AI4PDE

PDEs with Laplacian on Point Cloud and Equivariance

Yunfeng Liao

30 August,2024

Presentation Overview

Literatures

1. Riemannian Manifold Learning. Tong Lin and Hongbin Zha. IEEE Trans., 2008.
2. Diffusion maps. Ronald R. Coifman, Stéphane Lafon. Applied and Computational Harmonic Analysis. 2006.
3. SOLVING PDES ON UNKNOWN MANIFOLDS WITH ML. Senwei Liang et al. Applied and Computational Harmonic Analysis, 2024.
4. SOLVING FORWARD AND INVERSE PDE PROBLEMS ON UNKNOWN MANIFOLDS VIA PHYSICS-INFORMED NEURAL OPERATORS .
5. Point Integral Method for Solving Poisson-type Equations on Manifolds from Point Clouds with Convergence Guarantees. Zhen Li et al. 2014.
6. Convergence of the Point Integral method for Poisson equation on point cloud. Jan Sun et al. 2014.

I: PDEs with Laplacian on Point Cloud:DM and PIM

How to represent a feature manifold?[1]

Major Manifold Learning Algorithms

Authors	Year	Algorithm	Property	Description and Comments
Tenenbaum et al. [13]	2000	ISOMAP	Isometric mapping	Computes the geodesic distances, and then uses MDS. Computationally expensive.
Roweis et al. [16]	2000	LLE	Preserving linear reconstruction weights	Computes the reconstruction weights for each point, and then minimizes the embedding cost by solving an eigenvalue problem.
Silva et al. [14]	2003	C-ISOMAP and L-ISOMAP	Conformal ISOMAP and landmark ISOMAP	C-ISOMAP preserves angles. L-ISOMAP efficiently approximates the original ISOMAP by choosing a small number of landmark points.
Belkin et al. [18]	2003	Laplacian eigenmaps	Locality preserving	Minimizing the squared gradient of an embedding map is equal to finding eigenfunctions of the Laplace-Beltrami operator.
Donoho et al. [20]	2003	HLLE or Hessian eigenmaps	Locally isometric to an open, connected subset	A modification of Laplacian eigenmaps by substituting the Hessian for the Laplacian. Computationally demanding.
Brand [22]	2003	Manifold charting	Preserving local variance and neighborhood	Decomposes the input data into locally linear patches, and then merges these patches into a single low-dimensional coordinate system by using affine transformations.
Zhang et al. [23]	2004	LTSA	Minimizing the global reconstruction error	First constructs the tangent space at each data point, and then aligns these tangent spaces with a global coordinate system.
Weinberger et al. [21]	2004	SDE	Local isometry	Maximizing the variance of the outputs, subject to the constraints of zero mean and local isometry. Computationally expensive by using semidefinite programming.
He et al. [19]	2005	Laplacianfaces	Linear version of Laplacian eigenmaps	The minimization problem reduces to a generalized eigenvalues problem.
Coifman et al. [24][25]	2005	Diffusion maps √	Preserving diffusion distances	Given a Markov random walk on the data, the diffusion map is constructed based on the first few eigenvalues and eigenvectors of the transition matrix *P*.
Sha et al. [26]	2005	Conformal eigenmaps	Angle-preserving embedding	Maximizing the similarity of triangles in each neighborhood. More faithfully preserving the global shape and the aspect ratio. Semidefinite programming is used to for optimization.
Law et al. [15]	2006	Incremental ISOMAP	Data are collected sequentially.	Efficiently updates all-pair shortest path distances, and solves an incremental eigenvalue problem.

