vision_ai # Distill

A Gentle Introduction to Graph Neural Networks

2022-01-19 | JiHyun Lee

hutom

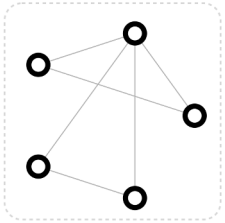
Content

1. What kind of data is most naturally phrased as a graph?
2. What makes graphs different from other types of data, and some of the specialized choices we have to make when using graphs
3. Build a modern GNN, walking through each of the parts of the model
    a. starting with historic modeling innovations in the field
4. GNN playground
    a. you can play around with a real-word task and dataset

Graph 란

- A graph represents the relations (***edges***) between a collection of entities (***nodes***).

Vertex (or node) attributes



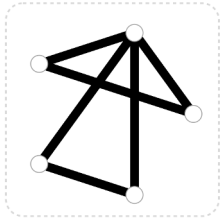
V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path

Three types of attributes we might find in a graph, hover over to highlight each attribute. Other types of graphs and attributes are explored in the Other types of graphs section.

Edge (or link) attributes and directions



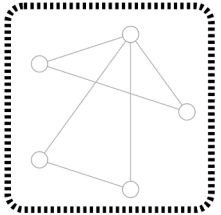
V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path

Three types of attributes we might find in a graph, hover over to highlight each attribute. Other types of graphs and attributes are explored in the Other types of graphs section.

Global (or master node) attributes



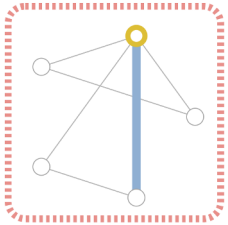
V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path

Three types of attributes we might find in a graph, hover over to highlight each attribute. Other types of graphs and attributes are explored in the Other types of graphs section.

- Each node, edge, entire graph can store information in each of these pieces of the graph

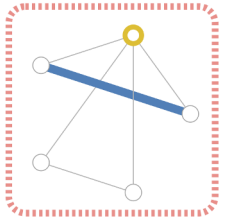


Vertex (or node) embedding

Edge (or link) attributes and embedding

Global (or master node) embedding

Information in the form of scalars or embeddings can be stored at each graph node (left) or edge (right).

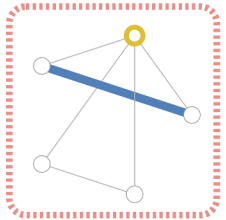


Vertex (or node) embedding

Edge (or link) attributes and embedding

Global (or master node) embedding

Information in the form of scalars or embeddings can be stored at each graph node (left) or edge (right).



Vertex (or node) embedding

Edge (or link) attributes and embedding

Global (or master node) embedding

Information in the form of scalars or embeddings can be stored at each graph node (left) or edge (right).

Graph 란

- Edge 종류에 따라 그래프를 구분할 수 있음
  - Undirected edge
    - no notion of source or destination nodes
  - Directed edge
    - (flow) source node -> destination node

The edges can be directed, where an edge e has a source node, v_{src}, and a destination node v_{dst}. In this case, information flows from v_{src} to v_{dst}. They can also be undirected, where there is no notion of source or destination nodes, and information flows both directions. Note that having a single undirected edge is equivalent to having one directed edge from v_{src} to v_{dst}, and another directed edge from v_{dst} to v_{src}.

Graphs and where to find them

Graph data

- Familiar with some types of graph data, such as social networks.
- **Two types of data that you might not think could be modeled as graphs**
  - **images and text.**

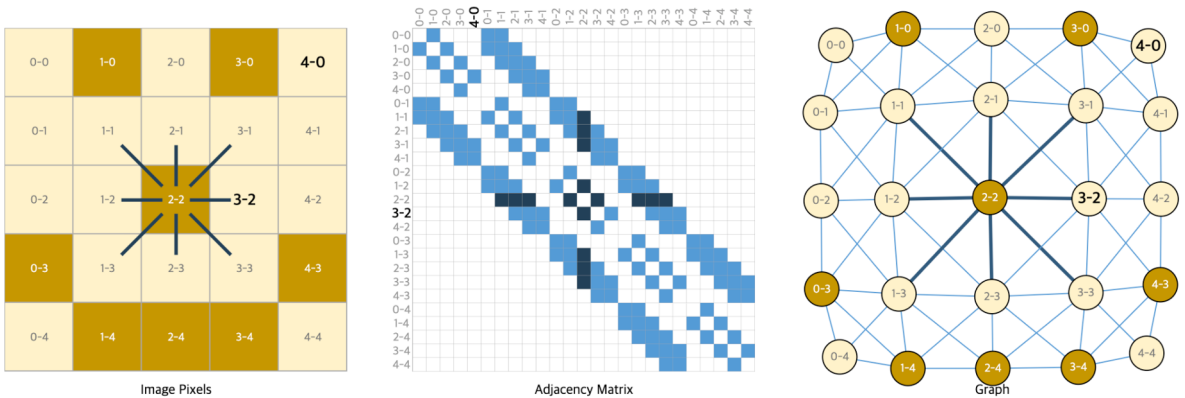
Images as graphs

- Each pixel represents a node and is connected via an edge to adjacent pixels.
  - 각각의 픽셀이 노드가 되고, 각 픽셀은 근접한 픽셀에 edge 로 연결되어 있음.
- 가장자리 외의 모든 픽셀은 8개의 이웃된 노드를 가지고 있으며, 각 노드에서는 3차원 벡터 information 을 포함 (the RGB value of the pixel)

5 *5 pixel image
웃는 얼굴 이미지

Adjacency matrix
두 노드가 엣지로 연결되어 있을 때 filled

Graph

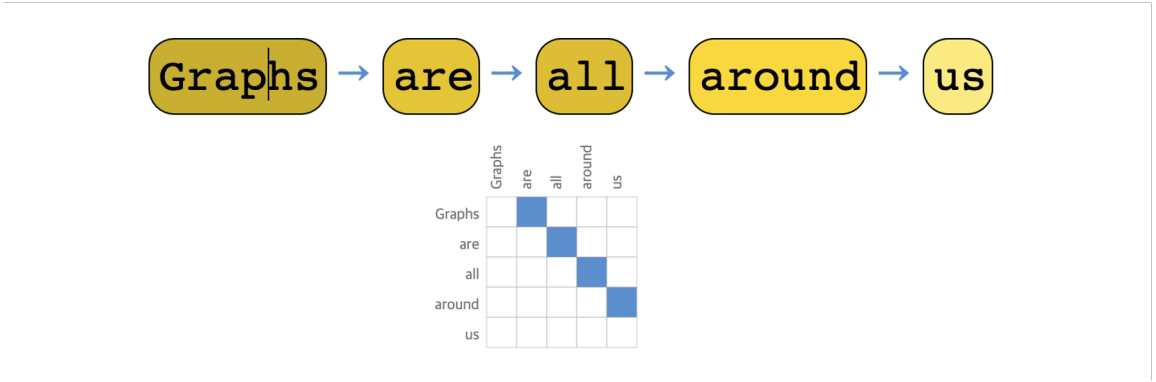


different views
of the same piece of data

Click on an image pixel to toggle its value, and see how the graph representation changes.

Text as graphs

- Text 를 index 의 sequence 로 표현함.
- simple directed graph
  - 각각의 단어 (character of index) 가 node, 각 node 는 다음 단어 (node) 를 follow



Graphs and where to find them

Graph-valued data in the wild

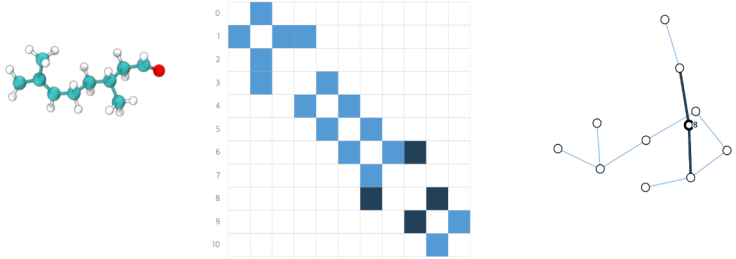
- Image 와 text 와는 다르게, 이웃된 노드의 수가 variable 함.

