GCNs: Mind the spectral gap

Candidate Number: 1004226
Theories of Deep Learning C7.1
Mathematical Institute
University of Oxford

Abstract

The application of Convolutional Neural Networks to Audio-Visual learning tasks has been enormously successful, in many cases achieving super-human performance [1]. The success in part is due to the local translational invariance of data defined over Euclidean domains. Increasingly we wish to learn functions over non-euclidean domains, such as graphs, leading to the development of Graph Convolutional Networks (GCNs). Despite the high performance of shallow GCNs, mysteriously, we experience diminishing returns with added depth. In this paper, we link the depth of a GCN to the well-studied spectral gap of a graph. In network science the spectral gap has been shown to strongly influence the mixing time of signals on a graph, and we argue it should be considered when determining the depth of GCN to use.

1 Introduction

The introduction of convolutional layers to neural networks has led to a paradigm shift in deep learning. Convolutional layers, modelled on the physiology of the visual cortex, incorporate the idea of local receptive fields. Each neuron in a convolutional layer is only partially connected to the previous layer. Coupled with feature maps, CNNs allow aggregation of local features within data, whilst requiring only a relatively small number of trainable parameters [2]. That we can do this, is fundamentally due to the stationarity and translational invariance of the data involved. Essentially, when processing an image, for example, we assume vertical lines are no more likely to appear in the top left corner of an image than the bottom right of an image [3]. These properties allow each of the receptive fields to share the same parameters thus greatly reducing the total number of parameters needed. Whilst these assumptions are well-defined on Euclidean domains (e.g. Image and Audio data), it is not immediately obvious how one would extend them to data defined on a non-Euclidean domain, such as a graph or a manifold.

More and more of the data we collect has an underlying network structure [4]. Examples include social networks, drug discovery networks and knowledge graphs. This has driven the generalisation of CNNs to methods which work on graphs. Such attempts include spectral methods (GCNNs [5] and their simplification GCNs [6]) as well as embedding methods, which attempt to project the graph into Euclidean space, whereby standard CNNs can be applied [7].

GCNs have achieved state of the art performance on many benchmarks. However, it has been observed increasing the number of layers of a GCN does not lead to corresponding improvements in test accuracy [8]. This paper investigates the relation of the spectral gap of the graph laplacian in relation to the depth of a GCN, with a view to explaining this phenomena.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Mathematical Background

2.1 ChebNet

We begin with a brief introduction to the concept of spectral graph convolutions proposed in [5], but first we need some basic terminology from graph theory.

Let $G = (V, E)$ be the graph with vertex set V and edge set E, where $|V| = n$. Then the adjacency matrix of a graph $A \in \mathbb{R}^{n \times n}$ is the symmetric matrix with $(a_{ij}) = 1$ whenever there is an edge between i and j. The degree matrix D is the diagonal matrix with the degree of each node on the diagonal, $D_{ii} = \sum_j A_{i,j}$. Finally, the (symmetric) normalised Laplacian is defined $L_{sym} := I - D^{-1/2} A D^{-1/2}$. Note that the normalised laplacian is a symmetric positive-semidefinite matrix and hence permits a full spectral decomposition $L = U \Lambda U^T$, where U holds an orthonormal basis of eigenvectors and every eigenvalue is real and non-negative. Due to similarities between the behaviour of these eigenvectors and the eigenfunctions associated to the classical Fourier transform, the operator U^T is sometimes called the graph Fourier transform. These similarities, by analogy, permit us an extension of the convolution operation to the graphical domain.

Let $\mathbf{x} \in \mathbb{R}^n$ a signal defined on the n vertices of a graph. Then the convolution of a filter $g_\theta = diag(\theta)$, $\theta \in \mathbb{R}^n$, with $\mathbf{x}$ is defined as

$$g_\theta * \mathbf{x} = U g_\theta U^T \mathbf{x}$$

Having defined graph convolution, we could then proceed to directly replace the standard convolutions in a CNN. Unlike CNNs though, this convolution is not local (it operates over the whole graph) and we have an $\mathcal{O}(n)$ learning complexity. In order to rectify these issues [5] suggested redefining a locaised filter operation as

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

so that the graph convolution becomes $g_\theta * \mathbf{x} = U g_\theta(\Lambda) U^T \mathbf{x}$. Now the convolution is K-localised, convolution with a vertex v requires only data located within K steps on the network. They also re-paramaterise g_θ using Chebyshev polynomials, which can further bring down computational complexity.

