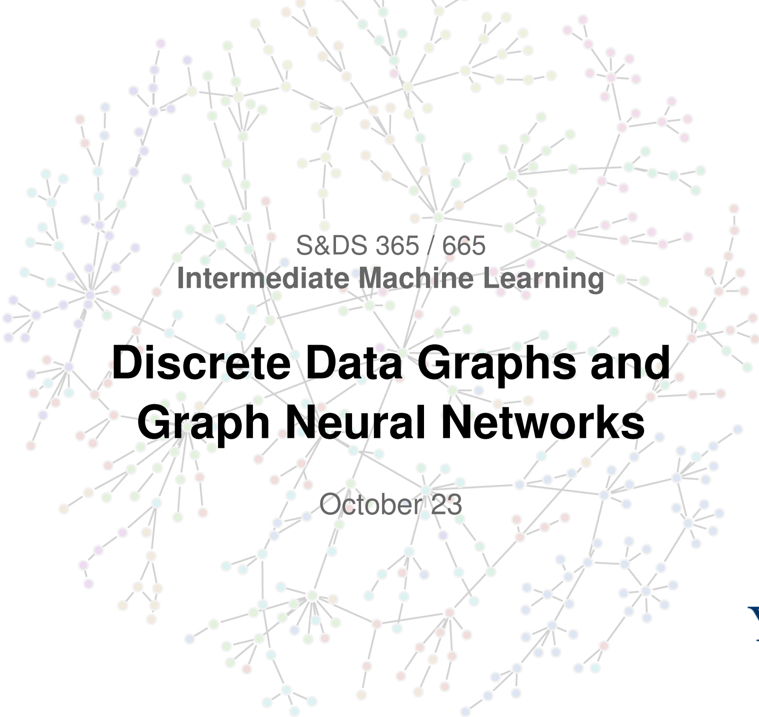


S&DS 365 / 665

Intermediate Machine Learning

Discrete Data Graphs and Graph Neural Networks

October 23

Yale

A rare lull

- Assignment 3 out; due next Wednesday
- Assignment 4 posted next week
- Midterm scores and mid-semester grades posted tomorrow

For today

- Quick recap
- Graphs for discrete data
- Intro to graph neural networks

Graphs

- A natural language for describing various data
- Give information about relationships between variables
- Associated with each multivariate distribution

Undirected Graphs

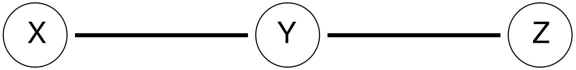
A graph $G = (V, E)$ has vertices V, edges E.

If $X = (X_1, \ldots, X_p)$ is a random variable, we will study graphs where there are p vertices, one for each X_j.

The graph will encode conditional independence relations among the variables.

Undirected graphs

Simplest case:



Here $V = \{X, Y, Z\}$ and $E = \{(X, Y), (Y, Z)\}$.

This encodes the independence relation

$$X \perp\!\!\!\perp Z \mid Y$$

which means that *X and Z are independent conditioned on Y*.

Markov Property

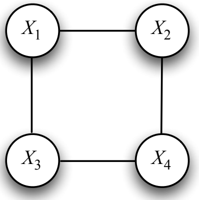
A probability distribution P satisfies the *global Markov property* with respect to a graph G if:

for any disjoint vertex subsets A, B, and C such that C separates A and B,

$$X_A \perp\!\!\!\perp X_B \mid X_C.$$

- X_A are the random variables X_j with $j \in A$.
- C separates A and B means that there is no path from A to B that does not pass through C.

