Adventures in Machine Learning

Maxim Perelstein, Cornell
BNL/Stony Brook Joint Seminar, November 7 2018
Talk 1: Boosted Top Tagging with Neural Networks

Almeida, Backovic, Cliche, Seung Lee, MP, 1501.05968
S. Choi, S. Lee, MP, 1806.01263

Talk 2: Monte Carlo Simulations with Neural Networks

Matthew Klimek, MP, 1810.11509
• Sources of boosted tops:
  
  • High-\(p_T\) tail of SM t-tbar
  
  • Extra Dimensions: KK gluon decays \(G^1 \rightarrow t\bar{t}\)
  
  • SUSY: e.g. gluino decays \(\tilde{g} \rightarrow t\bar{t}\tilde{\chi}^0\)
  
  • Spin-1/2 top partners: \(T \rightarrow tZ, th\)

• As interesting new physics scale is pushed higher by LHC bounds, boosted tops become ever more important in searches for BSM
• Cluster jets with a large cone, typically $\Delta R = 1.0$ (“fat jets”)

• Each boosted top appears as one fat jet

• Challenge: distinguish “QCD jets” (light quark/gluon-initiated) from “boosted tops”, based on “jet substructure”

• QCD rates are $>>$ top rates, so need high efficiency and good rejection power (i.e. small mis-tag rate)

\[
\text{Efficiency} = \text{Prob}(\text{top-tag}|\text{top}) \\
\text{Mis-tag} = \text{Prob}(\text{top-tag}|\text{QCD})
\]
Top Taggers

• Since the subject became popular (circa 2009), many jet-substructure observables and “tagging algorithms” have been proposed.

• Simplest observable is the jet invariant mass (corrected to remove effects of pile-up, by “pruning”, “trimming”, etc.)

• Other methods include “N-subjettiness”, template algorithms, etc.

![Graph showing mistag rate vs. top-jet tagging efficiency for different algorithms.](image-url)
Jet as an Image

• We propose a new algorithm to distinguish top-jets from QCD-jets

• We only use HCAL information

• HCAL output = digital image of the jet: each cell=pixel, energy deposit in each cell = grayscale color/intensity  \[\text{[Cogan, Kagan, Strauss, Schwarzmann, '14]}\]

• Top-jets and QCD-jets make different patterns - apply techniques from pattern recognition (a.k.a. computer vision)! Our algorithm uses Artificial Neural Network (ANN) approach
The goal of training is to choose weights that minimize this function. We use the back-propagation algorithm.

\[ y_i \rightarrow h_i^{(1)} = f(W_{ij}^{(1)} \varepsilon_j + b_i^{(1)}) \rightarrow \cdots \rightarrow h_i^{(l)} = f(W_{ij}^{(l)} h_j^{(l-1)} + b_i^{(l)}) \rightarrow Y = f(W_j^{(O)} h_j^{(l)} + b^{(O)}), \in [0, 1] \]

“Activation Function”: \[ f(z) = \frac{1}{1 + e^{-z}}. \] (sigmoid)

- ANN is a highly non-linear (but fully deterministic) map from \( N \) inputs to 1 output
- Our ANN has \( 30 \times 30 = 900 \) inputs (~0.1x0.1 HCAL cells); 2 hidden layers of 100 nodes each; and 1 output node
- There are \( \sim 100,000 \) “neurons” (connections), each with its own “weight” \( W \)
First NN: “Perceptron”
Frank Rosenblatt, Cornell, 1957
Network Training

• The weights $W$ are determined through a “training” procedure:

  • Generate large MC samples of top-jets (SM ttbar) and QCD jets (dijet)
  • “Feed” these samples to ANN, record output $Y_i$ for each jet
  • Compute the “error function” (desired outputs: $y_i=1$ for top, $y_i=0$ for QCD):
    \[
    \text{Log-loss} = -\frac{1}{N} \sum_{i=1}^{N} \left[ y_i \log(Y_i) + (1 - y_i) \log(1 - Y_i) \right].
    \]
  • Adjust weights iteratively to minimize the error function
  • Minimizing a function of 100,000 variables is not trivial, but there are well-know numerical techniques for this; we use the back-propagation algorithm, with “batch gradient descent with momentum” minimization

