Notes on *geometric deep learning*

These are notes on the following paper of Bronstein-Bruna-Cohen-Veličović [8], titled

Geometric deep learning: grids, groups, graphs, geodesics, gauges

The text below is not entirely my writing, nor is it taken entirely verbatim.

Contents

---

1 Introduction

Modern neural network (NN) design is built on two algorithmic principles: hierarchical feature learning (concerning the architecture of the NN), and learning by local gradient-descent driven by backpropagation (concerning the learning dynamics undergone by the NN).

We model an instance of training data is an element of some high-dimensional vector space, making a generic learning problem subject to the curse of dimensionality. Fortunately, most tasks of interest are not generic, inheriting regularities from the underlying low-dimensionality and structure of the physical world.

Exploiting known symmetries of a large system is a useful, classical remedy against the curse of dimensionality, and forms the basis of most physical theories. The notes [8] construct a blueprint for neural network architecture which incorporates these "physical" priors, termed **geometric priors** throughout the notes. Importantly, this blueprint provides a unified perspective of the most successful neural network architectures.

1.1 Categories and groups

Definition 1.1. A ***graph*** is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set whose elements are called ***vertices***. The set $\mathcal{E}$ consists of ***edges***, defined to be a multi-set of exactly two vertices in $\mathcal{V}$, not necessarily distinct.

Definition 1.2. A ***directed graph*** is a pair of sets $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ of vertices and ***arrows*** (or ***directed edges***). An ***arrow*** is an *ordered pair* of vertices.

Definition 1.3. Consider an arrow f of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, specifically $f \equiv (a, b) \in \mathcal{A}$, with $a, b \in \mathcal{V}$. The operations dom and cod act on the arrows $f \in \mathcal{A}$ via $\mathrm{dom}\, f = a$, $\mathrm{cod}\, f = b$, and are called the ***domain*** operation and ***codomain*** operation, respectively.

Given two arrows, f and g in some directed graph, we say that the ordered pair of arrows (g, f) is a ***composable pair*** if $\mathrm{dom}\, g = \mathrm{cod}\, f$. Going forward, let us express the relations $a = \mathrm{dom}\, f$ and $b = \mathrm{cod}\, f$ more concisely via

$$f : a \to b \qquad \text{or equivalently,} \qquad a \xrightarrow{f} b$$

The next definition formalizes the behavior of a collection of structure-respecting maps between mathematical objects.

Definition 1.4. A ***category*** is a directed graph $C = (\mathcal{O}, \mathcal{A})$, whose vertices $\mathcal{O}$ we call ***objects***, such that

(i) For each object $a \in \mathcal{O}$, there is a unique ***identity*** arrow $\mathrm{id}_a \equiv 1_a : a \to a$, defined by the following property: for all arrows $f : b \to a$ and $g : a \to c$, composition with the identity arrow id_a gives

$$\mathrm{id}_a \circ f = f \quad \text{and} \quad g \circ \mathrm{id}_a = g$$

(ii) For each composable pair (g, f) of arrows, there is a unique arrow $g \circ f$ called their ***composite***, with $g \circ f : \mathrm{dom}\, f \to \mathrm{cod}\, g$, such that the composition operation is associative. Namely, for given objects and arrows in the configuration $a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{k} d$, one always has the equality $k \circ (g \circ f) = (k \circ g) \circ f$.

Examples are given in the appendix. Given a category $\mathcal{C} = (\mathcal{O}, \mathcal{A})$, let

$$\hom(b, c) := \{ f : f \in \mathcal{A},\ \mathrm{dom}\, f = b \in \mathcal{O},\ \mathrm{cod}\, f = c \in \mathcal{O} \}$$

denote the set of arrows from b to c. Henceforth, we use the terms ***morphism*** and arrow interchangeably.

Groups are collections of symmetries. A ***group*** G is a category $\mathcal{C} = (\mathcal{O}, \mathcal{A})$ with $\mathcal{O} = \{o\}$ (so that we may identify G with the collection of arrows $\mathcal{A}$) such that each arrow has a unique inverse: for $g \in \mathcal{A}$, there is an arrow h such that $g \circ h = h \circ g = \mathrm{id}_o$.

Each arrow $g \in \mathcal{A}$ thus has $\mathrm{dom}\, g = \mathrm{cod}\, g = o$. As remarked, the arrows $g \in \mathcal{A}$ correspond to group elements $g \in G$. The categorical interpretation suggests that the group *acts* on some abstract object $o \in \mathcal{O}$. In the present context, we care how groups act on data, and how this action is represented to a computer.

Group representations

Linear representation theory allows us to study groups using linear algebra. (a source). We start by considering a function $\varphi : G \times V \to V$, where G is a group, and where V is a vector space over $\mathbb{R}$. This allows us to identify group elements g with functions $\varphi(g, \cdot) : V \to V$ from the vector space to itself. When the map φ is understood, or general (as now), we write $g.v$ in place of $\varphi(g, v)$, and we write $(g.)$ in place of $\varphi(g, \cdot)$.

The 'representatives' $(g.)$ of these group elements g can be composed, and if this compositional structure is compatible with the original group operation, we say φ is a ***group action*** on V. Specifically, φ should satisfy $e.v = v$ for all $v \in V$, where e denotes the identity element of G, and in general one has $(gh).v = g.(h.v)$.

The map φ is $\mathbb{R}$***-linear*** if it is compatible with the $\mathbb{R}$-vector space structure on V, i.e. additive and homogeneous. Specifically, if for all $v, w \in V$ and all scalars $\lambda \in \mathbb{R}$, one has $g.(v + w) = g.v + g.w$ and $g.(\lambda v) = \lambda g.v$.

Definition 1.5. An $\mathbb{R}$***-linear representation*** of group G over $\mathbb{R}$-vector space V is an $\mathbb{R}$-linear group action on V.

The next example illustrates how linear group representations arise naturally when considering group actions on data. As mentioned, we consider input data as members of some vector space V, which we may assume to be finite dimensional for any practical discussion. Specifically, we consider some finite, discrete domain Ω, which may also have the structure of an undirected graph.

A ***signal*** over Ω is a function $x : \Omega \to \mathbb{R}^s$, where s is the number of ***channels***. The vector space $\mathcal{X}(\Omega, \mathbb{R}^s)$ is defined to be the collection of all such signals, for given Ω and s

Example 1 (RGB image)**.** Consider, for some $n \in \mathbb{N}$, a signal domain $\Omega = \mathbb{T}_n^2$, where $\mathbb{T}_n^2$ denotes the two-dimensional discrete torus of side-length n, namely $(\mathbb{Z}/n\mathbb{Z})^2$. This domain has natural graph as well as group structures.

If we imagine each vertex of Ω to be a pixel, we can express an $n \times n$-pixel color (RGB) image as a signal $x : \Omega \to \mathbb{R}^3$, with the first, second and third coordinates of $\mathbb{R}^3$ reporting R, G and B values of a given pixel. We make two observations:

(1) As a vector space, $\mathcal{X}(\Omega)$ is isomorphic to $\mathbb{R}^d$, with d typically very large. In the above example, $d = 3n^2$, which is thirty-thousand for a $n \times n \equiv 100 \times 100$ pixel image.

(2) Any group action on Ω induces a group action on $\mathcal{X}(\Omega)$.

Expanding on the latter, consider a group action of G on domain Ω. As the torus Ω already has group structure, it is natural to think of it acting on itself through translations, i.e. we now additionally consider $G = \mathbb{T}_n^2$.

The action of $G \equiv \mathbb{T}_n^2$ on itself $\Omega \equiv \mathbb{T}_n^2$ induces a G-action on $\mathcal{X}(\Omega)$ as follows: for $g \in G$ signal $x \in \mathcal{X}(\Omega)$, the action $(g, x) \mapsto \boldsymbol{g}.x \in \mathcal{X}(\Omega)$ is defined pointwise at each $u \in \Omega$:

$$(\boldsymbol{g}.x)(u) := x(g.\omega),$$

where the bold $(\boldsymbol{g}.)$ is used to distinguish the action on signals from the action on the domain.

To summarize: any G-action on the domain Ω induces an $\mathbb{R}$-linear representation of G over the vector space of signals on Ω.

Example 2 (One-hot encoding)**.** It seems like standard practice to encode the collection of classes associated to some ML classification problem as an orthonormal basis. These are given (to the computer) in the usual coordinate basis

$$e_1 \equiv (1, 0, \dots, 0),\ e_2 \equiv (0, 1, \dots, 0),\ \dots,\ e_n \equiv (0, \dots, 0, 1)\,,$$

hence the nomenclature ***one-hot***. In the preceding example, if one considers a one-hot encoding of the vertices of $\mathbb{T}^2_n$, we see that each signal is expressed with respect to this coordinate system, in the sense that $x = \sum_{j=1}^{n} x_j e_j$.

This kind of encoding is useful for considering general symmetries of the domain. For instance, if permuting node labels is a relevant symmetry, the action of the symmetric group $\mathfrak{S}_n$ is naturally represented by $n \times n$ permutation matrices.

Signals on graphs as node features

The following definition reformulates the notion of a signal over the nodes of some graph as *node features*.

Definition 1.6. We say a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is equipped with ***node features*** if for each $v \in \mathcal{V}$, one has the additional data of an s-dimensional vector $x(v) \in \mathbb{R}^s$, called the ***features*** of node v.

Remark 1.7. The term 'features' is compatible with the usage in ML, supposing that our input signal has domain some graph $\mathcal{G}$. In this case, we can think of a neural network as a sequence of node-layers built "on top of" the graph $\mathcal{G}$. An input signal endows the first node layer of a NN with features, and the weights of the neural network propagate these through to node features on the nodes of the rest of the network. The features on the last layer of the network can be read off as the output of the NN function.

1.2 Equivariance

We henceforth consider groups G acting on some space of signals $\mathcal{X}(\Omega)$ over a (fixed) signal domain Ω. The group action is encoded in a linear representation ρ, assumed to be described in a given ***input coordinate system***, just as we would need to specify to a computer. Thus, if $\dim(\mathcal{X}(\Omega)) = n$, for each $g \in G$, we have that $\rho(g)$ is an $n \times n$ matrix with real entries.

The learning dynamics occur in the ***hypothesis space*** H, a collection of functions

$$\mathsf{H} \subset \{\, F : \mathcal{X}(\Omega) \to \mathcal{Y} \,\},$$

where $\mathcal{Y}$ is some unspecified (context-specific) space of output signals. We describe H explicitly in Subsection 1.4, up to hyperparameters such as depth and layer widths. A key aspect of this blueprint is that $F \in \mathsf{H}$ should (according to the blueprint) be expressed as a composition of functions, most of which are G-equivariant. The requirement of G-equivariance constitutes a 'geometric prior' from which one can derive the architecture of a generic CNN when G corresponds to translations, and a family of generalizations for other domains and group actions.

Definition 1.8. Let ρ be a representation of group G over $\mathcal{X}(\Omega)$, and let ρ' be a representation of the same group over the $\mathbb{R}$-vector space $\mathcal{Y}$. A function $F : \mathcal{X}(\Omega) \to \mathcal{Y}$ is ***G-equivariant*** if for all $g \in G$ and $x \in \mathcal{X}(\Omega)$, we have $\rho'(g)F(x) = F(\rho(g)x)$. We say F is ***G-invariant*** if this holds when ρ' is the trivial representation, which is to say $F(\rho(g)x) = F(x)$ for all $g \in G$ and $x \in \mathcal{X}(\Omega)$.

Example 3. Suppose we are given either a set $\mathcal{V}$, or more generally a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\#\mathcal{V} = n$ in either case. As discussed, a signal over $\mathcal{V} = \{v_1, \dots, v_n\}$ can be thought of as a collection of node features $\{\, x(v_1), \dots, x(v_n) \,\}$, with $x(v_j) \in \mathbb{R}^s$. We stack the node features as rows of the $n \times s$ ***design matrix***

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},$$

which is effectively the same object as signal x, provided the vertices are labeled as described. The action of $g \in \mathfrak{S}_n$ on this input data is naturally represented as an $n \times n$ permutation matrix, $\boldsymbol{P} \equiv \rho(g)$. One standard way to construct a permutation invariant function F in this setting is through the following recipe: a function ψ is independently applied to every node's features, and φ is applied on its sum-aggregated outputs.

$$F(\boldsymbol{X}) = \varphi\left(\sum_{j=1}^{n} \psi(x_j) \right).$$

Such a function can be thought of as reporting some 'global statistic' of signal x.

Equivariance manifests even more naturally. Suppose we want to apply a function F to the signal to *update* the node features, producing a set of ***latent*** (node) features. [a] We can stack these latent features into another design matrix,

$\boldsymbol{H} = F(\boldsymbol{X})$. The order of the rows of $\boldsymbol{H}$ should clearly be tied to the order of the rows of $\boldsymbol{X}$, i.e. permutation equivariant: for any permutation matrix $\boldsymbol{P}$, it holds that $F(\boldsymbol{PX}) = \boldsymbol{P}F(\boldsymbol{X})$.

As a concrete example of a permutation equivariant function, consider a weight matrix $\theta \in \mathbb{R}^{s \times s'}$. This matrix can be used to map a length-s feature vector at a given node to some new, updated feature vector with s'-channels. Applying this matrix to each row of the design matrix is an example of a ***shared node-wise linear transform***, and constitutes a linear, $\mathfrak{S}_n$-equivariant map. Note that, if one wishes to express this map in coordinates, it seems the correct object to consider is a 3-tensor, "Θ," constructed as a stack of n copies of θ.

[a]This is the case in which the NN outputs an image segmentation mask; the underlying domain does not change, but the features at each node are updated to the extent that they may not even agree on the number of channels.

The above example considered signals over the nodes of a graph only, thus label permutation symmetry applies equally well, regardless of the graph structure (or lack of structure) underlying such functions.

In the case that signals x have a domain with graph structure, it feels right to want to work with a hypothesis space recognizing this structure, which is encoded by the graph's adjacency matrix. This is the $n \times n$ matrix $\boldsymbol{A} \equiv (a_{uv})$, where $a_{uv} = 1$ iff $u \sim v$, and $a_{uv} = 0$ otherwise. This is to say that we wish to consider functions $F \in \mathsf{H}$ with $F \equiv F(\boldsymbol{X}, \boldsymbol{A})$. Such a function is (label) ***permutation invariant*** if $F(\boldsymbol{PX}, \boldsymbol{PAP}^{\mathrm{T}}) = F(\boldsymbol{X}, \boldsymbol{A})$, and is ***permutation equivariant*** if

$$F(\boldsymbol{PX}, \boldsymbol{PAP}^{T}) = \boldsymbol{P}F(\boldsymbol{X}, \boldsymbol{A}) \tag{1.1}$$

for any permutation matrix $\boldsymbol{P}$.

Remark 1.9. *(to return to)* One can characterize linear $\mathfrak{S}_n$-equivariant functions on nodes, as discussed in Example 3, namely functions F satisfying $F(\boldsymbol{PX}) = \boldsymbol{P}F(\boldsymbol{X})$. From [8],

> *"One can verify any such map can be written as a linear combination of two generators, the identity and the average. As observed by Maron et al. 2018, any linear* $\mathbf{F}$ *satisfying* (1.1) *can be expressed as a linear combination of fifteen linear generators; remarkably, this family is independent of* $n \equiv \#\mathcal{V}$*."*

Remark 1.10. Amongst the generators just discussed, the geometric learning blueprint "specifically advocates" *(where??)* for those that are also local, in the sense that the output on node u directly depends on its neighboring nodes in the graph.

We can formalize this constraint explicitly, by defining the 1-***hop neighborhood*** of node u as

$$\mathcal{N}(u) := \{v : \{u, v\} \in \mathcal{E}\},$$

as well as the corresponding ***neighborhood features***,

$$\boldsymbol{X}_{\mathcal{N}(u)} := \{\{\, x_v : v \in \mathcal{N}(u)\,\}\},$$

which is a multiset (indicated by double-brackets) for the simple reason that distinct nodes may be decorated with the same feature vector.

Example 4. The node-wise linear transformation described in Example 3 can be thought of as operating on '0-hop neighborhoods' of nodes. We generalize this with an example of a function operating on 1-hop neighborhoods. Instead of a fixed map between feature spaces $\theta : \mathbb{R}^s \to \mathbb{R}^{s'}$ *cloned* to a pointwise map Θ, we specify a *local* function

$$\varphi \equiv \varphi(\, x_u,\, \boldsymbol{X}_{\mathcal{N}(u)}\,)$$

operating on the features of a node as well as those of its 1-hop neighborhood.

We may construct a permutation equivariant function Φ (the analogue of Θ here, just as φ corresponds to θ) by applying φ to each 1-hop neighborhood in isolation, and then collecting these into a new feature matrix. As before, for $\mathcal{V} = \{v_1, \ldots, v_n\}$, we write x_j in place of $x_{v(j)}$, and we similarly write $\mathcal{N}_j$ instead of $\mathcal{N}(v(j))$.

$$\Phi(\boldsymbol{X}, \boldsymbol{A}) = \begin{bmatrix} \varphi(\, x_1,\, \boldsymbol{X}_{\mathcal{N}_1}\,) \\ \varphi(\, x_2,\, \boldsymbol{X}_{\mathcal{N}_2}\,) \\ \vdots \\ \varphi(\, x_n,\, \boldsymbol{X}_{\mathcal{N}_n}\,) \end{bmatrix}$$

The permutation equivariance of Φ rests on the output of φ being independent of the ordering of the nodes in $\mathcal{N}_u$. Thus, if φ is permutation invariant (a local averaging) this property is satisfied.

[8] : *"The choice of φ plays a crucial role in the expressive power of the learning scheme."*

Remark 1.11. The authors [8] emphasize that, when considering signals $x \equiv \boldsymbol{X}$ over graphs, it is natural to consider a hypothesis space whose functions operates on the pair $(\boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{A}$ is the adjacency matrix of the signal domain. Thus, for such signals the domain naturally becomes part of the input.

