Summary Questions of the last lecture

- What is the key idea to mitigate the scalability problem of SSL for millions of data?

→ We approximate the graph signal by a linear combination of only the k smoothest eigenvectors with k Fourier coefficients and then we the k Fourier coefficients are used for optimization variables instead of the N elements of the graph signal.

$$\boldsymbol{f}^* = \underset{\boldsymbol{f}\in\mathbb{R}^{n_l+n_u}}{\operatorname{argmin}} (\boldsymbol{f}-\boldsymbol{y})^T \boldsymbol{C}(\boldsymbol{f}-\boldsymbol{y}) + \boldsymbol{f}^T \boldsymbol{L}\boldsymbol{f}$$

$$\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \text{ and } \boldsymbol{f} = \boldsymbol{U}\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}\in\mathbb{R}^{n_l+n_u}}{\operatorname{argmin}} (\boldsymbol{U}\boldsymbol{\alpha}-\boldsymbol{y})^T \boldsymbol{C}(\boldsymbol{U}\boldsymbol{\alpha}-\boldsymbol{y}) + \boldsymbol{\alpha}^T \boldsymbol{\Lambda}\boldsymbol{\alpha}$$

only first k eigenvectors in $\boldsymbol{f} = \bar{\boldsymbol{U}}\bar{\boldsymbol{\alpha}}$!

$$\bar{\boldsymbol{\alpha}}^* = (\bar{\boldsymbol{\Lambda}} + \bar{\boldsymbol{U}}^T \boldsymbol{C}\bar{\boldsymbol{U}})^{-1}\bar{\boldsymbol{U}}^T \boldsymbol{C}\boldsymbol{y}$$

It requires $k \times k$ *matrix inversion*

Summary Questions of the lecture

- What happens to $\boldsymbol{L}$ when $N \to \infty$?

→ As $N \to \infty$, Laplacian ($\boldsymbol{L}$) smoothness normalized by N^2 over a graph signal $\boldsymbol{f}$ defined on data set $\{x_i\}$ sampled from $p(x)$ converges to a weighted smoothness operator on a continuous function $F(x)$ defined over a measure space $\mathcal{X}$.

$$\mathcal{L}_p(F) = \frac{1}{2}\int \big(F(\boldsymbol{x}_1) - F(\boldsymbol{x}_2)\big)^2 W(\boldsymbol{x}_1, \boldsymbol{x}_2) p(\boldsymbol{x}_1) p(\boldsymbol{x}_2) d\boldsymbol{x}_1 d\boldsymbol{x}_2$$

where $W(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2)/(2\sigma^2)$

As $N \to \infty$, $\frac{1}{N^2}\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} = \frac{1}{2N^2}\sum_{i,j} W_{ij}\big(f_i - f_j\big)^2 \to \mathcal{L}_p(F)$

Summary Questions of the lecture

- What does the weighted smoothness operator in Hilbert space define, instead of eigenvectors of Laplacian L?

→ It defines eigenfunctions, each of which represents a function having a unique frequency (eigenvalue) in the Hilbert space.

First eigenfunction: $\phi_1(\boldsymbol{x}) = \min_{F:\int F^2(\boldsymbol{x})p(\boldsymbol{x})D(\boldsymbol{x})d\boldsymbol{x}=1} \mathcal{L}_p(F)$

What is the solution? $\phi_1(\boldsymbol{x}) = 1$ because $\mathcal{L}_p(1) = 0$

How to define $\phi_2(\boldsymbol{x})$? same, constraining to be orthogonal to $\phi_1(\boldsymbol{x})$

$$\int F(\boldsymbol{x})\phi_1(\boldsymbol{x})\, p(\boldsymbol{x})D(\boldsymbol{x})d\boldsymbol{x} = 0$$

How to define eigenvalues? $\lambda_k = \mathcal{L}_p(\phi_k)$

Summary Questions of the lecture

- How can we obtain numerical eigenfunctions with sampled large dada?

→ After rotating the coordinate of the sample space using PCA, we divide each coordinate into B number of bins, and determine the discrete probability distribution by counting the number of nodes (samples) that fall into each bin. Then, we solve the generalized eigenvalue problem to obtain an approximate eigenfunction having a constant in each bin.

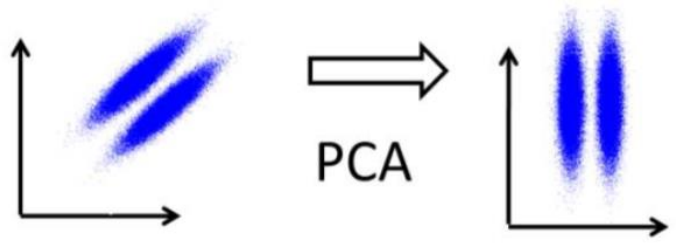
Find $\boldsymbol{R}$ such that $\boldsymbol{s} = \boldsymbol{Rx}$

For each "independent" s_k approximate $p(s_k)$

Given $p(s_k)$ numerically solve for eigensystem of $\mathcal{L}_{p_k}$

$$\left(\widetilde{\boldsymbol{D}} - \boldsymbol{P}\widetilde{\boldsymbol{W}}\boldsymbol{P}\right)g = \lambda\widetilde{\boldsymbol{D}}g$$

