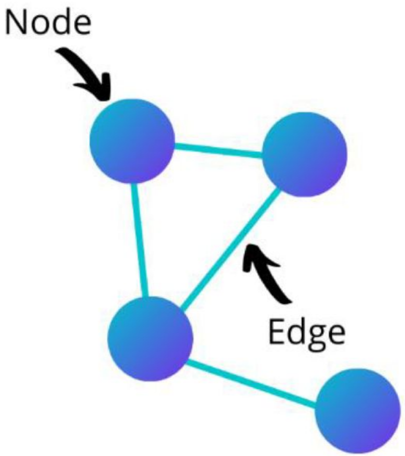
Graph Neural Network

Graph + Neural Network

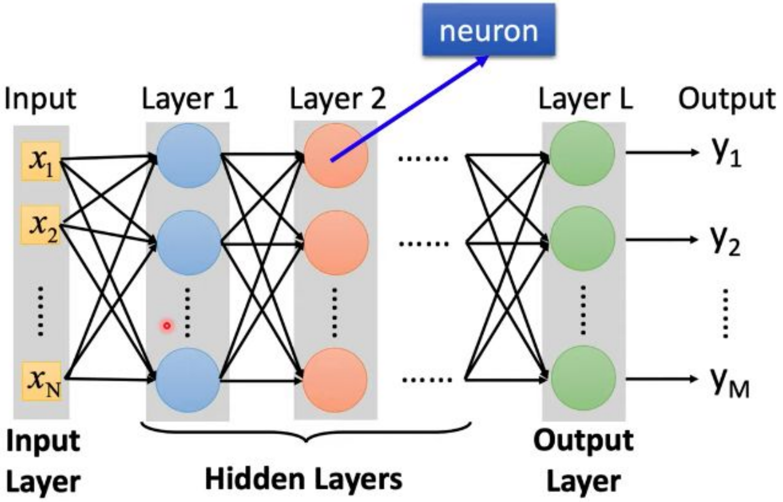
Graph



- Nodes: contain data

- Edges: specify structure (how data are related)

Neural Network



CNN

RNN

Transformer

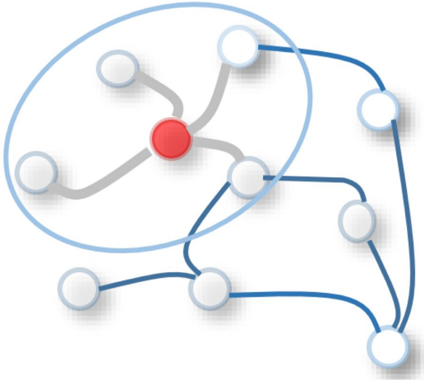
GNN Convolution Categories

Spatial-based convolutional GNN

- generalize the concept of convolution

Spectral-based convolutional GNN

- from signal processing



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image with
3 x 3 kernel

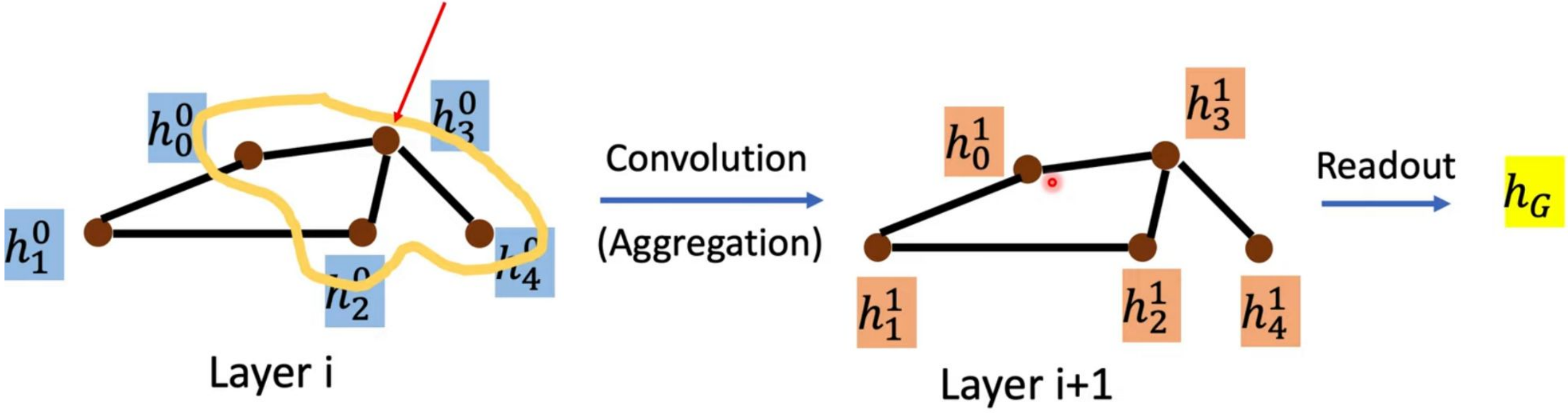
Spatial-based Convolutional Graph Neural Network

