10-418/10-618 Machine Learning for Structured Data

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Graph Neural Networks

+

Final Exam Review

Matt Gormley
Lecture 26
Dec. 9, 2022

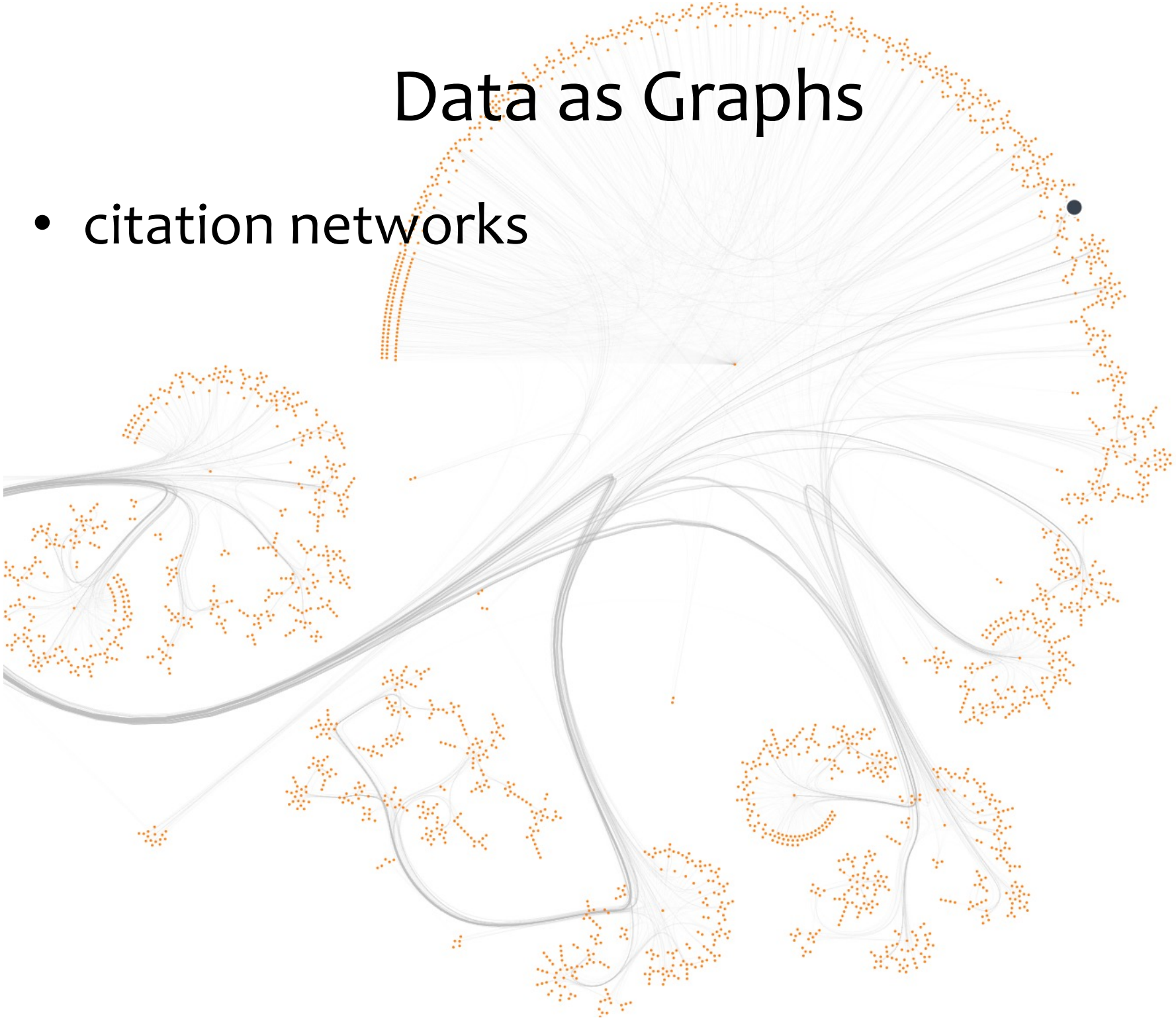
Reminders

- 10-618 Mini-Project
  - Team Formation Due: Tue, Nov 29
  - Proposal Due: Thu, Dec 1
  - Summary & Code Due: Fri, Dec 9
- Practice Problems 2
  - Out: Wed, Dec 8
- Exam 2:
  - Thu, Dec 15, 5:30 – 7:30 PM

GRAPH NEURAL NETWORKS

Data as Graphs

- citation networks



Data as Graphs

- molecules

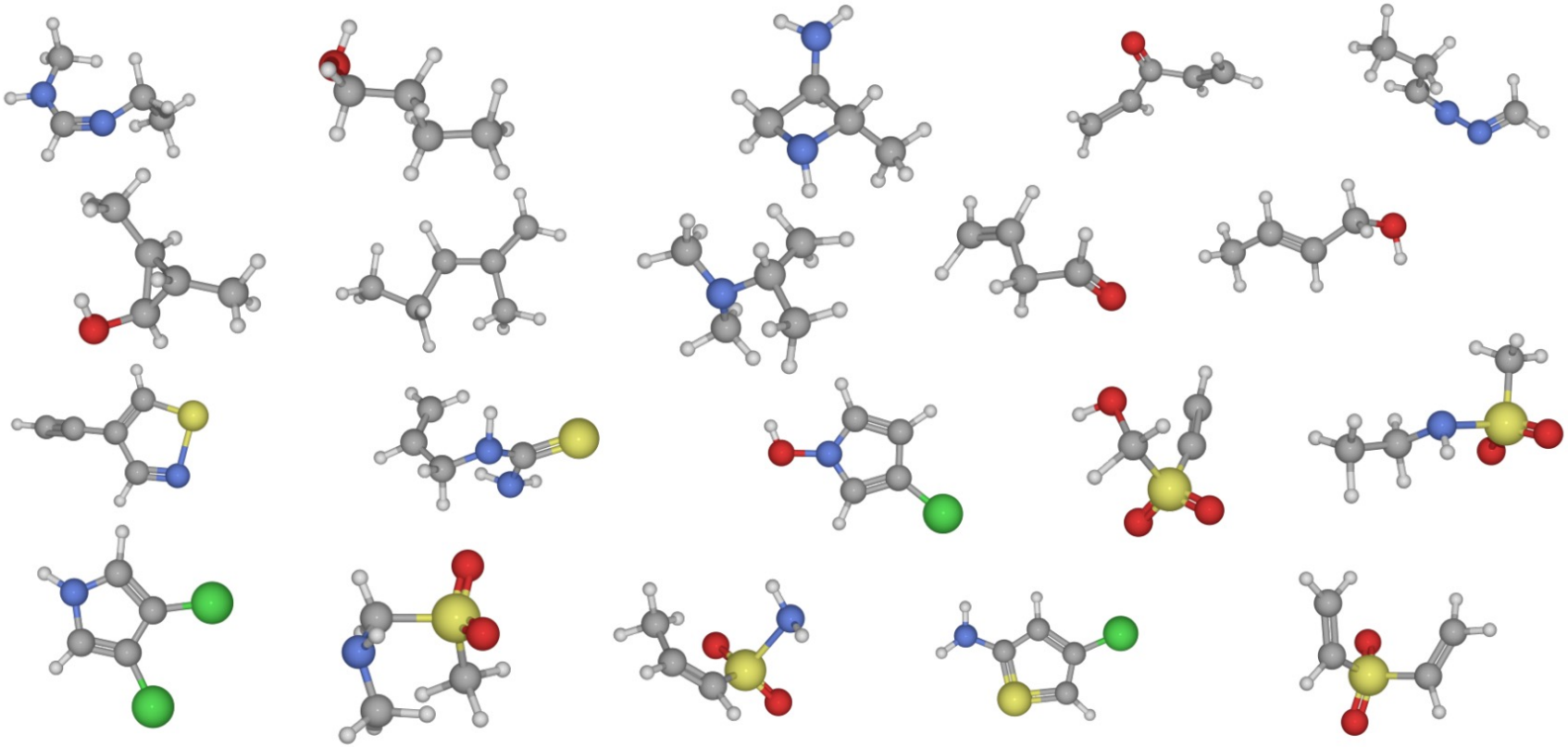


Figure from https://chemrxiv.org/engage/chemrxiv/article-details/6246035d3affe4aa143c3848

