Semi-Supervised Classification with Graph Convolutional Networks

Authors: Kipf, Thomas N., and Max Welling

Presenter: Parham Hamouni
Facilitators: Karim Khayrat

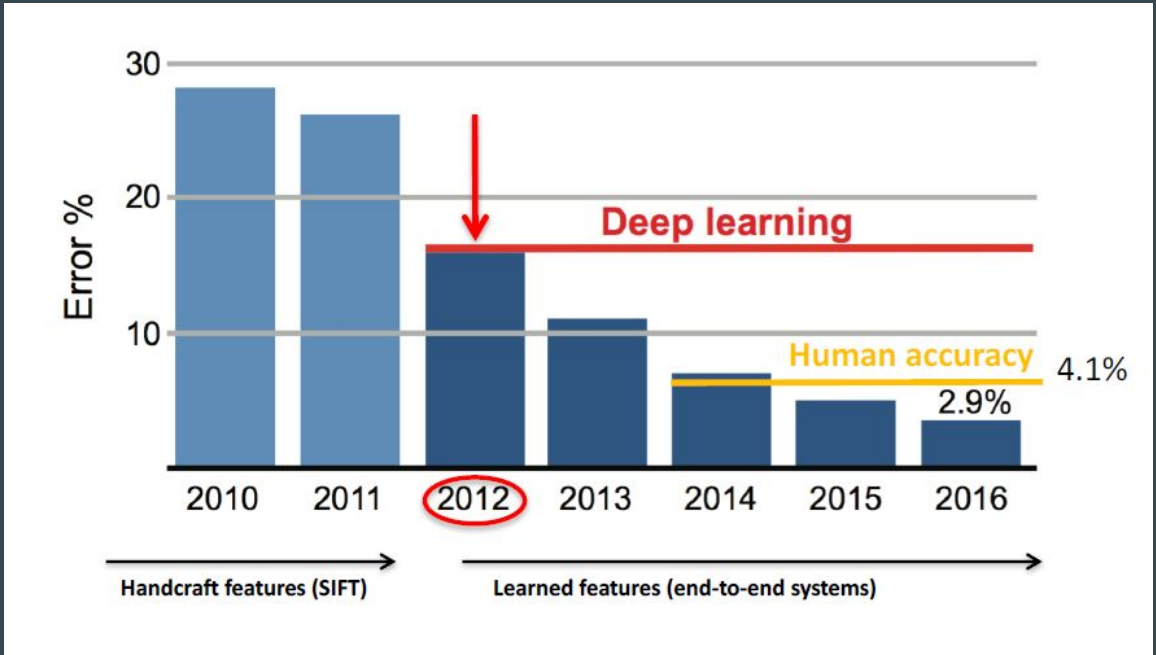
27 March 2019
Lunch and Learn

AISC

Overview

- CNN
- When CNNs are powerful?
- Euclidean vs Non-Euclidean
- CNN on Euclidean Data
- GCN
- Why GCN works?
- GCN as Weisefeiler-Lehman (WL)
- Experiment and results

