

AI504: Programming for Artificial Intelligence

Week 13: Graph Neural Networks

Edward Choi

Grad School of AI

edwardchoi@kaist.ac.kr

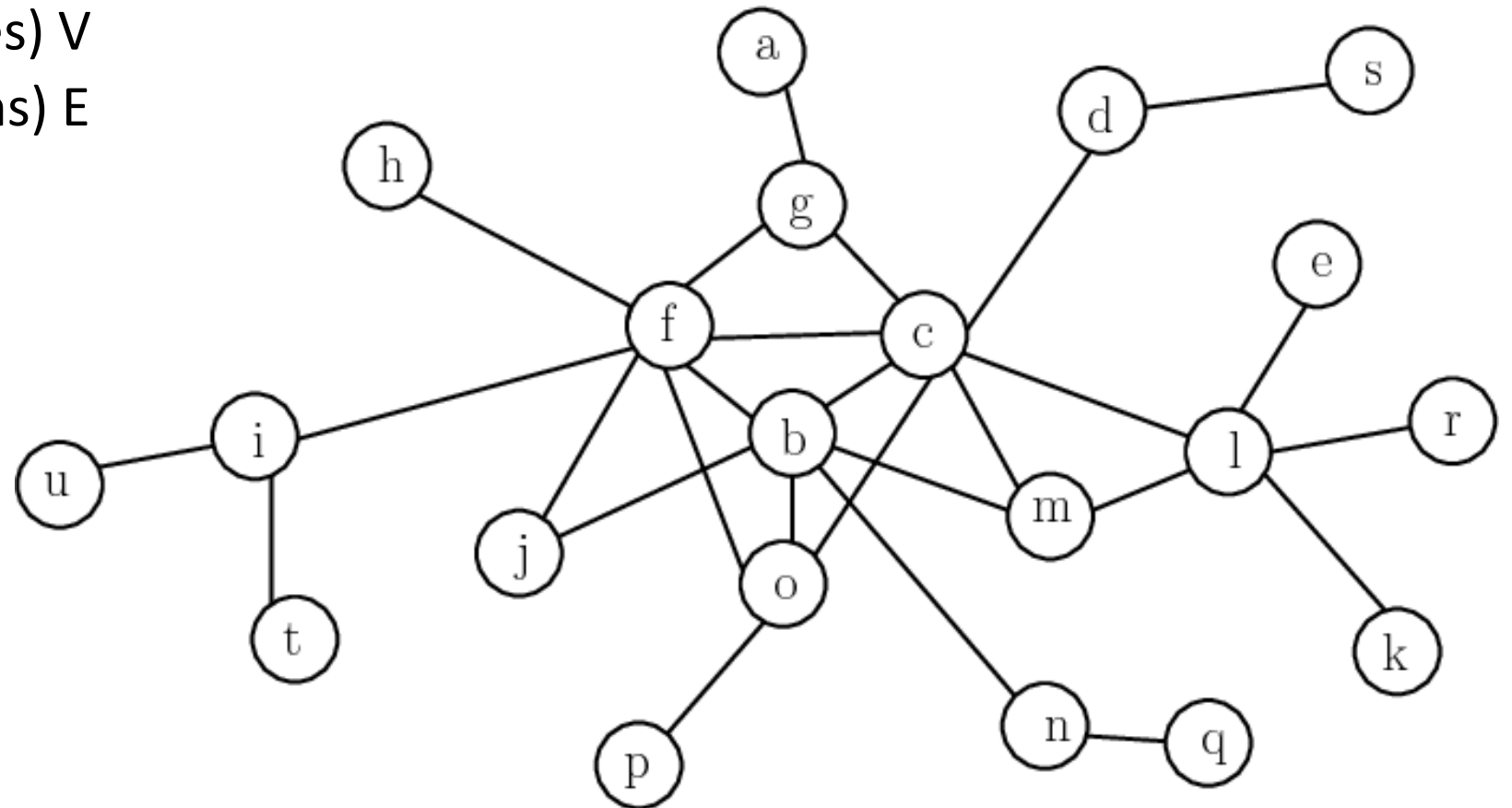
Index

- Graphs
- Graph Convolution
 - Relation with ConvNets
- Graph Neural Networks & Transformer
 - Self-attention VS graph convolution

Graphs

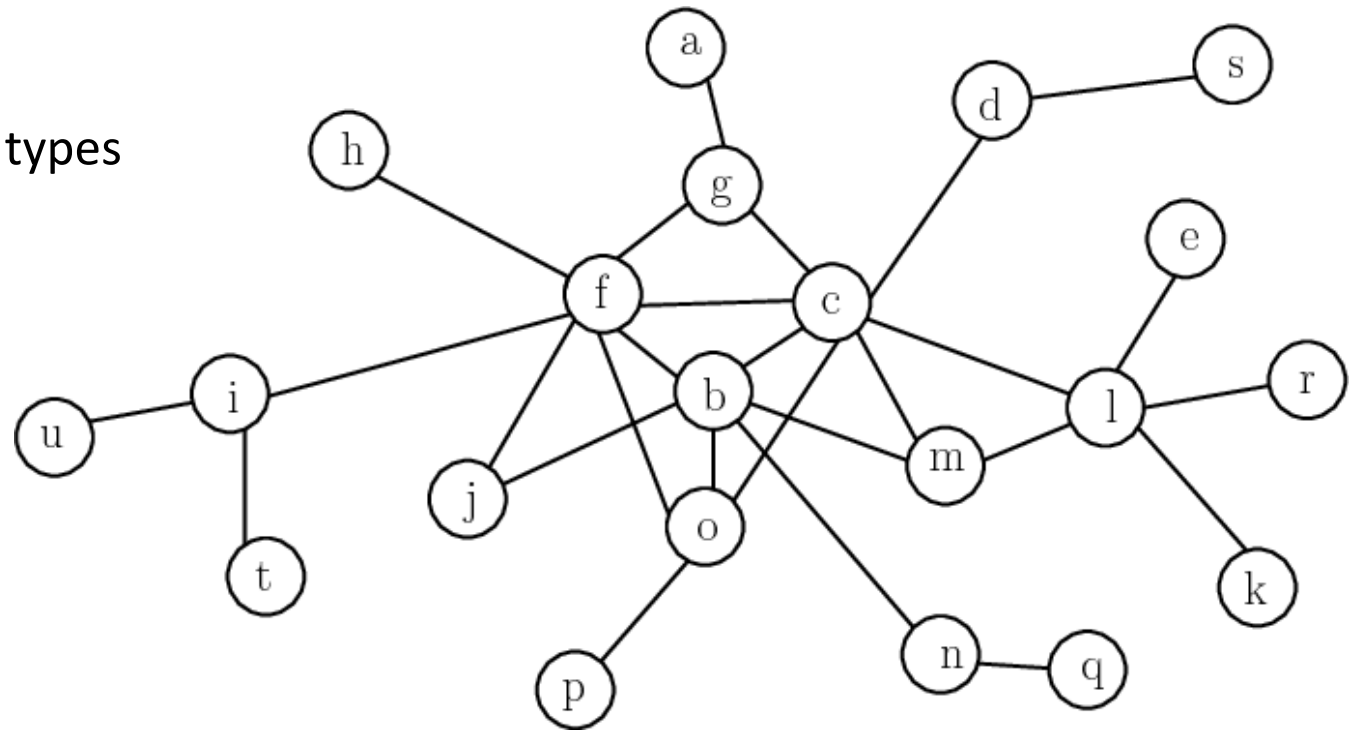
Graph

- Data consists of
 - Nodes (i.e. Vertices) V
 - Edges (i.e. relations) E



Graph

- Data consists of
 - Nodes (i.e. Vertices) V
 - Could have node-specific features
 - Edges (i.e. relations) E
 - Could be undirected or directed
 - There could be multiple relation types

