
E9: 309 ADL 6-1-2021
http://leap.ee.iisc.ac.in/sriram/teaching/ADL2020/

Housekeeping

- Midterm project III
  - Abstract submission deadline (Jan 10th)
    - ✓ Evaluation after final exam (1st week of Feb)
- Final Exam (as per IISc schedule)
  - ✓ Jan 23rd afternoon!

Deep learning on graphs

Lets start with an example

- Citation dataset
- Citeseer (collection of papers) or Cora

(7 labels) (6 labels)

Cora		CiteSeer	
Neural Networks	726	HCI	304
Case Based	285	IR	532
Reinforcement Learning	214	Agents	463
Probabilistic Methods	379	AI	115
Genetic Algorithms	406	ML	308
Rule Learning	131	DB	388
Theory	344		

Table 3: Class distribution for the citation datasets.

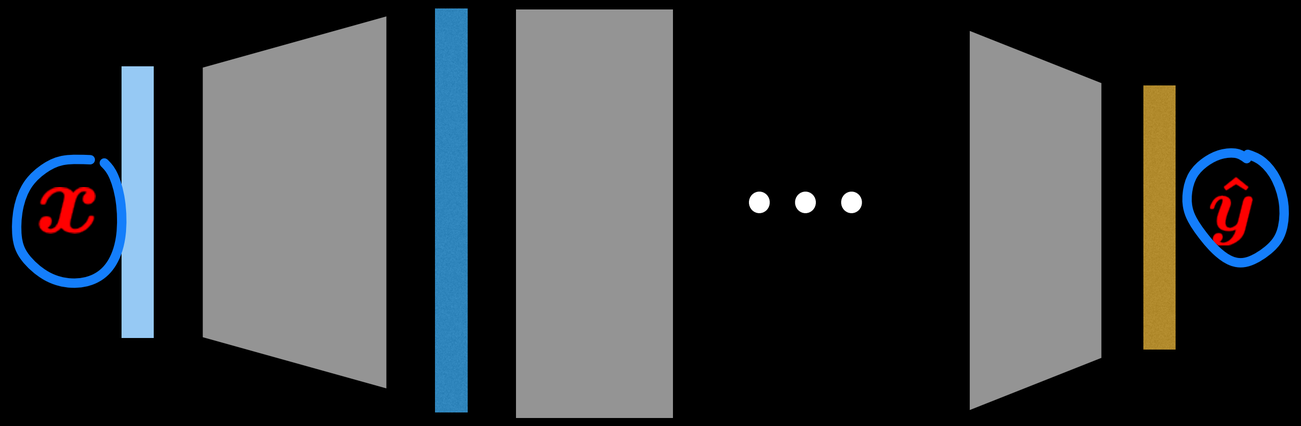
Input features

- Each paper in the dataset is converted to features (say Bag of Words)

Standard deep learning

$x(t) \leftrightarrow x(t+1)$

- Find a relationship between the inputs and outputs



y - true targets

$x_i - x_j$

- Find a relationship between the BoW features and output class labels
- Ignored any dependence between samples.