- 여기서 소개하는 데이터는 그래프 이외의 형태로 표현하기 어려움. (그래프로써 가장 잘 표현되는 데이터 종류)

1) **Molecules as graphs**
   node (atoms), edges (covalent bonds (공유 결합))

1) **Social networks as graphs**
   node (people by modelling individuals), edges (their relationships)

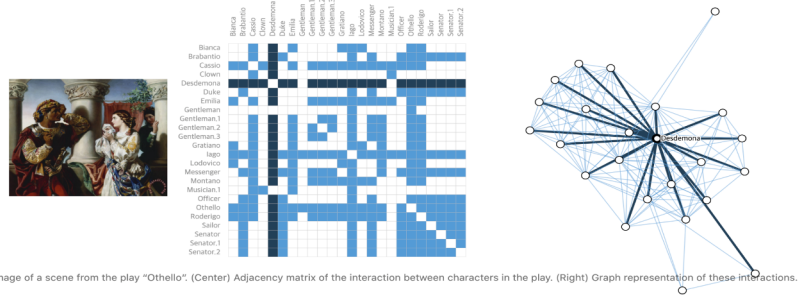
1) **Citation networks as graphs**
   node (each paper), edge (each directed)



(Left) 3d representation of the Citronellal molecule (Center) Adjacency matrix of the bonds in the molecule (Right) Graph representation of the molecule.

Adjacency matrix

Graph representation of the molecule



(Left) Image of a scene from the play "Othello", (Center) Adjacency matrix of the interaction between characters in the play. (Right) Graph representation of these interactions.

Adjacency matrix

Graph representation of these interactions

Graph-valued data in the wild

- Image 와 text 와는 다르게, 이웃된 노드의 수가 variable 함.

- 여기서 소개하는 데이터는 그래프 이외의 형태로 표현하기 어려움. (그래프로써 가장 잘 표현되는 데이터 종류)

  4) Other examples

  Machine learning models, programming code and math equations
  node (variables, objects), edge (operations that have these variables as input and output)

Graph dataset

					Edges per node (degree)		
Dataset	**Domain**	**graphs**	**nodes**	**edges**	**min**	**mean**	**max**
karate club	Social network	1	34	78		4.5	17
qm9	Small molecules	134k	≤ 9	≤26	1	2	5
Cora	Citation network	1	23,166	91,500	1	7.8	379
Wikipedia links, English	Knowledge graph	1	12M	378M		62.24	1M

Summary statistics on graphs found in the real world. Numbers are dependent on featurization decisions. More useful statistics and graphs can be found in KONECT [14]

What types of problems have graph structured data?

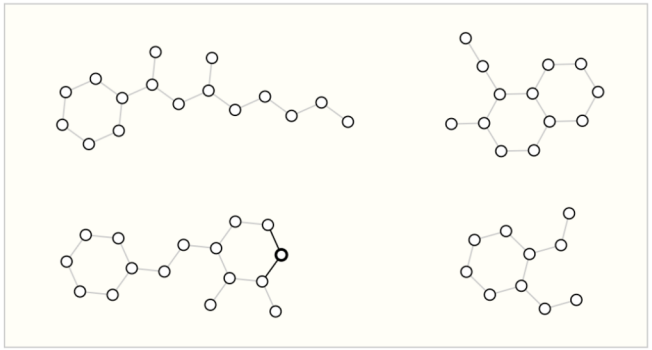
이러한 그래프를 가지고, 실제 어떤 task 를 해결할 수 있을까?

General types of prediction tasks on graphs: graph-level, node-level, edge-level

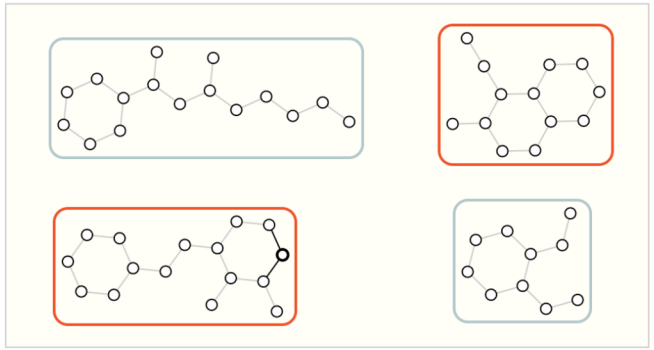
Graph-level task

- 그래프 전체에 대한 하나의 속성 예측 predict the property of an entire graph)
  - e.g. 어떤 분자가 냄새가 나는가? 그래프가 두 개의 rings 를 포함하고 있는가?

- image classification with MNIST and CIFAR → 하나의 이미지에 대한 레이블 예측
- Sentiment analysis (감정 분석) → 하나의 문장에 대한 감정 예측



Input: graphs



Output: labels for each graph, (e.g., "does the graph contain two rings?")

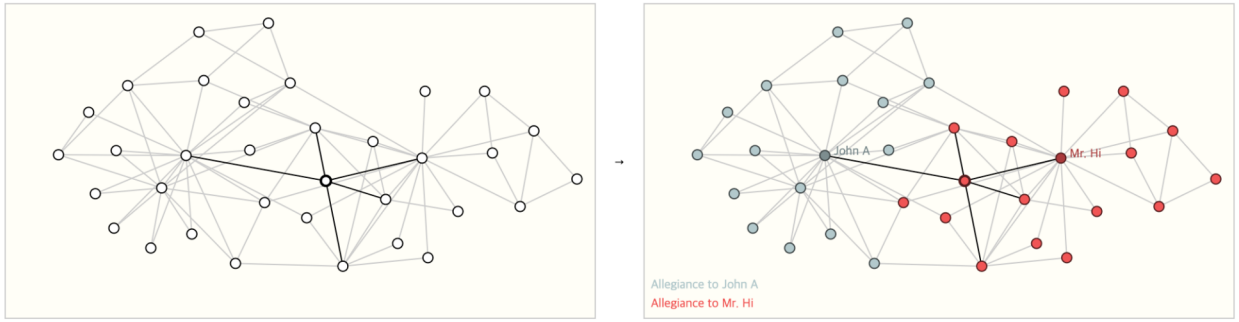
What types of problems have graph structured data?

Node-level task

- 그래프 내부의 각 노드의 identity or role 예측하는 문제
- Classic example of node-level prediction problem is Zach's karate club.
    - Mr.Hi (Instructor) 와 John H (Administrator) 사이의 불화로 인해, karate 수강생들이 어떤 social network 를 형성하였는지 파악하는 문제, 어느 강사에게 의리를 지켰나..!

- 이 경우, 노드 간 거리가 먼 것은 label 생성에 직접적인 연관이 있음.

- Image segmentation → 이미지의 픽셀 당 label
- Text → predicting the parts-of-speech of each word in a sentence (e.g. noun, verb, adverb, etc)



Input: graph with unlabled nodes

Output: graph node labels

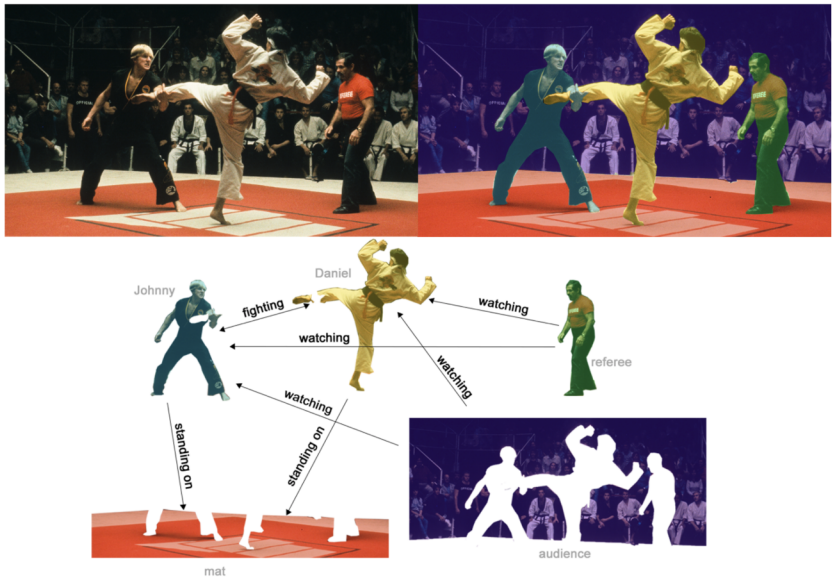
On the left we have the initial conditions of the problem, on the right we have a possible solution, where each node has been classified based on the alliance. The dataset can be used in other graph problems like unsupervised learning.

What types of problems have graph structured data?

Edge-level task

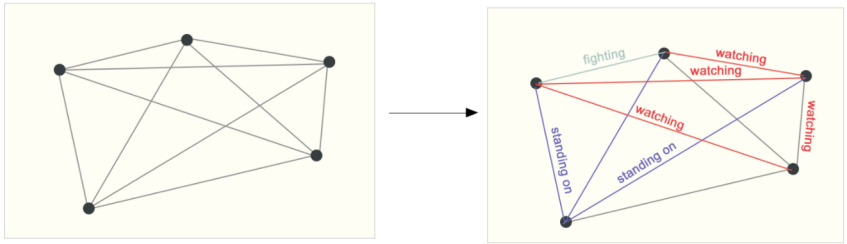
- Edge prediction 은 여전히 풀어야 할 문제로 남아있음.
- One example of edge-level inference is in image scene understanding.
  - 딥러닝은 이미지에서 object를 구별하는 것을 넘어, object 간 relationship 을 예측할 수 있음. → edge-level classification
  - 이미지의 object 를 node 로 표현하고, 각 노드 간 관계를 edge 로 연결할 수 있음.