Data as Graphs

- semantic parsing

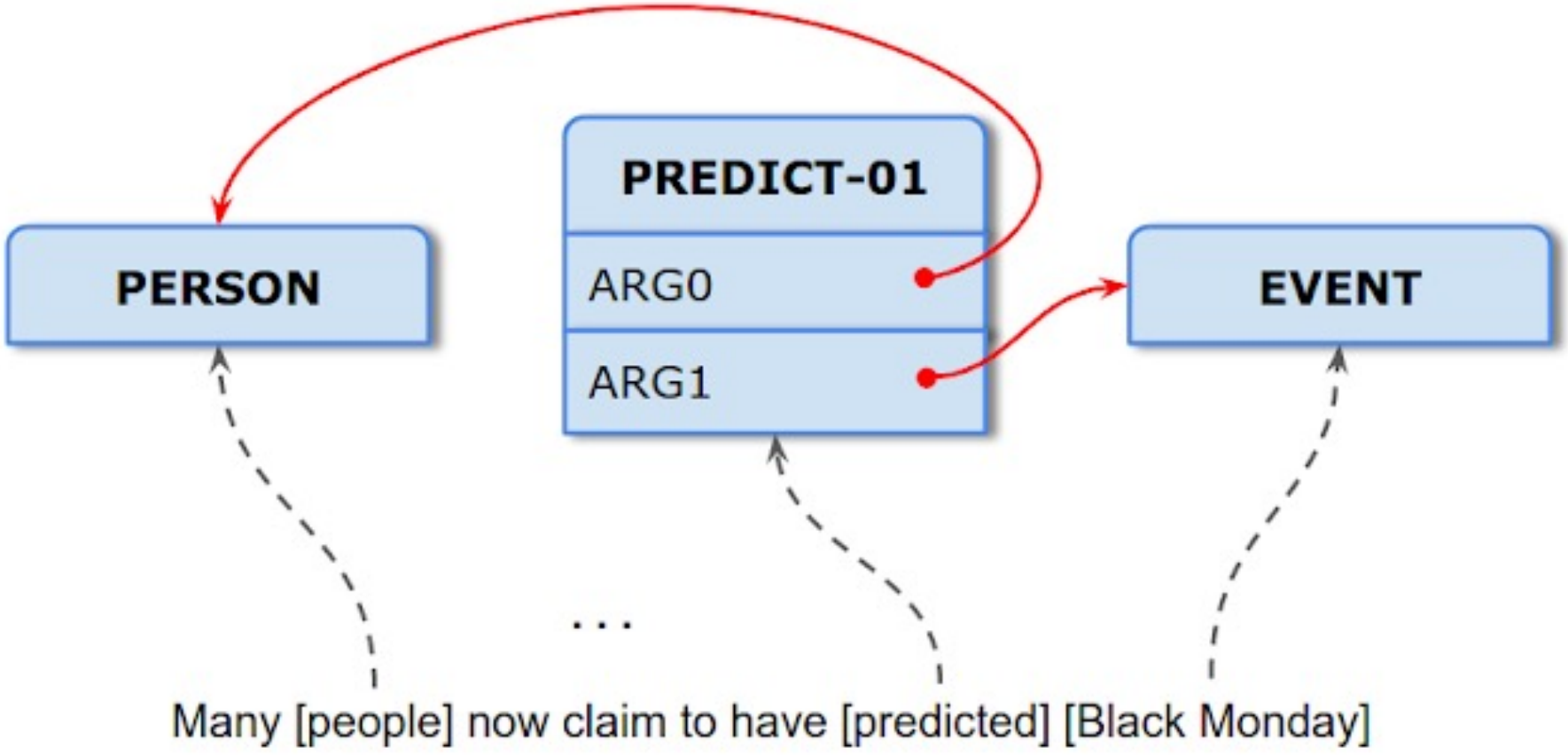
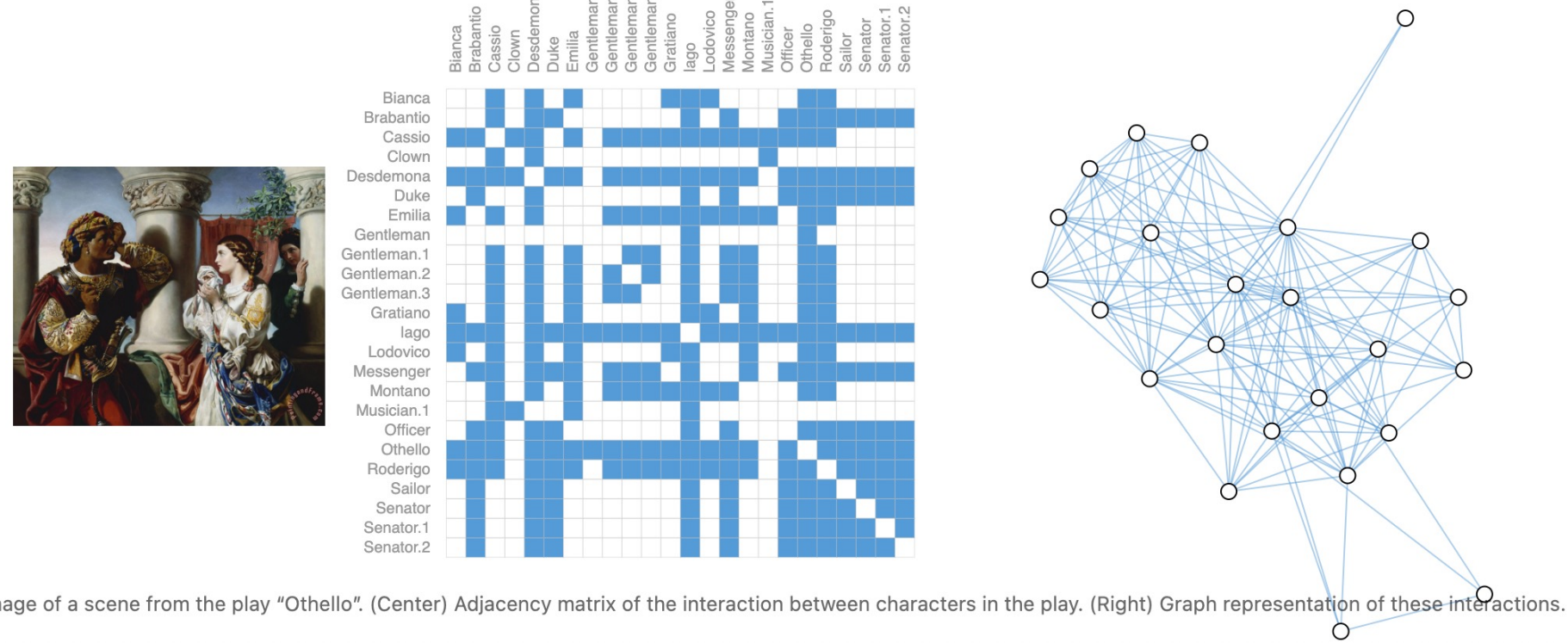


Figure from https://ai.googleblog.com/2017/11/sling-natural-language-frame-semantic.html

