EMERGING TRENDS IN
SOFTWARE FOR STATISTICS
AND MACHINE LEARNING

@KyleCranmer
New York University
Department of Physics
Center for Data Science
A PROBLEM I’M GRAPPLING WITH

Approaches to statistical inference that natively integrate Monte Carlo simulation and machine learning

<table>
<thead>
<tr>
<th>Hypothesis Testing</th>
<th>Classification</th>
<th>$F : X \rightarrow {0, 1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Likelihood</td>
<td>Regression</td>
<td>$F : X \rightarrow \Theta$</td>
</tr>
<tr>
<td>Likelihood-based Inference</td>
<td>Conditional Density Estimation / Variational Inference</td>
<td>$F : X, \Theta \rightarrow \mathbb{R}$</td>
</tr>
<tr>
<td>Data Modeling</td>
<td>Conditional Generative Models</td>
<td>$F : \Theta \rightarrow X$</td>
</tr>
</tbody>
</table>
Collaborative Statistical Modeling

\[ f_{\text{tot}}(D_{\text{sim}}, G | \alpha) = \prod_{c \in \text{channels}} \left[ \text{Pois}(n_c | \nu_c(\alpha)) \prod_{e=1}^{n_c} f_c(x_{ce} | \alpha) \right] \cdot \prod_{p \in S} f_p(a_p | \alpha_p) \]
THE PLAYERS

\[ p(x|\theta, \nu) \]

\( \theta \)
parameters of interest

\( \nu \)
nuisance parameters

PREDICTION

forward modeling
generation
simulation

(z: latent variables)

INFERENCEx

inverse problem
measurement
parameter estimation

observed data
covariates
simulated data
Probabilistic programming frameworks

RooFit serves us well, but shows limits in terms of **scalability**.

Using a data flow graph framework, RooFit would be **distributed**, **GPU-enabled** and automatically **differentiable**.

Feasibility? Certainly **within reach**! As illustrated by our tentative proof-of-concepts carl.distributions [Gilles Louppe] and tensorprob [Igor Babuschkin, now at DeepMind]. See also Edward.

---

**Edward**

A library for probabilistic modeling, inference, and criticism.

Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.

It supports **modeling** with

---

**Dustin Tran**

Ph.D. Student
Columbia University
dustin@cs.columbia.edu (@dustintran)
http://dustintran.com

---

**Matthew Feickert**

High Energy Physics Ph.D. Candidate
Southern Methodist University.matthew.feickert@cern.ch or mfeickert@smu.edu
GitHub: matthewfeickert @HEPfeickert
QCD-INSPIRED RECURSIVE NEURAL NETWORKS

Work with Gilles Louppe, Kyunghyun Cho, Cyril Becot (arXiv:1702.00748)

- Use sequential recombination jet algorithms to provide network topology (on a per-jet basis)
- Example of ML models embed with physics knowledge
- Need a dynamic computation graph, complicates efficient batch training
QCD-INSPIRED RECURSIVE NEURAL NETWORKS

Work with Gilles Louppe, Kyunghyun Cho, Cyril Becot
(arXiv:1702.00748)

• Use sequential recombination jet algorithms to
  provide network topology (on a per-jet basis)
• Example of ML models embed with physics
  knowledge
• Need a dynamic computation graph, complicates
  efficient batch training

FIG. 3. Jet classification performance for various input representations of the RNN classifier, using $k_T$ topologies for the embedding. The plot shows that there is significant improvement from removing the image processing step and that significant gains can be made with more accurate measurements of the 4-momenta.

FIG. 4. Jet classification performance of the RNN classifier based on various network topologies for the embedding (particles scenario). This plot shows that topology is significant, as supported by the fact that results for $k_T$, C/A and desc-$p_T$ topologies improve over results for anti-$k_T$, a$-p_T$ and random binary trees. Best results are achieved for C/A and desc-$p_T$ topologies, depending on the metric considered.

Further supported by the poor performance of the random binary tree topology. We expected however that a simple sequence (represented as a degenerate binary tree) based on ascending and descending $p_T$ ordering would not perform particularly well, particularly since the topology does not use any angular information. Surprisingly, the simple descending $p_T$ ordering slightly outperforms the RNNs based on $k_T$ and C/A topologies. The descending $p_T$ network has the highest $p_T$ 4-momenta near the root of the tree, which we expect to be the most important.

