

Emerging Directions in Deep Learning

Spectral methods on graphs

Mihai-Sorin Stupariu, 2025-2026

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)
- ▶ Define spaces $L^2(\mathcal{V}), L^2(\mathcal{E})$

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) g(i), \quad f, g : \mathcal{V} \rightarrow \mathbb{R},$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) G(i,j) \quad F, G : \mathcal{E} \rightarrow \mathbb{R}, \text{ skew symmetric}$$

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)
- ▶ Define spaces $L^2(\mathcal{V}), L^2(\mathcal{E})$

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) g(i), \quad f, g : \mathcal{V} \rightarrow \mathbb{R},$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) G(i,j) \quad F, G : \mathcal{E} \rightarrow \mathbb{R}, \text{ skew symmetric}$$

- ▶ The **graph gradient** $\nabla : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{E})$ is defined by
$$(\nabla f)(i,j) = f(i) - f(j)$$

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)
- ▶ Define spaces $L^2(\mathcal{V}), L^2(\mathcal{E})$

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) g(i), \quad f, g : \mathcal{V} \rightarrow \mathbb{R},$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) G(i,j) \quad F, G : \mathcal{E} \rightarrow \mathbb{R}, \text{ skew symmetric}$$

- ▶ The **graph gradient** $\nabla : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{E})$ is defined by

$$(\nabla f)(i,j) = f(i) - f(j)$$

- ▶ The **graph divergence** $\text{div} : L^2(\mathcal{E}) \rightarrow L^2(\mathcal{V})$ is defined as

$$(\text{div} F)(i) = \frac{1}{a_i} \sum_{j, (i,j) \in \mathcal{E}} w_{ij} F(i,j)$$

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)
- ▶ Define spaces $L^2(\mathcal{V}), L^2(\mathcal{E})$

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) g(i), \quad f, g : \mathcal{V} \rightarrow \mathbb{R},$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) G(i,j) \quad F, G : \mathcal{E} \rightarrow \mathbb{R}, \text{ skew symmetric}$$

- ▶ The **graph gradient** $\nabla : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{E})$ is defined by

$$(\nabla f)(i,j) = f(i) - f(j)$$

- ▶ The **graph divergence** $\text{div} : L^2(\mathcal{E}) \rightarrow L^2(\mathcal{V})$ is defined as

$$(\text{div} F)(i) = \frac{1}{a_i} \sum_{j, (i,j) \in \mathcal{E}} w_{ij} F(i,j)$$

- ▶ The two operators are adjoint w.r.t. the inner products (see next slide)

Discrete differential operators on triangle meshes / graphs

- ▶ Weighted graph: graph $(\mathcal{V}, \mathcal{E})$ with weights $((a_i)_{i \in \mathcal{V}}, (w_{ij})_{(i,j) \in \mathcal{E}})$, $a_i > 0$, $w_{ij} \geq 0, \forall i, j$ (examples: $a_i = 1, \forall i$, $w_{ij} = e^{-\|v_i - v_j\|^2 / 2\sigma^2}$ - for embedded graphs)
- ▶ Define spaces $L^2(\mathcal{V}), L^2(\mathcal{E})$

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) g(i), \quad f, g : \mathcal{V} \rightarrow \mathbb{R},$$

$$\langle F, G \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) G(i,j) \quad F, G : \mathcal{E} \rightarrow \mathbb{R}, \text{ skew symmetric}$$

- ▶ The **graph gradient** $\nabla : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{E})$ is defined by

$$(\nabla f)(i,j) = f(i) - f(j)$$

- ▶ The **graph divergence** $\text{div} : L^2(\mathcal{E}) \rightarrow L^2(\mathcal{V})$ is defined as

$$(\text{div} F)(i) = \frac{1}{a_i} \sum_{j, (i,j) \in \mathcal{E}} w_{ij} F(i,j)$$

- ▶ The two operators are adjoint w.r.t. the inner products (see next slide)
- ▶ The **graph Laplacian** $\Delta : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{V})$ is defined as

$$\Delta = \text{div} \nabla; \quad \text{it holds } (\Delta f)(i) = \frac{1}{a_i} \sum_{(i,j) \in \mathcal{E}} w_{ij} (f(i) - f(j)). \quad (1)$$

