Some new developments in Machine Learning for High Energy Physics

Prasanth Shyamsundar, Fermi National Accelerator Laboratory

Anomalies 2021, Indian Institute of Technology Hyderabad

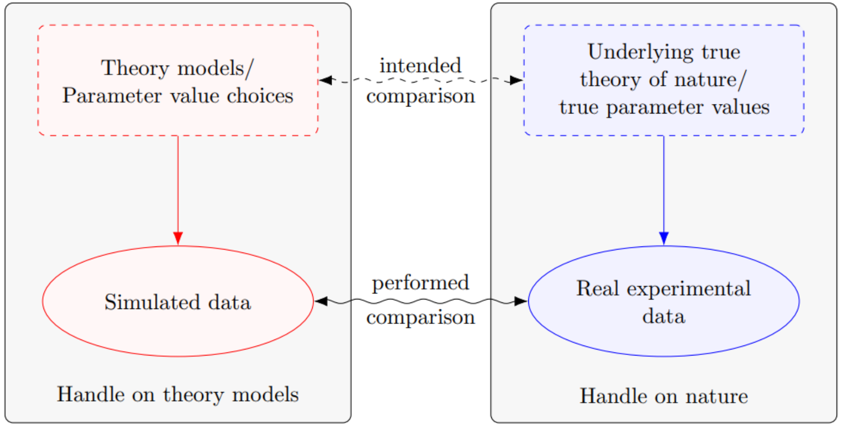
November 11, 2021

Machine Learning at a glance

- There are three major aspects to applying Machine Learning to a problem
  - Design a (highly) parameterized machine to perform a task.
  - Design a cost function to evaluate the performance of the machine.
  - Train the machine by optimizing the cost function.
- Some examples:
  - ML-based classifiers
  - ML-based generators (e.g., Generative Adversarial Networks)

Collider data analysis at a glance

- Experimental data from colliders is compared against simulated data from different theory models, to see which model fits the data best.
- Quintessential analyses:
  - Signal search (e.g., standard model vs standard model+new physics)
  - Parameter measurement



Collider data analysis at a glance

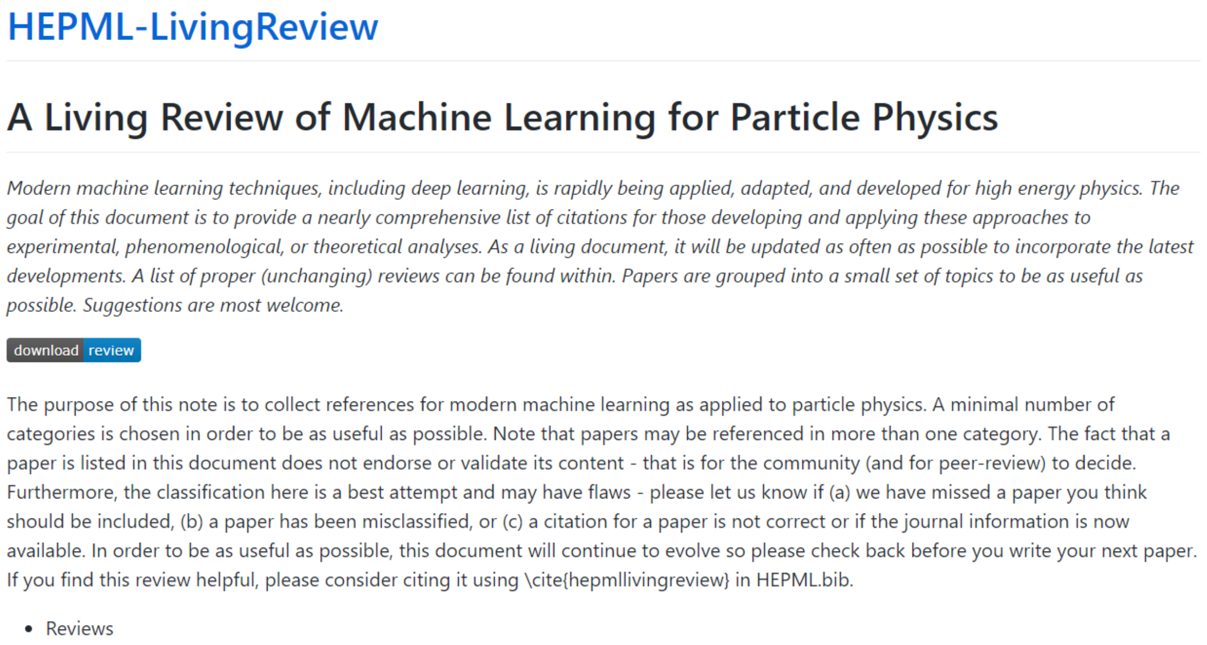
- Data from modern colliders is extremely high-dimensional (Tens of thousands of detector hits).
- Their analysis proceeds in several stages (not sequential):
  - Particle reconstruction (Detector hits to particles reaching the detectors)
  - Object reconstruction (clustering particles into jets)
  - Construction of kinematic variables (e.g., invariant masses of sets of reconstructed objects)
  - Event selection and categorization (e.g., making the data “signal rich”)
  - etc…
- Closely related is the simulation pipeline to generate data:
  - Lagrangian → Parton level events → Parton shower and hadronization → Detector simulation → Electronics simulation