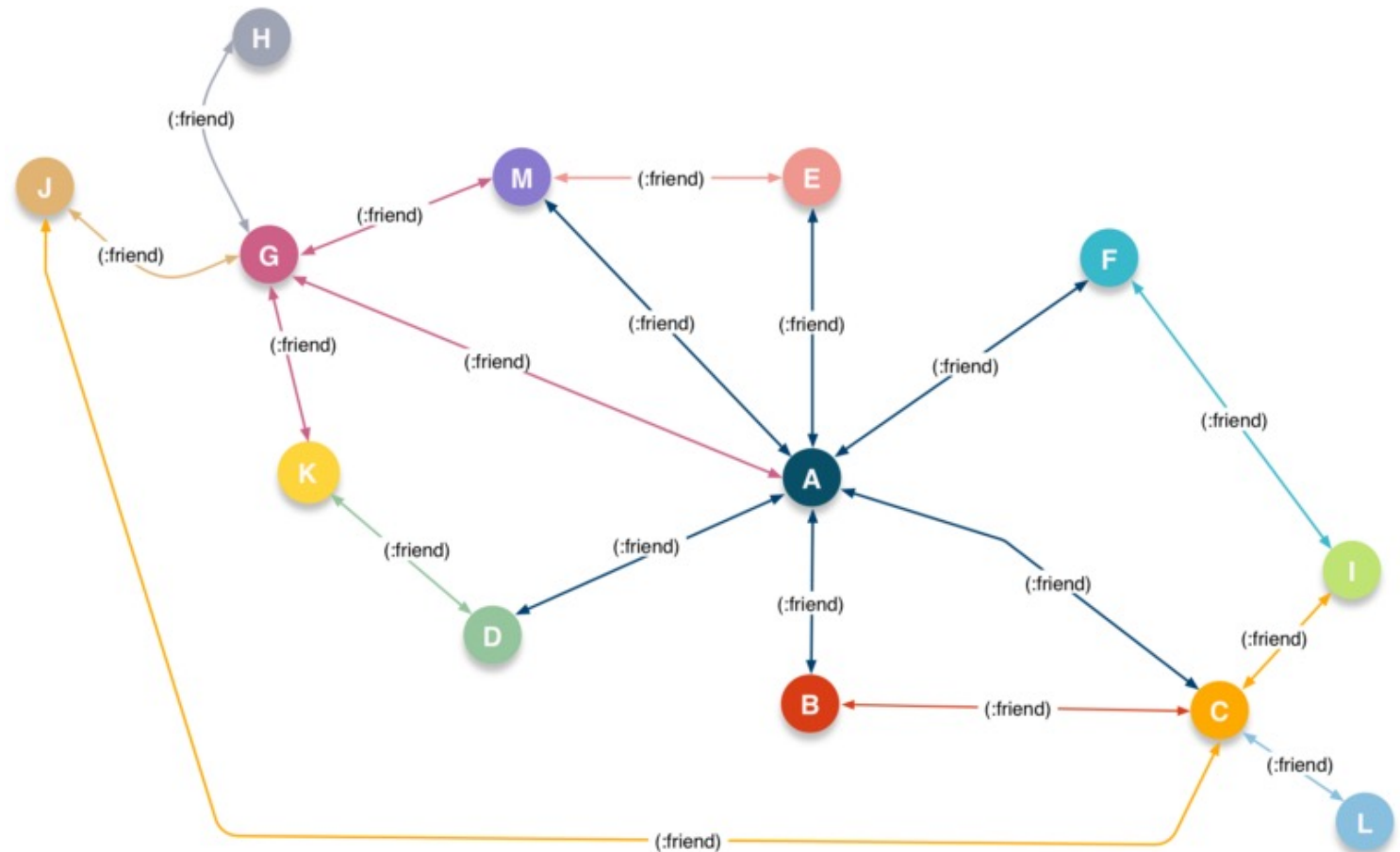


Graph

- Data consists of
 - Nodes (i.e. Vertices) V
 - Edges (i.e. relations) E
- Many datasets are graphs
 - Facebook friends
 - Web documents
 - Road networks
 - Chemical compounds
 - Knowledge bases

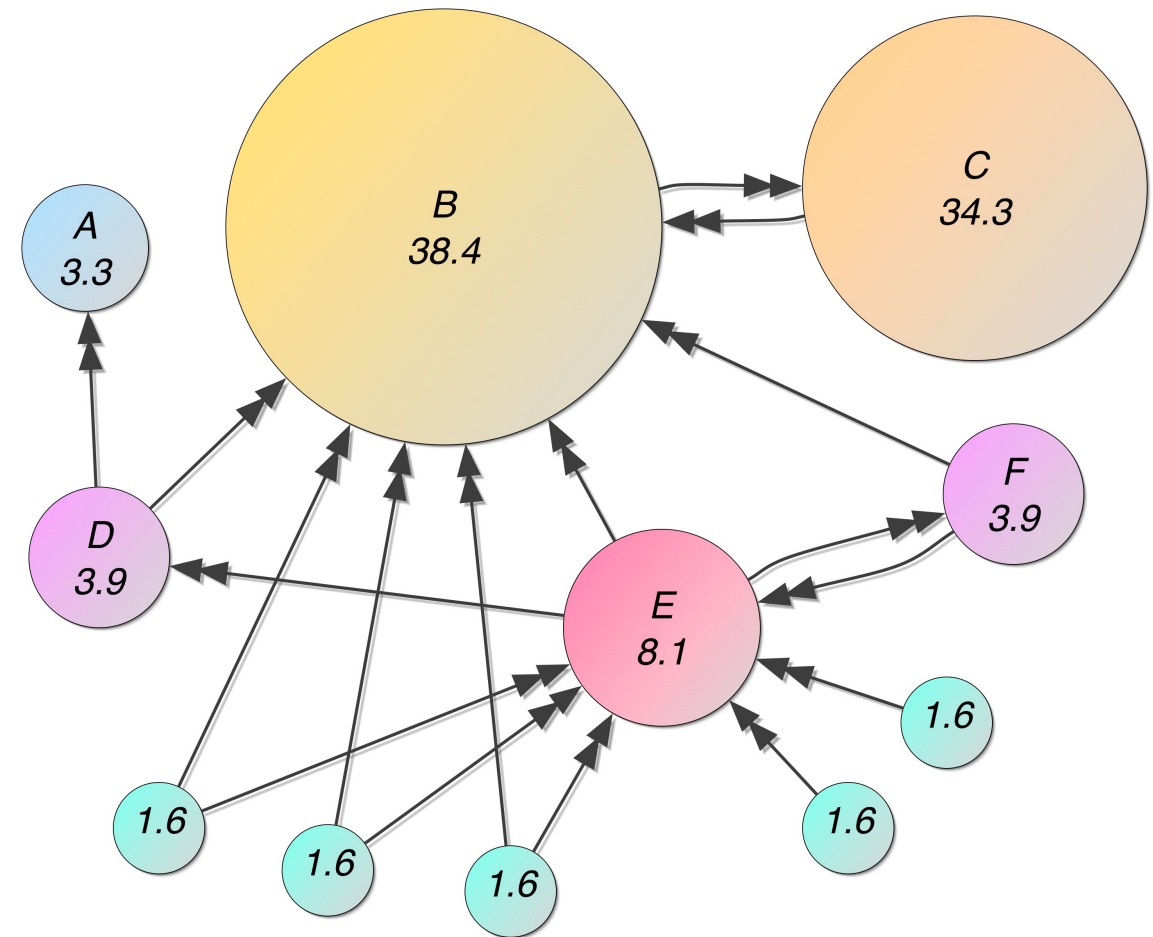
Graph Data Examples

- Friends Network
 - Nodes are unspecific
 - Undirected edges
- Social Network Analysis
 - Hot topic since SNS
 - Can find influencers
 - Can recommend friends