The two operators are adjoint w.r.t. the inner products

Adjoint means:

$$\langle \nabla f, F \rangle_{L^2(\mathcal{E})} = \langle f, \operatorname{div} F \rangle_{L^2(\mathcal{V})}.$$

For the LHS term one has:

$$\langle \nabla f, F \rangle_{L^2(\mathcal{E})} = \sum_{(i,j) \in \mathcal{E}} w_{ij} \nabla f(i,j) F(i,j) = \sum_{(i,j) \in \mathcal{E}} w_{ij} (f(i) - f(j)) F(i,j).$$

For the RHS term one has:

$$\langle f, \operatorname{div} F \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f(i) (\operatorname{div} F)(i) = \sum_{i \in \mathcal{V}} a_i f(i) \frac{1}{a_i} \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j).$$

Using now that F is skew symmetric ($F(i,j) = -F(j,i)$, $\forall i,j$) one gets

$$\sum_{i \in \mathcal{V}} f(i) \sum_{(i,j) \in \mathcal{E}} w_{ij} F(i,j) = \sum_{(i,j) \in \mathcal{E}} w_{ij} (f(i) - f(j)) F(i,j),$$

which yields the desired result.

The Laplacian is a linear operator

The equation (1) indicates that the Laplacian can be interpreted as a linear operator,

$$(f(1), \dots, f(n))^T \mapsto A^{-1} \cdot M \cdot (f(1), \dots, f(n))^T, \quad (2)$$

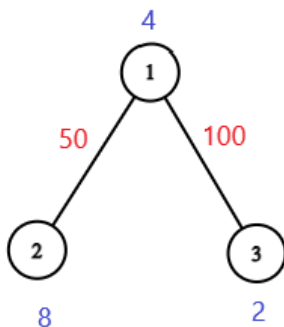
where $A = \text{diag}(a_1, \dots, a_n)$ is given by the weights of the vertices and

$$M = (\xi_{ij})_{i,j}, \quad \xi_{ii} = \sum_{\sum (i,j) \in \mathcal{E}} w_{ij}, \quad \xi_{ij} = -w_{ij}$$

is given by the edge weights.

Example

We consider the weighted graph in the figure, having three nodes (with weights 4, 8, and 2, respectively) and with two edges (having weights 50 and 100, respectively),



The node 1 has the weight 4, the node 2 has the weight 8 and the node 3 has the weight 2. The edges (1,2) and (2,1) have the weight 50. The edges (1,3) and (3,1) have the weight 100.

Example

We take now a function in the space $L^2(\mathcal{V})$ and we apply the steps previously described for determining its Laplacian. Let

$$f : \mathcal{V} \rightarrow \mathbb{R}, \quad f(1) = 9, \quad f(2) = -5, \quad f(3) = 7.$$

One can interpret f as vector:

$$f \equiv (9, -5, 7)^T.$$

Example

The **graph gradient** $\nabla : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{E})$ is defined by

$$(\nabla f)(i,j) = f(i) - f(j),$$

for any pair (i,j) corresponding to an edge. For the other pairs (i,j) we set the value to be zero. In the case of the function f previously introduced, we have

$$(\nabla f)(1,2) = f(1) - f(2) = 14, \quad (\nabla f)(2,1) = -14,$$

$$(\nabla f)(1,3) = 2, \quad (\nabla f)(3,1) = -2.$$

The function $\nabla f : \mathcal{E} \rightarrow \mathbb{R}$ can be regarded as a matrix

$$\nabla f \equiv \begin{pmatrix} 0 & 14 & 2 \\ -14 & 0 & 0 \\ -2 & 0 & 0 \end{pmatrix}$$

Example

The **graph Laplacian** $\Delta : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{V})$ is defined as

$$\Delta = \operatorname{div} \nabla; \quad \text{it holds } (\Delta f)(i) = \frac{1}{a_i} \sum_{(i,j) \in \mathcal{E}} w_{ij}(f(i) - f(j)).$$

