DS-GA 1006 Fall 2017

Capstone Project Progress Memo

Voyagers:

Yurui Mu, Sudhir Nallam

Xiaoyu Wang, Wenqing Zhu

Nov 9, 2017

---

0. **Project Description from Original Proposal**

   QCD-Aware Recursive Neural Networks for Jet Physics[1] uses techniques of NLP to represent the jet substructure as a 'jet embedding', which maps a set of 4-momenta(vector representing the particle) into R^q. Unfortunately, there is a large background from jets produced by more mundane QCD processes, which interferes the process of source determination. In our project we propose a few techniques to make the model proposed in [1] more robust to systematic uncertainties.

1. **Problem Statement**

   In paper [1], it presents a hybrid approach that leverages the structure of sequential recombination jet algorithms and deep neural networks. To explain this problem with an analogy, it is much like a sentence is composed of words following a syntactic structure organized as a parse tree, a jet is also composed of 4-momenta following a structure dictated by QCD and organized via the clustering history of a sequential recombination jet algorithm. Think of the recursive neural network as providing a 'jet embedding'.

We propose to embed the full particle content of an event by feeding a sequence of jet-embeddings into a recurrent network. Before this, we formalize the classification tasks at the jet-level and event-level. We describe a collision event as being composed of a varying number of particles, and each particle is represented by its 4-momentum vector. The 4-momenta in each event can be clustered into jets with a sequential recombination jet algorithm. The jet algorithm has clustered the event into M jets, each of which can be represented by a binary tree. For jet-level classification or regression, the goal is to build a predictive model minimizing some loss function L^{jet}. Similarly, for event-level classification or regression, assume each collision event in the training data comes with labels or regression values, and the goal is to build a predictive model minimizing some loss function L^{event}.

2. Data

(1) Data Description

In this project we are using the simulator results from LHC(Large Hadron Collider). In order to focus attention on the impact of the network architectures and the projection of input4-momenta into images, we are using boosted W tagging. The signal $(y = 1)$ corresponds to a hadronically decaying W boson with $200 < p_T < 500GeV$, while the background$(y = 0)$ corresponds to a QCD jet with the same range of p_T. The data we obtained is the same as in [1]. We are using the full-event records from $PYTHIA$ benchmark samples, including both particle-level data and towers obtained from $DELPHES$ detector simulation. Also we have prepossessed $anti-k_t$ jet clustering, cropping, trimming, and normalization data at hand.

Our training data was collected by sampling from the original data a total of 100,000 signal and background jets with equal prior. The testing data was assembled similarly by sampling 100,000 signal and background jets independent of training set.

(2) **Data Preprocessing**

We have several steps in data preprocessing. After loading in the original data, we first loop in the data of each jet, append them together into jets. And then for each jet, we compute their p_t. And then by the order of p_T, we build recursive trees. When building the trees, we want to make sure that the left tree always have higher p_T than the right trees.

After preprocessing, in each of record, we have 49 jets informations, including 4-dimension vectors, η, energy, mass, ϕ, p_t, root id and corresponding tree structure.

3. **Methodology**

The central idea of our modeling approach is recursive embedding. As previously mentioned in the problem statement, here we will describe our model on two levels, the individual jets and a full event. In the case of individual jets, particles are topologically structured as a binary tree t_j. The embedding of h_k^{jets} of node k is recursively defined as following:

$$h_k^{jet} = \begin{cases} u_k & if\ k\ is\ a\ leaf \\ \sigma\left(W_h \begin{Bmatrix} h_{KL}^{jet} \\ h_{KR}^{jet} \\ u_k \end{Bmatrix} + b_h \right) & otherwise \end{cases}$$

$$o_k = \begin{cases} v_{i(k)} & if\ k\ is\ a\ leaf \\ o_{k_L} + o_{k_R} & otherwise \end{cases}$$

$$u_k = \sigma(W_u g(o_k) + b_u) \tag{1}$$

where k from 1,.....$2N_j - 1$ denotes the nodes of the binary tree t_j, the left and right children of node k is denoted by k_L and k_R. W_h, b_h, W_u, and b_u are the shared parameters to be learned. σ is the ReLU activation function and g is a function that extract the kinematic features, p, η, θ, ϕ, E and p_T from the 4-momentum o_k.

