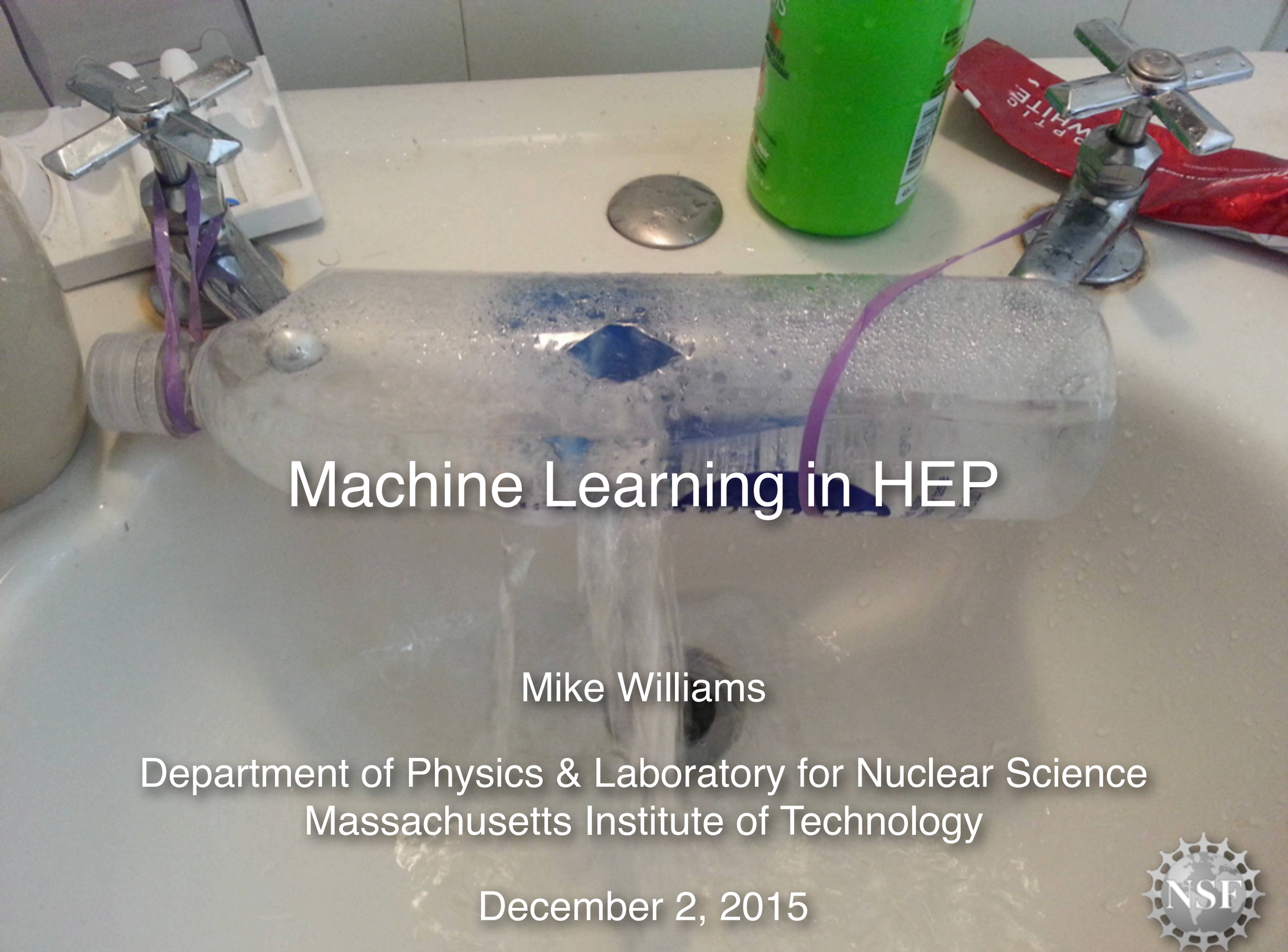


Machine Learning in HEP

Mike Williams

Department of Physics & Laboratory for Nuclear Science
Massachusetts Institute of Technology

