Sharif Unversity of Technology

Convolutional Deep Networks on Graphs

Author:
Khashayar Gatmiry

Supervisors:
Prof. Mahdie Soleymani

Department or Computer Engineering

February 10, 2018

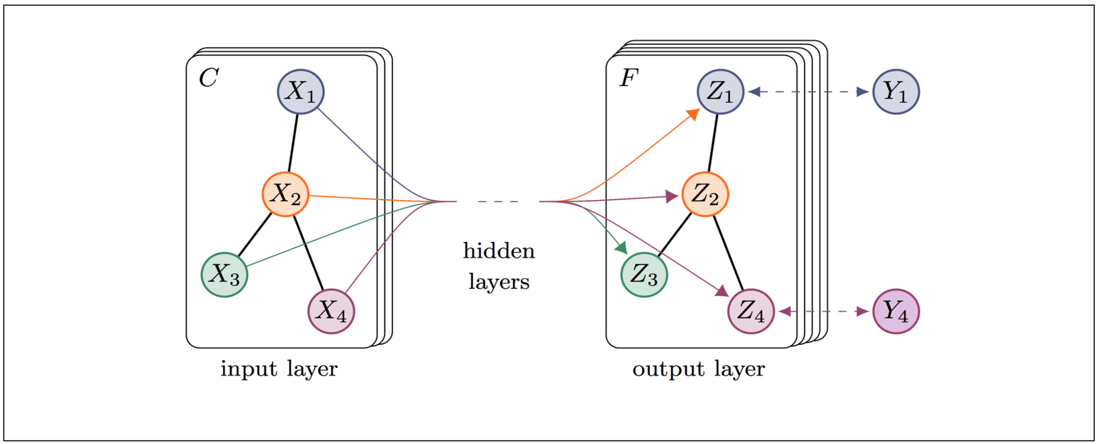

FIGURE 1: CNNs on graphs to extract features (Kipf and Welling, 2016).

0.1 Abstract

Throughout this brief report, we discuss on suitable architectures of Convolutional Neural Networks which are applicable on graph inputs. We begin by briefly summarizing the recent innovations and state of the art architectures in this new area, and we continue with our ideas.

0.2 Introduction

Convolutional Neural Network (CNN) is a special kind of Neural Network which is designed to exploit the local structures embedded in a high-dimensional space to extract local features in each layer, and merging these features in higher level layers (Krizhevsky, Sutskever, and Hinton, 2012).
Note that each layer of a CNN consist of a linear operator of the input, combined with a non-linearity (by applying a non-linear function to the output of the layer). There are two main ingredients in the definition of a CNN; First, it should becomes a diagonal operator in the frequency domain. Second, it should extract "local" features. More precisely, each feature need to be in correspondence with a local community of points in the input space. However, the notion of locality does not possess a rigorous definition in the literature.
Convolutional Neural Network has been successfully applied on Euclidian spaces, including Images in the past decades. The reason for its success is mainly based on two important factors; Firstly, the notion of locality is well-defined on euclidian spaces. Secondly, the output of each layer can be computed very efficiently by convolving the input linearly with a low-dimensional filter. Hence, the filter can extract similar local structures from the hole input points (or pixels) as features fed to the next layer. Moreover, the low-dimensionality of the filter parameters avoids over-fitting.
Recently, there has been a great interest in generalizing CNN nets beyond Euclidian spaces, to graph structures, which let us to take advantage of their power in tasks such as classification on graphs. (Monti et al., 2017) , (Kipf and Welling, 2016) , (Bruna et al., 2013) , (Henaff, Bruna, and LeCun, 2015) ,

(Scarselli et al., 2009),(Duvenaud et al., 2015), (Defferrard, Bresson, and Vandergheynst, 2016).

