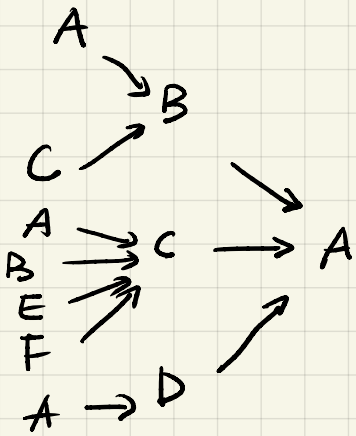


Idea: generate node representation based on network neighborhoods. (similar to tight binding, locality is the key!)

Now, let's see how to perform the convolution:



Graph



Computation graph for node A.

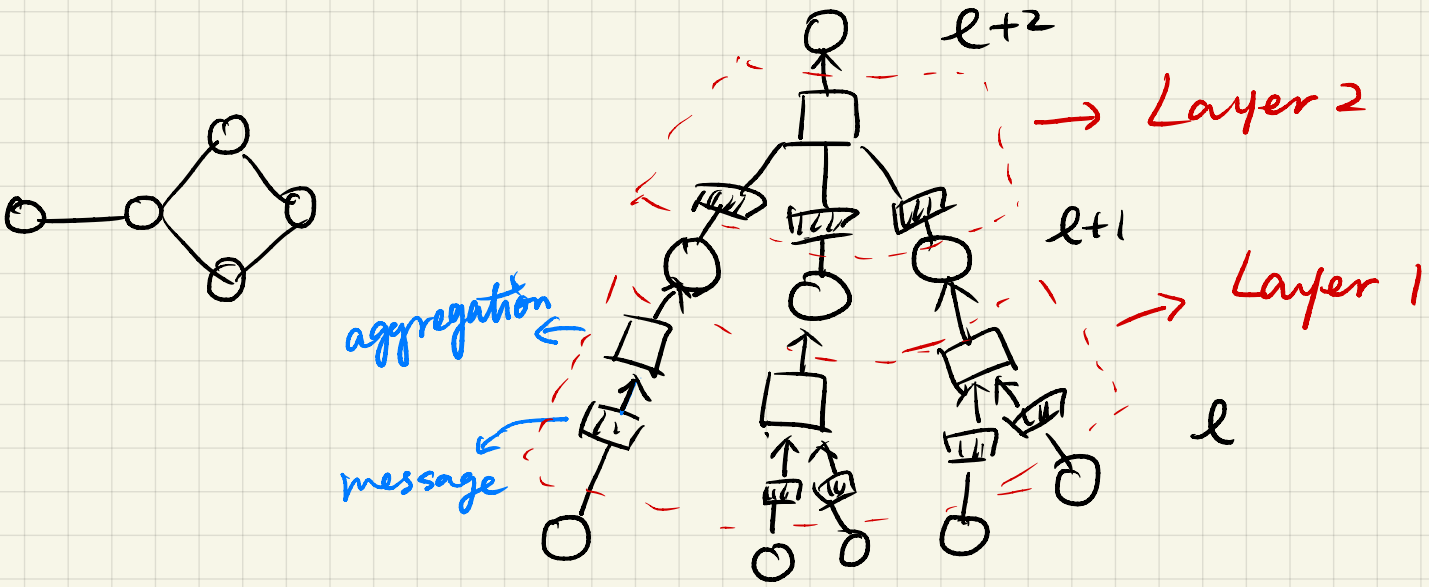
like a NN!

- Computation graph is a widely used tool for compiling programs, optimizing the allocation of computation.
- Each node has its own computation graph.
- Network neighborhood relation $\leftrightarrow$ computation graph.

A general GNN framework : Message Passing NN

J. Gilmer (DeepMind) arXiv(2017)

Basic idea : GNN layer = Message + Aggregation.



A GNN layer :

- Compress a set of vectors into a single vector
- Two steps : { message computation, aggregation.

Message computation :

$$m_u^{(l)} = MSG^{(l)}\left(h_u^{(l-1)}\right)$$ — message function.

Example : a linear layer

$$m_u^{(l)} = W^{(l)} h_u^{(l-1)}$$

↳ weight matrix

Aggregation:

multiple vectors → single vector.

$$h_v^{(l)} = AGG^{(l)}\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right).$$

($N(v)$: neighbors.)

Example: Sum(·), mean(·), Max(·)

$$h_v^{(l)} = CONCAT\left(AGG\left(\left\{m_u^{(l)}, u \in N(v)\right\}\right), m_v^{(l)}\right)$$

$m_v^{(l)} = B^{(l)} h_v^{(l-1)}$

Activation:

↳ nonlinearity & expressiveness

$\sigma(\cdot)$ {ReLU, sigmoid, etc} can be added to message and/or aggregation.

- Matrix form for graph convolution is available.
- One key nature is that after one GNN layer, the graph itself doesn't change.

⇒ Oversmoothing issue & much fewer GNN layer can be used in a GNN architecture.

Attention: a mechanism that focuses on important neighbors and fade out the rest.