December 2, 2015

Intro

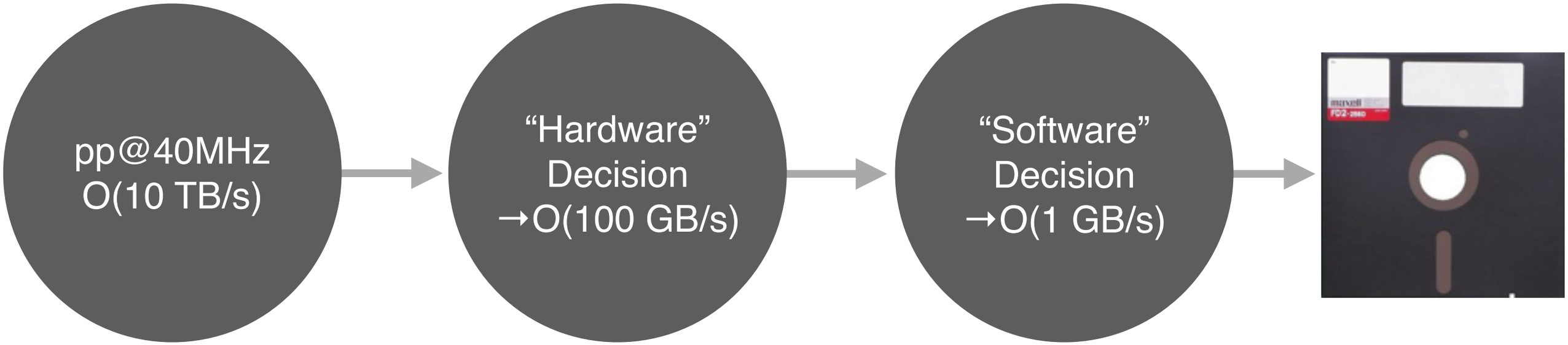
I made the following assumptions when writing this talk:

- Everybody here is an expert on ML.
- By this point today, we've heard a lot about how ML is currently used in HEP, about deep learning, etc.
- Everyone is getting tired of hearing talks, looking at slides, etc.

I'll try and by quick and focus on concepts.

Triggers

The raw data rates the LHC detectors are exposed to are far too large to record every event; therefore, “potentially interesting” events must be selected for recording in real time.



N.b., LHCb plans to remove the “hardware” level in Run 3.

“Inclusive” BDT Trigger?

Major “concerns” to using a BDT in a high-level trigger:

- If keep regions are small relative to the resolution/stability of the detector, the signal could oscillate in/out of them. This would result in a less efficient trigger that is likely very difficult to understand.

- Signal samples are unlikely to be perfect during training. How can we prevent loss of performance due to mismodeling? Can one make an “inclusive” BDT trigger from a small subset of possible signals?

- Trigger algorithms must be extremely fast.

“You’ll run that over my dead body.” --LHCb senior physicist (2010); and they were certainly in the majority opinion.

• Known unknowns: data isn’t perfectly calibrated during running. In principle we can solve this, but in practice it’s difficult.

• Unknown unknowns: a priori we assume that, in fact, we don’t really know exactly what we’re looking for -- and that we’ll learn things from the data itself.

• CPU usage.

“Inclusive” BDT Trigger!

Simple solution is to discretize all features (by hand) prior to training:

- Discretization scheme can easily prevent keep-region size from being comparable to detector resolution/stability resulting in constant performance.

- Discretization scheme also prevents the BDT from learning unwanted aspects of the training signals, e.g., possible correlations between features and interesting signal properties -- and naturally results in good efficiency for any similar type of signals (whatever they may be).

- Discretizing leads to a finite number of possible responses which may be cached in memory -- the BDT can be converted into a 1-D array look-up making it take ~zero time to evaluate.

Since the “grower” of the tree controls/shapes the growth, I called it a “Bonsai BDT” (BBDT). Implemented by LHCb in March 2011.

V.Gligorov, MW, JINST 8 (2012) P02013.

ML Triggers @ LHCb

• BBDT used in Run 1 “Topological” trigger (Topo) by LHCb. Given about 30-40% of the output bandwidth, and used in ~60% of all LHCb papers thus far (about 180 to-date). Performance better than thought possible, selects nearly 100% pure b-quark hadron decays.

• No variation in performance observed during stable running in 2 years of Run 1. Very few signals published were in the training samples, but excellent performance nonetheless.

