Graph Neural Networks

March 11, 2020

OUTLINE

Classical Convolutional Neural Networks

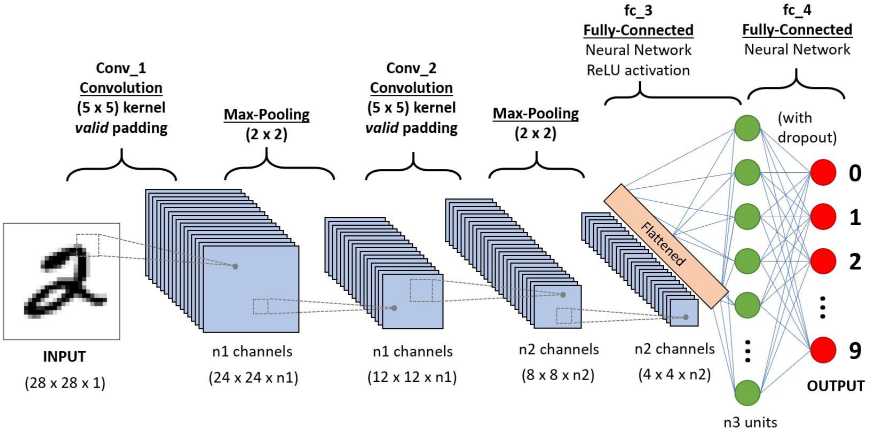
- Convolution
- Pooling
- Invariance properties

Graph Convolutional Neural Networks

- Spectral and spatial approaches
- Filtering graph signals
- Importance of smoothness functionals
- State-of-the-arts
- Application in semi-supervised learning