Let's define an attention weight:

$$e_{uv} = a\left(W^{(l)} h_u^{(l-1)}, W^{(l)} h_v^{(l-1)}\right)$$

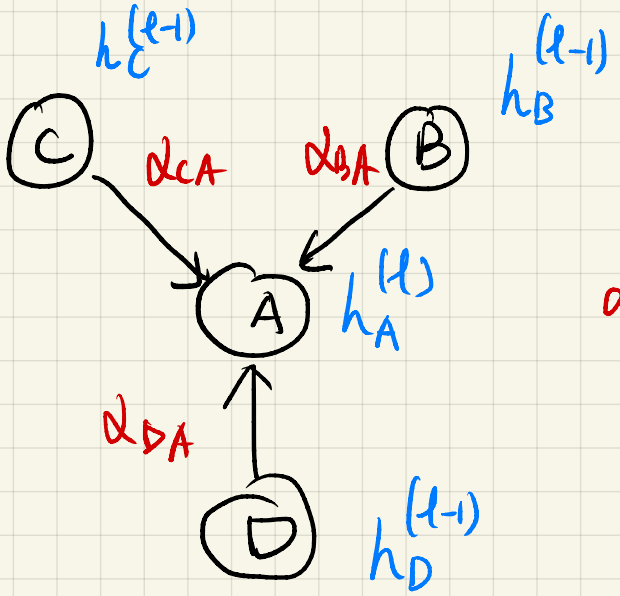
↳ the importance of u's message to v.

Normalize the attention weights:

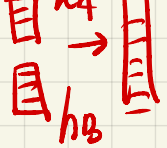
$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{k \in N(v)} \exp(e_{uv})} \Rightarrow \sum_{u \in N(v)} \alpha_{uv} = 1$$

Now, we calculate the weighted sum:

$$h_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)}\right)$$



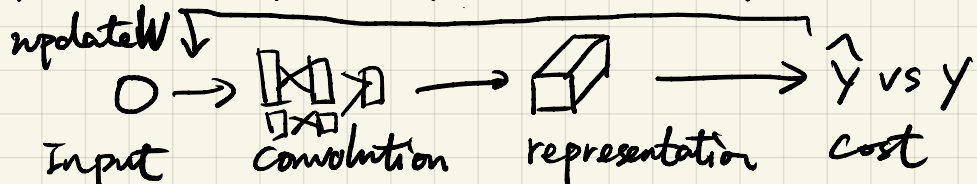
In reality, a can be a single-layer NN or a geometric vector.



e_{AB} are learned in GNN.

Now, we introduced three NN architectures:

NN, CNN, and GNN. They differ in the ways to represent input data and to facilitate information passing layer by layer efficiently.



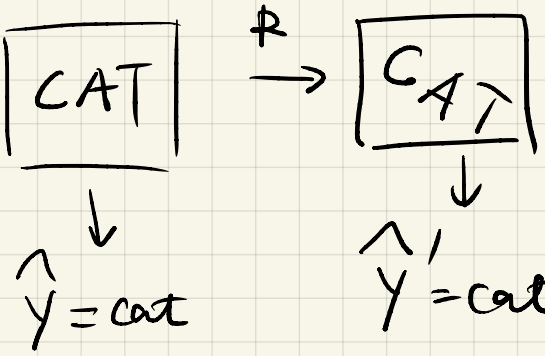
How to incorporate symmetry?

If symmetry can be written in the cost, problem solved!

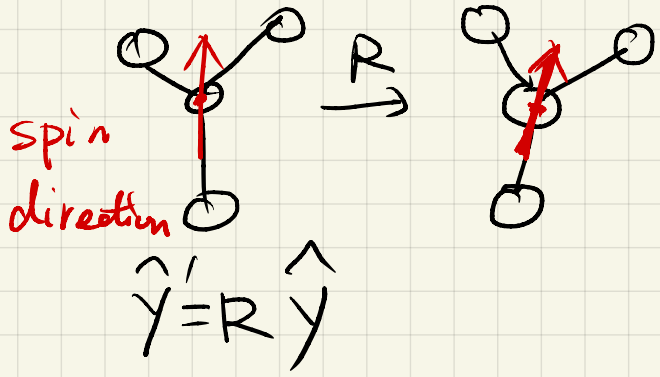
Example: PINN (conservation laws are written in the cost)

If trained with enough transformed samples, NN may approximate the symmetry, but it doesn't satisfy the symmetry exactly!

In image processing,



In atomic system,



The right NN symmetry depends on output type

- Scalar ⇒ invariant network
- Vector / tensor / Hamiltonian element / field ⇒ equivariant network needed

Equivariant neural networks (ENN)

Question: How should a prediction change when the input is rotated, translated, reflected, or permuted ?

OR:

How symmetry becomes constraints on admissible function?

ENN are NN with symmetry constraint built into the architecture.

Invariance vs Equivariance in NN framework.

A feed-forward NN can be viewed as just a function f mapping inputs x to outputs y.

inputs of NN ↑

Let's say there is a symmetry group G acting on x:

$$x \to T_g(x)$$

x: input object

g: symmetry operation in group G.

$T_g(x)$: transformed input.

The NN f is **invariant** if:

$$f(T_g(x)) = f(x) \quad \text{for any } g \in G$$

Given two separate actions $\{T_g\}$ and $\{T'_g\}$ of G on X and Y.
The NN f is **equivariant** if:

$$f(T_g(x)) = T'_g(f(x)). \quad \text{for any } g \in G.$$

From this point of view, equivariance is a more general situation that invariance.

Next, we will learn how to build equivariance on the "neuron" level using group representation theory.