Data as Graphs

- social networks

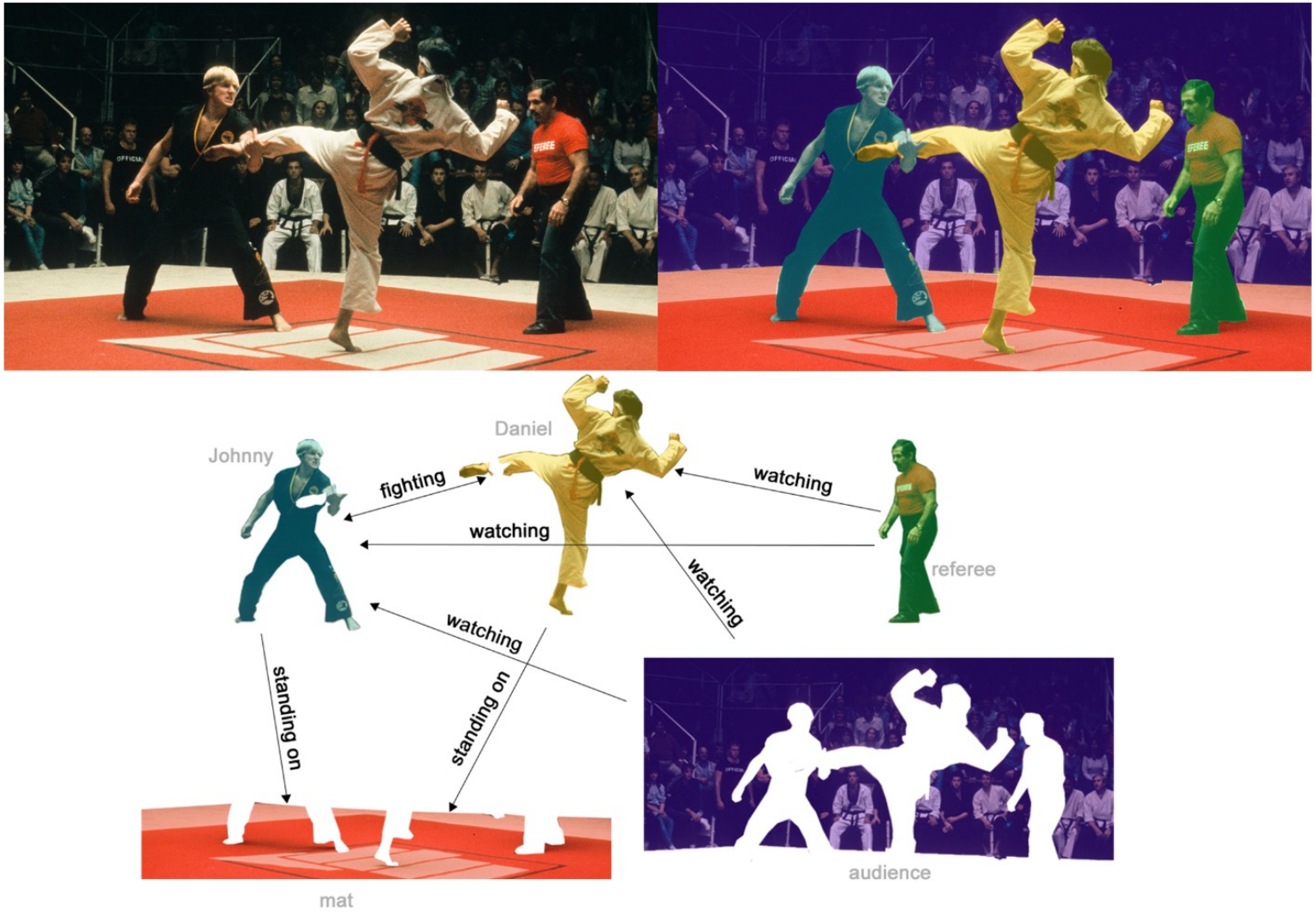


(Left) Image of a scene from the play "Othello". (Center) Adjacency matrix of the interaction between characters in the play. (Right) Graph representation of these interactions.

Figure from https://distill.pub/2021/gnn-intro/

Data as Graphs

- images



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.

Figure from https://distill.pub/2021/gnn-intro/

Graph Neural Networks

Decomposition of tasks for GNNs

- *Node-level*
  - **node classification:** predict a label for each node
  - **node regression:** predict a value for each node
- *Edge-level*
  - **edge classification:** predict a label for each edge
  - **link prediction:** predict presence/absence/strength of an edge
- *Graph-level*
  - **graph classification:** predict a label for the entire graph
  - **graph regression:** predict a value for the entire graph

Types of GNNs

A **Taxonomy** of Graph Neural Networks (GNNs) from Wu et al. (2020):

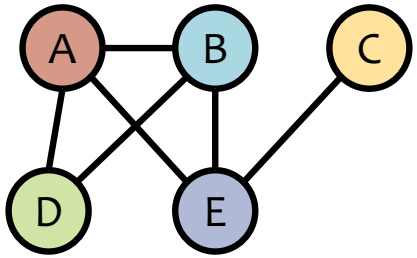
1. Recurrent GNNs
2. Convolutional GNNs
   a. Spectral-based
   b. Spatial-based
3. Graph autoencoders
   a. for network embedding
   b. for graph generation
4. Spatial-temporal GNNs

Taxonomy from https://arxiv.org/pdf/1901.00596.pdf

Node and Edge Representations

- *Def*: each node v has a **node feature vector $\mathbf{x}_v \in \mathbb{R}^M$**
- *Def*: each edge e has an **edge feature vector $\mathbf{x}_e \in \mathbb{R}^{M'}$**
- For undirected graphs, we (usually) assume there is only **one vector per undirected edge** (i.e. not one for each of the two corresponding directed edges)

$\mathbf{x}_A$	.1	-7	…	2
$\mathbf{x}_B$	4	-2	…	3
$\mathbf{x}_C$	-5	1	…	1
$\mathbf{x}_D$	0	3	…	.5
$\mathbf{x}_E$	.6	-.3	…	.1



$\mathbf{x}_{A,B}$	2	.4	…	-2
$\mathbf{x}_{A,D}$	.3	.1	…	-.5
$\mathbf{x}_{A,E}$	-5	1	…	4
$\mathbf{x}_{B,D}$	.9	9	…	-9
$\mathbf{x}_{B,E}$	1	-4	…	7
$\mathbf{x}_{C,E}$	6	-.2	…	0

Node and Edge Representations

- *Def*: each node v has a **node feature vector** $\mathbf{x}_v \in \mathbb{R}^M$
- *Def*: each edge e has an **edge feature vector** $\mathbf{x}_e \in \mathbb{R}^{M'}$
- For undirected graphs, we (usually) assume there is only **one vector per undirected edge** (i.e. not one for each of the two corresponding directed edges)

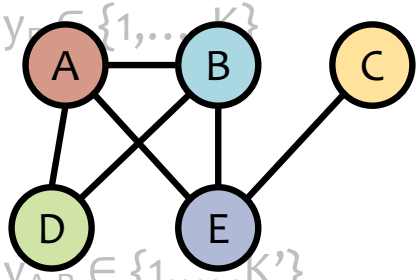
$\mathbf{x}_A$	.1	-7	…	2
$\mathbf{x}_B$	4	-2	…	3
$\mathbf{x}_C$	-5	1	…	1
$\mathbf{x}_D$	0	3	…	.5
$\mathbf{x}_E$	.6	-.3	…	.1

