FERMILAB-SLIDES-22-031-QIS

Constructing Sensitive and Robust Collider Observables with Machine Learning

Prasanth Shyamsundar, Fermilab Quantum Institute

Rutgers NHETC Seminar, Rutgers University

May 17, 2022

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

Based on...

Deep-Learned Event Variables for Collider Phenomenology, arXiv:2105.10126 [hep-ph]

Doojin Kim, K.C. Kong, Konstantin Matchev, Myeonghun Park, and Prasanth Shyamsundar

FERMILAB-PUB-21-244-QIS
MI-TH-2111

Deep-Learned Event Variables for Collider Phenomenology

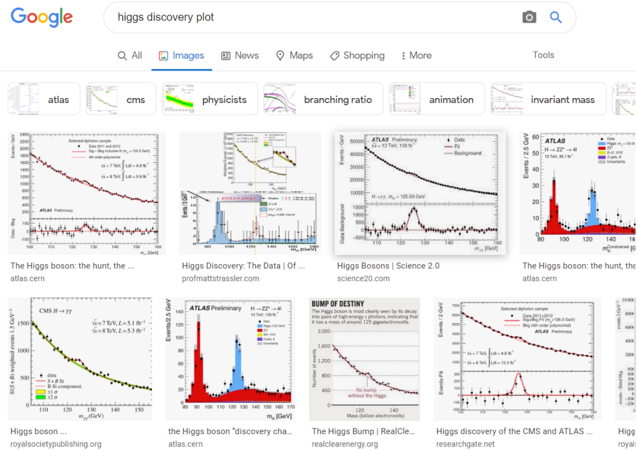
Doojin Kim,[1,*] Kyoungchul Kong,[2,†] Konstantin T. Matchev,[3,‡]
Myeonghun Park,[4,5,6,§] and Prasanth Shyamsundar[7,¶]

[1] *Mitchell Institute for Fundamental Physics and Astronomy, Department of Physics and Astronomy, Texas A&M University, College Station, TX 77843, USA*
[2] *Department of Physics and Astronomy, University of Kansas, Lawrence, KS 66045, USA*
[3] *Institute for Fundamental Theory, Physics Department, University of Florida, Gainesville, FL 32611, USA*
[4] *Institute of Convergence Fundamental Studies, Seoultech, Seoul, 01811, Korea*
[5] *School of Physics, KIAS, Seoul 02455, Korea*
[6] *Center for Theoretical Physics of the Universe, Institute for Basic Science, Daejeon 34126 Korea*
[7] *Fermilab Quantum Institute, Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*

The choice of optimal event variables is crucial for achieving the maximal sensitivity of experimental analyses. Over time, physicists have derived suitable kinematic variables for many typical event topologies in collider physics. Here we introduce a deep learning technique to design good event variables, which are sensitive over a wide range of values for the unknown model parameters. We demonstrate that the neural networks trained with our technique on some simple event topologies are able to reproduce standard event variables like invariant mass, transverse mass, and stransverse mass. The method is automatable, completely general, and can be used to derive sensitive, previously unknown, event variables for other, more complex event topologies.

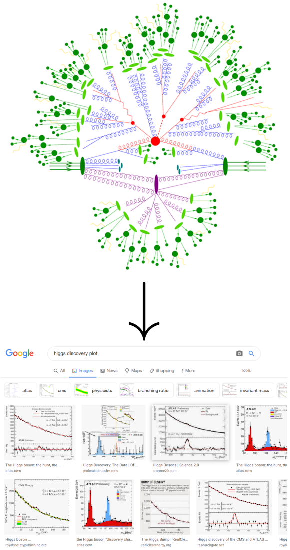
21 May 2021

What are event variables?



- Event observables, roughly, are the variables which get analyzed (histogrammed, curvefitted, etc.) in collider analyses.
- They play an important role in high energy physics.

What are event variables?



- Collider experimental data is extremely high-dimensional. At the level of raw electronic readouts from the detector, event information can be $\sim 10^6$-dimensional.
- We don't analyze this raw information directly. The dimensionality of the data is reduced over various stages of reconstruction
  - Interpret electronic readouts as detector hits
  - Reconstruct the particles that reach the detector from the detector hits
  - Perform jet clustering to get a handle on parton-level objects.
  - **Condense the reconstructed parton level objects into low-dimensional observables.**
- They are typically human-engineered, using our domain knowledge of the underlying processes.
- Examples include invariant mass, transverse mass m_T, m_{T2}, etc.

Why use event variables?