• Outcome: a set of weights such that $Y_i$ close to 1 for top jets, close to 0 for QCD jets

• ANN “learns” how to tell them apart, using all available info! (or: it just constructed a complicated but optimal - in some sense - observable)
Network Testing

- Once training is complete, all weights and biases are fixed
- Generate a new, independent large MC sample of top and QCD jets
- Feed these jets to ANN and see how well it can tell them apart

To discuss the performance of the ANN tagger, it is convenient to define efficiency...
Network Testing

- Once training is complete, all W’s are fixed
- Generate a new, independent large MC sample of top and QCD jets
- Feed these jets to ANN and see how well it can tell them apart

Set tagging threshold, compute efficiency and mistag rate
ANN tagger outperforms the “standard” algorithms applied to the same MC samples, especially for high-pT tops.
Some Images
Suggests that the # of “prongs” (subjets) and/or angular size are the dominant discriminants
Correlation with Other Taggers

Table 1. Correlation coefficients between the ANN score and the output of alternative taggers, in a variety of samples.

<table>
<thead>
<tr>
<th>Tagger</th>
<th>Top $p_T \in [500, 600]$</th>
<th>Top $p_T \in [1100, 1200]$</th>
<th>Dijet $p_T \in [500, 600]$</th>
<th>Dijet $p_T \in [1100, 1200]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOM</td>
<td>0.50</td>
<td>0.52</td>
<td>0.52</td>
<td>0.65</td>
</tr>
<tr>
<td>N-sub.</td>
<td>0.59</td>
<td>0.52</td>
<td>0.48</td>
<td>0.31</td>
</tr>
<tr>
<td>ATLAS</td>
<td>0.33</td>
<td>0.44</td>
<td>0.42</td>
<td>0.72</td>
</tr>
</tbody>
</table>

Fairly well correlated... but NN found some additional information not captured by others
2018 Update: Convolutional NN

Advanced NN architecture yields improved performance

[Software: MXNet]

Output: (0, 1)

Advanced NN architecture yields improved performance
Scales in High-Energy Collision

- \(10^{-16} \text{ cm}\): core event (e.g. BSM production+decay)
  - perturbative expansion in coupling constant

- \(10^{-16} \rightarrow 10^{-13} \text{ cm}\): parton shower (gluon emission/splitting)
  - perturbative expansion in \(\log(Q^2/s)\);
  - independent of new physics

- \(10^{-13} \text{ cm}\): hadronization (form pions, kaons, etc.)
  - non-perturbative QCD;
  - requires non-first-principles modeling;
  - independent of new physics

- \(10^{-1} - 10^3 \text{ cm}\): particles interact with detector

MC Challenge: simulate this multi-scale process
Is NN Learning MC Artifacts?

- NN training and validation used Monte Carlo samples of top/QCD jets
- Since NN map is complicated, it not clear what features are important for tagging, and whether these features are well-modeled by MC
- Data validation is needed (task for experimentalists)
- Necessary condition: NN output must be unaffected by soft/collinear splittings in the parton shower ("Infrared/Collinear Safety")

\[ O_n(p_1, \ldots, p_i, p_{i+1}, \ldots, p_n) \to O_{n-1}(p_1, \ldots, p_i + p_{i+1}, \ldots, p_n) \quad \text{if} \quad p_i \cdot p_{i+1} \to 0 \]

- To test IRC safety, we apply NN tagger to parton-level samples, compare output with and without an extra soft/collinear parton
Tagging Parton-Level Events

- CNN tagger trained on particle-level events was applied to parton-level top events.
- Similar output distribution indicates that most of the important information is already present in parton-level events.