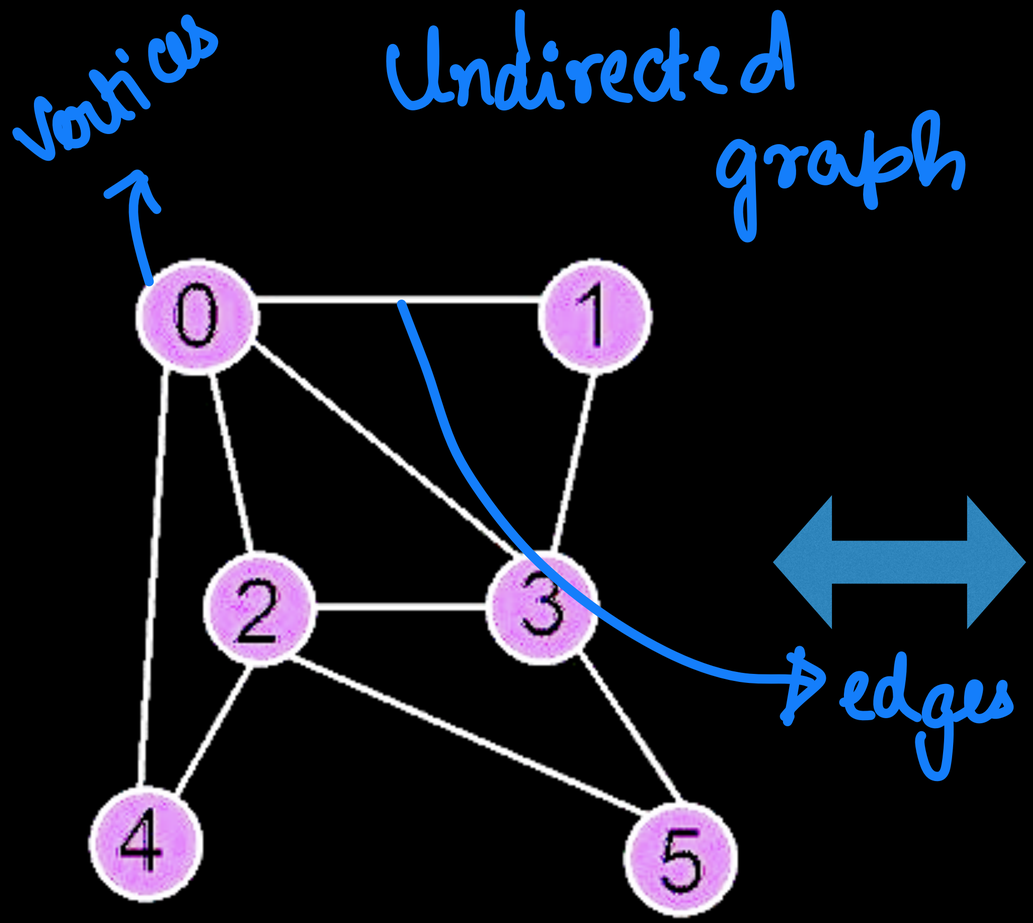
Given additional information between inputs

- For example in the citation dataset with N papers
  - If we know which papers cite which other papers
  - This can be represented as a binary matrix of size NxN
  - Relationship among documents exist which are not exploited in standard deep models.