Machine learning in collider data analysis

- Machine learning has been employed in almost every aspect of collider data analysis.
- First applications were in background vs signal classification, based on reconstructed events.
- Since then, ML has been used/proposed for
  - Classification using lower-level data (jet images)
  - Jet tagging (gluon jet vs quark jets, b-jet taggers, top taggers, etc.)
  - Particle reconstruction (from detector hits)
  - Jet clustering
  - Direct parameter measurement from relatively high-dim data
  - Data generation (to speed-up the simulation process)
  - Triggers
  - etc…

A Living Review of Machine Learning for Particle Physics

- https://iml-wg.github.io/HEPML-LivingReview/



HEPML-LivingReview

A Living Review of Machine Learning for Particle Physics

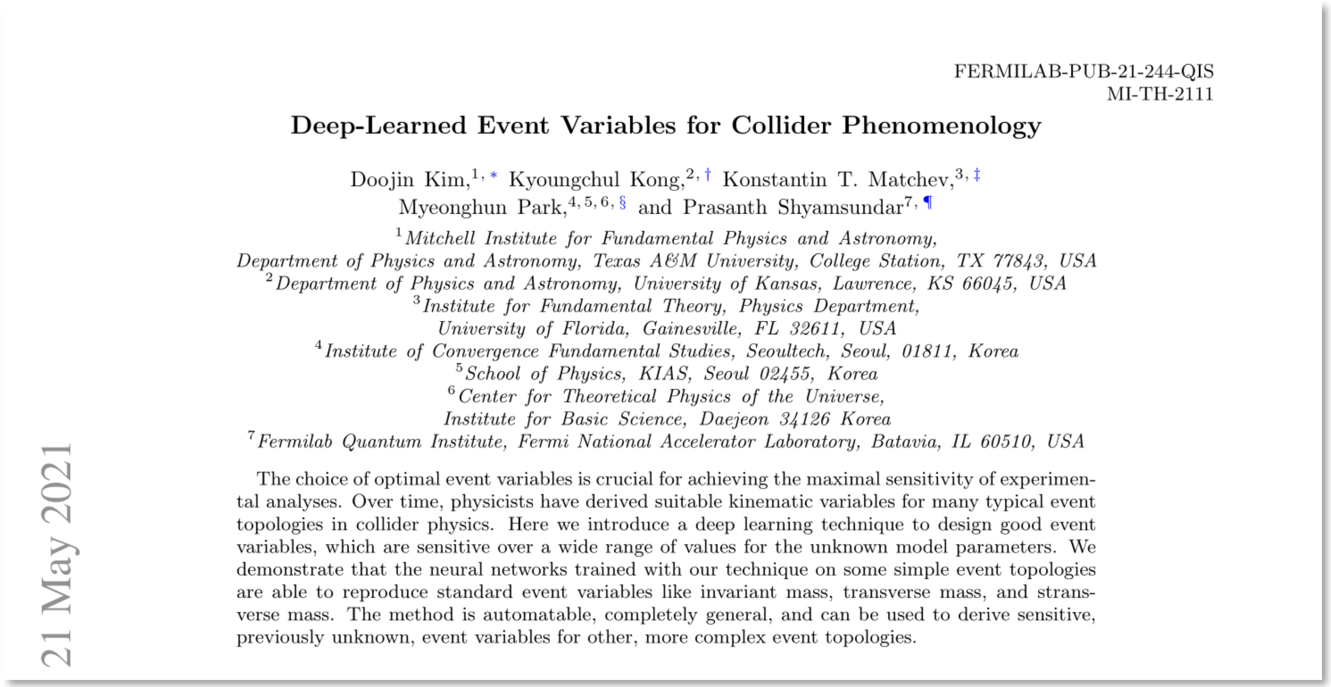
Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

