EPFL

Graph Networks for Wind Nowcasting

EE-452. Graph Networks offer a flexible way of handling non-regular dataset such as sparse measures in the 3D space.

1 Introduction

Air Traffic Controllers (ATC) need to have access to reliable wind speed forecasts to organize the airspace efficiently. At cruising altitudes, the only measures that one has access to are measured by airplanes that record the wind along their trajectories, and fortunately, as there are a lot of planes flying in the airspace ATCs have recorded a lot of data. Their interest is to be able to a good prediction during the window of time they operate which is rather short (30min up to a few hours) In that range, called nowcasting, one gets the most accurate forecast by extrapolating from the latest measures available rather than trying to solve expensive numerical equations solvers. It is in that range that deep learning methods can offer a lot, as meteorological data is often highly available. Sadly this data are recorded along trajectories and is sparse in space, so we cannot uses standard methods such as CNN to deal with it. We noticed that it could be a good candidate for Graph Networks are their inductive bias makes them good candidates for this task.

This work will use graph networks to forecast the wind speed at cruising altitudes. We will use the *SkySoft ATM MALAT Wind Speed Dataset* [Berling et al., 2021] and starting from an Graph Element Network (GEN) [Alet et al., 2019] implementation, we will analyse the effect of the topology of the graphs and different nodes and edge features on the performance of the model.

1.1 Problem formulation

Let $\mathbb{X}, \mathbb{I}, \mathbb{O}$ be three bounded metric spaces, and define two functions $f : \mathbb{X} \to \mathbb{I}$, mapping from data to the input space, and $g : \mathbb{X} \to \mathbb{O}$ mapping to the output space. In the case of wind nowcasting, $\mathbb{X} = \mathbb{R}^3 \times \mathbb{R}^+$, which corresponds to 3 spatial coordinates and one temporal and $\mathbb{I} = \mathbb{O} = \mathbb{R}^2$ which encode the wind in the (x, y) plane. We want to learn the transformation which maps a function f to g. We only have access to f by data $(x, i) \in \mathbb{X} \times \mathbb{I}$ of measures i at position x and we know g only through a set $(x', o) \in \mathbb{X} \times \mathbb{O}$ of measurements o at position x'. So we learn this mapping from data using multiple sets of meaures corresponding to many (f, g) pairs with the objective of minimizing the distance in the output metric space $\mathbb{O}$ between predictions and references.

EDIC
Idiap Research Institute
Machine Learning Group

Arnaud Pannatier
arnaud.pannatier@epfl.ch
arnaudpannatier.ch

EPFL

2 Related works

2.1 Graph Element Networks (GENs)

Graph Element Networks [Alet et al., 2019] aim to model SFTs using a non-regular graphs with nodes in the underlying space $\mathbb{X}$. Each measurement (x, i) is encoded using a small MLP, the initial latent feature of one node is then computing by averaging the embeddings of the measurements based on their distance to the node. The model then process this latent variable using $T \in \mathbb{N}$ steps of message passing, using the following equations:

$$m_{ij}^{t+1} = m_e(l_i^t, l_j^t) \tag{1}$$

$$l_j^{t+1} = m_n(l_i^t, \sum_{(i,j)\in\mathcal{N}} m_{ij}^{t+1}) \tag{2}$$

Where m_e, m_n are multilayer perceptrons and l_i^t is the latent state of nodes i at the t-th step of message passing. In that representation no edge features are used. In order to predict a value at a new query position, the model linearly extrapolates in latent space, and decodes using a small MLP modelling the transformation from latent to output space.

[AP] advantages – drawbacks

2.2 Finite Element Networks

Traditional PDEs solvers are model the domain using regular grids, but this has some drawbacks as some part of the space might be more complicated to model than the other. One way of dealing with this problem is to use a Finite Element method approach [Hughes, 2012], which uses a non-regular graph that can be denser in the more complex regions and coarser in the smoother regions.

[AP] advantages – drawbacks

3 Exploration

3.1 Graph Types

Usually the mesh for FEM simulations is created beforehand with denser region where we expect the field to behave more complicated way. Meshes for this simulation are usually a triangulation of the region of interest. The triangulation subdivise the space into non overlapping triangles, such that each triangle have either a side or a vertex in common or are disjoint. A density parameter define the finess of the triangulation. The graph generated by such this triangulation scheme is such that there is an edge between two nodes only if there are close in the euclidean sense. The degree distribution of these graphs has only small values corresponding to the local neighborhood.

[AP] TODO : graph

For GENs, the original way of creating graphs is to create a grid mesh that covers the whole space, and to encode the node position as parameters of the model, so that it can be optimized by gradient descent during training. In that case, the graph structure remains similar to the initial grid structure.

[AP] TODO : graph

In both apporaches, only those graph structures have been used. In this work, we will consider three approaches for our graph topology. The first one, considered as baseline is constructed as follow : First we will take the k-means of the measure positions and we will add edges between the nearest neighbors. The second one will be a random network were the nodes position and their edges are taken at random, we will try different parameters p and analyse its effect on the quality of the model.

For the last graph structure, we will use a Barabási-Albert model scheme for growing a graph to a given number of nodes, we will then associate to each node one of the k-means position.

[AP] TODO : graph

3.2 Features

In this work we will focus on nodes and edges features as we are using the graph network to do regression over the whole space. We won't need to aggregate the features over the whole graph.

3.3 Nodes features

As described in section 2.1, GEN are only using nodes features that are computed by aggregating the learned embeddings of the measures based on their distance. We will visualise their value in some very unbalanced cases and the effect of message passing on the features. This will help us to choose the message passing steps and guide us through the analysis of the different graph structures.