$y_A \in \{1,\dots,K\}$

$y_B \in \{1,\dots,K\}$

$y_C \in \{1,\dots,K\}$

$y_D \in \{1,\dots,K\}$



$\mathbf{x}_{A,B}$	2	.4	…	-2
$\mathbf{x}_{A,D}$	.3	.1	…	-.5
$\mathbf{x}_{A,E}$	-5	1	…	4
$\mathbf{x}_{B,D}$	.9	9	…	-9
$\mathbf{x}_{B,E}$	1	-4	…	7
$\mathbf{x}_{C,E}$	6	-.2	…	0

$y_{A,B} \in \{1,\dots,K'\}$

$y_{A,D} \in \{1,\dots,K'\}$

$y_{A,E} \in \{1,\dots,K'\}$

$y_{B,D} \in \{1,\dots,K'\}$

$y_{B,E} \in \{1,\dots,K'\}$

$y_{C,E} \in \{1,\dots,K'\}$

SPATIAL GRAPH NEURAL NETWORKS

Spatial Graph Neural Networks

Whiteboard:

- Basic node-only GNN
- Basic neighbor-only GNN
- Visualizing the k-hop neighborhood computation graph
- Incorporating self-loops
- Normalization techniques
- Adding edge features

GNN Properties

Whiteboard:

- Permutation Invariance
- Permutation Equivariance

GRAPH CONVOLUTIONAL NETWORK (GCN)

Graph Convolutional Network (GCN)

- Introduced by Kipf & Welling (2017)
  - Popular example of a Spectral GNN
  - The implementation (with all of its approximations) be interpreted as a Spatial GNN
- Two key ideas:
  - self-loop updates (A' = A + I)
  - symmetric normalization
- GCN layer:

$$\mathbf{h}_{v_j}^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v' \in \mathcal{N}(v_j) \cup \{v_j\}} \frac{\mathbf{h}_{v'}}{\sqrt{|\mathcal{N}(v_j)||\mathcal{N}(v')|}}\right)$$

Graph Convolutional Network (GCN)

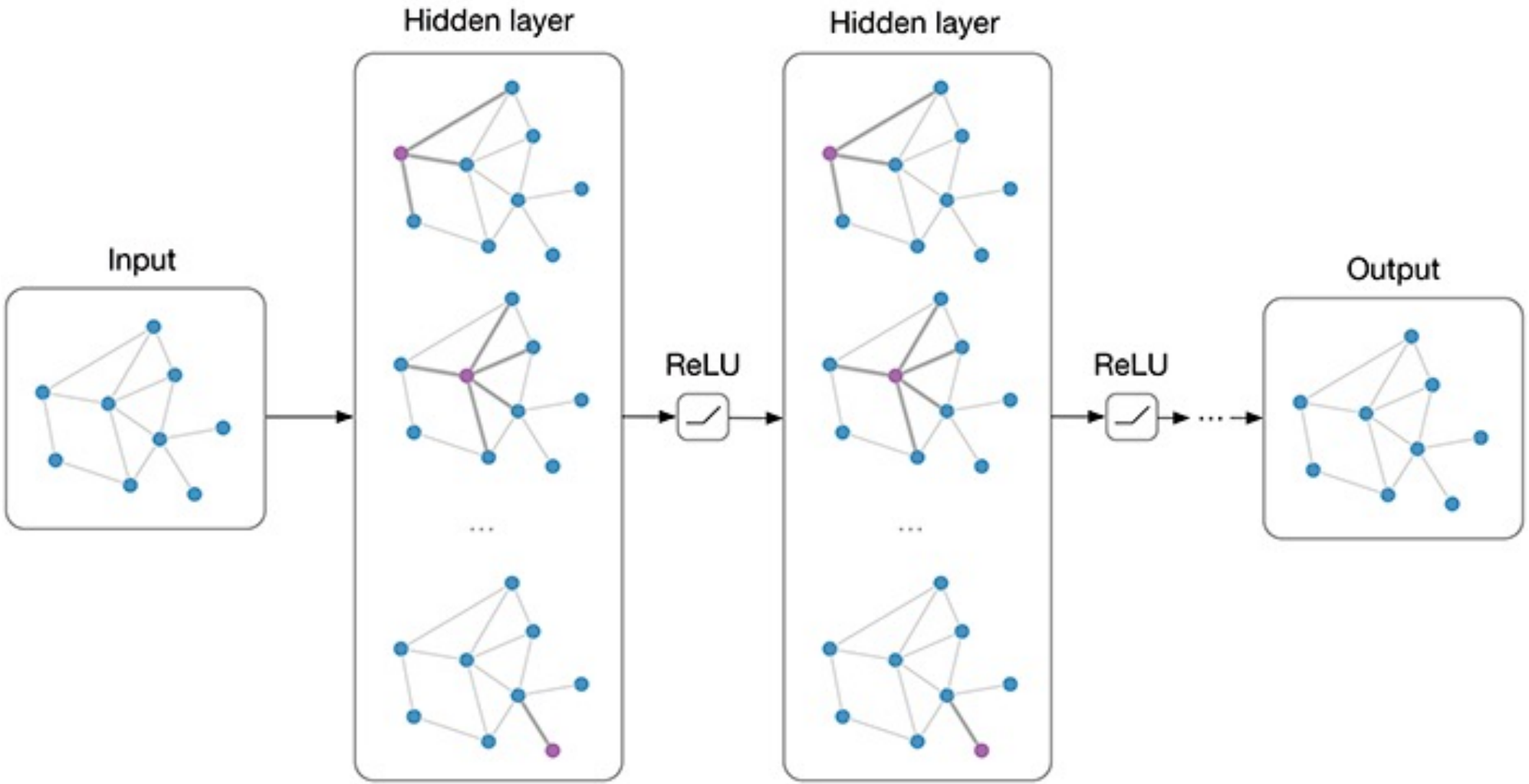
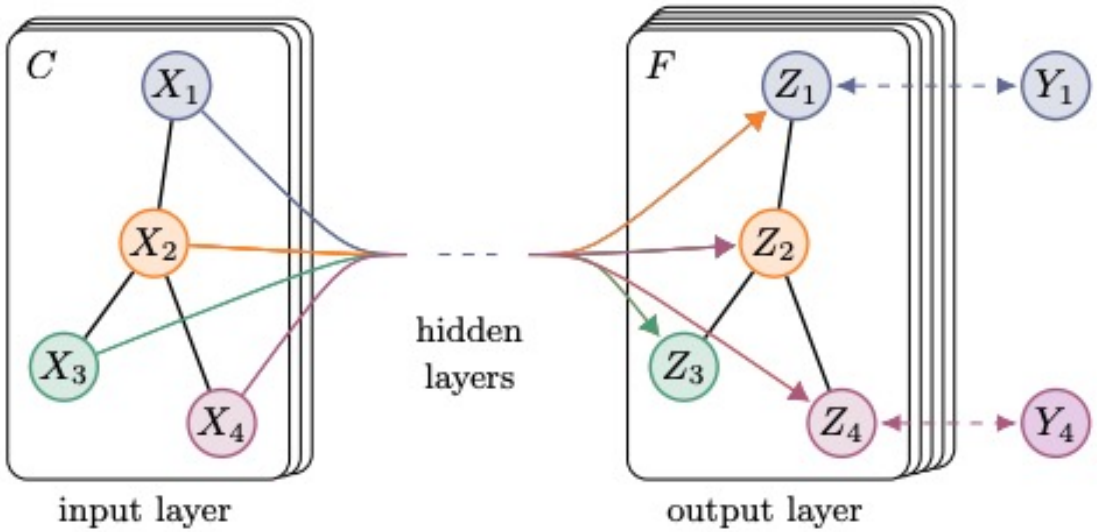


Figure from https://tkipf.github.io/graph-convolutional-networks/

Graph Convolutional Network (GCN)

- Trained in a semi-supervised fashion (i.e. not all nodes will have labels)



(a) Graph Convolutional Network

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

where $\mathcal{Y}_L$ is the set of node indices that have labels.