Aggregation

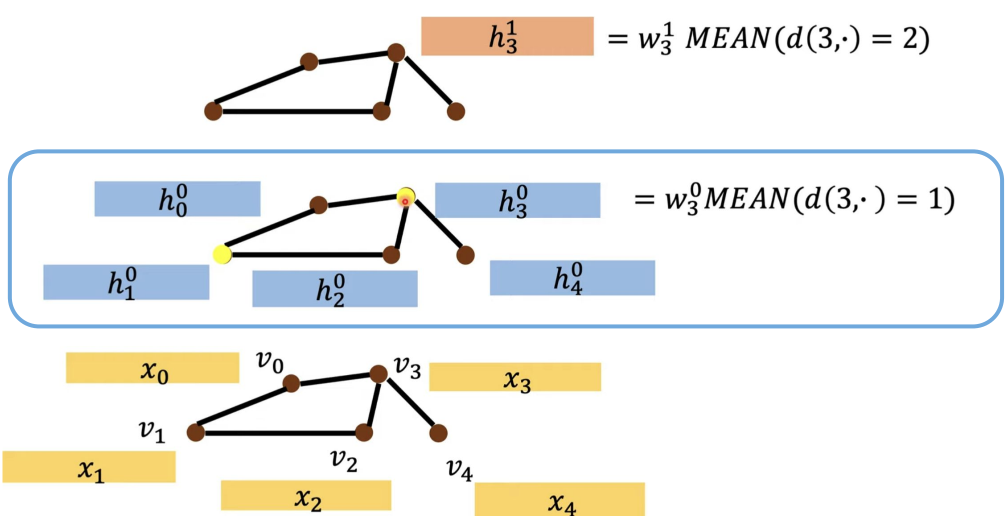
- use neighbor feature to update hidden feature in next layer

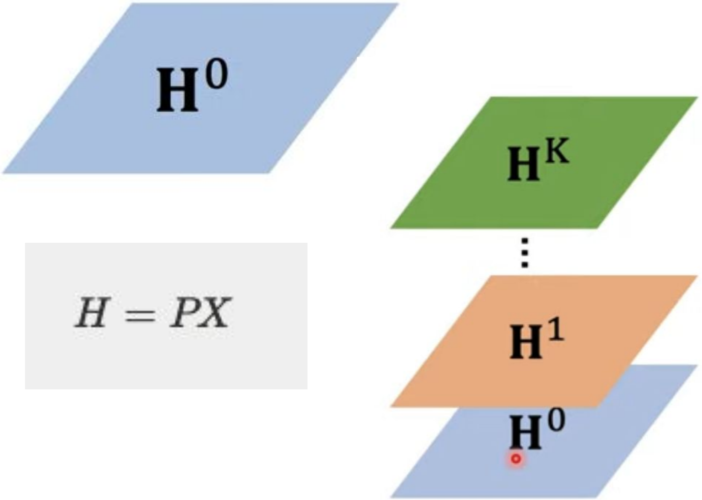
Readout

- summarizes all the nodes feature to represent the whole graph



DCNN (Diffusion-Convolution Neural Network)

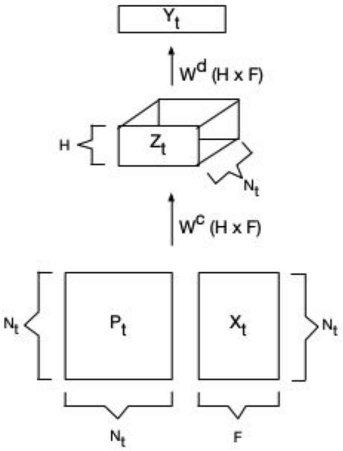




$$H = PX$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} P = D^{-1}A = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad X = \begin{bmatrix} h_0^0 \\ h_1^0 \\ h_2^0 \\ h_3^0 \\ h_4^0 \end{bmatrix} = \begin{bmatrix} f_{00} & f_{01} & f_{02} & f_{03} & f_{04} \\ f_{10} & f_{11} & f_{12} & f_{13} & f_{14} \\ f_{20} & f_{21} & f_{22} & f_{23} & f_{24} \\ f_{30} & f_{31} & f_{32} & f_{33} & f_{34} \\ f_{40} & f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix}$$

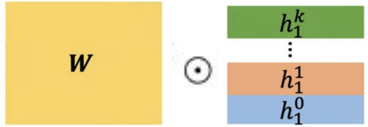
DCNN (Diffusion-Convolution Neural Network)



