Artificial Event Variables for Collider Analyses

Prasanth Shyamsundar, Fermilab Quantum Institute

Phenomenology 2021 Symposium

May 24, 2021

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

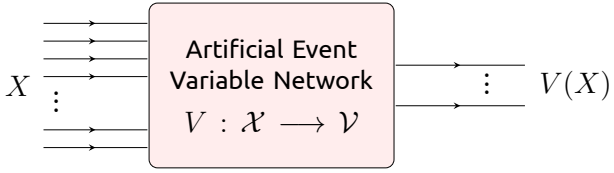
Doojin Kim,[1,*] Kyoungchul Kong,[2,†] Konstantin T. Matchev,[3,‡]
Myeonghun Park,[4,5,6,§] and Prasanth Shyamsundar[7,¶]

[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.

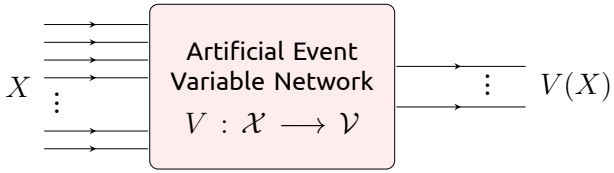
Synthesizing event variables with machine learning

- **What are event variables?**
  Dimensionality reducing transformations. Eg: invariant mass, transverse mass
  High-dimensional event description $\longrightarrow$ low-dimensional variables
- **Why use event variables?**
  - Curse of dimensionality. Easier to analyze low-dimensional data.
  - Sensitive to presence of signal or parameter value over a range of values of unknown parameters.
  - Easier to validate simulation models in low-dimensional dataspace.
- **How to model an event variable with a neural network?** Easy!



- **How to train such a network?** Not straightforward!

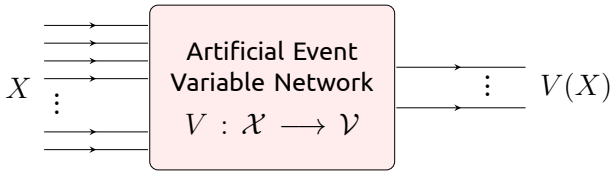
Distinction from other ML approaches



The goals here are very different from common goals in HEP-ML.
V needs to be "useful" over a range of parameter values.

- V is not a sig-bkg classifier, which typically works for chosen "study point".
- Parameterized networks map each event onto a function, e.g., $X \to \mathcal{L}(\Theta|X)$.
  We're not doing this.
- We aren't training V to match or predict some monte-carlo truth.
  It's not about finding the right distance metric from V to a target.

The beginnings of a training strategy…



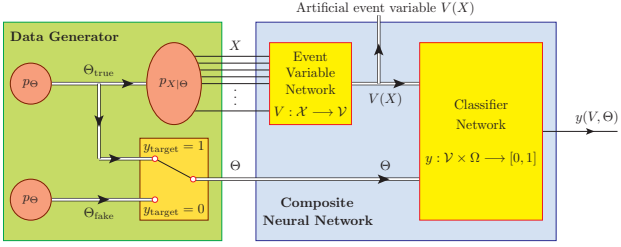
- **Goal:** Train the network to be useful over a range of parameter values.
- One interpretation of the goal:
  - Train the network so that V carries a lot of **information** about the underlying unknown parameters Θ.
  - Mass variables, for example, carry a lot of information about the underlying mass parameters—that's why they are used in measurement of m_t, m_W, m_Z, etc.
- **How:**
  - Come up with a task to be performed using V.
  - Train the network to perform the task well.

Some information theory…

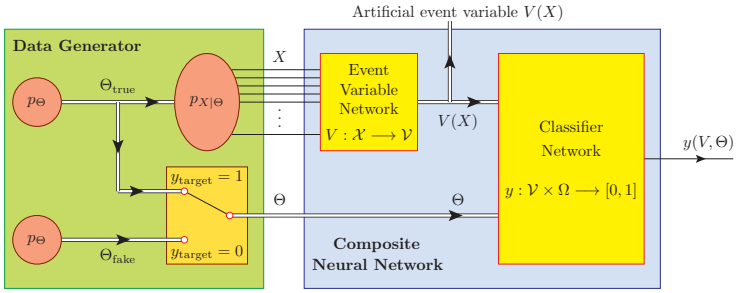
- $p_\Theta \equiv$ Prior on the unknown parameters Θ
  $p_{X|\Theta} \equiv$ Dist. of the event X conditional on Θ, $V(X) \equiv$ Event variable
- Mutual information between the event variable V and parameter Θ:

