The field of geometric deep learning is booming, there's no way around it. And graph neural networks are the rockstars, sitting in the driver seat.

And graphs were clearly at the center of a lot of attention at ICLR 2021.
Not only thanks to Michael Bronstein's amazing keynote speech on geometric deep learning, but also thanks to a number of interesting papers ranging from theory, methods and applications.

In this post, I will try to summarize what these papers were mainly about, what seem to be the current trends, and what we can expect from the future.

Methods

WASSERSTEIN EMBEDDING FOR GRAPH LEARNING

This paper is all about the development of a new framework for embedding entire graphs. Graph embedding methods typically require several steps, namely i) feature aggregation to produce node embeddings, followed by some type of ii) graph coarsening or pooling, and a final iii) classification step.

However, current methods for graph classification fail to scale with the number of graphs in the dataset. As an example, kernel methods typically need to compute pairwise similarities between the graphs, which is computationally heavy when the dataset is large.

The approach proposed in the paper employs optimal transport methods to measure the dissimilarity between the graphs. Similar approaches already exist, but the authors here derive a direct linar map to compute the embeddings, making the algorithm linear in the number of input graphs.

The first step in the process is to produce the node embeddings, which is done via a simple non-parametric diffusion process. After that, the node representations across layers are concatenated to get the final embedding.
These representations are then mapped to the Wasserstein embeddings, with the idea that the L2 distance between the resulting vectors approximates the 2-Wasserstein distance.

These final representation can then be used in any kind of downstream classifier or kernel method.

With this approach, the authors achieve SOTA or competitive results on the obgb-molhiv dataset for molecular property prediction and on several TUD benchmark datasets.

Simple Spectral Graph Convolution

Another graph convolution method to combat oversmoothing, derived from spectral principles.

Here, the authors propose a spectral-based graph convolutional layer, called Simple Spectral Graph Convolution (S2GC), which is based on the Markov Diffusion Kernel (MDK). The authors show that S2GC is capable of aggregating k-hop neighbourhood information without oversmoothing.

The initial proposed layer has the form

$$\hat{Y} = \text{softmax}(\frac{1}{K}\sum_{k=0}^{K}\widetilde{\mathbf{T}}^{k}\mathbf{X}\mathbf{W}).$$

with $\hat{T}$ being the normalized adjacency matrix with added self-loops.

However, this formulation can still incur in oversmoothing. Thus, the authors incorporate the possibility to interpolate between self-information and neighbor aggregation:

$$\hat{Y} = \text{softmax}\left(\frac{1}{K}\sum_{k=1}^{K}\left((1-\alpha)\,\widetilde{\mathbf{T}}^{k}\mathbf{X} + \alpha\mathbf{X}\right)\mathbf{W}\right).$$

The model is evaluated on the tasks of text and node classification, as well as node clustering and community detection. In general, the results are competitive with previous models, but this new formulation appears to be able to combat oversmoothing as the receptive field increases.

It is also showed in the paper that the proposed filter, by design, will give the highest weight to the closest neighborhood of a node, and that the model can incorporate larger receptive fields without undermining contributions of smaller receptive fields, which might be the reason why this model doesn't suffer from oversmoothing.

ADAPTIVE UNIVERSAL GENERALIZED PAGERANK GRAPH NEURAL NETWORK

The goal here is to build a model that can adapt to both homophily and heterophily settings, and combat oversmoothing. The proposed approach incorporates the generalized page rank (GPR) algorithm in Graph Neural Networks (GNNs).

Simply put, the GPR algorithm assigns scores to nodes in a graph that are then used for clustering purposes.

The GPR + GNN process is as follows:

$$\hat{\mathbf{P}} = \text{softmax}(\mathbf{Z}),\ \mathbf{Z} = \sum_{k=0}^{K} \gamma_k \mathbf{H}^{(k)},\ \mathbf{H}^{(k)} = \tilde{\mathbf{A}}_{\text{sym}} \mathbf{H}^{(k-1)},\ \mathbf{H}_{i:}^{(0)} = f_\theta(\mathbf{X}_{i:}),$$

Here, the initial node representation is derived by a neural network f_θ.
After that, the typical graph diffusion is performed using the symmetric adjacency matrix in order to get representations that aggregate information from 1 to k-hop neighborhoods.
To get the final representations, the hidden representaions from 1 to k are summed and weighted by values γ_k. These weights are trained jointly with f_θ.

As the authors point out, the weights γ_k give the model the ability to adaptively control the contribution of each propagation step and adjust it to the node label pattern, thus adapting to both homophily and etherophily settings.

Moreover, oversmoothing should be combated by the model by assigning less weight to large-range propagation steps whenever they are not beneficial in the training procedure.

HOW TO FIND YOUR FRIENDLY NEIGHBORHOOD: GRAPH ATTENTION DESIGN WITH SELF-SUPERVISION

The paper is an exploration of attention in graph neural networks. Moreover, the authors propose to use attention-based, self-supervised link-prediction as an auxiliary task when doing node classification.