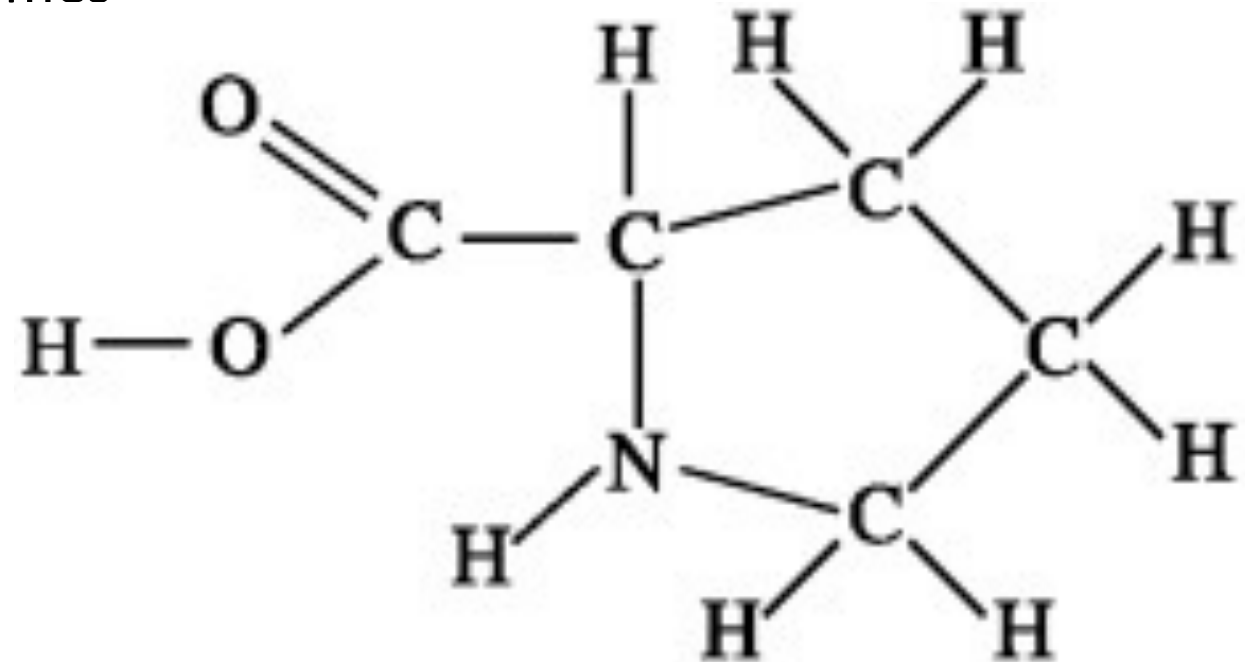
Graph Data Example

- Web documents
- PageRank
 - Made Google Search possible
 - Unspecific node
 - Each webpage is just a node
 - Directed edges
 - Outgoing links are edges
 - Calculated based on random walk.
 - The more incoming links, the more valuable a webpage is!



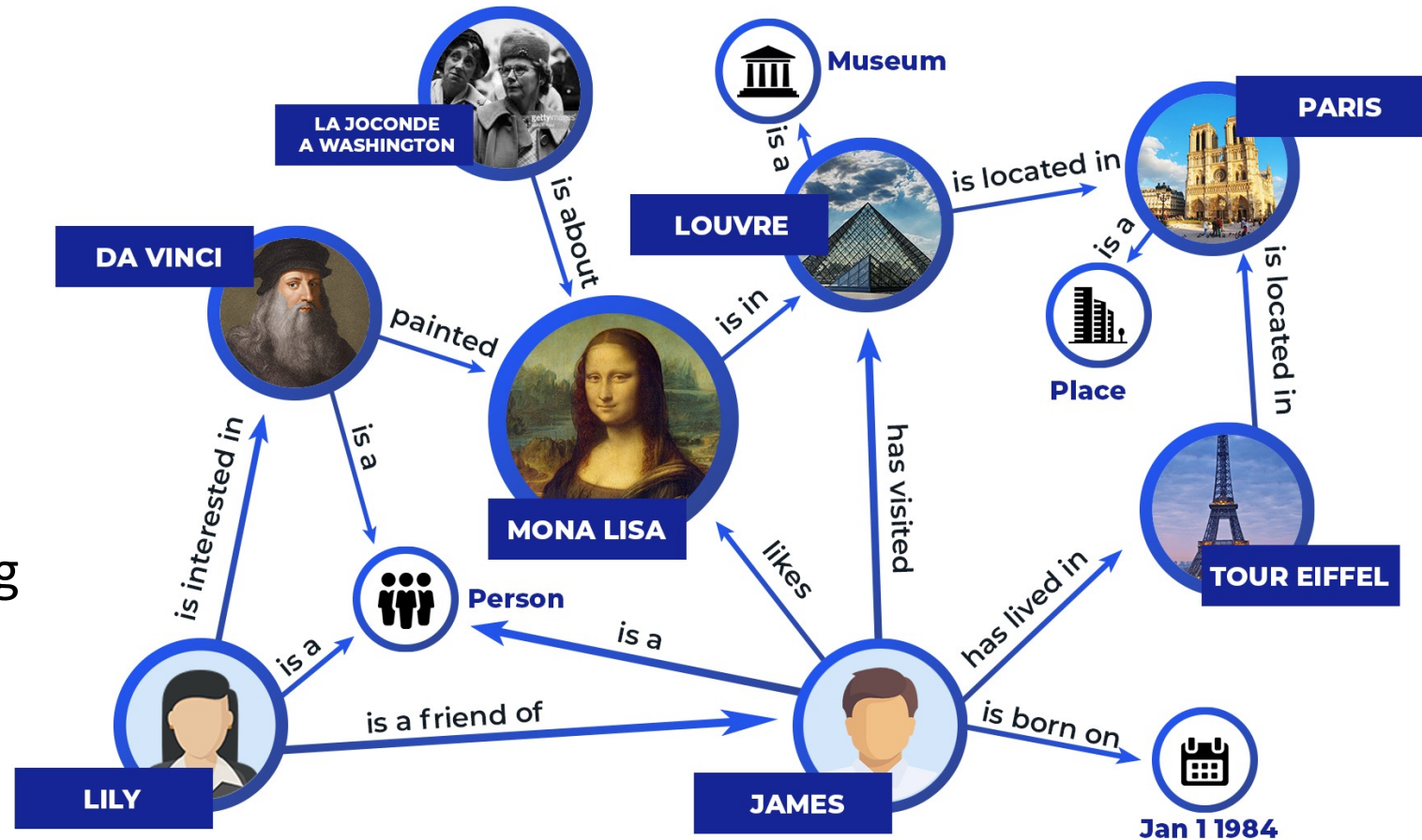
Graph Data Example

- Chemical structures
- Unspecific duplicate nodes
 - Same C (carbon) is used multiple times
- Undirected multi-type edges
 - Single bond, double bond
- Hot topic these days
 - Drug development
 - Toxicity prediction



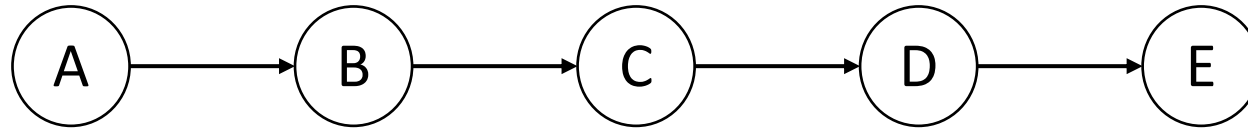
Graph Data Example

- Knowledge Base
- Multiple node types
 - Entity, value
- Directed, multi-type edges
 - is_a, parent_of, located_at
- Structured knowledge
 - Popular topic
 - Knowledge grounded reasoning



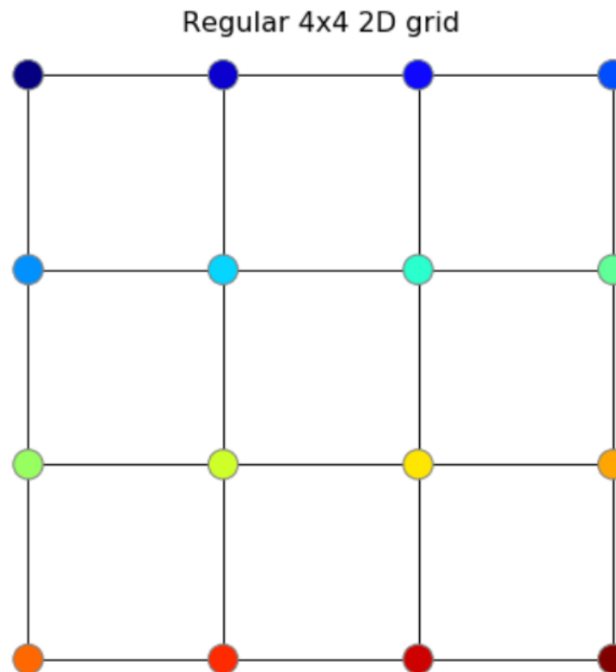
Generally Speaking...