5개의 entities



In (b), above, the original image (a) has been segmented into five entities: each of the fighters, the referee, the audience and the mat. (C) shows the relationships between these entities.

model output 에 따라 일부 connection 은 prune (삭제) 될 수 있음. (e.g. audience - mat)



Input: fully connected graph, unlabeled edges

Output: labels for edges

On the left we have an initial graph built from the previous visual scene. On the right is a possible edge-labeling of this graph when some connections were pruned based on the model's output.

What types of problems have graph structured data?

Edge-level task

- CNN에 기반을 둔 많은 방법들이 이미지에서 물체를 탐지하는데, 최첨단 성능을 달성했지만, 그 물체들 사이의 관계까지는 알지 못한다.
  CNN 으로 탐지된 물체들을 아래 그림(왼쪽)의 방법을 통해 scene graph 를 만들어서 관계를 파악할 수 있다.

- 또한 기존 이미지 생성 방법은 GAN 이나 AutoEncoder 를 사용하였는데,
  아래 그림(오른쪽)의 방법을 사용하면 scene graph 로부터 이미지를 생성할 수 있다.

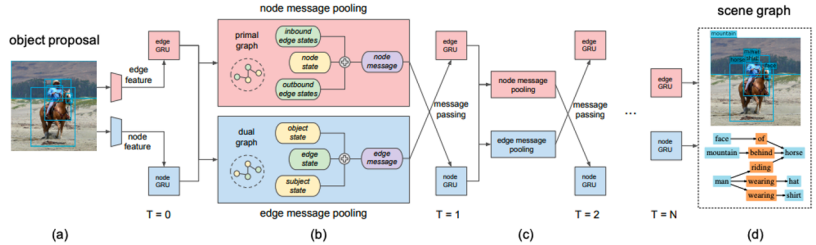


Figure 3. An illustration of our model architecture (Sec. 3). The model first extracts visual features of nodes and edges from a set of object proposals, and edge GRUs and node GRUs then take the visual features as initial input and produce a set of hidden states (a). Then a node message pooling function computes messages that are passed to the node GRU in the next iteration from the hidden states. Similarly, an edge message pooling function computes messages and feed to the edge GRU (b). The ⊕ symbol denotes a learnt weighted sum. The model iteratively updates the hidden states of the GRUs (c). At the last iteration step, the hidden states of the GRUs are used to predict object categories, bounding box offsets, and relationship types (d).

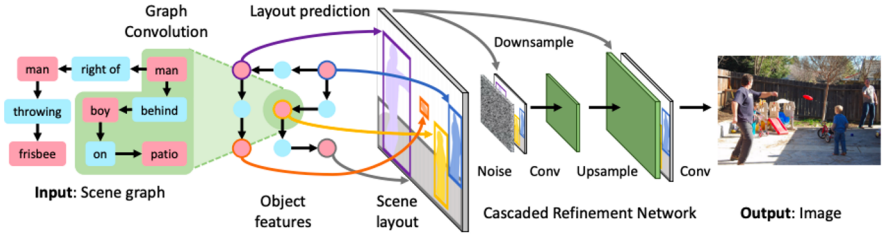


Figure 2. Overview of our image generation network f for generating images from scene graphs. The input to the model is a *scene graph* specifying objects and relationships; it is processed with a *graph convolution network* (Figure 3) which passes information along edges to compute embedding vectors for all objects. These vectors are used to predict bounding boxes and segmentation masks for objects, which are combined to form a *scene layout* (Figure 4). The layout is converted to an image using a *cascaded refinement network* (CRN) [6]. The model is trained adversarially against a pair of *discriminator networks*. During training the model observes ground-truth object bounding boxes and (optionally) segmentation masks, but these are predicted by the model at test-time.

The challenges of using graphs in machine learning

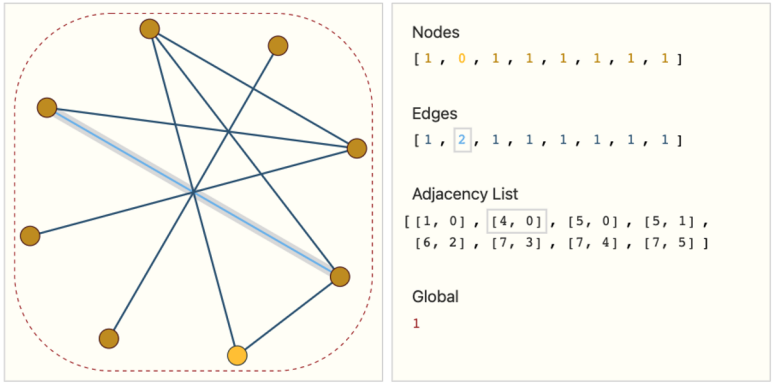
이러한 그래프 데이터를 Neural networks 를 가지고 문제를 어떻게 해결할 것인가?

First step : Think about how we will represent graphs to be compatible with neural networks.

---

1. **우리는 그래프를 예측에 사용하는데 최대 4개까지의 information 을 활용함.**
   a. [feature matrix] nodes, edges, global-context and [Adjacency matrix] connectivity
      i. nodes, edge, global-context 의 경우 직관적으로 표현 가능.
      - feature matrix N by index i and storing the feature for node in N

      i. **Connectivity 의 경우, adjacency matrix 로 표현이 가능**
      - **하나의 그래프에 노드가 너무 많거나, edge 가 너무 많으면 sparse adjacency matrices 가 될 가능성이 있다. ⇒ space-inefficient**

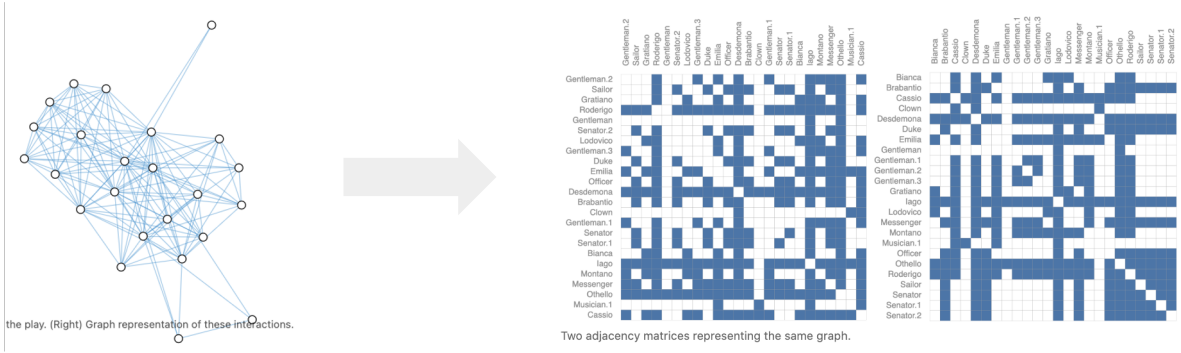


Adjacency lists

- sparse matrices 를 memory-efficient 하게 representing
- 노드와 노드 간의 edge 를 tuple (i, j) 로 표현. → k 개의 entity adjacency list

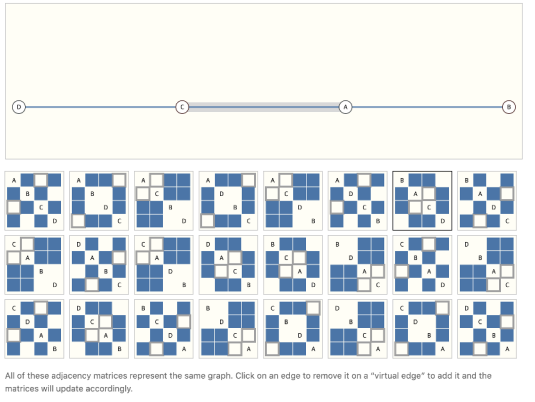
2. Many adjacency matrices that can encode the same connectivity

a. 하나의 Othello graph 는 두 개의 서로 다른 adjacency matrics 로 표현 가능.



the play. (Right) Graph representation of these interactions.

Two adjacency matrices representing the same graph.