Apparently, there is room to improve self-attention mechanism in graph neural networks, as for example GATs have typically showed performance improvements, but the improvements are not consistent across datasets, and it's not even clear what graph attention actually learns.

Let's recall that the GAT's attention comes in the form of $a^T[Wh_i||Wh_j]$. In addition to this, the authors investigate dot-product attention, which is in the form $(Wh_i) * (Wh_j)$.

In the proposed self-supervision framework, the attention is used to predict the presence/absence of edge between node pairs. In essence, it's an auxiliary link prediction task.

The authors explore 4 different attention mechanisms: the original one from GAT, dot product attention, scaled dot product, and a mix of dot product and GAT attention.

The final loss is the combination of the cross-entropy on the node labels (for the node classification task), and the self-supervised graph attention losses, plus an L2 penalty term.

Given this framework, the authors pose some research question, which I'll summarize here.

1. **Does graph attention learn label agreement?** The authors here seem to think that, due to oversmoothing in GAT, if an edge exists between nodes with different labels, that it will be hard to distinguish them. Thus, they postulate that ideal attention should give all the weights to label-agreed neighbors. In their experiments, they show that GAT attention learns label-agreement better that dot-product.
2. **Is graph attention predictive for edge presence?** On the link predictiont ask, dot-product attention consistently outperforms GAT attention.
3. **Which graph attention should we use for given graphs?** The hypothesis here is that different attention mechanisms have will have different abilities to model graphs under various homophily and average degree. Here, the best performing model in low-homphily settings employs scaled dot-product attention with self-supervision, showing that self-supervision can be useful. However, when homphily and average degree are high enough, there is no difference in performance between all the models, including a vanilla GCN.

All of the experiments done so far were done using synthetic dataset, as they allow for controlling several graph properties. However, the authors show that the design choice generalize to many real-world datasets.

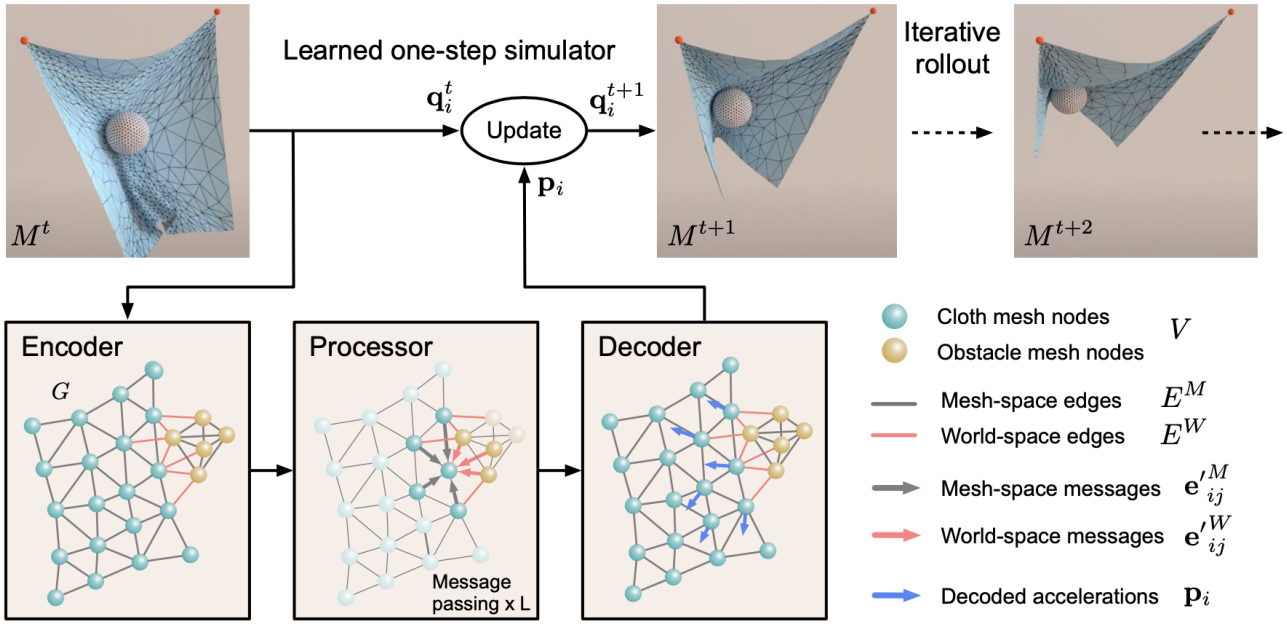
Learning mesh-based simulations with graph networks

Here the authors introduce MeshGraphNets, a framework for learning mesh-based simulations using graph neural networks.
These simulations allow modelling of complex physical systems, such as cloth, fluid dynamics, and air flow.

The original simulation meshes are represented by a set of mesh nodes and edges. Each node is associated with a coordinate in mesh space, and with the value of the quantity that the system is modelling. Some systems, called *Lagrangian*, are also endowed with a 3D so-called *world-space* coordinate for the nodes.