Thus, we get

$$\begin{aligned} (\operatorname{div} \nabla f)(1) &= \frac{1}{a_1} \sum_{(1,j) \in \mathcal{E}} w_{1j} F(1,j) = \frac{1}{a_1} \sum_{(1,j) \in \mathcal{E}} w_{1j}(f(1) - f(j)) = \\ &= \frac{1}{4}(50 \cdot 14 + 100 \cdot 2) = 225. \end{aligned}$$

Analogously,

$$(\operatorname{div} \nabla f)(2) = -87.5, \quad (\operatorname{div} \nabla f)(3) = -100.$$

Finally, the Laplacian can be interpreted as a map that maps the vector $(9, -5, 7)^T$ to the vector $(225, -87.5, -100)^T$.

Example

As indicated in equation (2), via natural identifications, the Laplacian corresponds to a $n \times n$ matrix, where n is the number of graph nodes. In the example, one has

$$A^{-1} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad M = \begin{pmatrix} 150 & -50 & 100 \\ -50 & 50 & 0 \\ -100 & 0 & 100 \end{pmatrix}.$$

The Laplacian corresponds to $A^{-1} \cdot M$, that is

$$\Delta \equiv \begin{pmatrix} \frac{75}{2} & -\frac{25}{2} & -25 \\ -\frac{25}{4} & \frac{25}{4} & 0 \\ -50 & 0 & 50 \end{pmatrix}.$$

More notation / definitions / conventions for the graph Laplacian. Basis of eigenvectors

- Denote by $W = (w_{ij})$ the matrix of edge weights, by $A = \text{diag}(a_1, \dots, a_n)$ the diagonal matrix of vertex weights and by $D = \text{diag}(\sum_{j \neq i} w_{ij})$ the degree matrix. The graph Laplacian operator applied to a function $f = (f(1), \dots, f(n))^T$ can be written as

$$\Delta f = A^{-1}(D - W)f$$

More notation / definitions / conventions for the graph Laplacian. Basis of eigenvectors

- ▶ Denote by $W = (w_{ij})$ the matrix of edge weights, by $A = \text{diag}(a_1, \dots, a_n)$ the diagonal matrix of vertex weights and by $D = \text{diag}(\sum_{j \neq i} w_{ij})$ the degree matrix. The graph Laplacian operator applied to a function $f = (f(1), \dots, f(n))^T$ can be written as

$$\Delta f = A^{-1}(D - W)f$$

- ▶ Suppose that the graph has n nodes. Then the Laplacian can be interpreted as a $n \times n$ matrix, which is symmetric and semi-positive definite (one has to take into account the definition of the inner product!).

More notation / definitions / conventions for the graph Laplacian. Basis of eigenvectors

- ▶ Denote by $W = (w_{ij})$ the matrix of edge weights, by $A = \text{diag}(a_1, \dots, a_n)$ the diagonal matrix of vertex weights and by $D = \text{diag}(\sum_{j \neq i} w_{ij})$ the degree matrix. The graph Laplacian operator applied to a function $f = (f(1), \dots, f(n))^T$ can be written as

$$\Delta f = A^{-1}(D - W)f$$

- ▶ Suppose that the graph has n nodes. Then the Laplacian can be interpreted as a $n \times n$ matrix, which is symmetric and semi-positive definite (one has to take into account the definition of the inner product!).
- ▶ Consider a basis of eigenvectors

$$\Phi = (\phi_1, \phi_2, \dots, \phi_n).$$

One can write

$$\Delta \Phi = \Phi \Lambda, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n),$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of Δ .