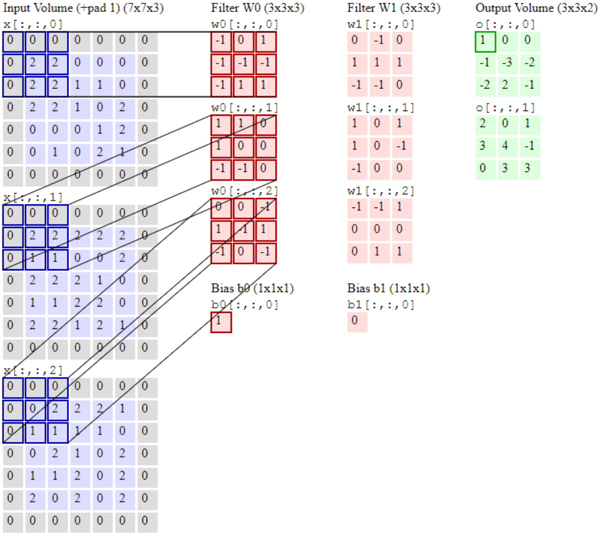
Convolutional Neural Networks

Architecture



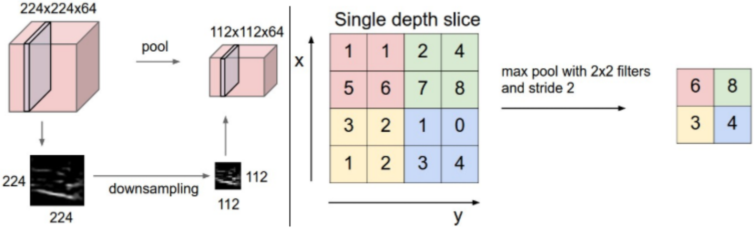
Convolutional Neural Networks

Example for convolution operation



Convolutional Neural Networks

Example for pooling operation

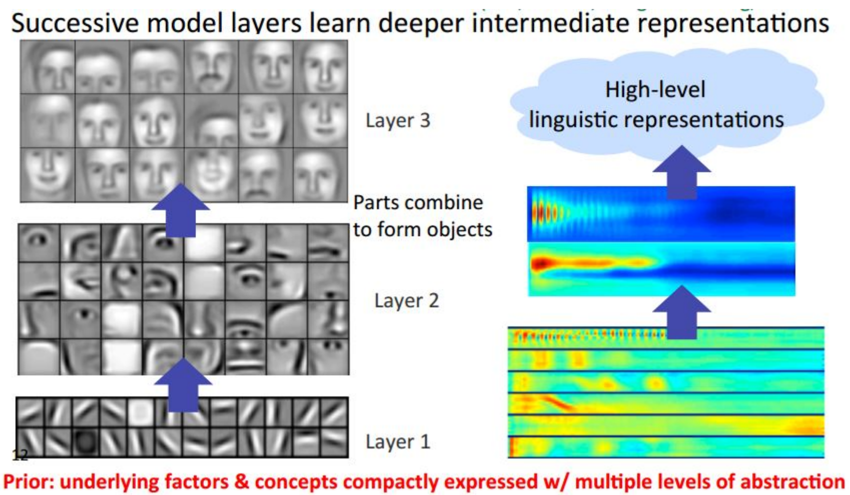


Convolutional Neural Networks

Invariance properties of CNN

Convolutional Neural Networks

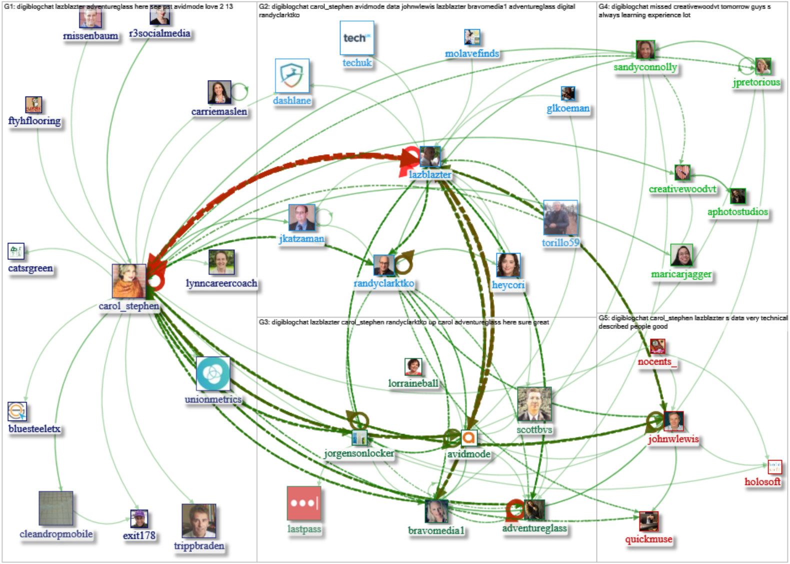
learning multiple levels of representation of increasing complexity



How to generalize Convolutional Neural Networks to "non-natural" data?