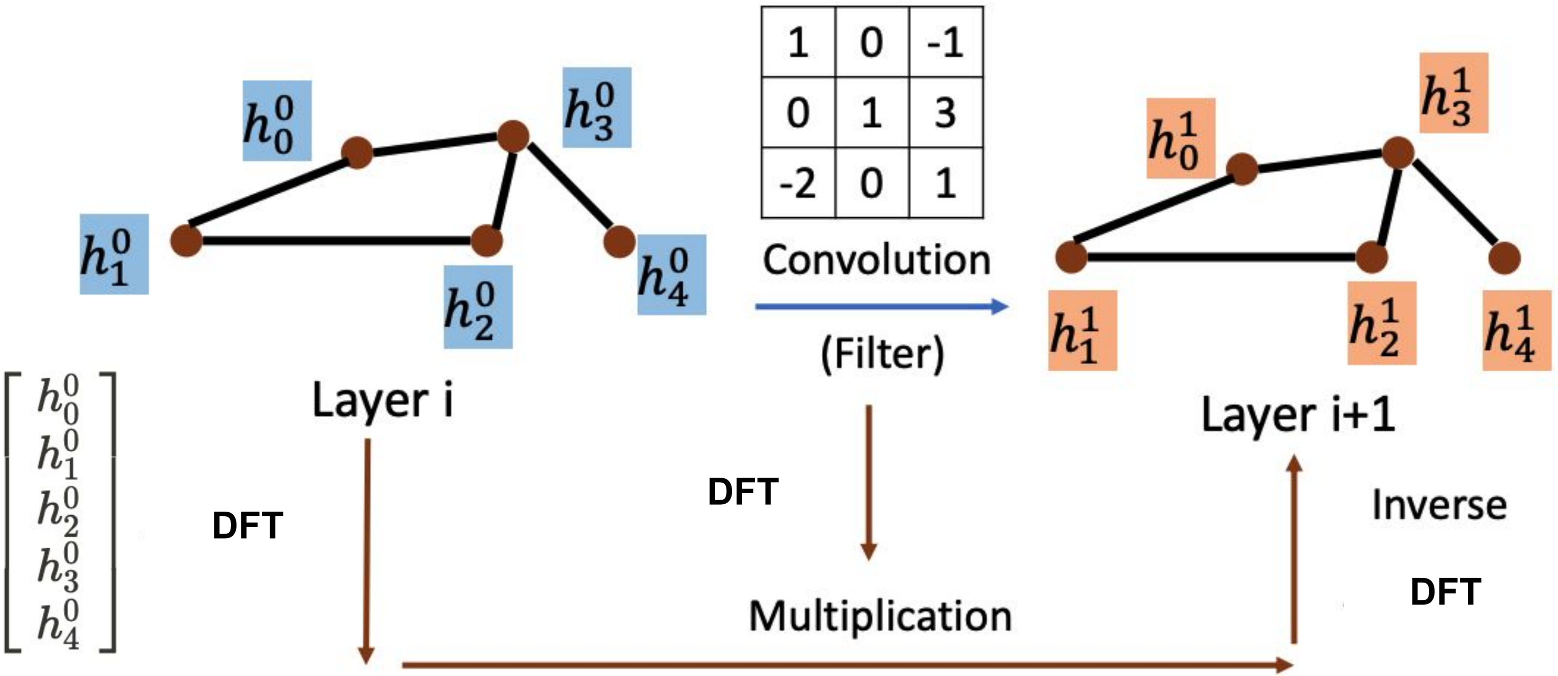
(a) Node classification

Node classification

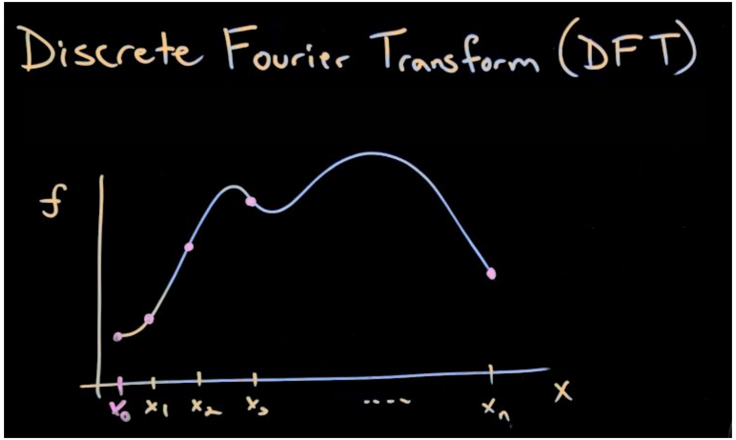
$$\hat{Y} = \arg\max \left(f \left(W^d \odot Z \right) \right)$$



Spectral-based Convolutional Graph Neural Network



Discrete Fourier Transform



$$\begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n}$$

$$f_k = \left(\sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n} \right) \frac{1}{n}$$

$$\omega_n = e^{-2\pi i/n}$$

Spectral Graph Theory

- ✓ Graph: $G = (V, E)$, $N = |V|$
- ✓ $A \in \mathbb{R}^{N \times N}$, adjacency matrix (weight matrix).

$$A_{i,j} = 0 \; if \; e_{i,j} \notin E, else \; A_{i,j} = w(i,j)$$

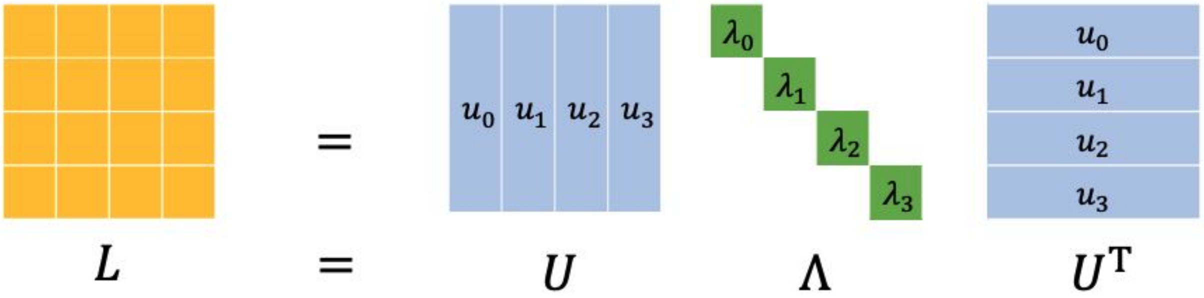
- ✓ We only consider undirected graph
- ✓ $D \in \mathbb{R}^{N \times N}$, degree matrix

$$D_{i,j} = \begin{cases} d(i) & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

(Sum of row i in A)

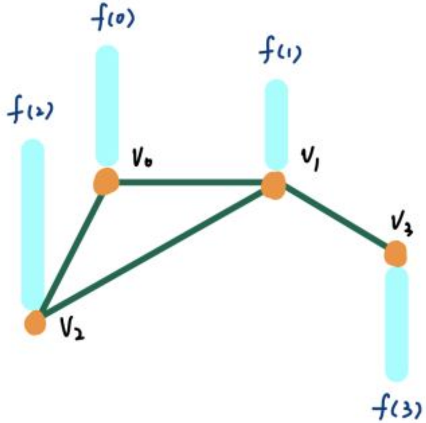
Spectral Graph Theory

- ✓ Graph Laplacian $L = D - A$ (Positive semidefinite)
- ✓ L is symmetric (for undirected graph)
- ✓ $L = U\Lambda U^{\mathrm{T}}$ (spectral decomposition)
- ✓ $\Lambda = \mathrm{diag}(\lambda_0, \dots, \lambda_{N-1}) \in \mathbb{R}^{N\times N}$
- ✓ $U = [u_0, \dots, u_{N-1}] \in \mathbb{R}^{N\times N}$, orthonormal