The GDL blueprint distinguishes between two contexts: one in which the input domain is fixed, and another in which the input domain varies from signal to signal. The preceding example demonstrates that, even in the former context, it can be essential that elements of H treat the (fixed) domain as an argument.

When the signal domain has geometric structure, it can often be leveraged to construct a *coarsening operator*, one of the components of a GDL block in the learning blueprint. Such an operator passes a signal $x \in \mathcal{X}(\Omega)$ to a signal $y \in \mathcal{X}(\Omega')$, where Ω' is a 'coarsened version' of Ω. As the blueprint calls for a sequence of such blocks, we often wish to act on the latent signal y by a pointwise non-linearity, followed by the action of an equivariant function on signals in $\mathcal{X}(\Omega')$. The domain may be fixed for each input, but this domain changes as the signal passes through the layers of the NN, and it is natural that the functions the NN is built out of should pass this data forward.

> [8] : *"Due to their additional structure, graphs and grids, unlike sets, can be coarsened in a non-trivial way, giving rise to a variety of pooling operations... more precisely, we cannot define a non-trivial coarsening assuming set structure alone. There exist established approaches that infer topological structure from unordered sets, and those can admit non-trivial coarsening"*

1.3 Tensors

Coordinate transformations of vectors

Consider a pair of orthonormal bases, the *old basis* $(e_1, \dots, e_n)$ and the *new basis* $(\tilde{e}_1, \dots, \tilde{e}_n)$. A signal $x \in \mathbb{R}^n$ can be expressed in either coordinate system

$$\begin{aligned} x &= e_1 x^1 + \cdots + e_n x^n \;\equiv\; e_j x^j \\ x &= \tilde{e}_1 \tilde{x}^1 + \cdots + \tilde{e}_n \tilde{x}^n \;\equiv\; \tilde{e}_j \tilde{x}^j, \end{aligned}$$

where we have used the Einstein summation convention. We adopt this notation going forward. We can express the basis $(\tilde{e}_j)$ in terms of the (e_j) via

$$\tilde{e}_j = e_i \boldsymbol{S}^i_j \,,$$

where $\boldsymbol{S}$ is called the *direct transformation matrix* from the old basis to the new. We let $\boldsymbol{T}$ denote $\boldsymbol{S}^{-1}$, and note that the coordinate coefficients transform *contravariantly*:

$$\tilde{x}^j = \boldsymbol{T}^j_i x^i \,.$$

We assume that the underlying vector space is $\mathbb{R}^n$, and moreover, we equip it with the usual inner product (in order to make sense of orthogonality), and note that the Riesz representation theorem allows us to assign a given orthonormal basis (e_j) of

vector space V to a canonical dual basis (e^j) on the dual vector space V^*. Elements of V^* are called dual vectors, or covectors. This canonical dual basis is given by

$$\begin{aligned} e^j &= \langle e_j, \cdot \rangle, \\ &\equiv e_i \, \delta^{ij}, \end{aligned}$$

and thus,

$$e_j = \delta_{ij} \, e^i.$$

Remark 1.12. The dual basis can be thought of as a 'coordinate selector,' acting on a vector $x \in V$ by

$$\begin{aligned} e^j(x) &= e^j \, e_k \, x^k \\ &= x^k \, e^j \, e_k \\ &= \delta^j_k \, x^k \\ &= x^j. \end{aligned}$$

Remark 1.13. Let us consider how the objects

$$\delta_{ij} \equiv e_i \, e_j, \quad \delta^j_i \equiv e_i \, e^k, \quad \delta^{ij} \equiv e^i \, e^j$$

transform with the bases, i.e. how to express the following new basis objects

$$\tilde{\delta}_{ij}, \quad \tilde{\delta}^j_i, \quad \tilde{\delta}^{ij}$$

in terms of the corresponding old, via the matrices $\boldsymbol{S}$ and $\boldsymbol{T}$. One has

$$\begin{aligned} \tilde{\delta}_{ij} = \tilde{e}_i \, \tilde{e}_j &= e_\ell \, \boldsymbol{S}^\ell_i \, e_k \, \boldsymbol{S}^k_j \\ &= \boldsymbol{S}^\ell_i \, \delta_{\ell k} \, \boldsymbol{S}^k_j \end{aligned}$$

Likewise, one has

$$\tilde{\delta}^j_i = \boldsymbol{S}^\ell_i \, \delta^k_\ell \, \boldsymbol{T}^j_k \quad \text{and} \quad \tilde{\delta}^{ij} = \boldsymbol{T}^i_\ell \, \delta^{\ell k} \, \boldsymbol{T}^j_k$$

Given a change of basis $(e_j) \mapsto (\tilde{e}_j)$, the induced transformation $(e^j) \mapsto (\tilde{e}^j)$ is contravariant, which is consistent with the transformations described above:

$$\begin{aligned} \tilde{e}^j &\equiv \tilde{e}_i \, \delta^{ij} \\ &= e_\ell \, \boldsymbol{S}^\ell_i \, \boldsymbol{T}^i_\ell \, \delta^{\ell k} \, \boldsymbol{T}^j_k \\ &= e_\ell \, \delta^{\ell k} \, \boldsymbol{T}^j_k \\ &= \boldsymbol{T}^j_k e^k \end{aligned}$$

Definition 1.14. A ***tensor of type***, or ***valency*** (r, s) over vector space V is a multilinear map

$$\boldsymbol{A} : \underbrace{V^* \times \ldots \times V^*}_{k \text{ copies}} \times \underbrace{V \times \ldots \times V}_{\ell \text{ copies}} \to \mathbb{R}$$

When a basis (e_j) is given, we can express $\boldsymbol{A}$ with respect to this coordinate system through a total of n^{r+s} scalars, denoted generically by $\boldsymbol{A}^{i_1,\ldots,i_r}_{\quad j_1,\ldots,j_s}$. The coordinate indices $i_1, \ldots, i_r$ are the ***contravariant indices***, while the $j_1, \ldots, j_s$ are the ***covariant indices***. They are so-named because of the transformation law of the entries under a change of basis $(e_j) \mapsto (\tilde{e}_j)$ induces the linear transformation

$$\tilde{\boldsymbol{A}}^{i_1,\ldots,i_r}_{\quad j_1,\ldots,j_s} = \boldsymbol{T}^{i_1}_{k_1} \ \ldots \ \boldsymbol{T}^{i_r}_{k_r} \, \boldsymbol{A}^{k_1,\ldots,k_r}_{\quad \ell_1,\ldots,\ell_s} \, \boldsymbol{S}^{\ell_1}_{j_1} \ \ldots \ \boldsymbol{S}^{\ell_s}_{j_s}$$

Let $\mathcal{T}(r, s)$ denote the vector space of type-(r, s) tensors over V. The dimension of this vector space is n^{r+s}, i.e. the number of scalar entries in the coordinate representation of such a tensor.

There are two operations on tensors of interest

- contraction
- outer product

1.4 GDL blueprint (with input domain fixed)

Setup

Our formal treatment of a classification problem starts with the following objects:

Ω : A graph with n vertices.

$(e_j)_{j=1}^n$: An 'input coordinate system.' Each vector in the orthonormal basis e_j corresponds to a vertex of Ω. This is the basis in which signals are expressed to a computer.

$\mathcal{X}(\Omega, \mathbb{R}^s)$: The space of s-channel signals $x : \Omega \to \mathbb{R}^s$. In terms of (e_j), one can write x as

$$x = \boldsymbol{x}^{cj} e_j,$$

where the index j varies over the nodes of Ω, and where index c indicates the channel. Note that $\boldsymbol{x} \equiv \boldsymbol{X}$, the so-called *design matrix* previously introduced.

G : A group G acting on Ω, via $\Omega \ni u \mapsto g.u \in \Omega$. This induces the 'pointwise' action on signals, $x \mapsto \boldsymbol{g}.x$.

ρ : The representation of the induced action of G, $x \mapsto \boldsymbol{g}.x$. Let $g \in G$. Supposing signals are expressed with respect to (e_j), we can then identify $\rho(g)$ with a TENSOR...

GDL block

Let Ω and Ω' be domains, G a symmetry group over Ω, and write $\Omega' \prec \Omega$ if Ω' arises from Ω through some coarsening operator (presumably this coarsening operator needs to commute with the group action).

(1) *linear G-equivariant layer*

$$B : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega', C')$$

such that $B(g.x) = g.B(x)$ for all $g \in G$ and $x \in \mathcal{X}(\Omega, C)$.

(2) *nonlinearity*, or *activation function*

$$s : C \to C'$$

applied domain-pointwise as $(\mathbf{s}(x))(u) = s(x(u))$.

(3) *local pooling operator* or *coarsening operator*

$$P : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega', C)$$

which gives us our notion $\Omega' \prec \Omega$.

The last ingredient is a global pooling layer applied last, compositionally.

(4) *G-invariant layer*, or *global pooling layer*

$$A : \mathcal{X}(\Omega, C) \to \mathcal{Y}$$

satisfying $A(g.x) = A(x)$ for all $g \in G$ and $x \in \mathcal{X}(\Omega, C)$.

Hypothesis space

These objects can be used to define a class of G-invariant functions $f : \mathcal{X}(\Omega, C) \to \mathcal{Y}$ of the form

$$f = A \circ \mathbf{s}_J \circ B_J \circ P_{J-1} \circ \cdots \circ P_1 \circ \mathbf{s}_1 \circ B_1,$$

where the blocks are selected so that the output space of each block matches the input space of the next one. Different blocks may exploit different choices of symmetry groups G.

1.5 Discussion

Remark 1.15. Shift-invariance arises naturally in vision and pattern recognition. In this case, the desired function $f \in \mathsf{H}$, typically implemented as a CNN, inputs an image and outputs the probability of the image to contain an object from a certain class. It is often reasonably assumed that the classification result should not be affected by the position of the object in the image, i.e., the function f must be shift-invariant.

Multi-layer perceptrons lack this property, a reason why early (1970s) attempts to apply these architectures to pattern recognition problems failed. The development of NN architectures with local weight sharing, as epitomized by CNNs, was among other reasons motivated by the need for shift-invariant object classification.

A prototypical application requiring shift-equivariance is image segmentation, where the output of f is a pixel-wise image mask. This segmentation mask must follow shifts in the input image. In this example, the domains of the input and output are the same, but since the input has three color channels while the output has *one channel per class*, the representations $(\rho, \mathcal{X}(\Omega, C))$ and $(\rho', \mathcal{Y} \equiv \mathcal{X}(\Omega, C'))$ are somewhat different.

Remark 1.16. When f is implemented as a CNN, it may be written as a composition of L functions, where L is determined by the depth and other hyperparameters:

$$f = f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1.$$

Examining the individual layer functions making up CNN, one finds they are not shift-invariant in general but rather shift-equivariant. The last function applied, namely f_L, is typically a “global-pooling" function that is genuinely shift-invariant, causing f to be shift-invariant, but to focus on this ignores the structure we will leverage for purposes of expressivity and regularity.

2 Geometric deep learning models

2.1 Learning with scalar-signals on a cyclic group

In the simplest setting, the underlying domain is a one-dimensional grid, and the signals only have a single channel. We can identify this grid Ω with the Cayley graph of cyclic group

$$C_n = \langle\, a : a^n = 1 \,\rangle \equiv \{\, 1, a, a^2, \ldots, a^{n-1} \,\}.$$

It is convenient to parametrize the group, and hence the grid, through the exponent of the generator

$$C_n \equiv \{0, 1, \ldots, n-1\}$$

as this indexing is consistent with the way most programming languages index vectors, and clearly we can reinterpret the group operation as addition modulo n. The vector space of single-channeled (i.e. real-valued) signals

$$\mathcal{X}(C_n, \mathbb{R}) = \{x : C_n \to \mathbb{R}\},$$

is finite dimensional, and each $x \in \mathcal{X}(C_n, \mathbb{R})$ may be expressed as

$$x = \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

with respect to some implicit coordinate system used by the computer, the "**input coordinate system**." This is the same coordinate system used by the representation ρ of translation group $G \equiv C_n$, which we now describe.

Given a vector $\theta = (\theta_0, \ldots, \theta_{n-1})$, recall the associated *circulant matrix* is the $n \times n$ matrix with entries

$$\boldsymbol{C}(\theta) := \left(\, \theta_{(u-v) \mod n} \right)_{0 \le u, v \le n-1}$$

Consider the case of $\theta_S := (0, 1, 0, \ldots, 0)^T$, the associated circulant matrix, $\boldsymbol{S} := \boldsymbol{C}(\theta_S)$ acts on vectors by shifting the entries of vectors to the right by one position, modulo n. This is a shift or translation operator, which we denote $\boldsymbol{S}$.

Lemma 2.1. *A matrix is circulant if and only if it commutes with $\boldsymbol{S}$. Moreover, given any two vectors $\theta, \eta \in \mathbb{R}^n$, one has $\boldsymbol{C}(\theta)\boldsymbol{C}(\eta) = \boldsymbol{C}(\eta)\boldsymbol{C}(\theta)$.*

The importance of $\boldsymbol{S}$ to the present discussion is that it generates a group isomorphic to the 'one-dimensional translation group' C_n. This is to say, a natural representation of $C_n = \langle\, a : a^n = 1 \,\rangle$ to consider is the group isomorphism induced by mapping

the generator a of C_n to $\boldsymbol{S}$. Specifically, the representation ρ of G over $\mathcal{X}(C_n, \mathbb{R})$ is given by

$$\rho(a^j) := \boldsymbol{S}^j$$

Corollary 2.2. *Any $f : \mathcal{X}(C_n, \mathbb{R}) \to \mathcal{X}(C_n, \mathbb{R})$ which is linear and C_n-equivariant can be expressed (in the input coordinate system) as an $n \times n$ circulant matrix $\boldsymbol{C}(\theta)$ for some vector θ.*

Example 5. Our previous recipe for designing an equivariant function $F = \Phi(\boldsymbol{X}, \boldsymbol{A})$ using a local aggregation function φ. In this case, we can express

$$\varphi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}(u)}) = \varphi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1}),$$

where the addition and subtraction in the indices above is understood to be modulo n.

If in addition, we insist that φ is linear, then it has the form

$$\varphi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1}) = \theta_{-1}\mathbf{x}_{u-1} + \theta_0\mathbf{x}_u + \theta_1\mathbf{x}_{u+1},$$

and in this case we can express $\mathbf{F} = \Phi(\mathbf{X}, \mathbf{A})$ through the following matrix multiplication:

$$\begin{bmatrix} \theta_0 & \theta_1 & & & \theta_{-1} \\ \theta_{-1} & \theta_0 & \theta_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \theta_{-1} & \theta_0 & \theta_1 \\ \theta_1 & & & \theta_{-1} & \theta_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n-2} \\ \mathbf{x}_{n-1} \end{bmatrix}$$

This special multi-diagonal structure is sometimes referred to as “weight sharing" in the machine learning literature.

Circulant matrices are synonymous with discrete convolutions; for $x \in \mathcal{X}(\Omega, \mathbb{R})$ and $\theta \in \mathbb{R}^n$, their **convolution** $x \star \theta$ is defined by

$$\begin{aligned}(x \star \theta)_u &:= \sum_{v=0}^{n-1} x_v \mod n\, \theta_{(u-v)} \mod n\,, \\ &\equiv \boldsymbol{C}(\theta)x\end{aligned}$$

Remark 2.3. This leads to a possible alternate but equivalent *definition* of convolution as a translation equivariant linear operation, moreover by replacing ‘translation’ by a more general group G, one can generalize convolution to settings whose domain has symmetry other than translational.

2.2 A simple CNN

Thus, we come to possibly the simplest neural network that we can construct through the above blueprint. Suppose we have a binary classification problem, with the following hypothesis space. Let H_1 denote the hypothesis space of functions $f : \mathcal{X}(C_n, \mathbb{R}) \to \{0, 1\}$ of the form

$$f = A \circ P_1 \circ \boldsymbol{r}_1 \circ B_1 \,,$$

where the components of f are

B_1 : A C_n-equivariant function, to be learned. It is represented as a circulant matrix $\boldsymbol{C}(\theta)$, where θ is a vector $\theta \equiv (\theta_0, \ldots, \theta_{n-1})$ whose entries θ_j are parameters to be learned.

$\boldsymbol{r}_1$: We consider the ReLU activation function, $r_1 : \mathbb{R} \to \mathbb{R}_{\geq 0}$ defined by $r_1(w) = \max(0, w)$, for $w \in \mathbb{R}$. The bold-face denotes the entry-wise action of this function on a given vector, thus for $y \equiv (y_1, \ldots, y_n) \in \mathcal{X}(C_n, \mathbb{R})$, which we imagine as the output of $B_1(x)$ for some input signal x, we have $\boldsymbol{r}_1(y) = (\max(0, y_1), \ldots, \max(0, y_n))$. There are no parameters to be learned, in this layer.

P_1 : A coarsening operator. In this case, let us say it is a group homomorphism $P_1 : C_n \to C_{n/d}$ for some divisor $d \mid n$ [1] , and let us say that it operates through max-pooling on the signal, over the pre-images of each element of $C_{n/d}$.

A : A global-pooling layer. We assume this has the form of a fully-connected layer, followed by a softmax. Specifically,

The categorical notation

$$\Omega_1 \equiv C_n \xrightarrow{f} \Omega \xrightarrow{g} \xrightarrow{k} d$$

is nice, as it makes explicit the domain at each stage in the composition.

