Advanced methods in applied statistics 2023
Using Graph Neural Networks in particle physics

Magnus Guldbæk Hansen
University of Copenhagen
08-03-2023

1 Introduction

The use of machine learning in particle physics has long been a mainstay. This has for instance been used for classification, both for particle identification and event selection, as well as regression for estimating energy levels of particles and event reconstruction as a whole [1, 2]. While neural networks like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are a powerfull tool, these methods are restricted by their dependence on the structure of the data, such as a grid-structure for an image. In this article, the focus is on the newer machine learning method of graph neural networks (GNNs) and how they alleviate this weakness, a presentation of the graph network formalism and the implementation of GNNs in particle physics.

2 Graph Neural Networks

As mentioned prior, CNNs are commonly used when it comes to machine learning in the physical sciences. They have their origins in computer vision, but can be generalized to other similarly structured problems. By taking the input values of our "image" or matrix, and putting it through a set of layers whose values is given by a combination of the previous layers and the weight of it's values, we can eventually start to classify features in our matrix.

However, in the more general scenario where our data cannot be organized in a grid-like structure, one may generalize to a graph instead. A graph is a set of nodes (or vertices), which contain information on the relevant objects we are interested in, and edges (or links) which describe relations between the nodes. This is where graph neural network comes in, since we cannot use CNNs or similar on a graph. There are many different versions of GNNs in use [2], depending on the needs of the problem at hand. As such, we use the generalised *graph network* formalism presented in [3], to describe how they work.

For a set of nodes $V = \{\boldsymbol{v}_i\}_{i=1:N_v}$, where v_i is the i'th attribute of the node and N_v is the amount of nodes, we can define our edges between them as $E = \{(\boldsymbol{e}_k, r_k, s_k)\}_{k=1:N_e}$, with e_k being the nodes k'th attribute, r_k and s_k referring to which nodes the edge is between and N_e being the amount of edges. If we denote the global graph attributes as $\boldsymbol{u}$ we can represent our graph as $G = \{\boldsymbol{u}, V, K\}$.

As an equivalent for layers in the CNN, we define what is called a graph network (GN) block. A GN block is composed of 3 update functions (ϕ^e, ϕ^v and ϕ^u) and 3 aggregation functions ($\rho^{e\to v}, \rho^{v\to u}$ and $\rho^{e\to u}$), that work on the combination of graph, nodes and edges to produce a new graph with the same structure. The update function takes some input and updates it, ie ϕ^v updates the attributes of a node. The aggregate function takes a variable amount of inputs, and returns a singular output. A simple aggregation function could for example be element-wise addition.

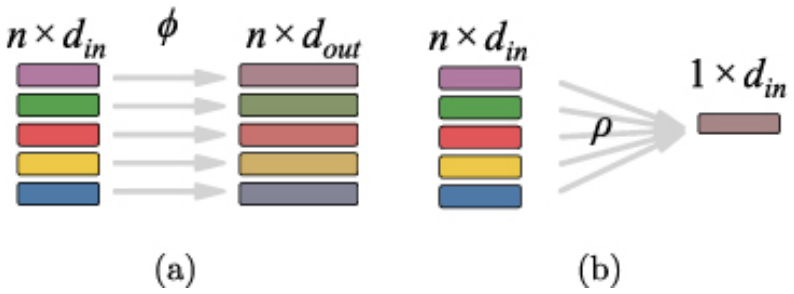

Figure 1: Visualisation of an update function (a) and aggregation function (b).

The general procedure for a GN block is as follows:

- Compute $\boldsymbol{e}'_k = \phi^e(\boldsymbol{e}_k, \boldsymbol{v}_{r_k}, \boldsymbol{v}_{s_k}, \boldsymbol{u})$, which is our updated edges and thus the relations for our new graph.
- Aggregate the edges, both locally with $\bar{\boldsymbol{e}}'_i = \rho^{e\to v}(E'_k)$ and globally with $\bar{\boldsymbol{e}}' = \rho^{e\to u}(E')$, where E'_i is the amount of nodes on the i'th node.
- We can now update our nodes to $\boldsymbol{v}'_i = \phi^v(\boldsymbol{e}'_i, \boldsymbol{v}_i, \boldsymbol{u})$.
- This is then aggregated over the whole graph to $\bar{\boldsymbol{v}}' = \rho^{v\to u}(V')$.
- The last step is then to update the attributes of the graph itself, $\boldsymbol{u}' = \phi^u(\boldsymbol{e}', \boldsymbol{v}', \boldsymbol{u})$.

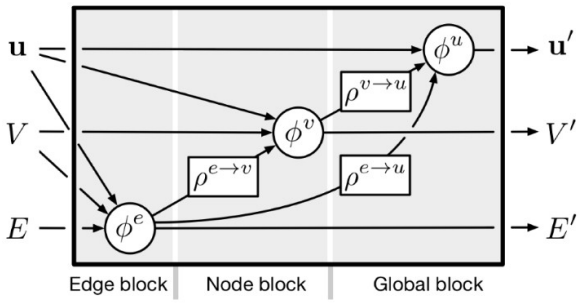

Figure 2: The structure of a GN block, showing how different inputs and functions feed into each other.

3 Jet flavour tagging at the ATLAS detector

One example of where GNN's have been used within the field of particle physics, is flavour tagging or jet classification [2]. Flavour tagging is the act of determining which particle decaying led to a spray of more stable particles, known as a jet. While the classification of b-jets, jets with origins in the decay of a b-hadron, was possible prior, the advent of GNN allows for higher precision and tagging other partices such as c-hadrons.

One such example is from [4], in which a GNN based algorithm, GN1, is compared to the previous algorithm, DL1r. DL1r had the downside of being dependent on a combination of lower level algorithms, off which some have to be manually optimized and others trained, which GN1 simplifies into one algorithm. Furthermore, the GN1 algorithm is able to not only able to classify jet flavours, but also the auxiliary attributes of track origins of the jets and the track-pair vertex compatibility, which denotes if two tracks share the same vertex. Both of these are then used in the tagging. Most importantly, it is shown that GN1 improves on DL1r. For instance the rejection of both c-jets and light-jets improves by factors 2.1 and 1.8 respectively, for jets from $t\tilde{t}$ in the $20 < p_T < 250GeV$ and a b-tagging efficiency of 70%. The GN1 algorithm also outshines the DL1r algorithm in multiple other areas.

References

[1] G. Carleo *et al.*, "Machine learning in the physical sciences," *Rev. Mod. Phys.*, vol. 91, no. 045002, pp. 13–22, 2019. [Online]. Available: `https://arxiv.org/abs/1903.10563v2`.

[2] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Mach. Learn.: Sci. Technol.*, vol. 2, no. 021001, 2021. [Online]. Available: `https://iopscience.iop.org/article/10.1088/2632-2153/abbf9a`.

[3] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, and V. Zambaldi, "Relational inductive biases, deep learning, and graph networks," pp. 10–13, 2018. [Online]. Available: `https://arxiv.org/abs/1806.01261v3`.

[4] T. A. collaboration, "Relational inductive biases, deep learning, and graph networks," 2022. [Online]. Available: `http://cds.cern.ch/record/2811135/`.