Figure from https://arxiv.org/pdf/1609.02907.pdf

Graph Convolutional Network (GCN)

- Derivation based on spectral principles allows evaluation of various alternative models that are closely related

6.2 Evaluation of Propagation Model

We compare different variants of our proposed per-layer propagation model on the citation network datasets. We follow the experimental set-up described in the previous section. Results are summarized in Table 3. The propagation model of our original GCN model is denoted by *renormalization trick* (in bold). In all other cases, the propagation model of both neural network layers is replaced with the model specified under *propagation model*. Reported numbers denote mean classification accuracy for 100 repeated runs with random weight matrix initializations. In case of multiple variables Θ_i per layer, we impose L2 regularization on all weight matrices of the first layer.

Table 3: Comparison of propagation models.

Description		**Propagation model**	**Citeseer**	**Cora**	**Pubmed**
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

Figure from https://arxiv.org/pdf/1609.02907.pdf

GNNS IN PRACTICE

GNN Example Datasets

Category	Data set	Source	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes
Citation Networks	Cora	[117]	1	2708	5429	1433	7
	Citeseer	[117]	1	3327	4732	3703	6
	Pubmed	[117]	1	19717	44338	500	3
	DBLP (v11)	[118]	1	4107340	36624464	-	-
Bio-chemical Graphs	PPI	[119]	24	56944	818716	50	121
	NCI-1	[120]	4110	29.87	32.30	37	2
	MUTAG	[121]	188	17.93	19.79	7	2
	D&D	[122]	1178	284.31	715.65	82	2
	PROTEIN	[123]	1113	39.06	72.81	4	2
	PTC	[124]	344	25.5	-	19	2
	QM9	[125]	133885	-	-	-	-
	Alchemy	[126]	119487	-	-	-	-
Social Networks	Reddit	[42]	1	232965	11606919	602	41
	BlogCatalog	[127]	1	10312	333983	-	39
Others	MNIST	[128]	70000	784	-	1	10
	METR-LA	[129]	1	207	1515	2	-
	Nell	[130]	1	65755	266144	61278	210

Table from https://arxiv.org/pdf/1901.00596.pdf

GNN Example Datasets

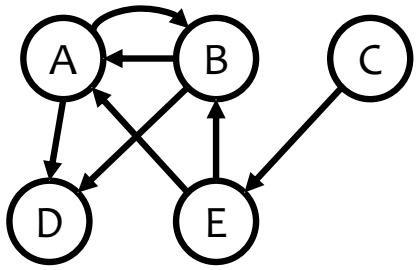
Category	Data set	Source	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes
Citation Networks	Cora	[117]	1	2708	5429	1433	7
	Citeseer	[117]	1	3327	4732	3703	6
	Pubmed	[117]	1	19717	44338	500	3
				4107340	36624464	-	-
				56944	818716	50	121
			110	29.87	32.30	37	2
			88	17.93	19.79	7	2
			178	284.31	715.65	82	2
			113	39.06	72.81	4	2
	PTC	[124]	344	25.5	-	19	2
	QM9	[125]	133885	-	-	-	-
	Alchemy	[126]	119487	-	-	-	-
Social Networks	Reddit	[42]	1	232965	11606919	602	41
	BlogCatalog	[127]	1	10312	333983	-	39
Others	MNIST	[128]	70000	784	-	1	10
	METR-LA	[129]	1	207	1515	2	-
	Nell	[130]	1	65755	266144	61278	210

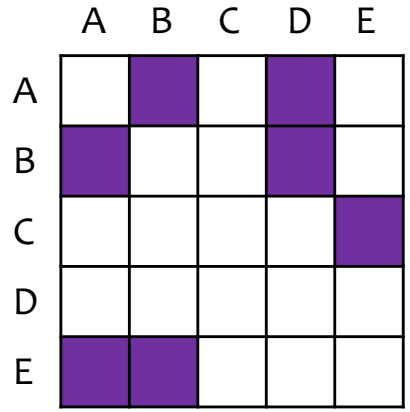
Most graphs we encounter are rather **sparse.**
That is, the number of edges is far less than $O(N^2)$ for N nodes

Table from https://arxiv.org/pdf/1901.00596.pdf

GNN Implementation Details

- Problem
  - Usually our adjacency matrix is quite **sparse**
  - Densely representing the matrix is out of the question
- Two implementation approaches for a graph neural network library:
  - sparsely represent the **adjacency matrix** and employ a sparse tensor library to implement message passing
  - directly represent the **adjacency list** and implement the message passing over the adjacency list (e.g. drop into C++ or a CUDA kernel)





[(A,B), (A,D), (B,A), (B,D), (C,E), (E,A), (E,B)]

GNN Example Datasets

Category	Data set	Source	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes
Citation Networks	Cora	[117]	1	2708	5429	1433	7
	Citeseer	[117]	1	3			6
	Pubmed	[117]	1	1			3
	DBLP (v11)	[118]	1	4107340	36624464	-	-
Bio-chemical Graphs	PPI	[119]	24	56944	818716	50	121
	NCI-1	[120]	4110	2			2
	MUTAG	[121]	188	1			2
	D&D	[122]	1178	2			2
	PROTEIN	[123]	1113	3			2
	PTC	[124]	344	2			2
	QM9	[125]	133885	-	-	-	-
	Alchemy	[126]	119487	-	-	-	-
Social Networks	Reddit	[42]	1	232965	11606919	602	41
	BlogCatalog	[127]	1	10312	333983	-	39
Others	MNIST	[128]	70000	784	-	1	10
	METR-LA	[129]	1	207	1515	2	-
	Nell	[130]	1	65755	266144	61278	210

Some datasets have one graph and our goal is to impute the missing parts (labels, values)

Some datasets have many graphs that can be divided into train/val/test sets

Table from https://arxiv.org/pdf/1901.00596.pdf

GNN Implementation Details

Batching

- An open problem for GNNs:
  - How to batch when training?
- Typical solutions are dataset/task specific
  - **Example:** if you only have one monolithic graph with tightly clustered neighborhoods (e.g. citation network), you can subsample small neighborhoods of the original graph (e.g. all nodes/edges within k hops of a randomly selected node)
  - **Example:** sometimes breaking up into a neighborhood doesn't make sense (e.g. neighborhood of one molecule is another molecule!) and you need to ensure that the entire graph is preserved when learning

PYTORCH C++ FRONTEND

PyTorch C++ Frontend

- What is the C++ Frontend?
  - Most applications employ the Python frontend for PyTorch
  - Yet most of PyTorch itself is implemented in C++
  - The C++ Frontend for PyTorch is an easy-to-use API that mirrors the Python Frontend
- Why use it?
  - Python
    - notoriously slow
    - requires tons of memory for lots of small objects
    - does not (easily) support multi-threading (see the Global Interpreter Lock, aka. GIL)
  - C++
    - nortoriuosly fast
    - uses minimal memory for lots of small objects
    - easily supports multi-threading
- Relevance to Structured Prediction
  - Many of the algorithms we employ are sparse in nature because of the graph structures underlying them (e.g. probabilistic graphical models)
  - Implementing these algorithms in C++ could easily give a 10x – 100x speedup and dramatic memory savings
- Main Takeaway
  - yes, you should go (re-)learn some C++

Python C++ Frontend

The same functionality in Python and C++ looks remarkably similar!

Python Frontend

```
import torch

model = torch.nn.Linear(num_features, 1)

optimizer = torch.optim.SGD(model.parameters())

data_loader = torch.utils.data.DataLoader(dataset)

for epoch in range(10):
    for example, label in data_loader:
        prediction = model.forward(example)
        loss = loss_function(prediction, label)
        loss.backward()
        optimizer.step()
```


Code example from FAIR lightning talk: https://www.youtube.com/watch?v=auRPXMMHJzc

Python C++ Frontend

The same functionality in Python and C++ looks remarkably similar!