[1] zero-padding

2.3 Implementation in TensorFlow keras

Let us make clear which libraries we are importing:

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
```

Remark 2.4. A note on 'python inheritance:' Inheritance allows us to define a class that inherits all the methods and properties from another class. The parent class is the class being inherited from, also called base class. The child class is the class that inherits from another class, also called derived class.

In the keras API, the **layers** class is the child of the **models** class, and all concrete layer methods inherit from the **layers** class, such as **Conv2D**.

Before considering a simple GDL block, let us examine how to initialize the specific layer type, **Conv2D**, which plays the role of the G-equivariant function in a generic GDL block, where G is the translation group.

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None, **kwargs
)
```

Let us discuss each argument of the **Conv2D** method:

- *filters*: This is the “dimension of the output space," namely the number of output channels. Given that this layer is plays the role of a G-equivariant layer, $E : \mathcal{X}(\Omega, \mathbb{R}^s) \to \mathcal{X}(\Omega, \mathbb{R}^{s'})$, the filters argument is the value of s'.
- *kernel size*: the height and width of the two-dimensional convolution window.
- *strides and dilation rate*: placing a non-trivial argument ($\neq 1$) in both of these slots will raise an error.

Let us consider the following GDL block:

```
model.add(
layers.Conv2D(
32,
(3,3),
activation = relu,
input_shape = (32,32,3)
)
)
model.add(layers.MaxPooling2D( (2,2) )
```

2.4 Convolutional neural networks

In section (4.2), we have fully characterized the class of linear and local translation equivariant operators, given by convolutions $C(\theta)x = x \star \theta$ with a localized filter θ. Let us first focus on scalar-valued (‘single-channel’ or ‘grayscale’) discretized images, where the domain is the grid $\Omega = [H] \times [W]$, and $\mathbf{x} \in \mathcal{X}(\Omega, \mathbb{R})$, with generic $u \in \Omega$ having coordinate description $u = (u_1, u_2)$.

Any convolution with a compactly supported filter of size $H^f \times W^f$ can be written as a linear combination of generators $\theta_{1,1}, \ldots, \theta_{H^f, W^f}$, given for example by the unit peaks

$$\theta_{vw}(u_1, u_2) = \delta(u_1 - v, u_2 - w).$$

Any local linear equivariant map is thus expressible as

$$F(\mathbf{x}) = \sum_{v=1}^{H^f} \sum_{w=1}^{W^f} \alpha_{vw} C(\theta_{vw})\mathbf{x}\,,$$

Remark 2.5. Note that we usually imagine $\mathbf{x}$ and θ_{vw} as matrices, but in the above display, both $\mathbf{x}$ and θ_{vw} have their two coordinate dimensions flattened into one, making $\mathbf{x}$ a vector and $C(\theta_{uv})$ a matrix.

In coordinates, this corresponds to the familiar two-dimensional convolution:

$$F(\mathbf{x}) = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \alpha_{ab} x_{u+a,\, v+b}$$

Other choices of the basis θ_{vw} are also possible, and will yield equivalent operations (for potentially different choices of α_{uv}). A popular example are *"directional derivatives"*:

$$\theta_{vw}(u_1, u_2) = \delta(u_1, u_2) - \delta(u_1 - v, u_2 - w), \quad (v, w) \neq (0, 0)$$

taken together with the local average, $\theta_0(u_1, u_2) = \frac{1}{H_f W_f}$. These are related to diffusion processes on the grid – [8].

When the scalar input channel is replaced by multiple channels (e.g., RGB colors, or more generally an arbitrary number of feature maps), the convolutional filter becomes a *convolutional tensor* expressing arbitrary linear combinations of input features into output maps. In coordinates, this can be expressed as

$$F(\mathbf{x})_{uvj} = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \sum_{c=1}^{M} \alpha_{jabc} x_{u+a,v+b,c} \quad j \in [N] \tag{2.1}$$

where M and N are respectively the number of input and output channels.

Efficient multiscale computation

As discussed in the GDL template for general symmetries, extracting translation invariant features out of the convolutional operator F requires a non-linear step. Convolutional features are processed through a non-linear *activation function* σ, acting element-wise on the input, i.e. $\sigma : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$, as $\sigma(\mathbf{x})(u) = \sigma(\mathbf{x}(u))$.

The most popular example at the time of writing is the **rectified linear unit (ReLU)**,

$$\sigma(x) = \max(x, 0)$$

> [8] : *"This non-linearity effectively* rectifies *the signals, pushing their energy towards lower frequencies, and enabling the computation of high-order interactions across scales by iterating the construction"*

In the early works of Fukushima and Miyake (1982) and LeCun et al. 1998, CNNs and similar architectures had a multiscale structure, where after each convolutional layer (2.1) one performs a grid coarsening $P : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega')$, where the grid Ω' has coarser resolution than Ω.

This allows for multiscale filters with effectively increasing receptive field, yet retaining a constant number of parameters per scale. Several coarsening strategies(called *pooling*) may be used, the most common are applying a low-pass anti-aliasing filter (e.g. local average) followed by grid downsampling, or non-linear max pooling (?)

In summary, a 'vanilla' CNN layer can be expressed as a composition of basic objects from the learning blueprint,

$$h = P(\sigma(F(x))$$

Deep and residual networks

A CNN architecture in its simplest form is specified by the hyperparameters

$$(H_k^f, W_k^f, N_k, p_k)_{k \leq K},$$

with $M_{k+1} = N_k$ and $p_k = 0, 1$ indicating whether grid coarsening is performed or not. While all these hyperparameters are important in practice, a particularly important question is to understand the role of depth K in CNN architectures, and what are the fundamental tradeoffs involved in choosing such a key hyperparameter, especially in relation to the filter sizes (H_k^f, W_k^f).

While a rigorous answer to this question is still elusive, mounting empirical evidence collected throughout recent years suggests a favorable tradeoff towards deeper (large K) yet thinner (small (H_k^f, W_k^f)).

A crucial insight by He et al. (2016) was to reparametrize each convolutional layer to model a perturbation of the previous features, rather than a generic non-linear transformation:

$$h = P(x + \sigma(F(x)))$$

The resulting residual networks provide several key advantages over the previous formulation. In essence, the residual parametrization is consistent with the view that the deep network is a discretization of an underlying continuous dynamical system, modelled as an ODE. Crucially, learning a dynamical system by modeling its velocity turns out to be much easier than learning its position directly. In our learning setup, this translates into an optimization landscape with more favorable geometry, leading to the ability to train much deeper architectures than before.

The key advantage of the ResNet parametrization has been rigorously analyzed in simple scenarios (Hardt and Ma, 2016), and remains an active area of theoretical investigation.

Finally, neural ODEs (Chen et al. 2018) are a recent popular architecture that pushes the analogy with ODEs even further, by learning the parameters of the ODE $\dot{\mathbf{x}} = \sigma(F(\mathbf{x}))$

Normalization

Another important algorithmic innovation that boosted empirical CNN performance significantly is the notion of **normalization**. In early models of neural activity, it was hypothesized that neurons perform some form of local 'gain control', where the layer coefficients x_k are replaced by

$$\tilde{x}_k = \sigma_k^{-1} \odot (x_k - \mu_k),$$

where μ_k and σ_k encode the first- and second-order moment information of x_k respectively. *"Further, they can be either computed globally or locally"*

In the context of Deep Learning, this principle was widely adopted through the **batch normalization layer** (Ioffe-Szegedy, 2015), followed by several variants. Despite some attempts to rigorously explain benefits of normalization in terms of better conditioned optimization landscapes (Santurkar et al., 2018), a general theory that can provide guiding principles is still missing at the time of writing.

Data augmentation

While CNNs encode the geometric priors associated with translation invariance and scale separation, they do not explicitly account for other known transformations that preserve semantic information, e.g. lighting or color changes, or small rotations and dilations.

A pragmatic approach to incorporate these priors with minimal architectural changes is to perform **data augmentation**, where one manually performs said transformations to the input images and adds them to the training set.

Data augmentation has been empirically successful and is widely used – not only to train statee-of-the-art vision architectures, but also to prop up several developments in self-supervised and causal representation learning (Chen et al., 2020; Grill et al., 2020; Mitrovic et al., 2020). However, it is provably sub-optimal in terms of sample complexity (Mei et al., 2021); a more efficient strategy considers instead architectures with richer invariance groups, as we discuss next.

2.5 Group-equivariant CNNs

As previously discussed, we can generalize the convolution operations from signals on Euclidean space to signals on any *homogeneous space* Ω acted upon by a group G. By analogy to the Euclidean convolution where a *translated* filter is matched with the signal, the idea of group convolution is to move the filter around the domain using the group action. By virtue of the transitivity of the group action, we can move the filter

to any position on Ω. [2] We now discuss several concrete examples of the general idea of group convolution, including implementation aspects and architectural choices.

Discrete group convolution

We begin by considering the case in which Ω and G are both discrete.

(1) As our first example, we consider medical volumetric images represented as signals on a three-dimensional grid. We think of input signals, a priori supported on a finite discrete box, as signals on $\Omega \equiv \mathbb{Z}^3$ through **zero-padding**. As symmetry group G, we consider "$\mathbb{Z}^3 \rtimes O_h$" [3] Here O_h is the octahedral group [4] generated by rotations about the three coordinate axes, so these are all orientation-preserving graph automorphisms of the lattice.

(2) As our second example, consider DNA sequences made up of four letters: C, G, A, T. The sequences are represented as signals on the one-dimensional grid $\Omega = \mathbb{Z}$, which take values in $\mathbb{R}^4$ naturally, through the **one-hot** encoding of each letter, i.e. each letter is identified with a distinct standard basis element of $\mathbb{R}^4$. Naturally, we can consider the discrete one-dimensional translation symmetry group $\mathbb{Z}$ as a factor of G acting on such signals.

DNA sequences have an additional interesting symmetry. This symmetry arises from the way DNA is physically embodied as a double helix, and the way it is read by the molecular machinery of the cell. Each strand of the double helix begins with what is called the 5′-end and ends with a 3′-end, with the 5′ on one strand complemented by a 3′ on the other strand. In other words, the two strands have opposite orientation.

Since the DNA molecule is always read starting at the 5′-end, but we do not know which one, a sequence such as ACCCTGG is equivalent to the reversed sequence with each letter replaced by its complement, CCAGGGT. This is called **reverse-complement symmetry** of the sequence. The two element group $\mathbb{Z}_2 = \{0, 1\}$ thus acts on strings in the following way: 0 is represented as the identity, while 1 is represented as the reverse-complement. The acting symmetry group is thus $\mathbb{Z} \times \mathbb{Z}_2$.

In this case, group convolution defined earlier is given as

$$(x \star \theta)(g) = \sum_{u \in \Omega} x_u \rho(g) \theta_u,$$

the inner product between the (single-channel) input signal x and a g-transformed filter θ by some $g \in G$, via $\rho(g)\theta_u = \theta_{g^{-1}u}$, and the output $x \star \theta$ is a function on G.

[2] So vertex-transitive graphs are the discrete analogue of a homogeneous space

[3] Does $\rtimes$ just denote the cartesian product for them?

[4] (do we need a prefix 'special' ?)

Transform+convolve approach

We show that group convolution can be implemented in two steps: a filter transformation step, and a translational convolution step.

In the context of the two examples mentioned, the filter transformation step consists of creating rotated (or reverse-complement transformed) copies of a basic filter, while the translational convolution is the same as in standard CNNs, and thus efficiently computable on hardware such as GPUs.

To see this, note that in both our examples (.. have product structure $G = T \times R$ ..) , we can write a general transformation $g \in G$ as a composition $g = tr$ of a “rotation" $r \in G$ and a translation $t \in G$. That ρ is a group representation, we have $\rho(g) = \rho(t)\rho(r)$, so

$$\begin{aligned}(x \star \theta)(tr) &= \sum_{u \in \Omega} x_u \rho(t)\rho(r)\theta_u \\ &= \sum_{u \in \Omega} x_u (\rho(r)\theta)_{u-t}\end{aligned}$$

The word rotation is in quotes because we can think of r as either a genuine rotation, as in the first example, where $r \in O_h$, or as an action of the reverse complement symmetry, $r \in \mathbb{Z}_2$, in the second example. The last equation above is the standard (planar Euclidean) convolution of the signal x and the transformed filter $\rho(h)\theta$.

Thus, to implement group convolution for these groups, we take the canonical filter θ, create transformed copies $\theta_r = \rho(r)\theta$ for each $r \in R$, and then convolve x with each of these filters: $(x \star \theta)(tr) = (x \star \theta_r)(t)$.

For both of our examples

3 Fourier analysis on groups

3.1 Fourier analysis on a cyclic group

Derivation of the discrete Fourier transform

- Recall that (diagonalizable) matrices are *jointly diagonalizable* if and only if they mutually commute.
- Thus, there exists a common eigenbasis for all the circulant matrices, and we can compute this basis by finding the eigenvectors of *any* circulant matrix.
- The basis of the shift operator happens to be the discrete Fourier basis. For $k = 0, 1, \ldots, n-1$, let

$$\varphi_k = \frac{1}{\sqrt{n}}\left(1,\, e^{2\pi \mathrm{i} k/n},\, e^{4\pi \mathrm{i} k/n},\, \ldots,\, e^{2\pi \mathrm{i}(n-1)k/n}\right)$$

- We can arrange this basis into an $n \times n$ "Fourier matrix" $\Phi = (\varphi_0, \ldots, \varphi_{n-1})$. Multiplication by Φ^* gives the *discrete Fourier transform (DFT)*, and multiplication by Φ gives the inverse DFT,

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{u=0}^{n-1} x_u e^{-2\pi \mathrm{i} k u/n}, \quad x_u = \frac{1}{\sqrt{n}} \sum_{u=0}^{n-1} \hat{x}_k e^{+2\pi \mathrm{i} k u/n}$$

- The eigenvalues of $\mathbf{C}(\boldsymbol{\theta})$ are the Fourier transform of the filter, which is a restatement of the convolution theorem:

$$\mathbf{C}(\boldsymbol{\theta})\boldsymbol{x} = \Phi \begin{bmatrix} \hat{\theta}_0 & & \\ & \ddots & \\ & & \hat{\theta}_{n-1} \end{bmatrix} \Phi^* \mathbf{x} = \boldsymbol{\Phi}(\hat{\boldsymbol{\theta}} \odot \hat{\mathbf{x}})$$

- Because the Fourier matrix Φ has special algebraic structure, the matrix products $\Phi^*\boldsymbol{x}$ and $\Phi\boldsymbol{x}$ can be computed with $O(n \log n)$ complexity using a *Fast Fourier Transform (FFT)* algorithm. This is a reason for the popularity of frequency domain filtering. Moreover, the filter is typically designed directly in the frequency domain, so $\hat{\boldsymbol{\theta}}$ is never explicitly computed.

On generalizing Fourier transform, and discussion of continuous version

- Realizing that the adjacency matrix of the ring graph is exactly the shift operator, one can develop the graph Fourier transform and an analogy of the convolution operator by computing the eigenvectors of the adjacency matrix.

- Attempts to develop graph neural networks by analogy to CNNs, sometimes termed *spectral GNNs*, exploited this exact blueprint. We will see later that this analogy has some important limitations: the first comes from the fact that a grid is fixed, and hence all signals on it can be represented in the same Fourier basis. On general graphs, the Fourier basis depends on the structure of the graph. "Hence, we cannot directly compare Fourier transforms on two different graphs, a problem that translated into a lack of generalization in machine learning problems. "

- Moreover without any notions of direction, we can only organize the Fourier basis functions by their frequencies (energies), and graph filters built in this way are oblivious of direction[5]

- We now consider the continuum analogue of the above discussion. Consider functions defined on $\Omega = \mathbb{R}$, and the translation operator $(S_v f)(u) = f(u - v)$.

- As previously discussed, if we apply S_v to the Fourier basis functions $\varphi_\xi(u) = e^{i\xi u}$ yields, by associativity of the exponent,

$$S_v e^{i\xi u} = e^{-i\xi v} e^{i\xi u},$$

  which implies φ_ξ is a complex eigenvector of S_v with complex eigenvalue $e^{-i\xi v}$.

- Moreover, the spectrum of the translation operator is simple, which means that any two functions with the same eigenvalue must be co-linear: suppose that $S_v f = e^{-i\xi_0 v} f$ for some frequency ξ_0. Taking the Fourier transform of both sides, we have for all ξ,

$$e^{i\xi v} \hat{f}(\xi) = e^{-i\xi_0 v} \hat{f}(\xi),$$

  implying $\hat{f}(\xi) = 0$ for $\xi \neq \xi_0$, thus $f = \alpha \varphi_{\xi_0}$.

- For a general linear operator C that is translation equivariant, which is to say $S_v C = C S_v$, we have

$$S_v C e^{i\xi u} = C S_v e^{i\xi u} = e^{-i\xi v} C e^{i\xi u},$$

  and thus $Ce^{i\xi u}$ is also an eigenfunction of S_v with eigenvalue $e^{-i\xi v}$, from which we once again use simplicity of the Fourier spectrum to conclude that $Ce^{i\xi u} = \beta \varphi_\xi(u)$.

- In other words, the Fourier basis is the eigenbasis of all translation equivariant operators.

- As a result, C is diagonal in the Fourier domain, and can be expressed as $Ce^{i\xi u} = \hat{p}_C(\xi) e^{i\xi u}$, where $\hat{p}_C(\xi)$ is a *transfer function* acting on different frequencies [6]

[5] but why is this important?

[6] so this just deforms one diagonal matrix into another

- For an arbitrary function $x(u)$, by linearity,

$$\begin{aligned}(Cx)(u) &= C\int_{-\infty}^{\infty}\hat{x}(\xi)e^{\mathrm{i}\xi u}\mathrm{d}\xi \\ &= \int_{-\infty}^{\infty}\hat{x}(\xi)\hat{p}_C(\xi)e^{\mathrm{i}\xi u}\mathrm{d}\xi \\ &= \int_{-\infty}^{\infty}p_C(v)x(u-v)\mathrm{d}v \\ &= (x\star p_C)(u),\end{aligned}$$

  where $p_C(u)$ is the inverse Fourier transform of $\hat{p}_C(\xi)$. It thus follows that every linear translation equivariant operator is a convolution. [7]

Fourier analysis on $\mathbb{R}$

Before describing multiscale representations we review some concepts in harmonic analysis. Arguably the most famous signal decomposition is the *Fourier transform*. In one dimension,

$$\hat{x}(\xi) \triangleq \int_{-\infty}^{\infty} x(u)\exp(-\mathrm{i}\xi u)\mathrm{d}u$$

expresses the function $x \in L^2(\Omega)$ on the domain $\Omega \equiv \mathbb{R}$ as a linear combination of orthogonal oscillating *basis functions* $\varphi_\xi(u) = \exp(\mathrm{i}\xi u)$ indexed by their *frequency* ξ, i.e. their rate of oscillation.

