Graph Convolution Networks

Arka Sadhu

IIT Bombay

September 1, 2017

Outline

Outline I

Introduction to Graph Convolutional Networks

- CNN are extremely efficient architectures for image and audio classification tasks.
- But CNN donot directly generalize to irregular domains such as graph.
- Want to generalize CNN to Graphs.
- Non-trivial because the distances are non-euclidean.

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.
  - Learning Convolutional Neural Networks for Graphs [ICML 2016].

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.
  - Learning Convolutional Neural Networks for Graphs [ICML 2016].
- Spectral Approach :
  Using the frequency characterization of CNN and using that to generalize to Graphical domain
  - Spectral Networks and Deep Locally Connected Networks on Graphs [Bruna et al. ICLR 2014].
  - Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering [Defferrard et al. NIPS 2016] (will be the main focus)
  - Semi-Supervised Classification with Graph Convolutional Networks [Kipf et al. ICLR 2017]

Outline I

Limitations of Spatial Approach

- Can't exactly define a neighborhood because the distances are not uniform.
- Ordering of nodes is problem specific.

Hence for the remainder we discuss the Spectral Approach

Outline I

A Basic Formulation

- Convolution in spectral (Fourier) domain is point wise multiplication.
- Fourier Basis is defined as the eigen basis of the laplacian operator.
- Can use Laplacian of a graph.

Defining the Problem on Graphs

- A feature description x_i for every node i; summarized in a NxD feature matrix X (N : number of nodes, D : number of input features)
- Adjacency Matrix A.
- Node level output Z (an NxF feature matrix, where F = number of output features per node).
- Every neural network can then be written as a non-linear function.

$$H^{(l+1)} = f(H^l, A)$$

Brief overview of Graph Laplacian

Let T denote the diagonal matrix with (v,v)-th entry having value d_v : degree of vertex v. Define L-matrix as

$$L(u,v) = \begin{cases} d_v & \text{if u = v} \\ -1 & \text{if u and v are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

And the Laplacian of the graph as

$$\mathcal{L}(u,v) = \begin{cases} 1 & \text{if u = v} \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if u and v are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Graph Laplacian (contd.)

$$\mathcal{L} = T^{-1/2} L T^{1/2}$$

With the convention $T^{-1}(v, v) = 0$ for $d_v = 0$.
When G is k-regular,

$$\mathcal{L} = I - \frac{1}{k} A$$

For a general graph

$$\mathcal{L} = I - T^{-1/2} A T^{1/2}$$

Outline I

Spectral Networks and Deep Locally Connected Networks on Graphs

- Mentions the use of both spatial and spectral construction.
- For the spectral part uses a spline and has k control points for it.

$$g_\theta(\Lambda) = B\theta$$

  Here B is the cubic B-spline basis and θ is a vector of control points.
- The datasets used (created) are quite interesting. Subsampled MNIST and MNIST on sphere to show how spectral networks can be used on graphs.

Outline I

Graph Fourier Transform

- Laplcian of the graph is real symmetric positive semidefinite, and thus can be written as

$$L = U \Lambda U^T$$

- Here $U = [u_0 .... u_{n-1}]$ is the fourier basis and $\Lambda = diag([\lambda_0 ... \lambda_{n-1}])$ are ordered real non-negative eigen values.
- Graph Fourier Transform of a signal x is $\hat{x} = U^T x$.

Spectral filtering of graph signals

- Defining convolution on graphs

$$x *_G y = U((U^T x) \odot (U^T y))$$

- Filtering by g_θ

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = U g_\theta(\Lambda) U^T x$$

- A non-parametric filter (all parameters free) would be defined as

$$g_\theta(\Lambda) = diag(\theta)$$

Polynomial Parametrization

- Problem with non-parametric filters is that not localized (we want something like k-neighborhood) and therefore their learning complexity becomes $O(n)$. This can be overcomed with use of a Polynomial filter

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$$

- The advantage we gain here is that nodes which are at a distance greater than K away from the node i, at which the filter is applied, are not affected. Hence we have gained localization.

Recursive formulation for fast filtering

- Still cost to filter is high $O(n^2)$ because of multiplication with U matrix.
- Therefore use recurrence relation of chebyshev polynomial instead.

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_K(\tilde{\Lambda})$$

  Here $\tilde{\Lambda}$ is scaled between $[-1, 1]$.
- This allows us to compute $\bar{x_k} = T_K \tilde{L} x$. And Therefore

$$y = g_\theta(L)x = [\bar{x_0} ... \bar{x_{k-1}}]\theta$$

- The cost is now $O(K|E|)$

Learning filters

- Trivial to show that backprop calculation can be done efficiently.

Graph Coarsening and Pooling

- Require efficient mechanism for pooling. Graph clustering as such is NP-hard and some approximations must be made.
- The paper uses Graclus algorithm for coarsening, and uses an intelligent way of rearranging the nodes [creating a balanced binary tree from the remaining singleton and fake nodes] so that the pooling now becomes equivalent to pooling a regular 1D signal.