C++ Frontend

```
#include <torch/torch.h>

torch::nn::Linear model(num_features, 1);

torch::optim::SGD optimizer(model.parameters());

torch::data::DataLoader data_loader(dataset);

for (size_t epoch = 0; epoch < 10; ++epoch) {
  for (auto [example, label] : data_loader) {
    auto prediction = model->forward(example);
    auto loss = loss_function(prediction, label);
    loss.backward();
    optimizer.step();
  }
}
```


Code example from FAIR lightning talk: https://www.youtube.com/watch?v=auRPXMMHJzc

Python C++ Frontend

Most likely, you would still implement your outer training loop in Python, but call out to a C++ module that implements an efficient version of your algorithm

Python Frontend

```
import torch

class Net(torch.nn.Module):
  def __init__(self, N, M):
    super(Net, self).__init__()
    self.W = torch.nn.Parameter(torch.randn(N, M))
    self.b = torch.nn.Parameter(torch.randn(M))

  def forward(self, input):
    return torch.addmm(self.b, input, self.W)
```

Code example from https://pytorch.org/tutorials/advanced/cpp_frontend.html

Python C++ Frontend

Most likely, you would still implement your outer training loop in Python, but call out to a C++ module that implements an efficient version of your algorithm

C++ Frontend

```
#include <torch/torch.h>

struct Net : torch::nn::Module {
  Net(int64_t N, int64_t M) {
    W = register_parameter("W", torch::randn({N, M}));
    b = register_parameter("b", torch::randn(M));
  }
  torch::Tensor forward(torch::Tensor input) {
    return torch::addmm(b, input, W);
  }
  torch::Tensor W, b;
};
```


Code example from https://pytorch.org/tutorials/advanced/cpp_frontend.html

GRAPH ATTENTION NETWORKS

Graph Attention Network

- The attention mechanism of the Transformer model (Vaswani et al., 2017) has influenced how we build models for language data, vision data, video data, etc.

- **Question**: How can the idea of attention also be applied to graphs?

- The Graph Attention Network (GAT) (Veličković et al., 2018) provides one answer…

Figure from https://arxiv.org/pdf/1710.10903.pdf

Graph Attention Network

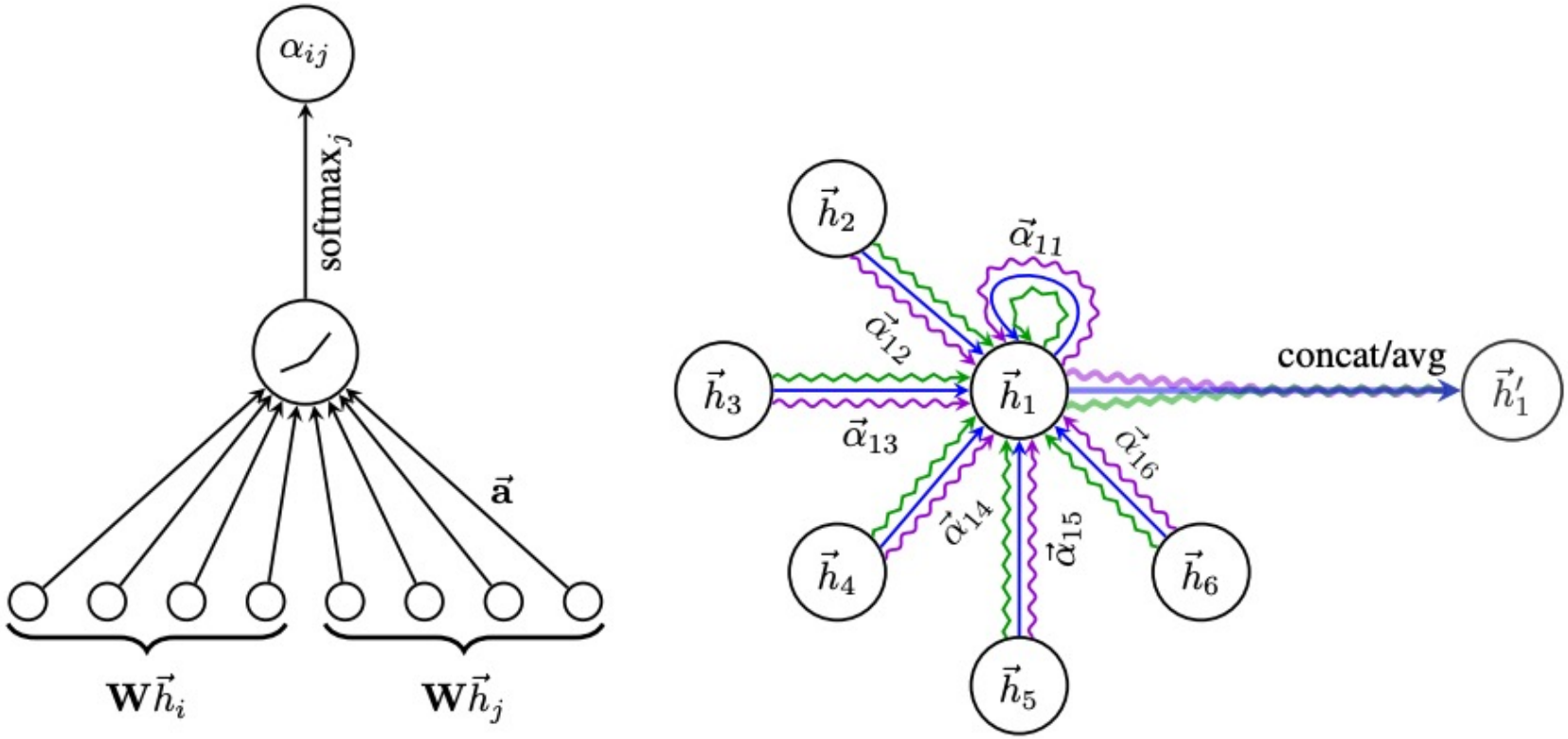
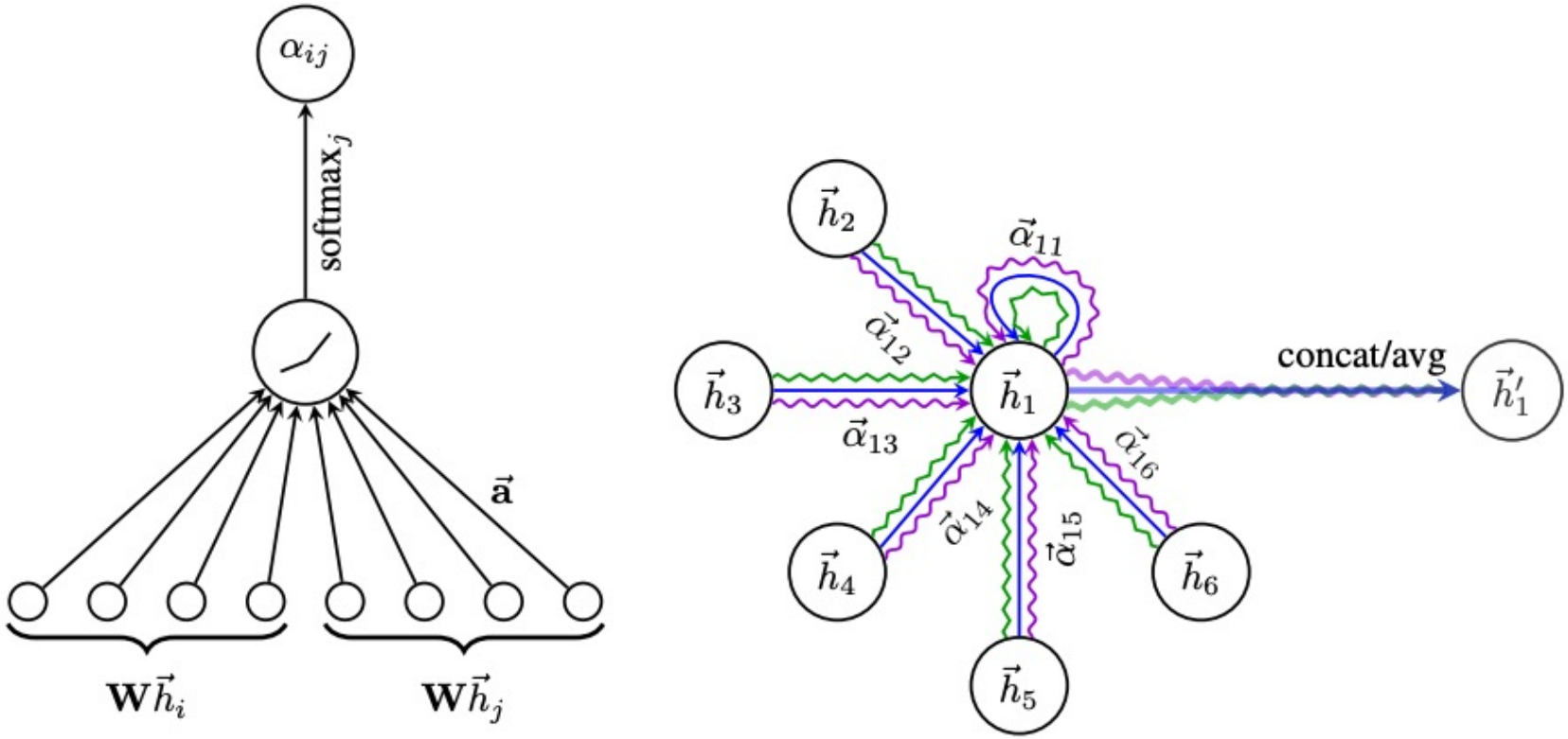


