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

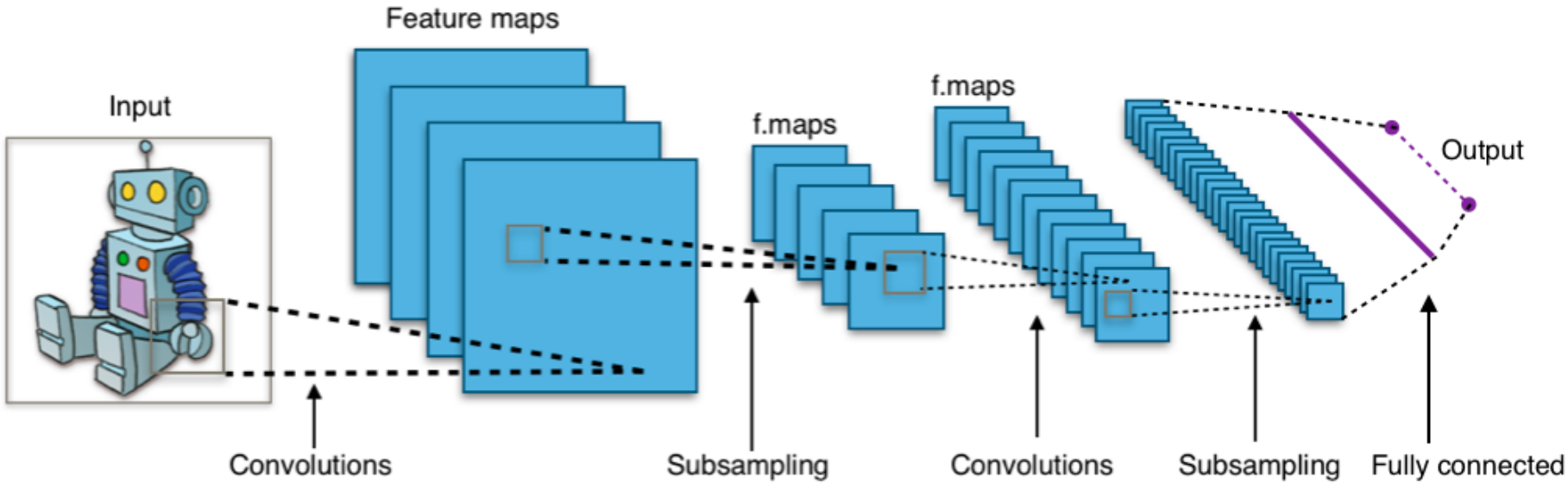
ABSTRACT

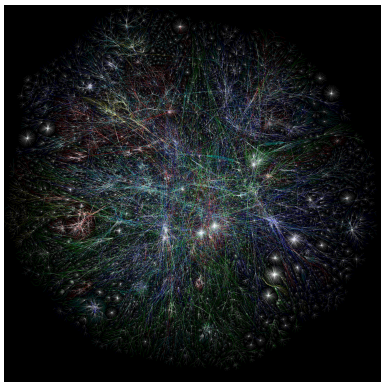
We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.