The Chebyshev polynomials can be defined recursively as $T_0 = 1$, $T_1 = x$ and

$$T_{n+1}(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

These polynomials form a basis for the Hilbert space of square integrable functions with $[-1, 1]$ as support, and measure $dy/\sqrt{1 - y^2}$. These properties allow us to calculate

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

where Λ is rescaled $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$ in order to have eigenvalues within $[-1, 1]$. Thus,

$$g_\theta(\Lambda) * \mathbf{x} = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})$$

The neural network model using this convolution is known as ChebNet.

2.2 GCN

The Graph Convolutional Network (GCN) introduced in [6] is in fact an approximation of ChebNet. The authors propose we take $K = 1$ and assume that $\lambda_{max} \approx 2$. Under these conditions $g_\theta * x = \theta_0 x + \theta_1 (L - I)x$. Then further constraining the parameters they set $\theta = \theta_0 = -\theta_1$. In which case we have:

$$g_\theta * \mathbf{x} = \theta(I + D^{-1/2} A D^{-1/2})\mathbf{x}$$

The authors further applied a re-normalisation trick $I + D^{-1/2}AD^{-1/2} \rightarrow \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ where $\tilde{A} = A + I$ and $\tilde{D}_{i,i} = \sum_j A_{i,j}$. Leaving us with a simple propagation rule for convolutional layers

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

Where σ is a non-linear activation function such as ReLU, $H^{(l-1)}$ is the signal from the previous layer and $\Theta^{(l)}$ is the weight matrix to be learned via training.

Using a simple two layer version of this neural network, with an appropriate loss function, [6] achieve state-of-the-art results on multiple datasets. However, it was observed by [8] that adding depth fails to provide performance increases. This is particularly surprising given that the two layer GCN is 2-localised, throwing away almost all of the global graph topology. In an attempt to explain this phenomena, [8] show that the convolution is essentially equivalent to Laplacian smoothing [10]. The convolution operation on a vertex v and signal x_v, updates v with the a weighted weighted average of the signals of itself and it's neighbours, thereby smoothing the signal.

$$\bar{x_v} \leftarrow \frac{1}{D_{v,v}+1} \sum_{u \in \Gamma(v) \bigcup \{v\}} x_u$$

When learning functions on graphs we are operating under the assumption that proximity on the graph indicates two signals are similar. Laplacian smoothing may be viewed as a preprocessing step which forces local signals to become more similar thus providing easier discrimination for the neural network. Unfortunately, repeated application of the smoothing operator will eventually force the whole graph to look uniform. In this way, we lose the ability to discriminate between signals. Having observed the current state of affairs, we will present a novel view of the graph convolution operation by relation to consensus dynamics a concept from Network Science.

3 Consensus dynamics

Consensus dynamics studies the task of taking a signal $\mathbf{x}(0) \in \mathbb{R}^n$ defined on a graph and forcing the nodes to realise the global average via an efficient distributed method. For example, within a social network, people (the nodes) may have differing views on some topic (the signal), they may then exchange views with their friends (neighbours) in order to come to some consensus. We wish to study the dynamics of the process by which global consensus is achieved. Using the notation of [9] consensus dynamics are systems

$$\mathbf{x}(t+1) = P\mathbf{x}(t)$$

where P is a stochastic matrix i.e. the rows of of P are non-negative and sum to 1. Notice the matrix $T := \tilde{D}^{-1}\tilde{A} = \tilde{D}^{-1/2}(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})\tilde{D}^{1/2}$ satisfies these requirements. This T may be considered as the transition matrix for an aperiodic irreducible random walk on G (with self loops added). Using standard results from the theory of Markov chains, or alternatively the Perron-Frobenius Theorem from [], we deduce the existence of a stationary distribution π of T, further $\pi_i = \frac{1}{d_i+1}$. Thus we have convergence to a consensus point

$$\lim_{t\to\infty} \mathbf{x}(t) = \lim_{t\to\infty} T^t\mathbf{x}(0) = \mathbf{1}(\pi^T\mathbf{x}(0))$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones. However we are more interested in the laplacian smoothing operator $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} = \tilde{D}^{1/2}T\tilde{D}^{-1/2}$. Again using results established in [], we may deduce:

$$[\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}]^t\mathbf{x}(0) = \pi^{1/2}(\pi^{1/2})^T\mathbf{x}(0) + \sum_{j\geq 2} \mu_j^t \phi_j \phi_j^T \mathbf{x}(0)$$