0.3 Notion of Generalization and Related Ideas

To generalize CNNs, we need to generalize their two important factors; their spectral property, and their locality. As we mentioned, a spectral map is one which becomes diagonalized in the frequency domain. Despite the continues space, a real valued function on the nodes of a graph has a finite domain.
For a given graph $G = (\mathcal{V}, \mathcal{E})$ with set of vertices $\mathcal{V}$ and set of edges $\mathcal{E}$, its Laplacian is defined by $\mathrm{L} = \mathrm{D} - \mathrm{A}$, where D is the diagonal degree matrix, consisting of the degree of edges on its diameter, and A is the adjacency matrix.
It's well-known from the literature of spectral graph theory, that the set of vectors which forms the basis of the frequency domain is obtained from the eigenvectors of the laplacian. Due to the fact that L is a symmetric matrix, there exist a set of its eigenvectors contains $n = |\mathrm{V}|$ orthogonal and unitary vectors. Let $\mathcal{U} = \{\mathrm{U}_i\}_{i=1}^{n}$ be a pairwise orthogonal set of eigenvectors with respect to L. Thus, L becomes diagonalized in the basis of $\mathcal{U}$. Let U be the matrix consisting of $\{\mathrm{U}_i\}_{i=1}^{n}$ as its columns. For diagonalized matrix ψ, we have

$$\mathrm{L} = \mathrm{U}\psi\mathrm{U}^{\top}, \quad \psi = \mathrm{diag}(\{\psi_i\}_{i=1}^{n}).$$

Note that since U has orthogonal columns, $\mathrm{U}^{\top}$ is actually the inverse of U.
Now regarding the spectral property, a convolutional layer g consist of should become diagonalized after basis transformation with respect to U. To define a convolutional layer more formally, we need to assume special ratios which are multiplied to the input vector in the frequency domain. Assume that our layer only consider the highest frequencies, which means that it simply igonre the frequencies below a certain threshold. Let us assume that in a given layer of the network, we have p group of input signals $F = (f_1, ..., f_p)$ on the graph nodes, and we want to extract q group of features $G = (g_1, ..., g_q)$ which are functions on the graph's nodes as f_i's. Suppose that $\Phi^{(\ell,\ell')} = \mathrm{diag}(\{\Phi_i^{(\ell,\ell')}\}_{i=1}^{k})$ is a $k \times k$ dimensional diagonal matrix corresponding to the ratios of our filter in the frequency domain, where we have assumed that only the ratios with respect to the top greatest frequencies are non-zero. For $1 \leq \ell \leq q$, define the ℓth layer of output as

$$g_\ell = \tau(\sum_{\ell'} \mathrm{U}_k \Phi^{(\ell,\ell')} \mathrm{U}_k^{\top} f_{\ell'}), \tag{1}$$

where τ is a non-linearity applied to the output.
After a convolutional layer, we can optionally put a max-pooling layer, using graph coarsening, by uniformly eliminating graph nodes.
We characterized the spectral property in equation 1. For locality, the filter ratios need to be a smooth function of the eigenvalues of Laplacian. However, this definition is not as rigorous as the one for spectral property. A popular

idea regarding the smoothness, is to define filter ratios as a polynomial map of eigenvalues of Laplacian. But due to equation 1 this means that g_ℓ is a polynomial function of Laplacian itself. Hence, there exist a polynomial $\mathcal{P}_\ell$ with degree r, such that

$$g_\ell = \tau(\mathcal{P}_\ell(\mathrm{L})), \tag{2}$$

which leads to

$$g_\ell = \tau(\mathrm{U}\mathcal{P}_\ell(\psi)\mathrm{U}^\top). \tag{3}$$

This form of representation has two major advantages for us; Firstly, the space of polynomials are dense in the space of continues function, which gives them the ability to model any kind of smooth function with respect to eigenvalues. Secondly, equation 2 gives us a computationally straightforward way to compute g_ℓ, without the need of matrix inversion. There were two major game-changing works regarding this idea, namely the Chebyshev filters, and Cayley filters that we express next.