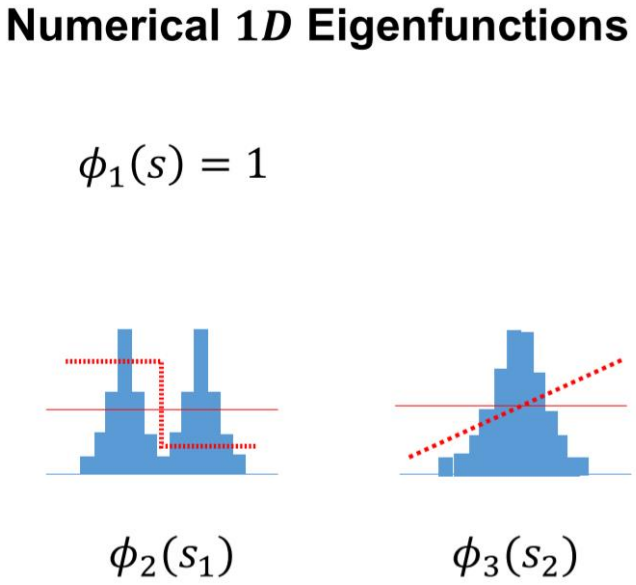
g – vector of length B ≡ number of bins

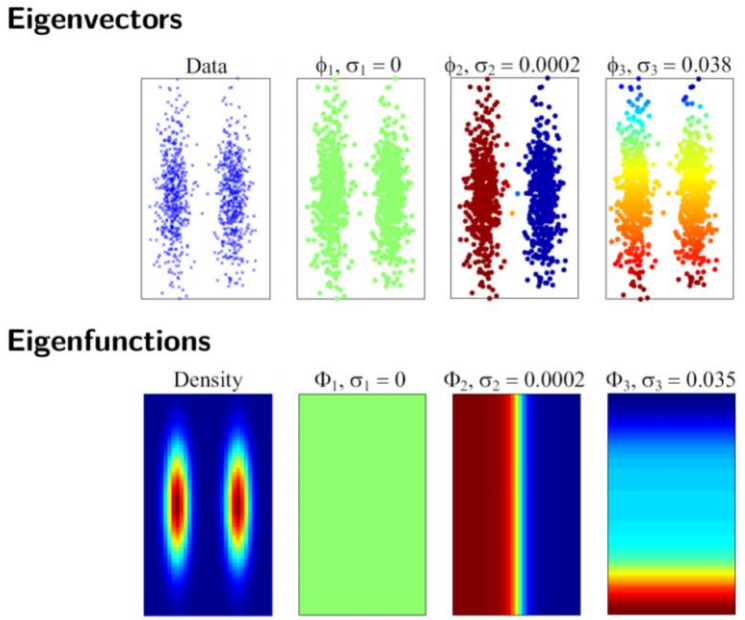


Summary Questions of the lecture

- How can we determine eigenvectors from the numerical eigenfunctions?

→ Each element value of an eigenvector is assigned by the eigenfunction value of the bin where the corresponding sample (node) falls into.



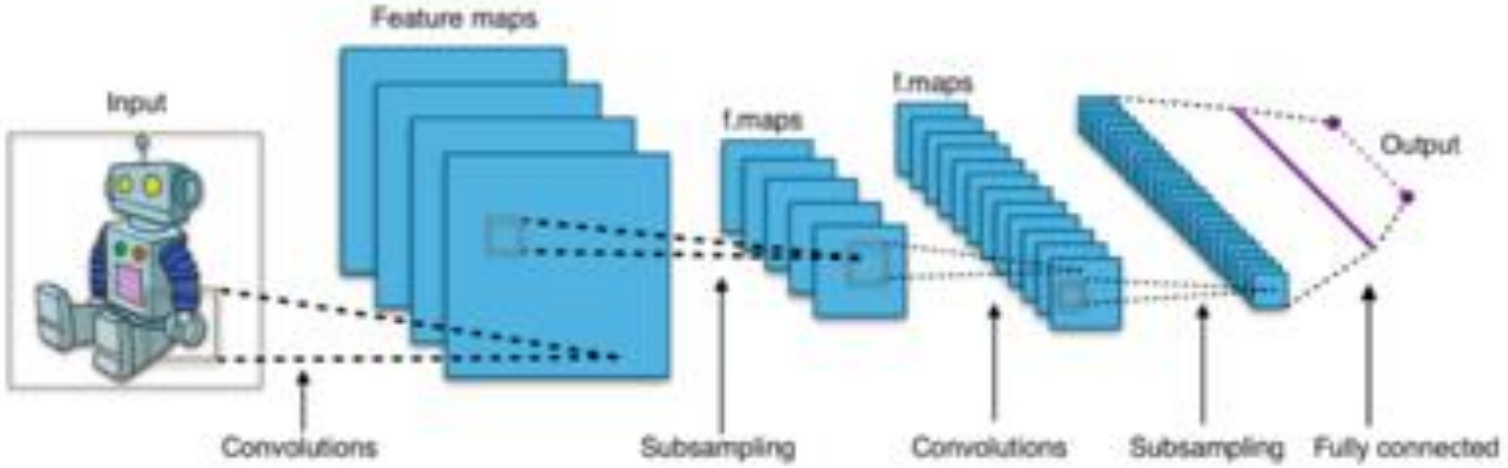


Outline of Lecture (2)

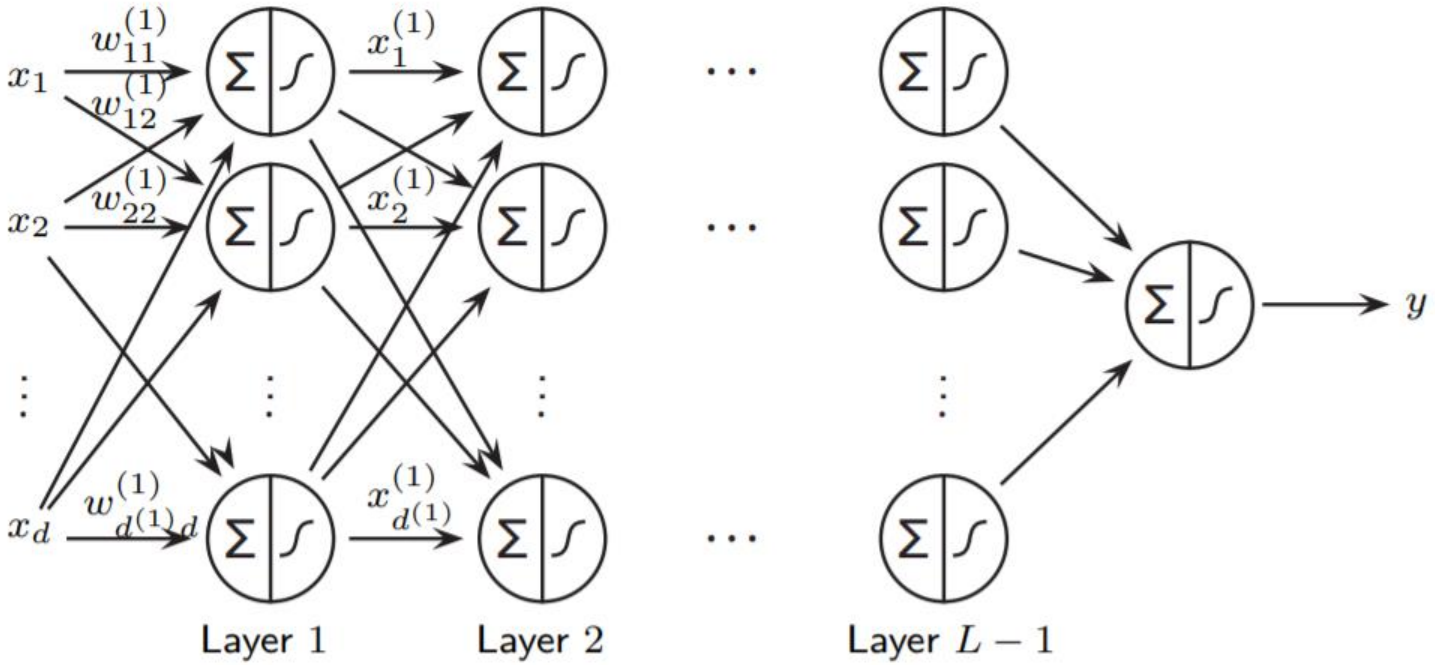
- Semi-supervised Learning (SSL) : conti.
  - SSL with Graph using Regularized Harmonic Functions
  - SSL with Graph using Soft Harmonic Functions
  - SSL with Graph using Manifold Regularization (out of sample extension)
  - SSL with Graph using Laplacian SVMs
  - SSL with Graph using Max-Margin Graph Cuts
  - Online SSL
  - SSL for large graph
- Review: Convolution Neural Networks (CNN)
  - Feedforward Neural Networks
  - Convolution Integral (Temporal)
  - Convolution Sum (Temporal)
  - Circular Convolution Sum
  - Convolution Sum (Spatial)
  - Convolutional Neural Networks
- Graph Convolution Networks (GCN)
  - What are issues on GCN
  - Graph Filtering in GCN
  - Graph Pooling in GCN
  - Spectral Filtering in GCN
  - Spatial Filtering in GCN
- Recent GCN papers
  - Original GNN (Scarselli et al. 2005)
  - Spectral Graph CNN (Bruna et al. ICLR 2014)
  - GCN (Kipf & Welling. ICLR 2017)
  - GraphSage (Hamilton et al. NIPS 2017)
  - GAT (Veličković et al. ICLR 2018)
  - MPNN (Glimer et al. ICML 2017)
- Link Analysis
  - PageRank
  - Diffusion
  - Recent Diffusion Papers