Starting with the root node and apply h_k^{jet} recursively down to the leaf nodes of the binary tree, the resulting embedding is now a single vector of fixed size. Moreover, this embedding effectively summarized/preserved the structured information from particles in forming jets. A classification or regression task can thus be followed by feeding these embeddings as input to the model. It is worth noted that all parameters(parameters of model and parameters of embedding) will eventually be learned jointly using back propagation. Thus, we will expect to see the final embeddings varies as the task requirement changes.

The full events level model lies above the individual jets level model. To be more specific, the embeddings of entire events will be constructed by feeding individual jets embeddings into a sequence based recurrent neural network. To begin with, we have a sequence of pairs $(v(t_j), h_1^{jet}(t_j))$ ordered by p_T where t_j is the unprocessed 4-momentum of the jet t_j and $h_1^{jet}(t_j)$ is its embedding. We processed this sequence with Gated Recurrent Unit(GRU), and the output h_M^{event} is then chained to a subsequent classifier to solve an event level classification task. Again all three sets of parameters(the inner jets embedding, GRU, the classifier) are trained jointly using backpropagation through structure to minimize event level loss.

4. Experimentation

Our experimentation can be divided into two approaches. In the first approach we are improving the results presented in the [1] by using various techniques. In [1] authors have used clean data for experimentation. For the second approach we are taking data with noise, and trying out various techniques to reduce the noise.

For the improving the results presented in [1], we are considering following changes: using more complex function to represent the non-linear function in the RNN, not including the root node into the RNN sequence, i.e using different weights for the root node and lastly, using input features which are independent of the observed features(jet mass) as replacement for nuisance features [2].

For improving the model performance with pileup/noise data, we are considering these two approaches. Firstly using gated network like LSTM/GRU which would handle the noise. And using domain adaptation technique presented in the [3]. We include additional adversarial loss to the already RNN network, there be making more robust to changes due to noise. Figure 1 shows the architecture of the network.

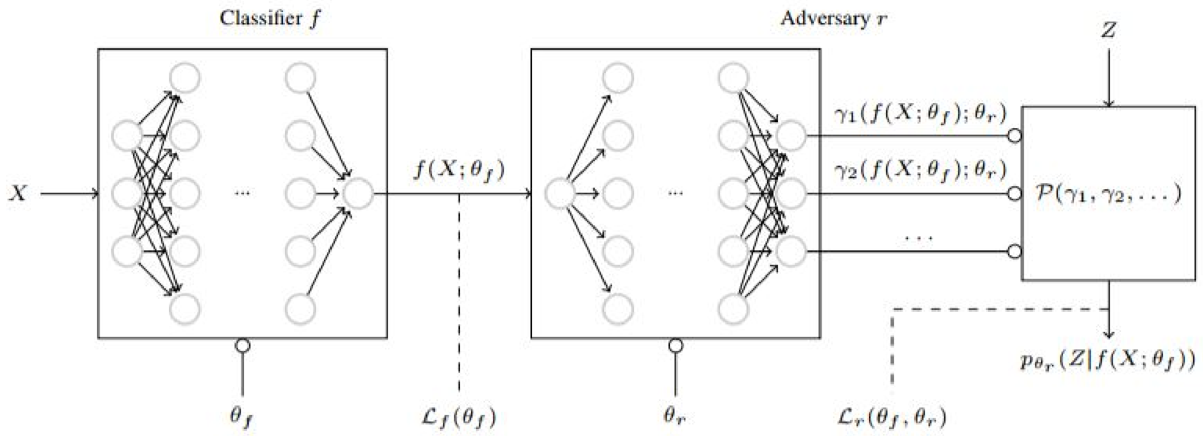
Classifier f
Adversary r
Z
X
$f(X;\theta_f)$
$\gamma_1(f(X;\theta_f);\theta_r)$
$\gamma_2(f(X;\theta_f);\theta_r)$
$\mathcal{P}(\gamma_1, \gamma_2, \ldots)$
$p_{\theta_r}(Z|f(X;\theta_f))$
θ_f
$\mathcal{L}_f(\theta_f)$
θ_r
$\mathcal{L}_r(\theta_f, \theta_r)$

Figure 1: : Architecture for the adversarial training of a binary classifier f against a nuisance parameters Z(taken from [3])

5. **Results**

Presently we have successfully completed following, replicated the results we got from the [1], for which we have taken the code from Louppe, Gilles(`https://github.com/isaachenrion/jets`). We have experimented with only one dataset, antikt-kt and figure 2 shows the ROC curve, which has AUC-ROC = 0.8914.