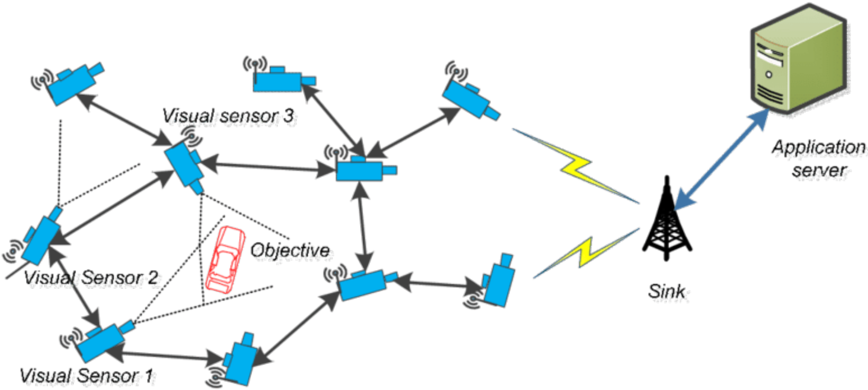
Graph Convolutional Neural Networks

Applications - social networks in computational social sciences



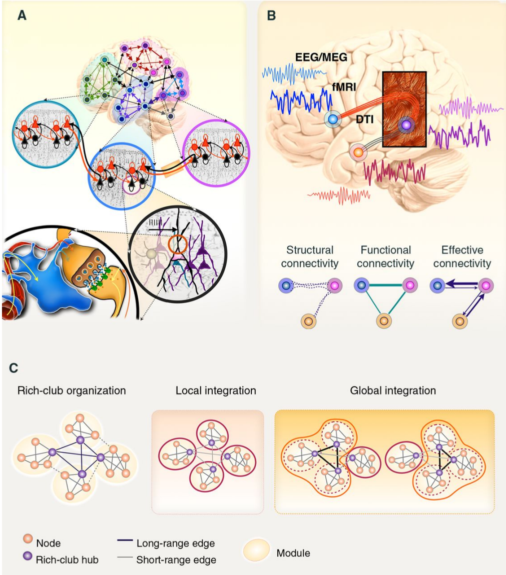
Graph Convolutional Neural Networks

Applications - sensor networks in communications



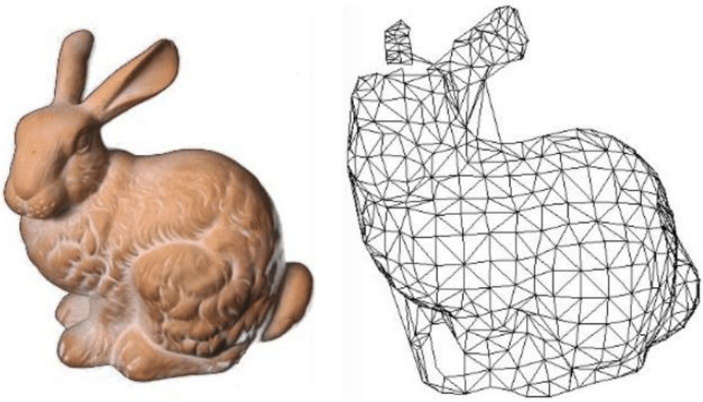
Graph Convolutional Neural Networks

Applications - functional networks in brain imaging



Graph Convolutional Neural Networks

Applications - meshed surfaces in computer graphics



1. Spectral
   - based on spectral graph theoretic approach.
   - uses signal processing techniques defined over graphs.
2. Spatial
   - directly define the convolution on the graph vertices.
   - algorithms can be deduced as a message passing mechanism.
   - For detailed discussion refer [1].

- We want to process signals defined on undirected and connected graphs $G = (V, E, W)$
  - V is a finite set of n vertices, E is a set of edges, $W \in \mathbb{R}^{n \times n}$ is a weighted adjacency matrix.

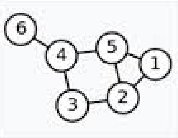
- Graph Laplacian G, is defined as $L = D - W \in \mathbb{R}^{n \times n}$ and normalized definition is $\tilde{L} = I_n - D^{-1/2} W D^{-1/2}$ where D is the degree matrix.

- As L is a real symmetric psd matrix, it has a complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1} \in \mathbb{R}^n$, known as the graph Fourier modes.

- Their associated ordered real nonnegative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$, identified as the frequencies of the graph.

- L is diagonalized by the Fourier basis $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$ such that $\tilde{L} = U \Lambda U^T$

- Example for W, D and L

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

- For any function $f \in \mathbb{R}^N$ defined on the vertices of graph G, its graph Fourier transform $\hat{f}$ is defined by

$$\hat{f}(l) = \langle u_l, f \rangle = \sum_{n=1}^{N} u_l^*(n) f(n)$$

- The inverse transform is,

$$f(n) = \sum_{l=0}^{N-1} \hat{f}(l) u_l(n)$$

- Convolution operator on graph $*_G$ is defined in the Fourier domain such that

$$f *_G y = U\big((U^T f) \odot (U^T y)\big),$$

where $\odot$ is the element-wise Hadamard product.