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Figure from https://arxiv.org/pdf/1710.10903.pdf

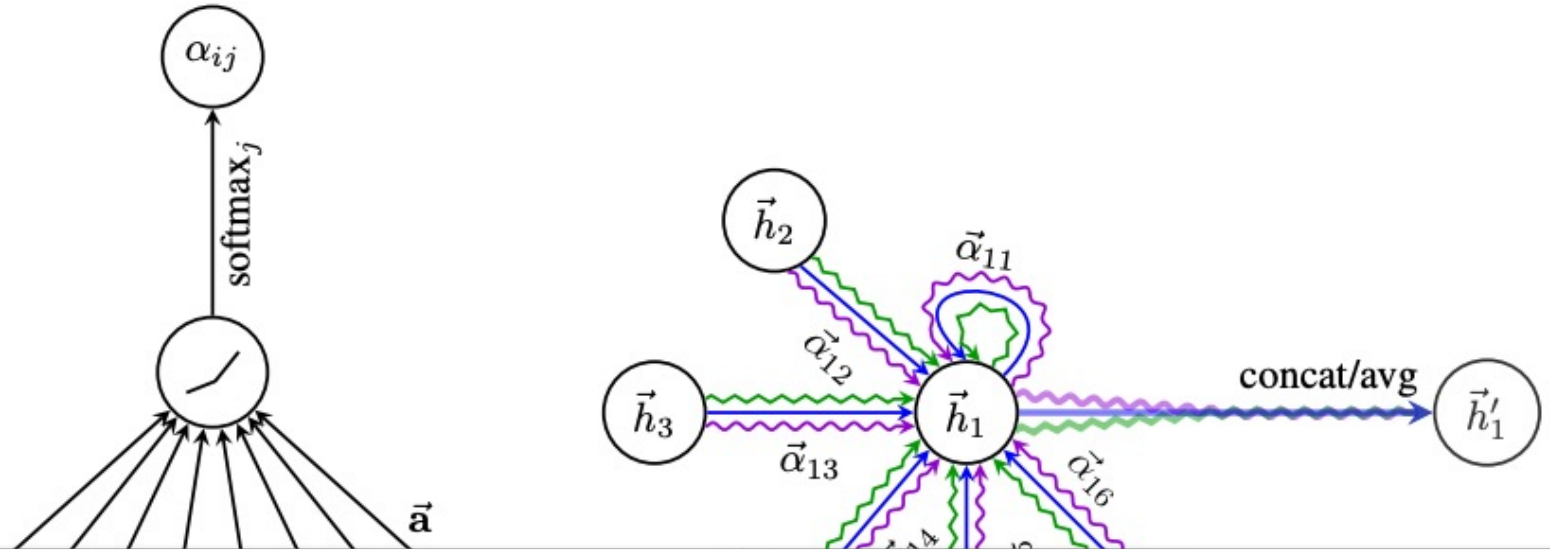
Graph Attention Network



$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Figure from https://arxiv.org/pdf/1710.10903.pdf

Graph Attention Network



To stabilize the learning process of self-attention, we have found extending our mechanism to employ *multi-head attention* to be beneficial, similarly to Vaswani et al. (2017). Specifically, K independent attention mechanisms execute the transformation of Equation 4, and then their features are concatenated, resulting in the following output feature representation:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{5}$$

where $\|$ represents concatenation, α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism (a^k), and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix. Note that, in this setting, the final returned output, $\mathbf{h}'$, will consist of KF' features (rather than F') for each node.

Figure from https://arxiv.org/pdf/1710.10903.pdf

Graph Attention Network

	Cora	Citeseer	Pubmed
Task	Transductive	Transductive	Transductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)
# Edges	5429	4732	44338
# Features/Node	1433	3703	500
# Classes	7	6	3
# Training Nodes	140	120	60
# Validation Nodes	500	500	500
# Test Nodes	1000	1000	1000

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

EXAM 2 LOGISTICS

Exam 2

- **Time / Location**
  - **Time: Thu, Dec. 15 at 5:30pm – 7:30pm, DH A302**
    During final exam period.
  - **Location: DH A302**
    Please arrive a few minutes early.
  - Please watch Piazza carefully for announcements.
- **Logistics**
  - Covered material: Lecture 11 – Lecture 26
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
    - Drawing
  - No electronic devices
  - You are allowed to **bring** one 8½ x 11 sheet of notes (front and back)

Topics for Exam 1

- Search-Based Structured Prediction
  - Reductions to Binary Classification
  - Learning to Search
  - RNN-LMs
  - seq2seq models
- Graphical Model Representation
  - Directed GMs vs. Undirected GMs vs. Factor Graphs
  - Bayesian Networks vs. Markov Random Fields vs. Conditional Random Fields
- Graphical Model Learning
  - ~~Fully observed Bayesian Network learning~~
  - Fully observed MRF learning
  - Fully observed CRF learning
  - Parameterization of a GM
  - Neural potential functions
- Exact Inference
  - Three inference problems: (1) marginals (2) partition function (3) most probably assignment
  - Variable Elimination
  - Belief Propagation (sum-product and max-product)

Topics for Exam 2

- Approximate Inference by Sampling
  - Monte Carlo Methods
  - Gibbs Sampling
  - Metropolis-Hastings
  - Markov Chains and MCMC
- Parameter Estimation
  - [Fully observed Bayesian Network learning]
  - Bayesian inference
  - Topic Modeling
- Approximate Inference by Optimization
  - Variational Inference
  - Mean Field Variational Inference
  - Coordinate Ascent V.I. (CAVI)
  - Variational EM
- Deep Generative Models
  - Variational Autoencoders
- MAP Inference and Learning
  - MAP Inference via MILP
  - MAP Inference via LP relaxation
  - Structured Perceptron
  - Structured SVM
- Other Topics
  - Causal Inference
  - Bayesian Nonparametrics
    - Dirichlet Process
    - DP Mixture Model
  - Graph Neural Networks

Sample Questions

MCMC

3. (1 point) Suppose you are given a first order Markov Chain over a series of random variables $Y_1, Y_2, Y_3, \ldots$. Let $P(Y_t)$ be the marginal probability for variable Y_t.