• Topo re-optimized for Run 2 by MIT-Yandex collaboration. Performance even better! See github.com/yandexdataschool/LHCb-topo-trigger.

• Software trigger at LHCb currently has 2 stages. ML now also used in first stage and given ~2/3 of bandwidth! (This also designed by MIT-Yandex.)

The Bonsai approach is “primitive” -- but it has worked extremely well (and really, as physicists we care about utility).

That said, can we do better in the future?

Future ML Triggers

Beyond the Bonsai:

- Known unknowns: LHCb now buffers the data on 5 PB of disk, runs the final calibrations, then makes the trigger decision. Data is now understood at the trigger level.

Solved!

- Unknown unknowns: For a given (known) category of signals, one could imagine solving this problem via brute force; however, what about BSM-type signals? E.g., in the totally inclusive single-displaced-track trigger used in stage one, we simply enforced the concept that performance should be monotonic in displacement and momentum -- even though in the actual training samples, material interactions lead to the ML algorithm not wanting to do this by default. Can we formalize this for more generic situations?

???

- CPU time: This is a pure CS problem. Yandex has some interesting solutions to this already that are sufficient for our needs.

Solved!

So, what would you say ya do here?!?

Can we formalize our intuition?

Example Domain-Specific Problems

In many cases in HEP, one difficulty in applying ML is that one cannot define a FOM in terms of S and B. Consider:

• Signal is a BSM particle with an unknown mass and lifetime. Many (typically most) discriminating features are strongly correlated with mass and lifetime. Sometimes one can rescale features to mostly remove the dependence on mass, but some detector responses simply cannot be rescaled.

• Training background sample comes from a sideband but features cannot be rescaled -- how can we ensure that the ML performance in the signal region is similar to the sideband?

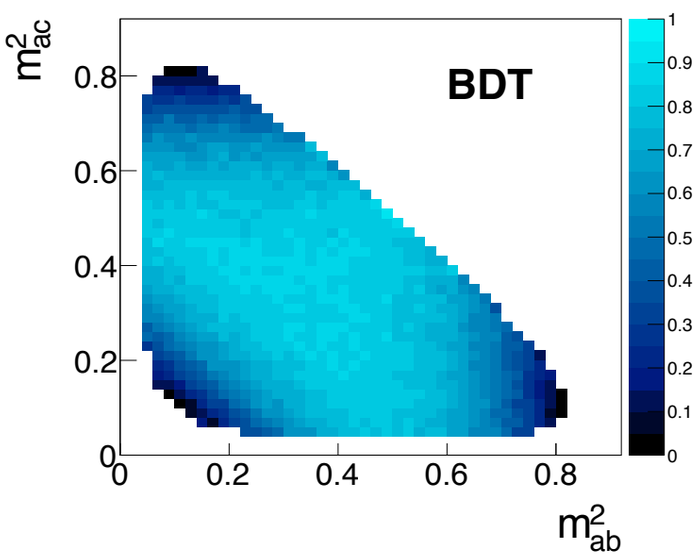
• Signal is a multi-particle final state whose angular distributions are unknown and, in fact, it is these we want to analyze. The background is highly nonuniform in these angles both in yield and in discrimination from signal. We don't want the ML algorithm to simply throw away some kinematic regions to enhance any integrated FOM(S,B).