Where $1 = \mu_1 > \mu 2 \geq \cdots \geq \mu_n > -1$ are the eigenvalues of $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ and $\phi_1, \phi_2, \cdots, \phi_n$ are the associated orthonormal eigenvectors. It is now easy to see that all the eigenmodes in the sum vanish in the limit, however for small values of t, the eigenmode associated to μ_2 plays an outsized role in the consensus dynamics.

3.1 The spectral gap

The spectral gap in this context is defined as $|\mu_1 - \mu_2|$ clearly this gap has strong implications for rate of convergence to the consensus, also known as the mixing time. The spectral gap then, must also have strong implications for GCNs. Indeed, if we take a GCN with k layers, and assume a trivial activation function, we obtain

$$Z = [\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}]^k X\Theta^{(1)}\Theta^{(1)}\dots\Theta^{(k)}$$

3.2 Numerical Simulations

To illustrate the connection further we perform a series of experiments on a the basic GCN architecture defined above. We generate graphs using a stochastic block model with two blocks. By varying the parameters for the out of cluster edge probability we can generate a wide range of spectral gaps. We initialise different Gaussian signals on each of the blocks proceeding to apply convolutions to the signals until they are indistinguishable. We record the mixing time (number of convolutions) against the spectral gap.

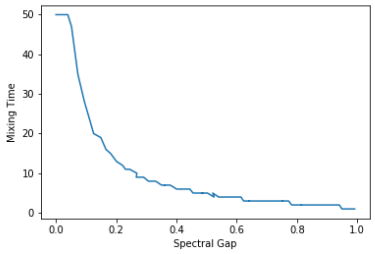

Figure 1: Comparison of mixing time against spectral gap on a stochastic block model with two blocks of 100 nodes each.

In figure 1, we observe that mixing time drops incredibly quickly when increasing the spectral gap. This long tailed distribution may in part explain why the addition of layers is futile for most GCNs.

To further show the dependence on the spectral gap, we investigate stochastic block models of varying sizes. Additionally, we perform the same experiments on some real-world networks.

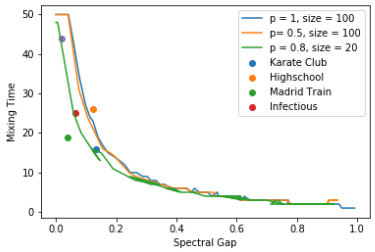

Figure 2: Comparison of mixing time against spectral gap on stochastic block models with varying parameters, along with some real-world networks.

In figure 2, we see that the mixing time is almost completely determined by the spectral gap. Finally we visualise, figure 3, the mixing induced by graph convolutions by projecting the signals on to the plane. We use the Karate Club as the underlying graph.

From this visualisation (figure 3) it is easy to grasp that there is a goldilocks zone somewhere between 5 and 10 layers where the graph convolutions are helpful, in fact they appear to be linearly separable. However too few layers, or too many layers results in the two signals being difficult to separate. The spectral gap for the Karate Club network is ≈ 0.13. Given how similar the mixing time is for networks with similar spectral gaps, we would hypothesize a similar goldilocks zone of 5 to 10

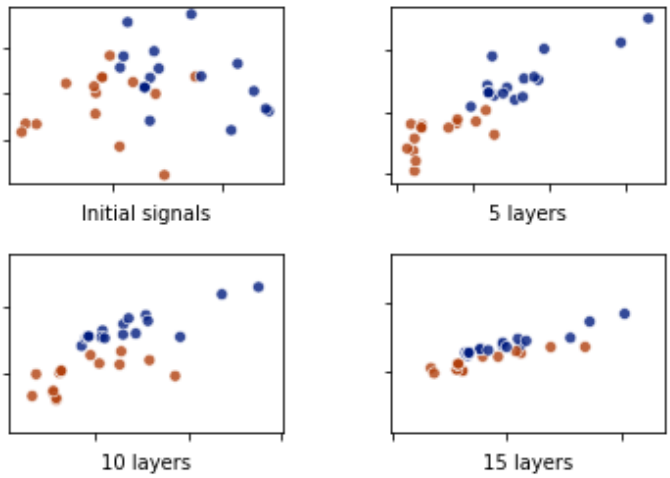

Figure 3: The mixing of signals on the Karate Club network after different numbers of graph convolutions.

layers for those networks. To illustrate the point we provide the same visualisation, figure 4, for the High school network which has a similar spectral gap ≈ 0.12. Indeed, we see what appears to be an optimal separation somewhere between 5 and 10 layers.

