Jet tagging with Neural Networks

Henry Day-Hall
Supervisors: Prof. Claire Shepherd-Themistocleous$^{1,2}$, Prof. Stefano Moretti$^1$, Prof. Srinandan Dasmahapatra$^1$, Dr. Emmanuel Olaiya$^2$

$^1$University of Southampton, UK
$^2$Rutherford Appleton Laboratory, UK

October 1, 2019
Objective

What we want to know.

What we can detect.
Jet tagging

Jets structure the image.

Each jet hopes to encapsulate the shower of a “?” parent particle from the hard interaction.
We can generate showers with known “?” parent in Monte Carlo.
Measuring the Jet

Jets can be tagged based on “Hand crafted” features. This is proven successful and in current use (CSVv2/DeepCSV). Difficult to know if information is being abandoned?
With cuts track $p_T > 5.$ and $\eta < 2.5.$
track $p_T > 5.$ and $\eta < 2.5.$
DeepCSV


Jet tagging with Neural Networks 7/44

With cuts

track $p_T > 1$. and $\eta < 2.5$. 
A heavy Higgs Decay

Here a heavy Higgs (400 GeV) decays to two SM higgs.

Heavy Higgs decays are frequently boosted and often create fat jets.
Sometimes all the b-jets fall on the forward calorimeter.
Difficulty in boosted topologies

The number of jets in track $p_T > 5.$ and $\eta < 2.5$ is 34.4% of the number jets in track $p_T > 3.$ and $\eta < 5.$ The number of b-jets in track $p_T > 5.$ and $\eta < 2.5$ is 35.4% of the number b-jets in track $p_T > 3.$ and $\eta < 5.$
Difficulty in boosted topologies

Some events gain considerably.
Difficulty in boosted topologies

Some events are only seen with more generous cuts.
Difficulty in boosted topologies

What happens if we work with the more generous cuts?

![Receiver Operator curve](image-url)

- **Blue line**: $p_T > 3, |\eta| < 5$
- **Orange line**: $p_T > 5, |\eta| < 2.5$
How to use Track data

- Transverse momentum, $p_T$.
- Psudo-rapidity, $\eta$.
- decay length $d$.
- ...

Different number of tracks in each jet mean this makes variable length vector.
Each neural network had a non linear function $f_{\text{nonlinear}}$ and a number of weight matrices $W_i$.

1. The input is a fixed length vector; $x_0 = (p_T, \eta, A, n_{SV}, d_V, \ldots)$. 
Classic Deep Neural Network

Each neural network had a non linear function $f_{\text{nonlinear}}$ and a number of weight matrices $W_i$.

1. The input is a fixed length vector; $x_0 = (p_T, \eta, A, n_{SV}, d_V, ...)$.
2. The network does a series of linear transformations, followed by $f_{\text{nonlinear}}$ (applied element wise).

\[ x_{i+1} = f_{\text{nonlinear}}(W_i x_i^T) \]
Each neural network had a non linear function $f_{\text{nonlinear}}$ and a number of weight matrices $W_i$.

1. The input is a fixed length vector; $x_0 = (p_T, \eta, A, n_{SV}, d_V, ...)$.

2. The network does a series of linear transformations, followed by $f_{\text{nonlinear}}$ (applied element wise).

$$x_{i+1} = f_{\text{nonlinear}}(W_i x_i^T)$$

3. After each of the weight matrices have been used the final output is much shorter vector $x_f$, the values of the elements indicated possible flavours of the jet; $x_f = (b\text{-jet}, c\text{-jet}, \text{background})$. 
Each neural network had a non linear function $f_{\text{nonlinear}}$ and a number of weight matrices $W_i$.

1. The input is a fixed length vector; $x_0 = (p_T, \eta, A, n_{SV}, d_V, \ldots)$.
2. The network does a series of linear transformations, followed by $f_{\text{nonlinear}}$ (applied element wise).

$$x_{i+1} = f_{\text{nonlinear}}(W_i x_i^T)$$

3. After each of the weight matrices have been used the final output is much shorter vector $x_f$, the values of the elements indicated possible flavours of the jet; $x_f = (b\text{-jet, } c\text{-jet, } \text{background})$.

The $W_i$ are picked to give the correct output for any jet. This only works for fixed length input.
Tree-like Recursive Neural Networks

An algorithm that propagates its internal state through a tree.