More notation / definitions / conventions for the graph Laplacian. Basis of eigenvectors

- ▶ Denote by $W = (w_{ij})$ the matrix of edge weights, by $A = \text{diag}(a_1, \dots, a_n)$ the diagonal matrix of vertex weights and by $D = \text{diag}(\sum_{j \neq i} w_{ij})$ the degree matrix. The graph Laplacian operator applied to a function $f = (f(1), \dots, f(n))^T$ can be written as

$$\Delta f = A^{-1}(D - W)f$$

- ▶ Suppose that the graph has n nodes. Then the Laplacian can be interpreted as a $n \times n$ matrix, which is symmetric and semi-positive definite (one has to take into account the definition of the inner product!).
- ▶ Consider a basis of eigenvectors

$$\Phi = (\phi_1, \phi_2, \dots, \phi_n).$$

One can write

$$\Delta \Phi = \Phi \Lambda, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n),$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of Δ .

- ▶ **Choices** for A : $A = \mathbb{I}$ (the unnormalized graph Laplacian); $A = D$ (the random walk Laplacian).

Definition of a convolution-type operation with the Laplacian

- ▶ **In the Convolution Theorem** all sums become finite, since the graph Laplacian has n eigenvectors. The **eigenvalues** of the graph Laplacian are given by solving the equation $\Delta f = f$ and accordingly the **eigenvectors**

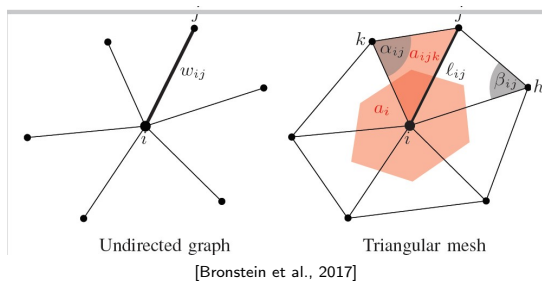
Definition of a convolution-type operation with the Laplacian

- ▶ **In the Convolution Theorem** all sums become finite, since the graph Laplacian has n eigenvectors. The **eigenvalues** of the graph Laplacian are given by solving the equation $\Delta f = \lambda f$ and accordingly the **eigenvectors**
- ▶ In matrix form, the generalized convolution $f \star g$ can be expressed as

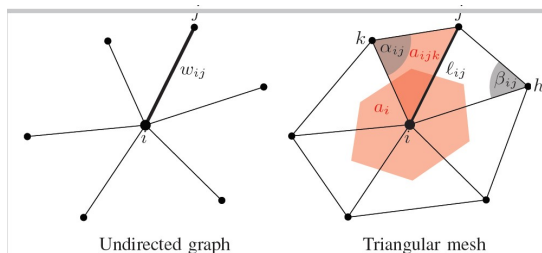
$$\Phi \text{diag}(\hat{g}) \Phi^T f,$$

where $\hat{g}$ is the spectral representation of the filter g and $\Phi = (\phi(1), \dots, \phi(n))$ denotes the Laplacian eigenvectors.

For triangle meshes



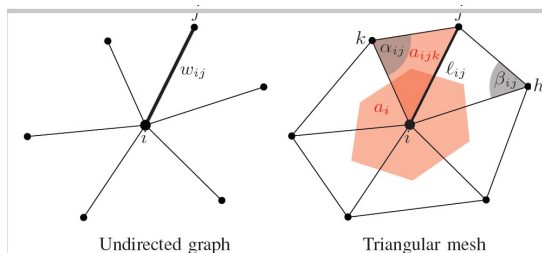
For triangle meshes



[Bronstein et al., 2017]

- ▶ Triangle mesh: triple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ verifying natural incidence requirements

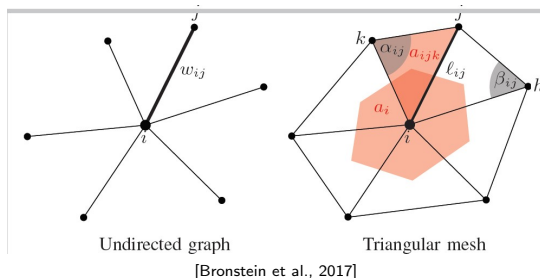
For triangle meshes



[Bronstein et al., 2017]

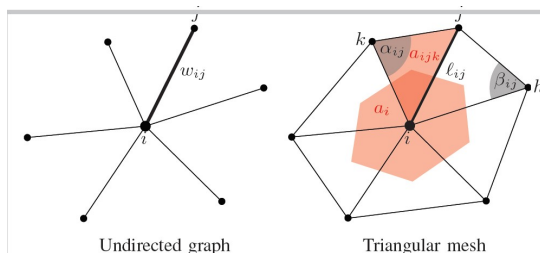
- ▶ Triangle mesh: triple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ verifying natural incidence requirements
- ▶ Riemannian metric: assign a length ℓ_{ij} to each edge, such that the triangle inequality is verified for each face.