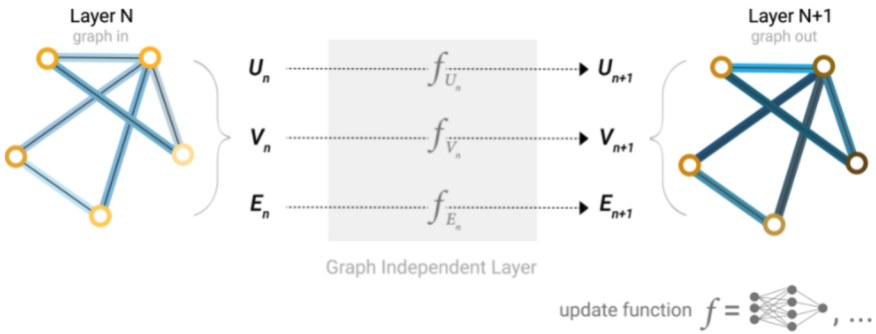
All of these adjacency matrices represent the same graph. Click on an edge to remove it on a "virtual edge" to add it and the matrices will update accordingly.

Using Graph Neural Networks (GNNs) to solve graph prediction tasks. (2009)

1. **A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances).**
   - GNN은 graph symmetries 보존한 상태로 그래프의 모든 속성 (nodes, edges, global-context) 에 대해 optimizable transformation 수행.

1. **GNN adopt a "graph-in, graph-out" architecture**
   - (노드에 각 information 을 load 한 채로) graph 를 input 으로 입력하면, input graph 의 connectivity 의 변화 없이, 노드, 엣지, global-context 의 embedding 값이 progressively transform.

The simplest GNN



A single layer of a simple GNN. A graph is the input, and each component (V,E,U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n-th layer of a GNN model.

GNN layer

- 각각의 컴포넌트 (노드, 엣지, 글로벌) 에 대해 분리된 MLP 사용
- 각각의 node vector 에 대해, apply MLP and get back a learned node-vector.
- 각각의 edge vector 에 대해, learning a per-edge embedding
- Also for the global-context vector, learning a single embedding for the entire graph.

GNN 이 인풋 그래프의 connectivity 를 업데이트 하지는 않기 때문에, output 그래프는 동일한 adjacency list, 동일한 수의 feature vector 로 표현 가능.
그러나, output 그래프는 각각의 노드, 엣지, global-context 의 embedding 이 업데이트!!

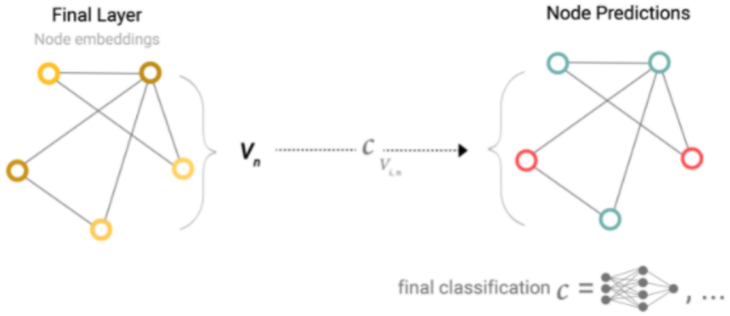
GNN Predictions by Pooling Information

How do we make predictions in any of the tasks?

Consider the case of binary classification (easily can be extended to the multi-class or regression case)

Case 1

- If the task is to make binary predictions on nodes, and the graph already contains node information → **각각의 node embedding 에 대해 linear classifier 적용.**



GNN Predictions by Pooling Information

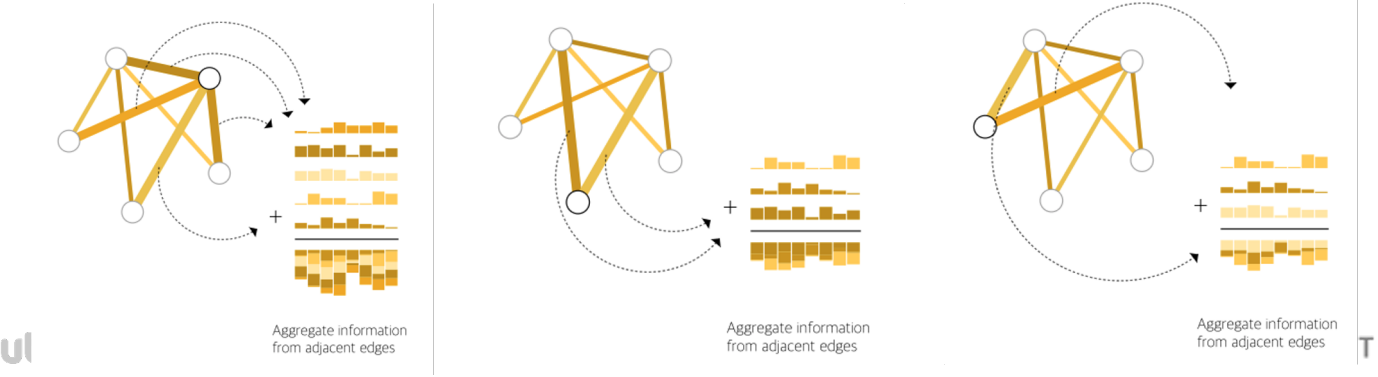
Case 2

- 그래프의 edges 대한 information 을 가지고 있는데, node 에 대한 prediction 을 수행하고 싶은 경우.
- **We need a way to collect information from edges and give them to nodes for prediction**
  - 우리는 edge 로부터 information 을 수집하여, 이를 node prediction 에 사용해야 한다. ⇒ **이걸 pooling 으로 수행함.**

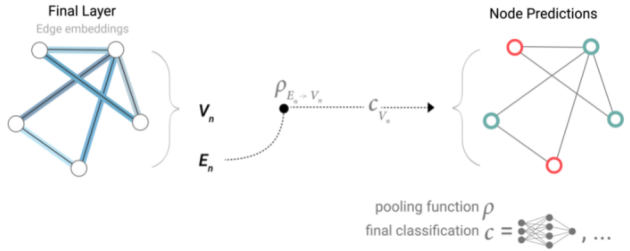
Pooling proceeds in two steps :

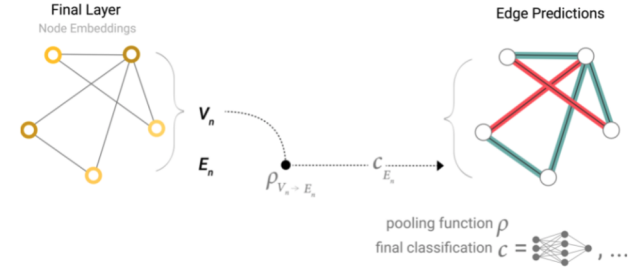
1. 각각의 item 을 pooled 하기 위해, 각 embedding 을 gather (모으고), concatenate (합친다) them into a matrix.
2. gathered embeddings (합쳐진 embedding) 을 aggregated (일반적으로 sum operation)

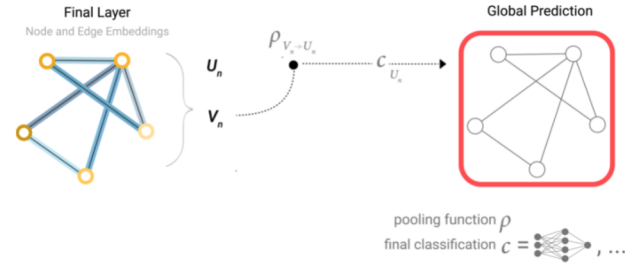
We represent the *pooling* operation by the letter ρ, and denote that we are gathering information from edges to nodes as $p_{E_n \to V_n}$.



GNN Predictions by Pooling Information







Use pooling to route (or pass) information to where it needs to go.

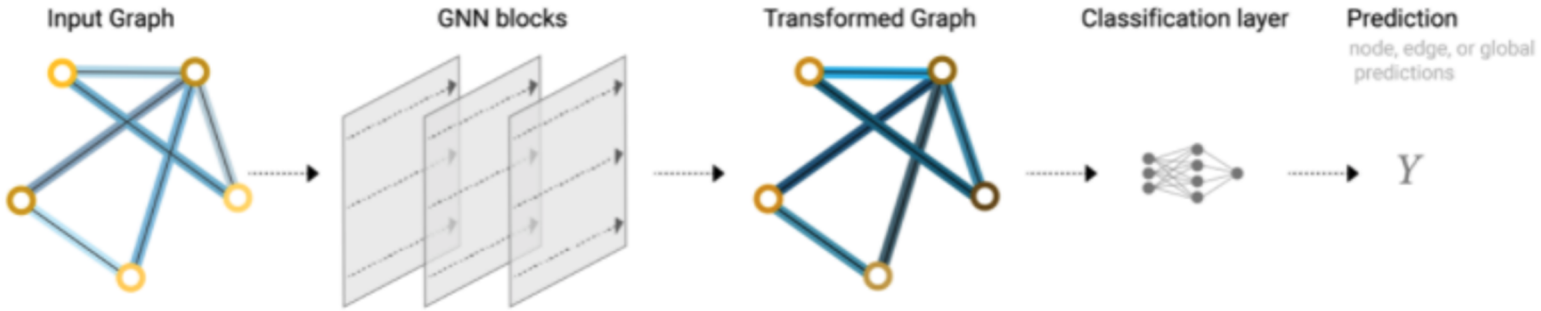
- Edge level features → Node level features

- Node level features → Edge level features
  - Edge level task : image entities (node) relationship (binary edge) 예측.