The proposed model first translates the mesh represantion into a multigraph, with the Lagrangian systems having additional"word"edges to enable interactions that might be local in 3D space but non-local in mesh space. These edges are created by spatial proximity.

In the graph, positional features are encoded in the edges. The initial features for both nodes and edges are then encoeed with simple MPLs, and that passed though several layers of message-passing.



The latent node features for the nodes are finally translated by a decoder MLP into output features p_i. These features are interpreted as derivatives of the system quantitiy that we are modelling, so that they can be used to update the state of the system at the next timestep.

The resulting simulations, which can be seen here, tend o run 1-2 orders of magnitude faster that the simulations they were trained on.

LEARNING PARAMETRISED GRAPH SHIFT OPERATORS

This paper provides and analysis for graph shift operators (GSO)* in graph learning, as well as a strategy for learning parametrized shift operators on graph.

The parametrization is achieved in the following way:

$$\gamma(A, \mathcal{S}) = m_1 D_a^{e_1} + m_2 D_a^{e_2} A_a D_a^{e_3} + m_3 I_n,$$

where D_a is the degre matrix, and A_a is the adjacency matrix with self-loops.

Depending on the parameters values, one can retrieve commonly used graph shift operators, as shown in the following table:

Screenshot 2021-05-11 at 16.08.26.png

The authors then show how to include this new parametrized GSO in common GNN architectures.

The exposition continues with a brief theoretical analysis, where for example it is shown that the parametrized GSO has real eigenvalues and eigenvectors, making it feasible to use in exhisting spectral network analysis frameworks where this property is required. Some other bounds useful for numerical stability are derived in the paper.

Bottleneck stuff

One of the main issues with graphnet is the *bottleneck* issue: GNNs struggle to propagate information between distant nodes in the graph. This paper introduces this issue as the *over-squashing* problem, since the network is not able to compress exponentially-growing information into fixed-sized vectors.

As a result, traditional GNNs perform poorly when the prediction task depends on long-range interaction. Moreover, it is shown that GNNs that absorm incoming edges equally, such as GCN and GIN, are more subsceptible to this issue than GAT of GGNN.

In most literature, GNNs were observed not to benefit from more than a few layers. The common explaantion for this is oversmoothing: node representations become indistinguishable when the number of layers increases. For long-range task, however, the paper hypothesizes that the explanation for the limited perfomance lies in oversquashing.

In fact, the bottleneck issue here is very similar to the issue for seq2seq models, which typically suffer from a bottleneck issue in the decoder when attention is not used, since the receptive field of a ndoe grows linearly with the number of steps.

For graph, the issue is strictly worse, since the number of nodes that fall in the receptive field of a target node grows exponentially with the number of message-passing steps.

The *problem radius* here is defined as a useful measure for defining the problem, and it correponds to the prioblem's required range of interaction. Clearly, this is typically unknown in advance, and is usually approximated empirially by tuning the number of layers.

When a prediciton problem as a problem radius r, the GNN muyst have as many layers K as r. Howeverly, since the number of nodes in the receptive field grows exponentially with K, the network need to squash an exponentially-growing amount of intomation into a fixed-size vector, and so crucial messages might fail to reach thei distant destinations, and the model would learn only short-ranged signals fromt he training data and fail to generalize at test time.

The authors show the NEIGHBORMATCH task a toy task that exhibits the need for long-range interactions between nodes.

Examples of tasks that require long-range interaction appear in the prediciton fo chemical properties of molecules, which might depend on the combination of atoms that reside on opposite sides of the molecule.

On the toy dataset, it is shown that, even if enogh layers are built into the network, the models underfit when the number of layers increases. The GAT and GGNN fails later that GCN and GIN. this difference can be exaplined by the neighbor aggregation computation. GCN and GIN aggregate all neighbors before combining them with the target node's representation. Thus they must compress the information flowing from all the leaves into a single vector, and only afterwars interact with the target node's own representation. In contrast, GAT uses attention to weight incoming messages given tha target node's representation. At the last layer only, the tatget node can ignore the irrelevant incoming edge, and absorb only the relevant edge.

A question posed is: if all GNNs have reached low training accuracy, how do these GNN models usually fit the training data in public datasets of long-range problems? The hypothesis is that they overfiit short-range signals and artifacts from the training set, rather that learning the long-range information that was sqashed in the bottleneck. Thus, they generalize poorly at test time.

Simple solution: Adding a fully-adjacent (FA) layer in the last layer. the FA later has every pair of nodes connected by an edge. This allows the topology-aware representations (that arrived at the last layer) to interact directly and consider nodes beyond thei original neighbors. This significanly reduces the error rate.

Moreover, the authors try to assess whether the issue might have to do with under-reaching rather that oversquashing. However, that is not the case, as they show that the networks already had enough layer to reach the radius needes for they task.