Chapter 11 – Geometric Deep Learning

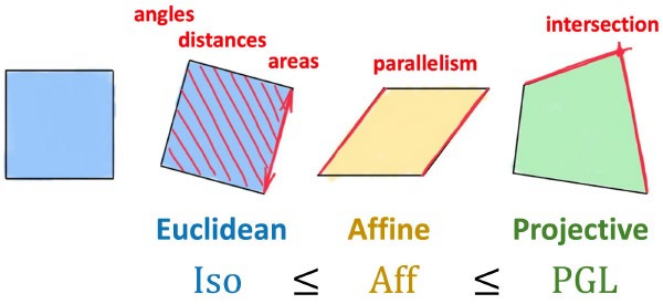
Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Introduction

Geometric Deep Learning is an advanced branch of Machine Learning which focuses on data with underlying geometric structure (graphs, manifolds, 3D shapes, etc.).

We can have multiple Geometries, like Euclidean (Distance), Affine Transformation (Parallelism) and Projective (Preserve intersection – If I transform these two lines and they still intersect in the same exact point, that's a Geometry).



This is important since we might have applications where we need to transform data through transformations and we want that our model still sees the data in the same way prior to the transformation.

1.1 Groups

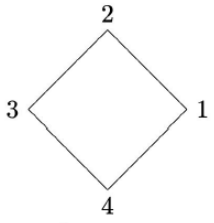
Let's recall the definition of a Group. It is a set with a (binary) operation which has the following properties:

- You take two elements of the group; you apply the operation and you'll always get an element of the group itself.
- No matter how we apply the operator, the result should be the same (associativity)
- There exists an identity e
  - $a\ x\ e = e\ x\ a = a$
- There is an inverse. Each element has its inverse:
  - $a^{-1}\ x\ a = e$

The Cayley Theorem states that any group with symmetries or permutations can be represented by a corresponding mathematical group. Essentially it says that if you have something that respects/follows a particular symmetry, you will always find a group which will represent that operation.

This is crucial because it now allows us to work with groups, offering advantages.

For instance, using the dihedral group, representing symmetries like rotations (r) and mirroring (s), simplifies operations. For example, combining rotations, like $r\ x\ r$, can be easily interpreted, such as rotating an object by 180 degrees. This is also called the *action* of a group on a set X.



1	r	r^2	r^3	s	rs	r^2s	r^3s
(1)	(1234)	$(13)(24)$	(1432)	(24)	$(12)(34)$	(13)	$(14)(23)$

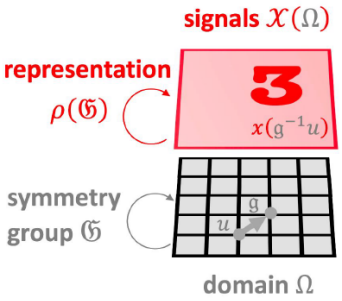
To generalize operations, like those in MLPs where we multiply $inputs\ x\ weights$, we represent groups with matrices. Specifically, we use matrices from the General Linear Group, which includes invertible matrices where determinant is also not zero. These matrices allow us to represent various transformations like rotations, translations, and reflections. **Representation** involves mapping an element from a group to a matrix, creating a representation function GL(V) -> General Linear Group. In the right image we can see the Translation Matrix which shifts each element by one position.

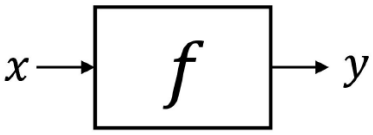
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Symmetry Prior

In many complex machine learning problems, like image classification, there's often a hidden structure called a "**symmetry prior**." For instance, in image analysis, an input image isn't just a bunch of numbers; it's a pattern on a grid. This grid has a certain structure, and we can describe its transformations using a group of symmetries, let's call it $\mathfrak{G}$.

For example, if we have an image of a cat, it's still a cat even if we shift it around (move it left, right, up, or down), so, this symmetry group helps us understand the inner patterns and relationships within the data, making our machine learning models more efficient.

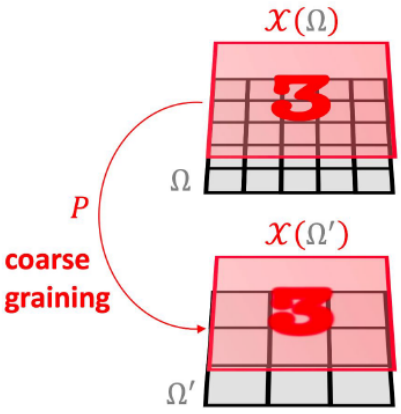


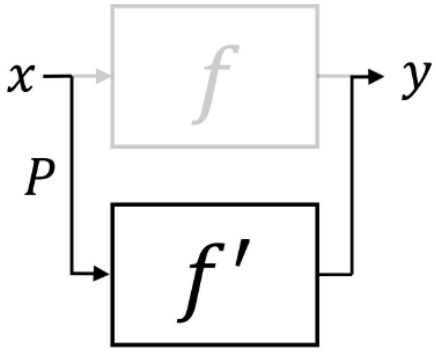


In the example on the right, the function f is the same if we take the above representation or the bottom representation because the signal is the same even after the application of the symmetry group $\mathfrak{G}$.