CNN

Brief **Review** on Convolution Neural Network



R: Feedforward Neural Networks



Convolution Neural Networks?

Linear layer $\mathbf{x}^{(l+1)} = \xi\left(\mathbf{W}^{(l+1)}\mathbf{x}^{(l)}\right)$

Activation, e.g. $\xi(x) = \tanh(x)$

Parameters layer weights $\mathbf{W}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$

R: Convolutional Neural Networks

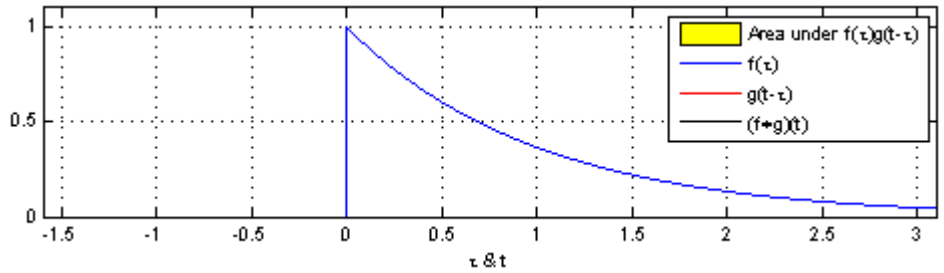
Convolution Integral (Temporal)

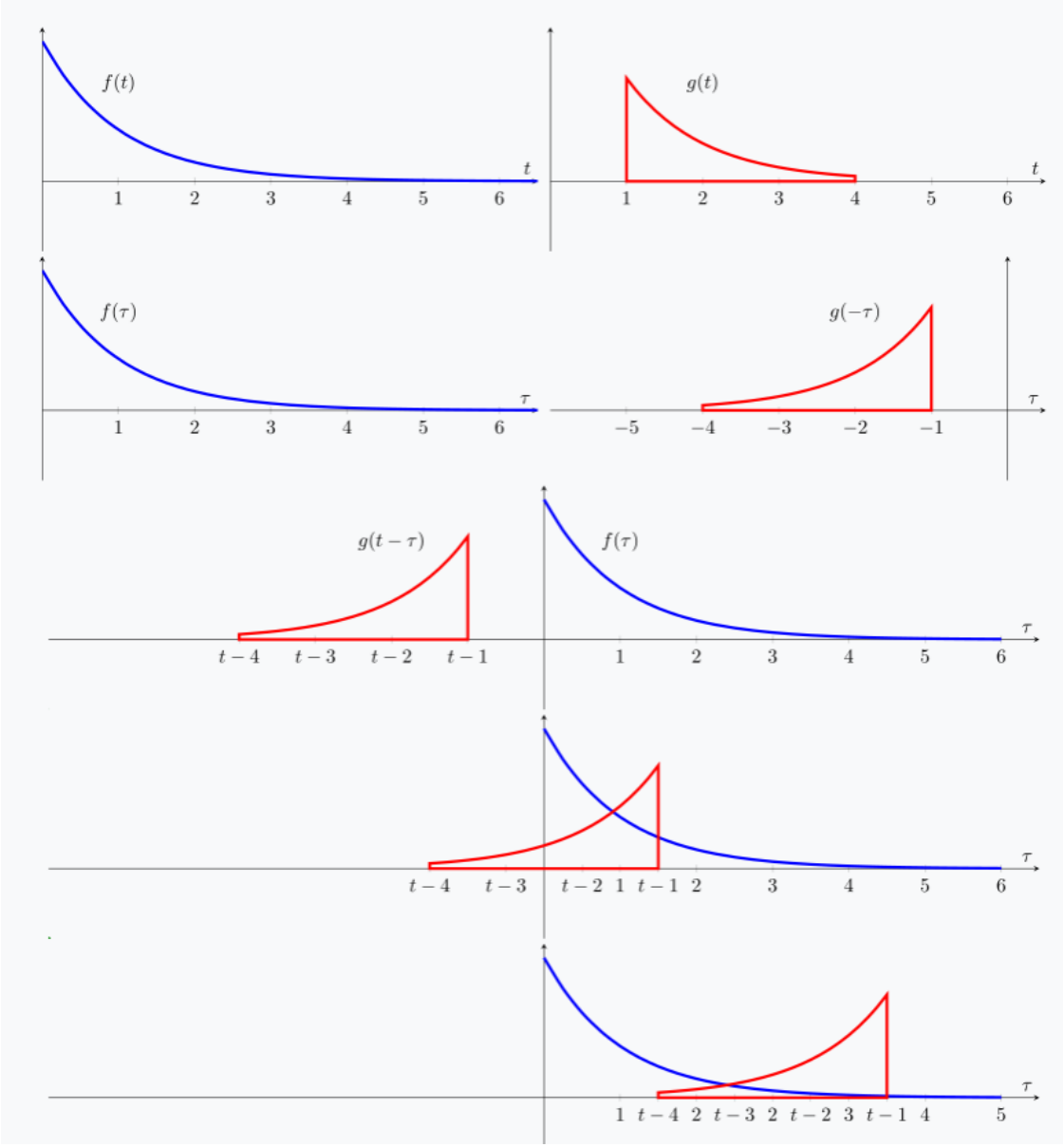
$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau$$