- Recall that the **convolution** of signal x with **filter** θ is

$$(x\star\theta)(u) \triangleq \int_{\mathbb{R}} x(v)\theta(u-v)\mathrm{d}v,$$

  which is a standard model of linear signal filtering. The **cross-correlation** of x with θ is identical to the convolution up to a sign change in the argument of the filter:

$$(x\star\theta)(u) \triangleq \int_{\mathbb{R}} x(v)\theta(u+v),$$

Remark 3.1. Following what seems to be a common ML convention, we use the *same notation* for both transforms: “since the filter is typically learnable, the distinction is purely notational. "

[7] (from text) The spectral characterization of the translation group is a particular case of a more general result in functional analysis, *Stone’s theorem*

- The Fourier transform intertwines convolution and multiplication, which is to say that for all frequences $\xi \in \mathbb{R}$,

$$\widehat{x \star \theta}(\xi) = \hat{x}(\xi) \cdot \hat{\theta}(\xi)$$

- Many fundamental differential operators such as the Laplacian are described as convolutions on Euclidean domains. Such operators can be defined intrinstically over very general geometries, providing a formal procedure of extending objects to the menagerie of domain-types we consider.

- An essential aspect of Fourier transforms is that they reveal *global* properties of the signal and the domain, such as smoothness or conductance. Such global behavior is convenient for studying domains with many global symmetries, such as translation, but less so for the more general symmetries we consider.

- We require a signal decomposition able to trade between spatial and frequential localisation, as we see next.

- Our discussion of grids highlighted how shifts and convolutions are intimately connected: convolutions are linear shift-equivariant operations, and any shift-equivariant linear operator is a convolution. These operators are diagonalized by the Fourier transform.

- Both convolution and the Fourier transform can be defined for any group of symmetries that we can sum or integrate over.

- Consider once again $\Omega = \mathbb{R}$. We can understand convolution as a pattern-matching operation: we match shifted copies of filter $\theta(u)$ with an input signal $x(u)$. The value of the "convolution" (but really, it is cross-correlation defined below) [8] $(x \star \theta)(u)$ at a point u is the inner product of the signal x with the filter shifted by u:

$$(x \star \theta)(u) = \int_{\mathbb{R}} x(v)\theta(u+v)\mathrm{d}v$$

- Note that in this case, u is both a point on the domain $\Omega = \mathbb{R}$, and also an element of the translation group, which we can identify with the domain itself.

- Thus we can generalize the above construction by replacing the translation group by another acting on Ω.

[8] I think I've asked this before, but is there any utility to the hypothesis space to allow ±1 in argument of filter to be learned, to allow both convolution and cross-corellation to coexist, going from layer to layer? This sounds like it messes with equivariance, and is not well-posed.

- As discussed previously in abstract, the action of the group G on the domain Ω induces a *representation* ρ of G on the space of signals $\mathcal{X}(\Omega)$ via $\rho(g)x(u) = x(g^{-1}u)$.
- In the previous example, G is the translation group, whose elements act by shifting the coordinates $u + v$, whereas $\rho(g)$ is the shift operator acting on signals as $(S_v x)(u) = x(u - v)$.
- In the abstract setting, in order to apply a filter to the signal, we invoke our assumption of $\mathcal{X}(\Omega)$ being a Hilbert space, with an inner product[9]

$$\langle x, \theta \rangle = \int_\Omega x(u)\theta(u)\mathrm{d}u$$

- Having defined how to transform signals (with the group) and match them with filters, we can define the *group convolution* for signals on Ω:

$$(x \star \theta)(g) = \langle x, \rho(g)\theta \rangle,$$

  and we note that $x \star \theta$ takes as argument elements g of the group G, rather than points in the domain Ω.

- Hence the next layer, which takes $x \star \theta$ as input, should act on signals defined on the group G, a point we return to shortly.
- Just as the traditional Euclidean convolution is shift-equivariant, the generalized convolution is G-equivariant.
- Here is a statement whose importance I don't yet understand: "the key observation is that matching the signal x with a g-transformed filter $\rho(g)\theta$ is the same as matching the inverse transformed signal $\rho(g^{-1}))x$ with the untransformed filter. We can express this as

$$\langle x, \rho(g)\theta \rangle = \langle \rho(g^{-1})x, \theta \rangle,$$

  a statement relating the adjoint of $\rho(g)$ to its inverse. The G-equivariance of the group convolution follows from

$$\begin{aligned}(\rho(h)x \star \theta)(g) &= \langle \rho(h)x, \rho(g)\theta \rangle \\ &= \langle x, \rho(h^{-1}g)\theta \rangle \\ &= \rho(h)(x \star \theta)(g)\end{aligned}$$

[9] so (why) is it wrong to view the filter as a signal? And do we need completeness or just an inner product?

Examples using specific groups

- We have studied above the case of a one-dimensional grid, the choice of domain here is $\Omega = \mathbb{Z}_n = \{0, \dots, n-1\}$ and the cyclic shift group $G = \mathbb{Z}_n$.
- In this case, an element $g \in G$ is identified with some $u = 0, \dots, n-1$ such that $g.v = v - u \mod n$, whereas the inverse element is $g^{-1}.v = v + u \mod n$.
- Importantly, in this example the elements of the group (shifts) are elements of the domain ("indices"). We can thus, with some abuse of notation, identify the two structures (i.e. , $\Omega = G$); our expression for group convolution in this case

$$(x \star \theta)(g) = \sum_{v=0}^{n-1} x_v \theta_{g^{-1}v}$$

  leads to the familiar convolution $(x \star \theta) = \sum_{v=0}^{n-1} x_v \theta_{v+u \mod n}$.

Spherical convolution

- Now consider the two-dimensional sphere $\Omega = \mathbb{S}^2$ with the group of rotations, the *special orthogonal group* $G = SO(3)$.
- This is a pedagogical choice, but has practical applications. In astrophysics, for example, observational data often naturally has spherical symmetry. Furthermore, spherical symmetries are very important in applications in chemistry when modeling molecules and trying to predict their properties.
- Representing a point on the sphere as a three-dimensional unit vector u, the action of the group can be represented as a 3×3 orthogonal matrix $\mathbf{R}$ with $\det(\mathbf{R}) = 1$. The spherical convolution can thus be written as the inner product between the signal and the rotated filter,

$$(x \star \theta)(\mathbf{R}) = \int_{S^2} x(u)\theta(\mathbf{R}^{-1}u)du$$

- Note that the group is now not identical to the domain; SO(3) is a Lie group, a (3-)manifold with group structure, while S^2 is two-dimensional.
- Consequently, convolution is a function on SO(3), rather than on Ω.
- This has important practical consequences: in our geometric deep learning blueprint, we concatenate multiple equivariant maps ("layers" in deep learning jargon) by applying a subsequent operator to the output of the previous one.

- In the case of translations, we can apply multple convolutions in sequence, since their outputs are all defined on the same domain Ω. In the general setting, since $x \star \theta$ is a function on G rather than Ω, we cannot use exactly the same operation subsequently – and this means the next operation must consider or “deal with" signals son G, i.e. $x \in \mathcal{X}(G)$.

- Our definition of group convolution allows for this: we can always consider G as a domain Ω, which is acted upon by group G, namely itself.

- This yields the representation $\rho(g)$ acting on $x \in \mathcal{X}(G)$ by $(\rho(g)x)(h) = x(g^{-1}h)$.

- Just as before, the inner product is defined by integrating the point-wise product of the signal and the filter over the domain, which now equals $\Omega = G$.

- In our example of spherical convolution, a second layer of convolution would thus have thee form

$$((x \star \theta) \star \eta)(\mathbf{R}) = \int_{\mathrm{SO}(3)} (x \star \theta)(\mathbf{Q})\eta(\mathbf{R}^{-1}\mathbf{Q})\mathrm{d}\mathbf{Q}$$

- As convolution involves an inner product defined through integration over the domain Ω, we can only use[10] it on domains Ω that are small (in the discrete case) or low-dimensional (in the continuous case)

- For instance, we cannot in practice perform convolutions with respect to the group of permutations.

- Nevertheless, we can still build equivariant convolutions for large groups G by working with signals defined on low-dimensional spaces Ω on which G acts. Indeed, it is possible to show that any equivariant linear map $f : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega')$ between two domains Ω, Ω' can be written as a generalized convolution, similar to the group convolution discussed here.

- The Fourier transform we derived in the previous section from the shift-equivariance property of the convolution can also be extended to a more general case by projecting the signal onto the matrix elements of irreducible representations of the symmetry group. In the case of SO(3), this gives rise to *spherical harmonics*.

- Finally, we point to the assumption that has so far underpinned our discussion in the section: whether Ω was a grid, plane or sphere, we could transform every point into any other point (vertex transitivity), intuitively meaning that all the

[10] in practice? They seem to imply that six dimensions is not small enough, which is surprising: “likewise, integrating over the affine group in three dimensions, which has a six-dimensional linear representation

points on the domain “look the same." A domain with such a property is a *homogeneous space,* and this simply generalizes the definition of vertex transitivity: for any $u, v \in \Omega$, there is $g \in G$ such that $g.u = v$.

4 Geometric objects

- In a previous example, we considered $\mathbb{S}^2$ as a manifold, and one with global symmetry structure (it is a homogeneous space).
- The majority of manifolds do not have such global symmetries, and in this case, we can't straightforwardly define an action of G on the space of signals over Ω and use it to "slide" filters around the manifold to construct a convolution.
- Nonetheless, manifolds have two types of invariance that we will explore in this section: transformations preserving metric structure and local reference frame change.
- In computer graphics and vision, manifolds are a common mathematical model of surfaces (embedded in three dimensions), used to represent boundaries of objects in some three dimensional scene.
- These objects allow for ignoring the internal structure of the object, if it is not relevant to the graphics, and in structural biology, internal folding of a protein molecule is often irrelevant for interactions that happen on the molecular surface.
- Manifolds are deformable, and many slight deformations of objects (as in the surface of a human figure gradually changing its pose) essentially preserve intrinsic metric structure, the embedding is changing. This furnishes an example of one of the more general symmetries mentioned: metric preserving maps, namely isometries.
- By the previous point, manifolds fall under the setting of *varying domains* in the geometric deep learning blueprint.

4.1 Riemannian manifolds

- A *manifold* is a paracompact Hausdorff space locally homeomorphic to $\mathbb{R}^n$. [11]
- Saying locally homeomorphic to $\mathbb{R}^n$ is somewhat vague: we don't want the dimension to vary from neighborhood to neighborhood, or across connected components.

[11] From wikipedia: *"In mathematics, a paracompact space is a topological space in which every open cover has an open refinement that is locally finite. These spaces were introduced by Dieudonné (1944). Every compact space is paracompact. Every paracompact Hausdorff space is normal, and a Hausdorff space is paracompact if and only if it admits partitions of unity subordinate to any open cover."* Thus paracompactness is a definition tailored for useful topological tools of bump functions and partitions of unity.

- To be specific, a manifold M must come equipped with an *atlas* $\mathcal{A}$ of charts $\alpha : U_\alpha \to \widetilde{U}_\alpha \subset \mathbb{R}^d$, each of which is a homeomorphism between $U_\alpha \subset M$ open and a bounded, open, connected domain in $\mathbb{R}^d$. The dimension d is the same for all charts, and we require the U_α to form an open cover of M.

- A *smooth manifold* is a manifold $(M, \mathcal{A})$ with the following "smooth compatibility" constraint on the atlas $\mathcal{A}$. Given two charts α and β with $U_\alpha \cap U_\beta$ non-empty, let

  $V_{\alpha,\beta} := \alpha(U_\alpha \cap U_\beta)$ be this region of overlap in "α-coordinates," and

  let $V_{\beta,\alpha} := \beta(U_\alpha \cap U_\beta)$ be this region of overlap in "β-coordinates."

  One can consider the map $T_{\alpha,\beta} : V_{\alpha,\beta} \to V_{\beta,\alpha}$ given by the composition $\beta \circ \alpha^{-1}$ restricted to $V_{\alpha,\beta}$, and while in general this is a homeomorphism, for a smooth manifold we require this to be a diffeomorphism.[12]

- With smooth structure, one can now define a *tangent space* T_xM at each $x \in M$ in the following way:

  We consider the family of curves of the form $\gamma : (-\epsilon, \epsilon) \to M$, for some small ϵ,

  $$C_x = \{\gamma : \ \gamma \text{ is smooth, } \gamma(0) = x \}$$

  where smoothness means "smooth when viewed from a(ny) coordinate chart." We can now from equivalence classes out of the above family, with the relation $\gamma \sim \eta$ if, under any chart whose domain is a neighborhood of x, one has $\gamma'(0) = \eta'(0)$. We then define

  $$T_xM := C_x / \sim$$

- Thus, tangent vectors are (or can be viewed as) equivalence classes of curves passing through a given $x \in M$, and curves are equivalent if their velocities agree at x.

- One can endow the collection of all tangent spaces

  $$TM := \bigsqcup_{x \in M} T_xM$$

  with the structure of a smooth $2d$-manifold, this space is called the *tangent bundle*.

- A Riemannian manifold (M, g) is a smooth manifold $(M, \mathcal{A})$ equipped with a *Riemannian metric* g, where $g_x : T_xM \times T_xM \to \mathbb{R}$ in a way depending smoothly on u, such that this map defines an inner product within each tangent space,

  $$\langle X, Y \rangle_u = g_u(X, Y),$$

  which gives us a notion of length and angle within each tangent space.

[12] maybe I want my neighborhoods to be contractible, in addition to connected?

- Note that tangent vectors are abstract geometric objects, and which exist without imposing any coordinate system on the manifold. If we express a tangent vector X as an array of numbers, we can only represent it as a list of coordinates $(x_1, \dots, x_d)$ relative to some local basis $\{X_1, \dots, X_d\} \subset T_u M$. Similarly, the metric can be expressed as a $d \times d$ matrix G with elements $g_{ij} = g_u(X_i, X_j)$ in that basis.

Riemannian manifolds and symmetry

- Properties which can be expressed entirely in terms of the metric g are called *intrinsic*. This is a crucial notion for the discussion, as according to the template, we seek to construct functions acting on signals defined over Ω that are invariant to isometries.
- We can generate such functions by looking at functions of intrinsic quantities, and we recover a notion of geometric stability when extending the discussion to approximate isometries.

4.2 Scalar and vector fields

- A (smooth) *scalar field* is a function of the form $x : \Omega \to \mathbb{R}$. Scalar fields form a vector space $\mathcal{X}(\Omega, \mathbb{R})$ with inner product

$$\langle x, y \rangle = \int_\Omega x(u) y(u) \, \mathrm{d}u,$$

  where $\mathrm{d}u$ is the volume form induced by the metric g.
- A (smooth) *tangent vector field* is a function of the form $X : \Omega \to T\Omega$ assigning to each point a tangent vector in the respective tangent space, i.e. $u \mapsto X(u) \in T_u \Omega$.
- Vector fields also form a vector space $\mathcal{X}(\Omega, T\Omega)$ with the inner product defined through the Riemannian metric,

$$\langle X, Y \rangle := \int_\Omega g_u(X(u), Y(u)) \mathrm{d}u$$

Intrinsic gradient

- Another way to think of (and define) vector fields is as a generalized notion of derivative. In classical calculus, one can locally linearize a smooth function through the *differential*

$$\mathrm{d}x(u) = x(u + \mathrm{d}u) - x(u),$$

  reporting the change of the value of the function x at point u after an infinitesimal displacement $\mathrm{d}u$.

- This is often impossible to generalize naively, as $u + \mathrm{d}u$ is meaningless without a group operation on the manifold.
- Tangent vectors can be used to model local infinitesimal displacement. In particular, we can operate on scalar fields through vector fields in a natural way.
- Given a smooth scalar field $x \in \mathcal{X}(\Omega, \mathbb{R})$ to be acted on by a vector field, we can actually characterize vector fields as linear maps

$$Y : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, \mathbb{R})$$

  satisfying the properties of *derivation*:

  [vanish on constants] $Y(c) = 0$ for any constant(-valued scalar field) $c \in \mathcal{X}(\Omega, \mathbb{R})$

  [linearity] $Y(x + z) = Y(x) + Y(z)$ for all $x, z \in \mathcal{X}(\Omega, \mathbb{R})$

  [product rule] $Y(xz) = Y(x)z + xY(z)$

- The relation between vector fields and the differential d is given by duality, one can think of d as a map taking some signal u to a *covector field*, namely a linear functional of a vector field. This map is defined by

$$\mathrm{d}x(Y) = Y(x)$$

- At each point u, thee differential can be regarded as a linear functional

$$\mathrm{d}x_u : T_u\Omega \to \mathbb{R}$$

  acting on tangent vectors $X \in T_u\Omega$.
- If we are given such an object $\mathrm{d}x_u$ on an inner product space, the Riesz representation theorem gives us the existence of some tangent vector "$\nabla x(u)$," with the notation here purely formal, which encodes the action of $\mathrm{d}x_u$.
- In the context of a Riemannian manifold equipped with metric tensor g, the inner product g_u gives rise to a canonical way of representing the cotangent vector $\mathrm{d}x_u$ through a tangent vector, namely $\nabla x(u)$ is the unique tangent vector satisfying

$$\mathrm{d}x_u(X) = g_u(\nabla x(u), X)$$

  for each $X \in T_u\Omega$.
- The notation here is suggestive of the definition to be made: we refer to the tangent vector $\nabla x(u) \in T_u\Omega$ as the *intrinsic gradient* of x.