Example

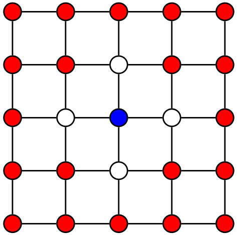


$$X_1 \perp\!\!\!\perp X_4 \mid X_2, X_3$$

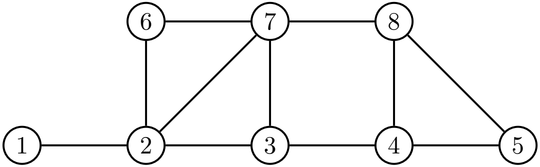
$$X_2 \perp\!\!\!\perp X_3 \mid X_1, X_4$$

Example: 2-dimensional grid

The blue node is independent of the red nodes given the white nodes.



Example



$C = \{3, 7\}$ separates $A = \{1, 2\}$ and $B = \{4, 8\}$. Hence,

$$\{X_1, X_2\} \perp\!\!\!\perp \{X_4, X_8\} \quad \Big| \quad \{X_3, X_7\}$$

Special case

If $(i, j) \notin E$ then

$$X_i \perp\!\!\!\perp X_j \;\Big|\; \{X_k : k \neq i, j\}$$

$$A = \{i\}, B = \{j\}, C = \{k \neq i, j\}$$

Special case

If $(i, j) \notin E$ then

$$X_i \perp\!\!\!\perp X_j \;\Big|\; \{X_k : k \neq i, j\}$$

$$A = \{i\}, B = \{j\}, C = \{k \neq i, j\}$$

Lack of an edge from i to j implies that X_i and X_j are independent given all of the other random variables.

Graph estimation

- A graph G represents the class of distributions, $\mathcal{P}(G)$, the distributions that are Markov with respect to G
- Graph estimation: Given n samples $X_1, \ldots, X_n \sim P$, estimate the graph G.

Factored form

Theorem (Hammersley, Clifford, Besag)

A positive distribution over random variables $X_1, \ldots, X_p$ satisfies the Markov properties of graph G if and only if it can be represented as

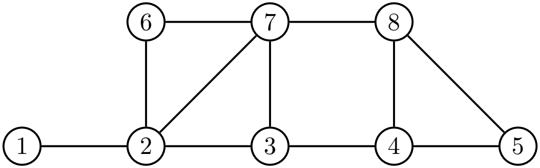
$$p(X) \propto \prod_{c \in \mathcal{C}} \psi_c(X_c)$$

where $\mathcal{C}$ is the set of cliques in the graph G.

set🀄任何两个节点都有边相连

A clique is a subset of vertices for which each pair is connected by an edge.

Example



$$
\begin{aligned}
P(X) \;\propto\; & \psi_{12}(X_1, X_2) \cdot \psi_{267}(X_2, X_6, X_7) \cdot \psi_{237}(X_2, X_3, X_7) \cdot \\
& \psi_{34}(X_3, X_4) \cdot \psi_{48}(X_4, X_8) \cdot \psi_{78}(X_7, X_8) \cdot \psi_{458}(X_4, X_5, X_8)
\end{aligned}
$$

Gaussian case

Let $\Omega = \Sigma^{-1}$ be the precision matrix.

A zero in Ω indicates a *lack of the corresponding edge* in the graph

So, the adjacency matrix of the graph is

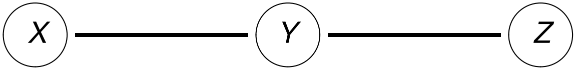
$$A = \big(\mathbb{1}(\Omega_{ij} \neq 0)\big)$$

That is,

$$A_{ij} = \begin{cases} 1 & \text{if } |\Omega_{ij}| > 0 \\ 0 & \text{otherwise} \end{cases}$$

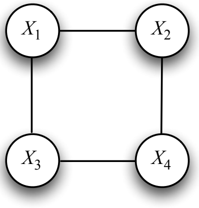
Gaussian case

$$\Omega \equiv \Sigma^{-1} = \begin{pmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{pmatrix}$$



Gaussian case

$$\Omega \equiv \Sigma^{-1} = \begin{pmatrix} * & * & * & 0 \\ * & * & 0 & * \\ * & 0 & * & * \\ 0 & * & * & * \end{pmatrix}$$



$$X_1 \perp\!\!\!\perp X_4 \mid X_2, X_3$$

Gaussian case: Algorithms

Two approaches:

- parallel lasso
- graphical lasso

Parallel Lasso:

1. For each $j = 1, \ldots, p$ (in parallel): Regress X_j on all other variables using the lasso.
2. Put an edge between X_i and X_j if each appears in the regression of the other.