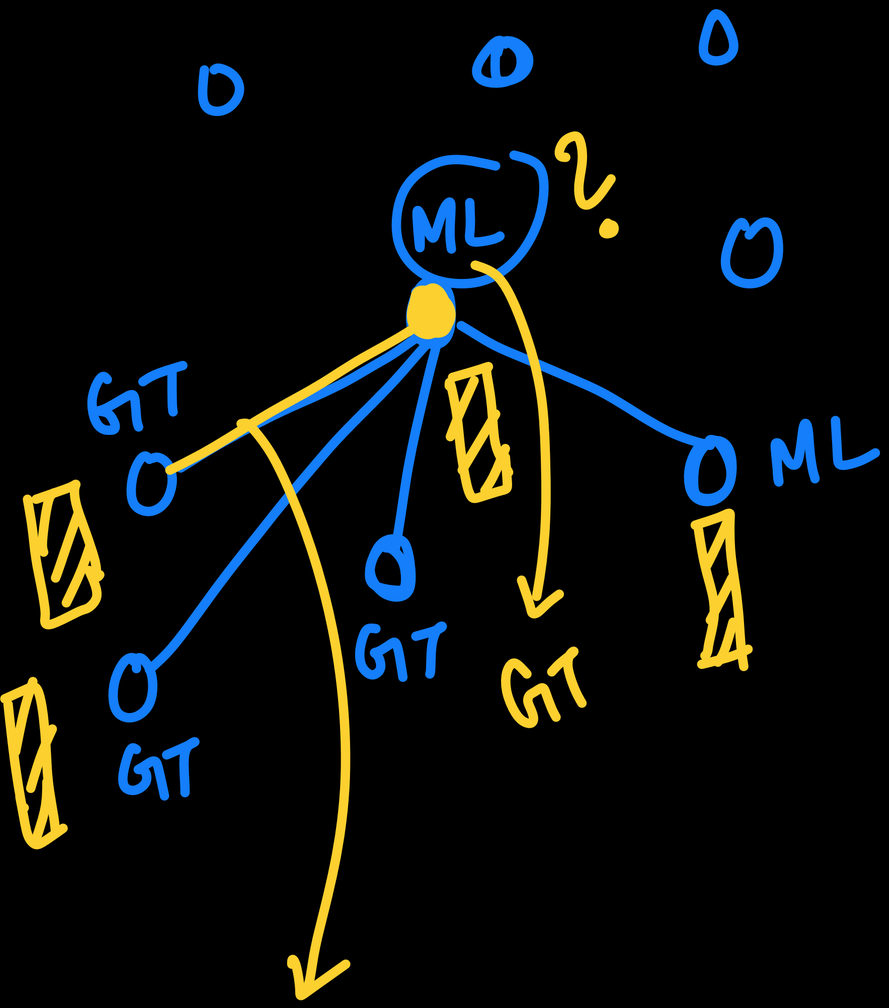
Symmetric

	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

Representing relationships in the input as graph



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0



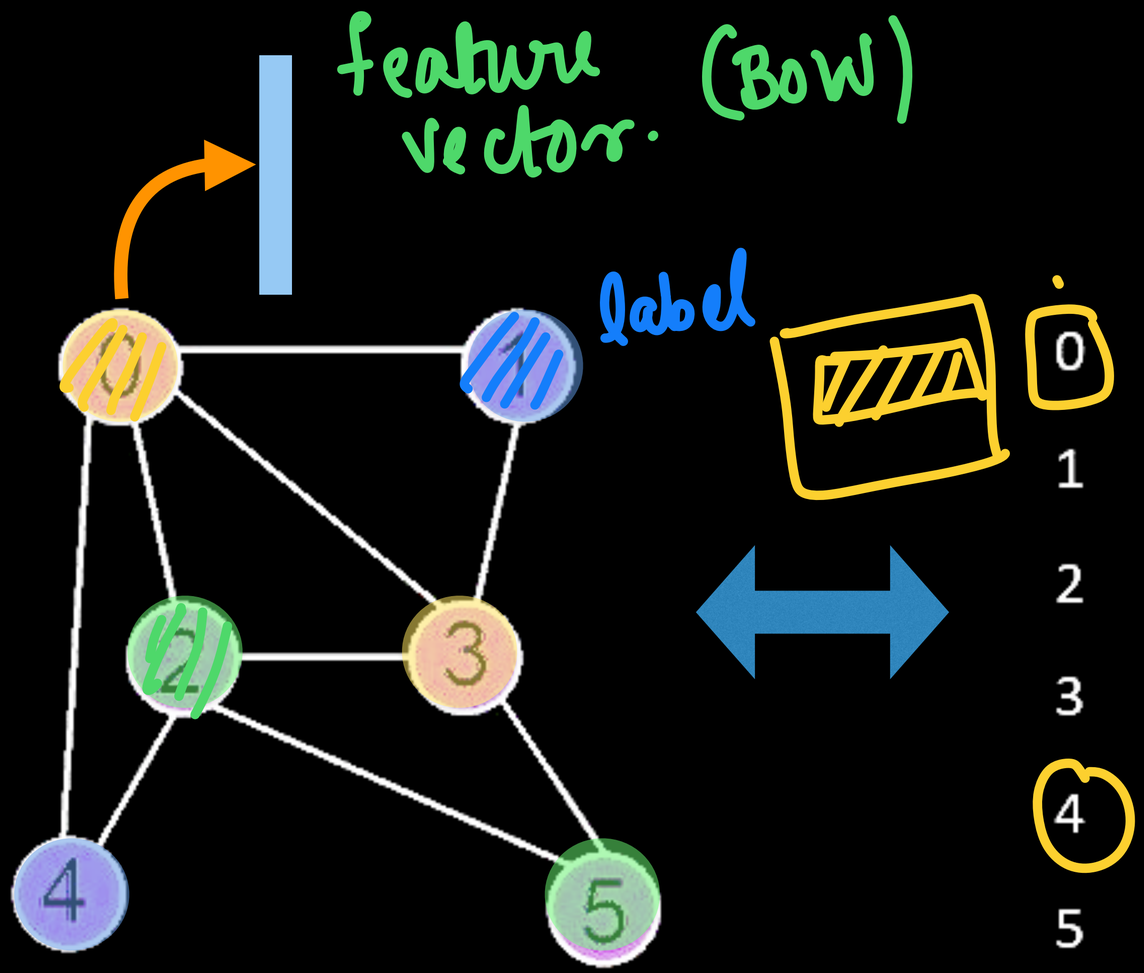
Step I

Feature smoothing

* Based on graph structure

Edge weight – Weight of connection

Representing relationships in the input as graph



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

Graph convolutional networks

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Simplifying Graph Convolutional Networks

Felix Wu [*1] **Tianyi Zhang** [*1] **Amauri Holanda de Souza Jr.** [*1 2] **Christopher Fifty** [1] **Tao Yu** [1]
Kilian Q. Weinberger [1]

GCN

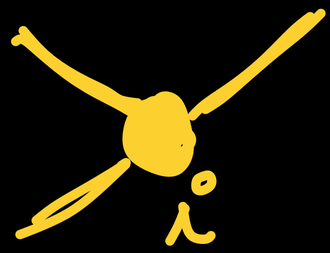
Graph definition (Undirected)

- (V,E) denotes the vertices and edges in a graph
  - ✓ |V| - denotes the number of vertices N