- At the level of the tangent bundle, the gradient can thus be considered as an operator $\nabla : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, T\Omega)$ with mapping given by $x(u) \mapsto \nabla x(u) \in T_u\Omega$.
- Thus, the gradient of scalar field x is a vector field ∇x.

Geodesics

- Consider a smooth curve $\gamma : [0, T] \to \Omega$ on the manifold with endpoints $u = \gamma(0)$ and $v = \gamma(T)$. Among all curves joining u and v, we are interested in those of minimum length, i.e. we seek γ minimizing the following length functional:

$$\begin{aligned}\ell(\gamma) &= \int_0^T \|\gamma'(t)\|_{\gamma(t)} \mathrm{d}t \\ &= \int_0^T g^{1/2}_{\gamma(t)}(\gamma'(t), \gamma'(t)) \mathrm{d}t\end{aligned}$$

- Such curves are called *geodesics*. Such curves do not necessarily need a Riemannian metric in order to be defined, but rather a more general notion of a *connection*, or *covariant derivative*.
- Given a Riemannian metric, there is a unique “compatible" connection called the *Levi-Civita connection*, where the connection notion of geodesic for the Levi-Civita connection is identical to the metric notion first introduced.
- The significance of geodesics to the learning blueprint is that we can use these to define a way to transport tangent vectors on the manifold, which we will need to construct a convolution-like operation.

4.3 Parallel transport

- One issue we have already encountered with manifolds is that we cannot directly add or subtract points in the domain. The same problem arises when trying to compare tangent vectors at *different* tangent spaces.
- Geodesics provide a mechanism to move vectors between distinct tangent spaces: given a tangent vector $X \in T_u\Omega$ and a geodesic γ joining u to v, one can define a new set of tangent vectors $X(t) \in T_{\gamma(t)}\Omega$, namely along the geodesic γ, by holding the length of $X(t)$ constant, as well as the angle between $X(t)$ and the velocity vector $\gamma'(t)$. These are summarized as follows

  $g_{\gamma(t)}(X(t), \gamma'(t)) = g_{\gamma(0)}(X, \gamma'(0)) = \text{const.}$

  $\|X(t)\|_{\gamma(t)} = \|X\|_u = \text{const.}$

- The map we are discussing, $\Gamma_{u\to v}(X) : T_u\Omega \to T_v\Omega$, which maps $X \in T_u\Omega$ to $X(T) \in T_v\Omega$ in the above notation, is called *parallel transport* or *connection*. Due to angle and length preservation conditions, parallel transport amounts to only rotation of the vector, so it can be associated with an element $g_{u\to v}$ of the special orthogonal group SO(s) (the latter called the *structure group*)

- The result of the transport, and hence the group element $g_{u\to v} \in$ SO(s) depends on the path taken.

Exponential map, geodesic distances, isometries

- Locally around a point u, it is always possible to define a unique geodesic in a given direction, $X \in T_u\Omega$, i.e. so that $\gamma(0) = u$ and $\gamma'(0) = X$.

- When $\gamma_X(t)$ is defined for all $t \geq 0$, the manifold is said to be *geodesically complete*, and this map (to be defined, and called the *exponential map*) is defined over the whole tangent space.

- Note that geodesic completenesss does not necessarily guarantee that exp is a global diffeeomorphism; the largest radius r about u for which $\exp_u(B_r(0)) \subset T_u(\Omega)$ is mapped diffeomorphically iss called the *injectivity radius*.

- The Hopf-Rinow theorem establishes the equivalence of geodesic and metric completeness. It guarantees that for any $u, v \in M$, there is γ joining u and v such that
$$d_g(u, v) = \ell(\gamma).$$

- Consider now a deformation of manifold Ω into another manifold $\tilde{\Omega}$ with Riemannian metric h, we assume that this map
$$\eta : (\Omega, g) \to (\tilde{\Omega}, h)$$
is a diffeomorphism, so that its differential $\mathrm{d}\eta : T\Omega \to T\tilde{\Omega}$ defines what's called the *pushforward* map between tangent bundles.

- The pushforward enables us to compare the two metrics g and h via the pullback η^* defined pointwise for $u \in M$ as
$$(\eta^* h)_u(X, Y) = h_{\eta(u)}(\mathrm{d}\eta_u(X), \mathrm{d}\eta_u(Y)).$$
If the pullback metric $\eta^* h$ coincides with g at every point, the map η is called a Riemannian isometry.

- By the Hopf-Rinow theorem mentioned above, this corresponds to an isometry at the level of metric spaces. Let d_g and d_h be the (abstract) metrics induced by Riemannian metrics g and h. A Riemannian isometry between (Ω, g) and $(\tilde{\Omega}, h)$ induces an isometry between metric spaces (Ω, d_g) and (Ω, d_h), in that

$$d_g(u, v) = d_h(\eta(u), \eta(v)) \quad \text{for all } u, v \in \Omega \ ,$$

  or more concisely, $d_g = d_h \circ (\eta \times \eta)$.

- On connected manifolds, the converse of Hopf-Rinow is also true: every metric isometry is also a Riemannian isometry.

- In our geometric deep learning blueprint, η is a model of domain deformations. When η is an isometry, intrinsic quantities are unaffected by such deformations. One can generalize exact (metric) isometries through notions of *metric dilation*

$$\mathrm{dil}(\eta) = \sup_{u \neq v \in \Omega} \frac{d_h(\,\eta(u),\, \eta(v)\,)}{d_g(\,u,\, v\,)}$$

  or *metric distortion*

$$\mathrm{dis}(\eta) = \sup_{u\, v \in \Omega} \Big| d_h(\,\eta(u), \eta(v)\,) - d_g(u, v) \Big| ,$$

  which respectively capture the relative and absolute change of geodesic distance under η.

- Note that the latter notion of metric distortion has already been used implicitly: viewing η as a domain deformation, recall that a function $f \in \mathcal{F}(\mathcal{X}(\Omega))$ is stable under domain deformations if

$$\|f(x, \Omega) - (x \circ \eta^{-1}, \tilde{\Omega})\| \leq C \|x\| \mathrm{dis}(\eta)$$

4.4 Spectral methods

- Given a Riemannian manifold Ω, the collection of Riemannian self isometries is denoted $\mathrm{Iso}(\Omega)$ *(one assumes they mention this because it will be the group lurking in the background)*

- Our aim is to construct intrinsic convolution-like operations on manifolds, which by construction will be invariant to isometric deformations.

- There are two ways of approaching this in the developed frameworks:

  One is to use analogy of the Fourier transform and to define convolution as a product in the Fourier domain.

  The other is to define convolution spatially, by “correlating" a filter locally with the signal.

- We discuss the spectral approach first.

- Recall that in the (discrete) Euclidean domains, the Fourier transform can be obtained through eigenvectors of any circulant matrix, which are jointly diagonalizable due to their commutativity. The Fourier transform is then the original object in the coordinates of this orthonormal eigenbasis.

- One example of a circulant matrix in the above setting is the discrete derivative, and in the continuum, we can use a differential operator to define an analogy of the Fourier transform on general domains.

- In Riemannian geometry, it is common to use the orthonormal basis of the Laplacian operator, which we now define on manifolds.

- Recall our definition of the intrinsic gradient operator,

$$\nabla : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, T\Omega),$$

  producing a tangent vector field indicating the local direction of steepest increase of some input scalar field on the manifold.

- In a similar manner, we can define the *divergence operator* $\nabla^* : \mathcal{X}(\Omega, T\Omega) \to \mathcal{X}(\Omega, \mathbb{R})$: if we think of a tangent vector field as a flow on the manifold, the divergence measures the net flow of a field at a point. We use the notation ∇^* (as opposed to *div*) to emphasize that the two operators are adjoint:

$$\langle X, \nabla x \rangle = \langle \nabla^* X, x \rangle$$

- the **Laplacian** (or Laplace-Beltrami operator) is an operator defined on $\mathcal{X}(\Omega)$, through $\Delta = \nabla^* \nabla$, which can be interpreted as the difference between the average of a function on an infinitesimal sphere around a point and the value of the function at the point itself.

- The Laplacian is self-adjoint, or “symmetric," through the following integration-by-parts identity

$$\langle \nabla x, \nabla x \rangle = \langle x, \Delta x \rangle = \langle \Delta x, x \rangle$$

- The quadratic form on the left is called the **Dirichlet energy** of the signal x, defined more concretely as

$$\mathfrak{c}^2(x) = \|\nabla x\|^2 = \int_\Omega \|\nabla x(u)\|_u^2 \mathrm{d}u = \int_\Omega g_u(\nabla x(u), \nabla x(u)) \mathrm{d}u,$$

  which measures the smoothness or elasticity of x.

- The Laplacian operator admits an eigendecomposition

$$\Delta \varphi_k = \lambda_k \varphi_k, \quad k = 0, 1, 2, \ldots$$

  with countable spectrum if the manifold is compact (which we always assume), and with orthogonal eigenfunctions $\langle \varphi_k, \varphi_\ell \rangle = \delta_{k,\ell}$, due to the self-adjointness of Δ.

- The Laplacian eigenbasis can be constructed through a Gram-Schmidt procedure, as a set of orthogonal minimizers of the Dirichlet energy,

$$\varphi_{k+1} = \mathrm{argmin}_\varphi \|\nabla \varphi\|^2 \quad \text{s.t.} \quad \|\varphi\| = 1 \text{ and } \langle \varphi, \varphi_j \rangle = 0$$

  for $j = 0, 1, \ldots, k$, allowing for the interpretation of $(\varphi_k)_{k \geq 0}$ as the "smoothest orthogonal basis on Ω." These eigenfunctions, as well as the eigenvalues $(\lambda_k)_{k \geq 0}$, together can be interpreted as analogous to the atoms and frequencies in the classical Fourier transform.

- This orthogonal basis allows us to expand square-integrable functions on Ω into Fourier series:

$$x(u) = \sum_{k \geq 0} \langle x, \varphi_k \rangle \varphi_k(u)$$

  where $\hat{x}_k := \langle x, \varphi_k \rangle$.

- Truncating the Fourier series results in an approximation error which may be bounded ([8] cite Aflalo-Kimmel 2013) as follows:

$$\left\| x - \sum_{k=0}^{N} \langle x, \varphi_k \rangle \varphi_k \right\|^2 \leq \frac{\|\nabla x\|^2}{\lambda_{N+1}}$$

- Aflalo et al. (2015) further showed that no other basis attains a better error, making the Laplacian eigenbasis *optimal* for representing smooth signals on manifolds.

Spectral convolution on manifolds

- **Spectral convolution** can be defined as the product of the Fourier transforms of the signal x and the filter θ,

$$(x \star \theta)(u) = \sum_{k \geq 0} (\hat{x}_k \cdot \hat{\theta}_k) \varphi_k(u),$$

this uses the intertwining of convolution and multiplication as a method for defining a non-Euclidean convolution.

- In practice, a direct computation of the display directly above appears to be prohibitively expensive due to the need to diagonalize the Laplacian.

- Even worse, it turns out geometrically unstable: the higher-frequency eigenfunctions of the Laplacian can change dramatically with even small near-isometric perturbations of the domain Ω.

- A more stable solution is provided by realizing the filter as a *spectral transfer function* of the form $\hat{p}(\Delta)$,

$$\begin{aligned}(\hat{p}(\Delta)x)(u) &= \sum_{k \geq 0} \hat{p}(\lambda_k) \langle x, \varphi_k \rangle \varphi_k(u) \\ &= \int_\Omega x(v) \sum_{k \geq 0} \hat{p}(\lambda_k) \varphi_k(v) \varphi_k(u) \mathrm{d}v\end{aligned}$$

- We can interpret the above display in two ways: either as a *spectral filter*, where we identify $\hat{\theta}_k = \hat{p}(\lambda_k)$, or as a spatial filter with a position dependent kernel, whose role is played by the sum in the second line

$$\theta(u, v) = \sum_{k \geq 0} \hat{p}(\lambda_k) \varphi_k(v) \varphi_k(u)$$

- The advantage of this formulation is that $\hat{p}(\lambda)$ can be characterized by a small number of coefficients, and choosing a good parametrization of the spectral transfer functions allows for an efficient computation of the filter, avoiding the spectral decomposition.

Spatial convolution on manifolds

- An alternative attempt at defining convolution on manifolds is to “match a filter at different points," like we did in the definition of group convolution

$$(x \star \theta)(g) = \langle x, \rho(g)\theta \rangle = \int_\Omega x(u) \theta(g^{-1}u) \mathrm{d}u,$$

noting that $x \star \theta$ takes group elements g as arguments. Here we could analogously write

$$(x \star \theta)(u) = \int_{T_u\Omega} x(\exp_u Y)\theta_u(Y)\mathrm{d}Y,$$

where we have used the (yet to be defined) **exponential map** $\exp_u$ to pull signal x back to a function on $T_u\Omega$. The filter θ_u is defined directly in the tangent space at each point, and is thus position-dependent.

- If one defines the filter θ intrinsically, such a convolution would be isometry invariant, a property mentioned as crucial in graphics in vision.
- There are substantial differences from the previous construction: firstly, because a manifold is generally not a homogeneous space, we don't have a group structure that allows us to "share" the filter θ across the manifold in a straightforward way.
- An analogy of this operation on the manifold (in our current attempt to define convolution on manifold) would require a parallel transport to move the filter θ from one tangent space to another.
- In general, the result of a parallel transport depends on the path taken between two points u and v, so *the way we move the filter around matters.*
- A second difference is that we can use the exponential map only locally, so the filter must be local, with support bounded by the injectivity radius.
- Thirdly and crucially, in order for a computer to work with the abstract geometric object $X \in T_uM$, with the goal being able to compute $\theta(X)$ for some tangent space filter θ, we must express X relative to a *local basis* $\omega_u : \mathbb{R}^s \to T_u\Omega$. Thus the coordinates used by the computer are the s-dimensional array $\mathbf{x} = \omega_u^{-1}(X)$. We can thus re-express convolution locally [13]

$$(x \star \theta)(u) = \int_{[0,1]^s} x(\exp_u(\omega_u \mathbf{y}))\theta(\mathbf{y})\mathrm{d}\mathbf{y},$$

and importantly, we can construct this convolution using the same filter for each point, as all data is pulled back to the unit cube, on which the filter is defined.

- As the exponential map is intrinsic (through the definition of geodesic)

discussion

[13] And also over the same domain; is this important? Beforehand the tangent space serving as the domain of integration was u-dependent. Though these are isomorphic, should not be regarded as "the same"

- We seem to assume we can carry the frame ω_u along to another manifold, i.e. $\omega'_u = \mathrm{d}\eta_u \circ \omega_u$. Obtaining such a frame (or *gauge* in physics terminology) given only the manifold Ω in a consistent manner is difficult.

- Firstly, a smooth global gauge may not exist: thiss is the ssituation on manifolds that are not *parallelizable*, meaning there does not exist a smooth non-vanishing tangent vector field.

- Secondly, we do not have a “canonical gauge on manifolds", so this choice is arbitrary, and our convolution seemss to depend on ω.

- This is a case where practice diverges from theory: in practice, it is possible to build frames that are mostly smooth, with a limited number of singularitiess, e.g. by taking the intrinsic gradient of some intrinsically defined scalar field on the manifold.

- Moreover, such constructions are “stable", i.e. the framess constructeed this way will be identical on isometric manifolds, and similarly on approximately isosmetric ones.

- The solution is not entirely satissfactory because near singularities, the filter orientation (being defined in a fixed manner relative to the gauge) will vary wildly, leading to a non-smooth feature map even if the input signal and filter are smooth.

- Moreover, there is no clear reason why a given direction at some point u should be considered “equivalent" to another direction at an altogether different point v.

- Thus despite *practical* alternativess, we will look for a more theoretically well0funded approach that would be altogether independent of the choice of gauge.

4.5 Gauges and bundles

Fiber bundles

Definition 4.1. A **fiber bundle** is a quadruple $(E, F, B, \pi : E \to B)$, where E, F and B are topological spaces respectively called the **total space**, **fiber** and **base space**. The function π is a continuous map from the total space to the base space, such that for each $b \in B$,

$$\pi^{-1}(b) \cong F,$$

where $\cong$ means that the two topological spaces are homeomorphic. We will assume the fiber bundles we consider are “locally trivial," in that the above quadruple must

satisfy the following: for each $b \in B$, there is an open neighborhood $U \in b$ in B such that

$$\pi^{-1}(U) \cong U \times F.$$

A “trivial" fiber bundle is one such that $E \cong B \times F$.