- We define $f = (f_1, \dots, f_n) \in \mathbb{R}^n$ as a signal on n nodes on a graph G, i.e, $f_i \in \mathbb{R} : i \leq n$ is a signal component on node: i.

- A signal f is filtered by the filter $\mathcal{F}$ as

$$y = \mathcal{F}f = U^T g_\theta(\Lambda) U f$$

  - where $g_\theta(\Lambda) \in \mathbb{R}^{n \times n} = \mathrm{diag}(g_\theta(\lambda_1), \dots, g_\theta(\lambda_n))$ is a diagonal matrix.
  - The function $g_\theta() : \mathbb{R} \to \mathbb{R}$ is defined as frequency response function of the filter $\mathcal{F}$.

- The graph filter is defined in terms of the function of eigen values of graph Laplacian, $g_\theta(\lambda)$.

- The eigenvectors of $\tilde{L}$ corresponding to lower frequencies or smaller eigenvalues are smoother on graphs.

- The smoothness corresponding to the $k-$th eigenvector can be quantified as,

$$\sum_{i \sim j} w_{ij}[u_k(i) - u_k(j)]^2 = u_k^T \tilde{L} u_k = \lambda_k$$

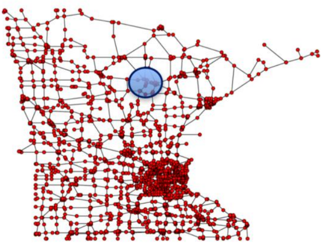
  - $\Rightarrow$ a smoothly varying graph signal will have eigenvectors with smaller eigenvalues.
  - This is under the assumption that neighborhood of topologically identical nodes would be similar.

- Smoothness functional of the entire graph is defined as,

$$S_G(f) = \sum_{i \sim j} w_{ij}(f_i - f_j)^2 = f^T \tilde{L} f \tag{1}$$

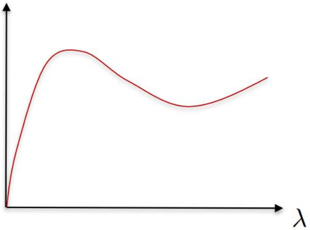
Spectral Graph Convolutional Neural Networks

Significance of smoothly varying graph signals



Spatial localization

$\Leftrightarrow$



Smooth spectral filter function

- Localization in space $\Leftrightarrow$ smoothness in frequency domain.

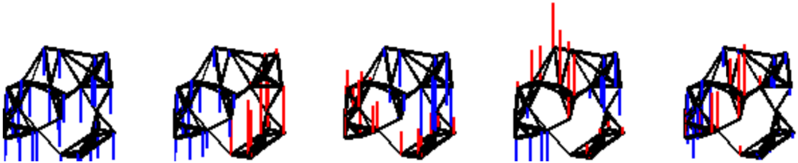


Figure 1: The first 5 eigenvectors of the normalized graph Laplacian corresponding to an arbitrary graph. Each line attached to a vertex is proportional to the value of the corresponding eigenvector at the vertex. Positive values (red) point up and negative values (blue) point down.

- Eigen vectors of smaller egien values are relatively smooth.

Spectral Graph Convolutional Neural Networks

Significance of smoothly varying graph signals - Practical considerations

- In the real world applications, the signals over the graph could be noisy.
- We should filter out high frequency content of the signal as it contains noise.
- Low frequency contents (eigenvectors corresponding to lower eigenvalues) should be maintained as it contains the robust information.
- Smoothness corresponds to spatial localization in the graphs.
- Spatial localization is important in the graph learning to infer local variability of the node neighborhoods.

Spectral GCNNs – state-of-the-arts

ChebyNet [2]

- Based on polynomial parametrization of localized filters

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k \tag{2}$$

where the parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients.

- The value at vertex j of the filter g_θ centered at vertex i is given by $(g_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$, where the kernel is localized via a convolution with a Kronecker delta function $\delta_i \in \mathbb{R}^n$.