- $\mathbf{A} = [a_{ij}]$ denote the adjacency matrix of the graph
  - ✓ Similarity or affinity of the vertices. Can also be binary matrix.
  - ✓ Symmetric and typically sparse matrix
- $\mathbf{D} = diag[d_1, d_2, ..., d_N]$ denote the degree matrix

similarity measure

$a_{ij} \geq 0$

N×N matrix



$$d_i = \sum_j a_{ij}$$

$$D = \begin{bmatrix} d_1 & & 0 \\ & d_2 & \\ 0 & & \ddots \\ & & & d_N \end{bmatrix}$$

Defining graphs

- Input features

$$\mathbf{x}_i \in \mathcal{R}^D \quad i = 1 \ldots N$$

- Input feature space

$$\mathbf{X} \in \mathcal{R}^{N \times D}$$

N × D

- Hidden layer initialization

$$\mathbf{H}^0 = \mathbf{X}$$

zeroth hidden layer

3-steps in Graph convolutional networks

- I. Feature propagation

Filtering

$$\bar{\mathbf{h}}_i^{k} = \frac{\mathbf{h}_i^{k-1}}{d_i + 1} + \sum_{j=1}^{N} \frac{a_{ij}}{\sqrt{(d_i+1)(d_j+1)}} \mathbf{h}_j^{k-1}$$