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

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



Blueprint of the training strategy



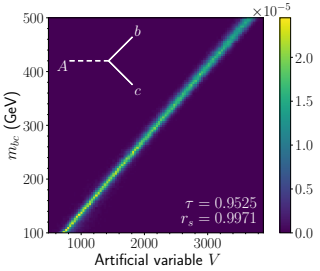
- **Training data:** $(X, \Theta) \sim p_X \otimes p_\Theta$ under class 0; $p_{(X,\Theta)}$ under class 1
- **Event variable network:** Transforms X to V.
- **Auxiliary Classifier Network:**
  Input is $(V, \Theta) \sim p_V \otimes p_\Theta$ under class 0; $p_{(V,\Theta)}$ under class 1.
- Train the 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)

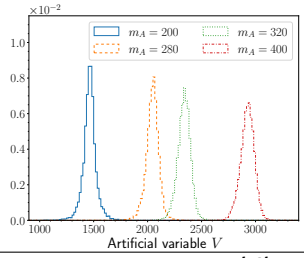
Example 1: Invariant mass

- $A \to b, c$ (both massless and visible)
- $\Theta \equiv m_A$
- m_A is chosen uniformly in the range $(100, 500)$.
- $\dim(X) = 8$; $\dim(V) = 1$
- 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?



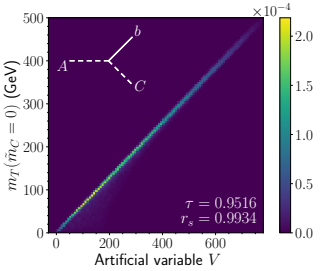
Event variable in action



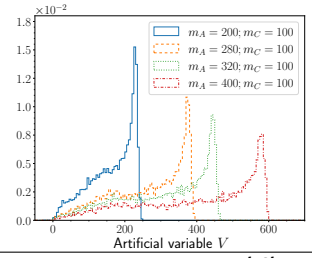
Example 2: Transverse mass m_T

- $A \to b_{\text{(massless, visible)}}, C_{\text{(invisible)}}$
- $\Theta \equiv (m_A, m_C)$ 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?



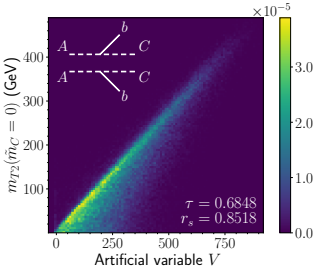
Event variable in action



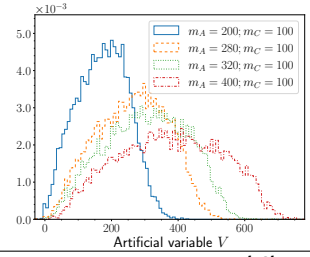
Example 3: Stransverse mass m_{T2}

- $pp \to A_1, A_2$
  $A_i \to b_{i\,\text{(massless, visible)}}, C_{i\,\text{(invisible)}}$
- $\Theta \equiv (m_A, m_C)$ 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



Summary and Outlook

Summary:

- 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.

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.
- Explore other notions of "usefulness" of an event variable.

Event variables vs typical ML approaches

Cons of the event variable approach:

- Dimensionality reduction $\implies$ loss of information $\implies$ loss of sensitivity.
  We're not going to reach the Cramér–Rao bound with event variables.

Pros of the event variable approach:

- Interpretable, by direct comparison against human-engineered features.
- **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.
- **Relatively robust against unknown modeling errors**
  - Goodness-of-fit tests (only) exist for low-dimensional data.
  - Easier to perform meaningful validation of the **joint-distribution** of low-dim data, than high-dim data… Control region validation, post-fit goodness-of-fit test, etc.
  - Dangerous, unknown modeling errors can be expected to be caught.

Event variables vs typical ML approaches

Not going after ultimate sensitivity

Cons of the event variable approach:

- Dimensionality reduction $\Longrightarrow$ loss of information $\Longrightarrow$ loss of sensitivity.
  We're not going to reach the Cramér–Rao bound with event variables.

Pros of the event variable approach:

- Interpretable, by direct comparison against human-engineered features.
- **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.
- **Relatively robust against unknown modeling errors**
  - Goodness-of-fit tests (only) exist for low-dimensional data.
  - Easier to perform meaningful validation of the **joint-distribution** of low-dim data, than high-dim data… Control region validation, post-fit goodness-of-fit test, etc.
  - Dangerous, unknown modeling errors can be expected to be caught.

Event variables vs typical ML approaches

Not going after ultimate sensitivity

Cons of the event variable approach:

- Dimensionality reduction $\Longrightarrow$ loss of information $\Longrightarrow$ loss of sensitivity.
  We're not going to reach the Cramér–Rao bound with event variables.

Pros of the event variable approach:

- Interpretable, by direct comparison against human-engineered features.
- **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.
- **Relatively robust against unknown modeling errors**
  - Goodness-of-fit tests (only) exist for low-dimensional data.
  - Easier to perform meaningful validation of the **joint-distribution** of low-dim data, than high-dim data… Control region validation, post-fit goodness-of-fit test, etc.
  - Dangerous, unknown modeling errors can be expected to be caught.

We're creating sensitive, robust, and reliable blackboxes

Thank You!