Socher, Perelygin, Wu, Chuang, Manning, Ng, Potts, EMNLP 1631 (2013)
A shower

Jets do have a tree structure in nature.

But we can’t observe it ...
Clustering order

The RNN will have one leaf matrix $M_l$ and one memory matrix $M_h$. It will also have a non linear function $f_{\text{nonlinear}}$.

Creating leaves;

1. Each track forms an input vector $x_0 = (p_T, \eta, d, \ldots)$
The RNN will have one leaf matrix $M_l$ and one memory matrix $M_h$. It will also have a non linear function $f_{\text{nonlinear}}$.

Creating leaves;

1. Each track forms an input vector $x_0 = (p_T, \eta, d, \ldots)$
2. The input becomes a memory vector;

$$h = f_{\text{nonlinear}}(M_l x_0^T)$$
Mechanics of the RNN

The RNN will have one leaf matrix $M_l$ and one memory matrix $M_h$. It will also have a non-linear function $f_{\text{nonlinear}}$.

Creating leaves;

1. Each track forms an input vector $x_0 = (p_T, \eta, d, ...)$
2. The input becomes a memory vector;
   $$ h = f_{\text{nonlinear}}(M_l x_0^T) $$

Joining internal nodes;

1. The memory vector from the left child $h_{\text{left}}$ and the memory vector from the right child $h_{\text{right}}$ are concatenated $h_{\text{children}} = (h_{\text{left}}, h_{\text{right}})$. 
Mechanics of the RNN

The RNN will have one leaf matrix $M_l$ and one memory matrix $M_h$. It will also have a non linear function $f_{\text{nonlinear}}$. 

Creating leaves:
1. Each track forms an input vector $x_0 = (p_T, \eta, d, ...)$
2. The input becomes a memory vector; 
   \[ h = f_{\text{nonlinear}}(M_l x_0^T) \]

Joining internal nodes:
1. The memory vector from the left child $h_{\text{left}}$ and the memory vector from the right child $h_{\text{right}}$ are concatenated $h_{\text{children}} = (h_{\text{left}}, h_{\text{right}})$. 
2. A new memory vector is formed; 
   \[ h = f_{\text{nonlinear}}(M_h h_{\text{children}}) \]
The RNN will have one leaf matrix $M_l$ and one memory matrix $M_h$. It will also have a non linear function $f_{\text{nonlinear}}$.

Creating leaves;
1. Each track forms an input vector $x_0 = (p_T, \eta, d, ...)$
2. The input becomes a memory vector;

$$h = f_{\text{nonlinear}}(M_l x_0^T)$$

Joining internal nodes;
1. The memory vector from the left child $h_{\text{left}}$ and the memory vector from the right child $h_{\text{right}}$ are concatenated $h_{\text{children}} = (h_{\text{left}}, h_{\text{right}})$. 
2. A new memory vector is formed;

$$h = f_{\text{nonlinear}}(M_h h_{\text{children}})$$

The $M_l$ and $M_h$ are picked to give the correct output for any jet. This can take any number of tracks.
Those steps will be applied down the jet cluster.
<table>
<thead>
<tr>
<th>$k_t$</th>
<th>0.9195 ± 0.0009</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/A</td>
<td><strong>0.9222 ± 0.0007</strong></td>
</tr>
<tr>
<td>anti-$k_t$</td>
<td>0.9156 ± 0.0012</td>
</tr>
<tr>
<td>asc-$p_T$</td>
<td>0.9137 ± 0.0046</td>
</tr>
<tr>
<td>desc-$p_T$</td>
<td>0.9212 ± 0.0005</td>
</tr>
<tr>
<td>random</td>
<td>0.9106 ± 0.0035</td>
</tr>
</tbody>
</table>

Not only is this successful, promisingly it is more successful when used with a clustering algorithm that offers more substructure.
When an RNN joins two nodes the information in the children is embedded in a space with half as many dimensions. This is done with the same transformation each time.
When an RNN joins two nodes the information in the children is embedded in a space with half as many dimensions. This is done with the same transformation each time.