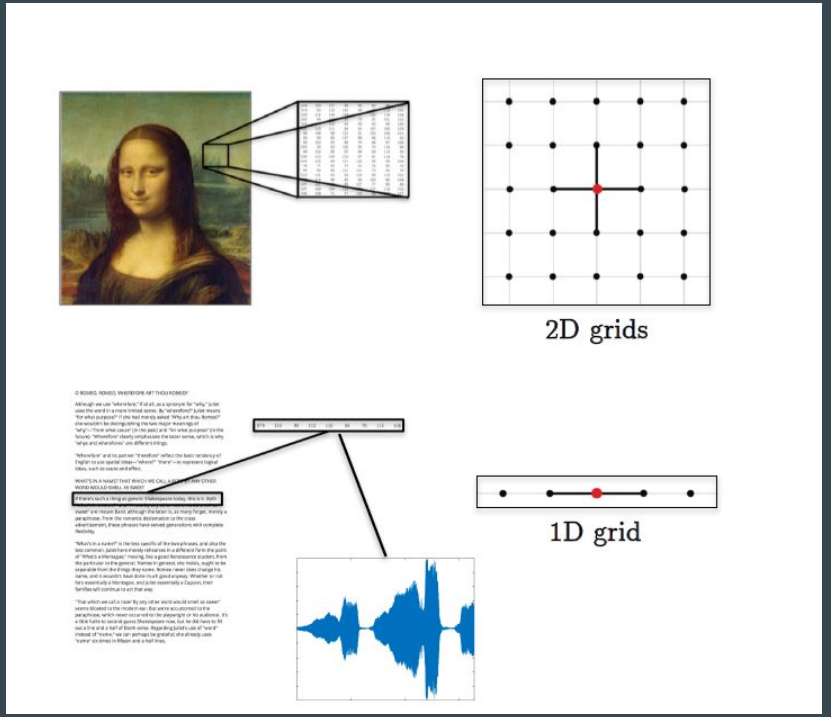
CNN



- CNNs are powerful to solve high dimensional learning problems.
- Is it possible to apply CNNs on Graphs?

When CNNs are powerful?

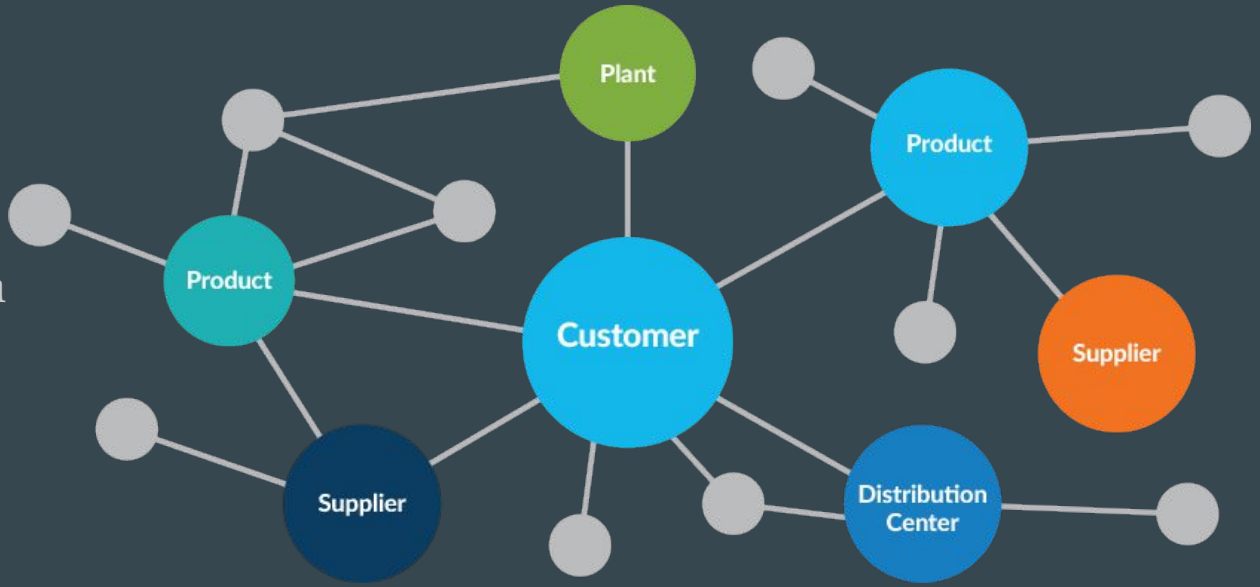
- Main assumption
  - Euclidean Data (images, videos, sounds) that are compositional
  - Extracting compositional features and feed them to classifier, etc. (end-to-end).
- Euclidean domain
  - These domains have nice regular spatial structures. (e.g. the distance between adjacent pixels are the same)
  - All CNN operations are math well defined and fast (convolution, pooling)