2.1 Coarse-Graining operator

If we have a big image $(100x100)$ with a tree, and we shrink it down to $20x20$, we still want to recognize the tree. This stability is called "**local stability**," means that if we have a function describing our image, we can use this tool to reduce its complexity (coarse-graining), and still get a pretty good approximation of our original image. It's like a way to simplify things without losing too much important information. It is given by $f = f' \cdot p$ where p is the **coarse-graining** operator.



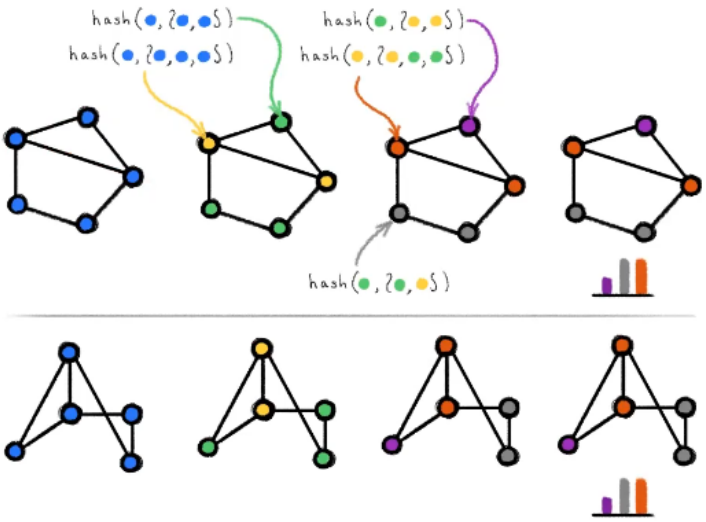


3. Weisfeiler-Lehman Test

The **Weisfeiler-Lehman Test** is a test to detect whether two Graphs are isomorphic or not.

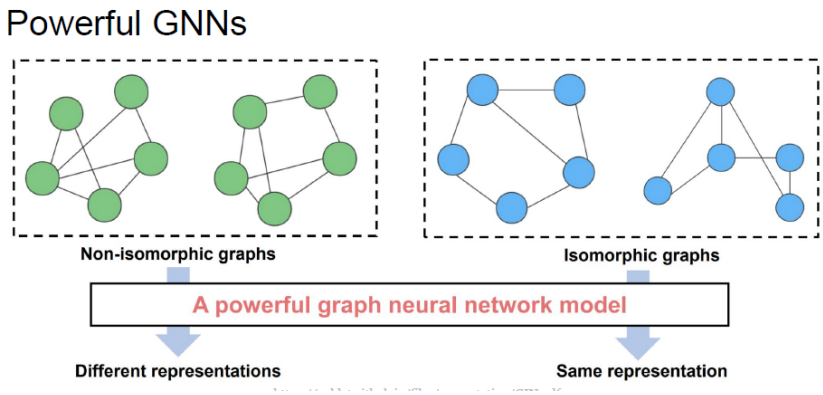
It all starts with nodes of identical colour. Then, at each step, the algorithm aggregates the colours of nodes and their neighbourhoods representing them as multisets, further hashing the aggregated colour multisets into unique new colours.

The algorithm stops upon reaching a stable colouring. If at that point the colourings of the two graphs differ, the graphs are deemed non-isomorphic. However, if the colourings are the same, the graphs are possibly (but not necessarily) isomorphic.



3.1 Powerful GNNs

A **powerful Graph Neural Network** should always get two different representations when two graphs are non-isomorphic and the same representation when the graphs are isomorphic.



-- Professor has not finished these slides since he got into Model Compression lecture, so what's coming next is just a summarized version from ChatGPT.

3.2 Equivariant Graph Neural Networks

Equivariant Graph Neural Networks are a type of neural network designed to maintain symmetry properties of data.

In the context of graphs, these networks preserve symmetries during data transformations, ensuring that the network's output is invariant to changes in input representations. This property is particularly useful for tasks involving structured data like molecular graphs or social networks.

3.3 Group Equivariant CNN with Discrete Convolution Example (MRI / CT 3D Scans / DNA):

Group Equivariant Convolutional Neural Networks (CNNs) with discrete convolution are tailored for tasks involving three-dimensional data, such as MRI and CT scans or DNA sequences. These networks leverage the concept of group equivariance, which means they maintain consistent behavior under various transformations. In the case of 3D data, this equivariance ensures robustness and accuracy in tasks like medical image analysis or genomics.

We talk about Discrete Convolution, referring to the fact that this operation is applied to discrete, grid-structured data, as opposed to continuous functions.

In practical terms, it means that the convolution is applied to specific grid locations, making it well-suited for processing digital images or other discretely sampled signals. In the context of Group Equivariant CNNs with discrete convolution for tasks involving three-dimensional data (like MRI or CT scans), this operation plays a crucial role in capturing local patterns and features in a way that is invariant to transformations like rotations or translations.

The discrete convolution operation is adapted to maintain symmetry properties, making it especially powerful for tasks where spatial relationships and orientations are essential.

3.4 Group Convolution (Filter transformation + translational convolution step):

Group Convolution involves two main steps: filter transformation and translational convolution.

In the filter transformation step, rotated copies of a basic filter are created to capture different orientations in the data. The translational convolution step, similar to standard CNNs, involves moving the filter across the input to capture local patterns.

This approach is powerful for tasks where spatial relationships and orientations are crucial, such as image processing or computer vision.