- Everything is a graph
- Sequences are a special case of graphs (i.e. directed chains)



Generally Speaking...

- Everything is a graph
- Images are a special case of a graph (i.e. undirected grids)



Graph Convolution

Graph Representation

- How can we represent graphs?
- Images can be represented by ConvNets
 - 128x128 RGB image → ResNet → 2048-dimensional feature vector
- Text can be represented by RNN / BERT
 - 20 token text → RNN/BERT → 20 contextualized embedding
- Graph?
 - Graph → ? → ?

Graph Representation

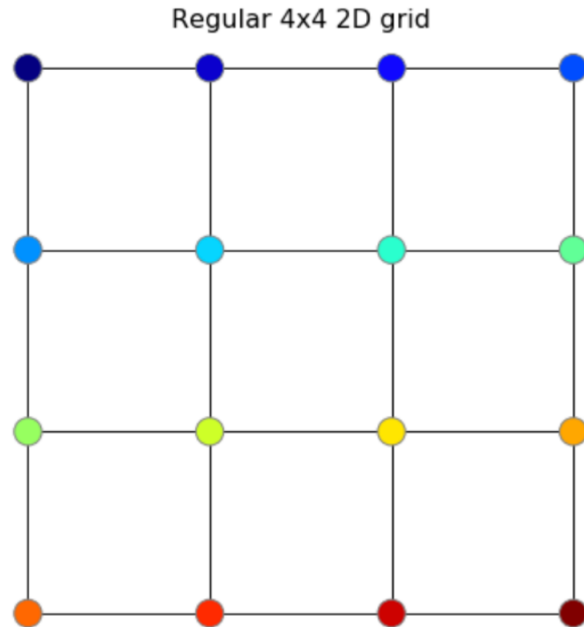
- How can we represent graphs?
- Images can be represented by ConvNets
 - 128x128 RGB image → ResNet → 2048-dimensional feature vector
- Text can be represented by RNN / BERT
 - 20 token text → RNN/BERT → 20 contextualized embedding
- Graph?
 - Graph with $|V|$ nodes and $|E|$ edges → ? → ?

Graph Representation

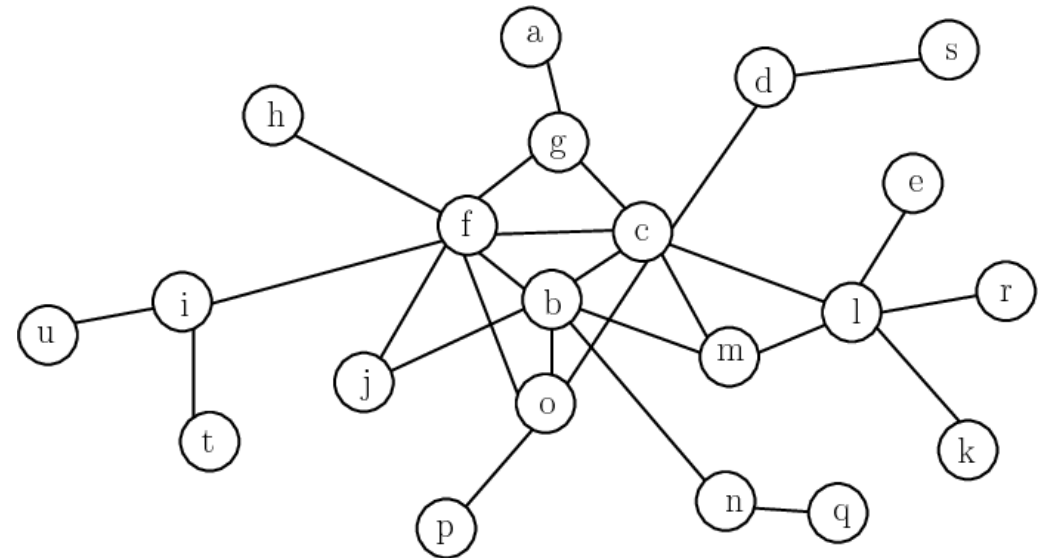
- How can we represent graphs?
- Images can be represented by ConvNets
 - 128x128 RGB image → ResNet → 2048-dimensional feature vector
- Text can be represented by RNN / BERT
 - 20 token text → RNN/BERT → 20 contextualized embedding
- Graph?
 - $|V|$ nodes and $|E|$ edges → ? → $|V|$ embeddings (and $|E|$ embeddings)

Why not ConvNet?

- Convolution filters assume same number of neighbors
 - General graphs assume no such thing...

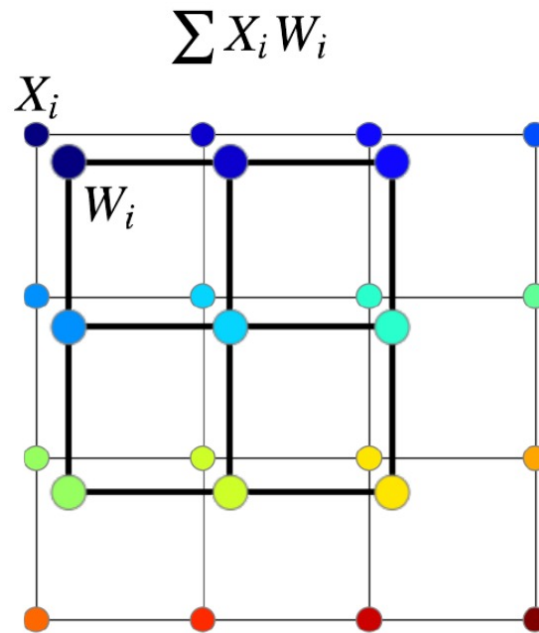


VS



Why not ConvNet?

- Convolution filters assume same number of neighbors
 - General graphs assume no such thing...
- But we can use the core principle of ConvNet filters
 - Aggregate features from the local neighbors



Graph Convolution Principle

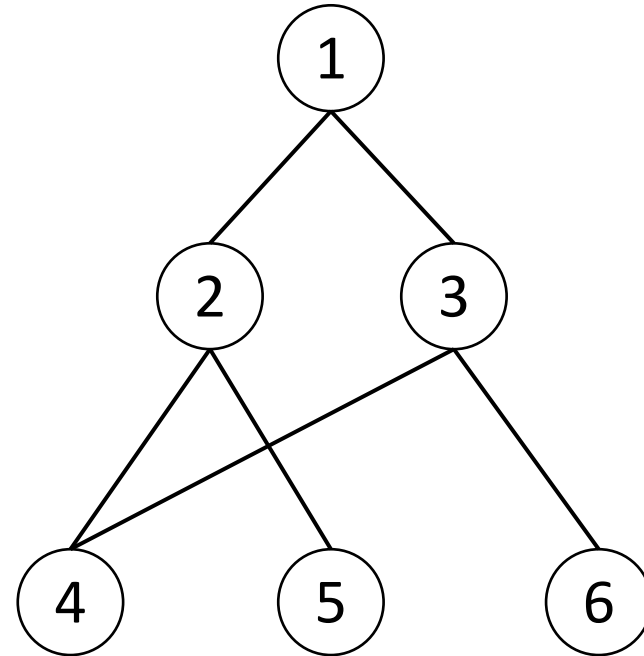
Given a graph $G = (V, E)$,

- At each node v_i , aggregate all neighbors' features
 - $\mathbf{a}_i = \sum_{v_j \in \mathcal{A}_i} f(v_j)$, where \mathcal{A}_i : Set of nodes connected to v_i
- Combine v_i 's feature with neighbors' features
 - $\mathbf{h}_i = g(f(v_i), \mathbf{a}_i)$
- $\mathbf{h}_i \rightarrow$ Representation of node v_i