Spectral Graph Theory

✓ Vertex domain signal



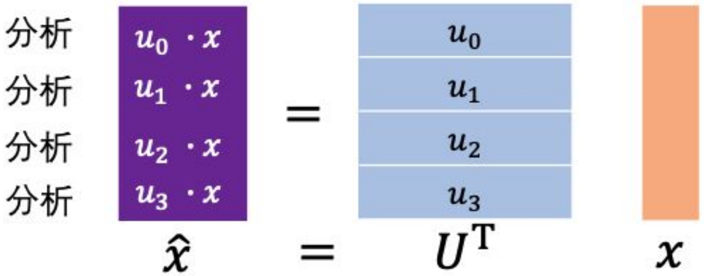
$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

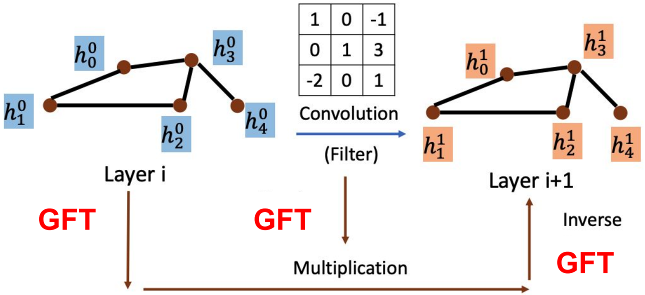
$$L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

$$f = \begin{bmatrix} 4 \\ 2 \\ 4 \\ -3 \end{bmatrix} \quad U = \begin{bmatrix} 0.5 & -0.41 & 0.71 & -0.29 \\ 0.5 & 0 & 0 & 0.87 \\ 0.5 & -0.41 & -0.71 & -0.29 \\ 0.5 & 0.82 & 0 & -0.29 \end{bmatrix}$$