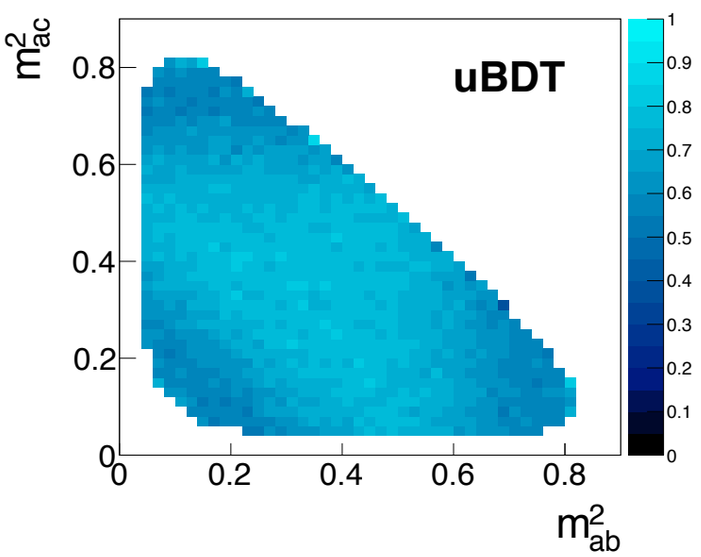
Custom Loss Functions

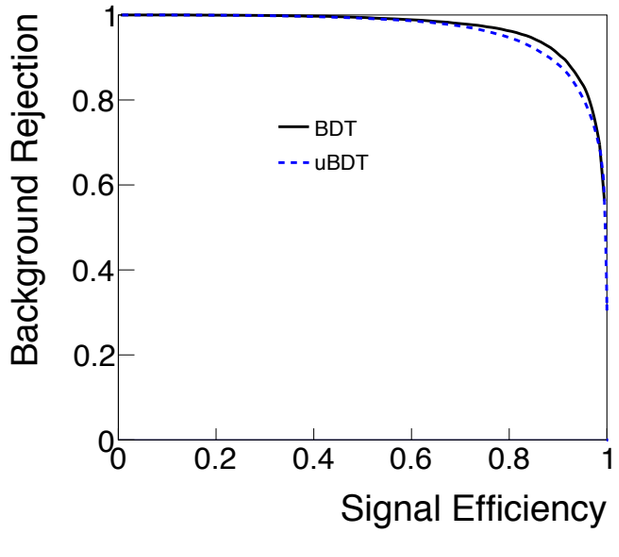
While FOM(S,B) may not exist, we can consider modifying the loss function to achieve our goals. E.g., all examples on the previous page work if the ML performance is roughly uniform in a known MVA space.

There are many ways to achieve this, e.g., using Loss = Loss(flat) + α Loss(ada), where Loss(flat) can be defined using a local CvM-based comparison of “local” and “global” ML response distributions -- then used this in gradient boosting (uGBFL).

Alternative (my original paper) alternates using flatness-based & Ada losses in boosting stages (uBoost).



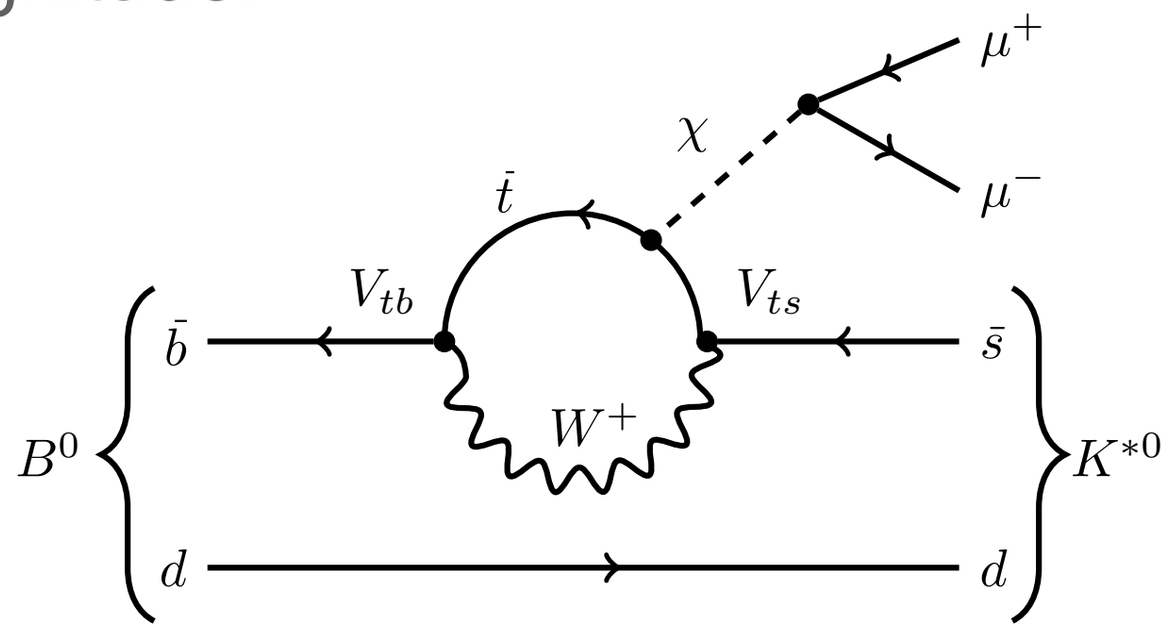




J.Stevens, MW, JINST 8 (2013) P12013.
A.Rogozhnikov,A.Bukva,V.Gligorov,A.Ustyuzhanin,MW, JINST 10 (2015) T03002.

