A graph is denoted by the pair (V, E), where V denotes the set of nodes and E the set of edges. The connections between vertices can be structured into an "adjacency" matrix A s.t. $A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{if } (i,j) \in E \end{cases}$

Often, we are interested in functions $f: V \to \mathbb{R}$.

In general, when we have a finite set V, the set of functions $f: V \to \mathbb{R}$ forms a $\dim(V)$ dimensional vector space with basis the functions $\delta_e(x) = \begin{cases} 1, & \text{if } x = e \\ 0, & \text{otherwise} \end{cases}$

Hence, a function is determined by how it acts on every node $x \in V$.

$$f \longleftrightarrow \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_m) \end{pmatrix}$$

$x_1, \dots, x_m$ are all nodes in V and $m = \dim(V)$

$(f(x_n))$

This is an isomorphism between $\mathbb{R}^n$ and the vector space of functions $f: V \to \mathbb{R}$. Therefore, we can think of f as a vector in $\mathbb{R}^n$.

There is an important graph operator that acts on f called the Laplacian. It is essentially the discrete or finite spaced version of the infinitesimal Laplacian ∇^2.

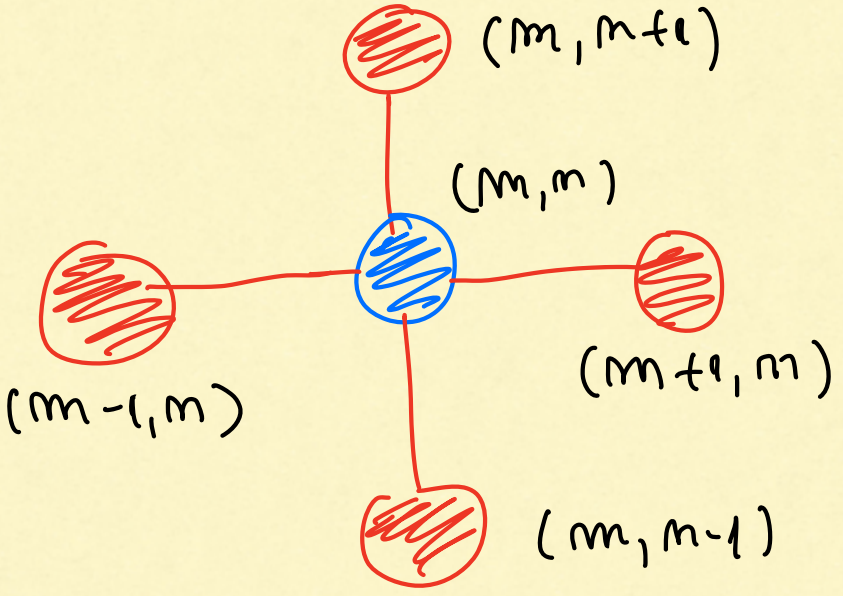
$$\text{So, } (\mathcal{L}f)(x) = \sum_{(x,y)\in E} A_{x,y}(f(x) - f(y)) \quad (1)$$
$$x \in V$$

is the sensible operator that could be defined and whose generalization to infinitesimal spacing is the usual Laplacian. We can see this by analysing a grid of points, each having four neighbors with $a_{x,y} = \frac{1}{(\delta x)^2}$ as a weight. δx being the distance between points on the evenly spaced grid. in 2D

$$(\Delta f)_{m,n} = (4f_{m,n} - f_{m+1,n} - f_{m-1,n} - f_{m,n-1} - f_{m,n+1}) / (\delta x)^2$$

which is the approximation of $\nabla^2 f$.

For getting (1) we see $L = D - A$ is the operator, where D is a diagonal matrix with $D_{ii} = \sum_{j \in V} A_{ij}$



On $\mathbb{R}$, $e^{i\omega x}$ are eigenfunctions of the 1D Laplacian operator $\frac{d}{dx^2}$. So, $f(x)$ is written as a sum of these eigenfunctions or linear combination in the finite case. Hence, in analogy it is justified to write the Fourier transform of f (now back to graphs; f above was on $\mathbb{R}$) as a linear combination of the

eigenvalues of $\mathcal{L}$.

$$\mathcal{L} X_i = \lambda_i X_i \quad i \in \overline{1,n}$$

As $\mathcal{L}$ is symmetric $\Leftrightarrow \lambda_n$ is real.

So, $\hat{f}(m) = \langle X_m, f \rangle = \sum_{n=1}^{N} X_m^*(n) f(n)$

$\hat{f}$: Fourier transform

From now I will write N as dim(V) instead of n, so that I can use n in the sum.