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\, d\tau$$

$$L \circ (f * g) = F(s)G(s)$$

cf. cross correlation



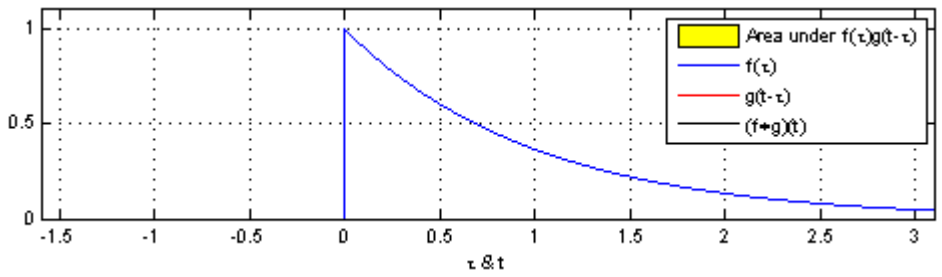


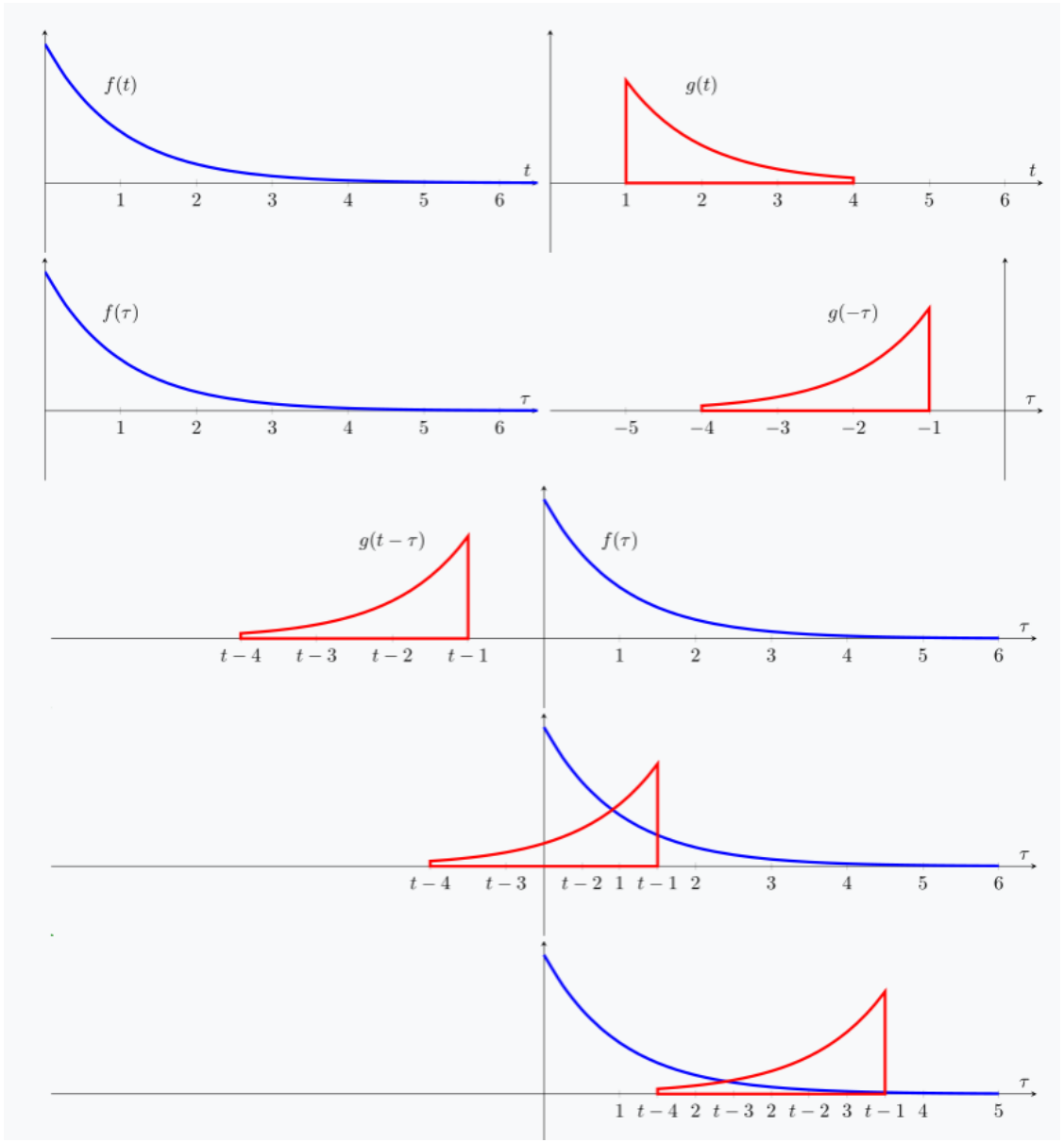
R: Convolutional Neural Networks

Convolution Sum (Temporal)

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$





R: Convolutional Neural Networks

Circular Convolution Sum

$$h_k = f * g = \sum_{i=0}^{n-1} f_i g_{k-i}$$

$$g \triangleq (\ldots, g_{n-1}, g_0, g_1, \cdots, g_{n-1}, g_0, g_1, \cdots, g_{n-1}, \ \ldots)$$
$$f \triangleq (f_0, f_1, \cdots, f_{n-1})$$
$$h \triangleq (\ldots, h_{n-1}, h_0, h_1, \ldots, h_{n-1}, h_0, h_1, \ldots, h_{n-1}, \ \ldots)$$