   True or False: The equilibrium distribution of the Markov Chain is $P^*(Y)$ if and only if $\lim_{t\to\infty} P(Y_t) = P^*(Y)$.

   ○ True

   ○ False

Sample Questions

MCMC

2. Consider a new distribution over four random variables $P(Z_1, Z_2, Z_3, Z_4)$ for which you build a Gibbs sampler.

   (a) (1 point) **Numerical Answer:** Suppose you begin with the sample $[z_0, z_1, z_2, z_3] = [0, 0, 0, 0]$. How many sampling steps are needed to transition to the sample $[1, 1, 1, 1]$?

   (b) (1 point) **Short Answer:** Describe or name a (closely related) sampling method that can transition from $[0, 0, 0, 0]$ to $[1, 1, 1, 1]$ in a smaller number of steps.

Sample Questions

Poll

Variational Inference

1. (1 point) **Select all that apply:** Which of the multivariate Gaussian distributions in Figure 4 can be perfectly recovered by a mean-field approximation?

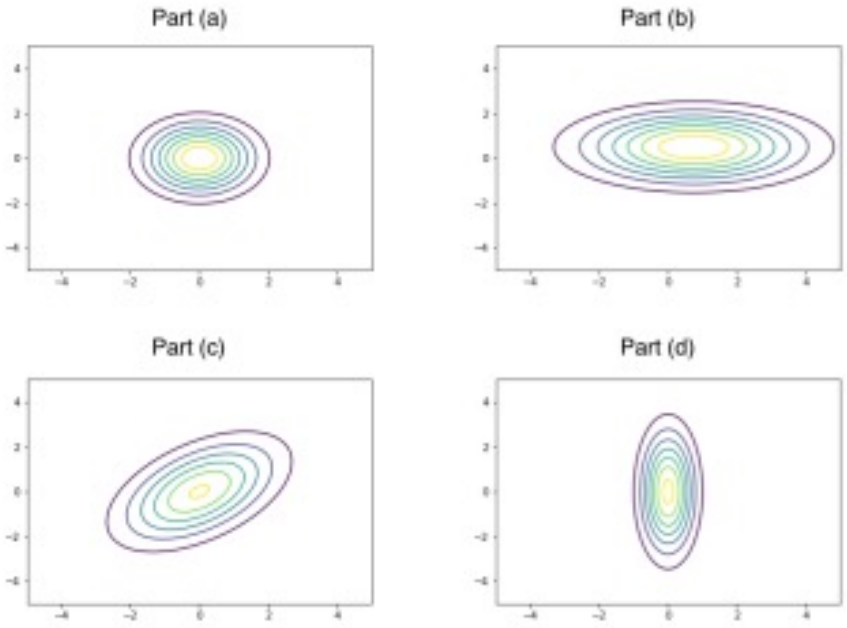


Figure 4: 2d Gaussian distributions with different means and covariance matrices.

Sample Questions

Variational Inference

2. Suppose you wish to run Variational EM for a true distribution $p_\alpha(\mathbf{x}, \mathbf{z})$ with a variational approximation $q_\theta(\mathbf{z})$.

   (a) (1 point) **Select all that apply:** Which of the following describe the Variational E-Step?

- ☐ keep θ fixed and update α
- ☐ keep α fixed and update θ
- ☐ run variational inference to minimize $KL(q_\theta || p_\alpha)$
- ☐ improve $E_{q_\theta}[\log p_\alpha(\mathbf{x}, \mathbf{z})]$ by adjusting α
- ☐ None of the above

Sample Questions

Variational Inference

2. Suppose you wish to run Variational EM for a true distribution $p_\alpha(\mathbf{x}, \mathbf{z})$ with a variational approximation $q_\theta(\mathbf{z})$.

   (b) (1 point) **Select all that apply:** Which of the following describe the Variational M-Step?

- ☐ keep θ fixed and update α
- ☐ keep α fixed and update θ
- ☐ run variational inference to minimize $KL(q_\theta || p_\alpha)$
- ☐ improve $E_{q_\theta}[\log p_\alpha(\mathbf{x}, \mathbf{z})]$ by adjusting α
- ☐ None of the above

Sample Questions

Poll

MAP Inference and Structured SVM

1. (1 point) **Numerical Answer**: Consider the following training example, scoring function, and loss function.

$\mathbf{x} = [2, -1, 3] \in \mathbb{R}^3$
$\mathbf{y}^* = [0, 0, 1] \in \{0,1\}^3$

$\text{score}(\mathbf{x}, \mathbf{y}) = \mathbf{y}^T\mathbf{x}$

$l(\mathbf{y}, \mathbf{y}^*) = 10*\text{HammingLoss}(\mathbf{y}, \mathbf{y}^*)$

(a) Which value of y would standard MAP inference return?

(b) Which value of y would loss-augmented MAP inference return?

2. (1 point) **Short Answer**: When does the LP relaxation of the MILP for MAP inference yield the most probable assignment to the variables of a Markov Random Field?

Sample Questions

Bayesian Nonparametrics

1. (1 point) **Numerical Answer**: Suppose you are working with a Gibbs Sampler for a Chinese Restaurant Process. In the current sample there are:
- 4 people at table A
- 2 people at table B
- 3 people at table C

In the next step, you resample the table assignment of an individual sitting at Table B. What are the respective probabilities associated with sitting at tables A, B, C, and at an unoccupied table?

2. (1 point) **True / False**: Bayesian nonparametric models do not have any parameters.

Sample Questions

Poll

Causal Inference

1. (1 point) **True / False**: In order to conclude that one variable has a causal effect on another variable, one must collect data from a randomized control trial.

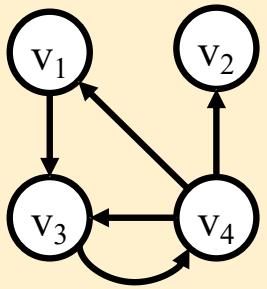
2. (1 point) **Short Answer**: How are the causal relationships between variables encoded in the path diagram of a structural causal model?

3. (1 point) **Short Answer**: What are the key algorithmic challenges associated with causal inference?

Sample Questions

Graph Neural Networks

1. (1 point) **Short Answer**: Consider the following graph G = (V,E):



Write an equation representing how the node representation h3 would be constructed from $v_1,\ldots, v_4$ if we applied a graph convolutional network (i.e., the GCN of Kipf and Welling) to the graph shown.

2. (1 point) **True / False**: A recurrent graph neural network builds up vector representations of nodes using only convolution and pooling layers, whereas a spatial graph neural network builds up vector representations of nodes using only spectral methods.

3. (1 point) **Short Answer**: Suppose we define a graph neural network using an undirected linear-chain over N variables. How many entries in the graph's adjacency matrix are non-zero?

Course Staff

Education Associate

Joshmin

TAs

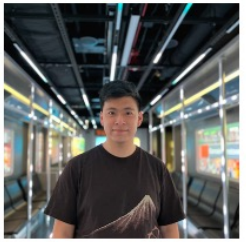
Eric Liang

Carl Qi

Mukuntha Narayanan

Riyaz

Harnoor Dhingra



Owen Wang