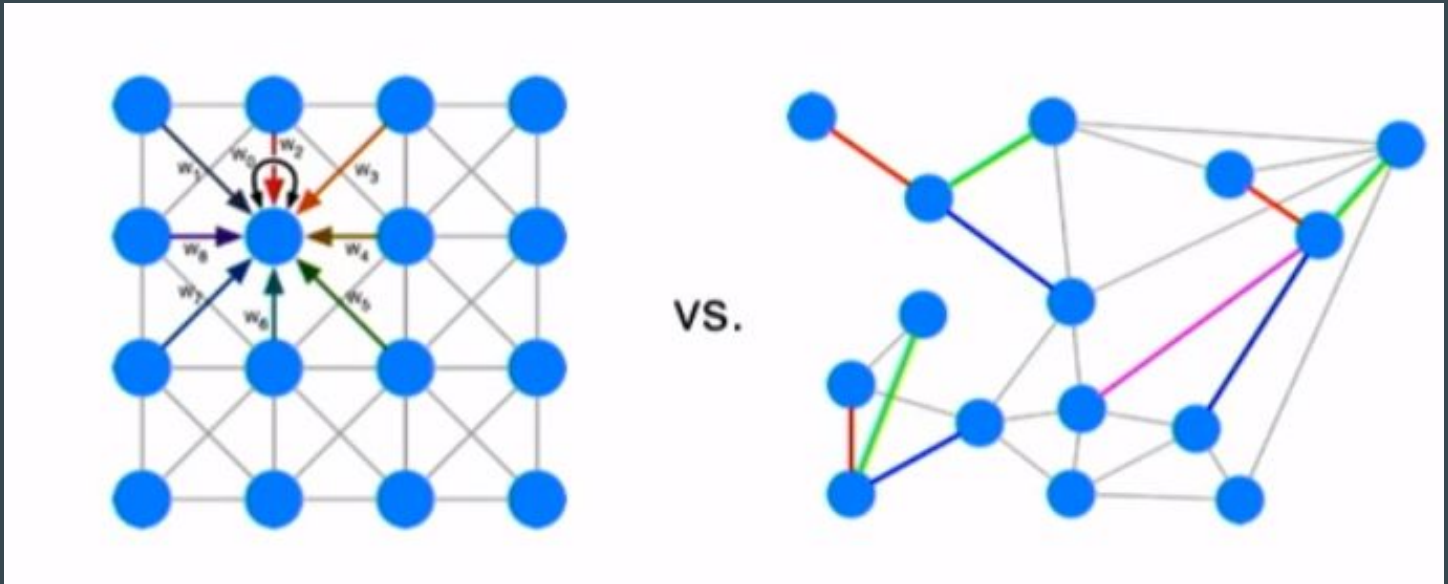
Non-Euclidean Data Domain

Examples include:

- Social Networks
- Web
- Molecule representation
- Knowledge graphs

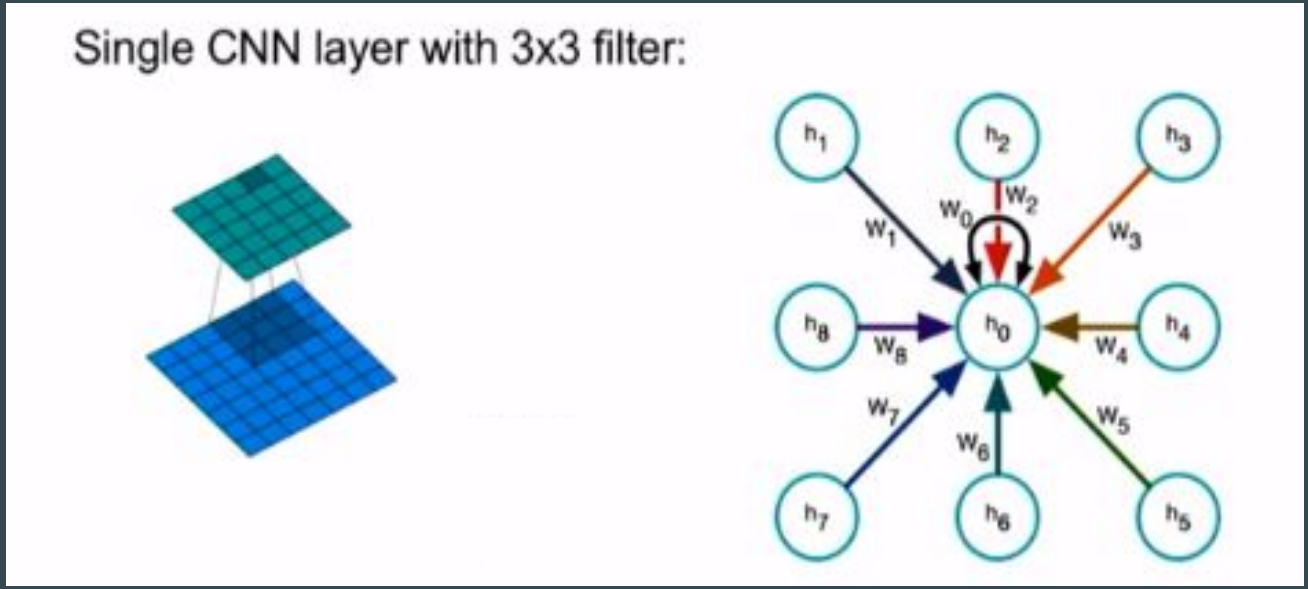


Euclidean vs Non-Euclidean Data Domain



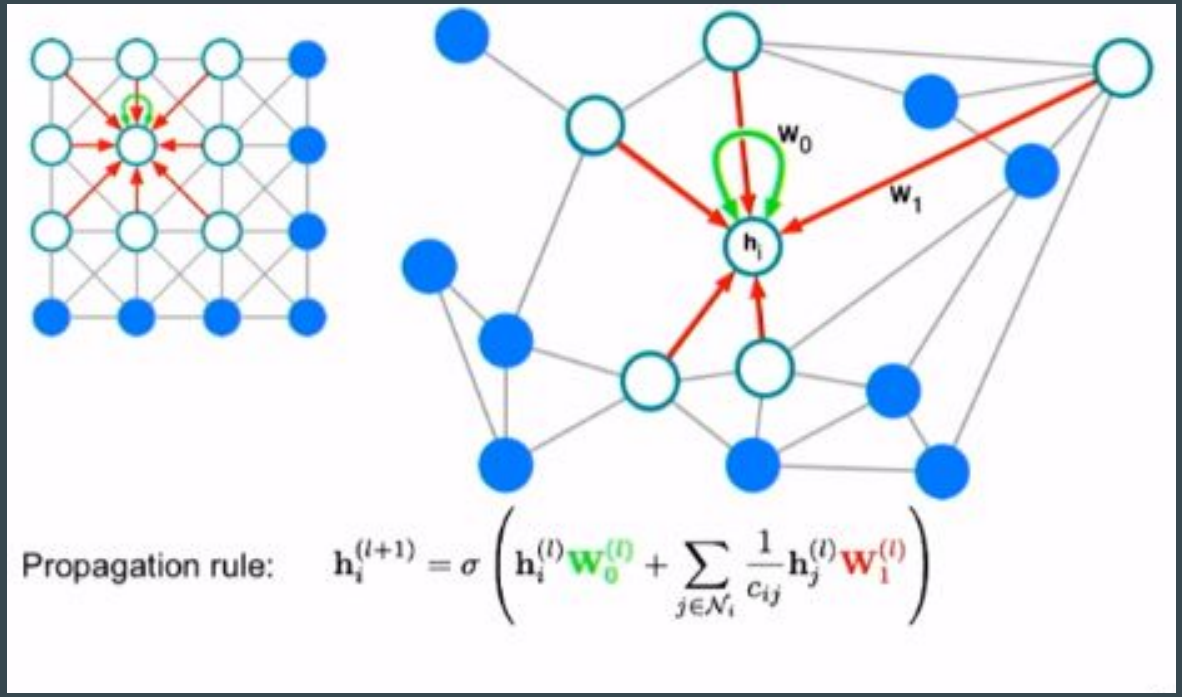
Source: Generalizing Convolutions for Deep Learning by Max Welling

CNN on Euclidean Data Domain



Source: Generalizing Convolutions for Deep Learning by Max Welling

Graph Convolutional Network (GCN)



Source: Generalizing Convolutions for Deep Learning by Max Welling

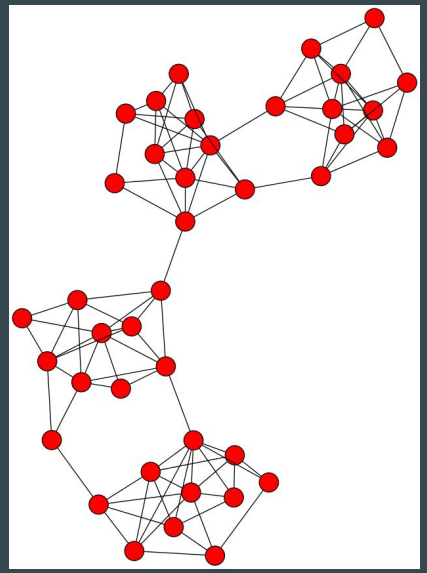
Why GCN works?

