Topics: #graphnets #equivariance
Link: https://arxiv.org/abs/2102.09844v1

What

EGNNs. Learn graphnets equivariant to rotations, translations, reflections and permutation. EGNN don't require expensive higher-order representations in intermediate layers, and scale to high-dimensional spaces.

Related works are tensor-field-networks, se3-transformer, radial-field-network, schnet.

Background

equivariance:

$T_g : X \to Y$ is a set of transformations on X for the abstract group $g \in G$. Any function $\phi : X \to Y$ is equivariant to g if there exists an equivalent transformation on its output space $S_g : Y \to Y$ such that

$$\phi(T_g(x)) = S_g(\phi(x))$$

What this means is that applying a transformation* to an input point, and they applying an equivariant function to the result, is the same as applying the function first and then some some transformation.

*a transformation the function is equivariant to

Example of common equivariant transformations we care about are translation, rotation (and reflection), and permutation.

Why

Inductive bayes are needed. Inductive biases can exploit simmetries. E.g. translation equivariance in CNNs and permutation equivariance in GNNs.

Many problems also exhibit 3D translation and rotational simmetries, such as point-clouds, 3D molecular structures and n-body particle simuations. Related work on this uses higher-order representations for intermediate network layers, and the computation requires expensive stuff. Addionally, often the data is restricted to scalar values and 3D vectors.

How

The layer is called EGCL. Consider a graph with feature node embeddings h_i and also a n -dimensional coordinate x_i associated with each of the nodes. The model will preserve equivariance to rotations and translations on these set of coordinates x_i, and will also preserve equivariance to permutation on the set of nodes V just like a normal GNN.

Concisely, $h^{l+1}, x^{l+1} = ECGL[h^l, x^l, edges]$.
Equations for the layer are:

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right) \tag{3}$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) \tag{4}$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \tag{5}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right) \tag{6}$$

So what is the difference from a standard GNN?

- Equation 3: the node message is computed based on the node hidden representation, the messaging nodes representation (neighbors), edge attributes (a_{ij}). The difference is that we also include the distance between the nodes sender and receiver nodes in coordinate space.
- Equation 4: the position of each particle is updated as a "vector field in a radial direction". What does this mean? Each particle x_i is updated by the weighted sum of all the relative differences with the other particles. The weights of this sum are the outputs of the function ϕ_x that takes as inputs the node embedding and outputs a scalar value.
- Equation 5 and 6 are nothing new with respect to a simple message-passing GNN.

E(n) equivariance

Sooo, the model should be translation equivariant on x for any translation vector $g \in R^n$ and should be rotation and reflection equivariant on x for any orthogonal matrix $Q \in R^{n \times n}$.

More formally,

$$Qx^{l+1} + g, h^{l_1} = ECGL(Qx^l + g, h^l)$$

We call these properties $E(n)$ invariance.
Let's go intuitevely about understanding why they hold for this model.

Let's consider a feature h^l that is already $E(n)$ invariant. The resulting edge embedding from equation 3 will also be $E(n)$ ivnariant, since it only depends on squared distances between the node coordinates, which are invariant.
Moving on, we see that Equation 4 computes the updated coordinates by a weighted sum of differences between node coordinates. This operation also preserves equivariance.
Finally, Equation 5 and 6 contain updated that only depend on h^l and m_{ij} which, as we've seen, are equivariant. This makes the whole ECGL layer equivariant.

Inferring the edges

Sometimes, edges are not given (e.g. in point clouds) and we would like to infer them. For this purpose, the authors propose a slight modification that allows to infer/predict the edge while still keeping the model $E(n)$ invariant.

And?

SOTA results in modelling dynamical systems, representation-learning in graph-autoencoders, prediction of molecular properties in QM9 dataset.