For triangle meshes



- ▶ Triangle mesh: triple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ verifying natural incidence requirements
- ▶ Riemannian metric: assign a length ℓ_{ij} to each edge, such that the triangle inequality is verified for each face.
- ▶ Define the area a_{mnp} of a face $(m, n, p) \in \mathcal{F}$ with the Heron formula, then the weights w_{ij} and the weights a_i , e.g. $a_i = \frac{1}{3} \sum_{j,k:(i,j,k) \in \mathcal{F}} a_{ijk}$ ("local area element")

For triangle meshes



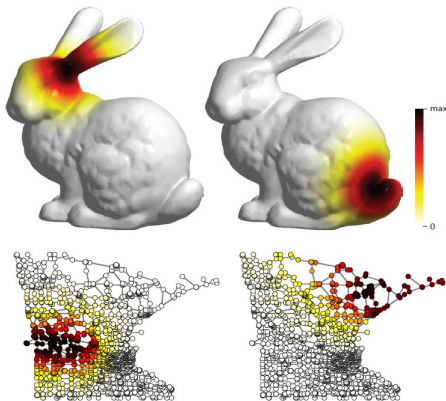
[Bronstein et al., 2017]

- ▶ Triangle mesh: triple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ verifying natural incidence requirements
- ▶ Riemannian metric: assign a length l_{ij} to each edge, such that the triangle inequality is verified for each face.
- ▶ Define the area a_{mnp} of a face $(m, n, p) \in \mathcal{F}$ with the Heron formula, then the weights w_{ij} and the weights a_i , e.g. $a_i = \frac{1}{3} \sum_{j,k:(i,j,k) \in \mathcal{F}} a_{ijk}$ ("local area element")
- ▶ In the case of an embedding, take $l_{ij} = \|v_i - v_j\|$, the weights become the **cotangent weights**

$$w_{ij} = \frac{1}{2}(\text{ctg}(\alpha_{ij}) + \text{ctg}(\beta_{ij}))$$

In the non-Euclidean framework...

... shift-invariance assumptions may fall; there is a lack of shift-invariance



[FIGS4] Examples of heat kernels on non-Euclidean domains (manifold, top; and graph, bottom). Observe how moving the heat kernel to a different location changes its shape, which is an indication of the lack of shift-invariance.

[Bronstein et al., 2017]

Spectral methods

- ▶ Assumption: convolutions are linear operators that commute with the Laplacian operator.

Spectral methods

- ▶ Assumption: convolutions are linear operators that commute with the Laplacian operator.
- ▶ On a weighted graph: one could generalize a convolutional architecture by restricting to linear operators that commute with the graph Laplacian.

Spectral methods

- ▶ Assumption: convolutions are linear operators that commute with the Laplacian operator.
- ▶ On a weighted graph: one could generalize a convolutional architecture by restricting to linear operators that commute with the graph Laplacian.
- ▶ This property requires operating on the spectrum of the graph weights, given by the eigenvectors of the graph Laplacian.

Spectral CNN -SCNN

- ▶ Bruna et al. 2014

Spectral CNN -SCNN

► Bruna et al. 2014

The convolution:

$$g_l = \xi \left(\sum_{l'=1}^q \Phi_k \Gamma_{l,l'} \Phi_k^T f_{l'} \right), \text{ where} \quad (3)$$

$(f_1, \dots, f_p) \in \mathcal{M}(n, p)$ is the p -dimensional input,
 $(g_1, \dots, g_q) \in \mathcal{M}(n, q)$ is the n -dimensional output ($n = |\mathcal{V}|$),
 $\Gamma_{l,l'}$ is a $n \times n$ ($k \times k$) diagonal matrix of spectral multipliers
representing a filter in the frequency domain, and
 ξ is a nonlinearity applied on the vertex-wise function values.