The purpose of this note is to collect references for modern machine learning as applied to particle physics. A minimal number of categories is chosen in order to be as useful as possible. Note that papers may be referenced in more than one category. The fact that a paper is listed in this document does not endorse or validate its content - that is for the community (and for peer-review) to decide. Furthermore, the classification here is a best attempt and may have flaws - please let us know if (a) we have missed a paper you think should be included, (b) a paper has been misclassified, or (c) a citation for a paper is not correct or if the journal information is now available. In order to be as useful as possible, this document will continue to evolve so please check back before you write your next paper. If you find this review helpful, please consider citing it using \cite{hepmllivingreview} in HEPML.bib.

- Reviews

Deep-Learned Event Variables for Collider Phenomenology

Based on Doojin Kim, K.C. Kong, Konstantin Matchev, Myeonghun Park, and Prasanth Shyamsundar, arXiv:2105.10126 [hep-ph]



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

Deep-Learned Event Variables for Collider Phenomenology

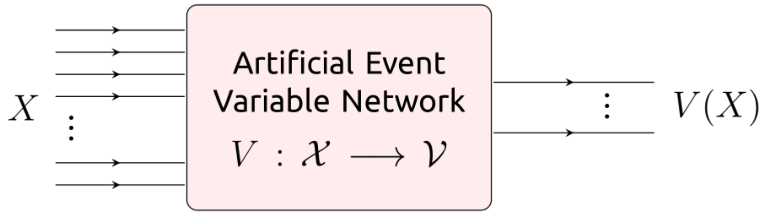
Doojin Kim,[1,*] Kyoungchul Kong,[2,†] Konstantin T. Matchev,[3,‡]
Myeonghun Park,[4,5,6,§] and Prasanth Shyamsundar[7,¶]

[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.

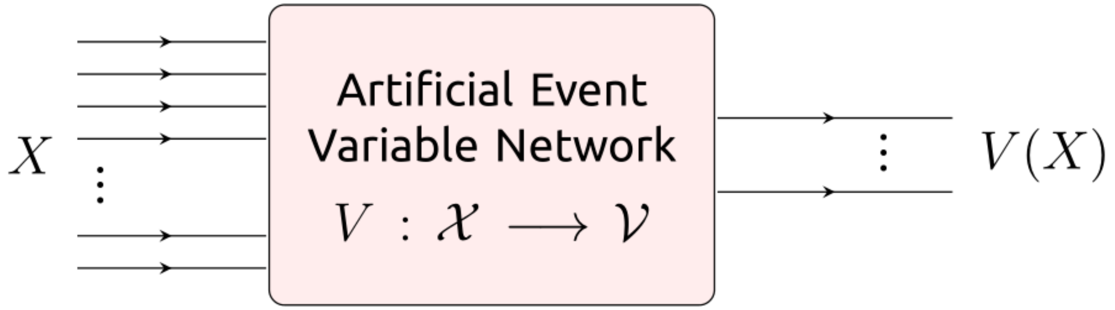
Synthesizing event variables with machine learning

- **What are event variables?**
  - Dimensionality reducing transformations (e.g., invariant mass, transverse mass)
  - High-dimensional event description → 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!