Graph Convolution Equation

- A: Adjacency matrix

0	1	1	0	0	0
1	0	0	1	1	0
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0



Graph Convolution Equation

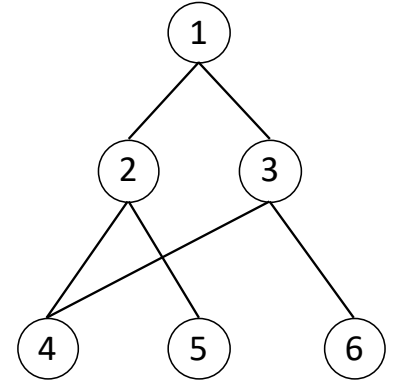
- A: Adjacency matrix
- X: Node index
- W: Node embedding vector

0	1	1	0	0	0
1	0	0	1	1	0
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0

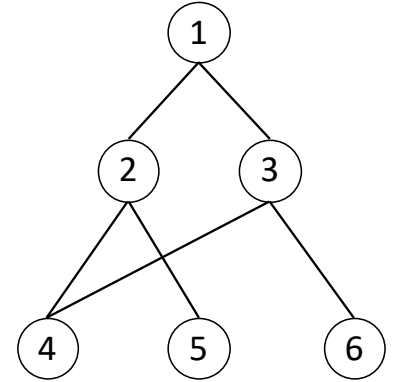
A

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

XW



Graph Convolution Equation



- AXW

0	1	1	0	0	0
1	0	0	1	1	0
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0

A

\times

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

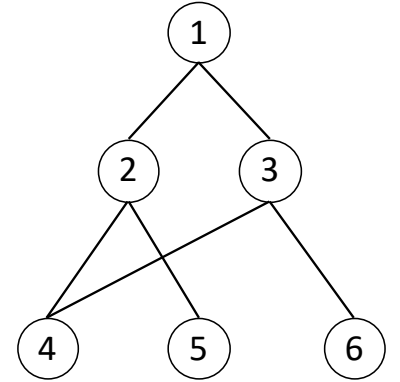
XW

$=$

$f(v_2) + f(v_3)$
$f(v_1) + f(v_4) + f(v_5)$
$f(v_1) + f(v_4) + f(v_6)$
$f(v_2) + f(v_3)$
$f(v_2)$
$f(v_3)$

AXW

Graph Convolution Equation



- $AXW \rightarrow$ Is this the node representations H ?

0	1	1	0	0	0
1	0	0	1	1	0
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0

A

\times

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

XW

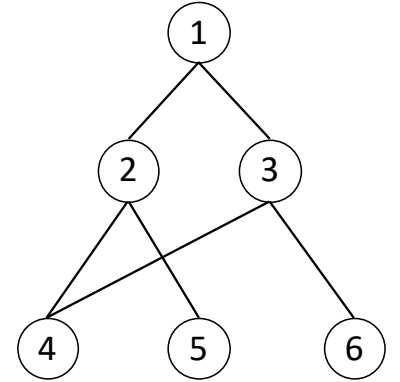
$=$

$f(v_2) + f(v_3)$
$f(v_1) + f(v_4) + f(v_5)$
$f(v_1) + f(v_4) + f(v_6)$
$f(v_2) + f(v_3)$
$f(v_2)$
$f(v_3)$

AXW

Graph Convolution Equation

- No, AXW is just neighbor aggregation.
- We need the combination step!



0	1	1	0	0	0
1	0	0	1	1	0
1	0	0	1	0	1
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0

A

\times

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

XW

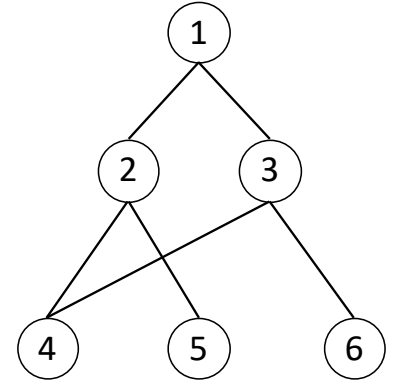
$=$

$f(v_2) + f(v_3)$
$f(v_1) + f(v_4) + f(v_5)$
$f(v_1) + f(v_4) + f(v_6)$
$f(v_2) + f(v_3)$
$f(v_2)$
$f(v_3)$

AXW

Graph Convolution Equation

- New $A' = A + I$
 - I : Identity matrix



1	1	1	0	0	0
1	1	0	1	1	0
1	0	1	1	0	1
0	1	1	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1

$$A' = A + I$$

×

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

$$XW$$

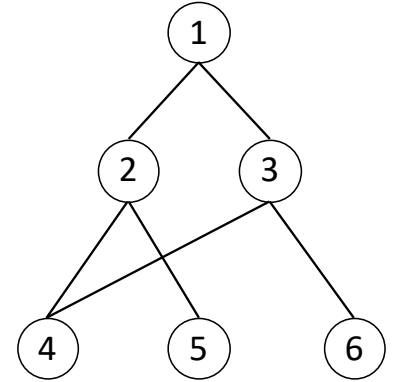
=

$f(v_1) + f(v_2) + f(v_3)$
$f(v_1) + f(v_2) + f(v_4) + f(v_5)$
$f(v_1) + f(v_3) + f(v_4) + f(v_6)$
$f(v_2) + f(v_3) + f(v_4)$
$f(v_2) + f(v_5)$
$f(v_3) + f(v_6)$

$$A'XW$$

Graph Convolution Equation

- $A'XW$ performs neighbor aggregation and combination
 - Combination function g is just simple summation.
- $A'XW$ can be node representations H !



1	1	1	0	0	0
1	1	0	1	1	0
1	0	1	1	0	1
0	1	1	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1

$$A' = A + I$$

×

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

$$XW$$

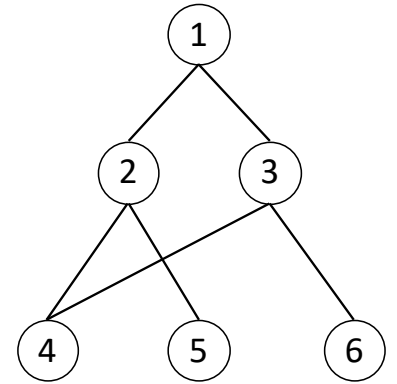
=

$h_1 = f(v_1) + f(v_2) + f(v_3)$
$h_2 = f(v_1) + f(v_2) + f(v_4) + f(v_5)$
$h_3 = f(v_1) + f(v_3) + f(v_4) + f(v_6)$
$h_4 = f(v_2) + f(v_3) + f(v_4)$
$h_5 = f(v_2) + f(v_5)$
$h_6 = f(v_3) + f(v_6)$

$$A'XW$$

Graph Convolution Equation

- But there is another problem:
- Scales of node features differ by the number of neighbors!
 - h_2 can be **twice** as large as h_5 !