What is a diffsion map(DM)?[2]

Random Walk on Graphs

Let (X,Ω,μ) be a measure space. $k: X\times X \to \mathbb{R}$ is a kernel s.t. $k(x,y)=k(y,x), k(x,y)\geq 0, \forall x,y\in X$. If the data X is a graph, then k is the edge weight on X. In this sense, define its degree $d(x) := \int k(x,y)d\mu(y)$ and the **random walk** from x to y is defined as:

$$\mathbb{P}(X^{t+1}=y|X^t=x)=\frac{k(x,y)}{d(x)}, y\in N(x) \tag{1}$$

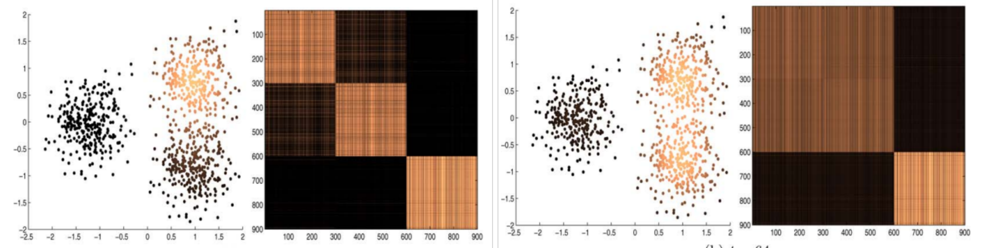
A random walk is a **Markov chain**. Index X as $1:n$ and the **transition matrix** $P_{n\times n}$ and the stationary distribution $\pi P=\pi\in\mathbb{R}^n$ by definition are:

$$P_{ij}=\frac{k(x_i,x_j)}{d(x_i)}, \pi_i=\frac{d(x_i)}{\sum_j d(x_j)} \tag{2}$$

We define eigenpairs $(\lambda_l,\psi_l)$of P , namely, $P\psi_l=\lambda_l\psi_l$, $1=\lambda_0\geq|\lambda_1|\geq|\lambda_2|\geq\ldots$.

Clustering with DM

The power of P, i.e., P^t suggests the **diffusion distance** as t varies: t acts as a **scale parameter**. A naive clustering algorithm is that: choose a point x_i and take $P^k x_i, k \leq T$ in the class together with x_i.



(a) $t = 8$ (b) $t = 64$

Now we wanna give the so-called **diffusion distance** a more precise definition.

DM[2]

Next, we define a useful kernel, a little similar to the normalized graph Laplacian matrix:

$$a(x,y) := \frac{k(x,y)}{\sqrt{\pi(x)\pi(y)}} \tag{3}$$

Along with a convolution operator $\mathscr{P}$over functions on X, a.k.a. the **diffusion operator** :

$$\mathscr{P}[f(x)] := \int a(x,y)f(y)d\mu(y) \tag{4}$$

$p_t(x_i, x_j) := P^t_{ij}$ and we define the **diffusion distances** D_t at time t as:

$$D_t(x,y)^2 := \int (p_t(x,u) - p_t(y,u))^2 \frac{d\mu(u)}{\pi(u)} \tag{5}$$

It is proved that:

$$D_t(x,y)^2 = \sum_{l \geq 1} \lambda_l^{2t} (\psi_l(x) - \psi_l(y))^2 \tag{6}$$

Coordinates given by DM[2]

As dealt with last time, we will abandon those too small $|\lambda_t|$, for instance, those $l > L$, and thus get a bunch of coordinates Ψ_t, called **diffusion maps**.

$$\Psi_t = (\psi_1, \psi_2, ..., \psi_L)^T \tag{7}$$

The story on manifold learning ends here. Then comes the PDE.

DM and Δ[2]

Story about Δ

Consider the eigenproblems on manifold M:

$$\Delta\phi_l(x) = \lambda_l\phi_l, x \in M; \partial_v\phi_l = 0, x \in \partial M \tag{8}$$

It might be tough to get ϕ_l, but one can solve the eigenproblems immediately given ϕ_l's :

$$\partial_t u = \Delta u, x \in M; \partial_v u_l = 0, x \in \partial M \tag{9}$$

by a linear composition of $e^{-\lambda t}\phi(x)$:

$$u = \sum_l k_l e^{-\lambda_l t}\phi_l(x) \tag{10}$$

The wave equation can also be solved the same way. That is one reason why the eigenproblems of Δ matters!