![Graphs showing output distributions for particle-level and parton-level events with a shared neural network.](image-url)
Infrared/Collinear Safety

- Plot difference in NN output on parton-level events with/without extra gluon, as a function of the gluon’s “relative $p_T$”:

$$p_T^g = \left| p_g - \frac{p_g \cdot p_q}{|p_q|^2} p_q \right|$$

- Observed convergence of the NN output with/without extra gluon in the IRC limit - numerical confirmation that the observable defined by the NN is IR-safe
Infrared AND Collinear Safety

collinear limit

|ΔNN| as a function of the gluon's angular separation from its nearest quark, $R_{qg}$. Red dots show the width of the |ΔNN| distribution. Background colors indicate the relative density of events for given $R_{qg}$.

Right panel: |ΔNN| width as a function of $R_{qg}$, binned in 10 NN output intervals. The lines indicate an interpolating curve (third-order polynomial) fit to the data in each NN output bin.

to probe the convergence of the CNN output in each of these limits separately. To this end, we study two observables. The first one is the angular separation between the gluon and the nearest quark, $R_{qg}$, which goes to zero in the collinear limit, but not the soft limit. The

soft limit

|ΔNN| as a function of the longitudinal momentum ratio defined in Eq. (3.2). Red dots show the width of the |ΔNN| distribution. Background colors indicate the relative density of events for given longitudinal momentum ratio.

Right panel: |ΔNN| width as a function of longitudinal momentum ratio, binned in 10 NN output intervals. The lines indicate an interpolating curve (third-order polynomial) fit to the data in each NN output bin.
“Multi-Dimensional” Tagging

[Csaki, De Freitas, Li, Ma, MP, Shu, 1811.01961, Appendix C]

Each hidden node takes a linear combination of the inputs, specified by the weights $w^i_1$ plus a constant bias $b_1$, and transforms it by some non-linear activation function $A$. The weights and biases together comprise the parameters of the net.

Figure 6. Jet tagging performance for the first region selection for fatjet $T < 800$ GeV by ROC curve. From the top left plot, we have the jet tagging efficiency for top, Higgs (top right), Z (bottom left) and W (bottom right) jet respectively. The $y$-axis are the 1 minus false rate. In the plot labels, we display the miss identification for 50% and 80% benchmark points.

- Two Maxpooling layers;
- Classification block layers, including two linear layers with Dropout of 50% and ReLu activation function. The final linear layer classifies the jet images into 6 categories: top fat jet, Higgs fat jet, W fat jet, Z fat jet, b fat jet and light jet.

Further, each jet is assigned randomly to either the training sample or the validation sample. In each sample, jets are divided into three bins according to their $p_T$: 200 GeV $< p_{jet} T < 400$ GeV, 400 GeV $< p_{jet} T < 800$ GeV and $p_{jet} T > 800$ GeV. Jets with $p_{jet} T < 200$ GeV are discarded, since they are not expected to exhibit relevant sub-structure. The CNN is trained using the training sample, separately for each $p_T$ bin. The performance is then tested using jets in the validation sample. The tagger performance can be characterized by the Receiver Operating Characteristic curve (ROC curve). For each pair of jet classes $a$ and $b$, the ROC curve shows "tagging efficiency" (probability of correctly tagging the jet of class $a$ as $a$) on the horizontal axis, and 1 "mistag rate" (the probability of incorrectly tagging jet of class $b$ as $a$) on the vertical axis. Benchmark working points used in the collider analysis correspond to 50% and 80% efficiency for the relevant jet class (top, Higgs, or W/Z, depending on the analysis). These benchmark points are labeled on each of the curves, and the corresponding mistag rates are listed in the plot legend.

6 output nodes: top, Higgs, W, Z, b, QCD jets
Jet substructure tagging (e.g. top vs. QCD jets) is essentially an image recognition problem

Neural Network seems a natural candidate to tackle this

Simple NN outperforms existing taggers/observables in MC studies, correlated with traditional observables but contain extra information (2015)

Convolutional NN performs even better (2018)

NN output seems “IR safe” to a good approximation - MC is probably not misleading

Studies with real data in progress in ATLAS/CMS (e.g. B. Nachman et.al.)