- Localization property-
  - If d_G is the shortest path distance, $d_G(i,j) > K$ implies $(L^K)_{i,j} = 0$.
  - Spectral filters represented by K^{th} order polynomials of the Laplacian are exactly K-localized.

- Higher orders of $\tilde{L}$ is approximated with Chebyshev polynomials.

- A linear model of ChebyNet.

- Limit the convolution operation to $K = 1$ in Equation 2, i.e,

$$g_\theta \star x \approx \theta_0 x + \theta_1 (L - I_n) x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

- Successive application of filters of this form then effectively convolve the k^{th}-order neighborhood of a node.
  - where k is the number of successive filtering operations or convolutional layers in the neural network model.

- In practice, above equation is redefined as,

$$g_\theta \star x \approx \theta \left(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x.$$

with a single parameter $\theta = \theta_0 = -\theta_1$.
  - $I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ now has eigenvalues in the range [0, 1.5]

Table 1: Frequency response function and output of spectral filters of GCNNs

Network	Freq. response ($g_\theta()$)	Output, y
ChebyNet [2]	$g_\theta(\lambda) = \sum_{k=0}^{K-1} \theta_k \lambda^k$	$y = U(\sum_{k=0}^{K-1} \theta_k \Lambda^k) U^T f$
GCN [3]	$g_\theta(\lambda) = \big(\theta(1-\lambda)\big)$	$y = \theta(I - \tilde{L})f$
GraphHeat [4]	$g_\theta(\lambda) = (1 + \exp(-s\lambda))$	$y = (\theta_0 I + \theta_1 e^{-s\tilde{L}})f$
IGCN [5]	$g_\theta(\lambda) = \big(\theta(1-\lambda)\big)^K$	$y = \theta(I - \tilde{L})^K f$

Spectral GCNNs - application in semi-supervised classification

Neural network architecture

- The network architecture defined in GCN [3] consists of two layers that takes the form

$$Z = \text{softmax}(\mathcal{F}(\tilde{L})\ \text{ReLU}(\mathcal{F}(\tilde{L})X\Theta^{(1)})\Theta^{(2)})$$

  - $\mathcal{F}(\tilde{L}) \in \mathbb{R}^{n\times n}$ is the filter
  - $X \in \mathbb{R}^{n\times d}$ is the input feature matrix
  - $\theta^{(1)} \in \mathbb{R}^{d\times c_1}$ - filter parameters of first layer (c_1 is the no: of filters)
  - $\theta^{(2)} \in \mathbb{R}^{c_1\times c_2}$ - filter parameters of 2nd layer (c_2 is the no: of filters)
  - ReLU is the activation function defined as $y = max(0, x)$
  - $\text{softmax}(x_i) = \exp(x_i)/\sum_j \exp(x_j)$ applied in row wise to matrix Z.
- Cost function optimized is the cross entropy function, $\mathcal{L}$, defined as,

$$\mathcal{L} = -\sum_{i\in\mathcal{Y}}\sum_{j=1}^{c_2} y_{ij} \ln(Z_{ij})$$

  - where $\mathcal{Y}$ is the set of nodes whose labels are known
  - y_{ij} is defined as 1 if label of node i is j and 0 otherwise.

Graph Convolutional Neural Networks

Tool for implementations

PyTorch geometric

- Various methods for deep learning on graphs and other irregular structures.
- Consists of an easy-to-use mini-batch loader.
- A large number of common benchmark datasets.
- Transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds.

References

[1] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1263–1272.

[2] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in neural information processing systems, 2016, pp. 3844–3852.

[3] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[4] B. Xu, H. Shen, Q. Cao, K. Cen, X. Cheng, Graph convolutional networks using heat kernel for semi-supervised learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 1928–1934.

[5] Q. Li, X.-M. Wu, H. Liu, X. Zhang, Z. Guan, Label efficient semi-supervised learning via graph filtering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9582–9591.


Questions?