1	1	1	0	0	0
1	1	0	1	1	0
1	0	1	1	0	1
0	1	1	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1

$$\mathbf{A}' = \mathbf{A} + \mathbf{I}$$

\times

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

$$\mathbf{XW}$$

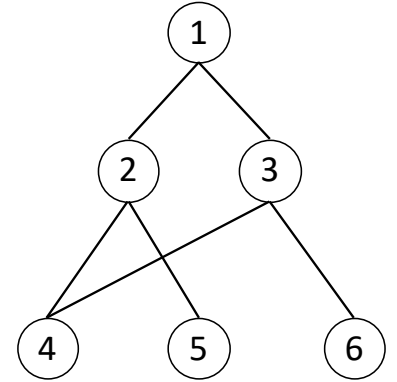
$=$

$h_1 = f(v_1) + f(v_2) + f(v_3)$
$h_2 = f(v_1) + f(v_2) + f(v_4) + f(v_5)$
$h_3 = f(v_1) + f(v_3) + f(v_4) + f(v_6)$
$h_4 = f(v_2) + f(v_3) + f(v_4)$
$h_5 = f(v_2) + f(v_5)$
$h_6 = f(v_3) + f(v_6)$

$$\mathbf{A}'\mathbf{XW}$$

Graph Convolution Equation

- D: Degree matrix
 - Diagonal matrix where $d_{i,i}$ = number of edges
- $D^{-1}A'$: Normalized adjacency matrix



.33	.33	.33	0	0	0
.25	.25	0	.25	.25	0
.25	0	.25	.25	0	.25
0	.33	.33	.33	0	0
0	.5	0	0	.5	0
0	0	.5	0	0	.5

$D^{-1}A'$

\times

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

XW

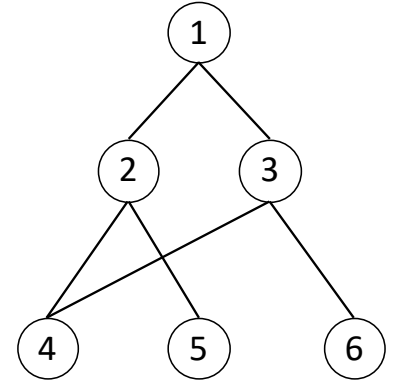
$=$

$h_1 = 0.33 * (f(v_1) + f(v_2) + \dots)$
$h_2 = 0.25 * (f(v_1) + f(v_2) + \dots)$
$h_3 = 0.25 * (f(v_1) + f(v_3) + \dots)$
$h_4 = 0.33 * (f(v_2) + f(v_3) + \dots)$
$h_5 = 0.5 * (f(v_2) + f(v_5))$
$h_6 = 0.5 * (f(v_3) + f(v_6))$

$D^{-1}A'XW$

Graph Convolution Equation

- One more thing:
 - $D^{-1}A'XW \rightarrow$ All linear operations
- Need non-linearity



$$\sigma \left(\begin{array}{|c|c|c|c|c|c|} \hline .33 & .33 & .33 & 0 & 0 & 0 \\ \hline .25 & .25 & 0 & .25 & .25 & 0 \\ \hline .25 & 0 & .25 & .25 & 0 & .25 \\ \hline 0 & .33 & .33 & .33 & 0 & 0 \\ \hline 0 & .5 & 0 & 0 & .5 & 0 \\ \hline 0 & 0 & .5 & 0 & 0 & .5 \\ \hline \end{array} \right) \times \begin{array}{|c|} \hline f(v_1) \\ \hline f(v_2) \\ \hline f(v_3) \\ \hline f(v_4) \\ \hline f(v_5) \\ \hline f(v_6) \\ \hline \end{array} = \sigma \left(\begin{array}{|c|} \hline h_1 = 0.33 * (f(v_1) + f(v_2) + ...) \\ \hline h_2 = 0.25 * (f(v_1) + f(v_2) + ...) \\ \hline h_3 = 0.25 * (f(v_1) + f(v_3) + ...) \\ \hline h_4 = 0.33 * (f(v_2) + f(v_3) + ...) \\ \hline h_5 = 0.5 * (f(v_2) + f(v_5)) \\ \hline h_6 = 0.5 * (f(v_3) + f(v_6)) \\ \hline \end{array} \right)$$

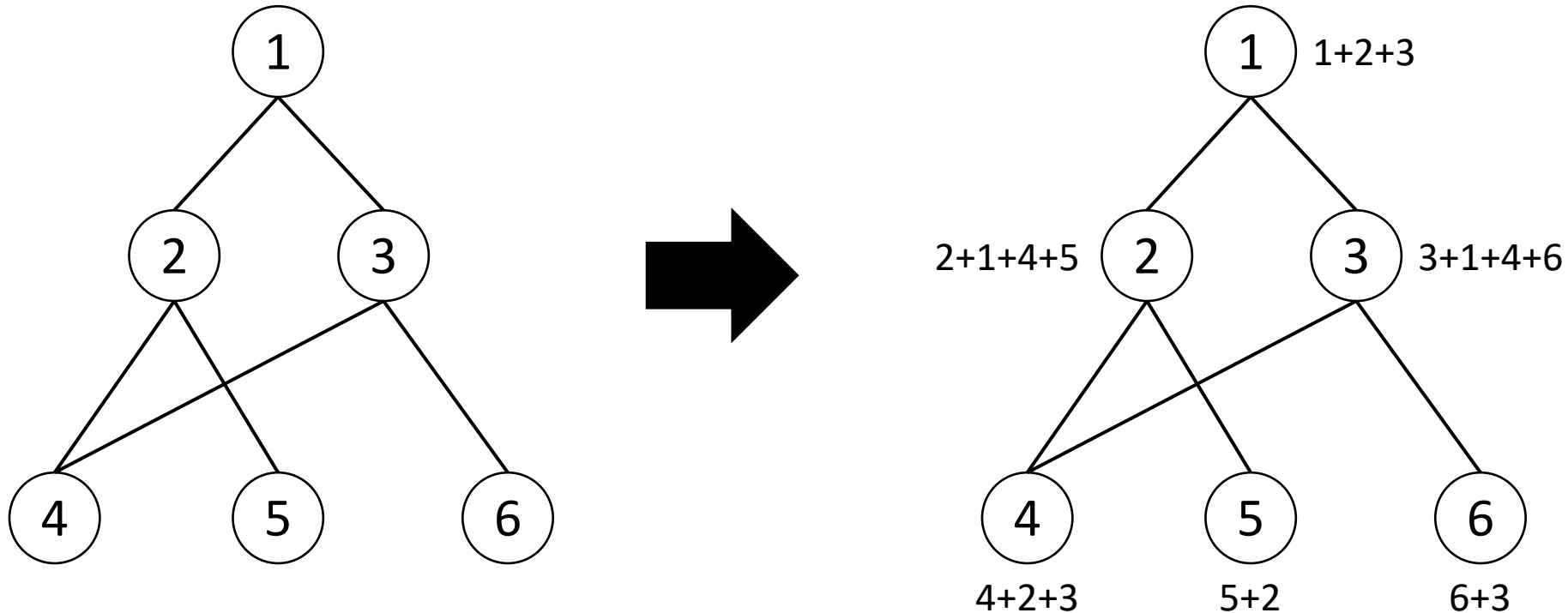
$D^{-1}A'$ **XW** **$D^{-1}A'XW$**

Graph Convolution Equation

- $\mathbf{H} = \sigma(\mathbf{D}^{-1}\mathbf{A}'\mathbf{X}\mathbf{W})$
 - $\mathbf{A}' = \mathbf{A} + \mathbf{I}$
 - \mathbf{D} = Degree matrix
 - σ = Non-linear activation
 - \mathbf{W} = Learnable parameters

Graph Convolution Equation

- $\mathbf{H}^{(1)} = \sigma(\mathbf{D}^{-1}\mathbf{A}'\mathbf{X}\mathbf{W})$
- This is aggregating neighbors just 1-hop away