Graphical Lasso (glasso)

- Assume a multivariate Gaussian model
- Subtract out the sample mean
- Minimize the negative log-likelihood of the data, subject to a constraint on the sum of the absolute values of the inverse covariance

Graphical Lasso (glasso)

The glasso optimizes the parameters of $\Omega = \Sigma^{-1}$ by minimizing:

$$\text{trace}(\Omega S_n) - \log|\Omega| + \lambda \sum_{j \neq k} |\Omega_{jk}|$$

where $|\Omega|$ is the determinant and S_n is the sample covariance

$$S_n = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$$

There is a blockwise gradient descent algorithm to minimize this, using iterative lassos

Discrete Graphical Models

Challenges of handling discrete data:

- Models don’t have closed from; can’t compute normalizing constant
- Need to use Gibbs sampling, variational inference
- No analogue of the graphical lasso

Discrete Graphical Models

- Positive distributions can be represented by an exponential family,

$$p(X) \propto \exp\left(\sum_{c \in \mathcal{C}} \phi_c(X_c)\right)$$

where $\phi_c = \log \psi_c$ from Hammersley-Clifford

- Special case: Ising Model (discrete Gaussian)

$$p_\beta(X) \propto \exp\left(\sum_{i \in V} \beta_i X_i + \sum_{(i,j) \in E} \beta_{ij} X_i X_j\right).$$

From edges to cliques

Take $\beta_i \equiv 0$ for simplicity

If we have a triangle (i, j, k) in the graph then the potential function corresponds to

$$\begin{aligned}\psi_{ijk}(X_i, X_j, X_k) &= e^{\beta_{ij} X_i X_j} \cdot e^{\beta_{jk} X_j X_k} \cdot e^{\beta_{ik} X_i X_k} \\ &= e^{\beta_{ij} X_i X_j + \beta_{jk} X_j X_k + \beta_{ik} X_i X_k}\end{aligned}$$

Ising

We have a graph with edges E and vertices V. Each node i has a random variable X_i that can be "up" ($X_i = 1$) or "down" ($X_i = 0$)

$$p_\beta(x_1, \ldots, x_n) \propto \exp\left(\sum_{s \in V} \beta_s x_s + \sum_{(s,t) \in E} \beta_{st} x_s x_t \right)$$

Since $2X_i - 1 \in \{-1, 1\}$ if $X_i \in \{0, 1\}$, can re-parameterize in terms of sample space $X_i = \pm 1$.

Ising

We have a graph with edges E and vertices V. Each node i has a random variable X_i that can be "up" ($X_i = 1$) or "down" ($X_i = 0$)

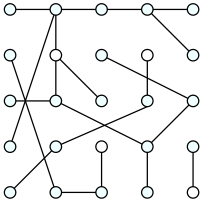
$$p_\beta(x_1, \ldots, x_n) \propto \exp\left(\sum_{s \in V} \beta_s x_s + \sum_{(s,t) \in E} \beta_{st} x_s \, x_t \right)$$

E are the set of edges, V are the vertices. Imagine the Z_i are votes of politicians, and the edges encode the social network of party affiliations

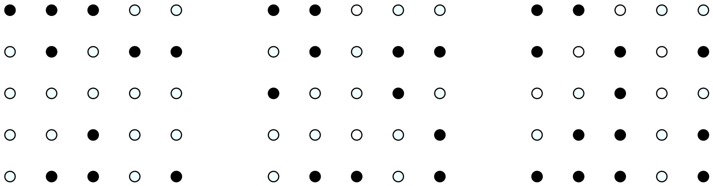
Since $2X_i - 1 \in \{-1, 1\}$ if $X_i \in \{0, 1\}$, can re-parameterize in terms of sample space $X_i = \pm 1$.