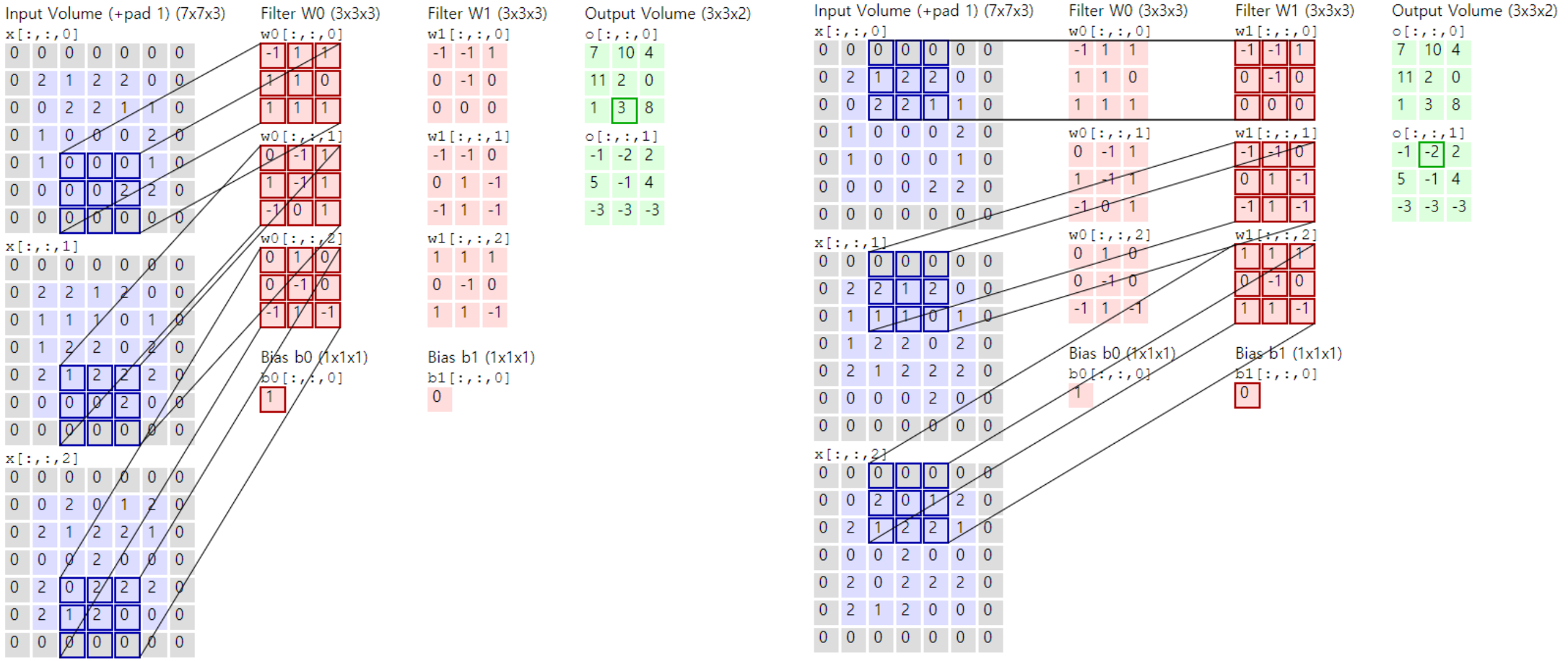
$$f * g = \begin{bmatrix} g_0 & g_{n-1} & \cdots & g_2 & g_1 \\ g_1 & g_0 & g_{n-1} & \cdots & g_2 \\ \vdots & g_1 & g_0 & \ddots & \vdots \\ g_{n-1} & \vdots & \ddots & \ddots & g_{n-2} \\ g_n & g_{n-1} & \cdots & g_1 & g_0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}$$

➡ Correlation Filter

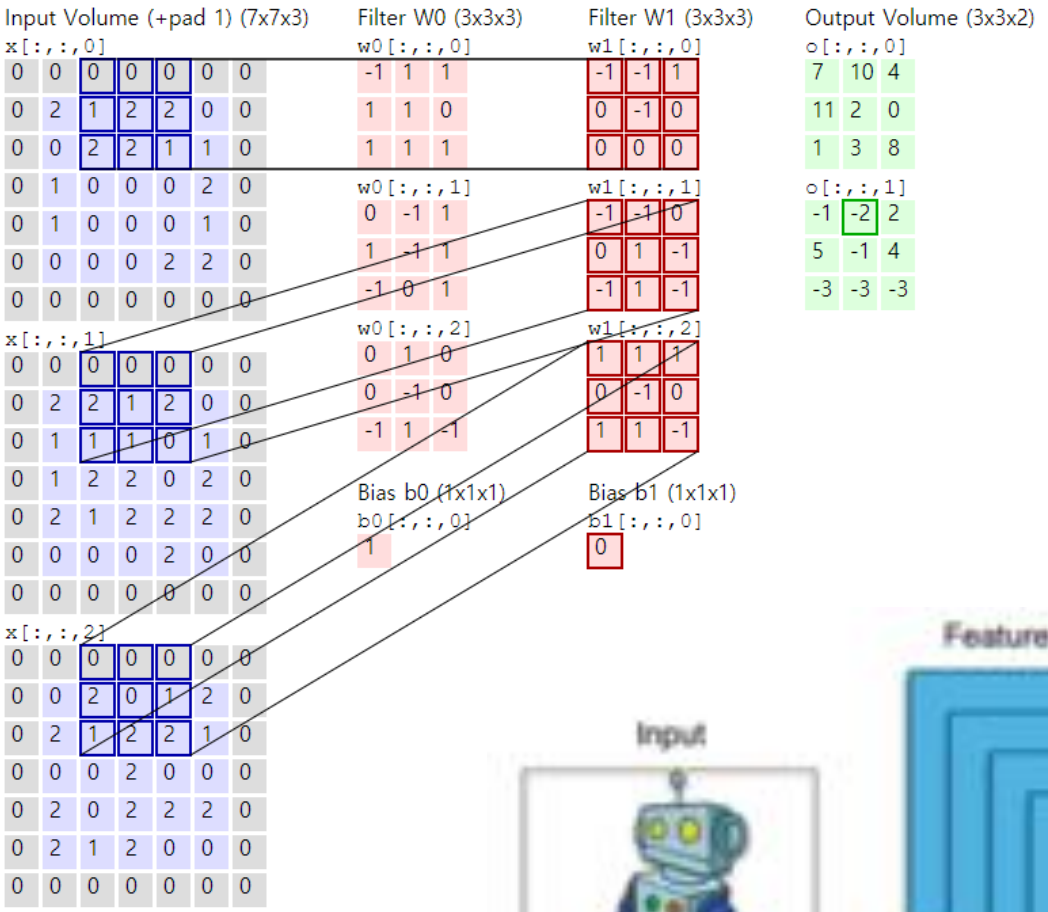
Circulant Matrix

R: Convolutional Neural Networks

Convolution Sum (Spatial)