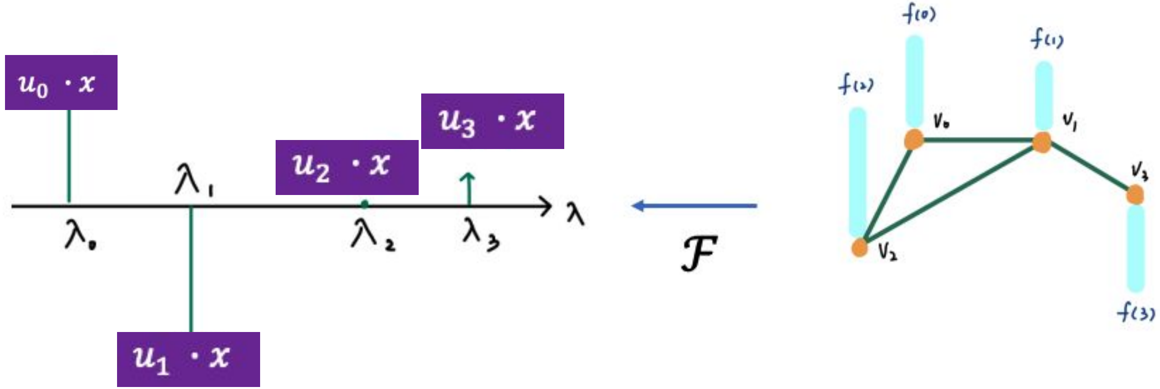
Spectral Graph Theory

✓ Graph Fourier Transform of signal x: $\hat{x} = U^T x$



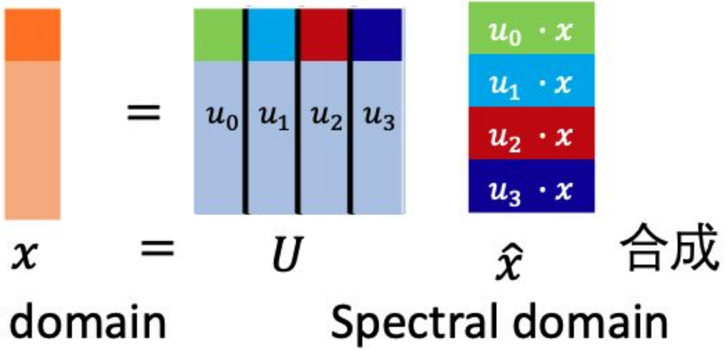


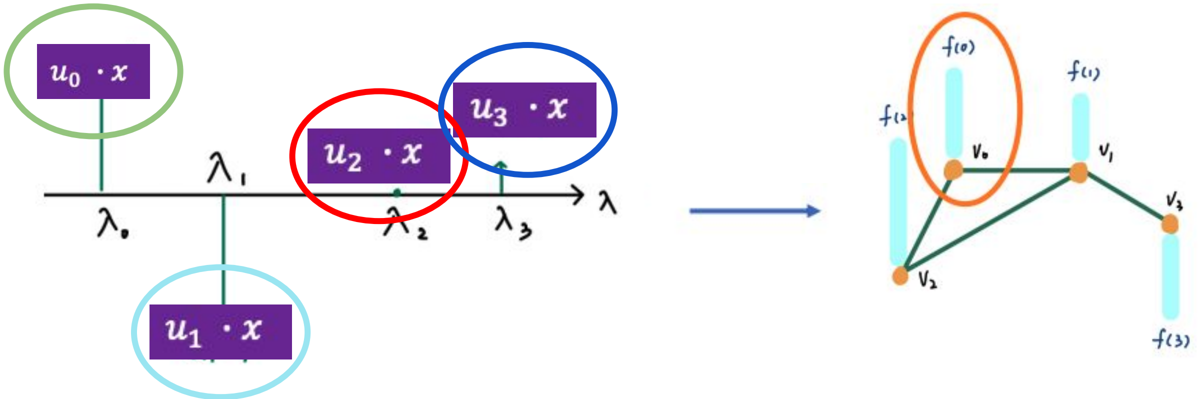
Spectral domain Vertex domain

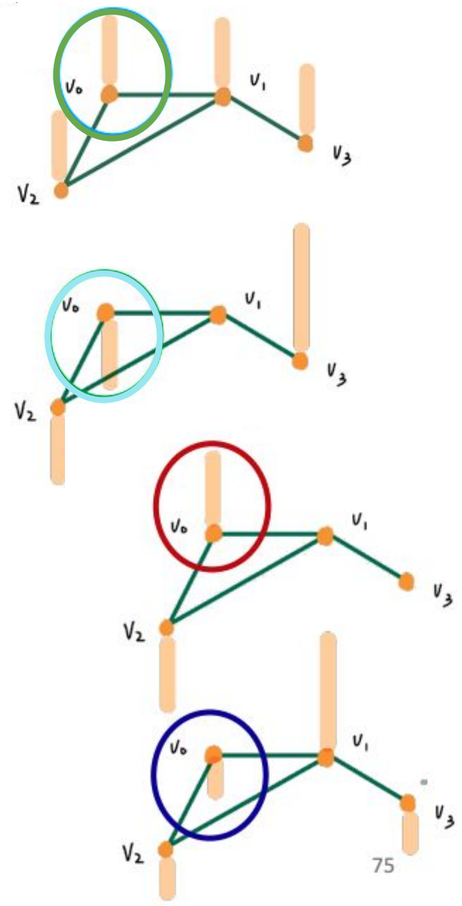


Spectral Graph Theory

- ✓ Inverse Graph Fourier Transform of signal $\hat{x}$: $x = U\hat{x}$







Spectral Graph Theory

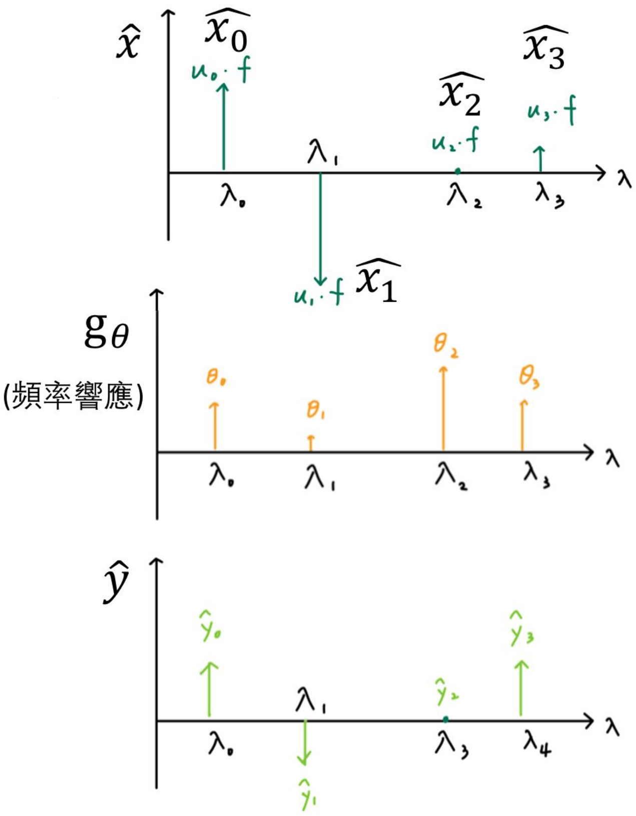
Filtering

$$\hat{y} = g_\theta(\Lambda)\,\hat{x}$$

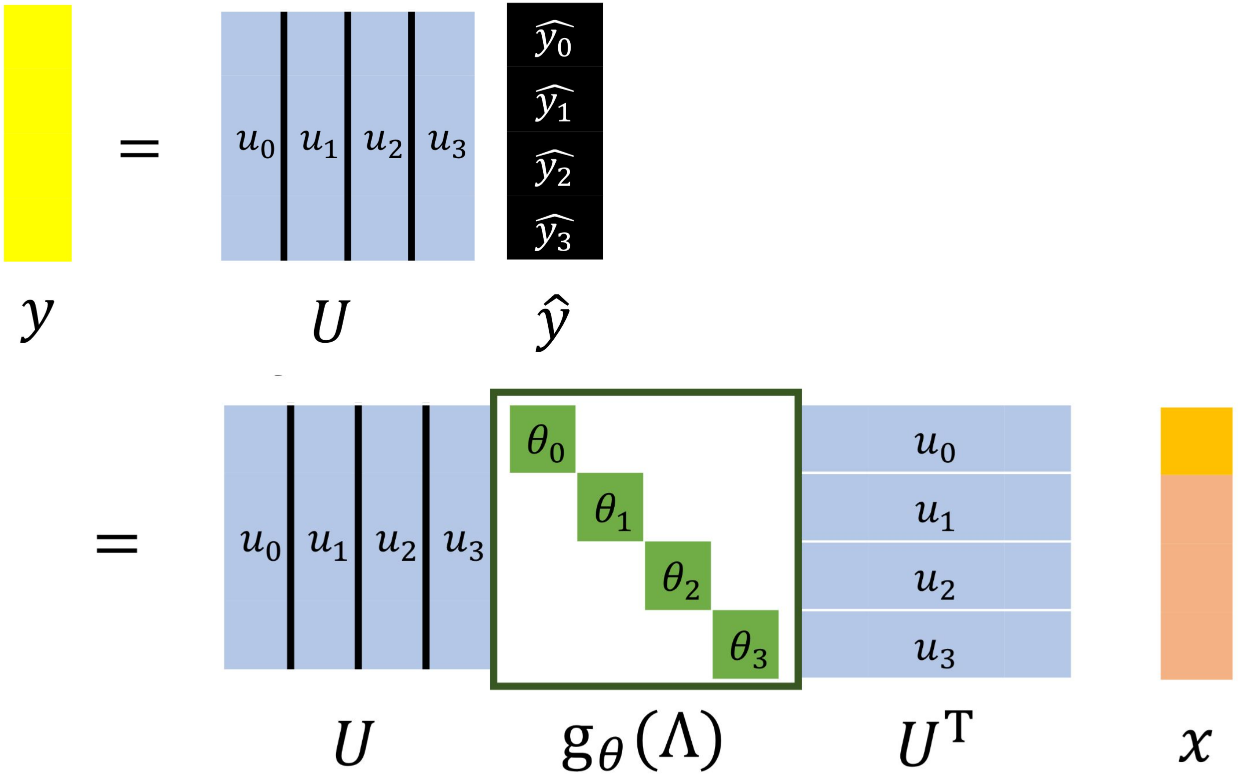
$$\begin{bmatrix} \widehat{y_0} \\ \widehat{y_1} \\ \widehat{y_2} \\ \widehat{y_3} \end{bmatrix} = \begin{bmatrix} \theta_0 & & & \\ & \theta_1 & & \\ & & \theta_2 & \\ & & & \theta_3 \end{bmatrix} \begin{bmatrix} \widehat{x_0} \\ \widehat{x_1} \\ \widehat{x_2} \\ \widehat{x_3} \end{bmatrix}$$

$$\hat{y} \quad = \quad g_\theta(\Lambda) \quad \hat{x}$$

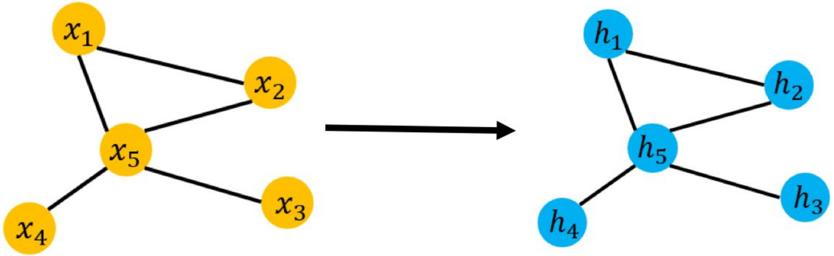
$$g_\theta(\lambda_\mathrm{i}) = \theta_i$$

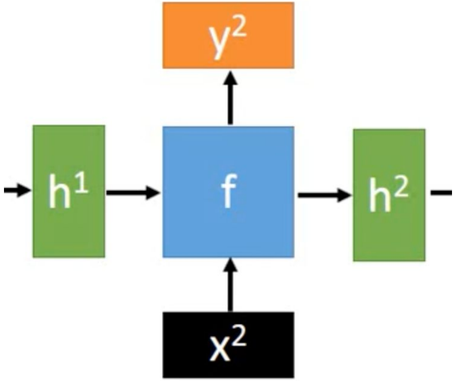


Spectral Graph Theory



Recurrent Graph Neural Networks (RecGNNs)





- A node **exchanges information** with its neighbors

$$h_v^{(t)} = \sum_{u \in N(v)} f_W(x_v, x_{(v,u)}^e, x_u, h_u^{(t-1)})$$

Node features   Node inputs   Node features

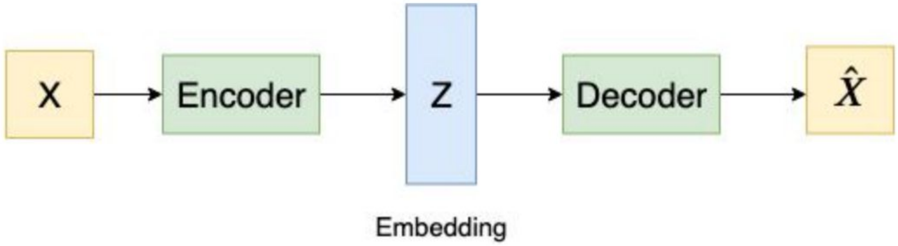
Gated Graph Recurrent Neural Networks

$$h_v^{(t)} = GRU\left(h_v^{(t-1)}, \sum_{u \in N(v)} \mathrm{W} h_u^{(t-1)}\right)$$