X
Artificial Event Variable Network
$V : \mathcal{X} \longrightarrow \mathcal{V}$
$V(X)$

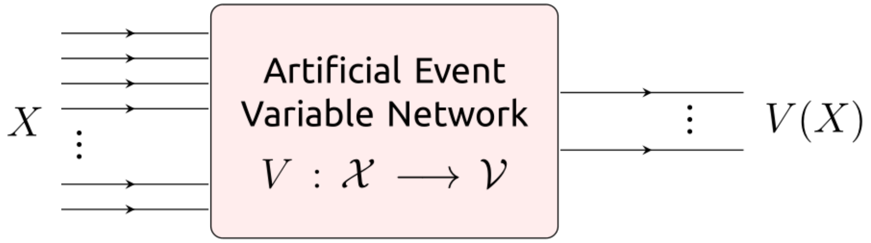
- **How to train such a network?** Not straightforward!

Distinction from other ML applications


X
Artificial Event Variable Network
$V : \mathcal{X} \longrightarrow \mathcal{V}$
$V(X)$

- 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., $X \to L(\Theta \mid X)$. 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…



X
Artificial Event Variable Network
$V : \mathcal{X} \longrightarrow \mathcal{V}$
$V(X)$

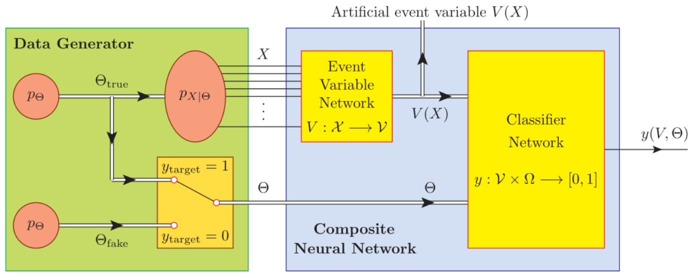
- **Goal:** Train the network to be useful over a range of parameter values.
- One interpretation of the goal:
  - Train the network so that V carries a lot of **information** about the underlying unknown parameters Θ.
  - Mass variables, for example, carry a lot of information about the underlying mass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

Some information theory…

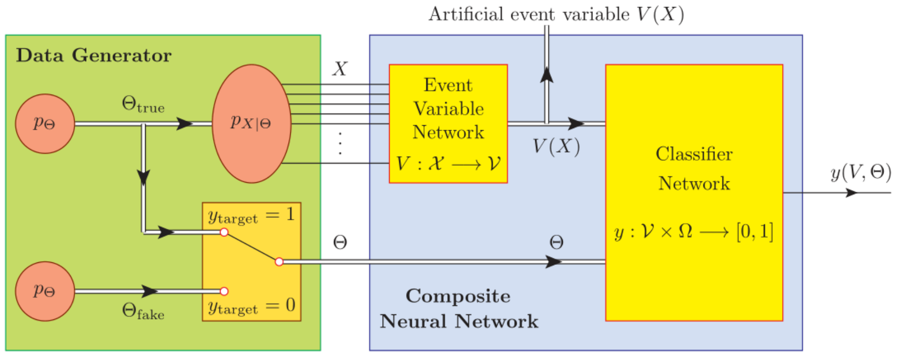
- p_Θ $\equiv$ Prior on the unknown parameters Θ
  $p_{X|\Theta}$ $\equiv$ Dist. of the event X conditional on Θ, $V(X) \equiv$ Event variable
- Mutual information between the event variable V and parameter Θ:

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

- $I(V\,;\,\Theta)$ 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

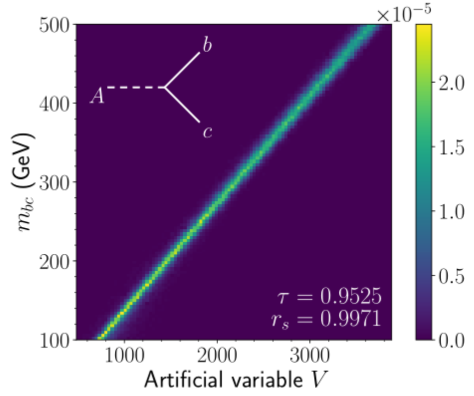


- Training data: $(X, \Theta) \sim p_X \otimes p_\Theta$ under class 0; $p_{(X,\Theta)}$ under class 1
- Event variable network: Transforms X to V .
- Auxiliary Classifier Network:
  Input is $(V, \Theta) \sim p_V \otimes p_\Theta$ under class 0; $p_{(V,\Theta)}$ under class 1.
- Train the composite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