DM and Δ

We've talked about the graph Laplacian L relates to Δ when the edge weight is $\exp\frac{|x-y|^2}{\epsilon}$. We will use this property to solve the PDE with Δ. Now we construct a **diffusion family** given $\alpha \in \mathbb{R}$.

Suppose $q(x)$ is the density of M(when M stands for a probability space), define the edge weight and its new distribution q_ϵ and new edge weight $k_\epsilon^{(\alpha)}$:

$$k_\epsilon(x,y) := h(\frac{|x-y|^2}{\epsilon}), q_\epsilon(x) := \int k_\epsilon(x,y)q(y)dy, k_\epsilon^{(\alpha)} := \frac{k_\epsilon(x,y)}{q_\epsilon^\alpha(x)q_\epsilon^\alpha(y)} \tag{11}$$

whose degree $d_\epsilon^{(\alpha)}$ and anisotropic transition kernel $p_{\epsilon,\alpha}$ is:

$$d_\epsilon^{(\alpha)}(x) = \int k_\epsilon^{(\alpha)}(x,y)q(y)dy, p_{\epsilon,\alpha} = \frac{k_\epsilon^{(\alpha)}(x,y)}{d_\epsilon^{(\alpha)}(x)} \tag{12}$$

Likewise, the convolution operator: $\mathscr{P}_{\epsilon,\alpha}[f(x)] := \int p_{\epsilon,\alpha}f(y)dy$ and define

$$\mathscr{L}_{\epsilon,\alpha} := \frac{I-\mathscr{P}_{\epsilon,\alpha}}{\epsilon} \tag{13}$$

DM and Δ

Theorem

$$\lim_{\epsilon \to 0} \mathscr{L}_{\epsilon,\alpha}[f] = \frac{\Delta(f q^{1-\alpha})}{q^{1-\alpha}} - \frac{\Delta q^{1-\alpha}}{q^{1-\alpha}} f \tag{14}$$

The proof is by magic[2].
Let $q = 1$, to solve $\mathscr{L}_{\epsilon,\alpha}[f] = 0$ is t o solve $\Delta f = 0$. At least now we know that $\lim_{\epsilon \to 0} \mathscr{L}_{\epsilon,1}$ and it is easy to yield: the Neumann heat kernel $e^{-t\Delta}$

$$\lim_{\epsilon \to 0} \mathscr{P}_{\epsilon,\alpha}^{\frac{\epsilon}{t}} = e^{-t\Delta} \tag{15}$$

Notice that $\mathscr{L}_{\epsilon,\alpha}$ is easy to compute on the point cloud. And therefore, one can solve PDE's that contain Δf on the point cloud by PINN readily, **without knowing anything about** M! That is what **[3]** all talks about. **[4]** published in July follows **[3]** with limited novelty.

Point Integral Method:PIM

The method in [2,3,4] can be viewed of somewhat a sort of PIM. Indeed, one can solve the PDE in one go[5]. A detailed proof lies in [6].
Consider the Neumann Poisson problem:

$$-\Delta u = f, x \in M; \partial_\nu u = g, x \in \partial M \tag{16}$$

As defined last time, $R(r)$ is positive and either decays exponentially or has compact support, with:

$$R'(r) := \int_r^{+\infty} R(s)ds \tag{17}$$

Then the PDE must satisfy:

$$-\frac{1}{t}\int_M (u(x)-u(y))R(\frac{|x-y|^2}{4t})dx = \int_M f(x)R'(\frac{|x-y|^2}{4t})dx + 2\int_{\partial M} g(x)R'(\frac{|x-y|^2}{4t})dx \tag{18}$$

PIM to solve Dirichlet Problem

However, PIM can only solve Neumann problem by giving a constraint. What if we wanna find out:

$$-\Delta u = f, x \in M; u = g, x \in \partial M \tag{19}$$

We can use a trick right here. Consider a Robin problem for a small β:

$$-\Delta u = f, x \in M; u + \beta \partial_\nu u = g, x \in \partial M \tag{20}$$

Replace g with $g - u$ in (18) as the PDE loss in PINN, one can solve (19). However, a small β may lead to ill-condition. One can solve it iteratively and use a β not necessarily to be small, which is called **Augmented Lagrangian Multiplier(ALM)**.