MNIST results

- Achieves close to classical CNN accuracy.

Table: MNIST performance

accuracy		loss		name
test	train	test	train	
98.87	99.62	1.02e+00	9.99e-01	cgconv_cgconv_fc_softmax
98.00	99.26	6.52e-02	2.77e-02	cgconv_softmax
96.75	96.78	1.12e+00	1.12e+00	fgconv_fgconv_fc_softmax
95.91	95.50	1.44e-01	1.53e-01	fgconv_softmax
97.66	97.79	1.09e+00	1.08e+00	sgconv_sgconv_fc_softmax
96.95	97.27	1.03e-01	9.46e-02	sgconv_softmax
92.18	92.47	3.14e-01	3.14e-01	softmax

Accuracy and Loss Plots

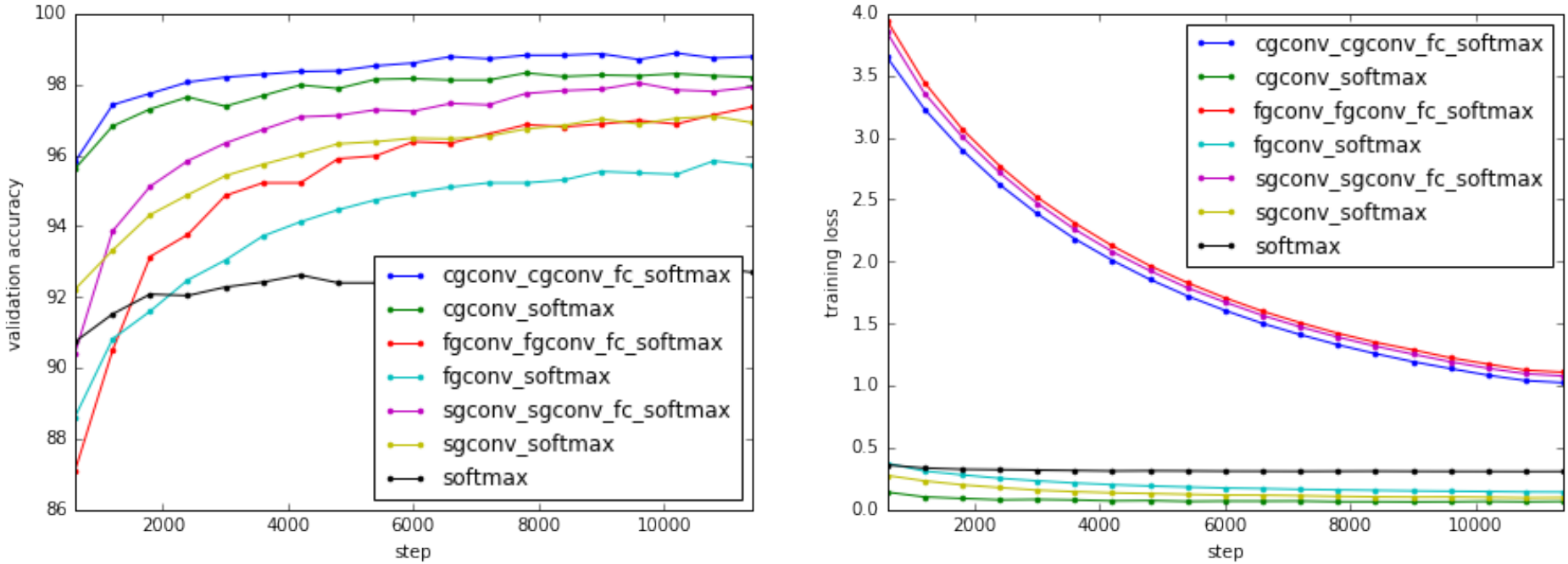

Figure: Accuracy and Loss function plot

Outline I

Fast approximate convolutions on the graph

- Layerwise propagation rule as

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

  Here $\tilde{A} = A + I_N$ the adjacency matrix with added self-loops. $\tilde{D}$ is the degree matrix.

- In the chebyshev approximation, limit to $K = 1$ and therefore the layer-wise convolution operation is linear function of the laplacian.

- Experimentally shows 2-3 layered GCN can effectively learn standard graph problems. Specifically it does decently well in the unsupervised case, and significantly good in the semi-supervised setting.

Outline I

Limitations

- We would like to model a network which doesn't require a rectangular input size and therefore be able to accomodate superpixels. In theory this should be possible, but there are a few points that we need to keep in mind.
  - Graph doesn't have orientation. There is no sense of up, down, left or right. The filters are rotationally invariant. This can be both advantageous as well as disadvantageous depending on the set-up of the problem. Spatial Transformer Networks learn the invariance to rotation as well as generic warping. But there is always the problem of '6' and '9' because they are equivalent in modulo rotation.
  - The filters are not directly transferable to another graph (because of the graph laplacian).