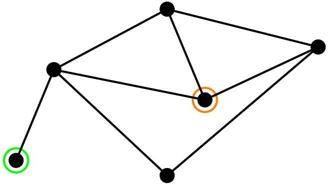
Semi-supervised Classification with Graph Convolutional Neural Networks

by Thomas N. Kipf and Max Welling

Daan Michiels

February 21, 2018

Semi-supervised classification on graphs



undirected, connected
N nodes
each node has C features

some nodes are labeled
want to label the rest

Existing approach:
smoothing labels over graph

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

Supervised loss

$$\sum_{ij} A_{ij} \| f(x_i) - f(x_j) \|^2$$

Idea:
CNN on graphs
(filtering? convolution?)

Proposal: spectral convolutions, to first order (kind of)

Convolutions on graphs: expensive $O(N^2)$

$\rightarrow$ Approximate, in this case "to first order"
(in Fourier domain, multiply by linear function)

Important property: only need a node and its neighbors!

After *renormalization trick*, get

activation of layer l

e.g. ReLU

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

trainable weights

and $H^{(0)} = X$, the node features.

$\tilde{A} = A + I_N$

$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

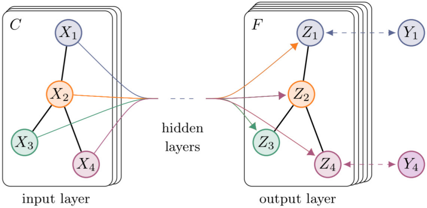
Classification setup

Calculate $\overset{N\times N}{\hat{A}} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ in pre-processing.
Two-layer model for classification:

$$Z = \mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\underbrace{\hat{A}X}_{N\times C}\underbrace{W^{(0)}}_{C\times H}\right)\overbrace{W^{(1)}}^{H\times F}\right).$$

Time complexity: $O(|\mathcal{E}|CHF)$

Batch gradient descent
Dropout
TensorFlow on GPU



Experiments

Three types: citation networks, knowledge graph, random graphs

Table 2: Summary of results in terms of classification accuracy (in percent).

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

Experiments

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)		$X\Theta_0 + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X \Theta$	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		$D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X \Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

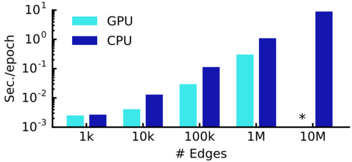


Figure 2: Wall-clock time per epoch for random graphs. (*) indicates out-of-memory error.

Limitations and future work

- Memory requirement
- No directed graphs, edge features
- Assumption of locality
- Importance of self-connections vs. edges? ($\tilde{A} = A + \lambda I_N$)
- Poor performance for highly regular graphs (e.g. image classification)

Reference: arXiv:1609.02907

Questions?