We assume that this part is the most important to get a good model. So we will explore different strategies and measure their impact on the model. First we will try to use no encoder and aggregate the normalized measured values. With this approach, we delegate the whole processing to the graph networks, and we think this should be already a strong baseline as the outputs values are strongly related to these initial values. We will then try to use fixed sine-cosine positional encodings [Vaswani et al., 2017] and 3D positional fixed sine-cosine positional encodings [Chu et al., 2021].

3.4 Edge features

A contribution of this work is to add edge features to GENs. We think that a good graph networks for space modelling should be able to compute approximation of the derivatives of values in the graph. We will take Euler scheme as an inspiration and have an explicit design for edge features that allows this kind of computation. We will make an ablation with learned features and see their effect on the quality of the model.

[AP] TODO: sketch

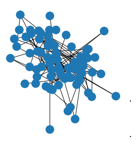
Euler equation:

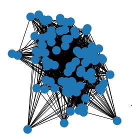
$$\rho \mathbf{g} - \nabla p = \rho \frac{d\mathbf{v}}{dt} \tag{3}$$

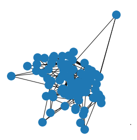
4 Exploitation

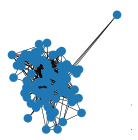
4.1 Graph Types

The performance of the different graph types are displayed in Table 1. We see that allowing position to move during training (ADAPT in the table) are always performing better than their fixed counterpart. Random Graphs are the baseline and we expect them to perform poorly as they are initialized randomly without taking into account additional insight that we can have about the dataset. We see that indeed their performance is worse than the other models.


(a) Random network with $p = 1\%$


(b) Random network with $p = 10\%$


(c) k-means, 3nn


(d) k-means, BA

Model		Nodes		
		10	100	1000
Random Network, $p = 1\%$	FIXED	0.80 $\pm$ 0.04	0.71 $\pm$ 0.02	0.70 $\pm$ 0.02
	ADAPT	0.69 $\pm$ 0.01	0.65 $\pm$ 0.02	**0.64 $\pm$ 0.02**
Random Network, $p = 10\%$	FIXED	0.76 $\pm$ 0.03	0.70 $\pm$ 0.02	0.76 $\pm$ 0.02
	ADAPT	0.69 $\pm$ 0.01	0.66 $\pm$ 0.01	0.66 $\pm$ 0.02
k-means, 3 nn.	FIXED	0.68 $\pm$ 0.01	0.69 $\pm$ 0.02	0.68 $\pm$ 0.01
	ADAPT	0.68 $\pm$ 0.01	0.66 $\pm$ 0.01	0.65 $\pm$ 0.01
k-means, BA	FIXED	0.69 $\pm$ 0.01	0.68 $\pm$ 0.01	0.66 $\pm$ 0.02
	ADAPT	0.68 $\pm$ 0.02	0.65 $\pm$ 0.02	**0.64 $\pm$ 0.01**

Table 1: We tried different graph topology for our model that use no embeddings. First, as a baseline, we tried Random Networks, were the position of each nodes is drawn from $\mathcal{N}(0, 2)$ and where there is an edge between two nodes with probability p. Then we tried to use k-means to get the position of the nodes and added edges either between the 3 nearest neighbors in the graph or following a Barabasi-Albert procedure. The results are the MSE error on the validation set, we ran ten different seeds to get the standard deviation. all measures are normalized.

Model		Nodes		
		10	100	1000
No Emb	FIXED	0.68 $\pm$ 0.01	0.69 $\pm$ 0.02	0.68 $\pm$ 0.01
	ADAPT	0.68 $\pm$ 0.01	0.66 $\pm$ 0.01	0.65 $\pm$ 0.01
MLP	FIXED	0.66 $\pm$ 0.01	0.66 $\pm$ 0.02	0.67 $\pm$ 0.02
	ADAPT	0.XX $\pm$ 0.XX	0.XX $\pm$ 0.XX	0. $\pm$ 0.
Added Pos-Enc + MLP	FIXED	1.03 $\pm$ 0.04	1.06 $\pm$ 0.02	1.04 $\pm$ 0.02
	ADAPT	0.XX $\pm$ 0.XX	0.XX $\pm$ 0.XX	**0.00 $\pm$ 0.00**
Concat Pos-Enc + MLP	FIXED	0.65 $\pm$ 0.01	0.63 $\pm$ 0.01	0.65 $\pm$ 0.02
	ADAPT	0.XX $\pm$ 0.XX	0.XX $\pm$ 0.XX	0.XX $\pm$ 0.

Table 2: We tried different types of nodes embeddings for our model. The first one, is the baseline is to use no embedding layer and keep the aggregated wind value as the node state. The second one use a small MLP encoder to project the wind value to an higher dimensional space. Finally we explored two version of positional encoding layer. The results are the MSE error on the validation set, we ran ten different seeds to get the standard deviation. all measures are normalized.

4.2 Nodes features

Have a good baseline.

5 Communication

- Good report
- Github
- Streamlit

6 Conclusion

References

[Alet et al., 2019] Alet, F., Jeewajee, A. K., Villalonga, M. B., Rodriguez, A., Lozano-Perez, T., and Kaelbling, L. (2019). Graph element networks: adaptive, structured computation and memory. In

Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 212–222. PMLR.

[Berling et al., 2021] Berling, D., Pannatier, A., and Fleuret, F. (2021). Skysoft atm malat wind speed.

[Chu et al., 2021] Chu, X., Tian, Z., Zhang, B., Wang, X., Wei, X., Xia, H., and Shen, C. (2021). Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*.

[Hughes, 2012] Hughes, T. J. (2012). *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.