- Curse of dimensionality — Analyzing high-dimensional data is difficult.
- Event variables efficiently retain information relevant to the analysis.
- Their distribution is sensitive to the underlying physics
  - presence of signal
  - values of unknown model parameters
- Easier to validate simulation models in low-dimensional dataspace.

  (more on this later)

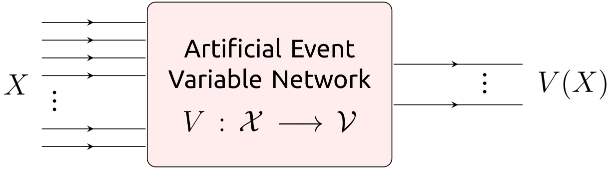
Machine Learning

- Machine learning has become an important part of collider physics analyses.
- Classifiers and taggers — one of the earliest and most common ML applications in collider physics.
  - Signal–Background classifier scores can be used in event selection make the dataset signal-rich.
  - Taggers can tag jets as being top or b jets
  - Machine-learning classifier scores (from 0 to 1) have also beeen used directly as analysis variables.
- Other ML applications range from performing object reconstruction, to using machines to perform simulations.
- **But so far there have been no ML approach to invent (discover?) new event variables.** (directly using ML outputs as event variables not withstanding)

  This work changes that

Synthesizing event variables with machine learning

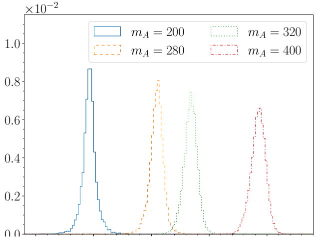
- **How to model an event variable with a neural network?** Easy!

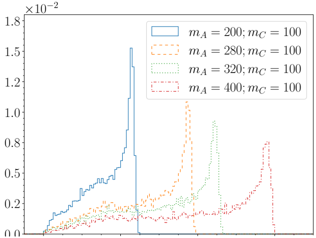


- **How to train such a network?** Not straightforward!
- **One approach:** Perform a full analysis with the variable V using simulated data. Use the (projected) sensitivity of the analysis as a performance metric to optimize.
  Difficulty: Performing an analysis for each training step could be highly inefficient.

Need a fast way to evaluate the usefulness of the variable V.

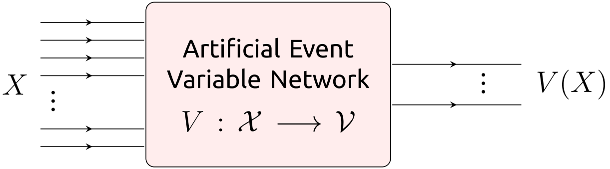
Characteristics of event variables





- Their distribution is sensitive to the value of underlying parameters.
  They contain **information** about the unknown parameters.
- The same variable works over a **range of unknown masses.**
  Contrast with typical approaches, where a different neural network is trained for each “study point”.

The beginnings of a training strategy…



- **Goal:** Train the network to be useful over a range of parameter values.
- One interpretation of the goal:
  - Train the network so that V carries a lot of information about the underlying unknown parameters Θ.
  - Mass variables, for example, carry a lot of information about the underlying mass parameters—that's why they are used in measurement of m_{t}, m_W, m_Z, etc.
- **How:**
  - Design a task to be performed using V.
  - Train the network to perform the task well.

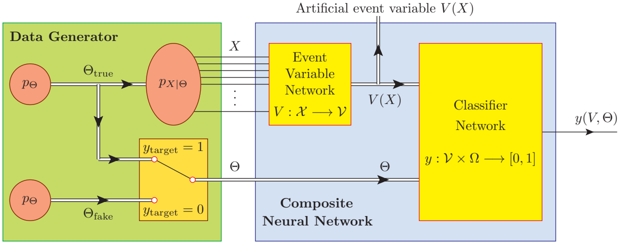
  This is the principle behind representation learning techniques like `word2vec`.

Some information theory...