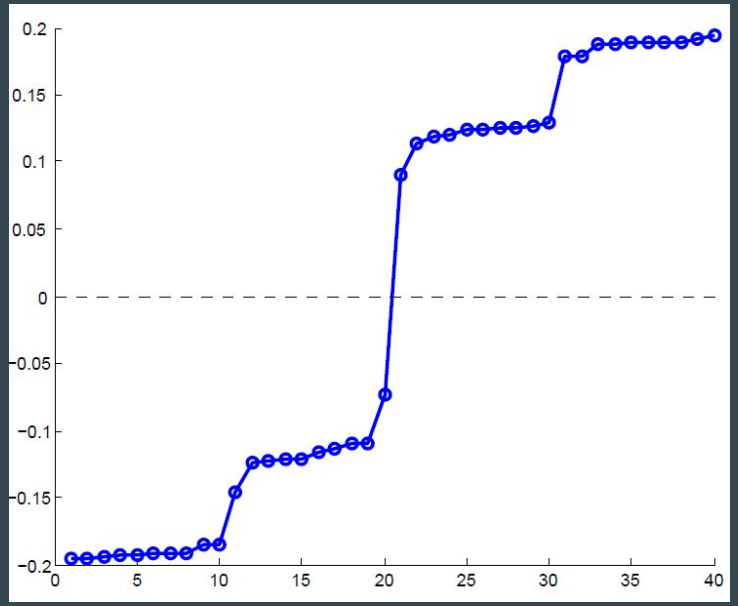
Spectral Graph Convolution on arbitrary kernel $g_\theta \star x = U g_\theta U^\top x\,,$

Spectral Graph Convolution on Graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$

A graph can be clustered by this operation, yet computationally expensive, as multiplication with the eigenvector matrix U is O(N2).

Spectral Graph Clustering





How to reduce the computational cost?

Approximation of Spectral Graph Clustering

- Well-approximated by a truncated expansion in terms of Chebyshev polynomials, ChebNet

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x$$

$$\tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

$$\theta' \in \mathbb{R}^K$$

Chebyshev coefficients

- GCN is the approximation of ChebNet!

$$\lambda_{\max} \approx 2,$$

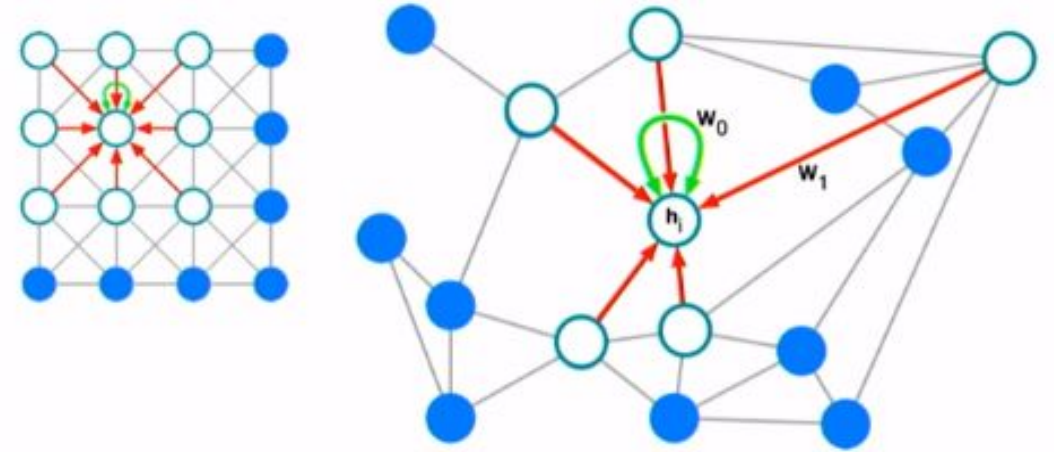
$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x$$

- Renormalization trick: $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- The complexity is now: $\mathcal{O}(|\mathcal{E}|FC)$

C-dimensional feature vector for every node,F filters or feature maps , E linear in the number of edges.