“Multi-dimensional tagging” (top/Higgs/W/Z/QCD jets) is also possible
Talk 1: Boosted Top Tagging with Neural Networks
Almeida, Backovic, Cliche, Seung Lee, MP, 1501.05968
S. Choi, S. Lee, MP, 1806.01263

Talk 2: Monte Carlo Simulations with Neural Networks
Matthew Klimek, MP, 1810.11509
MC Simulation/Integration

• Monte Carlo Problem: Given a function $f(y)$, such that $f(y) \geq 0$, generate a set of “random” points $\{y_i\}$ with density proportional to $f(y)$.

• In particle physics, typically $y$=phase space points, $f(y)$=differential cross section or decay rate, $\{y_i\}$=Monte Carlo sample (“pseudo-experiment”) 

• Most Naive MC algorithm: randomly select points in 2D box, discard the points with $z > f(y)$.

• Fraction of points that are actually used = “unweighting efficiency”: $\epsilon(y) = \frac{f(y)}{f_{max}}$

integration: $\int f(y) dy = f_{max} \int \epsilon(y) dy$

Problem: Resonances, Collinear/Infrared Singularities

$\epsilon \ll 1$

In modern applications, $f(y)$ is often numerically expensive to evaluate (e.g. NNLO - may require numerical integrations)
Importance Sampling

- Classic solution: construct a number of “bounding boxes” in yz plane, covering the function’s domain, with heights adjusted to correspond to local values of f(y)

- Classic implementation: VEGAS [Lepage, 1978]

- Divide the domain into N bins, roughly compute “weight” = \( \int_{\text{bin}} e(y) \, dy \) in each bin

- Iteratively adjust bin boundaries until each bin contains the same weight

- Simulation: choose a bin at random (equal probabilities), then follow Naive algorithm in that bin. Repeat.

Construct a piecewise-constant approximation to f(y), then sample from that distribution.
Importance Sampling as a Map

- Importance sampling can also be described as a map from “input space” $x$ to “target space” $y$
- Randomly choose $x \in [0, 1]$ (uniform distribution)
- Deterministic, piecewise-linear map $x \rightarrow y(x)$
- Equivalent to “pick a box + random point within the box”
- Unweighting: keep the point with probability $P(y) = f(y) \left| \frac{dy}{dx} \right|$
MC with Neural Networks

- Idea: Generalize importance sampling from piecewise-linear to nonlinear maps

- Simulation would be 100% efficient if we found a nonlinear map such that

\[
\left| \frac{dy}{dx} \right|^{-1} = f(y)
\]

- Generalization to functions in N dimensions (same dimensionality for input and target spaces, =dimensionality of phase space)

\[
J = \det \frac{\partial y_i}{\partial x_j}, \quad |J|^{-1} = f(y)
\]

- Universal Approximation Theorem: under mild assumptions, a neural network can approximate any continuous functional map \( \mathcal{I}_N \rightarrow \mathcal{I}_N \) (where \( \mathcal{I}_N \) is an N-dimensional hypercube)

[Cybenko, '89; Hornik, '91]

- This makes a NN a natural choice to implement nonlinear importance sampling
MC with Neural Networks

• Error function: Kullback-Leibler divergence between $|J|^{-1}$ and $f(y)$

$$D_{KL}[p_y(y); f(y)] = \int p_y(y) \log \frac{p_y(y)}{f(y)} dy$$

• Training: generate a batch of 100 points, compute $D_{KL}$, adjust weights, iterate

\[
\begin{align*}
\text{input } I & \xrightarrow{\Delta w \propto -\nabla_w L(w)} \text{y}_w(x) \\
\text{y}_w(x) & \xrightarrow{L(w) = D_{KL}(p_y(y_w(x)); f(y_w(x)))} \text{target } T
\end{align*}
\]
**Output Functions**

- An important subtlety is the choice of output function (=activation function for the last layer)

\[
S(x) = \frac{1}{1 + e^{-x}}
\]

- "soft clipping function":

\[
SC(x) = \frac{1}{p} \log \left( \frac{1 + e^{px}}{1 + e^{p(x-1)}} \right)
\]

With this choice, a traditional ELU activation function is sufficient.