For replacing the ReLu in the RNN with more complex non-linear function, we have used network-in-network architecture [4]. Replaced the ReLu activation with 1x1 convolutions. ROC curve for this architecture is shown in the figure 3, with AUC-ROC = 0.9146. Code is available at `https://github.com/NYU-CDS-Capstone-Project/Voyager`.

6. **Discussion/Future Plan**

We will be continuing to implement the various approaches listed in the experimentation

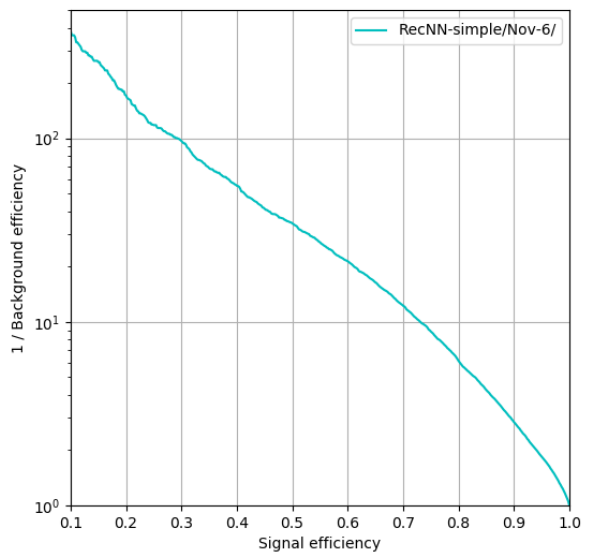

Figure 2: : ROC curve for QCD process

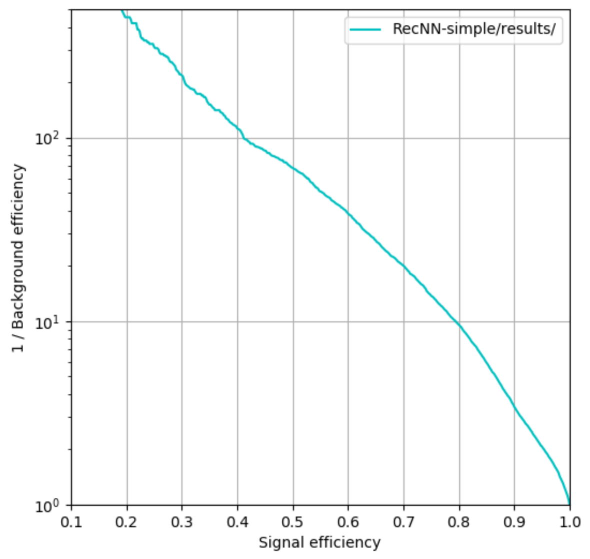

Figure 3: : ROC curve for QCD process with Network in Network architecture

section. We will also start working on data with noise added in. The approach we take is to introduce an adversarial loss which is mentioned in [3]. This approach allows us to train a classifier that is independent of nuisance factors, which in this case will be the background noise we introduced in the data. Besides adding the loss piece, another approach will be trying is to implement GRU model on the noisy dataset and see if the model will learn to not picking up the noise. To go even further, we plan also train a classifier pivoting on jet mass.

Roadmap

(1) **On Data Without Noises**

We want to compare the experiment result across following proposed models.

i. Simple RecNN

ii. Simple RecNN with 1x1x1 convolution

iii. Gated RecNN

iv. Gated RecNN with 1x1x1 convolution

v. Simple RecNN with Adversarial Loss (pivot on jet mass)

(2) **On Data With Noises**

We would like to try and apply the methodology proposed in paper [3], and use adversarial loss to train classifiers that are independent of nuisance factors. And we plan to implement and compare results across following models.

i. Simple RecNN with Adversarial Loss

ii. Gated RecNN

iii. Gated RecNN with Adversarial Loss

References

[1] G. Louppe, K. Cho, C Becot, and K. Cranmer. QCD-Aware Recursive Neural Networks for Jet Physics. *ArXiv e-prints*, February 2017.

[2] Chase Shimmin, Peter Sadowski, Pierre Baldi, Edison Weik, Daniel Whiteson, Edward Goul, and Andreas Sgaard. Decorrelated Jet Substructure Tagging using Adversarial Neural Networks. *Phys. Rev.*, D96(7):074034, 2017.

[3] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to Pivot with Adversarial Networks. 2016.

[4] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.