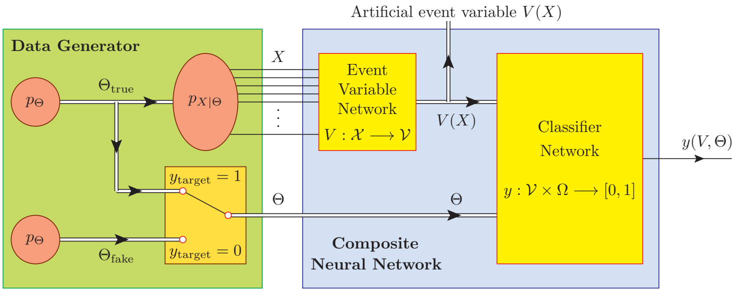
- $p_\Theta \equiv$ Prior on the unknown parameters Θ
  $p_{X|\Theta} \equiv$ Dist. of the event X conditional on Θ, $V(X) \equiv$ Event variable
- Mutual information between the event variable V and parameter Θ:

$$I(V;\Theta) = \int dv \int d\theta \; p_{(V,\Theta)}(v,\theta) \;\; \ln\left[\frac{p_{(V,\Theta)}(v,\theta)}{p_V(v)\, p_\Theta(\theta)}\right]$$

- $I(V;\Theta)$ is the KL divergence from $p_V \otimes p_\Theta$ to $p_{(V,\Theta)}$. It captures their distinguishability.
- **Idea:** Train V so that the two distributions are highly distinguishable.



Blueprint of the training strategy



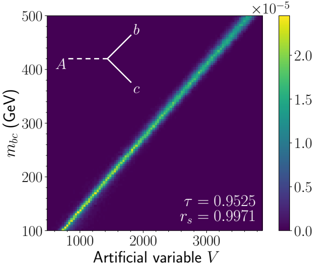
- **Training data:** $(X, \Theta) \sim p_X \otimes p_\Theta$ under class 0; $p_{(X,\Theta)}$ under class 1
- **Event variable network:** Transforms X to V.
- **Auxiliary Classifier Network:**
  Input is $(V, \Theta) \sim p_V \otimes p_\Theta$ under class 0; $p_{(V,\Theta)}$ under class 1.
- Train the composite network as a classifier.
  - Auxiliary classifier distinguishes between $p_V \otimes p_\Theta$ and $p_{(V,\Theta)}$.
  - Event Variable Network **makes them highly distinguishable**.

(actualizing the idea from the last slide)

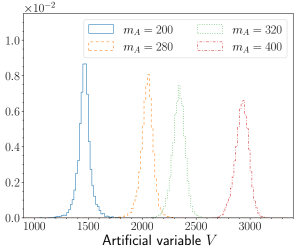
Example 1: Invariant mass

- $A \to b, c$ (both massless and visible)
- $\Theta \equiv m_A$
- m_A is chosen uniformly in the range $(100, 500)$.
- X is the 4-momenta of b and c
  $\dim(X) = 8$; $\dim(V) = 1$
- We want the event variable to work
  even when A is not at rest, and
  for different (m_B, m_C) values
  - E_A is uniformly sampled from $(m_A, 1500)$.
  - Direction of A is chosen uniformly at random.
- We sample events from the phasespace, and train the event variable network.
- The machine ends up learning m_{bc}!

What has the machine learned?



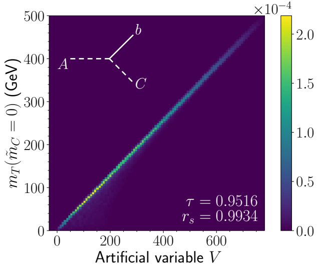
Event variable in action



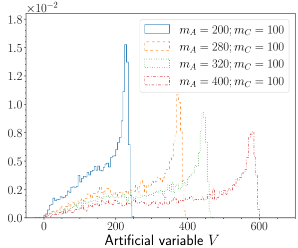
Example 2: Transverse mass m_T

- $A \to b$(massless, visible), C(invisible)
- $\Theta \equiv (m_A, m_C)$ chosen from an appropriate prior.
- $\dim(X) = 6$; $\dim(V) = 1$
- Other parameters
  - E_A is uniformly sampled from $(m_A, 1500)$.
  - Direction of A is chosen along the $\pm z$-axis.
- The machine ends up learning m_T!

What has the machine learned?



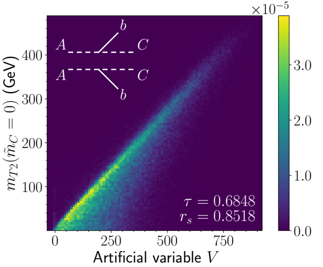
Event variable in action



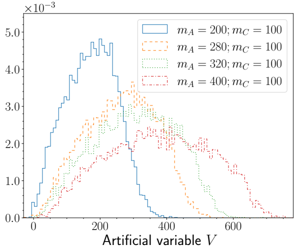
Example 3: Stransverse mass m_{T2}

- $pp \to A_1, A_2$
  $A_i \to b_{i\,\text{(massless, visible)}}, C_{i\,\text{(invisible)}}$
- $\Theta \equiv (m_A, m_C)$ chosen from an appropriate prior.
- $\dim(X) = 10$; $\dim(V) = 1$
- Other parameters
  - m_{pp} is sampled from $(2m_A, 1500)$.
  - E_{pp} is sampled from $(m_{pp}, 2500)$.
  - Direction of pp is chosen along the $\pm z$-axis.
- Unlike invariant and tranverse mass, stransverse mass m_{T2} does not have singular features, and isn't guaranteed to be optimal for the task.

