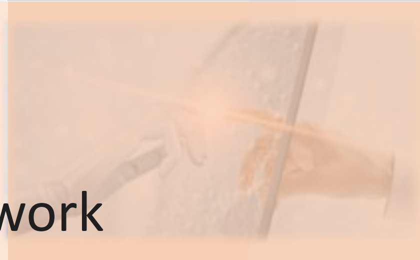


Graphical Neural Network

Korea Institute of Energy Technology

1

Methods

AI

- Convolution methods
- Spectral convolution
- Spatial convolution

Korea Institute of Energy Technology 2022-04-11 2

2

GCN – Math Background

KENTECH Korea Institute of Energy Technology AI

- Graph signal processing
- Normalized graph Laplacian
  - $L = I - D^{-1/2}AD^{-1/2}$
  - L is symmetric and PSD ➔ $L = U\Lambda U^T$ where U is matrix of eigenvalues and Λ is diagonal matrix of eigenvalues
  - $UU^T = I$
- Graph Fourier transform
  - $F(x) = U^T x = \hat{x}$
  - Inverse Fourier transform: $\mathrm{x} = F^{-1}(\hat{x}) = U\hat{x}$
    ➔ $\mathrm{x} = \sum_i \hat{x}_i u_i$
- Graph convolution with filter, g
  - $x * g = F^{-1}(F(x) \odot F(g)) = U(U^T x \odot U^T g)$ where $\odot$ is element-wise product
  - Denote the filter as $g_\theta = diag(U^T g)$
    ➔ $x * g_\theta = U g_\theta U^T x$

GCN – Math Background

KENTECH Korea Institute of Energy Technology AI

- Chebyshev approximation
  - Approximate g_θ by Chebyshev polynomials of U
  - $g_\theta = \sum_{i=1}^{K} \theta_i' T_i(\tilde{\Lambda})$ where $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I$ and $\theta_i' \in R^K$ is a vector of Chebyshev coefficients

$T_0(x) = 0, T_1(x) = x$
$T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$

  - ➔ $x * g_\theta = U\left(\sum_{i=1}^{K} \theta_i T_i(\tilde{\Lambda})\right)U^T x = \sum_{i=1}^{K} \theta_i T_i(\tilde{L})\, x$

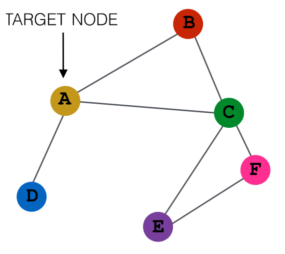
Let $\tilde{L} = \frac{2}{\lambda_{max}}L - I$
$T_i(\tilde{L}) = UT_i(\tilde{\Lambda})U^T$
Proof by induction

- Further simplification
  - $x * g_\theta = \sum_{i=1}^{K} \theta_i T_i(\tilde{L})\, x$
  - Let K=1, $\lambda_{max} = 2$
    ➔ $x * g_\theta = \theta_0 x - \theta_1 D^{-1/2}AD^{-1/2}x$
  - Let $\theta = \theta_0 = -\theta_1$
    ➔ $x * g_\theta = \theta(I + D^{-1/2}AD^{-1/2})\mathrm{x}$

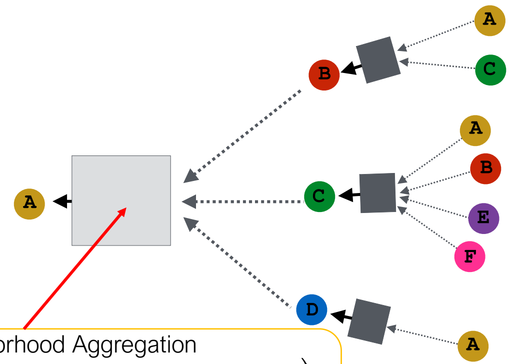
Kipf & Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

  - In case of multi-channel input & output
    ➔ $H = X * g_\Theta = f(\bar{A}X\Theta)$ where $\bar{A} = I + D^{-1/2}AD^{-1/2}$, f(·) an activation function