Graph Estimation

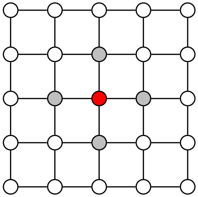
- Given n i.i.d. samples from an Ising distribution, $\{X_i, i = 1, \ldots, n\}$, (each is a p-vector of $\{0, 1\}$ values) identify underlying graph



- Multiple examples are observed:



Local Distributions



- Consider Ising model $p_\beta(X) \propto \exp\left(\sum_{i\in V}\beta_i X_i + \sum_{(i,j)\in E}\beta_{ij}X_iX_j\right)$.
- Conditioned on $(x_2,\ldots,x_p)$, variable $X_1 \in \{0,1\}$ has probability mass function given by a logistic function,

$$p(X_1 = 1 \mid x_2,\ldots,x_p) \;=\; \text{sigmoid}\left(\beta_1 + \sum_{j\in\mathcal{N}(1)}\beta_{1j}x_j\right)$$

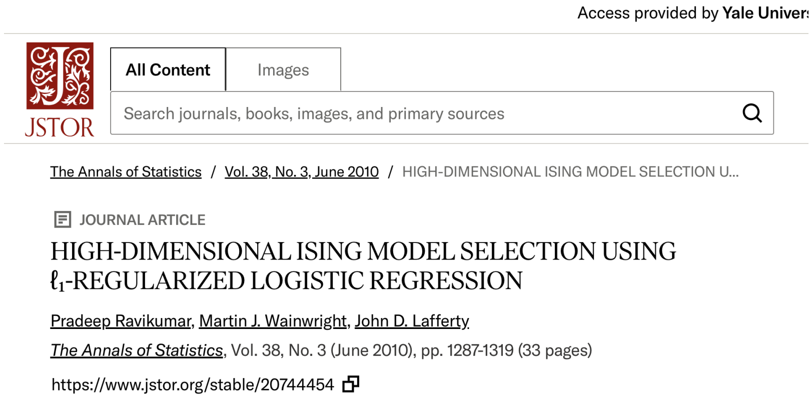
Parallel lasso (sparse logistic regressions)

Strategy

- Perform ℓ_1 regularized logistic regression of each node X_i on $X_{\setminus i} = \{X_j,\ j \neq i\}$ to estimate neighbors $\widehat{\mathcal{N}}(i)$
- Two versions:
  - Create an edge (i, j) if $j \in \widehat{\mathcal{N}}(i)$ *and* $i \in \widehat{\mathcal{N}}(j)$
  - Create an edge (i, j) if $j \in \widehat{\mathcal{N}}(i)$ *or* $i \in \widehat{\mathcal{N}}(j)$

https://www.jstor.org/stable/20744454

Parallel lasso (sparse logistic regressions)



Access provided by **Yale Univer**

The Annals of Statistics / Vol. 38, No. 3, June 2010 / HIGH-DIMENSIONAL ISING MODEL SELECTION U...