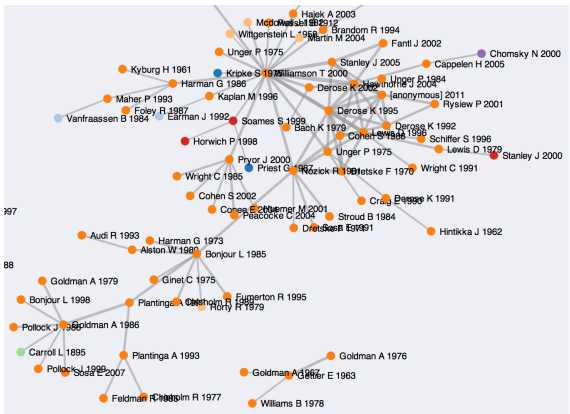
Pierre-Simon Laplace

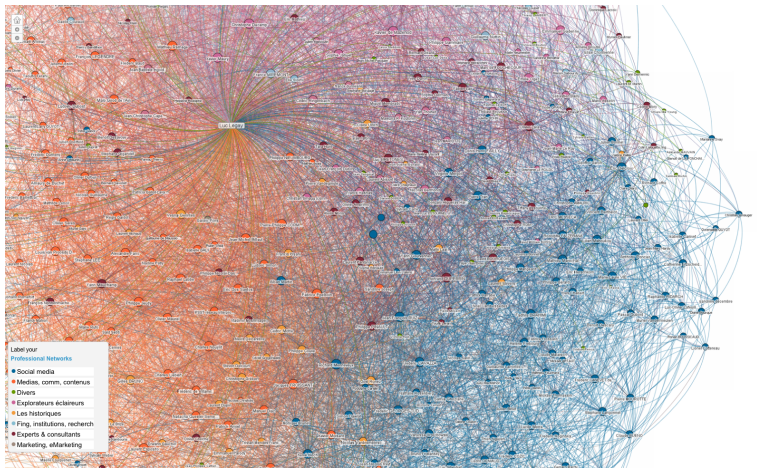
December 4, 2017

Why Do We Need Graph Convolutional Networks





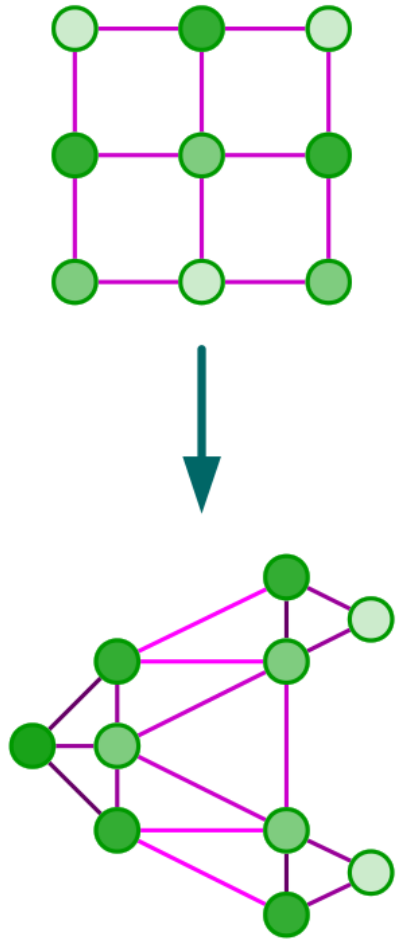




Why Do We Need Graph Convolutional Networks (cont'd)

Graphs vs Euclidean grids

- Irregular sampling
- Weighted edges
- No orientation (in general)

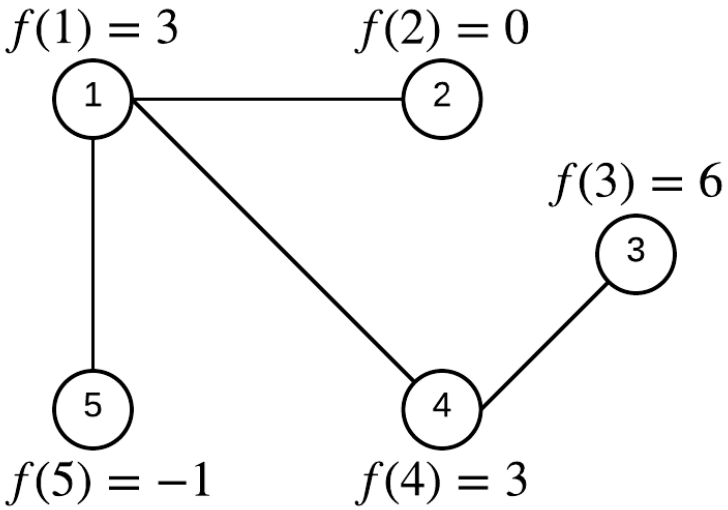


Challenges

1. Formulate convolution on graphs
2. Make them efficient

Outline

Graph Gradient



$$f : \mathcal{V} \to \mathbb{R}$$
$$\nabla f = (f(v) - f(w))_{v,w \in E}$$

$$E = \{(1,2), (1,4), (1,5), \mathbf{(3,4)}\}$$
$$\nabla f = (-3, 0, -4, \mathbf{3})$$