GCN



INPUT GRAPH

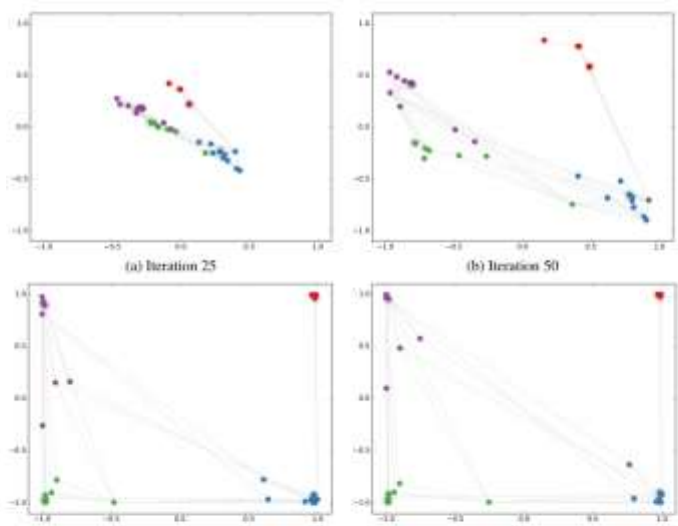


GCN Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

Results

- Node Embedding of GCN



(a) Iteration 25 (b) Iteration 50

(e) Iteration 200 (f) Iteration 300

GCN-Results

KENTECH Korea Institute of Energy Technology AI

- Semi-supervised node classification (Accuracy)

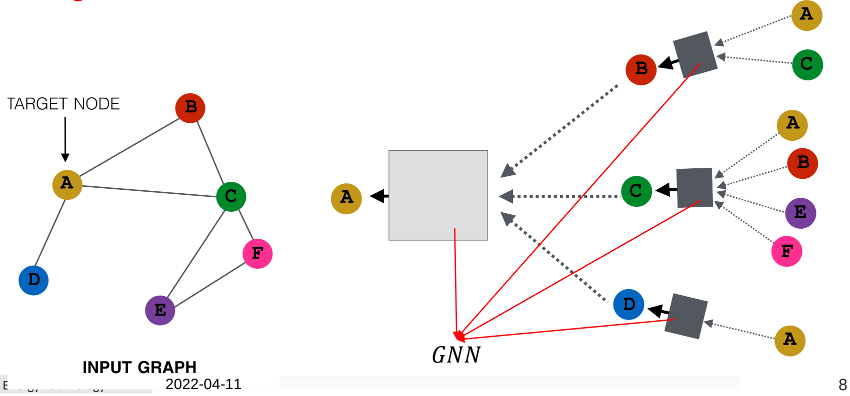
Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

GraphSAGE

KENTECH Korea Institute of Energy Technology AI

Hamilton, Ying, & Leskovec, "Inductive representation learning on large graphs," NIPS, 2017

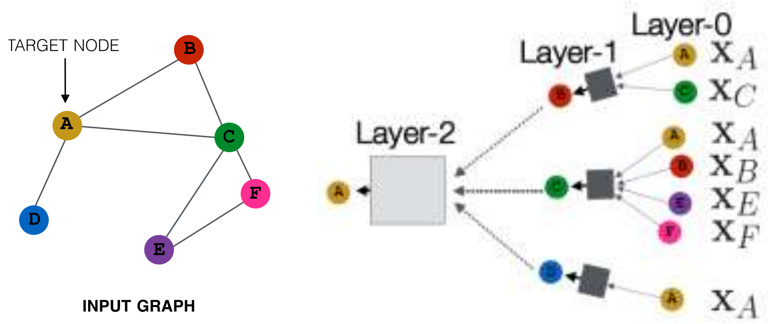
- Compute node embedding based on local network neighborhoods
- Aggregation strategies
  - Max
  - **Average**



Stacking Layers

KENTECH Korea Institute of Energy Technology AI

- Model can be of arbitrary depth (Layers)
  - Nodes have embedding at each layer
  - Layer-0 embedding of node u is its input feature, x_u
  - Layer-K embedding gets information from nodes that are K hops away



Average Aggregation

KENTECH Korea Institute of Energy Technology AI

- Average information(message) from neighbors

$$\mathbf{h}_v^0 = \mathbf{x}_v \quad (Initialize\ node\ embedding)$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, \ldots, K\} \quad (Update)$$

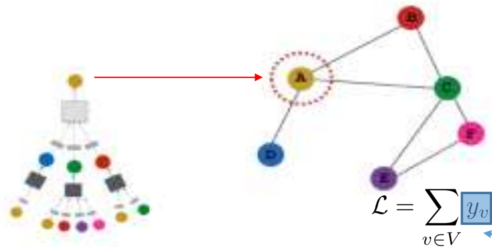
$$\mathbf{z}_v = \mathbf{h}_v^K \quad (After\ k-step\ aggregation)$$