JOURNAL ARTICLE

HIGH-DIMENSIONAL ISING MODEL SELECTION USING ℓ_1-REGULARIZED LOGISTIC REGRESSION

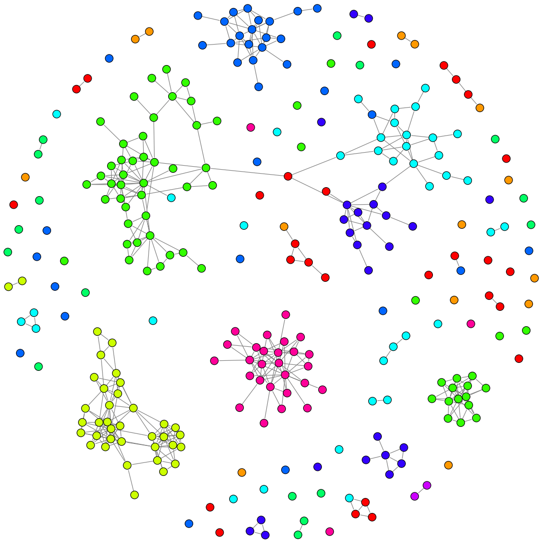
Pradeep Ravikumar, Martin J. Wainwright, John D. Lafferty

The Annals of Statistics, Vol. 38, No. 3 (June 2010), pp. 1287-1319 (33 pages)

https://www.jstor.org/stable/20744454

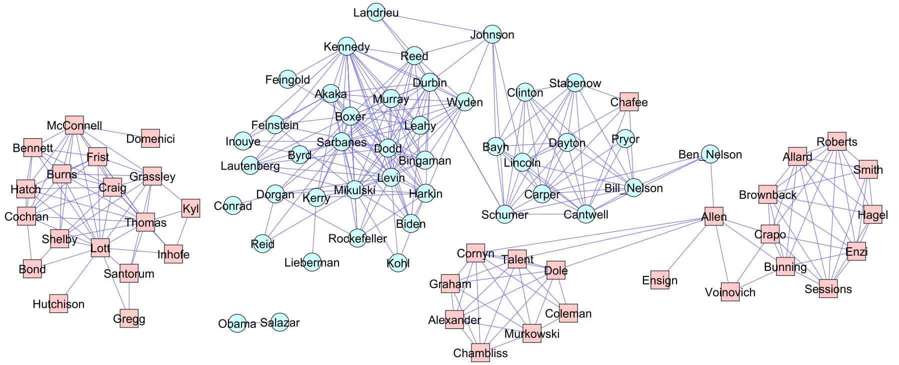
https://www.jstor.org/stable/20744454

S&P 500: Ising Model (Price up or down?)



Voting Data

Voting records of US Senate, 2006-2008



Example of Banerjee, El Gahoui, and d'Asepremont

Scaling behavior: Performance with data size

Maximum degree d of the p variables. Sample size n must satisfy

Ising model:	$n \geq d^3 \log p$
Graphical lasso:	$n \geq d^2 \log p$
Parallel lasso:	$n \geq d \log p$
Lower bound:	$n \geq d \log p$

- Each method makes different *incoherence assumptions*:
  - Correlations between unrelated variables not too large

https://www.jstor.org/stable/20744454

Summary: Graphs for discrete data

- A positive distribution factors into product of potential functions on the cliques of the graph
- Graphs and independence relations are same for discrete data
- Ising models are discrete Gaussians
- No version of the graphical lasso holds for discrete data; instead, we use the parallel lasso

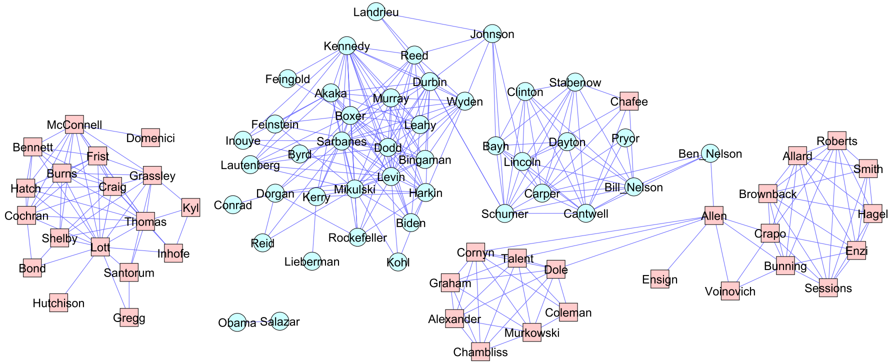
Graph neural networks

Next, we'll discuss graph neural networks, following this article:

`https://distill.pub/2021/understanding-gnns/`

Recall: Voting Data

Voting records of US Senate, 2006-2008

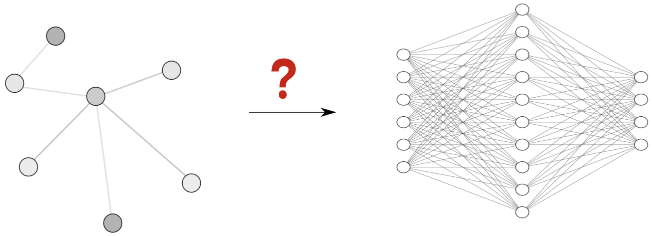


Suppose we have the graph and other covariates for each node or edge.
How can we classify the senators according to political orientation?

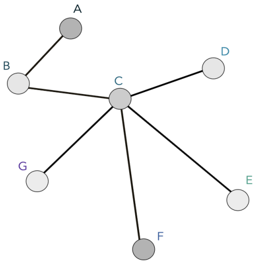
Graph neural networks

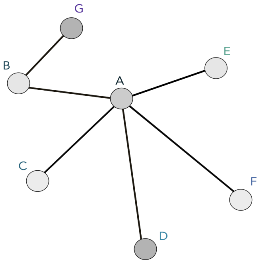
- We’ll discuss how GNNs correspond to CNNs
- The graph Laplacian plays a central role

Equivariance problem

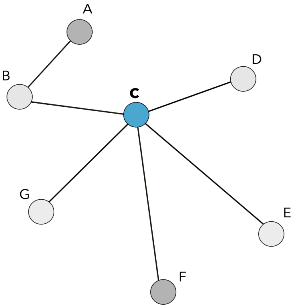


Equivariance problem





Graph Laplacian



Input Graph G

$$
\begin{array}{c|ccccccc}
 & A & B & \mathbf{C} & D & E & F & G \\
\hline
A & 1 & -1 & & & & & \\
B & -1 & 2 & -1 & & & & \\
\mathbf{C} & & -1 & 5 & -1 & -1 & -1 & -1 \\
D & & & -1 & 1 & & & \\
E & & & -1 & & 1 & & \\
F & & & -1 & & & 1 & \\
G & & & -1 & & & & 1
\end{array}
$$

Laplacian L of G

Polynomials of the Laplacian

$$p_w(L) = w_0 I_n + w_1 L + w_2 L^2 + \cdots w_d L^d$$

If dist$(u, v) > i$ then the (u, v) entry of L^i is zero

- This is analogous to a CNN filter (kernel)
- The weights w_i play role of filter coefficients
- Degree d of polynomial plays role of the size of the kernel

The Laplacian is a Mercer kernel

- Symmetric $L_{uv} = L_{vu}$
- Positive-definite:

$$f^T L f = \mathrm{trace}(L f f^T) = \sum_{(u,v) \in E} (f_u - f_v)^2 \geq 0$$

For more on properties of such kernels: https://www.ml.cmu.edu/research/dap-papers/kondor-diffusion-kernels.pdf, https://mlg.eng.cam.ac.uk/zoubin/papers/ssl-book.pdf

Classical Laplacian

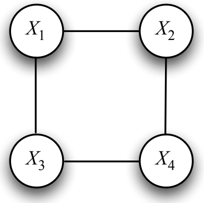
$$\Delta f = \text{trace}\left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right) = \frac{\partial^2 f}{\partial x_1^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2}$$

The Laplace operator in its various manifestations is the most beautiful and central object in all of mathematics. Probability theory, mathematical physics, Fourier analysis, partial differential equations, the theory of Lie groups, and differential geometry all revolve around this sun, and its light even penetrates such obscure regions as number theory and algebraic geometry.