The inverse is thus $f(n) = \sum_{l=1}^{N} \hat{f}(l) X_l(n)$. $n \in V$

We would like to have the same interpretation as in CNN by considering $K * f$ (the convolution between a kernel K and a function f in $\mathbb{R}^N$). What should be the convolution in the current context?

If we think of $K : \mathbb{R}^+ \to \mathbb{R}^+$ as a positive

function of only one variable, then we might propose that k should act on the eigenvalues of L, i.e $k(\lambda_\ell)$ because k should somehow behave as a filter by which a strength or weights of some of the basis elements in the Fourier transform change. I want to emphasize that this convolution looks different than in CNN's, where we translated inside integrals, here translation is meaningless, since we have only coefficients in $\mathbb{N}$, but not in $\mathbb{Z}$. Therefore, we are force to disregard some of the properties since the current setting is different. Furthermore, we want $k * f$ to be as in CNN's a new function, hence

$$(k * f)(m) = \sum_{\ell=1}^{N} k(\lambda_\ell)\, \hat{f}(\ell)\, x_\ell(m)$$

$m \in V$

is a good candidate for the operation if we combine all the observations above. As you can see $k * f$ is the function whose

Fourier coefficients are $K(\lambda_\ell)\hat{f}(\ell)$, a scaled version of $\hat{f}(\ell)$. Since K has a continuous domain we can require to be a continuous function and also restrict its domain to $[0, \lambda_N]$ since $\mathcal{L}$ is positive semi-definite with positive eigenvalues and we can relabel n.t. λ_N is the biggest. Using Chebyshev polynomials to approximate K as a minimax approximation

$$\min_{p \in P_m} \max_{x \in [0, \lambda_N]} |K(x) - p(x)|.$$

P_m - the space of polynomials of degree $\leq m$.

These approximations are indeed essential because is computationally expensive to compute eigenvectors and eigenvalues. Hence,

$$(K * f)(m) = \sum_{m \in V} \sum_{\ell=1}^{N} \sum_{i=1}^{L} \theta_i T_i(\lambda_\ell) \hat{f}(\ell) x_\ell(m)$$

there is because T_i form an orthogonal basis

for the Hilbert space $L^2([0, \lambda_N], d\mu)$, where $d\mu$ is a measure scaled from $L^2([-1,1], \frac{dy}{\sqrt{1-y^2}})$ on which T_i are usually defined.

$y_m(2)$ L is the maximum degree of T_i s(deg T_i $\geq i$). If we sum with $L \to \infty$ then we get exactly K, but instead we will approximate by Truncating. Also, θ_i are the coefficients corresponding to this linear combination (T_i forms a basis as I said above).

Now, we have neural networks that are universal function approximators, so we are not limited to only Chebyshev poly. The order of polynomials increases if we stack multiple operations $K*(K*\ldots*t)$ with $L = 1$ (linear model). Therefore,

$$(K * t)(m) \approx \sum_{l=1}^{N} (\theta_0 T_0(\lambda_l) + \theta_1 T_1(\lambda_l)) \hat{t}(l) \chi_{l(m)}$$

$$\sum_{l=1}^{N} \hat{t}(l) \chi_l(m)$$

$$= \sum_{k=1} (\theta_0 + \theta_1 \lambda_k) f(k) x_k^{(m)}$$

Which in matrix notation can be written as $\theta (I_n + \Delta) f$ (3)

If we replace $\Delta \to \Delta - I_n := \tilde{\Delta}$

$\Rightarrow \theta \tilde{\Delta} f$, where θ is a vector now.

For our final architecture with input $X \in \mathbb{R}^{n \times c}$, where c is the dimension of the feature space (for example in a graph with papers we can embed the key words in vectors, so every node is associated an embedding or feature), we obtain

$$H^{(l+1)} = \phi(\tilde{\Delta} H^{(l)} W)$$ as the activation in layer $l+1$ with nonlinearity ϕ and weight matrix W. One consequence of this construction is the permutation equivariance that our graph should have.

Under a permutation of the nodes, either locally or on a whole graph:

$$\tilde{L} \rightarrow P \tilde{L} P^T$$

$$H^{(l)} \rightarrow P H^{(l)}$$

, where P is a permutation matrix.

We see that $H^{(l+1)} \rightarrow P H^{(l+1)}$ as desired.

To motivate in (3) why I replaced the eigenvalue part with L is because the transformation can be written as $U \Lambda U^T$ (the eigendecomposition of L), which is just L.