- Node level features → Global property
  - 사용 가능한 모든 node information 을 gather 한 후, aggregate.
  - CNN 의 Global Average Pooling layer 와 유사한 형태.
  - e.g 분자 속성 예측
- Edge level features 에 대해서도 동일하게 작동.

GNN Predictions by Pooling Information

End-to-end prediction task with a GNN model



Pooling technique 을 통하여, GNN models 을 좀 더 sophisticated 하게 constructing 함.
⇒ 새로운 attribute 를 얻었을 때, 다른 attribute 로 information 을 어떻게 pass 할 것인지만 정의하면 됨.

그러나, 아직 GNN layer 내부의 connectivity 를 사용하지 않았음 → Passing messages
⇒ 각각의 노드, 엣지, global context 는 독립적으로 processed (가공됨).
⇒ 현재까지는 prediction 을 위해서 information pooling 할 때만 connectivity 사용.

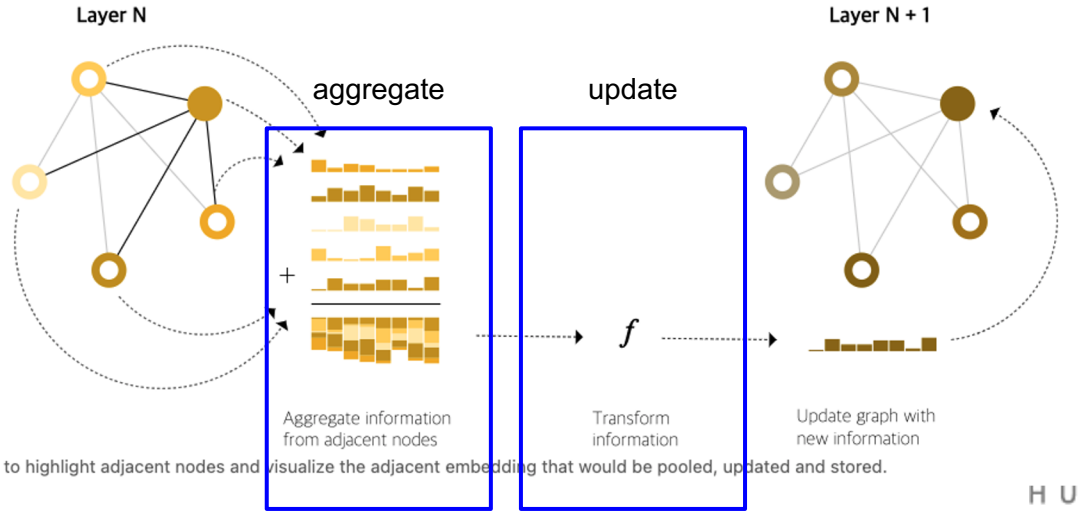
Passing messages between parts of the graph

- Neighboring nodes and edges exchange information and influence each other's updated embeddings.
  - 이웃된 노드와 엣지가 각각 embedding 을 업데이트 할 때, information exchange 와 영향을 줌.

Message passing works in three steps :

1. **[gather messages]** 그래프의 각각의 노드는 g function 을 통하여, 이웃된 노드 embeddings 을 gather
2. **[aggregate them]** Aggregate function (e.g. sum) 을 통하여, 모든 messages Aggregate
3. **[update them]** All pooled messages 는 update function 을 통과 (일반적으로 학습된 neural network).

message passing 역시 nodes 와 edge 간에 적용될 수 있음.

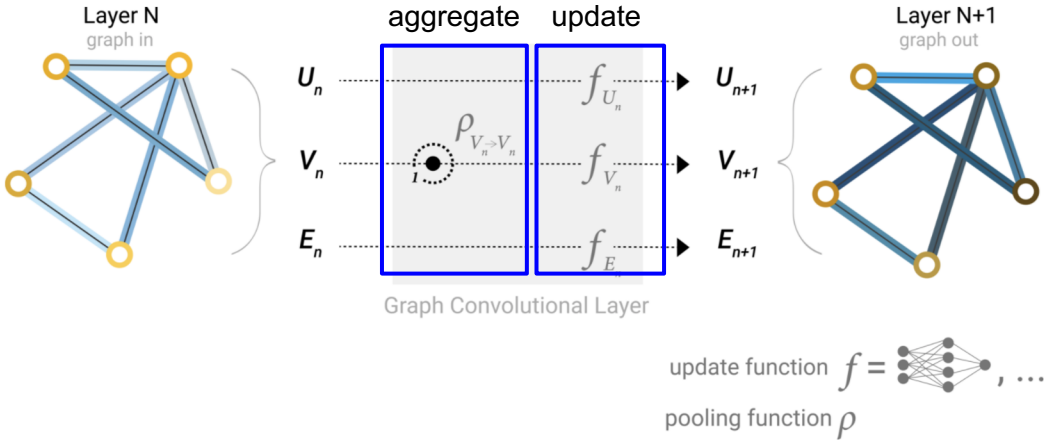


Passing messages between parts of the graph

- Neighboring nodes and edges exchange information and influence each other's updated embeddings.
  - 이웃된 노드와 엣지가 각각 embedding 을 업데이트 할 때, information exchange 와 영향을 줌.

Message passing works in three steps :

1. **[gather messages]** 그래프의 각각의 노드는 g function 을 통하여, 이웃된 노드 embeddings 을 gather
2. **[aggregate them]** Aggregate function (e.g. sum) 을 통하여, 모든 messages Aggregate
3. **[update them]** All pooled messages 는 update function 을 통과 (일반적으로 학습된 neural network).



Schematic for a GCN architecture, which updates node representations of a graph by pooling neighboring nodes at a distance of one degree.

**GNN layer 를 쌓음으로써,
하나의 노드는 전체 그래프에 대한
information 을 포함할 수 있다.**

Passing messages between parts of the graph

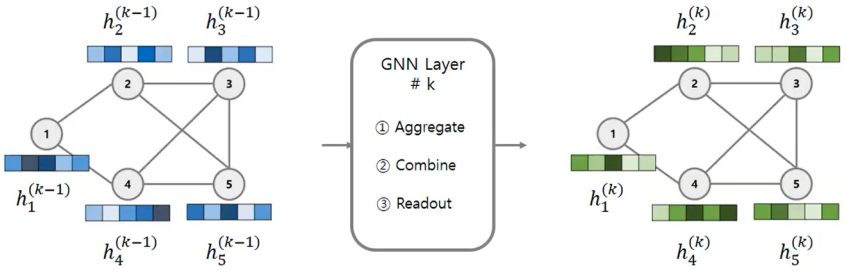
Message passing works in three steps :

1. **[gather messages]** 그래프의 각각의 노드는 g function 을 통하여, 이웃된 노드 embeddings 을 gather
2. **[aggregate them]** Aggregate function (e.g. sum) 을 통하여, 모든 messages Aggregate
3. **[update them]** All pooled messages 는 update function 을 통과 (일반적으로 학습된 neural network).

- GNN Notation
  - $h_v^{(k)}$: hidden embedding node v at kth GNN layer
  - $v = target\ node$
  - $N(v) = neightbor\ nodes\ of\ v$
  - $u = neightbor\ node \in N(v)$

$$a_v^{(k-1)} = aggregate^k(\{h_u^{(k-1)}, \forall u \in N(v)\})$$

$$h_v^{(k)} = combine^k(a_u^{(k-1)}, h_v^{(k-1)})$$



$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

trainable matrices (i.e., what we learn)

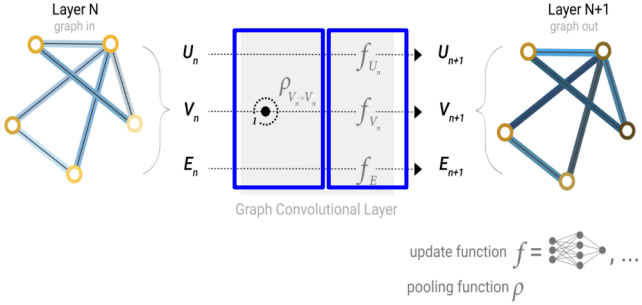
$$\mathbf{z}_v = \mathbf{h}_v^K$$

After K-layers of neighborhood aggregation, we get output embeddings for each node.

Graph Neural Networks

Message passing works in three steps :

1. **[gather messages]**
2. **[aggregate them]**
3. **[update them]**



Schematic for a GCN architecture, which updates node representations of a graph by pooling neighboring nodes at a distance of one degree.

