Derivation of Tensor GCN

August 2023

1 GCN

1.1 Spectral Graph Convolution

Fourier basis U $\in \mathbf{R}^{\mathbf{n}\times\mathbf{n}}$ can transform a graph signal $x \in \mathbf{R}^{\mathbf{n}}$(a scalar for every node) into $\hat{x} = U^T x$, which is the counterpart of x in the Fourier domain.
U is also the matrix of eigenvectors of the normalized graph Laplacian L_{sym}, this can be written as:
Ł$_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
$= D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$
$= D^{-\frac{1}{2}} D D^{-\frac{1}{2}} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
$= I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$,
we take L_{sym} as L in further discussion. We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in \mathbf{R}^{\mathbf{n}}$ with a filter g in the Fourier domain, the Fourier basis can transform x and g into Fourier domain or reverse them back, i.e.:$x \star g = U g_\theta U^T x$, where g_θ is a filter in the Fourier domain. We can understand g_θ as a function of the eigenvalues of L, i.e. $g_\theta(\Lambda)$.

1.2 ChebNet

$g_\theta(\Lambda)$ can be well-approximated by a truncated expansion in terms of Chebyshev polynomials:

$g_\theta = diag(U^T g) \to g_\theta(\Lambda) \approx \sum_{k=0}^{K} \beta_k T_k(\hat{\Lambda})$

i.e.: $g_\theta = \begin{pmatrix} \hat{g}(\lambda_1) & & \\ & \dots & \\ & & \hat{g}(\lambda_n) \end{pmatrix} \Rightarrow g_\theta \approx \begin{pmatrix} \sum_{k=0}^{K} \beta_k T_k\left(\hat{\lambda}_1\right) & & \\ & \dots & \\ & & \sum_{k=0}^{K} \beta_k T_k\left(\hat{\lambda}_n\right) \end{pmatrix}$,

with a rescaled $\hat{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$. Λ_{max} denotes the largest eigenvalue of L, $\beta \in \mathbf{R}^{\mathbf{K}}$ is now a vector of Chebyshev coefficients.The Chebyshev polynomials are recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$.
Going back to the definition of a convolution of a signal x with a filter g, we

now have: $x \star g = U g_\theta U^T x \approx \sum_{k=0}^{K} \beta_k T_k(\hat{L}) x$, with $\hat{L} = \frac{2}{\lambda_{max}} L - I_N$.
A neural network model based on graph convolutions can therefore be built by stacking multiple convolutional layers of the form of this:

$$x \star g \approx \sum_{k=0}^{K} \beta_k T_k(\hat{L}) x,$$

which can reduce the number of parameters that need to be optimized in the graph convolution from N(number of samples) to $K+1$(K^{th}-order Chebyshev polynomials).

1.3 GCN

GCN can be seen as a simplified version of ChebNet. The core of GCN is to limit the layer-wise convolution operation to K = 1, thus we have $T_0(\hat{L}) = I_N$, $T_1(\hat{L}) = \hat{L}$, with $\hat{L} = \frac{2}{\lambda_{max}} L - I_N$, $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. GCN further approximate $\lambda_{max} \approx 2$, under these approximations we can simplifies graph convolution:

$$\begin{aligned}
&x \star g_\theta = \textstyle\sum_{k=0}^{K} \beta_k T_k(\hat{L}) x = \sum_{k=0}^{1} \beta_k T_k(\hat{L}) x \\
&= \beta_0 T_0(\hat{L}) x + \beta_1 T_1(\hat{L}) x \\
&= \left(\beta_0 + \beta_1 \hat{L}\right) x \\
&= \left(\beta_0 + \beta_1 \left(L - I_n\right)\right) x \\
&= \left(\beta_0 - \beta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x,
\end{aligned}$$

in this formula we only have two free parameters β_0 and β_1. In practice, GCN constrains the number of parameters further to address overfitting and to minimize the number of operations per layer. Let $\beta_0 = -\beta_1 = \theta$, then we have the following expression:

$$x \star g_\theta \approx \theta(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x,$$

note that $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvalues in the range $[0, 2]$. Repeated application of this operator can therefore lead to numerical instabilities and exploding/vanishing gradients when used in a deep neural network model. To alleviate this problem, GCN introduce the following renormalization trick: $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Note that θ is a scalar here, which represents a graph convolution $x \star g$ has only a singal learnable parameter without taking input and output channels into consider.
Generalize this definition to a signal $X \in \mathbf{R^{N \times C}}$ with C input channels(C is the dimension of the feature vector) and F filters as follows:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta,$$

where $\Theta \in \mathbf{R^{C \times F}}$ is now a matrix of filter parameters, and $Z \in \mathbf{R^{N \times F}}$ is the convolved signal matrix. Note that each input channel requires a θ and so does each output channel, therefore leads to $\Theta \in \mathbf{R^{C \times F}}$.

2 Tensor GCN(TGCN)

Going back to the Chebyshev polynomials form of graph convolution: $x \star g \approx \sum_{k=0}^{K} \beta_k T_k(\hat{L})x$, GCN set K = 1 in an effort to simplify ChebNet. In Tensor GCN we set K = 2, note that $T_2(\hat{L}) = 2\hat{L}^2 - I_N$ the form of ChebNet is then:
$x \star g \approx \sum_{k=0}^{2} \beta_k T_k(\hat{L})x$
$= \beta_0 T_0(\hat{L})x + \beta_1 T_1(\hat{L})x + \beta_2 T_2(\hat{L})x$
$= \beta_0 x + \beta_1 \hat{L}x + \beta_2(2\hat{L}^2 - I_N)x$
$= [x, \hat{L}x, 2\hat{L}^2 x - x]\theta,$
with $\theta = [\beta_0, \beta_1, \beta_2]^T$. Replace $2\hat{L}^2$ with a higher dimensional laplacian $\mathcal{L}_3 \in \mathbf{R^{n \times n \times n}}$ to make the third term of the Chebyshev polynomial a higher-order supplementary term, therefore the graph convolution becomes:

$$[x, \hat{L}x, f(\mathcal{L}_3, x) - x]\theta = [x, \hat{L} \times_2 x^T, (\mathcal{L}_3 \times_2 x^T \times_3 x^T) - x]\theta,$$

Generalize this definition like GCN, we have:

$$Z = [X, \hat{L}X, M - X]\Theta,$$

with $M = [\mathcal{L}_3 \times_2 x_1^T \times_3 x_1^T, ..., \mathcal{L}_3 \times_2 x_C^T \times_3 x_C^T]$, $X \in \mathbf{R^{N \times C}}$, $\Theta \in \mathbf{R^{3C \times F}}$. Note that $\mathcal{L}_3 \times_2 x^T \times_3 x^T = L_3(x \bigotimes x)$, where '$\bigotimes$' stands for Kronecker product, $L_3 \in \mathbf{R^{N \times N^2}}$ is an unfolding of $\mathcal{L}_3$. The expression of M can therefore become:

$$M = [L_3(x_1 \bigotimes x_1), ..., L_3(x_C \bigotimes x_C)] = L_3 X * X,$$

where '*' stands for Khatri-Rao product. The simplified Tensor GCN model is as follow:

$$Z = [X, \hat{L}X, (L_3 X * X) - X]\Theta.$$