ALM

Procedure 1 ALM for Dirichlet Problem

$k = 0$, $w^0 = 0$.

repeat

  Solving the following integral equation to get v^k,

$$L_t v^k(\mathbf{y}) - \frac{2}{\beta}\int_{\partial\mathcal{M}} (g(\mathbf{x}) - v^k(\mathbf{x}) + \beta w^k(\mathbf{x}))\bar{R}_t(\mathbf{x},\mathbf{y})\mathrm{d}\tau_{\mathbf{x}} = \int_{\mathcal{M}} f(\mathbf{x})\bar{R}_t(\mathbf{x},\mathbf{y})\mathrm{d}\mu_{\mathbf{x}}.$$

  $w^{k+1} = w^k + \frac{1}{\beta}(g - (v^k|_{\partial\mathcal{M}}))$, $k = k+1$

until $\|g - (v^{k-1}|_{\partial\mathcal{M}})\| == 0$

$u = v^k$

where v^k, w^k are the iterative versions of $u, \partial_\nu u$, resp.

II: Equivariance

PDE solvers need invariance and equivariance?

Example

Think about estimating the curvature at a point on a point cloud. We input a local graph G and expect a real number as the output. It is apparent that this number is invariant when rotating, reflecting or translating G.

Example

Consider solving PDE on a disc Ω. $\Delta u = 0, x \in \Omega; u = f(x), x \in \partial\Omega$ The symmetry of Ω gives: $\forall A \in SO(2)$, the neural operator $\mathrm{NN}: f \mapsto u$ should satisfy $f \circ A \mapsto u \circ A = \lambda_A \circ u$

Literature

7 Theoretical aspects of group equivariant neural networks. Carlos Esteves. arXiv.

8 Group Equivariant Convolutional Networks. Taco S. Cohen,2016. (First)

9 Steerable CNNs. Taco S. Cohen. ICLR 2017.

10 Spherical CNN. Cohen et al. ICLR 2018 best paper.

11 “Learning SO(3) equivariant representations with spherical cnns”.Carlos Esteves et al. ECCV,2018.

12 Clebsch–Gordan Net . Risi Kondor. et al. NeurIPS 2018.

13 The 3D steerable CNNs. Taco Cohen et al. NeurIPS 2018.

14 Tensor field networks. Nathaniel Thomas et al.

Group

G is a group, if $\forall x, y \in G, xy \in G, x^{-1} \in G, e \in G, ex = xe = x \in G$

Group representation

$f : G \to H$ is a group homomorphism if $f(xy) = f(x)f(y)$. A **group representation** $\rho : G \to GL(V)$ is a special group homomorphism where V is usually $\mathbb{R}^n$ or $\mathbb{C}^n$. If $\rho_g^H \rho_g = I, \forall g \in G$, ρ is **unitary**. A **character** of ρ is $\chi_{\rho_g} := \operatorname{tr} \rho_g$

Example

Consider a finite group, a 4-order dihedral group $D_4 := \{s^i r^j : s^2 = r^4 = e, sr^j = r^{4-j}s\}$. $|D_4| = 8$. $\rho_g = 1, \forall g \in D_4$(trivial representation) $\rho'_{s^i r^j} = (-1)^i$, $\rho''_{s^i r^j} = \begin{pmatrix} (-1)^i & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \frac{j\pi}{2} & -\sin \frac{j\pi}{2} \\ \sin \frac{j\pi}{2} & \cos \frac{j\pi}{2} \end{pmatrix}$

Preliminary: Harr Measure

Harr Measure

Harr measure μ is a measure of the Hausdorff topological group G, whose open sets are generated by the Borel set $\mathscr{B}(G)$, s.t. $\forall E \in \mathscr{B}(G), g \in G, \mu(E) = \mu(gE)$, i.e., **left-invariant**.

