Deep Networks on Graph-Structured Data

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

1 Introduction

Generalizing convolutional networks to graph-structured data is an important open problem. Thus far ConvNets have proved extremely successful for a large variety of tasks in computer vision and acoustic modeling [**?**, **?**]. This is largely due to their ability to efficiently exploit stationarity and local statistics to greatly reduce the number of parameters in the network without sacrificing the capacity to accurately represent the data. As a result, one is able to train very large networks while limiting the overfitting problem. The implicit assumption behind a ConvNet is that the data lives on a lattice, a specific type of graph. Images can be thought of as signals defined on a 2-D lattice, where each pixel is a node which is connected to its immediate neighbors. Similarly, audio waveforms can be viewed as signals defined on a 1-D lattice, where each node is a time point.

However, many types of data are defined on more complex graphs. For example, text documents represented as bags of words can be thought of as signals defined on a graph whose nodes are vocabulary terms and whose weights represent some similarity measure between terms, such as co-occurence statistics. In medicine, a patient's gene expression data can be viewed as a signal defined on the graph imposed by the regulatory network. In fact, computer vision and audio, which are the main focus of research efforts in deep learning, only represent a special case of data defined on an extremely simple graph.

In this work we propose two different approaches to generalizing ConvNets to data defined arbitrary graphs. We explore these approaches in two areas of application for which it has not been possible to apply convolutional networks: text categorization and bioinformatics. Our results show that our method is capable of matching or outperforming large, fully-connected networks trained with dropout using fewer parameters.

2 Related Work

Spectral Net

Locally connected from Coates et Al.

Recent paper by Masci et al.

Identification of graphical model.

Stephane's paper on learning wavelets using pairing.

3 Generalizing Convolutions in Graphs

3.1 Spectral Networks

Our work builds upon [?] which defines spectral networks. We recall the definition here and its main properties.

A spectral network generalizes a convolutional network through the Graph Fourier Transform, which is in turn defined via a generalization of the Laplacian operator on the grid to the graph Laplacian. An input vector $x \in \mathbb{R}^N$ is seen as a a signal defined on a graph G with N nodes.

Definition 1. *Let W be a $N \times N$ similarity matrix representing a graph G, and let $L = D - W$ be its graph Laplacian with $D = W\mathbf{1}$, with eigenvectors $U = (u_1, \ldots, u_N)$. Then a graph convolution of input signals x with filters g on G is defined by $x *_G g = U^T (Ux \odot Ug)$, where $\odot$ represents a point-wise product.*

Here, the unitary matrix U plays the role of the Fourier Transform in $\mathbb{R}^d$. There are several ways of computing the graph Laplacian L [?]. In this paper, we choose the normalized version $L = I - D^{-1/2} W D^{-1/2}$, where D is a diagonal matrix with entries $D_{ii} = \sum_j W_{ij}$. Note that in the case where W represents the lattice, from the definition of L we recover the discrete Laplacian operator Δ. Also note that the Laplacian commutes with the translation operator, which is diagonalized in the Fourier basis. It follows that the eigenvectors of Δ are given by the Discrete Fourier Transform (DFT) matrix. We then recover a classical convolution operator that noting that convolutions are by definition linear operators that diagonalize in the Fourier domain (also known as the Convolution Theorem [?]).

Learning filters in a Graph thus amounts to learning spectral multipliers $w_g = (w_1, \ldots, w_N)$

$$x *_G g := U^T (\text{diag}(w_g) U x) \ .$$

Extending the convolution to inputs x with multiple input channels is straightforward. If x is a signal with M input channels and N locations, we apply the transformation U on each channel, and then use multipliers $w_g = (w_{i,j} \ ; \ i \leq N \ , j \leq M)$.

However, for each feature map g we need convolutional kernels are typically restricted to have small spatial support, independent of the number of input pixels N, which enables the model to learn a number of parameters independent of N. In order to recover a similar learning complexity in the spectral domain, it is thus necessary to restrict the class of spectral multipliers to those corresponding to localized filters.