$h_v^{(0)} = x_v$ is node input

Auto-Encoders in Graph

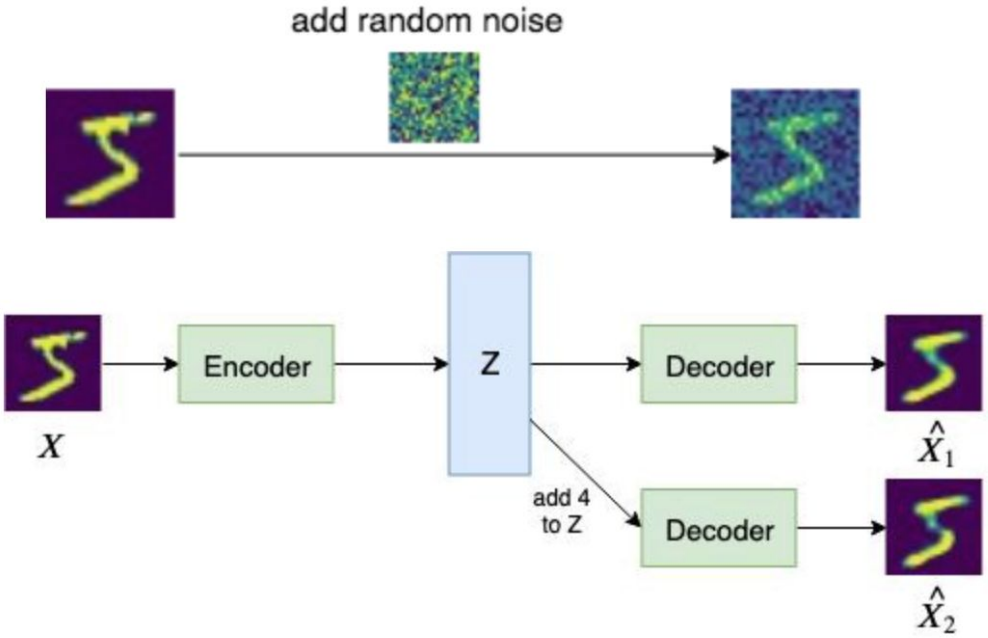
Traditional Auto-Encoders

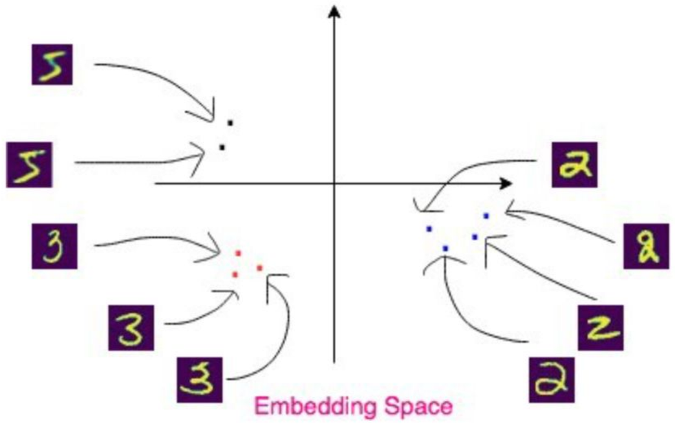


$$L(X, \hat{X}) = ||X - \hat{X}||^2$$

Advantage of using auto-encoder

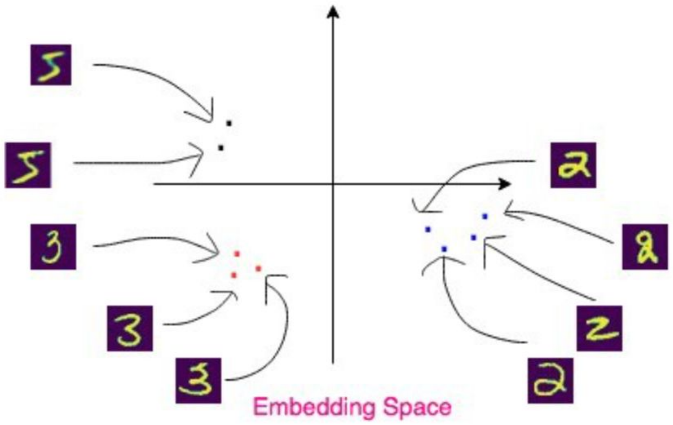
- save storage and computer resources
- noise

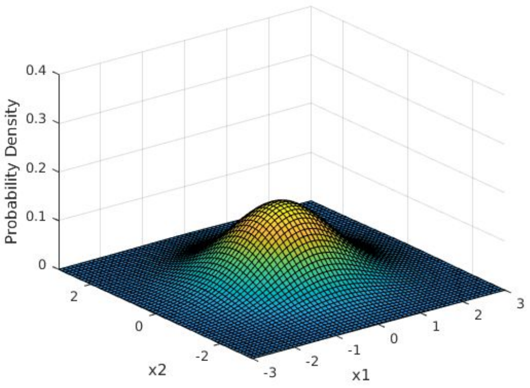




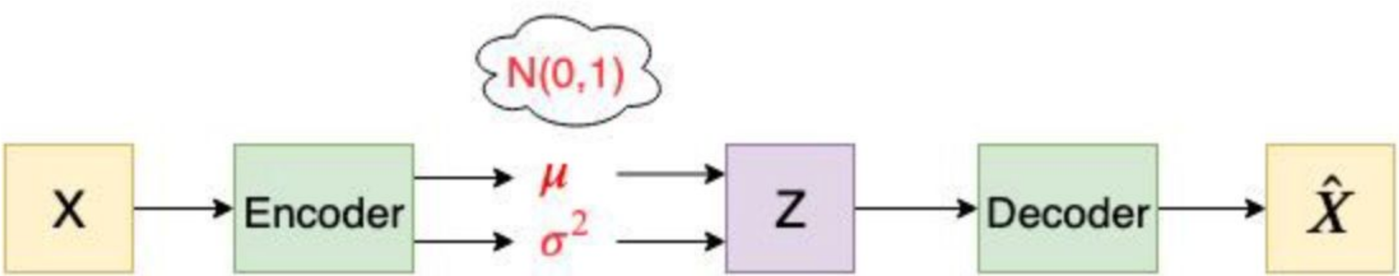
Variational auto-encoder

- can generate new data
- embed to a distribution rather than a point





Variational auto-encoder



$q_\phi(z|x_i)$ $p_\theta(x_i|z)$

$$l_i(\theta,\phi) = -E_{z\sim q_\phi(z|x_i)}[log_{p_\theta}(x_i|z)] + KL(q_\phi(z|x_i)||p(z))$$