0.3.1 Chebyshev Nets

Chebyshev polynomials, if their domain is constrained to $[-1, 1]$ interval, are dense in the space of continues functions from interval $[-1, 1]$ (Bronstein et al., 2017). Therefore, it's logical to pick $\mathcal{P}_\ell$ as a Chebyshev polynomial. On the other hand, Chebyshev polynomials are computable with respect to a simple recursive relation, which makes them computable in linear time of sample size, hence very efficient. Chebyshev polynomials can be defined as

$$\begin{aligned} &T_0(x) = 1, \\ &T_1(x) = x, \\ &T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x). \end{aligned}$$

Hence

$$g(\mathrm{L}) = \tau(\sum_{j=1}^{r-1} \alpha_j T_j(\tilde{\mathrm{L}})), \tag{4}$$

where $\tilde{L} = 2\lambda_n^{-1}\mathrm{L} - \mathrm{I}$ is a normalization of L with respect to the largest eigenvalue λ_n, because as we said, Chebyshev polynomials are dense in the domain $[-1, 1]$.

This recursive equation gives us a tool to compute the output of each layer in a linear computational cost, with respect to n (matrix size):

$$\begin{aligned} &f^{(j)} = 2\tilde{\mathrm{L}}f^{(j-1)} - f^{(j-2)}, \\ &f^{(0)} = f, \\ &f^{(1)} = \tilde{\mathrm{L}}f. \end{aligned}$$

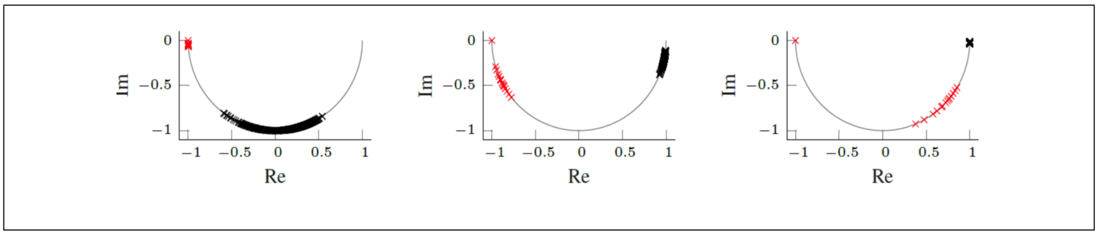

FIGURE 2: Ability of Cayley nets to concentrate on a specific low-band of frequency, in spite of Chebychev nets. (Levie et al., 2017).

0.3.2 Cayley Nets

A more advanced idea with this regards is a new work which take advantage of Cayley polynomials for our smooth map of eigenvalues (Levie et al., 2017). The disadvantage of Chebyshev polynomials was that they was unable to focus on small bands of frequency, with a constrained degree. However, it can be the case that our labels are much more corollated with a specific low-band frequency of the graph. Hence, in order to get information from a low intensity frequency by Chebyshev polynomials, we need to increase the degree r, so that the Chebyshev polynomial attains higher values in $\epsilon \ll 1$. Nevertheless, Cayley polynomials can overcome this issue. Formally, a Cayley polynomial is defined on the complex space by

$$\mathcal{C}_{c,h}(x) = c_0 + 2\mathrm{Re}\{\sum_{j=1}^{r} c_j(hx-i)^j(hx+i)^{-j}\} = c_0 + 2\mathrm{Re}\{\sum_{j=1}^{r} c_j(\frac{hx-i}{hx+i})^j\}. \tag{5}$$

This leads to

$$\begin{aligned} \mathcal{C}_{c,h}(\mathrm{L}) = c_0 + 2\mathrm{Re}\{\sum_{j=1}^{r} c_j(h\mathrm{L}-i)^j(h\mathrm{L}+i)^{-j}\} = \\ c_0 + 2\mathrm{U}.\mathrm{Re}\{\sum_{j=1}^{r} c_j(\frac{h\psi-i}{h\psi+i})^j\}.\mathrm{U}^\top = \\ c_0 + 2\mathrm{U}.\mathrm{Re}\{\sum_{j=1}^{r} c_j \mathrm{diag}[\{(\frac{h\psi_s-i}{h\psi_s+i})^j\}_{s=1}^{n}]\}.\mathrm{U}^\top\ . \end{aligned}$$

From a geometric point of view, Cayley polynomial multiplies the eigenvalue with a scaling factor h, and then maps it to the complex space, by considering its imaginary part equal to 1. Then, it projects the complex point on the norm-1 unit ball, and multiply its angle with a degree $1 \leq d \leq r$. At the end, it projects the point to the real line, by taking its real part. One can see that this procedure can easily change the arrange of eigenvalues, thus it can focus on a specific band of frequency in the unit ball of complex space. This fact is illustrated in Figure 2.