Examples

annulus
Mobius band
Tangent bundle

Definition 4.2 (vector bundle)**.**

- In the context of geometric deep learning, fiberss are used to model the feature spaces at each point in the manifold Ω, the dimension of the fiber being the number of channels.
- In this setting, an important symmetry called **gauge symmetry** may present itself.
- Consider d-dimensional manifold Ω with its tangent bundle $T\Omega$, so that the number of channels is $s = d$ in this setting. Let $X : \Omega \to T\Omega$ be a vector field. Relative to a gauge ω for the tangent bundle, X is represented as a function $\mathbf{x} : \Omega \to \mathbb{R}^s$.
- Importantly, we are interested in the underlying geometric object X, whose representation as a function $\mathbf{x} \in \mathcal{X}(\Omega, \mathbb{R}^s)$ *depends on the choice of gauge* ω)

Tangent bundles and the structure group

- When we change the gauge, we need to apply at each point an invertible matrix that maps the old gauge to the new one.
- This matrix is unique for every pair of gauges at each point, but possibly different at different points. In other words, a *gauge transformation* is a mapping $g : \Omega \to \mathrm{GL}(s)$, where $\mathrm{GL}(s)$ is the general linear group of invertible $s \times s$ matrices. It acts on gauge $\omega_u : \mathbb{R}^s \to T_u\Omega$, to produce a new gauge

$$\omega'_u = \omega_u \circ g_u : \mathbb{R}^s \to T_u\Omega$$

- The gauge transformation acts on a coordinate vector field at each point via

$$\mathbf{x}'(u) = g_u^{-1}\mathbf{x}(u)$$

- Let us note the underlying vector field remains unchanged:

$$\begin{aligned} X(u) &= \omega'_u(\mathbf{x}'(u)) \\ &= \omega_u(g_u g_u^{-1}\mathbf{x}(u)) \\ &= \omega_u(\mathbf{x}_u) \\ &= X(u) \end{aligned}$$

- *I did not understand the next sentence starting "more generally, we may have a field of geometric quantities...* because I'm unsure of how ρ_2, ρ_1 relate to the representation ρ of GL(s) mentioned..
- Sometimes we may wish to restrict attention to frames with a certain property, such as orthogonal frames, right-handed frames, etc.. Unsurprisingly, we are interested in a group of transformations on frames which preserve these properties, respectively $O(s)$ and $SO(s)$.
- In general, we have a group G called the **structure group** of the bundle, and a **gauge transformation** is a map $g : \Omega \to G$.
- "A key observation is that in all cases with the given property, for any two frames at a given point there exists exactly one gauge transformation relating them."

Definition 4.3 (Sections over a fiber bundle)**.**

Examples

- vector field
- calc of variation like perturbations of middle circle of annulus

- Gauge theory is not specific to tangent bundles, it applies to vector bundles; one can model an "infinitely high resolution" RGB image as a section over the trivial vector bundle

$$[0,1]^2 \times \mathbb{R}^3$$

- It is customary to express an RGB image relative to a gauge that has basis vectors R, G and B, in that order. But we may equally well permute the basis vectors (color channels) independently at each position, as long as we remember the frame (order of channels).
- As a computational option this is "rather pointless", but we will see shortly it is conceptually useful to think about gauge transformations for the space of RGB colors, because it allows us to express a gauge symmetry - in this case,

an equivalence between the colors - and make functions defined on images respect this symmetry (treating each color equivalently).

- A gauge transformation is somehow “internal" to the neural network, it does not change the underlying RGB image.[a]
- In ML applications, we are interested in constructing functions $f \in \mathcal{F}(\mathcal{X}(\Omega))$ on such images, implemented as layers of a neural network.
- It follows that if, for whatever reason, we were to apply a gauge transformation to our image, we would need also to change the function f (network layers) so as to preserve their meaning.
- Consider, for simplicity, a 1×1 convolution, i.e. a map that takes an RGB pixel $\mathbf{x}(u) \in \mathbb{R}^3$ to a feature vector $\mathbf{y}(u) \in \mathbb{R}^C$.
- According to our geometric learning blueprint, the output is associated with a group representation ρ_{out}, in this case, a C-dimensional representation of the structure group $G = \mathfrak{S}_3$. Similarly, the input iss associated with a representation $\rho_{in}(g) = g$.
- If we apply a gauge transformation to the input, we would need to change the linear map 1×1 convolution $f : \mathbb{R}^3 \to \mathbb{R}^C$ into

$$f' = \rho_{out}^{-1}(g) \circ f \circ \rho_{in}(g)$$

  so that the output feature vector $\mathbf{y}(u) = f(\mathbf{x}(u)$ transforms like $\mathbf{y}'(u) = \rho_{out}(g_u)\mathbf{y}_u$
- Indeed,

$$\begin{aligned}\mathbf{y}' &= f'(\mathbf{x}') = \rho_{out}^{-1}(g) f(\rho_{in}(g)\rho_{in}^{-1}(g)\mathbf{x} \\ &= \rho_{out}^{-1}(g) f(\mathbf{x})\end{aligned}$$

[a] So maybe this is why acting on RGB basis by $\mathfrak{S}_3$ to augment is not useful, maybe even harmful.

Gauge symmetries

- To say we consider gauge transformations as symmetries is to say that any two gauges related by a gauge transformation are to be considered equivalent. [14]
- In general we can consider a group G and a collection of frames at every point u such that for any two of them there is a unique $g(u) \in G$ that maps one frame onto the other.

[14] How is this related to a Grassmannian?

- When we regard gauge transformations as symmetries in our GDL blueprint, we are interested in making functions f in the hypothesis space equivariant under gauge transformations.

- Concretely, this means that if we apply a gauge transformation to the input, the output should undergo the same transformation (perhaps acting via a different representation of G [15]).

- We noted before that when we change the gauge, the function f should be changed as well, but for a gauge equivariant map, this is not the case: consider again the RGB color space example. The map $f : \mathbb{R}^3 \to \mathbb{R}^C$ is equivariant if $f \circ \rho_{in}(g) = \rho_{out}(g) \circ f$.

- "Thus, the coordinate expression of a gauge equivariant map is independent of the gauge, in the same way that in the case of graphs, we applied the same function regardless of how the input nodes were permuted.

- However, unlike the case of graphs and other examples covered so far, gauge transformations act not on Ω, but separately on each of the feature vectors $x(u)$ by a transformation $g(u) \in G$ for each $u \in \Omega$.

- Further considerations enter the picture when we look at filters on manifolds with larger spatial support. Let us first consider the example of a mapping $f : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, \mathbb{R})$ from scalar fields to scalar fields on a d-dimensional manifold.

- Unlike vectors and other geometric quantities, scalars have no orientation, so a scalar field $x \in \mathcal{X}(\Omega, \mathbb{R})$ is invariant to gauge transformations.

- Having characterized such f in terms of convolutions, we can write f as before under the "spatial filter" interpretation of convolution, as a convolution-like operation with a position dependent filter $\theta : \Omega \times \Omega \to \mathbb{R}$,

$$(x \star \theta)(u) = \int_\Omega \theta(u, v) x(v),$$

  which implies we have a potentially different filter $\theta_u = \theta(u, \circ)$ at each point, and is synonymous with losing spatial weight sharing. I think their point is that gauge symmetry does not alone lead to weight sharing.

- Consider now a more interesting case of a mapping $f : \mathcal{X}(\Omega, T\Omega) \to \mathcal{X}(\Omega, T\Omega)$ from vector fields to vector fields. Relative to a gauge, the input and output vector fields $X, Y \in \mathcal{X}(\Omega, T\Omega)$ are vector-valued functions $\mathbf{x}, \mathbf{y} \in \mathcal{X}(\Omega, \mathbb{R}^s)$, with $s \equiv d$ in this case.

[15] so these distinct representations are the ρ_1 and ρ_2 I didn't understand above

- A general linear map between such functions can be written using the same equation used for scalars, only replacing the scalar kernel by a matrix-valued one, $\mathbf{\Theta} : \Omega \times \Omega \to \mathbb{R}^{s \times s}$.

- The matrix $\mathbf{\Theta}(u, v)$ should map tangent vectors in $T_v\Omega$ to tangent vectors in $T_u\Omega$, but these points have different gauges that we may change arbitrarily and independently. That is, the filter would have to satisfy $\mathbf{\Theta}(u, v) = \rho^{-1}(g(u))\mathbf{\Theta}(u, v)\rho(g(v))$ for all $u, v \in \Omega$. Here ρ denotes the action of G on vectors, given by an $s \times s$ rotation matrix.

- Since $g(u)$ and $g(v)$ can be chosen freely, this is an overly strong constraint on the filter, in fact it implies the filter is zero.

- A better approach is to first transport the vectors to a common tangent space by means of the connection, and then impose gauge equivariance with respect to a single gauge transformation at one point only. [16]

- A natural point to chose is the argument of the convolution, u. One can then define the following map between vector fields:

$$(\mathbf{x} \star \mathbf{\Theta})(u) = \int_\Omega \mathbf{\Theta}(u, v)\rho(g_{v \to u})\mathbf{x}(v)\mathrm{d}v,$$

  where $g_{u \to v} \in G$ denotes the parallel transport along the geodesic joining v to u.[17]

- Under a gauge transformation g_u, this element transforms as

$$g_{u \to v} \mapsto g_u^{-1} g_{u \to v} g_v,$$

  and the field itself transforms as $\mathbf{x}(v) \mapsto \rho(g_v)\mathbf{x}(v)$.

- If the filter commutes with the structure group representation, $\mathbf{\Theta}(u, v)\rho(g_u) = \rho(g_u)\mathbf{\Theta}(u, v)$, the previous equation defines a *gauge-equivariant convolution*, which transfsorms as

$$(\mathbf{x}' \star \mathbf{\Theta})(u) = \rho^{-1}(g_u)(\mathbf{x} \star \mathbf{\Theta})(u)$$

  Umm... where is the group action on the LHS? Is this a typo they fix?

4.6 Geometric graphs and meshes

Definition 4.4 (2-complex)**.**

Definition 4.5 (Triangulation)**.** face orientation

[16] This really encourages thinking locally

[17] This also assumes locality, i.e. filter with local support, in order to have a unique geodesic

- The condition “Each edge shared by exactly two triangles" (either baked into above definition, or asserted as a specialization after) ensures the 1-hop neighborhoods around each node are disk-like, and the mesh thus constitutes a discrete manifold. These are called *manifold meshes* in practice?
- We can equip these discrete objects with metrics analogous to the Riemannian setting.
- In the simplest instance, the mesh is considered embedded in $\mathbb{R}^3$ (particularly in graphics and vision) and one can define the metric structure on the mesh through the pullback ℓ of the Euclidean metric under the embedding.
- Any property which can be expressed solely in terms of ℓ is *intrinsic*.
- By analogy to graphs, let us assume a mesh with n-nodes, each associated with an s-dimenssional feature vector, which we can arrange (assuming some arbitrary ordering) into an $n \times d$ matrix $\mathbf{X}$.
- The features can represent the geometric coordinates of the nodes ass well as additional properties such as colors, normals, or when such objects are used to model molecules, we can store chemical information like atomic number at each node.

Mesh Laplacian

- We can discretize the Laplacian on a manifold mesh in the following way:

$$(\Delta \mathbf{X})_u = \sum_{v \in \mathcal{N}_u} w_{uv}(\mathbf{x}_u - \mathbf{x}_v),$$

  or in matrix notation, as an $n \times n$ symmetric matrix $\Delta = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is called the *degree matrix*, and $d_u = \sum_v w_{uv}$ is the *degree* of node u.
- The display directly above performs local permutation-invariant aggregation of neighbor features

$$\varphi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}(u)}) = d_u \mathbf{x}_u - \sum_{v \in \mathcal{N}_u} \omega_{uv} \mathbf{x}_v,$$

  so that $\mathbf{F}(\mathbf{X}) = \Delta \mathbf{X}$ is an instance of the general blueprint for constructing permutation-equivariant functions on graphs.
- Let us remark that there is nothing about the graph Laplacian defined above which is specific to meshes. Even on an unweighted graph, one can take $\mathbf{W} = \mathbf{A}$, the latter the adjacency matrix.

- Laplacian's constructed in this way are called combinatorial.

- For geometric graphs (which do not necessarily have the additional structure of meshes, but whose nodes do have spatial coordinates that induce a metric in the form of edge-lengths), it is common to use weights related to the metric, for instance $w_{uv} \propto e^{-\ell_{uv}}$.

- Meshes have an advantage in that we can exploit the additional structure afforded by the weights, and we can define the edge weights according to the *cotangent formula*

$$w_{uv} = \frac{\cot \angle_{uqv} + \cot \angle_{upv}}{2a_u}$$

  where $\angle_{uqv}$ and $\angle_{upv}$ are the two angles opposite the shared edge (u, v) within the trianglulation (making use of manifold mesh property). The factor a_u is the local area element,

$$a_u = \frac{1}{3} \sum_{v,\, q\,:\,(u,v,q) \in \mathcal{F}} a_{uvq},$$

  which is parsed as follows: the faces of the triangulation are denoted $\mathcal{F}$. The sum is taking place over all triangles which use the vertex u, thus we are effectively looking at the 1-hop neighborhood of u. Each triangle in this neighborhood has a barycenter, and the collection of these barycenters form the vertices of a polygon contained in the 1-hop neighborhood triangles. The factor a_u is the area of this polygon.

- The cotangent Laplacian can be shown to have multiple convenient properties: it is positive-semidefinite, thus it can be used for spectral analysis with its eigenvalues considered the analogue of frequencies.

- It is also local, as discussed, depending only on the 1-hop neighbors.

- Most importantly, it has the property that the cotangent Laplacian converges in an appropriate sense to the continuous Laplacian, when the mesh size tends to zero.

- The cotangent Laplacian is thus a discrete analogue of the Laplacian operator on a Riemannian manifold.

- While one expects the Laplacian to be intrinssic, this is not obvious, and it takes some effort to express the cotangent weights entirely in terms of the discrete metric ℓ as

$$w_{uv} = \frac{-\ell_{uv}^2 + \ell_{vq}^2 + \ell_{uq}^2}{8a_{uvq}} + \frac{-\ell_{uv}^2 + \ell_{vp}^2 + \ell_{up}^2}{8a_{uvp}},$$

where the area of the triangles a_{ijk} is given as

$$a_{uvq} = \sqrt{s_{uvq}(s_{uvq} - \ell_{uv})(s_{uvq} - \ell_{vq})(s_{uvq} - \ell_{uq})},$$

"using *Heron's semiperimeter formula* with $s_{uvq} = \frac{1}{2}(\ell uv + \ell_{uq} + \ell_{vq})$."

- This endows the Laplacian with *isometry invariance*. Moreover, as observed, it is invariant to the permutation of nodes in $\mathcal{N}_u$, as it involves aggregation in the form of summation.
- While on general graphs, this is a necessary evil due to the lack of a canonical ordering of neighbors, on meshes we can order the 1-hop neighbors according to some orientation, and the only ambiguity is the selection of the first node.
- This, iinstead of any possible permutation we need to account for *cyclic shifts*, or rotations, which intuitively corresponds to the ambiguity arising from SO(2) gauge transformations.

Spectral analysis on meshes

- The orthogonal eigenvectors $\Phi = (\varphi_1, \ldots, \varphi_n)$ diagonalizing the Laplacian matrix, i.e.

  $$\Delta = \Phi \Lambda \Phi^T$$

  where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, are used as the non-Euclidean analogy of the Fourier basis.
- We can design a filter θ directly in the Fourier domain, denoted $\hat{\theta}$, and one can perform spectral convolution on the mesh on the Fourier side via multiplication

  $$X \star \theta = \Phi \text{diag}(\Phi^T \theta)(\Phi^T X) = \Phi \text{diag}(\hat{\theta})\hat{X}$$

- It is tempting to exploit this spectral definition of convolution to generalize CNNs to graphs, which was done by Bruna et al in 2013.
- However, it appears that the non-Euclidean Fourier transform is extremely sensitive to even minor perturbations of the underlying mesh or graph.
- Thus it is best suited to settings in which the domain is fixed, and not changing from signal to signal. Unluckily, this makes the Fourier transform based approach inappropriate in graphics and vision settings, in which one trains a neural network on one set of $3D$ shapes and tests on a different set.

- As noted previously, it is preferable to use spectral filters dfined through some transfer function $\hat{p}$ acting on the Laplacian.

$$\hat{p}(\Delta)X = \Phi\hat{p}(\Lambda)\Phi^T X = \Phi\text{diag}(\hat{p}(\lambda_1), \dots, \hat{p}(\lambda_n))\hat{X}$$

- When $\hat{p}$ can be expressed in terms of matrix-vector products, the eigendecomposition of the $n \times n$ matrix Δ can be avoided altogether. For example, one can use polynomials of degree r as filter functions (Defferrard et al. 2016)

$$\hat{p}(\Delta)X = \sum_{k=0}^{r} \alpha_k \Delta^k X$$

- This amounts to multiplication of the $n \times s$ feature matrix X by the $n \times n$ Laplacian matrix r times. Since the Laplacian is typically sparse, with $O(\#\mathcal{E})$ non-zero elements, this operation hass low complexity of $O(\#\mathcal{E})$.

- *Furthermore, since the Laplacian is local, a polynomial filter of degree r is localized in r-hop neighborhood.*

- Note that this means that the support of the filter depends on the resolution of the mesh, so that an equivalent support on a finer mesh requires using larger r.

- For this reason, in computer graphs, it is more common to use *rational filters*, since they are resulution-independent. There are many ways to define such filters (see Patane 2020), the most common being as a polynomial of some rational function, e.g. $\frac{\lambda-1}{\lambda+1}$. More generally, one can use a complex function, such as the *Cayley transform* $\frac{\lambda-\mathrm{i}}{\lambda+\mathrm{i}}$ that maps the real line to the unit circle in the complex plane. Both of these functions are Mobius transformationss.

- Levie at al (2018) used spectral fiilters expressed as Cayley polynomials, real rational functions with complex coefficients $\alpha_\ell \in \mathbb{C}$.

$$\hat{p}(\lambda) = \text{Re}\left(\sum_{\ell=0}^{r} \alpha_\ell \left(\frac{\lambda - \mathrm{i}}{\lambda + \mathrm{i}}\right)^\ell\right)$$

- When applied to matrices, computation of the Cayley polynomial requires matrix inversion

$$\hat{p}(\Delta) = \text{Re}\left(\sum_{\ell=0}^{r} \alpha_\ell (\Delta - \mathrm{i}I)^\ell (\Delta - \mathrm{i}I)^{-\ell}\right),$$

  which can be carried out approximately with linear complexity.

- Unlike polynomial filters, rational filters do not have local support, but do have exponential decay.