For that purpose, we seek to express spatial localization of filters in terms of their spectral multipliers. In the grid, smoothness in the frequency domain corresponds to the spatial decay, since

$$\left| \frac{\partial^k \hat{x}(\xi)}{\partial \xi^k} \right| \leq C \int u^k |x(u)| du \ ,$$

where $\hat{x}(\xi)$ is the Fourier transform of x. In [?] it was suggested to use the same principle in a general graph, by considering a smoothing kernel $\mathcal{K} \in \mathbb{R}^{N \times N_0}$, such as splines, and searching for spectral multipliers of the form

$$w_g = \mathcal{K} \tilde{w}_g \ .$$

The algorithm which implements the graph convolution is described in **??**.

3.2 Pooling with Hierarchical Graph Clustering

In image and speech applications, and in order to reduce the complexity of the model, it is often useful to trade-off spatial resolution with feature resolution as the representation becomes deeper. For that purpose, pooling layers compute statistics in local neighborhoods, such as the average amplitude, energy or maximum activation.

The same layers can be defined in a Graph by providing the equivalent notion of neighborhood. In this work, we construct such neighborhoods at different scales using multi-resolution spectral clustering [?], and consider both average and max-pooling as in standard convolutional network architectures.

Algorithm 1 Train Graph Convolution Layer

```
1: Given GFT matrix $U$, interpolation kernel $K$, weights $w$.
2: **Forward Pass:**
3: Fetch input batch $x$ and gradients w.r.t outputs $\nabla y$.
4: Compute interpolated weights: $w_{f'f} = K\tilde{w_{f'f}}$.
5: Compute output: $y_{sf'} = U^T \left( \sum_f U x_{sf} \odot w_{f'f} \right)$.
6: **Backward Pass:**
7: Compute gradient w.r.t input: $\nabla x_{sf} = U^T \left( \sum_{f'} \nabla y_{sf'} \odot w_{f'f} \right)$
8: Compute gradient w.r.t interpolated weights: $\nabla w_{f'f} = U^T \left( \sum_s \nabla y_{sf'} \odot x_{sf} \right)$
9: Compute gradient w.r.t weights $\nabla \tilde{w_{f'f}} = K^T \nabla w_{f'f}$.
```

4 Graph Estimation

Whereas some recognition tasks, such as those considered in [**?**], might have a prior knowledge of the graph structure of the input data, many other real-world applications do not have such knowledge. It is thus necessary to estimate a similarity matrix W from the data before constructing the spectral network. We consider in this paper two possible graph constructions.

4.1 Estimation from Feature Correlation

Given data $X \in \mathbb{R}^{N \times L}$, where L is the number of samples and N the number of features, the simplest approach to estimating a graph structure from the data is to consider a distance between features i and j given by

$$d(i,j) = \|X_i - X_j\|^2 \ ,$$

where X_i is the i-th row of X.

This distance is then used to build a Gaussian diffusion Kernel [**?**]

$$\omega(i,j) = \exp^{-\frac{d(i,j)}{\sigma^2}} \ . \qquad (1)$$

In our experiments, consider the variant of self-tuning diffusion kernel [**?**]

$$\omega(i,j) = \exp^{-\frac{d(i,j)}{\sigma_i \sigma_j}} \ ,$$

where σ_i is computed as the distance $d(i, i_k)$ corresponding to the k-th nearest neighbor i_k of feature i.

The main advantage of (**??**) is that it does not require labeled data. Therefore, it is possible to estimate the similarity using several datasets that share the same features, for example in text classification. While correlations are typically sufficient to reveal the intrinsic geometrical structure of images [**?**], the effects of higher-order statistics might be non-negligible in other contexts, especially in presence of sparsity.

Another strategy is to use ℓ_1 penalized logistic regression [**?**].

4.2 Supervised Graph Estimation

Use a fully connected network to determine the feature similarity.

5 Experiments

6 Discussion

The graph construction using the Supervised Graph Estimation. Related to discovering latent graphical models

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*, 2013.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.