Search for QCD Instantons

US State Department CERN Program

Maria Bressan
Tancredi Carli

Dec 14, 2022
The QCD Instanton

- A non-perturbative tunneling process predicted by the Standard Model
  - Quantum chromodynamics: theory of quarks and gluons
  - Electroweak instantons could explain the imbalance of matter and antimatter in the universe (baryon number is broken)
- QCD instantons can be produced at LHC:

**Signatures:**
- Each kinematically allowed flavor of quark-antiquark pair produced
- Isotropic decay
- All quarks are all either left- or right-handed (Chirality broken)

We look at quark flavor with quark tagging algorithms and isotropic decay with a neural network
New b-tagging Algorithm

- **b-tagging** - algorithms to identify hadrons containing b-quarks (b-hadrons)
  - Longer lifetime -> secondary vertices
- Use dedicated algorithms developed for low-momentum b-hadrons ([ATL-COM-PHYS-2019-113](#))
- Algorithms create ‘objects’ that represent possible b-hadrons
- Use Monte-Carlo simulated data to study our algorithm performance
b-tagging Algorithm Efficiency

Efficiency vs transverse momentum

![Graph showing efficiency vs transverse momentum](image)

Three algorithms:
- trackJet
- VrtSecInclusive
- SoftBVrtClusterTool
- # of b-hadrons (not to scale)

Although the efficiency is low, we may be able to use the most promising objects (trackJets) to provide flavor information for signal/background discrimination.

Efficiency definition:

\[
\text{Efficiency} = \frac{\text{# of b-hadrons ‘matched’ to a b-tagged object}}{\text{# b-hadrons}}
\]
Isotropic Decay: Use a Deep Neural Network

Classify signal and background events using a neural network

Weights: these are what the network ‘learns’ by minimizing a loss function

Neurons: encode some defining properties of the data. Value determined by inputs and weights

Output: a value between 0-1, how sure the network is of its classification

Values add up to 1

source
Deep Neural Network Inputs

Inputs: X (pt, eta, phi of tracks) and Y (instanton or bkg)

X: 

```
[[ 4.0313496e+03 -1.9271221e+00 -1.1949540e+00 ]
 [ 3.9717961e+03 -2.0108027e+00  2.1147134e+00 ]
 [ 2.3905178e+03 -1.8544655e+00 -1.3523459e+00 ]
...]

[[ 2.5283616e+03 -6.3040692e-01  1.1877756e+00 ]
 [ 2.5093069e+03  1.0891840e+00  2.2480328e+00 ]
 [ 2.4515750e+03  3.1898698e-01 -3.3545771e-01 ]
...]
```

Y: 

```
[[0. 1.]
 [0. 1.]
 [0. 1.]
...]

[[1. 0.]
 [1. 0.]
 [1. 0.]]
```

Python packages for machine learning:

- TensorFlow
- EnergyFlow

arXiv:1810.05165
Variable length input
Doesn’t depend on ordering

Can input additional track properties
Decisions for the User to Make

- Applying **cuts** on tracks
  - Track selection: tracks belonging to primary and secondary vertices
- **Batch size**
  - Number of events given to neural network for training before adjusting weights
- **Number of epochs**
  - Number of times we train the neural net with the whole data set
- Normalize inputs: all numbers [-1,1]
- Add additional inputs
  - b-tagging information, charge, pion/proton/kaon-ID via energy loss

To make decisions: try everything and see what works best
Learning

### Optimal Parameters:

- **Batch size** of ~300 events (out of ~1 million total)
- **20 epochs**

Additional inputs and track selection did not improve performance.

<table>
<thead>
<tr>
<th>Mass (GeV)</th>
<th>Loss vs Epoch</th>
<th>Accuracy vs Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td><img src="image" alt="loss vs epoch #" /></td>
<td><img src="image" alt="accuracy vs epoch #" /></td>
</tr>
<tr>
<td>120</td>
<td><img src="image" alt="loss vs epoch #" /></td>
<td><img src="image" alt="accuracy vs epoch #" /></td>
</tr>
</tbody>
</table>

**Training data:** what we train the neural network with.

**Validation data:** data that is held out to evaluate performance while training.

Running on ATLAS Data

Using only 10% of data (blinding). Background normalized to number of data events. Expected number of signal event for data luminosity (signal cross-section very uncertain).

Low mass: high cross section but low separation
High mass: low cross section but better separation
In Conclusion…

Next steps:

● Learn more about Machine Learning
● Give particle IDs to neural network
● Run over all mass samples
● Fix real and simulated data normalization
● Come back to CERN for my PhD?

I learned about:

● How to address open-ended research questions
● Using RDataFrame for ROOT
● Writing functions in C++
● Tensorflow and Energyflow for Python
● How particle searches are conducted
● How ATLAS takes data
● Everyday life at CERN
Thanks for listening! Questions?