Graph Convolutional Network (GCN)

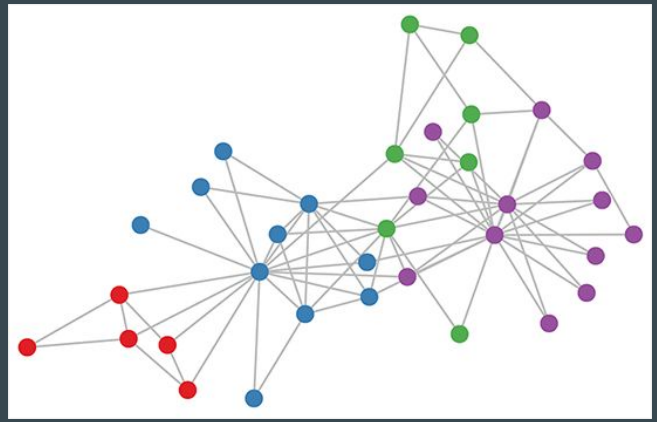


Propagation rule: $\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$

Source: Generalizing Convolutions for Deep Learning by Max Welling

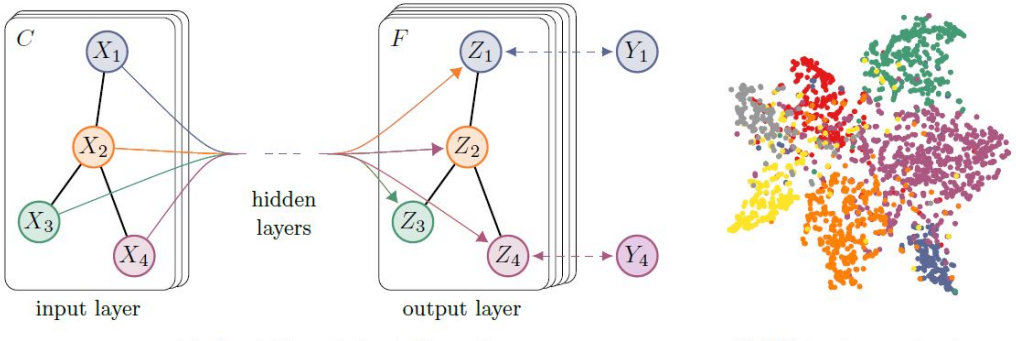
Semi-Supervised node classification task

- Assumption: Connected nodes can possibly share labels
  - Input: some labelled and unlabelled data in a graph
  - Output: predict the node label of unlabeled nodes
- Embedding based approaches:
  - Embed each node to a numerical Euclidean Space
  - Train the classifier on them
- With GCN, the training would be end to end!



Karate club graph, colors denote communities obtained via modularity-based clustering (Brandes et al., 2008).