We wish to preserve information about the target vector (here jet flavour).
Implicit assertions


- When an RNN joins two nodes the information in the children is embedded in a space with half as many dimensions. This is done with the same transformation each time.
- We wish to preserve information about the target vector (here jet flavour).
- The best pairings will be nodes that relate to each other in a consistent way.
Implicit assertions


- When an RNN joins two nodes the information in the children is embedded in a space with half as many dimensions. This is done with the same transformation each time.
- We wish to preserve information about the target vector (here jet flavour).
- The best pairings will be nodes that relate to each other in a consistent way.
- Jet clustering combines tracks such that their information about jet flavour has a consistent relation.
Thank you for listening.

Backup slides and full text summary beyond this point.
Shower Jet alignment

Jet shower alignment

Jet tagging with Neural Networks
Internals of a shower
Internals of a shower
Internals of a shower
One of the goals of collide phenomenology is to determine what occurred at collision in the hard interaction. This is not directly observable. What we can detect is the lower energy decay products of the hard interaction, these form hits on the barrel of the detector.
A jet is a group of nearby hits in the detector, thought to be caused by particles with a common ancestor. Jet tagging is the process of estimating the identity of the common ancestor.
In order to understand what a jet from a particular ancestor (for example a jet from a b-quark) might look like we can generate this jet in a Monte Carlo simulation and look at the hits this produces. These simulated jets must then be used as the bases for developing tools that can estimate the flavour of real jets.
There are a number of properties of the whole jet that can be used to tag it. Each jet has a transverse momentum and an angle for example.
We can sketch how this might go with a simple boosted decision tree. Here, events are considered signal if they have been tagged by one or more b-hadrons. Two hyper parameters are available for optimisation, number of estimators and maximum tree depth. We can see that the optimal classifier is of limited complexity. This may be somewhat influenced by the lack of subtlety in interpolation.
Here is the output of the deep neural network used by CMS. It uses higher level input features and gets very good results, and functions of lower pt inputs.
Difficulty in boosted topology

Visualising the events created to train the BDT we are not surprised to see collimated jets. Lots of b-jets are boosted right onto the forward calorimeter. The pseudo rapidity cuts are removing these.
Could we use track data?

Each jet contains at least two tracks and each track has variables such as transfers momentum and angle. These could be used in a deep neural network if we trimmed and padded the track list somehow so that each jet ended up with the same number of tracks, but it is tricky to do this in a physically consistent way. Really a different mechanism is needed.
A method that has been hugely successful is to use a Deep Neural Network to process these measurements to classify the jet. This works as described (see bullet points on slides). It has the limitation that each jet must be represented by a vector of the same length. This is fine for properties of the jet as a whole, but if we wish to include properties of individual tracks in the jet it causes problems.
Tree shaped recursive neural networks are a much better match for this data structure. They were originally developed to do sentiment analysis on text and they have the form of a parse tree. The top node is the root, and this give the output of the tree, teh bottom nodes are the leaves, these take observations in. Internal nodes each have a memory state that is created from the memory of their two children. Memory propagates up the tree from the leaves to the root.
A shower

The parse tree looks strikingly like the image of a Monte Carlo shower that we saw earlier. This is encouraging, but unfortunately we only know the structure of the shower in simulated data. In real data we will only see the end result.
The jet is the measurable parallel to the shower, and it too has a tree like structure. To decide which tracks belong in the jet pairs of tracks are combined into pseudo-jets according to their energy and angular displacement and these pseudo-jets are themselves combined until an angular cutoff is reached. This combination procedure is illustrated by the orange nodes. It can be seen that the jet does not manage to get every blue observable node from the shower, these are false negatives, and it takes a couple of nodes from outside the shower, these are false positives. The tree structure the jet forms is not a mirror of the shower’s tree. We can create it in real data so it can be used for an RNN.
The particular RNN we will need is tree-like and has 2 trainable matrices. This works as described (see bullet points on slides). This can take every track as input.
Implicit assertions

Other studies have very successfully implemented this method, see louppe et al. It has been seen that using the parings of a jet clustering algorithm is more effective than randomly pairing the tracks. Using what we know of RNNs we can then infer some things about jet clustering order.

RNNs join pairs of nodes, each representing information as a point in n dimensional space to a new point in n dimensions. This means that information about the target in the children is embedded in half as many dimensions. It uses the same transformation for every pair. For this transformation to preserve information about the target the pairs must be chosen such that the relationship between their information is consistent. So when we transform them it can be preserved.

If a choice of paring is better than random parings it is because the choice has some consistency in these relationships. Jet clustering is better than random pairing, therefore track combined by jet clustering have some consistency in their relationship to information about jet flavour.