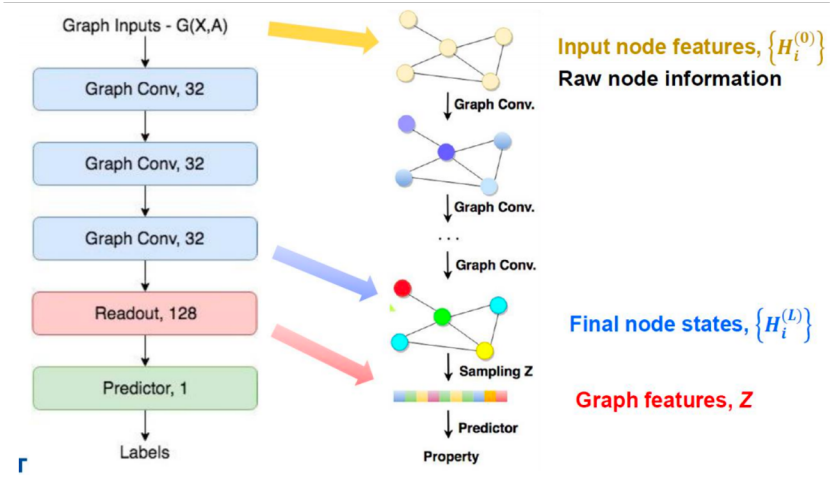
aggregation function used in pooling: max, mean or sum

- GNN
  - **Aggregate, Combine function의 정의에 따라** 다양한 방식의 모델이 존재함.

TABLE 2
Different variants of graph neural networks.

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	Neural FPs	$\mathbf{h}_{\mathcal{N}_v}^t = \mathbf{h}_v^{t-1} + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}_k^{t-1}$	$\mathbf{h}_v^t = \sigma(\mathbf{h}_{\mathcal{N}_v}^t \mathbf{W}_L^{\mathcal{N}_v})$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1_N^T \mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f(\mathbf{W}^c \odot \mathbf{N})$
	GraphSAGE	$\mathbf{h}_{\mathcal{N}_v}^t = \text{AGGREGATE}_t(\{\mathbf{h}_u^{t-1}, \forall u \in \mathcal{N}_v\})$	$\mathbf{h}_v^t = \sigma(\mathbf{W}^t \cdot [\mathbf{h}_v^{t-1} \| \mathbf{h}_{\mathcal{N}_v}^t])$
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_k]))}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}_{\mathcal{N}_v}^t = \sigma\left(\sum_{k\in\mathcal{N}_v} \alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}_{\mathcal{N}_v}^t = \Big\|_{m=1}^{M} \sigma\left(\sum_{k\in\mathcal{N}_v} \alpha_{vk}^m \mathbf{W}^m \mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}_{\mathcal{N}_v}^t = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v} \alpha_{vk}^m \mathbf{W}^m \mathbf{h}_k\right)$	$\mathbf{h}_v^t = \mathbf{h}_{\mathcal{N}_v}^t$
Gated Graph Neural Networks	GGNN	$\mathbf{h}_{\mathcal{N}_v}^t = \sum_{k\in\mathcal{N}_v} \mathbf{h}_k^{t-1} + \mathbf{b}$	$\mathbf{z}_v^t = \sigma(\mathbf{W}^z\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}^z\mathbf{h}_v^{t-1})$ $\mathbf{r}_v^t = \sigma(\mathbf{W}^r\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}^r\mathbf{h}_v^{t-1})$ $\tilde{\mathbf{h}}_v^t = \tanh(\mathbf{W}\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}))$ $\mathbf{h}_v^t = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \tilde{\mathbf{h}}_v^t$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}_{\mathcal{N}_v}^t = \sum_{k\in\mathcal{N}_v} \mathbf{h}_k^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i\mathbf{x}_v^t + \mathbf{U}^i\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma(\mathbf{W}^f\mathbf{x}_v^t + \mathbf{U}^f\mathbf{h}_k^{t-1} + \mathbf{b}^f)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o\mathbf{x}_v^t + \mathbf{U}^o\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u\mathbf{x}_v^t + \mathbf{U}^u\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{k\in\mathcal{N}_v} \mathbf{f}_{vk}^t \odot \mathbf{c}_k^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t)$
	Tree LSTM (N-ary)	$\mathbf{h}_{\mathcal{N}_v}^{ti} = \sum_{l=1}^{K} \mathbf{U}_l^i \mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v k}^{tf} = \sum_{l=1}^{K} \mathbf{U}_{kl}^f \mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{to} = \sum_{l=1}^{K} \mathbf{U}_l^o \mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{tu} = \sum_{l=1}^{K} \mathbf{U}_l^u \mathbf{h}_{vl}^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{ti} + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma(\mathbf{W}^f\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v k}^{tf} + \mathbf{b}^f)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{to} + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{tu} + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{l=1}^{K} \mathbf{f}_{vl}^t \odot \mathbf{c}_{vl}^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t)$
	Graph LSTM in [44]	$\mathbf{h}_{\mathcal{N}_v}^{ti} = \sum_{k\in\mathcal{N}_v} \mathbf{U}_{m(v,k)}^i \mathbf{h}_k^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{to} = \sum_{k\in\mathcal{N}_v} \mathbf{U}_{m(v,k)}^o \mathbf{h}_k^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{tu} = \sum_{k\in\mathcal{N}_v} \mathbf{U}_{m(v,k)}^u \mathbf{h}_k^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{ti} + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma(\mathbf{W}^f\mathbf{x}_v^t + \mathbf{U}_{m(v,k)}^f \mathbf{h}_k^{t-1} + \mathbf{b}^f)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{to} + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{tu} + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t \odot \mathbf{u}_v^t + \sum_{k\in\mathcal{N}_v} \mathbf{f}_{vk}^t \odot \mathbf{c}_k^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t \odot \tanh(\mathbf{c}_v^t)$

- **GCN (Graph Convolutional Networks)**
  - 가장 기본적인 GNN 모델
  - Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)
    (1) 자기 자신과 이웃 노드에 대해서 동일한 파라미터를 사용하고,
    (2) neighbor 들에 대해서 normalization 을 적용하여 차등적으로 이웃의 feature 들을 반영



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

GCN (Graph Convolutional Networks)

1. Node Feature Matrix 인 X와 Adjacency Matrix 인 A 를 갖는 graph 를 input 으로 하였을 때,

1. 원하는 만큼의 convolution 연산을 거치고 (중간 중간 activation)

1. Readout 을 적용한 후, fully connected layer 를 거침.

readout : graph convolution layer 를 통해 생성된 latent feature matrix 를 그래프 전체를 표현하는 하나의 벡터로 변환하는 함수.

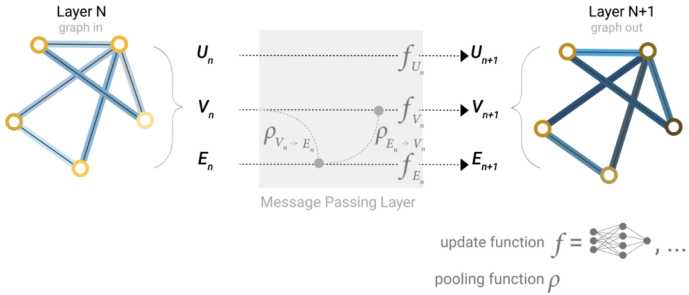
Learning edge representations (design 요소)

그래프에 저장된 node 와 edge 는 동일한 size 또는 동일한 shape 이 아닐 수 있음.
이를 해결하기 위해,

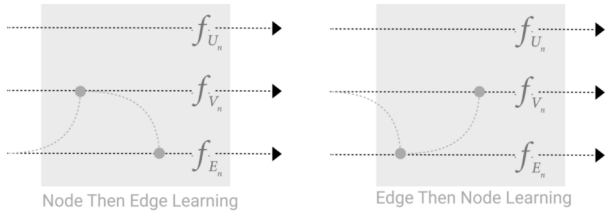
1. edge space 에서 node space 로의 linear mapping 을 학습하거나,
2. update function 을 통과하기 전에 하나의 요소가 다른 하나를 concatenate 하는 방법이 있다.

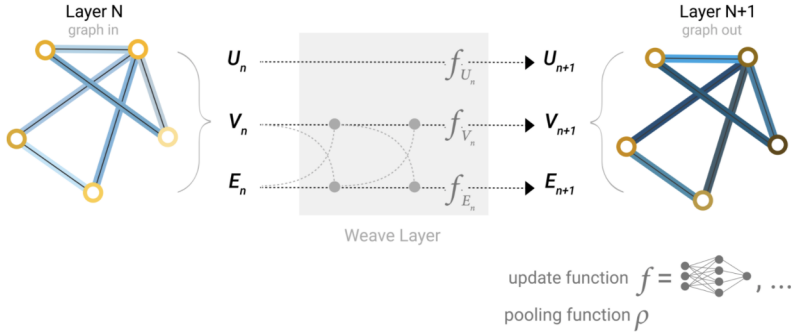
어떤 attributes 를 업데이트 할 것이냐, 또는 어떤 순서로 업데이트 할 것이냐는 GNN 을 구축하는 하나의 design 요소.