Left Action Operator

f is a square-integrable function on G, denoted by $f \in L^2(G)$. $\lambda_g : L^2(G) \to L^2(G)$ is the left action by $\lambda_h f(g) := f(h^{-1}g)$.

If μ is Harr measure and λ_g is the left action operator, then:

$$\int_G \lambda_h[f(g)]d\mu(g) = \int_G f(h^{-1}g)d\mu(g) = \int_G f(g)\lambda_h d\mu(g) = \int_G f(g)d\mu(g) \tag{21}$$

Preliminary: Fourier Transform on Groups

For a unitary representation $\rho(g)$, we denote $\phi_{x,y}(g) = (\rho_g x, y)$ (with notation abuse, $\rho_g = \rho(g)$). The matrix element of $\rho_{ij}(g) := \phi_{e_i,e_j}(g) = (\rho_g e_i, e_j)$.
Let G' be the equivalence class of all **unitary irreducible representions**, a.k.a. **unirreps**, $\sqrt{d_\rho}\rho_{ij}, [\rho] \in G'$ forms an orthogonal basis of $L^2(G)$ where d_ρ is the dimension of the representation space.

$$f(g) = \sum_{[\rho]\in G'} \sum_{i,j}^{d_\rho} c_{ij}^{\rho} \rho_{ij}(g), c_{ij}^{\rho} = d_\rho \int_G f(g) \rho_{ij}^*(g) dg \tag{22}$$

Fourier Transform

The Fourier transform of a function on G is a linear map of the representation space!

$$\hat{f}(\rho) := \int_G f(g) \rho_g^* dg \Rightarrow f(g) = \sum_{[\rho]\in G'} d_\rho \operatorname{tr}(\hat{f}(\rho)\rho_g) \tag{23}$$

Equivariance

Convolution and Cross-correlation

$$(f * k)(g) := \int_G f(u)k(u^{-1}g)d\mu(u), (f \star k)(u) := \int_G f(u)k(g^{-1}u)d\mu(u) \tag{24}$$

Equivariance

We say f is **equivariant** to a group action G if $\forall g, \exists h(g), f \circ g(x) = h(g) \circ f(x)$. Fortuantely, **all convolution and cross-correlation are equivariant**, supported by the theorem:

$$(\lambda_u f) * k = \lambda_u(f * k), (\lambda_u f) \star k = \lambda_u(f \star k) \tag{25}$$

Convolution Theorem

$$\widehat{f * k} = \hat{f}\hat{k}, \widehat{f \star k} = \hat{f}\hat{k}^{\dagger}, \hat{k}^{\dagger}(\rho) := \int k(u)\rho(u^{-1})^* d\mu(u) \tag{26}$$

Applications: The Simplest

Literatures

8 Group Equivariant Convolutional Networks. Taco S. Cohen,2016. (First)

9 Steerable CNNs. Taco S. Cohen. ICLR 2017.

For finite group, the integral is defined as:

$$(f * k)(g) = \frac{1}{|G|} \sum_{x \in G} f(x) k(x^{-1} g) \tag{27}$$

In [8][9], the input is an image. We wish to construct the equivariance of the group, D_4 together with translation. If (i, j) denotes the coordinates of the pixel in an image, then (i, j) corresponds to the group action, i.e., translating i, j units along x, y axis, resp. This explains why **the usual convolution in CNN** makes sense**[8]**:

$$f * k_\theta = \sum_y f(x - y) k_\theta(y) \tag{28}$$

where k_θ is learnable. However, this only ensures **translation equivariance**.

Applications: The Simplest

[9] Steerable CNN

[9] brings in the D_4 equivariance by group representations **to reduce the parameters**. However, a usual $f * k$ doesn't involve any representation, so we use its Fourier transform instead. The layer is defined as:

$$\sigma(\hat{f}[v]\hat{k}) \tag{29}$$

where v is the feature from the previous layer and $\hat{k}$ is defined as since D_4 has 4 1-order unirreps and 1 2-order unirreps. $\hat{k}$ is defined as a linear combination of these unirreps and thus has 5 coefficients to learn.