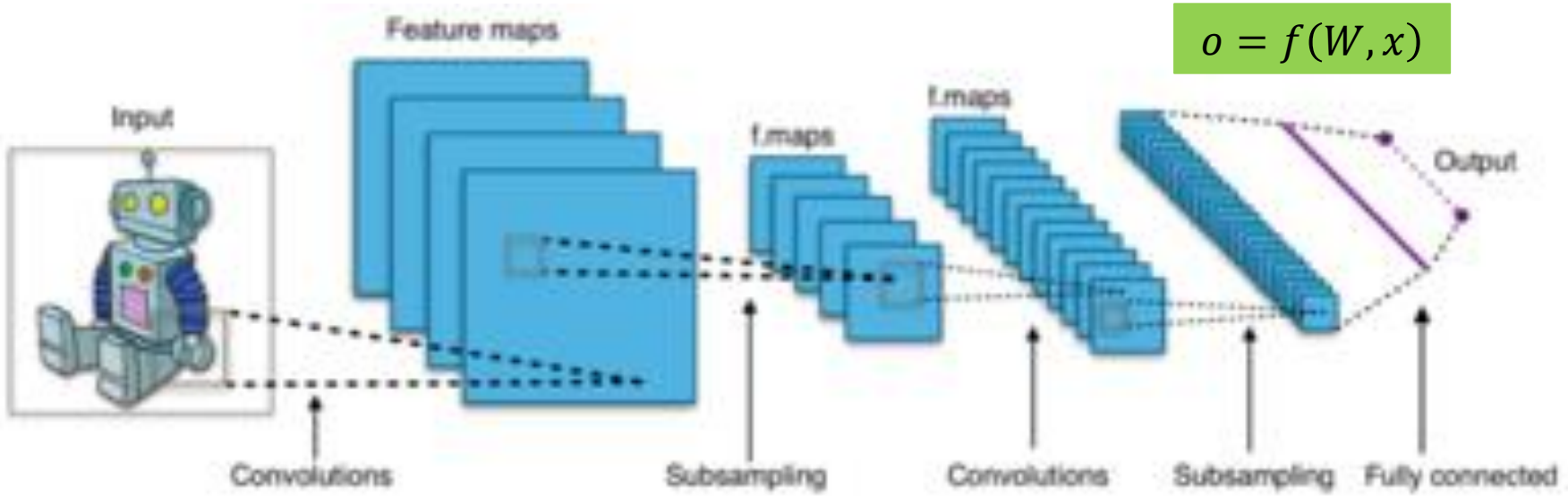
R: Convolutional Neural Networks



$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'$$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

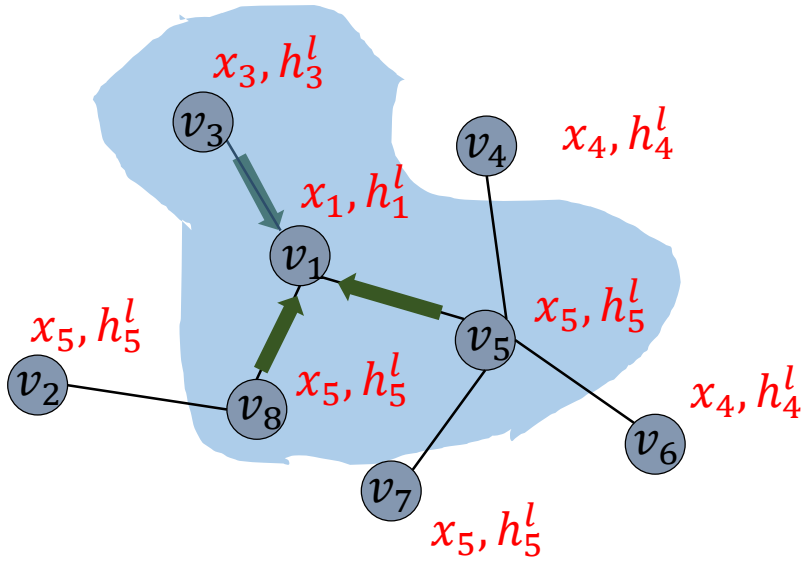
$o = f(W, x)$



$GCN(\mathcal{G})$

Geometric deep learning via graph convolution …

$$h_i^{(l+1)} = \sum_{v_j \in N(v_i)} f\left(x_i, w_{ij}, h_j^{(l)}, x_j\right)$$



GCN: What are issues?

Issues:

- Laplacian smoothing(LS) on input data space
  -
  - →
  - →
- LS under the fixed graph structure
  -
  - →
  - →

GCN: What are issues?

Issues:

- Laplacian smoothing(LS) on input data space
  - Inefficient due to redundancy by high dimension

  → Do LS in feature space

  → Feature embedding in GCN
- LS under the fixed graph structure
  -

  →

  →

GCN: What are issues?

Issues:

- Laplacian smoothing(LS) on input data space
  - Inefficient due to redundancy by high dimension

  → Do LS in feature space

  → Feature embedding in GCN
- LS under the fixed graph structure
  - Highly dependent on graph structure?

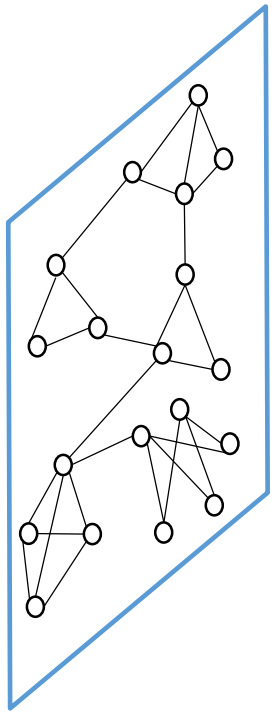
  → Update graph structure

  → How to do? → Attentional aggregation in GCN, Diffusion, etc.