Overall View of the GCN



(a) Graph Convolutional Network

(b) Hidden layer activations

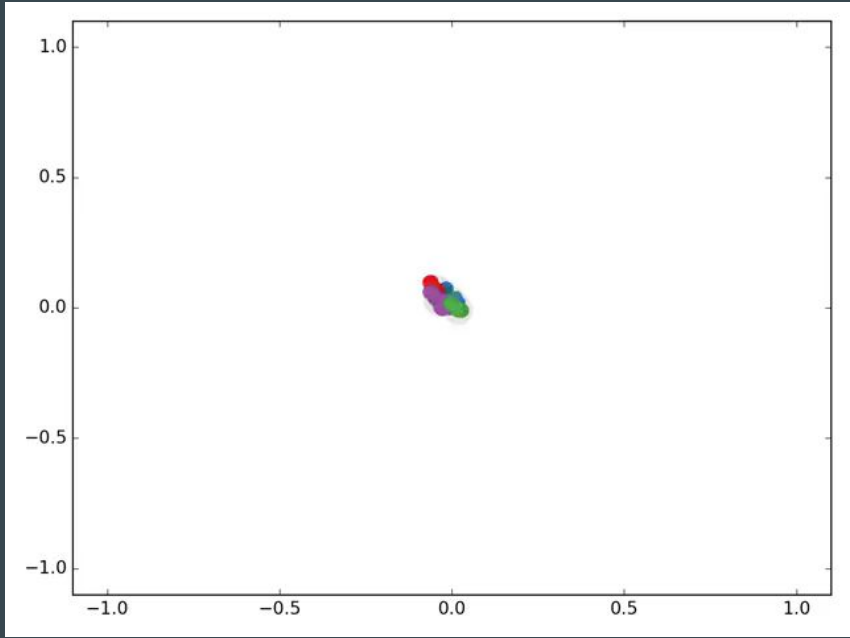
Loss Function

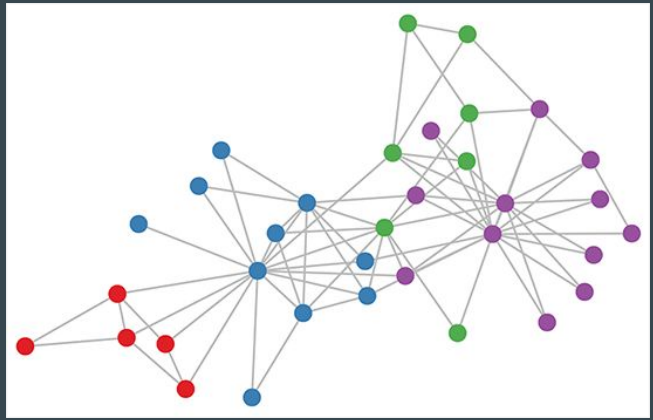
$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

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

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big)$$

GCN





GCN as WL?

Algorithm 1: WL-1 algorithm (Weisfeiler & Lehmann, 1968)

Input: Initial node coloring $(h_1^{(0)}, h_2^{(0)}, ..., h_N^{(0)})$

Output: Final node coloring $(h_1^{(T)}, h_2^{(T)}, ..., h_N^{(T)})$

$t \leftarrow 0$;

repeat

  for $v_i \in \mathcal{V}$ **do**

    $h_i^{(t+1)} \leftarrow \text{hash}\left(\sum_{j \in \mathcal{N}_i} h_j^{(t)}\right)$;

  $t \leftarrow t + 1$;

until *stable node coloring is reached*;

$$h_i^{(t+1)} \leftarrow \text{hash}\left(\sum_{j \in \mathcal{N}_i} h_j^{(t)}\right);$$

GCN

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W^{(l)}\right)$$

Experiment setup

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

Results

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

Limitations

- Memory requirement
- Directed edges and edge features
- Limiting assumptions

$$\tilde{A} = A + \lambda I_N$$

Key takeaways

- Graph and non-Euclidean domain needs special considerations to apply deep learning success
- GCN has mathematical explanation
- GCN is one of the most successful algorithms in deep learning on graphs

Discussion Points

1. Why not more layers, just two layers of convolution?
2. Why spectral graph clustering uses Graph Laplacian matrix?
3. How to make GCN scalable to big data?
4. How is it different from gated graph neural network?
5. More recent frameworks such as graph attention networks have improved the state-of-the-art...

Thank you

•••

Q&A

Spectral graph clustering

Three basic stages:

1. Pre-processing
    - Construct a matrix representation of the graph
2. Decomposition
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
3. Grouping
    - Assign points to two or more clusters, based on the new representation

Why eigenvalues and clustering?

- For a symmetric matrix:

$$\lambda_2 = \min_x \frac{x^T M\ x}{x^T x}$$

$$\mathrm{x}^{\mathrm{T}}\mathrm{L}\,\mathrm{x} = \sum_{i,j=1}^{n} L_{ij}\, x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$$

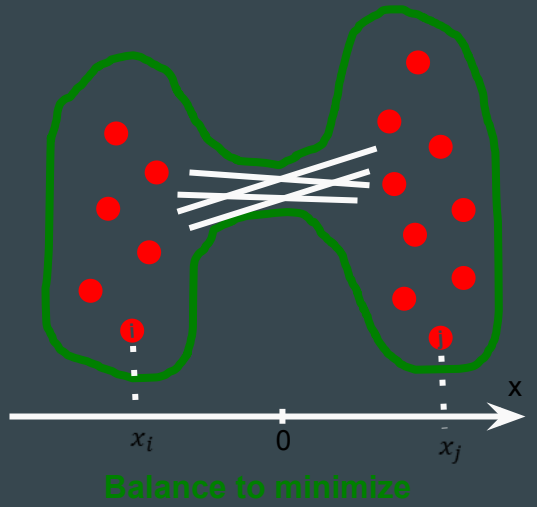
$$= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$$

$$= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2x_i x_j) = \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$$

Why eigenvalues and clustering?

- x is unit vector: $\sum_i x_i^2 = 1$
- x is orthogonal to 1st eigenvector $(1, \ldots, 1)$ thus: $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$



Multiple partitioning

- **Two basic approaches:**
  - **Recursive bi-partitioning** [Hagen et al., ’92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - **Cluster multiple eigenvectors** [Shi-Malik, ’00]
    - Build a reduced space from multiple eigenvectors
    - Commonly used in recent papers
    - A preferable approach…