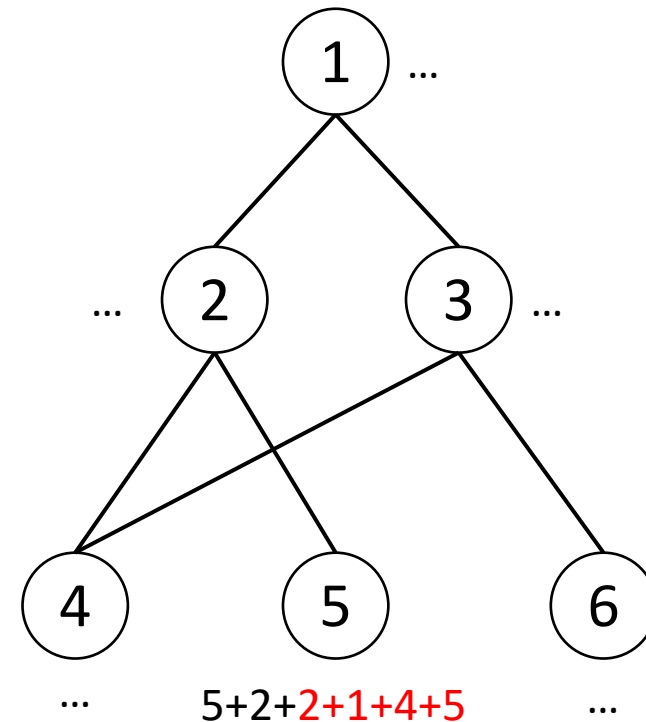
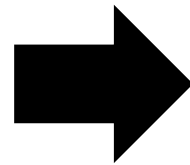
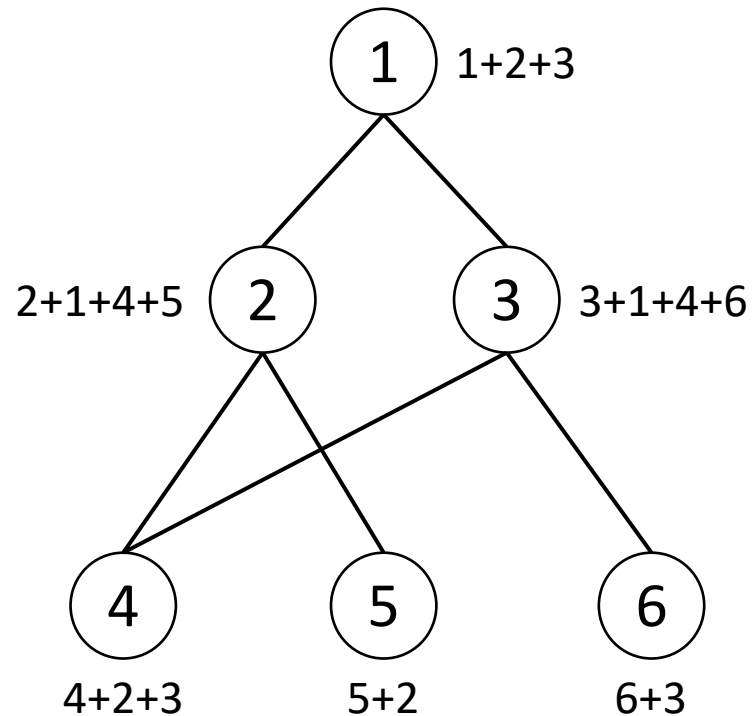
Graph Convolution Equation

- $\mathbf{H}^{(1)} = \sigma(\mathbf{D}^{-1}\mathbf{A}'\mathbf{X}\mathbf{W})$
- This is aggregating neighbors just 1-hop away
- How do we aggregate neighbors 2-hops away?

Graph Convolution Equation

- How do we aggregate neighbors 2-hops away?

$$\rightarrow H^{(2)} = \sigma(D^{-1}A'H^{(1)}W^{(2)})$$



Graph Convolution Equation

- How do we aggregate neighbors k-hops away?

$$\rightarrow \mathbf{H}^{(k)} = \sigma(\mathbf{D}^{-1} \mathbf{A}' \mathbf{H}^{(k-1)} \mathbf{W}^{(k)})$$

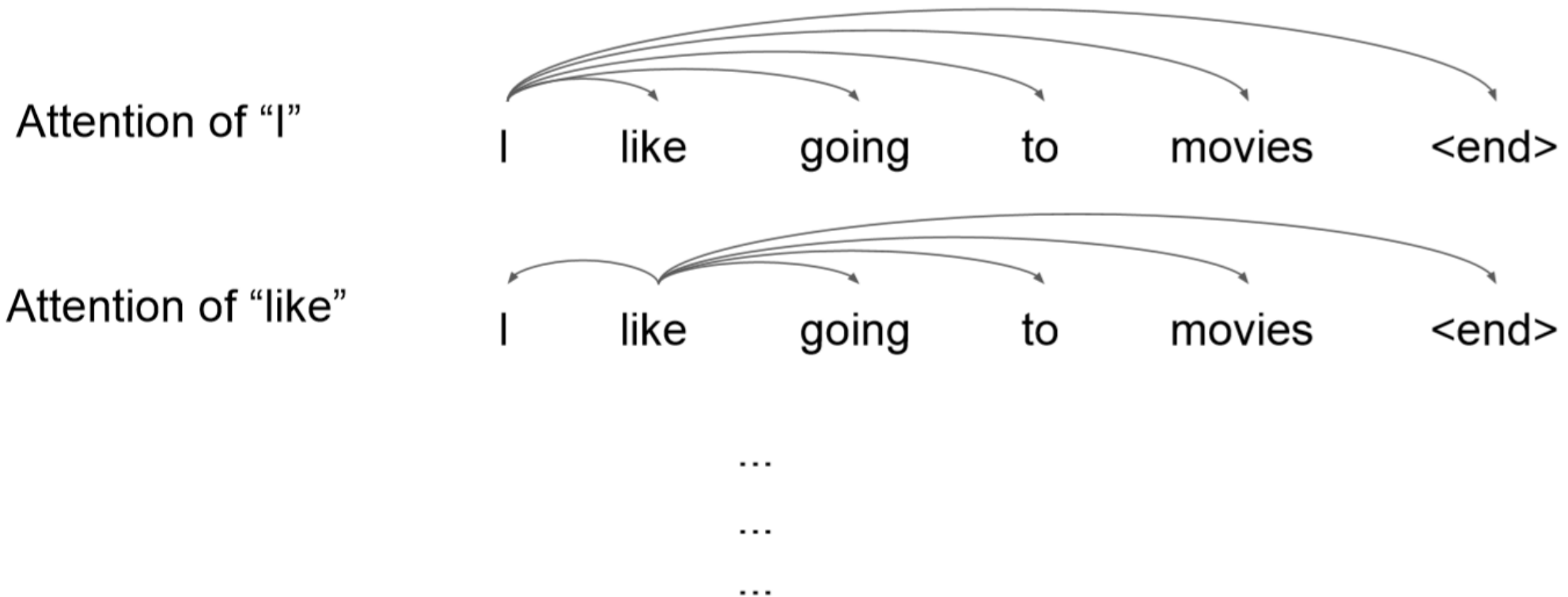
Graph Convolution Variations

- Different normalization
 - $\mathbf{H}^{(k)} = \sigma(\mathbf{D}^{-1/2} \mathbf{A}' \mathbf{D}^{-1/2} \mathbf{H}^{(k-1)} \mathbf{W}^{(k)})$
 - Motivated by spectral graph convolution
 - Whole theory regarding graph laplacian...
- Different Combination step
 - Instead of summation $g(f(v_i), \mathbf{a}_i) = f(v_i) + \mathbf{a}_i$,
 - Use linear layer $g(f(v_i), \mathbf{a}_i) = \mathbf{W} \cdot [f(v_i); \mathbf{a}_i]$
- Nodes become RNNs
 - More sophisticated way to accumulate N-hop information
- Many more variations → Called Graph Neural Networks (GNN)

GNN & Transformer

Attention is All You Need

- Vaswani et al. 2017
- Let's use only attentions to handle sequences.



Self-Attention

- $Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$

0.5	0.1	0.0	0.2	0.2	0.0
0.2	0.6	0.0	0.0	0.1	0.0
..
..
..
..

$$Softmax\left(\frac{QK^T}{\sqrt{d}}\right)$$

I
like
going
to
movies
<end>

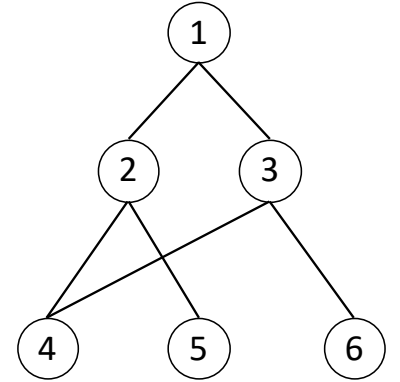
V



$0.5*I + 0.1*like + 0.2*to + 0.2* movies$
$0.2*I + 0.6*like + 0.1*movies$
...
...
...
...