Spectral CNN -SCNN

► Bruna et al. 2014

The convolution:

$$g_l = \xi \left(\sum_{l'=1}^q \Phi_k \Gamma_{l,l'} \Phi_k^T f_{l'} \right), \text{ where} \quad (3)$$

$(f_1, \dots, f_p) \in \mathcal{M}(n, p)$ is the p -dimensional input,
 $(g_1, \dots, g_q) \in \mathcal{M}(n, q)$ is the n -dimensional output ($n = |\mathcal{V}|$),
 $\Gamma_{l,l'}$ is a $n \times n$ ($k \times k$) diagonal matrix of spectral multipliers representing a filter in the frequency domain, and
 ξ is a nonlinearity applied on the vertex-wise function values.

- One can use only the first k eigenvectors in (3) (by appropriately setting the diagonal matrix). Reason: only the first Laplacian eigenvectors describing the smooth structure of the graph are useful in practice.

Spectral CNN -SCNN

► Bruna et al. 2014

The convolution:

$$g_l = \xi \left(\sum_{l'=1}^q \Phi_k \Gamma_{l,l'} \Phi_k^T f_{l'} \right), \text{ where} \quad (3)$$

$(f_1, \dots, f_p) \in \mathcal{M}(n, p)$ is the p -dimensional input,
 $(g_1, \dots, g_q) \in \mathcal{M}(n, q)$ is the n -dimensional output ($n = |\mathcal{V}|$),
 $\Gamma_{l,l'}$ is a $n \times n$ ($k \times k$) diagonal matrix of spectral multipliers
representing a filter in the frequency domain, and
 ξ is a nonlinearity applied on the vertex-wise function values.

- One can use only the first k eigenvectors in (3) (by appropriately setting the diagonal matrix). Reason: only the first Laplacian eigenvectors describing the smooth structure of the graph are useful in practice.
- Pooling: replaced by graph coarsening (only a fraction of the graph vertices is retained).

Spectral CNN -SCNN. Advantages

- ▶ If the graph has an underlying group invariance, such a construction can discover it. In particular, standard CNNs can be redefined from the spectral domain.

Spectral CNN -SCNN. Advantages

- ▶ If the graph has an underlying group invariance, such a construction can discover it. In particular, standard CNNs can be redefined from the spectral domain.
- ▶ Certain flexibility for getting a better learning complexity.

Spectral CNN -SCNN. Limitations

- ▶ The spectral filter coefficients are basis dependent (depend on Φ).

Spectral CNN -SCNN. Limitations

- ▶ The spectral filter coefficients are basis dependent (depend on Φ).
- ▶ It is possible to get a joint diagonalization over suitable domains: see e.g., [Eynard et al., 2015](#). Such an approach works for social network analysis, but is hardly achievable in computer graphics tasks.

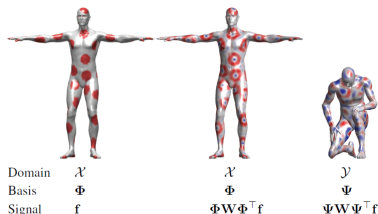


Fig. 2. A toy example illustrating the difficulty of generalizing spectral filtering across non-Euclidean domains. Left: a function defined on a manifold (function values are represented by color); middle: result of the application of an edge-detection filter in the frequency domain; right: the same filter applied on the same function but on a different (nearly-isometric) domain produces a completely different result. The reason for this behavior is that the Fourier basis is domain-dependent, and the filter coefficients learnt on one domain cannot be applied to another one in a straightforward manner.

[Bronstein et al., 2017]

Spectral CNN -SCNN. Limitations

- ▶ The spectral filter coefficients are basis dependent (depend on Φ).
- ▶ It is possible to get a joint diagonalization over suitable domains: see e.g., [Eynard et al., 2015](#). Such an approach works for social network analysis, but is hardly achievable in computer graphics tasks.