---

3 Layers, 128 nodes

- ELU/SC
- Sinh/Sigmoid
- Bendavid 1707.00028
Sample Applications

- Simulate 3-body decay of a scalar $X$, with a resonance $Y$

Choose phase-space coordinates $m_{23}, \theta_{1(23)}$

Simulated with $\Gamma_Y/m_Y = 10^{-2}, 10^{-3}, 10^{-4}$

Achieved unweighting efficiency 30-70%, depending on resonance width

MadGraph (off-the-shelf) efficiency: 6%
Sample Applications

• Simulate 3-body decay of a scalar $X$, with resonances in two channels

\[ X \rightarrow Y_1 \rightarrow \gamma \rightarrow 3 \]
\[ X \rightarrow Y_2 \rightarrow \gamma \rightarrow 3 \]

• NN was able to learn both the feature aligned with coordinate axis, and the feature with complicated shape in these coordinates

• In contrast, VEGAS needs each feature to be aligned with a coordinate axis (coordinate choice handled separately by “multi-channeling”)

For multi-dimensional integrals, VEGAS needs any sharp feature to be aligned with a grid axis. What about matrix elements that have multiple, non-orthogonal sharp features? (E.g. multiple resonances)

This is currently handled with multi-channel integration: Define multiple grids, each aligned with one feature, and sample from all. Potential slow, and relative weights among grids must be tuned.

NN output

VEGAS grid/output
Sample Applications

• A more realistic example: \( e^+ e^- \rightarrow q\bar{q}g \)

\[
\frac{d\sigma}{dm^2_{qg} dm^2_{\bar{q}g}} \propto \frac{(s - m^2_{qg})^2 + (s - m^2_{\bar{q}g})^2}{m^2_{qg} m^2_{\bar{q}g}},
\]

• Soft/collinear singularities need to impose kinematic cuts

• Simple rectangular cuts aligned with target-space coordinates can be simply handled by redefining the target space boundaries

• In practice we need to be able to handle more general cuts:

\[
Y \geq Y_{cut} \quad \text{where} \quad Y = Y(y_1, \ldots, y_N)
\]

• Naively, we could just replace \( f(y) \rightarrow \theta(Y(y) - Y_{cut}) f(y) \)

• However NN target function must be differentiable! So we opt for

\[
f(y) \rightarrow \kappa(Y(y) - Y_{cut}) f(y) \quad \text{with} \quad \kappa(x) = \begin{cases} 
1 & x > x_{cut} \\
(x/x_{cut})^n & x < x_{cut}
\end{cases}
\]
Sample Applications

- A more realistic example: $e^+e^- \rightarrow q\bar{q}g$

$$\frac{d\sigma}{dm_{qg}^2 dm_{\bar{q}g}^2} \propto \frac{(s - m_{qg}^2)^2 + (s - m_{\bar{q}g}^2)^2}{m_{qg}^2 m_{\bar{q}g}^2}.$$  

- In this example, we used $n=8$.

- Unweighting efficiency is 70% (vs. 4% for off-the-shelf MadGraph)
Talk 2: Conclusions/Outlook

- **Neural Network** seems a natural candidate to realize "nonlinear importance sampling"

- With a bit of tweaking (e.g. proprietary "soft clipping" output function), we got it to work

- Can handle **resonances**, in a nicely coordinate-choice-independent way

- Can handle **soft/collinear enhancements**, generic kinematic cuts

- High **unweighting efficiency** achieved in all examples

- This may be a crucial advantage in situations when matrix element is computationally expensive to evaluate

- Next: Integration with automated Matrix Element calculators

- Next-to-next: Parton showers? NLO?