Limitations:

1. Directions arbitrarily
   - Numerical properties of the gradient wouldn't be well-defined
2. Fix your reference point as a single vertex
   - Gradient on edges going "out of" that vertex

Graph Gradient (cont'd)

Euclidean norm of the gradient:

$$\sum_{(v,w)\in E} (f(v) - f(w))^2 = x^T \boldsymbol{A} x$$

Expressing a Quadratic Form with a Matrix

$$\begin{aligned} ax^2 + 2bxy + cy^2 &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} ax + by \\ bx + cy \end{bmatrix} \\ &= x\,(ax + by) + y\,(bx + cy) \\ &= ax^2 + 2bxy + cy^2 \end{aligned}$$

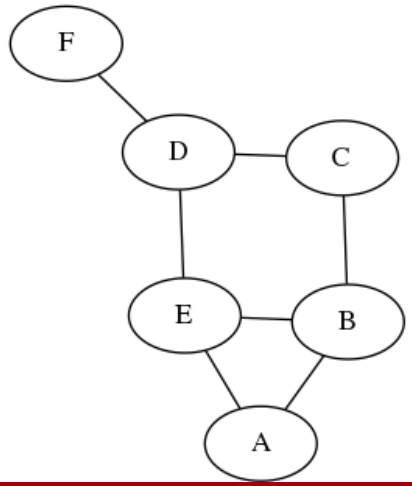
Graph Laplacian

$$L_{i,j} = \begin{cases} \deg(v_i) & i = j \\ -A_{ij} & v_i \sim v_j \\ 0 & \text{otherwise} \end{cases}$$

$$L = D - A \in \mathbb{R}^{n \times n} : \textit{combinatorial}$$

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} : \textit{normalized}$$

$$= I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Degree Matrix

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency Matrix

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \end{array}$$

Laplacian Matrix

Linear Algebra

Definition

A matrix A is orthogonally diagonalizable if and only if there is an orthogonal matrix U such that $A = UDU^{-1}$ where D is a diagonal matrix.

Remark

Recall that any orthogonal matrix U is invertible and also that $U^{-1} = U^T$. Thus we can say that a matrix A is orthogonally diagonalizable if there is a square matrix U such that $A = UDU^T$ where D is a diagonal matrix.

Remark

Any orthogonally diagonalizable A must be symmetric.

$$A^T = (UDU^T)^T = ((U^T)^T D^T U^T = UDU^T = A$$

Graph Fourier Transform

L is symmetric and positive semidefinite (PSD) $\rightarrow L = U\Lambda U^T$ *(EVD)*

- Graph Fourier basis $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$
- Graph frequencies $\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$

Graph Fourier Transform (GFT):

- Graph signal $f : \mathcal{V} \rightarrow \mathbb{R}$ seen as $\mathbb{R}^n$
- Transform: $\hat{f} = \mathcal{F}_{\mathcal{G}}\{f\} = U^T f \in \mathbb{R}^n$
- Inverse: $f = U\hat{f} = UU^T f = f$

Graph Convolution

Convolution

$$u[t] * v[t] = \sum_{k=-\infty}^{\infty} u[k]v[t-k]$$

$$u[t] * v[t] \Leftrightarrow U[f]V[f]$$

$$x *_{\mathcal{G}} g = \underbrace{U\underbrace{(U^T g \odot U^T x)}_{\mathcal{O}(n^2)}}_{\mathcal{O}(n^3)}$$

$$= U(\hat{g} \odot U^T x)$$

Conveniently written as:

$$x *_{\mathcal{G}} g = U \begin{bmatrix} \hat{g}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_n) \end{bmatrix} U^T x$$

$$= U\hat{g}(\Lambda)U^T x$$

$$= \hat{g}(L)x$$

Graph Convolution (cont'd)

$$y = \hat{g}_\theta(L)x = U\hat{g}_\theta(\Lambda)U^T x$$

Non-parametric filter:

$$\hat{g}_\theta(\Lambda) = \text{diag}(\theta), \theta \in \mathbb{R}^n$$

- Non-localized in vertex domain
- Learning complexity is $\mathcal{O}(n)$

Localized Filters

$$\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k \Lambda^k, \theta \in \mathbb{R}^K$$