What has the machine learned?



Event variable in action



What's next?

Now, we can go after previously unknown event variables.

- Event topologies for which the best kinematic variables are yet to be discovered.
- Humans are good at finding the good 1d kinematic event variables. What's the best 2d or 3d event variable? (excluding obvious cases like two resonant decays)
- Variables that incorporate more physics that just the event kinematics—qcd effects, parton distribution functions, etc.
- Event variables that take non-traditional attributes as inputs, e.g., b-tag score, jet energy resolution.
- Explore other ways to quantify the "usefulness" of an event variable (ongoing work).

Why is this work significant?

A valid argument against our approach:

- This is cool and all. But why limit machine learning to producing event variables? Won't that just bottleneck the amount of information used by the analyses?
- In many ways, this approach is the anti-thesis of end-to-end machine learning.
- Why not use machines to directly analyze high-dimensional data, like many other techniques?

The argument for our approach:

Robustness & reliability of the resulting analyses.

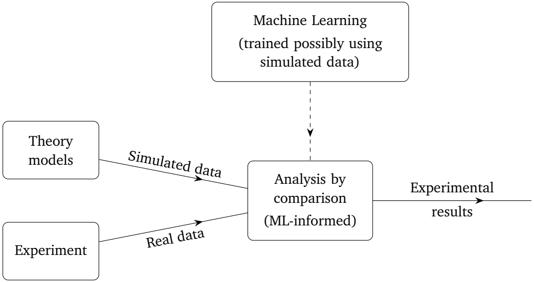
Why is this work significant?

Two major challenges to the robustness of ML-based HEP analyses:

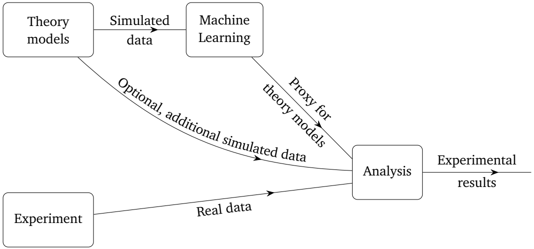
- Uncertainty quantification
- Errors in simulation models

Let's look at these one at a time, and see how they affect our work.

Uncertainty quantification by type of ML usage



- Uncertainty quantification is optional.
- Errors and uncertainties in ML lead to **suboptimal sensitivity**.



- Uncertainty quantification is critical.
- Errors and uncertainties in ML lead to **incorrect results**.

Our approach falls on the left.

Errors in simulation models

- HEP analyses are based on simulations.
- Several aspects of our simulation models are based on heuristics.
- We know that our simulation models are inaccurate, even after accounting for *known* systematics.

Q: How does one trust data analyses performed using inaccurate simulations?
A: Through meticulous data validation, performed at various stages of the analysis.

Robustness of an analysis = Interpretability or “validatability”?

- Data validation is more science than math (more so than hypothesis testing and parameter measurement).
- Data validation is analysis specific. (black board discussion...)

  We **can perform signal search and parameter measurement** with high-dim data. But we **cannot perform data validation** in high dimensions.
- In my opinion, the robustness of a collider analysis technique is more strongly related to

  whether the simulations can be meaningfuly validated *for that analysis*

  than to

  whether we can interpret or explain the analysis variable.

  (unless we have an unambiguous, smoking gun analysis)
- Analyses using low-dimensional observables, that aren’t tuned to specific “study points” are easier to meaningfully validate. Our technique fits the bill.

Summary and Outlook

- We have a technique for training neural networks into being good event variables.
- The network ends up learning traditional variables like invariant mass, m_T, and m_{T2} in the appropriate event topologies.
- **Works over a range of parameter values**
- **Trivially generalizable (in the ML sense)**
  - Variables are derived using phasespace generated events.
  - Yet, they are useful in the analysis of real datasets — just need suitable simulations to create templates.
- **The resulting technique will offer a degree of robustness against unknown modeling errors.**

Thank you!