Loss function

$$p(X, Z; \theta) = p(X; \theta) p(Z|X; \theta)$$

$$\Rightarrow \log p(X, Z; \theta) = \log p(X; \theta) + \log p(Z|X; \theta)$$

$$\Rightarrow \log p(X; \theta) = \log p(X, Z; \theta) - \log p(Z|X; \theta)$$

$$\Rightarrow \text{introduce an arbitrary distribution } q(Z) \text{ on both sizeds and integrate over } Z$$

$$\begin{aligned}
\int q(Z) \log p(X; \theta) dZ &= \int q(Z) \log p(X, Z; \theta) dZ - \int q(Z) \log p(Z|X; \theta) dZ \\
&= \left(\int q(Z) \log p(X, Z; \theta) dZ - \int q(Z) \log q(Z) dZ \right) \\
&+ \left(q(Z) \log q(Z) dZ - \int q(Z) \log p(Z|X; \theta) dZ \right) \\
&= L(X, q, \theta) + KL(q(Z) || p(Z|X; \theta))
\end{aligned}$$

Loss function

note that the first term doesn't change,

$$\int q(Z)\log p(X;\theta)dZ = \log p(X;\theta)\int q(Z)dZ = \log p(X;\theta) \because p(X;\theta) \text{ is irrelevant to } Z$$

$$\Rightarrow \log p(X;\theta) = L(X,q,\theta) + KL(q(Z)||p(Z|X;\theta)),$$

$$\text{where} \begin{cases} \begin{aligned} L(X,q,\theta) &= \int q(Z)\log p(X,Z;\theta)dZ - \int q(Z)\log q(Z)dZ \\ &= E_{Z\sim q(Z)}[\log p(X,Z;\theta)dZ] - E_{Z\sim q(Z)}\log q(Z) \\ &= E_{Z\sim q(Z)}\log p(X|Z;\theta) + E_{Z\sim q(Z)}\log p(Z;\theta) - E_{Z\sim q(Z)}\log q(Z) \leftarrow (1.) \end{aligned} \\ KL(q(Z)||p(Z|X;\theta)) = \int q(Z)\log\frac{q(Z)}{p(Z|X;\theta)}dZ \end{cases}$$

Loss function

note that KL divergence is non-negative, hence $\log p(X;\theta) \geq L(X,q,\theta)$,

we can increase $\log p(X;\theta)$ by increasing $L(X,q,\theta)$

$\Rightarrow L(X,q,\theta) = \log p(X;\theta) - KL(q(Z)||p(Z|X;\theta))$

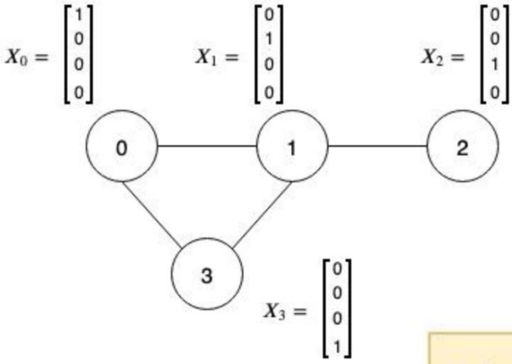
$\Rightarrow$note that the equality holds for ant choice of $q(Z)$,

we can introduce a distribution$q(Z|X;\theta')$

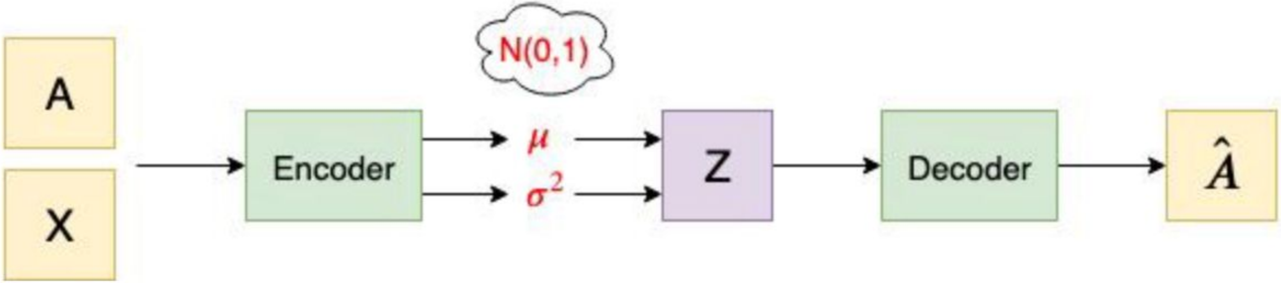
modeled by another neural network with parameter θ' to obtain,

$$
\begin{aligned}
L(X,q,\theta) &= \log p(X;\theta) - KL(q(Z|X;\theta')||p(Z|X;\theta)) \\
&= E_{Z\sim q(Z|X;\theta')}\log p(X|Z;\theta) + E_{Z\sim q(Z|X;\theta')}\log p(Z) - E_{Z\sim q(Z|X;\theta')}\log q(Z|X;\theta') \because (1.) \\
&= E_{Z\sim q(Z|X;\theta')}\log p(X|Z;\theta) - KL(q(Z|X;\theta')||p(Z))
\end{aligned}
$$

Variational Graph Auto-Encoder



$$X = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$



$$L = E_{q(Z|X,A)}[logp(A|Z)] - KL[q(Z|X,A)||p(Z)]$$

Variational Graph Auto-Encoder