Hammond et al. (2011)

- Value at j of g_θ centered at i: $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$
- $d_{\mathcal{G}}(i,j) > K$ implies $(L^K)_{i,j} = 0$
- K-localized
- Learning complexity is $\mathcal{O}(K)$

Recursive Formulation for Fast Filtering

$$\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\hat{\Lambda}), \quad \tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$$

- Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$
- Filtering: $y = \hat{g}_\theta(L)x = \sum_{k=0}^{K} \theta_k T_k(\tilde{L})x$
- Recurrence: $y = \hat{g}_\theta(L)x = [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta$ where $\bar{x}_k = T_k(\tilde{L})x = 2L\bar{x}_{k-1} - \bar{x}_{k-2}$ with $\bar{x}_0 = x$ and $\bar{x}_1 = \tilde{L}x$
- Computational complexity is $\mathcal{O}(K\mathcal{E})$

Recursive Formulation for Fast Filtering (cont'd)

When $K = 1$:

- Alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions
- Allows us to build deeper models
  - Linear w.r.t. L and therefore a linear function on the graph Laplacian spectrum

$$y = \hat{g}_\theta(L)x = \theta_0 x + \theta_1(L - I_N)x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$
$$= \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

Recursive Formulation for Fast Filtering (cont'd)

Constrain number of parameters to address overfitting by $\theta = \theta_0 = -\theta_1$

Renormalization trick: $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$

- $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvalues in $[0, 2]$
- Repeated application of this operation causes vanishing gradients
- $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

General Formula

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

where $X \in \mathbb{R}^{N \times C}$ and $\Theta \in \mathbb{R}^{C \times F}$

Example

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

- $W^{(0)} \in \mathbb{R}^{C\times H}$: input-to-hidden with H feature maps
- $W^{(1)} \in \mathbb{R}^{H\times F}$: hidden-to-output

Cross-entropy error over all labeled examples:

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

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

Example (cont'd)

$W^{(0)}$ and $W^{(1)}$ are trained using batch gradient descent using the full dataset

- Sparse representation of A requires $\mathcal{O}(\mathcal{E})$ memory
- Dropout is used for stochasticity in the training process

Node Classification

Problem

Classifying nodes in a graph where labels are only available for a small subset of nodes

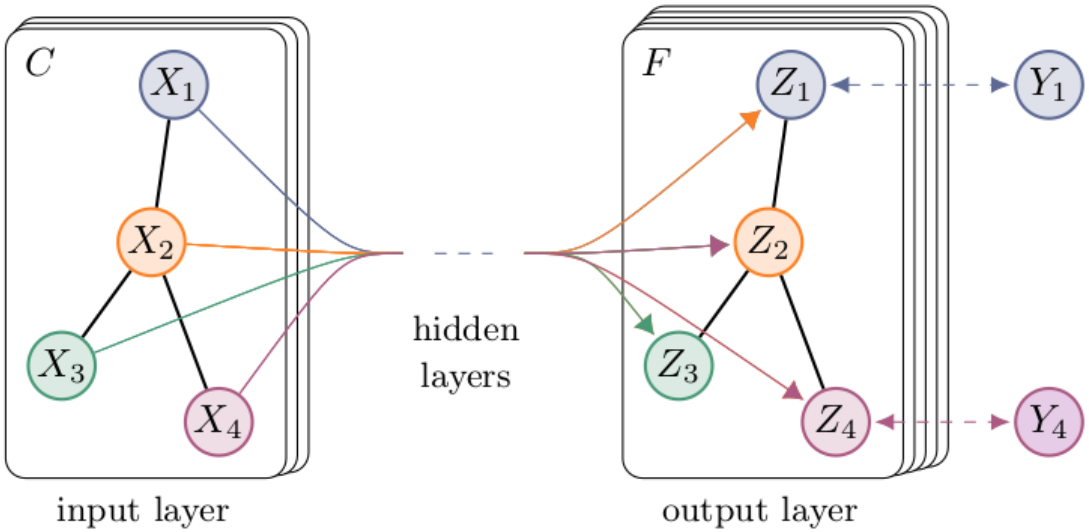
$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg}$$