Edward Nelson, *Tensor Analysis*

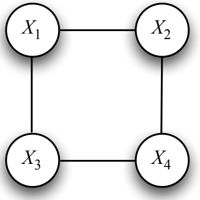
My Ph.D. supervisor :-)

Sanity checks



What is the Laplacian *L*?

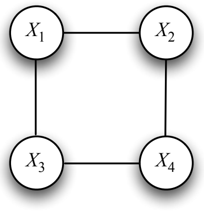
Sanity checks



What is the Laplacian L?

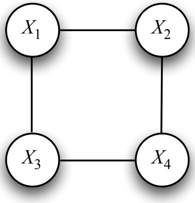
$$L = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

Sanity checks



What is L^2?

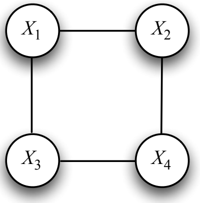
Sanity checks



What is L^2?

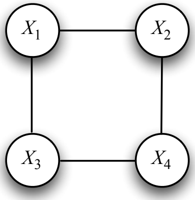
$$L^2 = \begin{pmatrix} 6 & -4 & -4 & 2 \\ -4 & 6 & 2 & -4 \\ -4 & 2 & 6 & -4 \\ 2 & -4 & -4 & 6 \end{pmatrix}$$

Sanity checks



If $x = (1, 2, 3, 4)^T$ what is $h = \text{ReLU}(Lx)$?

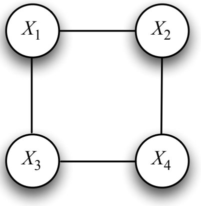
Sanity checks



If $x = (1, 2, 3, 4)^T$ what is $h = \text{ReLU}(Lx)$?

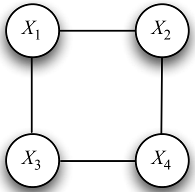
$$\text{ReLU}(Lx) = \text{ReLU}((-3, -1, 1, 3)^T) = (0, 0, 1, 3)^T$$

Sanity checks



If $x = (1, 2, 3, 4)^T$ what is $x^T L x$?

Sanity checks



If $x = (1, 2, 3, 4)^T$ what is $x^T Lx$?

$$x^T Lx = \sum_{(u,v)\in E} (x_u - x_v)^2 = 10$$

Equivariance

If we compute a feature map f, we'd like it to not depend on a reordering P of the nodes.

The reordering

$$\begin{aligned} 1 &\leftarrow 2 \\ 2 &\leftarrow 3 \\ 3 &\leftarrow 1 \end{aligned}$$

corresponds to permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Whence equivariance

A transformation $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is equivariant if

$$f(Px) = Pf(x)$$

for any permuation matrix P, where $PP^T = I$.

The transformed data and Laplacian are

$$\begin{aligned} x &\longrightarrow Px \\ L &\longrightarrow PLP^T \\ L^i &\longrightarrow PL^iP^T \end{aligned}$$

Whence equivariance

A transformation $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is equivariant if

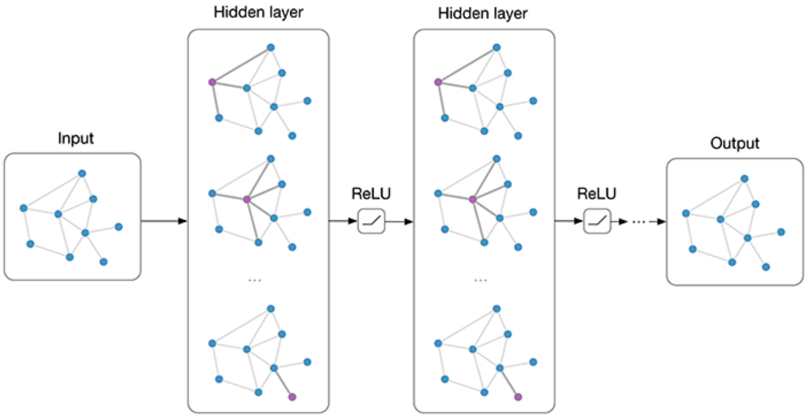
$$f(Px) = Pf(x)$$

for any permuation matrix P, where $PP^T = I$.