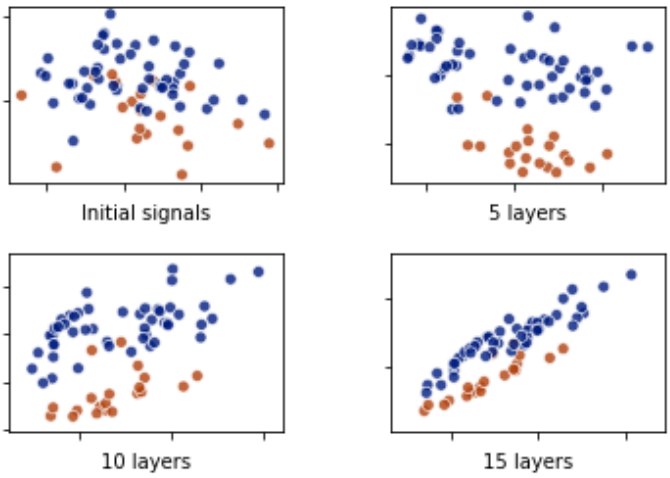

Figure 4: The mixing of signals on the High School network after different numbers of graph convolutions.

4 Discussion

In this paper, we have attempted to provide new insight into the performance of Graph Neural Networks. By linking the convolution operator to consensus dynamics on Networks, we have shown the number of layers in a GCN is in some sense equivalent to a 'goldilocks' choice of timescale in consensus dynamics. Further, by use of numerical simulations, we presented the view that the most important factor for picking a timescale is the spectral gap of the symmetric graph laplacian. Ideally,

we would have liked to show that the spectral gap retains its importance with non-linear activation functions, learned feature maps and larger data sets. However, due to the constraints of time we reserve these investigations as a possible avenue for future research.

If the importance does hold, we note that approximating the spectral gap of the normalised laplacian can be performed efficiently (by way of the power method, Rayleigh quotient iteration or even by more recent methods such as Lanczos iteration [11]). Hence, rather than relying on trial and error, calculating the spectral gap could provide a useful aid for depth selection in GCNs. Whilst other strategies to tackle the problem of locality have been suggested, such as, [12,13], these methods assume that some nodes are more important than others (based on their graph centrality). The spectral gap is free from such qualitative judgements, its utility, therefore, is somewhat more general.

1. Cireşan D, Meier U, Masci J, Schmidhuber J (2012) Multi-column deep neural network for traffic sign classification. Neural Networks 32:333-338.

2. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86:2278-2324.

3. Field D (1989) What The Statistics Of Natural Images Tell Us About Visual Coding. Human Vision, Visual Processing, and Digital Display.

4. Lazer D et al. (2009) SOCIAL SCIENCE: Computational Social Science. Science 323:721-723.

5. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. Advances in Neural Information Processing Systems 29.

6. Kipf T, Welling M (2017) Semi-Supervised Classification With Graph Neural Networks. International Conference on Learning Representations 2017 - ICLR '17.

7. Weston J, Ratle F, Collobert R (2008) Deep learning via semi-supervised embedding. Proceedings of the 25th international conference on Machine learning - ICML '08.

8. Li Q, Han Z, Wu X-M (2018) Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. Thirty-Second AAAI Conference on Artificial Intelligence.

9. Fangani F (2014) Consensus dynamics over networks.

10. Taubin G Curve and surface smoothing without shrinkage. Proceedings of IEEE International Conference on Computer Vision.

11. Lanczos C (1950) An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of Research of the National Bureau of Standards 45:255.

12. Xu K et al. (2018) Representation Learning on Graphs with Jumping Knowledge Networks. Proceedings of the 35th International Conference on Machine Learning, PMLR 80:5453-5462.

13.Abu-El-Haija S, Kapoor A, Perozzi B, Lee J (2018) N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification arXiv:1802.08888v1 [cs.LG].