- A crucial difference compared to the direct computation of the Fourier transform is that polynomial and rational filters are stable under approximate isometric deformations of the underlying graph or mesh. [18]

Meshes as operators and functional maps

- The paradigm of functional maps suggests thinking of meshes as operators. As we will show, this allows us to obtain more interesting types of invariance by exploiting the additional structure of meshes.
- For the purpose of the discussion, assume the mesh $\mathcal{T}$ is constructed upon embedded nodes with coordinates X.
- If we construct an intrinsic operator like the Laplacian, it can be shown it encodes completely the structure of the mesh, and one can recover the mesh (up to its isometric embedding) [19]
- Thus we will assume a general operator, or $n \times n$ matrix $Q(\mathcal{T}, X)$ is a representation of our mesh.
- Learning functions of the form $f(X, \mathcal{T})$ can be rephrased as learning functions of the form $f(Q)$.
- Similar to graphs and sets, the nodes of meshes also have no canonical ordering, i.e. functions on meshes must satisfy the permutation invariance or equivariance conditions,

$$f(Q) = f(PQP^T)$$
$$PF(Q) = F(PQP^T)$$

  or any permutation matrix P.
- However, compared to general graphs we now have more structure: we may assume our mesh arises from the discretization of some underlying continuous surface Ω.
- It is thus possible to have a different mesh $\mathcal{T}' = (\mathcal{V}', \mathcal{E}', \mathcal{F}')$ with n' nodes and coordinates X', but representing the same object Ω as $\mathcal{T}$.
- Importantly, the meshes $\mathcal{T}$ and $\mathcal{T}'$ can have a different connectivity structure and even different number of nodes, $(n' \neq n)$, in particular they are in general not isomorphic as 2-complexes.

[18] In signal processing, polynomial filters are termed *finite impulse response* (FIR) whereas rational filters are *infinite impulse response* (IIR)

[19] Is this more elegantly expressed functorally?

- Functional maps were introduced by Ovsjanikov et al. (2012) as a generalization of the notion of correspondence to such settings, replacing the correspondence between points on the two domains (i.e. a map $\eta : \Omega \to \Omega'$) with correspondence between functions, i.e. a map $C : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega')$.

- A **functional map** is a linear operator C, represented as an $n' \times n$ matrix, establishing a correspondence between signals x' and x on the respective domains as
$$x' = Cx$$

- Rustamov et al. (2013) showed that in order to guarantee *area-preserving* mapping, the functional map must be orthogonal, $C^T C = I$, in particular $C \in O(n)$ and C is invertible.

- The functional map establishes a relation between the operator representation of meshes,
$$Q' = CQC^T, \quad Q = C^T Q' C$$
which we can reinterpret as follows: given an operator representation Q of $\mathcal{T}$ and a functional map C, we can construct its representaiton Q' of $\mathcal{T}'$ by first mapping the signal from $\mathcal{T}'$ to $\mathcal{T}$, using C^T, then applying the operator Q, and then mapping the signal back to $\mathcal{T}'$, using C.

- This leads to a more general class of remeshing invariant (or equivariant) functions on meshes, satissfying
$$f(Q) = f(CQC^T) = f(Q') \qquad CF(Q) = F(CQC^T) = F(Q')$$
for any $C \in O(n)$.

- The previous setting of permutation invariance an equivariance is a particular case, which can be thought of as a trivial remeshing in which only the order of the nodes is changed.

- Wang wet al (2019a) showed that given an eigendecomposition of the operator $Q = V\Lambda V^T$, any remeshing invariant or equivariant function can be expressed as $f(Q) = f(\Lambda)$ and $F(Q) = VF(\Lambda)$.

- In other words, remeshing-invariant functions involve only the spectrum of Q.

- Indeed, functions of Laplacian eigenvalues have been proven in practice to be robust to surface discretization and perturbation.

5 Learning in high dimensions

- The simplest formalization of supervised learning considers a set of N observations
$$\mathcal{D} = \{ (x_i, y_i) \}_{i=1}^{N},$$
assumed to be drawn in an i.i.d. fashion from an underlying data distribution $\mathbb{P}_{\text{data}}$, which is defined over $\mathcal{X} \times \mathcal{Y}$. Here $\mathcal{X}$ and $\mathcal{Y}$ are respectively the data and the label domains.

- Importantly, $\mathcal{X}$ is a high-dimensional space; one typically assumes $\mathcal{X} = \mathbb{R}^d$ for some large dimension d.

- We further assume the labels y are generated by an *unknown* function f, so that $y_i = f(x_i)$ for each $i = 1, \dots, N$.

- The learning problem is then estimation of the function f. This is typically done through a parametrized class of functions, or *hypothesis space*,
$$\mathcal{F} = \{ f_\theta : \theta \in \Theta \}$$
Neural networks (NNs) are examples of such parametric function classes, in which case $\theta \in \Theta$ is a vector network weights.

- In this idealized setup, labels have no noise. Modern deep learning systems typically operate in this so-called *interpolating regime*, where the estimated $f \in \mathcal{F}$ satisfy $\tilde{f}(x_i) = f(x_i)$ for all $i = 1, \dots, N$ (so “interpolating regime" seems to mean that all training data must be correctly classified).

- In an ideal world, the performance of a given $f_\theta \in \mathcal{F}$ would be measured in terms of the expected performance on new samples drawn from $\mathbb{P}_{\text{true}}$, given a good choice of *loss* $L(\cdot, \cdot)$
$$\mathcal{L}(f_\theta) \triangleq \mathbb{E}_{\text{true}}\, L(\, f_\theta(x), f(x)\,),$$
in practice, this expectation (governing the random variable x) is approximated empirically.

- A successful learning scheme thus needs to encode the appropriate notion of regularity, or *inductive bias* for f, imposed through the construction of the function class $\mathcal{F}$ and the use of *regularisation*.

5.1 Inductive bias via function regularity

- Even very simple choices of NN architecture yield dense hypothesis spaces, the subject of various *universal approximation theorems*.

- Universal approximation does not imply an *absence* of inductive bias. Given a hypothesis space $\mathcal{F}$ with universal approximation, consider a functional $\mathfrak{c} : \mathcal{F} \to \mathbb{R}_+$ quantifying an abstract notion of *function complexity* (Here, complexity is not used in the sense of spin glasses). Regularizing the given learning problem with respect to the functional $\mathfrak{c}$ amounts to seeking functions $f_* \in \mathcal{F}$ such that

$$f_* \in \operatorname{argmin}_{g \in \mathcal{F}} \mathfrak{c}(g),$$

  subject to the "interpolating regime" constraint that $f_*(x_i) = f(x_i)$ for all $i = 1, \dots, N$.

- For most function spaces, $\mathfrak{c}$ can be defined as a norm.

- In the context of neural networks, $\mathfrak{c}$ factors through the parametrization, in that

$$\mathfrak{c}(f_\theta) \equiv \mathfrak{c}(\theta)$$

  The ℓ_2-norm of the network weights, called *weight decay*, as well as the so called *path-norm* are popular choices in deep learning literature.

- From a Bayesian perspective, such $\mathfrak{c}$ may be interpreted as the negative log of the prior for the function of interest. [20]

- The regularizing functional $\mathfrak{c}$ can be enforced explicitly, by incorporating $\mathfrak{c}$ into the loss, or implicitly, as the result of an optimization scheme compatible with $\mathfrak{c}$. For instance, it is considered well-known that gradient-descent on an underdetermined least-squares objective will choose interpolating solutions with minimal ℓ_2-norm.

- All in all, a natural question arises: how are we to define effective priors that capture the expected regularities and complexities of real-world prediction tasks?

5.2 The curse of dimensionality

- Consider a classical notion of regularity: 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$, i.e. those satisfying $|f(x) - f(x')| \leq \|x - x'\|$ for all $x, x' \in \mathcal{X}$.

- If our only knowledge of the target function f is that it is 1-Lipschtiz, how many observations do we expect to require to ensure our estimate $\tilde{f}$ will be close to f? The general answer is necessarily exponential in the dimension d of $\mathcal{X}$.

[20] expand

- The situation is not better if one replaces the Lipschitz prior by, for instance, the Sobolev class $\mathcal{H}^s(\mathcal{X})$. There is an established minimax rate of approximation and learning for the Sobolev class of the order $\epsilon^{-d/s}$. This shows additional smoothness assumptions on f can only improve the statistical picture when $s \propto d$, an unrealistic assumption.

- The point of these examples is that we can't escape the curse of dimensionality if our only priors on the hypothesis space come from these classical notions of regularity.

- Fully-connected NNs define function spaces with more flexible notions of rigidity, due to the structure of the underlying computation graph. For instance, one can choose a regularization promoting sparsity in the weight vector θ. This regularization allows these architectures to break this curse of dimensionality, but at the expense of making strong assumptions on the nature of the target function f. For instance, that f depends on a collection of low-dimensional projections of the input.

- In most real-world applications, functions of interest tend to exhibit complex long-range correlations unable to be expressed with low-dimensional projections.

- It is thus necessary to look for alternative source of regularity.

5.3 Geometric priors

- There is hope against the curse of dimensionality for physically-structured data, where one can employ fundamental principles of *symmetry* and *scale separation*.

- The symmetry considered respects the structure of the *domain* Ω of input signals.

- To be clear, we assume our machine learning system operates on **signals** (functions) on some domain Ω. The space of C-valued signals on domain Ω is

$$\mathcal{X}(\Omega, C) = \{x : \Omega \to C\}$$

  Here C a vector space, whose dimensions are called **channels**. An inner-product on $\mathcal{X}(\Omega, C)$ is defined (presumably, as mentioned, to leverage some functional analysis?) via

$$\langle x, y \rangle_{\mathcal{X}} \triangleq \int_\Omega \langle\, x(u), y(u) \,\rangle_C \, \mu(\mathrm{d}u)\,,$$

  When Ω is discrete, μ can be taken as counting measure; going forward we omit μ from the notation, writing $\mathrm{d}u$ for brevity.

- Our learning system exhibits “scale separation" when important characteristics of the signal are preserved when flattening (or informally, projecting) the signal into a representative defined over a coarser domain.
- A prior on the hypothesis space respecting the symmetry of the domain and exhibiting scale-separation is called a **geometric prior**.
- Geometric priors are built into convolutional neural networks (CNNs) in the form of convolutional, weight-sharing filters (exploiting translational symmetry) and pooling (connected to scale separation).
- The main goal of what the authors call *geometric deep learning* is to systematically extend ideas already present in CNN architecture to learning settings with other domains, such as graphs and manifolds.

5.4 Deformation stability

- The symmetry formalism introduced above captures an idealised world where we know exactly which transformations are to be considered as symmetries, and we want to respect these symmetries *exactly*. However, the real world is noisy and this model falls short in two ways.
- While simple groups provide a way to understand global symmetries of the domain Ω (and by extension, of signals on it, $\mathcal{X}(\Omega)$), they do not capture *local* symmetries well.
- Consider a video scene with several objects, each moving along its own different direction. At subsequent frames, the resulting scene will contain approximately the same semantic information, yet no global translation explains the transformation from one frame to another.
- The discussion below distinguishes between two scenarios: the first in which the domain Ω is fixed, and where signals $x \in \mathcal{X}(\Omega)$ are undergoing deformations, and the setting where the domain Ω itself may be deformed.

Stability to signal deformations

- In many applications, we know a priori that a small deformation of the signal x should not change the output of $f(x)$. It is tempting to consider such deformations as symmetries. For instance, we could view small *(with respect to Dirichlet energy, for instance)* diffeomorphisms $\tau \in \mathrm{Diff}(\Omega)$, or even small *(localized?)* bijections as symmetries.

- "Small" deformations can however be composed to form "large" deformations, so small deformations do not form a group.
- Large deformations can materially change the semantic content of the input, and so it is not a good idea to use the full group $\mathrm{Diff}(\Omega)$ as the symmetry group used to build a geometric prior.
- A better approach is to quantify how far a given $\tau \in \mathrm{Diff}(\Omega)$, by using a "complexity measure" $\mathfrak{c}(\tau)$ such that $\mathfrak{c}(\tau) = 0$ whenever $\tau \in G$.
- Examining our previous definition of exact invariance and equivariance under group actions, observe that we may relax (or generalize) these notions using the notion of *deformation stability* (or *approximate invariance*):

$$\|f(\rho(\tau)x) - f(x)\| \leq C\mathfrak{c}(\tau)\|x\|, \quad \text{for all } x \in \mathcal{X}(\Omega) \tag{5.1}$$

  where $\rho(\tau)x(u) = x(\tau^{-1}u)$ as before, and where C is a universal constant. A function $f \in \mathcal{F}(\mathcal{X}(\Omega))$ satisfying (5.1) is said to be ($\mathfrak{c}$-)**geometrically stable**.
- Since $\mathfrak{c}(\tau) = 0$ for $\tau \in G$, this definition generalises the G-invariance property defined above. Its utility in applications depends on introducing an appropriate deformation cost.
- In the case of images defined over a continuous Euclidean plane, a popular choice is the *Dirichlet energy* $\mathfrak{c}_{\text{Dirichlet}}$,

$$\mathfrak{c}_{\text{Dirichlet}}(\tau) \triangleq \left(\int_\Omega \|\nabla\tau(u)\|^2 \mathrm{d}u\right)^{1/2}$$

  which measures the "elasticity" of τ.

Stability to domain deformations

- Often the object being deformed is not the signal, but the underlying domain Ω itself.
- Letting $\mathcal{D}$ denote the space of all possible variable domains [21]. We suppose one can define for $\Omega, \tilde{\Omega} \in \mathcal{D}$ an appropriate metric $\mathfrak{d}(\Omega, \tilde{\Omega})$ satisfying $\mathfrak{d}(\Omega, \tilde{\Omega}) = 0$ if Ω and $\tilde{\Omega}$ are equivalent in some sense.
- For example, the *graph edit distance* vanishes when graphs are isomorphic, and the Gromov-Hausdorff distance between Riemannian manifolds [22] vanishes when two manifolds are isometric.

[21] (such as the space of all graphs, or the space of Riemannian manifolds)
[22] viewed as metric spaces equipped with geodesic distances

- A common construction of such distances between domains relies on some group $G = (\eta)_{\eta \in G}$ of invertible mappings $\eta : \Omega \to \tilde{\Omega}$ which can be used to to "align" the metric structure of two domains.

- One can then compare the distance or adjacency structures of Ω and $\tilde{\Omega}$ (which are denoted d and $\tilde{d}$ respectively), and in particular one can define a distance between the metric spaces (Ω, d) and $(\tilde{\Omega}, \tilde{d})$ by

$$\mathfrak{d}(\Omega, \tilde{\Omega}) = \inf_{\eta \in G} \left\| d - \tilde{d} \circ (\eta \times \eta) \right\|_{\Omega \times \Omega} \tag{5.2}$$

  To be clear, for fixed η, we "clone" this to a map $\eta \times \eta$ on pairs of points in Ω. The function $\tilde{d} \circ (\eta \times \eta)$ is a real-valued function on $\Omega \times \Omega$ which reports the $\tilde{d}$-distance between the η-images of any pair in $\Omega \times \Omega$. Likewise, d is a function on $\Omega \times \Omega$. The norm in the display above reports the (sup?) norm of the difference between these two functions.

- Slightly misusing notation, we write $\mathcal{X}(\mathcal{D}) = \{(\mathcal{X}(\Omega), \Omega) : \Omega \in \mathcal{D}\}$ for the ensemble of possible input signals defined over a varying domain. A function $f : \mathcal{X}(\mathcal{D}) \to \mathcal{Y}$ is **stable to domain deformations** if

$$\|f(x, \Omega) - f(\tilde{x}, \tilde{\Omega})\| \leq C \|x\| \mathfrak{d}(\Omega, \tilde{\Omega})$$

  for all $\Omega, \tilde{\Omega} \in \mathcal{D}$ and $x \in \mathcal{X}(\Omega)$. [23]

- It can be shown that stability to domain deformations is a natural generalisation of stability of signal deformations, by viewing the latter in terms of deformations of the volume form. [24]

Remark 5.1 (J)**.** Is the geometric learning framework related to / enhanced by the notion of "concentration compactness," in which quotienting by a group compactifies a space of functions?

---

[23] *(Q: what about the signal $\tilde{x}$?)*

[24] Expand, reference is Gama et al. 2019

5.5 Scale separation

- Deformation stability substantially strengthens the global symmetry priors, but is not sufficient in itself to overcome the curse of dimensionality. Informally speaking, there are still “too many" functions that respect

$$\|f(\rho(\tau)x) - f(x)\| \leq C\mathfrak{c}(\tau)\|x\| \quad \text{for all } x \in \mathcal{X}(\Omega)$$

  without further structure on the prior.

- A key insight to strengthen geometric priors is to exploit the multiscale structure of physical tasks.

Multiscale representations

- The essential insight of multi-scale methods is to decompose functions defined over the domain Ω into elementary functions which are localised *both in space and frequency*.

- In the case of wavelets, this is achieved by “correlating" a translated and dilated filter (*mother wavelet*) ψ, producing a representation called a *continuous wavelet transform*

$$(W_\psi x)(u, \xi) = \xi^{-1/2} \int_{-\infty}^{\infty} \psi\left(\frac{v-u}{\xi}\right) x(v)\mathrm{d}v$$

  The translated and dilated filters are called *wavelet atoms*; their spatial position and dilation correspond to the coordinates u and ξ of the wavelet transform, usually sampled dyadically, e.g. $\xi = 2^{-j}$ and $u = 2^{-j}k$, where j is called the *scale*.

- Multi-scale signal representations capture regularity properties beyond global smoothness, and the multiscale, localised wavelet decompositions considered here are *stable* in ways that Fourier decompositions are not.

- Consider $\tau \in \mathrm{Aut}(\Omega)$ and its associated linear representation $\rho(\tau)$. When $\tau(u) = u - v$ is a shift, the operator $\rho(\tau)$ is a *shift operator*, say S_v that commutes with convolution.