0.4 Our Contribution

0.4.1 The Dual Representation and Commute Distance

For a graph G, if we can divide the Laplacian L into the degree matrix D to obtain the probability matrix of a random walk $\mathcal{W}$ in the graph, where we go from each node to one of its neighbors with regards to the weight of the edge between them. Given $\mathcal{W}$, we define the commute distance between any two nodes v_1, v_2 as the expected number of edges we should see, starting from one of them to reach the other, and coming back:

$$d_{\mathcal{W}}(x, y) = \mathbb{E}_{\mathcal{W}}[d(x, y, x)].$$

From the literature of Spectral graph theory (Von Luxburg, 2007), it's known that there exist an embedding of the graph's nodes into a high-dimensional Euclidian space, where the euclidian distance between any two points equals their random distance in the graph. More precisely, let $\{\mathrm{U}^{(i)}\}_{i=1}^{n}$ be U's rows. If we define

$$\mathcal{J}_i = \frac{1}{\sqrt{\psi_i}} \mathrm{U}^{(i)} \ , \ \ i = 1, ..., n, \tag{6}$$

which corresponds to the ith node's embedding in $\mathbb{R}^n$. This is called the dual representation of the graph. Note that there essentially exist a zero eigenvalue, for which we replace zero instead of $\frac{1}{\sqrt{\psi_i}}$ in the above definition (it's corresponding to the generalized inverse of a matrix, which can be find more specifically in Luxburg tutorial on Spectral Graph Theory).

0.4.2 ϵ-Approximation of Spectral Property

We have a consistent definition for locality in euclidian spaces. Hence, we can use this dual representation to define a good framework for locality.

We say that the linear map g is ϵ-approximate spectral, if for each eigenvector of Laplacian U_i, there exist an ϵ perturbation of U_i which is an eigenvector of g. More precisely, an ϵ perturbation of vector U_i is vector $\tilde{\mathrm{U}}_i$, such that

$$\mathrm{U}_i - \epsilon \leq \tilde{\mathrm{U}}_i \leq \mathrm{U}_i + \epsilon.$$

Now we set a threshold ϵ, and for each point $\mathcal{J}_j$ in the dual space consider the norm-∞ ball $\mathcal{B}_{\frac{\epsilon}{\sqrt{n}}}(\mathcal{J}_j)$ with radius $\frac{\epsilon}{\sqrt{n}}$ and center $\mathcal{J}_j$. More formally, for each node j, Let $\mathcal{Q}_j$ be the set consisting of all other nodes' indices which have $\ell - \infty$ distance less than $\frac{\epsilon}{\sqrt{n}}$ in the dual space:

$$\mathcal{Q}_j = \{i \mid \parallel \mathcal{J}_i - \mathcal{J}_j \parallel_\infty \le \frac{\epsilon}{\sqrt{n}}\}. \tag{7}$$

Let $m = \text{argmin}_i |\mathcal{Q}_i|$. Then for each i, define $\tilde{\mathcal{Q}}_i$ the set consisting of the m closest nodes in $\mathcal{Q}_i$ to $\mathcal{J}_i$ with respect to the $\ell - \infty$ norm in the dual space. Moreover, consider the set of normalized parameters $\kappa = \{\kappa_1, ..., \kappa_m\}$ for our model, such that $||\kappa||_2 = 1$. For each j consider a desired map $\sigma_j : \tilde{\mathcal{Q}}_j \to \{1, ..., m\}$, and define the n-dimensional vector K_j which has $\kappa_{\sigma(i)}$ in the ith index if $i \in \tilde{\mathcal{Q}}_j$, and zero otherwise. note that our parameter vector κ is fixed and does not change with respect to j. A little while later, we will present a logical choice for σ, but for now, suppose it is desired. At the end, define the matrix K consisting of $\{\mathrm{K}_i^\top\}_{i=1}^n$ as its rows.