Remark

[9] is actually a bit different. It analyzes which patterns are meaningful after rotation and reflection since the kernel is only 3×3.

Representations of $SL(2,\mathbb{C})$, SU(2),SO(3)

$SL(2,\mathbb{C})$

The representation space is the homogeneous polynomials of degree $2l$. Its matrix element is given by:

$$g = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL(2,\mathbb{C}), \det g = 1, \rho^l_{ij}(g) = \frac{\langle (-bx+a)^{l+j}(dx-c)^{l-j}, x^{l-i} \rangle}{\sqrt{(l-j)!(l+j)!(l-i)!(l+i)!}} \tag{30}$$

where the inner product is **Bombieri scalar product** defined on the homogeneous polynomials.

$SU(2)$

It can be factorized: $SU(2) \ni g = \begin{pmatrix} e^{-i\frac{\alpha}{2}} & \\ & e^{i\frac{\alpha}{2}} \end{pmatrix} \begin{pmatrix} \cos\frac{\beta}{2} & -\sin\frac{\beta}{2} \\ \sin\frac{\beta}{2} & \cos\frac{\beta}{2} \end{pmatrix} \begin{pmatrix} e^{-i\frac{\gamma}{2}} & \\ & e^{i\frac{\gamma}{2}} \end{pmatrix}$ Thus only those in the form of $\begin{pmatrix} e^{-i\frac{\alpha}{2}} & \\ & e^{i\frac{\alpha}{2}} \end{pmatrix}$ and $\begin{pmatrix} \cos\frac{\beta}{2} & -\sin\frac{\beta}{2} \\ \sin\frac{\beta}{2} & \cos\frac{\beta}{2} \end{pmatrix}$ needs considering.

Representations of $SL(2,\mathbb{C})$, SU(2),SO(3)

For $\begin{pmatrix} e^{-i\frac{\alpha}{2}} & \\ & e^{i\frac{\alpha}{2}} \end{pmatrix}$, plug in eq.(30) and yield

$$\rho^l(\begin{pmatrix} e^{-i\frac{\alpha}{2}} & \\ & e^{i\frac{\alpha}{2}} \end{pmatrix}) = \mathrm{diag}\{e^{-i\alpha m}\}, 1 \leq m \leq l \tag{31}$$

For the rotation matrix, substitute it into eq. define $P^l_{mn}(\cos\beta) = \rho^l_{mn}(\begin{pmatrix} \cos\frac{\beta}{2} & -\sin\frac{\beta}{2} \\ \sin\frac{\beta}{2} & \cos\frac{\beta}{2} \end{pmatrix})$ instead, we have

$$\rho^l_{mn}(g) = e^{-im\alpha - in\gamma} P^l_{mn}(\cos\beta) \tag{32}$$

This is the unirreps of $SU(2)$. ρ^l, P^l are termed **Wigner-D matrix** and **Wigner-d matrix**, resp.P^l_{mn} is **Jacobbi** polynomials, which degenerate into **Legendre poly.** P^l_{00} and **associated Legendre poly.** P^l_{m0}.

Representations of SO(3)

Ways to construct new representations

- **Lift**. Consider a quotient group, $H := G/Z$ and assume we have a representation ρ for H, then one can extend ρ to G by $\rho(g) := \rho(hZ)$ if $g \in hZ$.
- Conversely, to restrict a ρ on G to H, $\rho(gZ)$ must be single-valued for any $x \in gZ$. **This is the case for** $SO(3)$.
- Assume we have representations ρ_1, ρ_2 for G, then $\rho_1 \otimes \rho_2$ is also a representation.

SO(3)

Topologically, $SO(3) \simeq SU(2)/\{I, -I\}$. Thus $\rho^l(I) = \rho^l(-I)$ must hold to yield ρ^l for $SO(3)$. This forces l to be **integers only**.