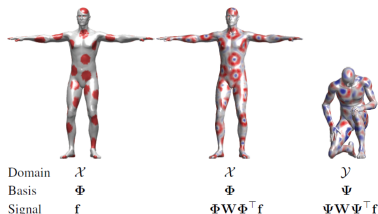


Fig. 2. A toy example illustrating the difficulty of generalizing spectral filtering across non-Euclidean domains. Left: a function defined on a manifold (function values are represented by color); middle: result of the application of an edge-detection filter in the frequency domain; right: the same filter applied on the same function but on a different (nearly-isometric) domain produces a completely different result. The reason for this behavior is that the Fourier basis is domain-dependent, and the filter coefficients learnt on one domain cannot be applied to another one in a straightforward manner.

[Bronstein et al., 2017]

- ▶ High computational complexity ($O(n^2)$)

SCNN with modified spectral multipliers

- ▶ Henaff et al., 2015

SCNN with modified spectral multipliers

- ▶ [Henaff et al., 2015](#)
- ▶ Aim: adapt the learning complexity to reduce the number of free parameters of the model. “Localization” (in standard CNNs: convolutional kernels with small spatial support).

SCNN with modified spectral multipliers

- ▶ [Henaff et al., 2015](#)
- ▶ Aim: adapt the learning complexity to reduce the number of free parameters of the model. “Localization” (in standard CNNs: convolutional kernels with small spatial support).
- ▶ Key ingredient: modify the the spectral multipliers $\Gamma_{l,l'}$

Spectral-based spectrum-free methods

- ▶ GraphCNN / ChebNet [Defferrard et al., 2016](#) Use the Chebyshev polynomial generated by a recurrence relation for getting an action of the filter that does not depend on the basis of Laplace operator and does not require an explicit computation of the Laplacian eigenvectors.

Spectral-based spectrum-free methods

- ▶ GraphCNN / ChebNet [Defferrard et al., 2016](#) Use the Chebyshev polynomial generated by a recurrence relation for getting an action of the filter that does not depend on the basis of Laplace operator and does not require an explicit computation of the Laplacian eigenvectors.
- ▶ Graph Convolutional Network (GCN) [Kipf and Welling, 2017](#) Further assumptions / simplifications lead to another action of the filter that does not depend on the basis of Laplace operator and does not require an explicit computation of the Laplacian eigenvectors. See also <https://tkipf.github.io/graph-convolutional-networks/>.

General methods on graphs

- Generalize the notion of applying the filtering operations directly on the graph via the graph weights. Learn generic filters as linear combinations of localized filters. They were ($\sim$ firstly) considered in the literature [Gori et al., 2005](#), [Scarselli et al., 2009](#).

General methods on graphs

- ▶ Generalize the notion of applying the filtering operations directly on the graph via the graph weights. Learn generic filters as linear combinations of localized filters. They were ($\sim$ firstly) considered in the literature [Gori et al., 2005](#), [Scarselli et al., 2009](#).
- ▶ More about signal processing on graphs [Shuman et al., 2013](#)

General methods on graphs

- ▶ Generalize the notion of applying the filtering operations directly on the graph via the graph weights. Learn generic filters as linear combinations of localized filters. They were ($\sim$ firstly) considered in the literature [Gori et al., 2005](#), [Scarselli et al., 2009](#).
- ▶ More about signal processing on graphs [Shuman et al., 2013](#)
- ▶ Message Passing Neural Network framework (Neighborhood Aggregation) [Gilmer et al., 2017](#)

General methods on graphs

- ▶ Generalize the notion of applying the filtering operations directly on the graph via the graph weights. Learn generic filters as linear combinations of localized filters. They were ($\sim$ firstly) considered in the literature [Gori et al., 2005](#), [Scarselli et al., 2009](#).
- ▶ More about signal processing on graphs [Shuman et al., 2013](#)
- ▶ Message Passing Neural Network framework (Neighborhood Aggregation) [Gilmer et al., 2017](#)
- ▶ Implementation in [PyTorch geometric Fey and Lensen, 2019](#) (including the [Laplacian](#)).