$k = 1$

$\mathbf{h}_i^1 \in \mathbb{R}^D$

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

$$\bar{\mathbf{D}} = \mathbf{D} + \mathbf{I}$$

Adding self connections

Normalized Adjacency matrix

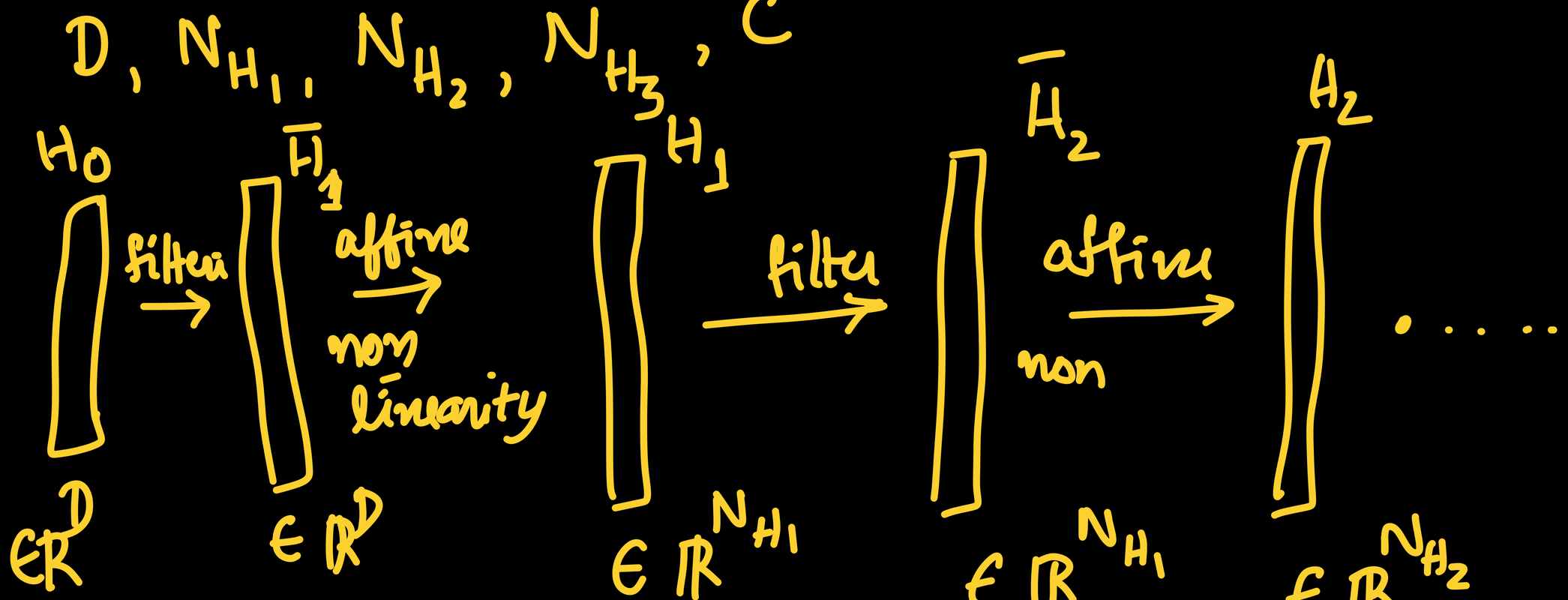
$$\mathbf{S} = \bar{\mathbf{D}}^{-\frac{1}{2}} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-\frac{1}{2}}$$

NXN

$$\bar{\mathbf{H}}^{l+1} = \mathbf{S}\mathbf{H}^{l}$$

not a learnable parameters

- Intuitively, this step smoothes the hidden representations locally along the edges of the graph and encourages similar predictions among locally connected nodes.


D, N_{H_1}, N_{H_2}, N_{H_3}, C
H_0
$\bar{H}_1$
H_1
$\bar{H}_2$
H_2
filter
affine
non linearity
filter
affine
non
$\in \mathbb{R}^D$
$\in \mathbb{R}^D$
$\in \mathbb{R}^{N_{H_1}}$
$\in \mathbb{R}^{N_{H_1}}$
$\in \mathbb{R}^{N_{H_2}}$
. . . .

Graph convolutions

- II. Weights and non-linearity

$$\mathbf{H}^{l+1} = \sigma(\bar{\mathbf{H}}^{l+1}\mathbf{W}^{l+1})$$

$l = 0$

learnable parameters

$\bar{H}^{l+1} \in \mathbb{R}^{N \times D}$, $l=0$

$W^{l+1} \in \mathbb{R}^{D \times N_{H_1}}$

$H^{l+1} \in \mathbb{R}^{N \times N_{H_1}}$

- Sigma is any non-linearity like sigmoid or ReLU
- A GCN layer is identical to a standard MLP. Each layer is associated with a learned weight matrix, and the smoothed hidden feature representations are transformed linearly.
- Finally, a nonlinear activation function such as ReLU is applied pointwise before outputting feature representation

Graph convolutions

- II. Output layer

$$\hat{\mathbf{Y}} = Softmax(\mathbf{S}\mathbf{H}^{L-1}\mathbf{W}^{L})$$

$\bar{H}^{L}$ $\in \mathbb{R}^{N_{H_{L-1}} \times C}$ $N \times C$ softmax on rows.