- After K-layers of neighborhood aggregation, we get output embedding for each node (z_v)
- We can feed these embedding into any loss function
- Run stochastic gradient descent to train

Training

KENTECH Korea Institute of Energy Technology AI

- Unsupervised training
  - When we do not have label, only graph structure can be used
  - Goal: make similar nodes have similar embeddings
    - Node proximity in the graph
- Supervised training
  - Ground truth or manually assigned labels
  - Directly train the model to make the input has the output of right label



classification w
eights

$$\mathcal{L} = \sum_{v \in V} \boxed{y_v} \log(\sigma(\boxed{\mathbf{z}_v^\top} \boxed{\theta})) + (1 - \boxed{y_v}) \log(1 - \sigma(\boxed{\mathbf{z}_v^\top} \boxed{\theta}))$$

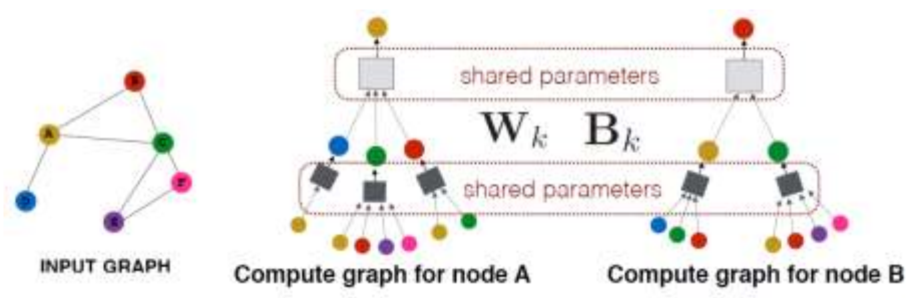
output node
embedding

node class label

Inductiveness

KENTECH Korea Institute of Energy Technology AI

- The same aggregation parameters are shared for all nodes:
  - The number of model parameters is sublinear in $|V|$ and we can **generalize to unseen nodes**



GCN & GraphSAGE

- Graph Convolutional Network (GCN) and GraphSAGE are slightly differ

Basic Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

VS.

GCN Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

same matrix for self and n eighbor embeddings

per-neighbor normalization

Graph Convolutional Network

- Empirically, GCN discovers the configuration that births the best results
  - More parameter sharing
  - Down-weights high degree neighbors

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

use the same transformation matrix fo r self and neighbor embeddings

instead of simple average, normalization varies across neighbors

$$\mathbf{H}^{(k+1)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}} \tilde{\mathbf{A}} \mathbf{D}^{-\frac{1}{2}} \mathbf{H}^{(k)} \mathbf{W}_k\right)$$

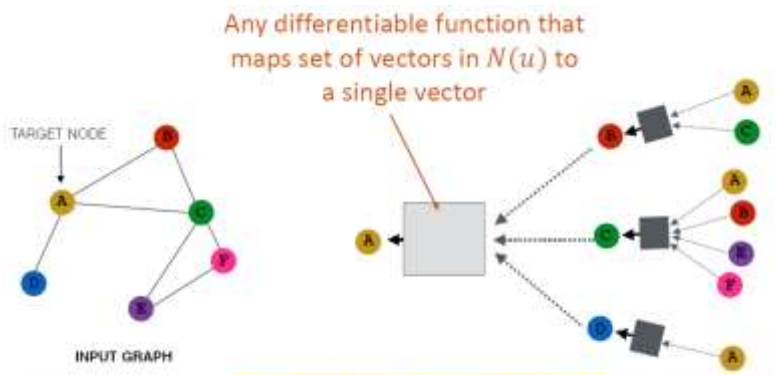
where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

$$\mathbf{D}_{ii} = \sum_j \mathbf{A}_{i,j}$$

2022-04-11

GraphSAGE

- Aggregation function
  - Average



$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{A}_k \cdot \text{AGG}(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}), \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

Neighbor Aggregation

- Variants of Aggregation function
  - Mean
$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$
  - Pool

element-wise mean/max
$$\text{AGG} = \gamma(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$$
  - LSTM
$$\text{AGG} = \text{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

- GAT (Graph Attention Network)
  - Both GCN & GraphSAGE use fixed weights
  - GAT automatically adjusts the importance (weights) of neighbors

Velickovic, et.al., “Graph attention networks,” ICLR, 2017.