$$\mathcal{L}_{reg} = \sum_{i,j} A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^T \Delta f(X)$$

Idea

- Encode the graph structure directly using a neural network model $f(X, A)$
- Train on a supervised target $\mathcal{L}_0$ for all nodes with labels
- Graph will allow the model to distribute gradient information from the $\mathcal{L}_0$
  - Learn representations of nodes both with and without labels

Fast Approximate Convolutions on Graphs



$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

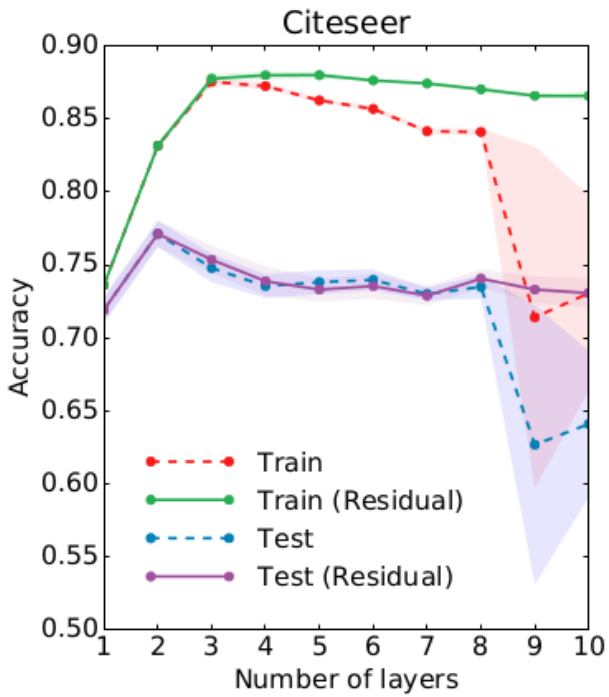
- $\tilde{A} = A + I_N$
- $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- $W^{(l)}$ is a trainable weight matrix
- $\sigma(\cdot)$ is an activation function
- $H^{(l)} \in \mathbb{R}^{N \times D}$; $H^{(0)} = X$