- where the predicted model outputs and the true targets are

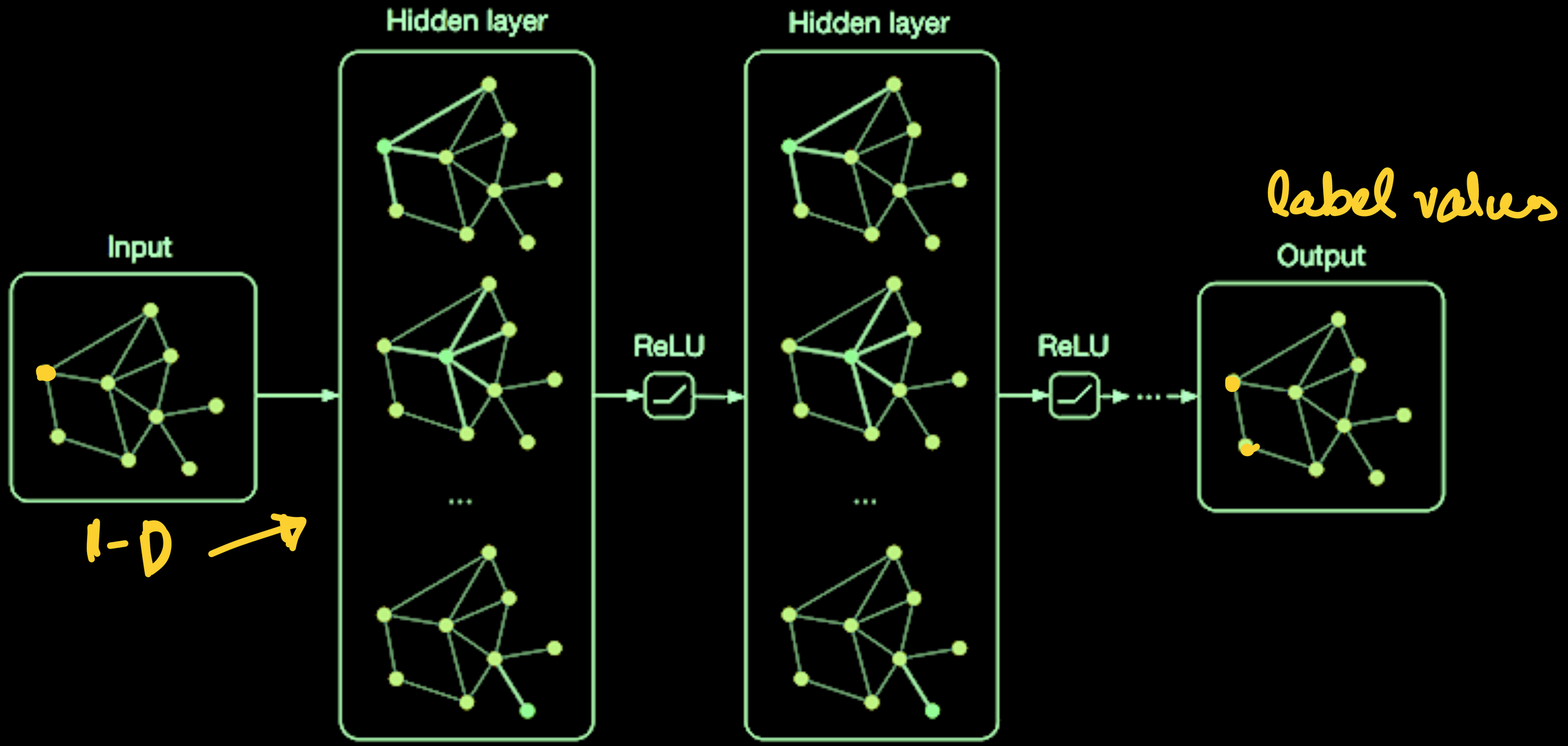
$$(\hat{\mathbf{Y}}, \mathbf{Y}) \in \mathcal{R}^{N \times C}$$