$Attention(Q, K, V)$

Graph Convolution



- $\mathbf{H}^{(k)} = \sigma(\mathbf{D}^{-1} \mathbf{A}' \mathbf{H}^{(k-1)} \mathbf{W}^{(k)})$

.33	.33	.33	0	0	0
.25	.25	0	.25	.25	0
.25	0	.25	.25	0	.25
0	.33	.33	.33	0	0
0	.5	0	0	.5	0
0	0	.5	0	0	.5

$\mathbf{D}^{-1} \mathbf{A}'$

×

$f(v_1)$
$f(v_2)$
$f(v_3)$
$f(v_4)$
$f(v_5)$
$f(v_6)$

\mathbf{XW}

=

$h_1 = 0.33 * (f(v_1) + f(v_2) + \dots)$
$h_2 = 0.25 * (f(v_1) + f(v_2) + \dots)$
$h_3 = 0.25 * (f(v_1) + f(v_3) + \dots)$
$h_4 = 0.33 * (f(v_2) + f(v_3) + \dots)$
$h_5 = 0.5 * (f(v_2) + f(v_5))$
$h_6 = 0.5 * (f(v_3) + f(v_6))$

$\mathbf{D}^{-1} \mathbf{A}' \mathbf{XW}$

Transformer & Graph Networks

- Graph Networks

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \mathbf{C}^{(j-1)} \mathbf{W}^{(j)})$

- Transformer

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\text{softmax}(\frac{\mathbf{Q}^{(j)} \mathbf{K}^{(j)\top}}{\sqrt{d}}) \mathbf{V}^{(j)})$

$$\mathbf{Q}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_Q^{(j)}, \quad \mathbf{K}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_K^{(j)}, \quad \mathbf{V}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_V^{(j)}$$

Transformer & Graph Networks

- Graph Networks

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \mathbf{C}^{(j-1)} \mathbf{W}^{(j)})$

Adjacency Matrix \Leftrightarrow Self-Attention

- Transformer

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\text{softmax}(\frac{\mathbf{Q}^{(j)} \mathbf{K}^{(j)\top}}{\sqrt{d}}) \mathbf{V}^{(j)})$

$$\mathbf{Q}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_Q^{(j)}, \quad \mathbf{K}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_K^{(j)}, \quad \mathbf{V}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_V^{(j)}$$

Transformer & Graph Networks

- Graph Networks

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \mathbf{C}^{(j-1)} \mathbf{W}^{(j)})$

Node Embedding \Leftrightarrow Token Embedding

- Transformer

- $\mathbf{C}^{(j)} = \text{MLP}^{(j)}(\text{softmax}(\frac{\mathbf{Q}^{(j)} \mathbf{K}^{(j)\top}}{\sqrt{d}}) \mathbf{V}^{(j)})$

$$\mathbf{Q}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_Q^{(j)}, \quad \mathbf{K}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_K^{(j)}, \quad \mathbf{V}^{(j)} = \mathbf{C}^{(j-1)} \mathbf{W}_V^{(j)}$$

Self-Attention VS Graph Convolution

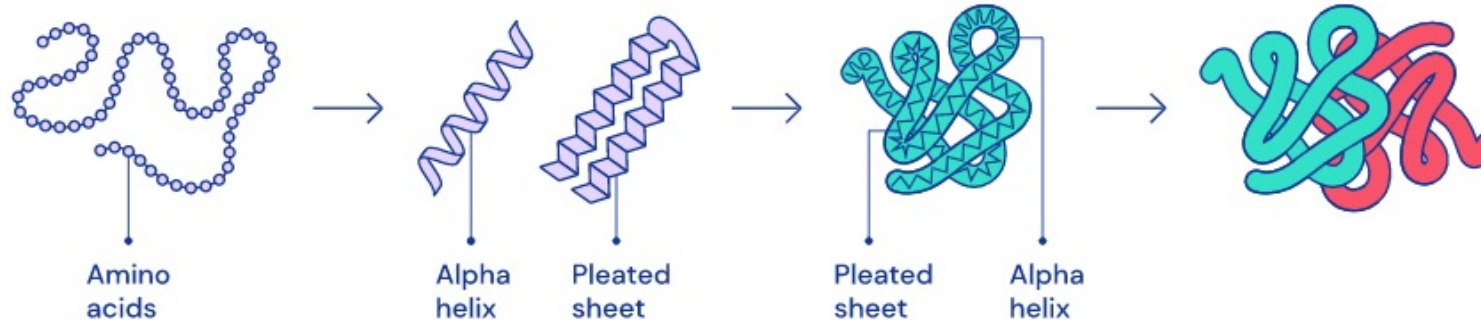
- Self-attention
 - Don't know graph structure
 - Assume (implicitly) fully-connected graph
 - Learn edge weights during training
 - Learn node embeddings in a data-driven fashion
- Graph convolution
 - Prior knowledge on graph structure
 - Learn node embeddings based on the fixed adjacency matrix

Transformer instead of GNN?

- Learn the edge weights with attention
 - Zero-out the probability of non-connected edges
 - Mask QK/\sqrt{d} with negative infinity matrix
 - Graph Attention Network (<https://arxiv.org/pdf/1710.10903.pdf>)
- Learning the structure of graphs with attention
 - Use self-attention to learn the underlying graph structure
 - Start with a prior knowledge driven adjacency matrix, then gradually evolve with self-attention
 - Prior knowledge: conditional probability between pairs of nodes
 - Graph Convolutional Transformer (<https://arxiv.org/pdf/1906.04716.pdf>)

Chemical Structures

- AlphaFold



- COVID-19

Small molecules	SMILES used in the DTI prediction	K_d in nM	MT-DTI Rank out of 3411
Atazanavir	<chem>COC(=O)NC(C(=O)NC(Cc1ccccc1)C(O)CN(Cc1ccc(-c2ccccc2)cc1)NC(=O)C(NC(=O)OC)C(C)(C)C(C)(C)C</chem>	94.94	66
Remdesivir*	<chem>CCC(CC)COC(=O)[C@H](C)N[P@](=O)(OC[C@H]1[C@H]([C@H]([C@H](O1)C#N)C2=CC=C3N2N=CN=C3N)O)O</chem> <chem>OC4=CC=CC=C4</chem>	113.13	70
Efavirenz*	<chem>O=C1Nc2ccc(Cl)cc2[C@@](C#CC2CC2)(C(F)(F)F)O1</chem>	199.17	116
Ritonavir	<chem>CC(C)c1nc(CN(C)C(=O)NC(C(=O)NC(Cc2ccccc2)CC(O)C(Cc2ccccc2)NC(=O)OCc2cncs2)C(C)C)cs1</chem>	204.05	119
Dolutegravir	<chem>CC1CCOC2Cn3cc(C(=O)NCc4ccc(F)cc4F)c(=O)c(O)c3C(=O)N12</chem>	336.91	162
Asunaprevir	<chem>C=CC1CC1(NC(=O)C1CC(Oc2ncc(OC)c3ccc(Cl)cc23)CN1C(=O)C(NC(=O)OC(C)(C)C(C)(C)C(=O)NS(=O)(=O)C1CC1</chem>	581.77	270
Ritonavir*	<chem>CC(C)c1nc(CN(C)C(=O)N[C@H](C(=O)N[C@@H](Cc2ccccc2)C[C@H](O)[C@H](Cc2ccccc2)NC(=O)OCc2cncs2)C(C)C)cs1</chem>	609.02	283
Simeprevir*	<chem>COc1ccc2c(O[C@H]3CC4C(=O)N(C)CCCC/C=C/[C@H]5C[C@@]5(C(=O)NS(=O)(=O)C5CC5)NC(=O)[C@@H]4C3)cc(-c3nc(C(C)C)cs3)nc2c1C</chem>	826.24	356

AI504: Programming for Artificial Intelligence

Week 13: Graph Neural Networks

Edward Choi

Grad School of AI

edwardchoi@kaist.ac.kr