GCN – appendix

- GCN
  - $H_{:,j}^{(k)} = \sigma\left(\sum_{i=1}^{f_{k-1}} U\Theta_{i,j}^{(k)} U^T H_{:,i}^{(k-1)}\right)$, $j = 1,2,\ldots,f_k$ with learnable filter, $g_\theta = \Theta_{i,j}^{(k)}$, σ is an activation function such as ReLU
    - k: layer index
    - $H^{(k-1)} \in R^{n \times f_{k-1}}$: input signal with f_{k-1} channels
    - $\Theta_{i,j}^{(k)}$: diagonal matrix
  - ➔ Complexity O(n^3) for eigen-decomposition

17

Appendix: Graph Laplacian

18

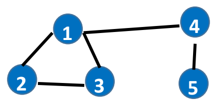
Spectral

- In signal processing
  - Fourier transform
  - Basis: Elementary sine and cosine waves of different frequencies
  - Represent signal as a sum of basis

- In graph
  - Basis: eigenvectors (values) of graph Laplacian matrix (L, defined later)
    - Eigenvector: Orthogonal components of graph
  - Spectral: Eigen-decomposition of L

Spectral Analysis

- Adjacency matrix, A where $A_{ij} = 1$ if edge (i, j) ∈ E



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

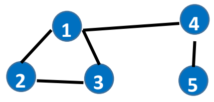
Symmetric
Has n real eigenvalues
n Eigenvectors are real valued and orthogonal

- Eigenpair (eigenvector, eigenvalue) of A: $A\boldsymbol{x}_i = \lambda_i \cdot \boldsymbol{x}_i$
- Spectrum: Eigenvectors of matrix A ordered by the magnitude (strength) of their eigenvalues

  $\Lambda = (\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n),\ \lambda_i \le \lambda_{i+1}$
- Analyze the spectrum

Adjacency and Degree Matrices

KENTECH Korea Institute of Energy Technology AI



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

Diagonal
Non-singular and positive definite (if no isolated node)

- N orthogonal eigenpairs
  - $\lambda_i = k_i,\ \boldsymbol{x}_i = (0, \dots, 1, 0, \dots, 0)$

i-th element

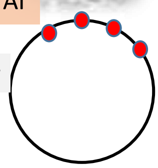
How about D-A?

Example: d-regular graph

KENTECH Korea Institute of Energy Technology AI

- d-regular graph
  - Every node has d edges

2-regular



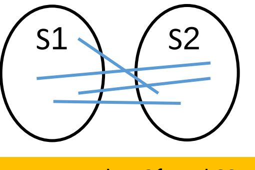
- Connected d-regular graph and its adjacency matrix, A

➔ $\boldsymbol{x}_n$ = $\boldsymbol{1}$ = $(1,1,\dots,1)$ is the largest (principal) eigenvector with eigenvalue λ_n= d

- The second largest (secondary) eigenvector, $\boldsymbol{x}_{n-1}$
- Eigenvectors, $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n-1}$, are orthogonal

➔ $\sum_k \boldsymbol{x}_n[k] \cdot \boldsymbol{x}_{n-1}[k] = 0$

➔ $\sum_k \boldsymbol{x}_{n-1}[k] = 0$



Assume |S1| = |S2|

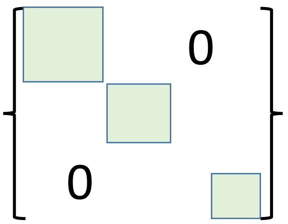
$$\boldsymbol{x}_{n-1}[k] = \begin{cases} 1, & if\ k \in S1 \\ -1, & if\ k \in S2 \end{cases}$$

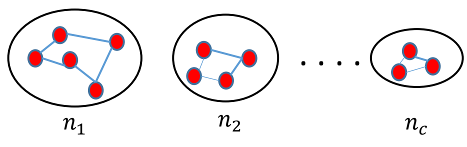
Now suppose that S1 and S2 are sparsely connected
Then eigenvector, $\boldsymbol{x}_{n-1}$, is a good partition
How to find such eigenvector?