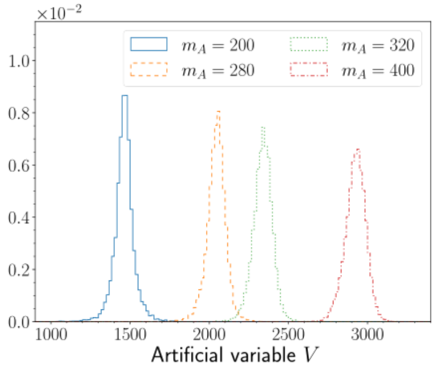
Example 1: Invariant mass

- $A \to b, c$ (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?



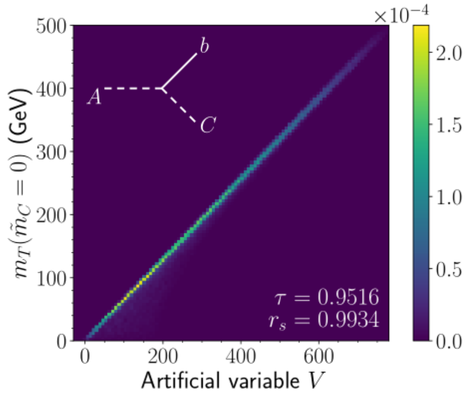
Event variable in action



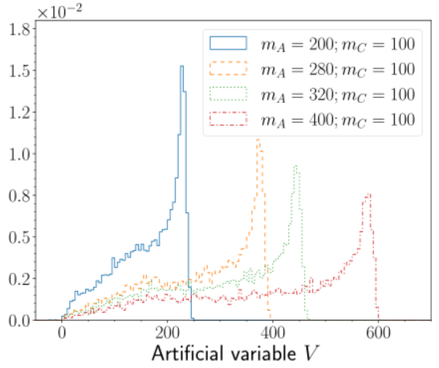
Example 2: Transverse mass m_T

- $A \to b$(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?



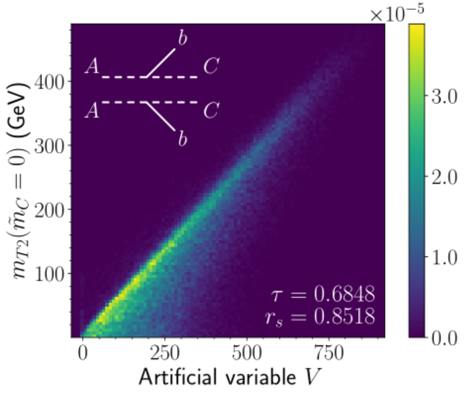
Event variable in action



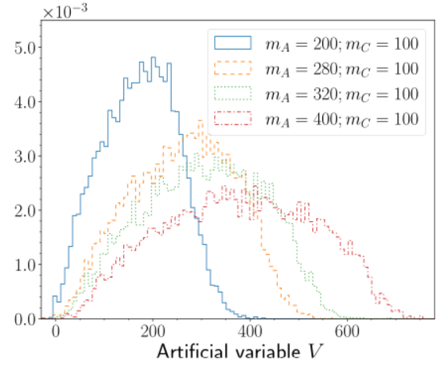
Example 3: Stransverse mass m_{T2}

- $pp \to A_1, A_2$
  $A_i \to b_i$ (massless, visible), C_i (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



Fermilab

Summary and Outlook

- Machine Learning continues to play an important role in collider data analysis.
- Existing ML applications are being refined and novel applications are being invented to improve the sensitivity and reach of our experiments.

- In this talk, we saw 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.
- Now, we can go after previously unknown event variables, for more complicated topologies
- Some advantages of event variables over typical ML approaches:
  - Works over a range of parameter values.
  - Relatively robust against unknown modeling errors (allows for control-region-validation)

Thank you!