- 노드 임베딩을 먼저 업데이트 하고, 엣지 임베딩을 업데이트 할 수 있고 반대도 가능함.
- 'weave' fashion : 새로운 노드, 엣지 representations 생성을 위한 4 가지 updated representations 방법
  - node to node (linear), edge to edge (linear), node to edge (edge layer), edge to node (node layer)



Architecture schematic for Message Passing layer. The first step "prepares" a message composed of information from an edge and it's connected nodes and then "passes" the message to the node.





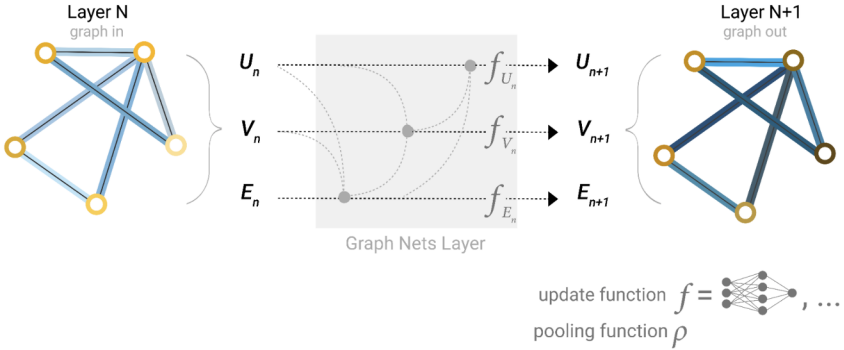
Some of the different ways we might combine edge and node representation in a GNN layer.

Adding global representations

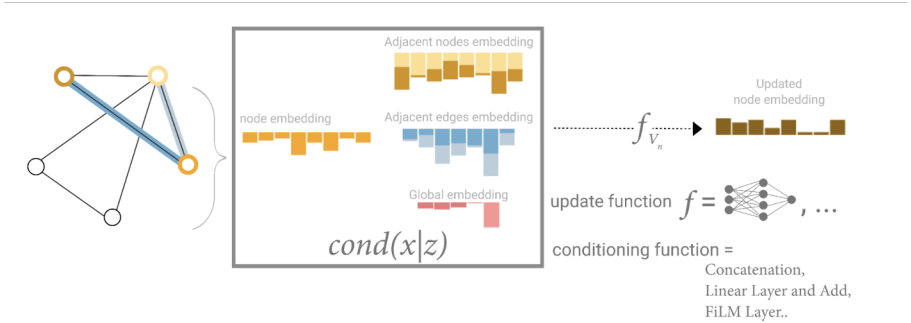
멀리 떨어져 있는 노드에 의존하여 prediction 을 수행할 때는, 효과적으로 transfer information 을 수행하지 못할 수 있다.

⇒ **master node (context vector)**

- global representation of a graph
- global context vector 는 모든 다른 노드와 엣지에 연결되어 있으며, pass information 의 bridge 역할을 하여, 그래프 전체에 대한 representation 을 나타냄.
- 이 관점에서, 모든 그래프의 속성이 학습된 representation 을 가지고 있으므로, 나머지 부분들 중에서 관심 attribute 를 조절하여 pooling 에 활용할 수 있다.



Schematic of a Graph Nets architecture leveraging global representations.



Schematic for conditioning the information of one node based on three other embeddings (adjacent nodes, adjacent edges, global). This step corresponds to the node operations in the Graph Nets Layer.

How these different components and architectures contribute to a GNN's ability to learn a real task?

Prediction task
- graph-level prediction task with small molecular graphs.
- molecular structure (graph)에 따라 'pungent' 냄새가 나는지 안 나는지. (binary classification)

Dataset
- Leffingwell Odor Dataset

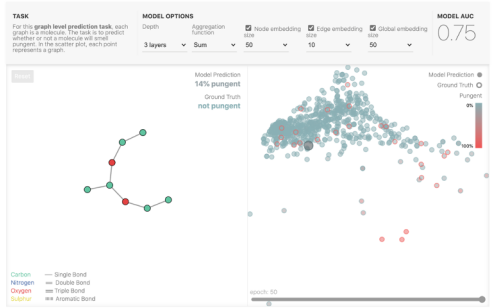
Graph 구조
- graph : 각각의 molecule
- nodes : atoms (containing a one-hot encoding for its atomic identity, Carbon, Nitrogen, Oxygen, Fluorine)
- edges : bonds (containing a one-hot encoding its bond type, Single, Double, Triple or Aromatic)

Activation function
- sigmoid activation for classification
- The update function is a 1-layer MLP with a relu activation function and a layer norm for normalization of activation
- The aggregation function used in pooling: max, mean or sum

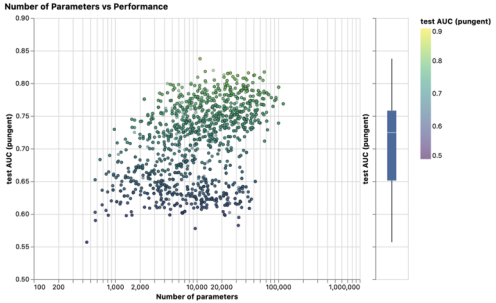
How these different components and architectures contribute to a GNN's ability to learn a real task?

- GNN playground

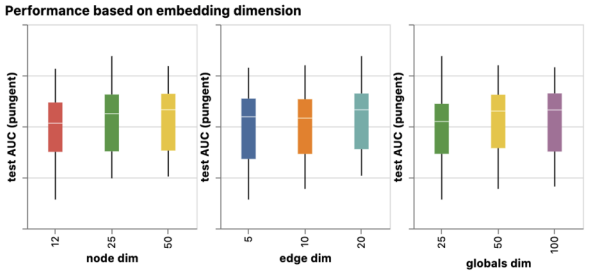
Some empirical GNN design lessons (다양한 design choice)



Edit the molecule to see how the prediction changes, or change the model params to load a different model. Select a different molecule in the scatter plot.



Scatterplot of each model's performance vs its number of trainable variables. Hover over a point to see the GNN architecture parameters.



Aggregate performance of models across varying node, edge, and global dimensions.

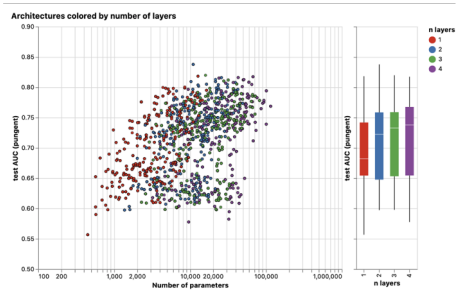


Chart of number of layers vs model performance, and scatterplot of model performance vs number of parameters. Each point is colored by the number of layers. Hover over a point to see the GNN architecture parameters.

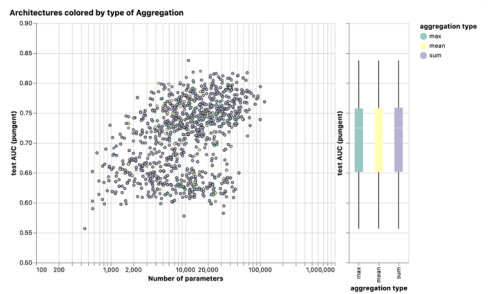


Chart of aggregation type vs model performance, and scatterplot of model performance vs number of parameters. Each point is colored by aggregation type. Hover over a point to see the GNN architecture parameters.