Example: Components

KENTECH Korea Institute of Energy Technology AI

- Consider a graph with c components of $n_1, n_2, \dots n_c$ nodes
  - Let each component is a d-regular graph
- Adjacency matrix



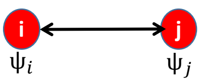


- $\boldsymbol{x}_1 = (\overbrace{1,1,.1}^{n_1},0,.,0)$, $\boldsymbol{x}_2 = (0,.,0,\overbrace{1,.,1}^{n_2},0.,0)$, $\boldsymbol{x}_c = (0,.,0,\overbrace{1,.,1}^{n_c})$ are eigenvectors with eigenvalue $\lambda_{n-c+1} = \lambda_{n-c+2} = \cdots \lambda_n = \text{d}$

Graph Laplacian

KENTECH Korea Institute of Energy Technology AI

- Diffusion (such as pressure, heat, …) between connected nodes



Diffusion constant

$$\frac{d\psi_i(t)}{dt} = C \cdot \sum_j A_{ij} \left(\psi_j(t) - \psi_i(t)\right)$$
$$= C \cdot \left(\sum_j A_{ij}\, \psi_j(t) - \sum_j A_{ij}\, \psi_i(t)\right)$$
$$= C \cdot \sum_j (A_{ij} - \delta_{ij} k_i)\ \psi_j(t)$$

- In a matrix form, $\frac{d\boldsymbol{\psi}(t)}{dt} = C(\mathbf{A} - \mathbf{D})\boldsymbol{\psi}(t)$
- $\frac{d\boldsymbol{\psi}(t)}{dt} + C(\mathbf{D} - \mathbf{A})\boldsymbol{\psi}(t) = 0$
- $\frac{d\boldsymbol{\psi}(t)}{dt} + C\mathbf{L}\boldsymbol{\psi}(t) = 0$, where

$$L_{ij} = \begin{cases} k_i, & if\ i = j \\ -1, & if\ i \neq j\ and\ (i,j) \in E \\ 0, & o.w. \end{cases}$$

Laplacian operator

Diffusion equation: $\frac{d\boldsymbol{\psi}(t)}{dt} + C\nabla^2\boldsymbol{\psi}(t) = 0$

Graph Laplacian

- Properties of L
  - Symmetric
  - Singular

    A1*=0** ➔ Eigenvector ***1=*(1,1,…,1)* with λ=0
  - Positive semidefinite

    ➔ all eigenvalues are nonnegative

    ➔ Eigenvectors are orthogonal
- Write $\boldsymbol{\psi}(t)$ as a linear combination of the eigenvectors

$$\boldsymbol{\psi}(t) = \sum_i a_i(t)\boldsymbol{v}_i$$

$$\frac{d\boldsymbol{\psi}(t)}{dt} + C\mathbf{L}\boldsymbol{\psi}(t) = 0$$

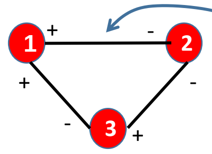
➔ $\sum_i \left(\frac{da_i}{dt} + C\lambda_i a_i\right)\boldsymbol{v}_i = 0$

➔ $\frac{da_i}{dt} + C\lambda_i a_i = 0$

➔ $a_i(t) = a_i(0)e^{-C\lambda_i t}$

Spectrum of Graph Laplacian

- Arbitrarily assign +/- signs to two ends of an edge
- Define a matrix ***B*** (Edge incidence matrix) such that
  - $\boldsymbol{B}_{ij} = \begin{cases} +1, & if\,(i,j) \in E \text{ and } i \text{ is a positive end} \\ -1, & if\,(i,j) \in E \text{ and } i \text{ is a negative end} \\ 0, & o.w. \end{cases}$



$$B = \begin{bmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{bmatrix}$$

- Claim: **L** = $\mathbf{B}^T\mathbf{B}$
  - $(\mathbf{B}^T\mathbf{B})_{ij} = \sum_k \mathbf{B}_{ki} \cdot \mathbf{B}_{kj}$
  - $= \mathbf{L}_{ij}$
- Consider an eigenpair $(\boldsymbol{v}_i, \lambda_i)$ of **L**

➔ $\boldsymbol{v}_i^T \boldsymbol{L}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i^T\boldsymbol{v}_i = \lambda_i$

|||

$\boldsymbol{v}_i^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{v}_i = (\boldsymbol{B}\boldsymbol{v}_i)^T\boldsymbol{B}\boldsymbol{v}_i$

Inner product of a real vector ➔ Nonnegative