- The model can be trained using a cross entropy loss $E(Y, \hat{Y})$ y_i - i^{th} row
  - or other types of logistic loss functions

$$E(Y, \hat{Y}) = \sum_{i \in V_L} CE(y_i, \hat{y}_i)$$

V_L - subset of vertices that have labels

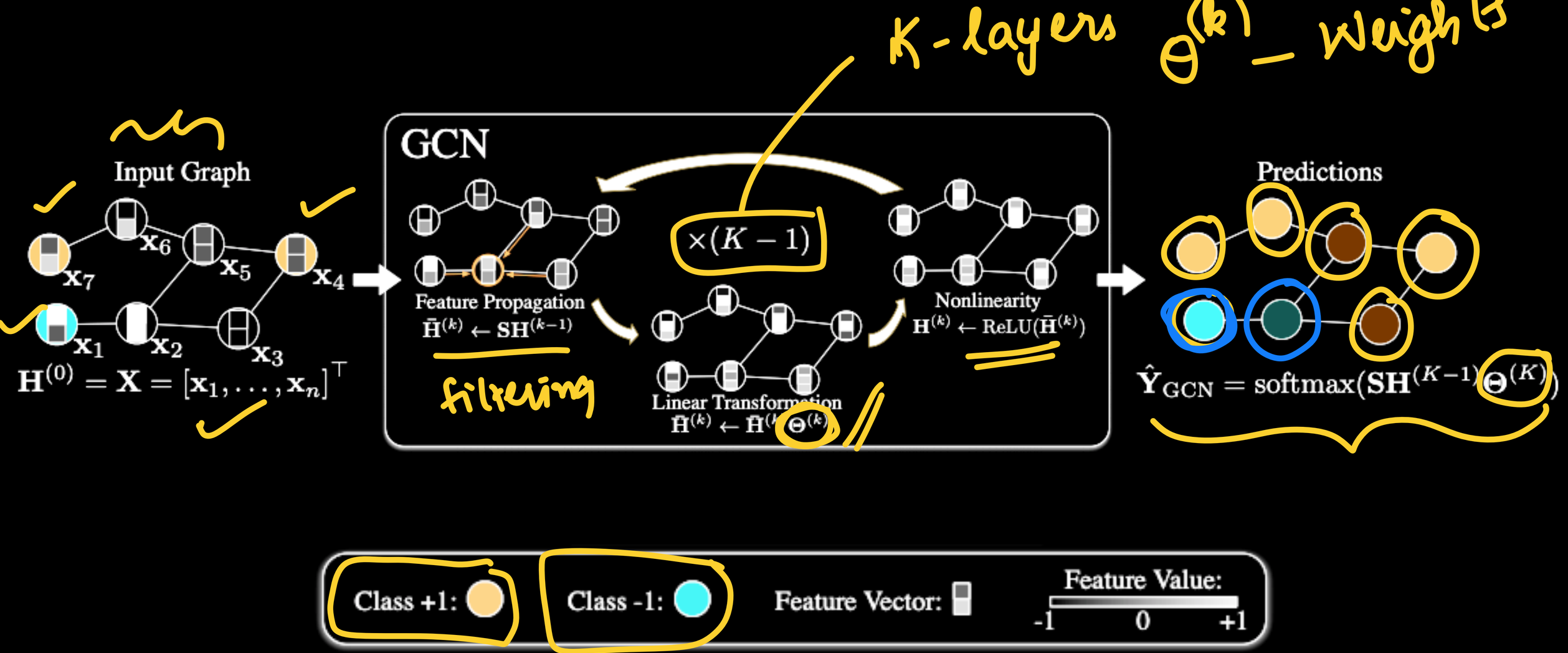
Overview of GCN



Application

- Note - GCN cannot be used for predictions of test data like standard MLP
  - The test data must be embedded in the graph and must be learned during training.
- Semi-supervised learning
  - Only a subset of the nodes have labels → part of adjacency matrix
    - ✓ Model can still be learned using loss function defined on the subset of the labelled nodes
    - ✓ Learning in GCN happens as full-batch learning on entire training set.

Semi-supervised learning using GCN



* Citation datasets

* Social networks ← Ads

* Blogs.

Theoretical results