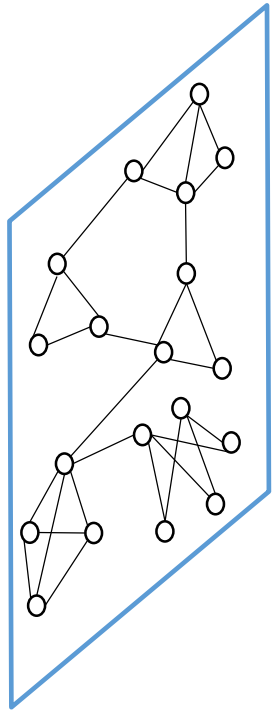
GCN: Two Main Operations

Graph filtering

Node feature representation
Node embedding



$$h_i^{(l+1)} = \sum_{v_j \in N(v_i)} f\left(l_i, w_{ij}, h_j^{(l)}, l_j\right)$$



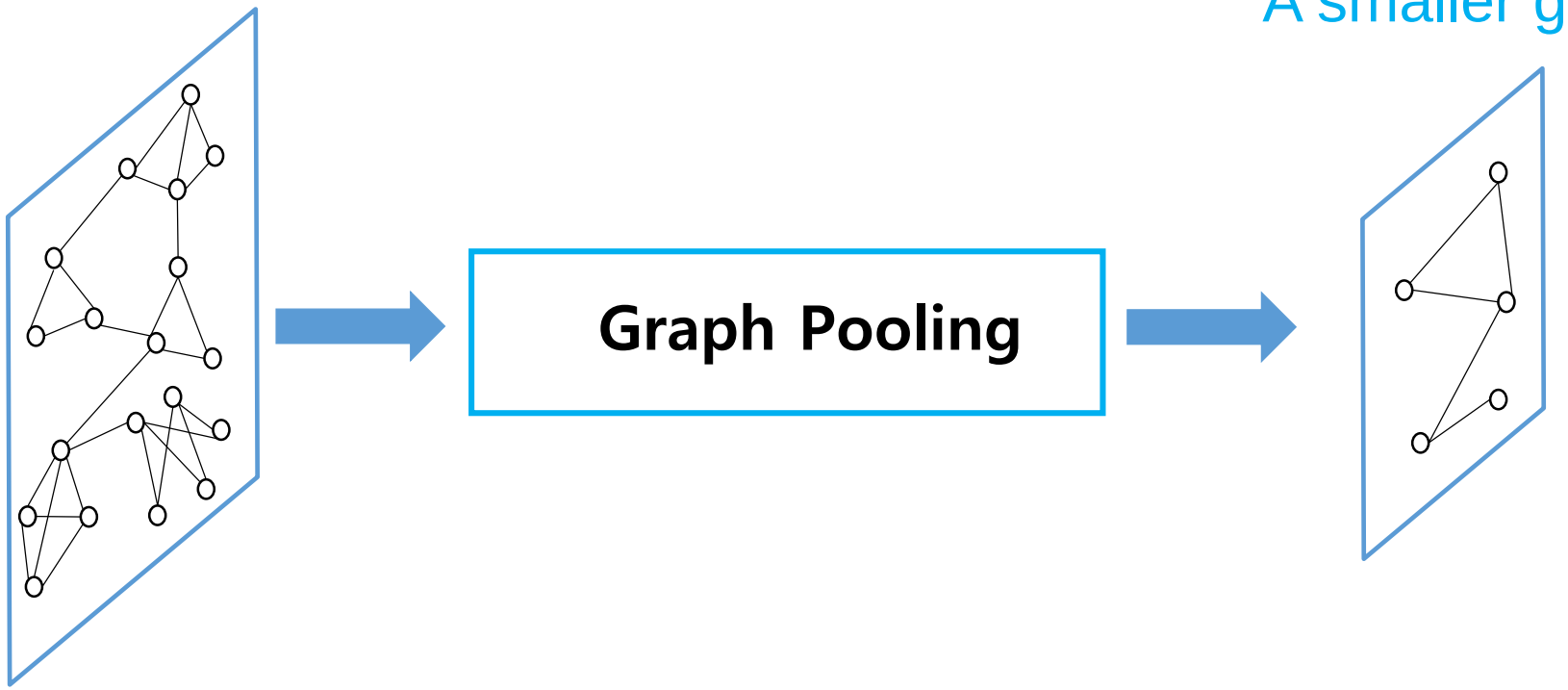
$A \in \{0,1\}^{n \times n}, X \in \mathbb{R}^{n \times d}$