Spherical Harmonic Functions Are Equivariant

The associated Legendre poly. $P^l_m = \sqrt{\frac{(l+m)!}{(l-m)!}} P^l_{m0}$ and the spherical harmonics are $Y^l_m(\theta,\phi) := \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} P^l_m(\cos\theta) e^{im\phi}$. A representation ρ must satisfy:

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2) \tag{33}$$

So its matrix elements must satisfy:

$$\rho^l_{m0}(g_1 g_2) = \sum_{n=-l}^{l} \rho^l_{mn}(g_1)\rho^l_{n0}(g_2) \Leftrightarrow Y^l_m(g_1 g_2) = \sum_{n=-l}^{l} (\rho^l_{mn}(g_1))^* Y^l_n(g_2) \tag{34}$$

If we identify $SO(3)$ as the element of S^2 , and rewrite g_1, g_2 by $g \in G, x \in S^2$, we have the equivariance for S^2 or $SO(3)$:

$$Y^l(gx) = (\rho^l(g))^* Y^l(x), Y^l(x) \in \mathbb{R}^{2l+1} \tag{35}$$

Spherical CNN

Spherical CNN [10,11]

Spherical CNN follows the idea above, by identifying S^2 as $SO(3)/SO(2)$. The first layer applies $f \star g = \int_{x \in S^2} k(g^{-1}x) f(x) dx$. To use sphereical harmonics, it uses Fourier transform on S^2[10] and the transform enjoys the property:

$$\widehat{f \star k}^l = (\hat{k}^l)^* (\hat{f}^l)^T \tag{36}$$

Likewise, [11] applies $f * g$ to construct equivariance instead.

Remark The first layer handles the unit vector in $\mathbb{R}^3$, while the subsequent layers tackle with vectors in $\mathbb{R}^3$. Thus $\hat{f}$ in the first layer utilizes the unireps induced from $SO(3)/SO(2)$ while other $\hat{f}$'s use the unireps of $SO(3)$ directly.

Clebsch-Gordan networks[12]

We know that if ρ_1, ρ_2 are irreps of $SO(3)$, then $\rho_1 \otimes \rho_2$ is also a representation of G, however, not necessarily irreducible. Clebsch-Gordan coefficients are used to make $\rho_1 \otimes \rho_2$ irreducible.

$$\rho_l = C^T_{l_1,l_2,l}(\rho_{l_1} \otimes \rho_{l_2})C_{l_1,l_2,l} \tag{37}$$

Features in the representation spaces of ρ_{l_1}, ρ_{l_2} are f^{l_1}, f^{l_2} . It is showed that the feature in the presentation space of ρ_l can be obtained via **CG transformation**:

$$f^l := (f^{l_1} \otimes f^{l_2})C_{l_1,l_2,l} \tag{38}$$

SE(3) Equivariance[13,14]

[13] implements $SE(3)$ equivariance. First, it enforces translation equivariance, i.e., $k \star f = \int_{\mathbb{R}^3} k^*(y-x)f(y)dy$. k, the function on G, is matrix-valued here. Then the $SO(3)$ equivariance property of f requires:

$$k \star (\pi_i(rx)f) = \pi_o(rx)(k \star f) \Rightarrow k(rx) = \rho_o(r)k(x)\rho_i(r)^{-1} \tag{39}$$

Use different irreps $\rho_i^{l_i}, \rho_o^{l_o}$, and it gives $k(rx) = \rho_o^{l_o} k_{l_o,l_i}(x)\rho_i^{l_i}$. Define

$$k_{l_o,l_i}(x) := R(|x|)\Phi(\frac{x}{|x|}), \Phi(x) := (Y^{l_o})^* \otimes (Y^{l_i})^T \tag{40}$$

, one can show that $\Phi(rx) = \rho_o^{l_o}\Phi(x)\rho_i^{l_i}$ and $R(|x|)$ is to be learned.
Works left is to vectorize (40) and use CG-coefficients to yield irreps of k_{l_o,l_i} with CG-coefficients that helps to decompose the tensor representation:

$$\rho^{l_1} \otimes \rho^{l_2} = C_{l_i,l_o}^T \bigoplus_{|l_1-l_2|\le l \le l_1+l_2} \rho^l C_{l_i,l_o} \tag{41}$$