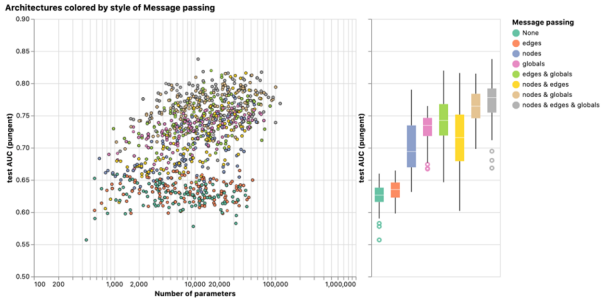


Chart of message passing vs model performance, and scatterplot of model performance vs number of parameters. Each point is colored by message passing. Hover over a point to see the GNN architecture parameters

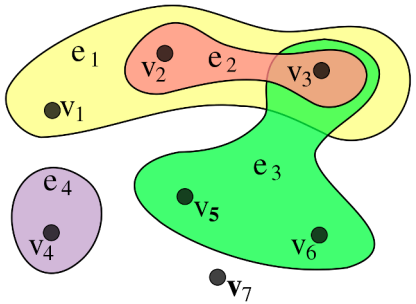
Other types of graphs (multigraphs, hypergraphs, hypernodes, hierarchical graphs)

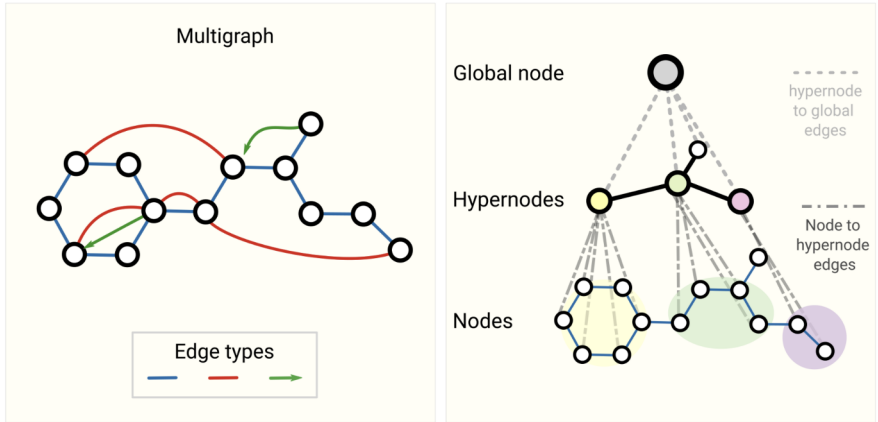
graph 구조는 다른 타입의 information 까지 수용 가능함.

1. **multigraphs**
   a. 한 쌍의 node 가 여러 타입의 edge 를 공유하는 경우.

1. **hyper-nodes, hierarchical graphs (nested graph)**
   a. hyper nodes: sub-graph, (즉, 노드 간 관계를 포함하고 있는 노드)
   b. node represents a molecule and edge is shared between two molecules if we have a way (reaction) of transforming one to the other.
   c. hierarchical information 을 표현하는데 유용함.

1. **hyper-graphs**
   a. edge 가 여러 개의 node 를 연결짓는 경우
   b. e.g. node 들의 communities 정의





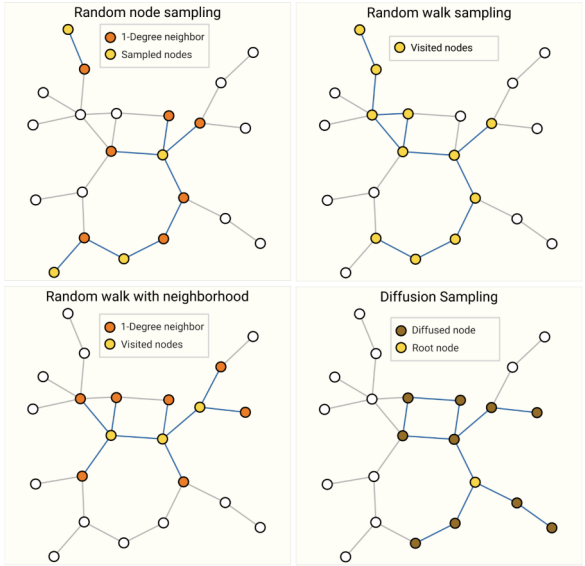
Schematic of more complex graphs. On the left we have an example of a multigraph with three edge types, including a directed edge. On the right we have a three-level hierarchical graph, the intermediate level nodes are hypernodes.

Sampling Graphs and Batching in GNNs

그래프 상 batch 는 서로 연결되어 있는 노드, 엣지의 수가 variability 하기 때문에, 고정된 batch size 를 사용할 수 없다.
The main idea for batching with graphs is to create subgraphs that preserve essential properties of the larger graph.

- **그래프의 원래 속성을 보존하는 subgraph 를 생성하여 batch 로 사용!**

⇒ 이러한 연산은 citation network 등에서는 적절할 수 있으나, molecules 등의 연산에서는 적절하지 않을 수 있다.
⇒ How to sample a graph is an open research question.



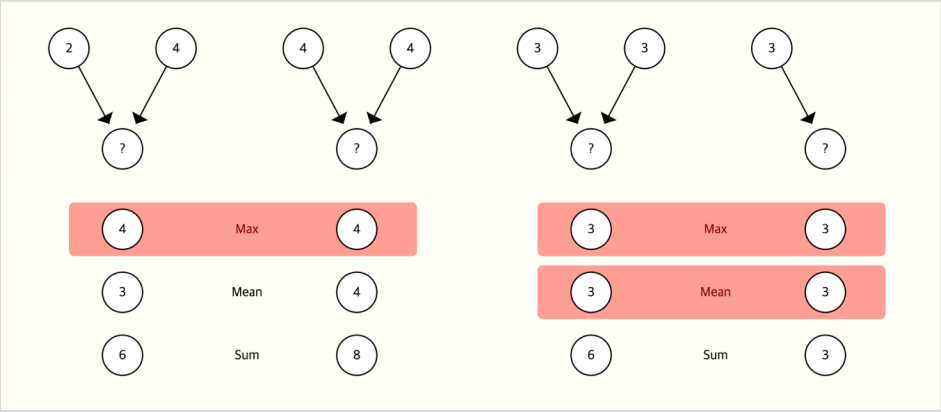
Four different ways of sampling the same graph. Choice of sampling strategy depends highly on context since they will generate different distributions of graph statistics (# nodes, #edges, etc.). For highly connected graphs, edges can be also subsampled.

동일한 그래프에서의 4가지 종류의 sampling 기법

- Sampling 기법은 그래프의 context 에 따라 선택할 수 있다.
  - 각 기법에 따라 subset-graph 가 달라지고, 다른 graph statistics 분포를 가짐.
- 또한 graph sampling 은 그래프가 memory 에 올라가지 않을 정도로 큰 경우에 유용하게 사용됨.

Comparing aggregation operations

이웃된 노드의 정보 (information) 을 pooling 할 때, 노드의 순서가 노드의 수에 invariant 한 smooth aggregation operation 이여야 한다.
또한, 유사한 input 그래프에 대해 유사한 aggregated output 생성 되는 것이 바람직함.
가장 간단한 permutation-invariant operations : sum, mean, max



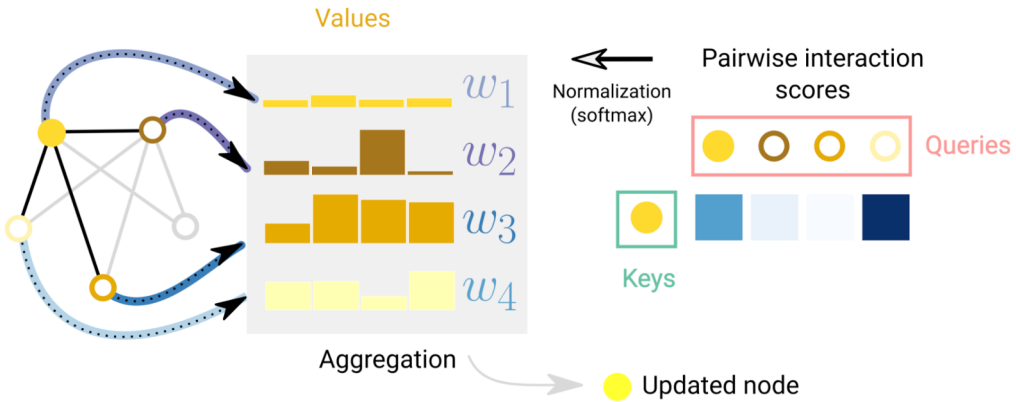
No pooling type can always distinguish between graph pairs such as max pooling on the left and sum / mean pooling on the right.

Graph Attention Networks

graph attributes 간 information communicating 을 attention 을 통해 수행할 수 있음.

- 기본적인 attention 개념과 동일하게, center node 와 neighboring node 간의 relevant 를 측정하여 score 로 표현.
  - graph 의 중요 노드에 가중치를 부여하여 구조를 학습하는 딥러닝 모델
  - 이웃 노드에 대한 중요도를 나타낸 attention weights 추가

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



Schematic of attention over one node with respect to it's adjacent nodes. For each edge an interaction score is computed, normalized and used to weight node embeddings.

Final thoughts

Graph 는 이미지, text 를 포함하여 다양한 형태를 데이터를 표현하는 데 좋은 데이터 타입임.

본 article 을 통해,

- Graph 를 대상으로 하는 neural network 를 구축하는 방법
- GNN 아키텍처를 생성하는데 중요한 design choice
- GNN playground 를 알아 봄.

최근 성공적인 GNN 연구는 새로운 문제를 해결하는데 좋은 기회가 되었으며, GNN field 는 앞으로의 연구가 기대되는 분야임.

This article is one of two Distill publications about graph neural networks.
Take a look at Understanding Convolutions on Graphs to understand how convolutions over images generalize naturally to convolutions over graphs.

End of the Document