- Since[25] convolution operators are diagonalised by the Fourier transform, such a (spatial) shift operator becomes (on the frequency side) a multiplicative phase:

$$\widehat{S_v x}(\xi) = \exp(-\mathrm{i}\xi v)\hat{x}(\xi)$$

  The *Fourier modulus* $f(x) = |\hat{x}|$ of the signal x is then a shift-invariant function, $f(S_v x) = f(x)$, which we use as a case-study in stability (of Fourier versus multi-scale representations).

[25] I want to try to say this in my own words

- Suppose however we have only an *approximate* translation, $\tau(u) = u - \tilde{\tau}(u)$, with $\|\nabla\tau\|_\infty \leq \epsilon$, one can show[26]

$$\frac{\| f(\rho(\tau)x) - f(x) \|}{\|x\|} = O(1),$$

  regardless of how small ϵ is, the "error" here, by which I mean $\|f(\rho(\tau)) - f(x)\|$, is proportional to the magnitude of the signal x.

- In contrast, one can show the wavelet decomposition $W_\psi x$ is *approximately equivariant* to deformations, in that

$$\frac{\|\rho(\tau)W_\psi(x)\|}{\|x\|} = O(\epsilon)$$

- In other words, decomposing the signal information into scales using localized filters rather than frequencies turns a global unstable representation of the signal into a family of locally stable features. [27]

- Importantly, such measurements at different scales are not yet invariant, and need to be progressively processed towards the low frequencies, hinting at the "deep" compositional nature of modern neural networks.

Scale separation prior

- Consider a multiscale coarsening of the data domain Ω into a heirarchy $\Omega_1, \dots, \Omega_J$. This can be done in a fairly natural way when working with a metric space.

- The "renormalization structure" $(\Omega_j)_{j=1}^J$ induces a sequence of signal-spaces

$$\mathcal{X}_j(\Omega_j, C_j) \triangleq \{x_j : \Omega_j \to C_j\}$$

Remark 5.2 (J). I find it important to point out that the channels are allowed to *change*. This is something I was not expecting when looking at a real-life convolutional net.

---

[26] It would be nice to write this computation out

[27] It seems that what they are also doing in some sense is decomposing the "global" translation symmetries themselves into a family of local symmetries.

- A function $f : \mathcal{X}(\Omega) \to \mathcal{Y}$ is *locally stable at scale* j if it admits a factorization of the form $f \approx f_j \circ P_j$, where
$$P_j : \mathcal{X}(\Omega) \to \mathcal{X}_j(\Omega_j)$$
is a non-linear *coarse-graining* and $f_j : \mathcal{X}(\Omega_j) \to \mathcal{Y}$ is a function “adapted" to the scale j.

- Effectively, this framework is based in the ansatz that while the target function f may depend on complex long-range interactions between features over the whole domain, in locally-stable functions it is possible to *separate* the interactions across scales, by first focusing on localized interactions that are then propagated towards the coarse scales, by first focusing on localized interactions that are then propagated towards the coarse scales.

- Such principles are of fundamental importance in many areas of physics and mathematics, through the renormalisation group, or leveraged in important numerical algorithms like the *Fast Multipole Method*, the latter developed to speed calculation of long-ranged forces in n-body problems.

- In ML, multi-scale representations and local invariance are the fundamental mathematical principles underpinning the efficiency of CNNs and Graph NNs, typically implemented in the form of *local pooling*

5.6 The blueprint of geometric deep learning

- One can combine principles of symmetry, geometric stability and scale separation into a blueprint for learning stable “representations" (in a non-algebraic sense) of high-dimensional data.

- These “representations" are functions f operating on signals $\mathcal{X}(\Omega, C)$ over the domain Ω, the latter acted on by a symmetry group G.

- Observe that expressivity, in this context, requires use of a non-linear object: if f is linear and G-invariant, then for all $x \in \mathcal{X}(\Omega)$,
$$f(x) = \frac{1}{\mu(G)} \int_G f(g.x)\mu(\mathrm{d}g) \equiv_{\text{(linearity)}} f\left(\frac{1}{\mu(G)} \int (g.x)\mu(\mathrm{d}g)\right)$$
where μ denotes Haar measure on G.

- In the context of images acted upon by a translation group, our expressivity would be limited to functions on the average RGB vector for each input, were we to restrict ourselves to linear components of our compositional model.

- The above reasoning shows the family of *linear invariants* is not very rich. The family of *linear equivariants* provides a much more powerful tool, as it enables the construction of “rich" and stable features, *by composition with appropriate non-linear maps*.
- Let $B : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega, C')$ is G-equivariant satisfying $B(g.x) = g.B(x)$ for all $x \in \mathcal{X}$ and $g \in G$. Consider also $s : C' \to C''$ an arbitrary non-linear map. The composition
$$U := \mathbf{s} \circ B : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega, C'')$$
is also G-equivariant.
- Above, $\mathbf{s} : \mathcal{X}(\Omega, C') \to \mathcal{X}(\Omega, C'')$ is the *element-wise instantiation of* s, defined by
$$(\mathbf{s}(x))(u) \triangleq s(x(u))$$
- A natural question is whether any G-invariant function can be approximated at arbitrary precision by such a model, for appropriate choices of B and σ.
- One can adapt standard universal approx. theorem arguments to show shallow “geometric" networks are also universal approximators, “by properly generalizing the group average to a general non-linear invariant."
- A single layer U cannot approximate functions with long-range interactions, but a composition of several such layers, $U_J \circ \cdots \circ U_1$ increases the *receptive field* while preserving the "stability properties" of local equivariants.

Remark 5.3. Is stability a mechanism for generalisation?

The receptive field is further increased by interleaving downsampling operators that coarsen the domain.

A Appendix

A.1 On sets and categories

We do not distinguish between a **set** [28] and the more general notion of a **class**. Categories are general enough that one should consider “classes," not sets, of objects and morphisms. This is implicitly (through the definition of a graph) ignored below.

There are adjectives like *small* and *locally-small* that one can prepend, seemingly to avoid set-theoretic considerations. The former means that the objects in the category are genuinely a set, the latter means that the collection of morphisms (or arrows) between any two objects are genuinely a set. These subtleties are lost on me, for now, so I will proceed naively and will avoid these terms if possible.

The first example given in [14] is the *metacategory of sets*: the objects are the class of all sets, and the arrows are the class of all functions. He later distinguishes it from the following object:

Example 6. **0** is the empty category (no objects, no arrows).

1 is the category with one object and one (identity) arrow.

2 is the category with two objects a, b and just one arrow $a \to b$ not the identity.

Sets A category is **discrete** when every arrow is an identity, and is thus determined by the underling set of objects; we can thus view sets and discrete categories as the same notion.

Monoids A **monoid** is a category with one object. It is thus determined by the collection of arrows from the object to itself, which obey the composition axiom, and one of which is the identity. A monoid is the same notion as a *semigroup* with identity.

Groups A **group** is a category with one object, in which every arrow has a (two-sided) inverse under composition; an arrow $e : a \to b$ is **invertible** if there is an arrow $e' : b \to a$ such that $e' \circ e = \mathrm{id}_a$ and $e \circ e' = \mathrm{id}_b$, in which case we write e^{-1} for e.

Matrices The set of all rectangular matrices $\mathbf{Matr}_{\mathbb{R}}$ with real-valued entries forms a category. The objects O are the positive integers, $O \equiv \mathbb{N} = 1, 2, \ldots$, and each $m \times n$ matrix A is regarded as an arrow $A : n \to m$. The composition law is

[28] sets will not be defined rigorously, we think of them as “a collection of things."

matrix multiplication, and so invertible arrows are precisely invertible matrices. We can replace the real numbers $\mathbb{R}$ in this example with the complex numbers $\mathbb{C}$, or with any commutative ring K.

Definition A.1. The category of sets is denoted **Set**, and it is defined as follows. We assume there is a big enough set U, whose elements are sets, and which we call the "universe," and we describe a set x as small if it is a member of the universe. We then take **Set** to be the category whose set U of objects is the set of small sets, and whose arrows are all possible functions between small sets.

The above definition is a special case of the following.

Definition A.2. Given a set of sets, V, we take $\mathbf{Ens}_V$ to be the category with objects all sets $X \in V$, and arrows all functions $f : X \to Y$, with the usual notion of composition. We may write **Ens** to denote a general category of this form, for some unspecified V.

Thus, $\mathbf{Set} = \mathbf{Ens}_U$ for some implicit *universe* U. Before considering the category **Grp** of groups, we give a category-free definition of these objects.

Definition A.3. A set G, equipped with binary operation $\diamond : G \times G \to G$ called *composition*, is a **group** if $(G, \diamond)$ satisfies the following (Below, we write $g \diamond h$ more concisely as gh):

(Associativity) : $(g_1 g_2)g_3 = g_1(g_2 g_3)$ for all $g_1, g_2, g_3 \in G$.

(Identity) : there is a unique element $e \in G$ such that $eg = ge = g$ for all $g \in G$.

(Inverse) : for each $g \in G$, there is a unique inverse $g^{-1} \in G$ such that $gg^{-1} = g^{-1}g = e$

(Closure) : the group is closed under the composition operation $\diamond$.

Example 7 (*Translations*)**.** The four groups below encode some kind of one-dimensional translational symmetry, in the first two cases, the binary operation is the usual notion of addition. In the latter two, it is addition modulo n and N respectively.

(i) The real numbers $\mathbb{R}$.

(ii) The subgroup of integers $\mathbb{Z}$.

(iii) The cyclic group

$$C_n := \mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, \ldots, n-1\}$$

with binary operation given by addition modulo n.

(iv) The circle of circumference N,

$$N \cdot S_1 := \mathbb{R}/N \cdot \mathbb{R} = [0, N).$$

Example 8 (*Permutations*)**.** Let $[n] := \{1, \ldots, n\}$, and define the **symmetric group** on n symbols $\mathfrak{S}_n$ to be the collection of bijections $[n] \to [n]$, with group operation $\circ$ the usual function composition.

Example 9 (*Products of groups*)**.** The cartesian product $G \times H$ of two groups has a natural group structure; the composition law is defined coordinate-wise through those of G and H. This is the **direct product** $G \times H$.

(i) The group $\mathbb{R}^d$ (d-dimensional Euclidean space) is the d-fold direct product of $\mathbb{R}$ with itself. Likewise, the group $\mathbb{Z}^d$ (the d-dimensional integer lattice) is the d-fold product of $\mathbb{Z}$ with itself.

(ii) $N \cdot (S_1)^{\times d}$ and $\mathbb{T}_n^d := C_n^{\times d}$ are respectively continuous and discrete tori.

Example 10 (*Linear groups*)**.** We write $\mathrm{GL}(n)$ to denote the **general linear group**, which is the collection of $n \times n$ matrices with real entries. We write $\mathrm{GL}_{\mathbb{C}}(n)$ for the analogous object over $\mathbb{C}$, and if V is a vector space, we write $\mathrm{GL}(V)$ for the group of (vector space) automorphisms from V to itself. Important subgroups of $\mathrm{GL}(n)$ are the **orthogonal group** $O(n)$ and **special orthogonal group** $SO(n)$. An important subgroup of $\mathrm{GL}_{\mathbb{C}}(n)$ is the **unitary group**.

The category **Grp** has objects $\mathcal{O}$ consisting of all groups, while morphisms or arrows $\mathcal{A}$ between two groups G and H consist of the set of group homomorphisms between G and H.

Definition A.4 (Group homomorphism)**.** A group homomorphism between two groups $(G, \diamond_G)$ and $(H, \diamond_H)$ is a function $\varphi : G \to H$ such that for all $g, g' \in G$, one has

$$\varphi(g \diamond_G g') = \varphi(g) \diamond_H \varphi(g')$$

Functors

Functors are a categorical generalization of group representations. Below, we will use the term **morphism** interchangeably with "arrow." The collection of categories **Cat** itself constitutes a category, whose morphisms are functors.

Definition A.5 ([14])**.** Let C and $\mathcal{B}$ be categories. A **functor** $T : C \to \mathcal{B}$ is a morphism between categories, and consists of two suitably related functions: the *object function* T, which assigns to each object c an object Tc of $\mathcal{B}$, and the *arrow function* (also written T) which assigns to each arrow $f : c \to c'$ of C an arrow $Tf : Tc \to Tc'$ of $\mathcal{B}$, such that

$T(\mathrm{id}_c) = \mathrm{id}_{Tc}$, and

$T(g \circ f) = Tg \circ Tf$, whenever (g, f) is a composable pair.

We now state three equivalent formulations of The following holds with $\mathbb{R}$ replaced by any field.

Lemma A.6. *Consider a group G and an $\mathbb{R}$-vector space V. The following data are equivalent:*

- *A representation of G on V, as in Definition 1.5.*
- *A group homomorphism $G \to GL(V)$.*
- *A covariant functor from G (considered as a one-object category) to the category of $\mathbb{R}$-vector spaces, $\mathbb{R}$-**Mod**.*

Functors and equivariance

Remark A.7. Given an $\mathbb{R}$-linear representation of G on vector space V, the map from $V \to V$ given by $v \mapsto g.v$ is invertible for any $g \in G$; this follows from the definition.

In particular, *(ii)* of Definition 1.5 implies that the map from $G \to \mathrm{GL}(V)$ defined by

$$g \mapsto (v \mapsto g.v)$$

is a group homomorphism. *"So by using a currying[29] argument, we have seen every group representation gives rise to a group homomorphism $G \to GL(V)$. Conversely, given a group homomorphism $\rho : G \to GL(V)$, we can uncurry to get a representation on V by setting $\varphi(g, v) := \rho(g)v$."*

The above remark demonstrates that the notion of an $\mathbb{R}$-linear representation of G on V is the same as the notion of a group homomorphism $G \to \mathrm{GL}(V)$.

[29] From Wikipedia: In mathematics and computer science, currying is the technique of converting a function that takes multiple arguments into a sequence of functions that each takes a single argument. Expand

References

[1] R. J. Adler and J. E. Taylor. *Random fields and geometry*. Springer Monographs in Mathematics. Springer, New York, 2007.

[2] G. Akemann, J. Baik, and P. Di Francesco, editors. *The Oxford handbook of random matrix theory*. Oxford University Press, Oxford, 2015. Paperback edition of the 2011 original [MR2920518].

[3] L.-P. Arguin, O. Zindy, et al. Poisson-dirichlet statistics for the extremes of the two-dimensional discrete gaussian free field. *Electronic Journal of Probability*, 20, 2015.

[4] A. Auffinger, G. Ben Arous, and J. Černý. Random matrices and complexity of spin glasses. *Comm. Pure Appl. Math.*, 66(2):165–201, 2013.

[5] E. Bolthausen and A.-S. Sznitman. On Ruelle's probability cascades and an abstract cavity method. *Comm. Math. Phys.*, 197(2):247–276, 1998.

[6] S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities*. Oxford University Press, Oxford, 2013. A nonasymptotic theory of independence, With a foreword by Michel Ledoux.

[7] A. Bovier. *Statistical mechanics of disordered systems*, volume 18 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2006. A mathematical perspective.

[8] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 04 2021.

[9] P. Carmona and Y. Hu. On the partition function of a directed polymer in a Gaussian random environment. *Probab. Theory Related Fields*, 124(3):431–457, 2002.

[10] O. Daviaud et al. Extremes of the discrete two-dimensional gaussian free field. *the Annals of Probability*, 34(3):962–986, 2006.

[11] A. Dembo and O. Zeitouni. *Large deviations techniques and applications*, volume 38 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, second edition, 1998.

[12] B. Derrida. Dynamical phase transitions and spin glasses. *Physics Reports*, 184(2):207 – 212, 1989.

[13] F. Guerra. Broken replica symmetry bounds in the mean field spin glass model. *Comm. Math. Phys.*, 233(1):1–12, 2003.

[14] S. M. Lane. *Categories for the Working Mathematician*. [[Graduate Texts in Mathematics]]; Vol. 5. [[Springer-Verlag|Springer Science+Business Media]], 1971.

[15] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab. A high-bias, low-variance introduction to machine learning for physicists.

[16] M. Mézard and A. Montanari. *Information, physics, and computation*. Oxford Graduate Texts. Oxford University Press, Oxford, 2009.

[17] M. Mézard, G. Parisi, and M. A. Virasoro. *Spin glass theory and beyond*, volume 9 of *World Scientific Lecture Notes in Physics*. World Scientific Publishing Co., Inc., Teaneck, NJ, 1987.

[18] D. Panchenko. *The Sherrington-Kirkpatrick model*. Springer Monographs in Mathematics. Springer, New York, 2013.

[19] G. Parisi. The physical meaning of replica symmetry breaking.

[20] R. Podgornik. C. de dominicis and i. giardina: Random fields and spin glasses: a field theory approach. *Journal of Statistical Physics*, 129(1):191–192, Aug 2007.

[21] S. Ruder. An overview of gradient descent optimization algorithms.

[22] D. Ruelle. A mathematical reformulation of Derrida's REM and GREM. *Comm. Math. Phys.*, 108(2):225–239, 1987.

[23] D. L. Stein and C. M. Newman. *Spin glasses and complexity*. Primers in Complex Systems. Princeton University Press, Princeton, NJ, 2013.

[24] E. Subag. The complexity of spherical p-spin models—a second moment approach. *Ann. Probab.*, 45(5):3385–3450, 2017.

[25] M. Talagrand. The Parisi formula. *Ann. of Math. (2)*, 163(1):221–263, 2006.

[26] M. Talagrand. *Mean field models for spin glasses: Volume I: Basic examples*, volume 54. Springer Science & Business Media, 2010.

[27] M. Talagrand. *Mean field models for spin glasses. Volume II*, volume 55 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge. A Series of Modern Surveys in Mathematics [Results in Mathematics and Related Areas. 3rd Series. A Series of Modern Surveys in Mathematics]*. Springer, Heidelberg, 2011. Advanced replica-symmetry and low temperature.

[28] P. G. Wolynes. Evolution, energy landscapes and the paradoxes of protein folding. *Biochimie*, 119:218–230, 2015.