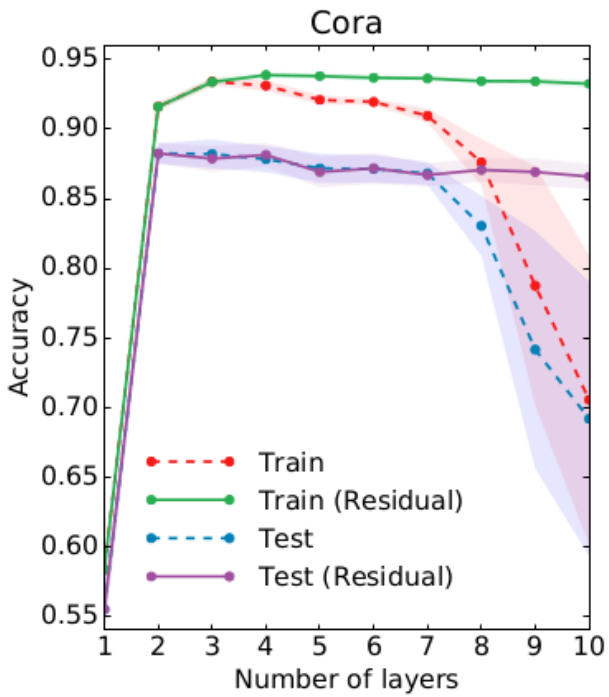
Experiments

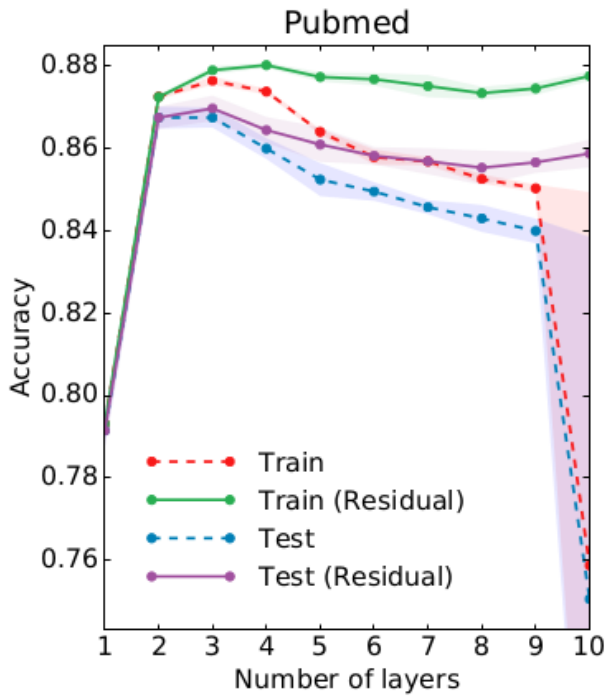
Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

- Citation networks:
  - Documents are nodes,citation links are edges (undirected)
  - Feature vectors: Sparse bag-of-words
  - Sample 20 nodes uniformly at random for each class as training data, but all feature vectors
- NELL:
  - Nodes: 55864 (relation) + 9891 (entity) = 67755
  - $(e_1, r, e_2) \rightarrow (e_1, r_1) \cup (e_2, r_2)$
  - 61278-dim sparse feature vector per node (one-hot)
  - Construct binary, symmetric adjacency matrix ($A_{ij} = 1$ if one or more edges between nodes i and j

Experiments (cont'd)







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)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

- Citeseer, Cora, and Pubmed: 0.5 (dropout rate), $5 * 10^{-4}$ (L2 regularization), 16 (number of hidden units)
- NELL: 0.1 (dropout rate), $1 * 10^{-5}$ (L2 regularization), 64 (number of hidden units)

Results (cont'd)

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
1^{st}-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**
1^{st}-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

Discussion

- Graph-Laplacian regularization are limited due to their assumption that edges encode mere similarity of nodes
  - Zhu et al. (2003)
  - Belkin et al. (2006)
  - Weston et al. (2012)
- Skip-gram based methods
  - multi-step pipeline which is difficult to optimize
  - Mikolov et al. (2013), Perozzi et al. (2014), Tang et al. (2015), Grover and Leskovec (2016)
- Proposed method supports propagation of feature information from neighboring nodes
  - improves classification accuracy
  - Only label information is aggregated in ICA

Limitations and Future Work

- Memory requirement: Full-batch gradient descent
  - Mini-batch stochastic gradient descent is a future work
- Does not support edge features, and limited to undirected graphs
  - Solution: Represent the original directed graph as an undirected bipartite graph (NELL case)
- Equal importance of self-connections vs. edges to neighboring nodes

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

Questions

Questions?

References I

Belkin, M., P. Niyogi, and V. Sindhwani (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research 7*(Nov), 2399–2434.

Grover, A. and J. Leskovec (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM.

Hammond, D. K., P. Vandergheynst, and R. Gribonval (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis 30*(2), 129–150.

References II

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119.

Perozzi, B., R. Al-Rfou, and S. Skiena (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM.

Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee.

References III

Weston, J., F. Ratle, H. Mobahi, and R. Collobert (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer.

Zhu, X., Z. Ghahramani, and J. D. Lafferty (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919.