$A \in \{0,1\}^{n \times n}, H_l \in \mathbb{R}^{n \times d_l}$

GCN: Two Main Operations

Graph pooling

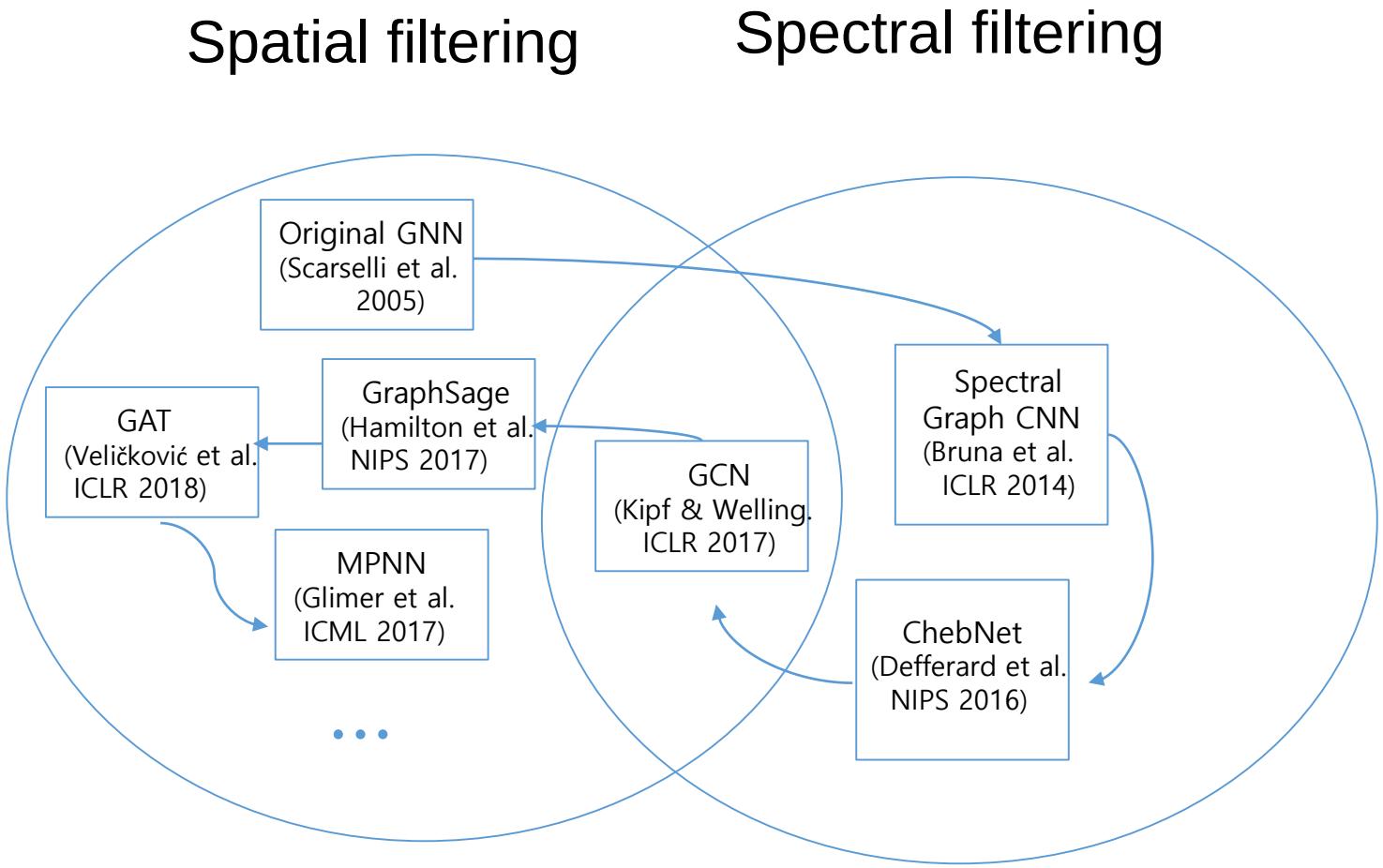
A smaller graph



$\boldsymbol{A} \in \{\boldsymbol{0},\boldsymbol{1}\}^{\boldsymbol{n}\times\boldsymbol{n}}, \boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{d}}$

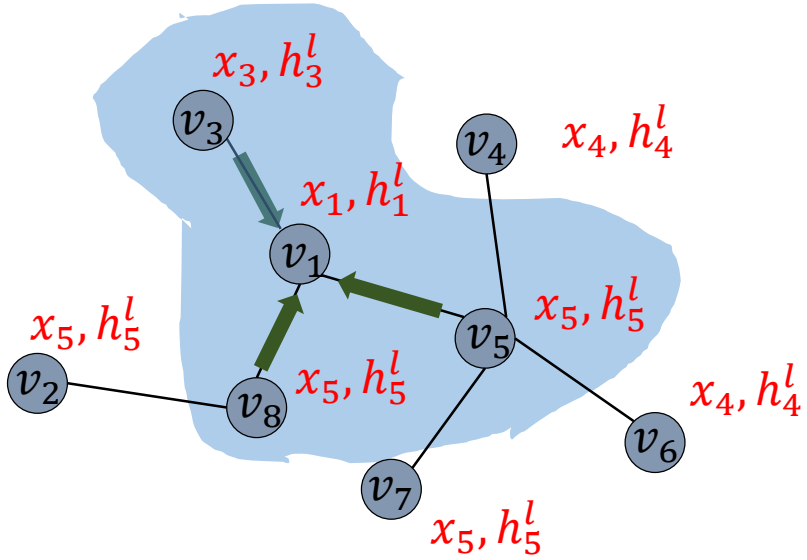
$\boldsymbol{A_p} \in \{\boldsymbol{0},\boldsymbol{1}\}^{\boldsymbol{n_p}\times\boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p}\times\boldsymbol{d_p}}$

GCN: Types of Operations for Graph Filtering



GCN: Graph Filtering

Spatial filtering



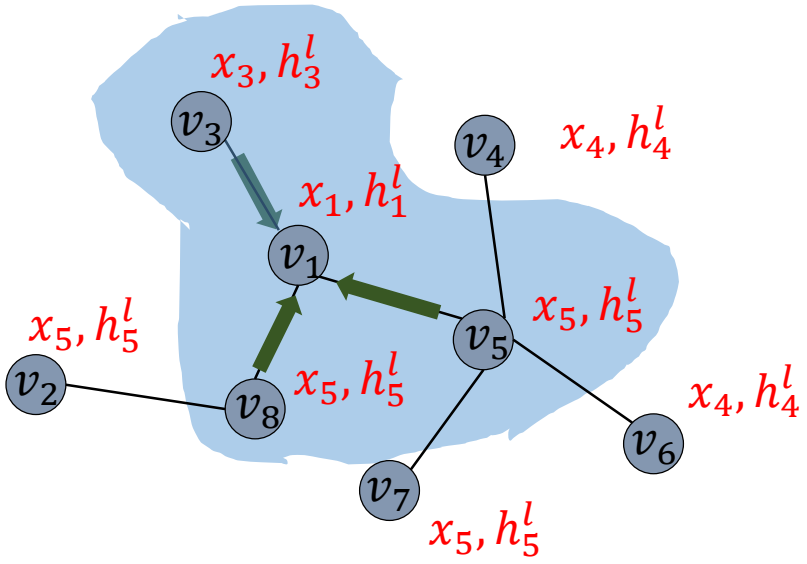
$\boldsymbol{h}_i$: hidden features

$\boldsymbol{x}_i$: input features

Original GNN: Graph neural networks for ranking web pages, IEEE *Web Intelligence* (Scarselli et al. 2005)

GCN: Graph Filtering

Spatial filtering



$\boldsymbol{h}_i$: hidden features

$\boldsymbol{x}_i$: input features

$$\boldsymbol{h}_i^{(l+1)} = \sum_{v_j \in N(v_i)} \boldsymbol{f}\left(\boldsymbol{x}_i, \boldsymbol{h}_j^{(l)}, \boldsymbol{x}_j\right), \qquad \forall\, v_i \in V.$$

$N(v_i)$: neighbors of the node v_i.

$\boldsymbol{f}(\cdot)$: feedforward neural network.

GNN

Summary Questions of the lecture

- Explain the concept of convolution sum and its meaning in the spatial and temporal applications.
- What problems in the conventional Laplacian smoothness applications does the GCN try to solve?
- Explain the two main operations in GCN.