We suspect this is the reason that the descending $p_T$ outperforms the ascending $p_T$ ordering on particles, but this is not supported by the performance on towers. A similar observation was already made in the context of natural languages [24–26], where tree-based models have at best only slightly outperformed simpler sequence-based networks. While recursive networks appear as a principled choice, it is conjectured that recurrent networks may in fact be able to discover and implicitly use recursive compositional structure by themselves, without supervision.

d. Gating
The last factor that we varied was whether or not to incorporate gating in the RNN. Adding gating increases the number of parameters to 48,761, but this is still about 20 times smaller than the number of parameters in the MaxOut architectures used in previous jet image studies. Table I shows the performance of the various RNN topologies with gating. While results improve significantly with gating, most notably in terms of $\mathcal{R} = 50\%$, the trends in terms of topologies remain unchanged.

e. Other variants
Finally, we also considered a number of other variants. For example, we jointly trained a classifier with the concatenated embeddings obtained over $k_T$ and anti-$k_T$ topologies, but saw no significant performance gain. We also tested the performance of recursive activations transferred across topologies. For instance, we used the recursive activation learned with a $k_T$ topology when applied to an anti-$k_T$ topology and observed a significant loss in performance. We also considered particle and tower level inputs with an additional trimming preprocessing step, which was used for the jet image studies, but we saw a significant loss in performance. While the trimming degraded classification performance, we did not evaluate the robustness to pileup that motivates trimming and other jet grooming procedures.

B. Infrared and Collinear Safety Studies
In proposing variables to characterize substructure, physicists have been equally concerned with classification performance and the ability to ensure various theoretical properties of those variables. In particular, initial work on jet algorithms focused on the Infrared-Collinear (IRC) safe conditions:

• Infrared safety. The model is robust to augmenting $e$ with additional particles $\{v_{N+1} \hookrightarrow \ldots \hookrightarrow v_{N+K}\}$
EVENT EMBEDDINGS

Jointly optimize jet embedding → event embedding → classifier

---

Appendix A: Gated recursive embedding of jets

The recursive activation proposed in the previous section suffers from two critical issues. First, it assumes that left-child, right-child and local node information \( h_k^L, h_k^R, u_k \) are all equally relevant for computing the new activation, while only some of this information may be needed and selected. Second, it forces information to pass through several levels of non-linearities and does not allow to propagate unchanged from leaves to root. Addressing these issues and generalizing from [5–7], we propose to recursively define a recursive activation equipped with reset and update gates as follows:

\[
\begin{align*}
    h_k &= \begin{cases} 
    u_k & \text{if } k \text{ is a leaf} \\
    z_H \tilde{h}_k + z_L h_k^L + z_R h_k^R + z_N u_k & \text{otherwise}
    \end{cases} \\
    u_k &= (W_{u\theta} + b_{u\theta}) \\
    o_k &= \begin{cases} 
    v_i(k) & \text{if } k \text{ is a leaf} \\
    o_k^L + o_k^R & \text{otherwise}
    \end{cases} \\
    \tilde{h}_k &= \begin{cases} 
    0 & \text{if } k \text{ is a leaf} \\
    W_{\tilde{h}\theta} \begin{bmatrix} h_k^L \ h_k^R \ u_k \end{bmatrix} + b_{\tilde{h}\theta} & \text{otherwise}
    \end{cases} \\
    z_H &= \text{softmax}(W_{z\theta} \begin{bmatrix} \tilde{h}_k \ \tilde{h}_k^L \ \tilde{h}_k^R \ \tilde{h}_k^N \end{bmatrix} + b_{z\theta}) \\
    r_L &= \text{sigmoid}(W_{r\theta} \begin{bmatrix} h_k^L \ h_k^R \ u_k \end{bmatrix} + b_{r\theta})
\end{align*}
\]

where \( W_{\tilde{h}\theta}, b_{\tilde{h}\theta} \), \( W_{z\theta}, b_{z\theta} \), \( W_{r\theta}, b_{r\theta} \) form together the shared parameters to be learned, \( \text{ReLU} \) denotes the ReLU activation function and \( \text{element-wise multiplication} \) denotes the element-wise multiplication. Intuitively, the reset gates \( r_L, r_R, r_N \) control how to actively select and then merge the left-child embedding \( h_k^L \), the right-child embedding \( h_k^R \) and the local node information \( u_k \) to form a new candidate activation \( \tilde{h}_k \). The final embedding \( h_k \) can then be regarded as a...
In the last few months:
TensorFlow Fold & PyTorch

Efficient dynamic generation of MC to estimate exclusion contour [Gilles Louppe & Lukas Heinrich]
In the last few months:
TensorFlow Fold & PyTorch

Efficient dynamic generation of MC to estimate exclusion contour
[Gilles Louppe & Lukas Heinrich]
The modern AI/ML software stack

Recent switch to

- Numerical computations with data flow graphs
  - TensorFlow, Theano, MXNet, etc
  - Support for CPUs and GPUs out of the box.
  - Automatic differentiation
  - Enable new ways of thinking (model composition, learning to learn, etc)

- Probabilistic programming languages
  - Stan, Anglican, Edward, etc

**Recommendation.** The next generation of physics software for high-level analysis should take notice and inspiration from the AI/ML community.