We claim that K is ϵ-approximate. In order to prove, we must show that it becomes diagonalized in the fourier basis, namely the eigenvectors of Laplacian. Hence, for each eigenvector of g such as U_i, we must show that U_i is the eigenvector of K. If we illustrate the sth entry of a desired vector v by $v(s)$, we have

$$\mathrm{K}_j\mathrm{U}_i = \sum_{s=1}^{n} \mathrm{K}_j(s)\mathrm{U}_i(s) = \sum_{s\in\tilde{\mathcal{Q}}_j} \mathrm{K}_j(s)\mathrm{U}_i(s) + \sum_{s\in\tilde{\mathcal{Q}}_j{}^c} \mathrm{K}_j(s)\mathrm{U}_i(s) = \sum_{s\in\tilde{\mathcal{Q}}_j} \kappa_{\sigma_j(s)}\mathrm{U}_i(s) = \\ \sum_{s\in\tilde{\mathcal{Q}}_j} \sqrt{\psi_s}\kappa_{\sigma_j(s)}\frac{1}{\sqrt{\psi_s}}\mathrm{U}_i(s) = \sum_{s\in\tilde{\mathcal{Q}}_j} \sqrt{\psi_s}\kappa_{\sigma_j(s)}\mathcal{J}_s(i).$$

But note that for each $s \in \tilde{\mathcal{Q}}_j$, equation 7 gives

$$\mathcal{J}_j(i) - \frac{\epsilon}{\sqrt{n}} \le \mathcal{J}_s(i) \le \mathcal{J}_j(i) + \frac{\epsilon}{\sqrt{n}}.$$

Thus, we have

$$\mathrm{K}_j\mathrm{U}_i \le \sum_{s\in\tilde{\mathcal{Q}}_j} \sqrt{\psi_s}\kappa_{\sigma_j(s)}(\mathcal{J}_j(i) + \frac{\epsilon}{\sqrt{n}}) = \sum_{s\in\tilde{\mathcal{Q}}_j} \kappa_{\sigma_j(s)}\mathrm{U}_i(j) + \sum_{s\in\tilde{\mathcal{Q}}_j} \sqrt{\psi_s}\kappa_{\sigma_j(s)}\frac{\epsilon}{\sqrt{n}} \le \\ (\sum_{z=1}^{m} \kappa_z)\mathrm{U}_i(j) + \frac{\epsilon}{\sqrt{n}}\sqrt{(\sum_{s\in\tilde{\mathcal{Q}}_j} \psi_s)(\sum_{s\in\tilde{\mathcal{Q}}_j} \kappa_s^2)} = \gamma\mathrm{U}_i(j) + \frac{\epsilon}{\sqrt{n}}\sqrt{n} = \gamma\mathrm{U}_i(j) + \epsilon. \tag{8}$$

Similarly, one can express

$$\mathrm{K}_j\mathrm{U}_i \ge \gamma\mathrm{U}_i(j) - \epsilon\sqrt{n}. \tag{9}$$

Combining both of inequalities 8 , 9 for each $1 \leq j \leq n$ reveals

$$\| \mathrm{KU}_i - \gamma \mathrm{U}_i \|_\infty \leq \epsilon \sqrt{n}. \tag{10}$$

Noting the fact that inequality 10 is true for any $1 \leq i \leq n$, we conclude that K is ϵ-approximate spectral as desired.

The embedding Theorem expressed in part 0.4.1 reveals that our architecture is absolutely localized, because we have designated the norm-∞ ball of a specific radius around each node as its locality. This is the most similar definition to the notion of locality in euclidian-spaces' CNNs, which is possible. On the other hand, we proved above, that our defined filter is ϵ-approximate spectral as well. This means that our Network is localized and spectral at the same time, as desired.