The transformed polynomial kernels are

$$\begin{aligned} f(Px) &= \sum_{i=0}^{d} w_i (PL^i P^T) Px \\ &= \sum_{i=0}^{d} w_i PL^i x \\ &= P \sum_{i=0}^{d} w_i L^i x \\ &= Pf(x) \end{aligned}$$

Building layers



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

Building layers

- In this example, the first layer gives three "feature maps"

$$\begin{aligned} h_1 &= \text{relu}(p_{w_1}(L)x) \\ h_2 &= \text{relu}(p_{w_2}(L)x) \\ h_3 &= \text{relu}(p_{w_3}(L)x) \end{aligned}$$

- The final output is then computed as

$$y = \text{relu}(\beta^T h)$$

- The hidden states h_i and output y are vectors, with a component for each node in the graph.

Course ad

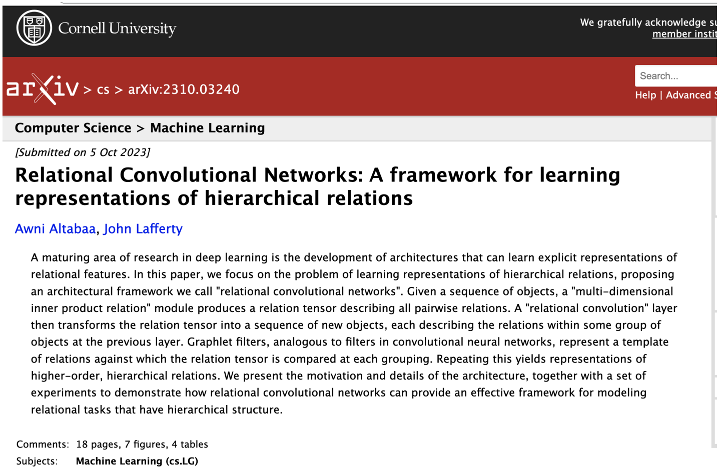
CPSC 483: Deep Learning on Graphs

Instructor: Rex Ying

Summary: Graph neural nets

- Certain data have natural graphical structure
- GNNs are analogues of CNNs for graphs
- Based on use of graph Laplacian
- Independent of ordering of nodes (equivariant)

A recent framework that combines ideas from CNNs and GNNs


Cornell University

We gratefully acknowledge su
member instit

arXiv > cs > arXiv:2310.03240

Search...

Help | Advanced S

Computer Science > Machine Learning

[Submitted on 5 Oct 2023]

Relational Convolutional Networks: A framework for learning representations of hierarchical relations

Awni Altabaa, John Lafferty

A maturing area of research in deep learning is the development of architectures that can learn explicit representations of relational features. In this paper, we focus on the problem of learning representations of hierarchical relations, proposing an architectural framework we call "relational convolutional networks". Given a sequence of objects, a "multi-dimensional inner product relation" module produces a relation tensor describing all pairwise relations. A "relational convolution" layer then transforms the relation tensor into a sequence of new objects, each describing the relations within some group of objects at the previous layer. Graphlet filters, analogous to filters in convolutional neural networks, represent a template of relations against which the relation tensor is compared at each grouping. Repeating this yields representations of higher-order, hierarchical relations. We present the motivation and details of the architecture, together with a set of experiments to demonstrate how relational convolutional networks can provide an effective framework for modeling relational tasks that have hierarchical structure.

Comments: 18 pages, 7 figures, 4 tables

Subjects: **Machine Learning (cs.LG)**

Next topic: Reinforcement learning