0.4.3 Computational Cost and determining σ

in order to find the architecture of the model based on the Laplacian, we need to apply eigendecomposition on L, which takes $O(n^3)$. But as soon as we find the right linear map in each layer of network, then forward and backward loops are computable very efficiently, in $O(mn)$ iterations, which is linear in sample size n.
in order to determine the map σ, its important to use a consistent definition through the locality ball of all the nodes. we apply PCA dimensionality reduction in each ball, then we consider κ_1 for the most-variance captured principal component, known as PCA1, consider κ_2 for PCA2 and so on. This way, it seems logical that all the nodes share the same family of parameters $\{\kappa_1, ..., \kappa_m\}$. In addition, note that The computational cost of applying PCA dimensionality reduction does not harm our linear time complexity for training and testing, since it corresponds to the architecture of our model and hence it's a preprocess.

0.4.4 Generalization of Cayley Nets using Family of Convex Balls

Regarding the geometric point of view in Cayley polynomials, we aim to generalize Cayley nets by giving another degree of freedom to the model. Consider the family of convex balls $\mathrm{B} = \{\mathrm{B}_i\}_{i=1}^{\infty}$ with regards to the equations $\{x^{2i} + y^{2i} = 1\}_{i=1}^{\infty}$. We replace the norm-2 euclidian ball in the complex space with a desired $\mathrm{B}_i \in \mathrm{B}$, and we apply the geometric procedure of Cayley polynomial, which we described in section 0.3.2, with respect to B_i. In order to this end, for an eigenvalue λ, we consider the complex number $h\lambda - i$ (as in Cayley polynomials), project it on the convex ball of B_i, multiply its angle (in complex space) by a degree $1 \leq d \leq r$, and finally project it on the real line. To do so, first we need to find the projection of a normalized complex number $(\alpha, \sqrt{1-\alpha^2})$ on B_i. we should have

$$(\alpha x)^{2i} + (\sqrt{1-\alpha^2} x)^{2i} = 1, \tag{11}$$

which gives

$$x = [\frac{1}{\alpha^{2i} + (1-\alpha^2)^i}]^{\frac{1}{2i}}. \tag{12}$$

But we can calculate the point p, which is the projection of $h\lambda - i$ on the norm-2 ball using the Cayley polynomial $\mathcal{C}_{c,h}$. Afterwards using equation 12 we can compute the projection of p on the convex ball B_i. This is computable without the need of eigendecomposition of Laplacian, using numerical methods for calculating the ith root of a positive definite matrix. This way, we obtain another degree of freedom on Cayley polynomials, the degree of convex balls $\{\mathrm{B}_i\}$ where we project on. However, the efficiency and danger of overfitting in this generalized model may become issues, which should be tested through experiments.

Bibliography

Bronstein, Michael M et al. (2017). *Geometric deep learning: going beyond euclidean data*. URL: https://arxiv.org/abs/1611.08097.

Bruna, Joan et al. (2013). “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203*.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in Neural Information Processing Systems*, pp. 3844–3852.

Duvenaud, David K et al. (2015). “Convolutional networks on graphs for learning molecular fingerprints”. In: *Advances in neural information processing systems*, pp. 2224–2232.

Henaff, Mikael, Joan Bruna, and Yann LeCun (2015). “Deep convolutional networks on graph-structured data”. In: *arXiv preprint arXiv:1506.05163*.

Kipf, Thomas N and Max Welling (2016). “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Levie, Ron et al. (2017). “Cayleynets: Graph convolutional neural networks with complex rational spectral filters”. In: *arXiv preprint arXiv:1705.07664*.

Monti, Federico et al. (2017). “Geometric deep learning on graphs and manifolds using mixture model CNNs”. In: *Proc. CVPR*. Vol. 1. 2, p. 3.

Scarselli, Franco et al. (2009). “The graph neural network model”. In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Von Luxburg, Ulrike (2007). “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4, pp. 395–416.