Example uBDT Usage

So-called $b \to s$ penguin decays are an excellent place to search for low-mass hidden-sector particles (e.g., anything that mixes with the Higgs sector). In the decay $B \to K^* X(\to \mu\mu)$, X can have a mass in the range ~210--4400 MeV, and its lifetime may take on a value covering many orders of magnitude.



How do we train our ML-based selection? Generating MC samples with different (m,tau) is expensive; however, we can easily generate a handful then enforce uniform performance in the (m,tau) plane using a uBDT.

Hidden Sectors

Search performed blindly by scanning dimuon mass and allowing (but not requiring) a displaced dimuon vertex (from the B decay point). Details on the strategy (which must deal with QCD resonances) can be found in MW, JINST 10 (2015) P06002.

Internally, we did a test: Trained a BDT and a uBDT (using uBoost, asking for uniform performance on signal in the (m,tau) plane) on 10 samples with various (m,tau) values. The BDT vs uBDT performance for these (m,tau) values was found to be consistent.

Next, we compared performance on samples with (m,tau) values NOT used in the training. The uBDT outperformed the BDT on all of them -- its performance showed no favoring of the training (m,tau) values. Furthermore, we made no attempt to exclude or augment features correlated with (m,tau), i.e., we were lazy.

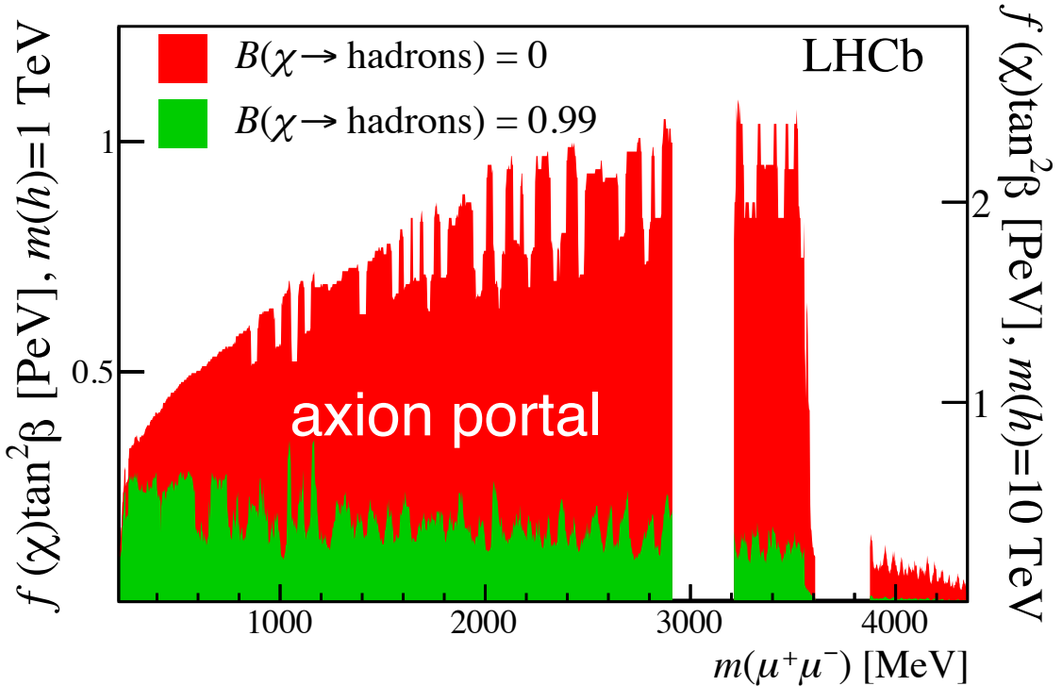
Hidden Sectors

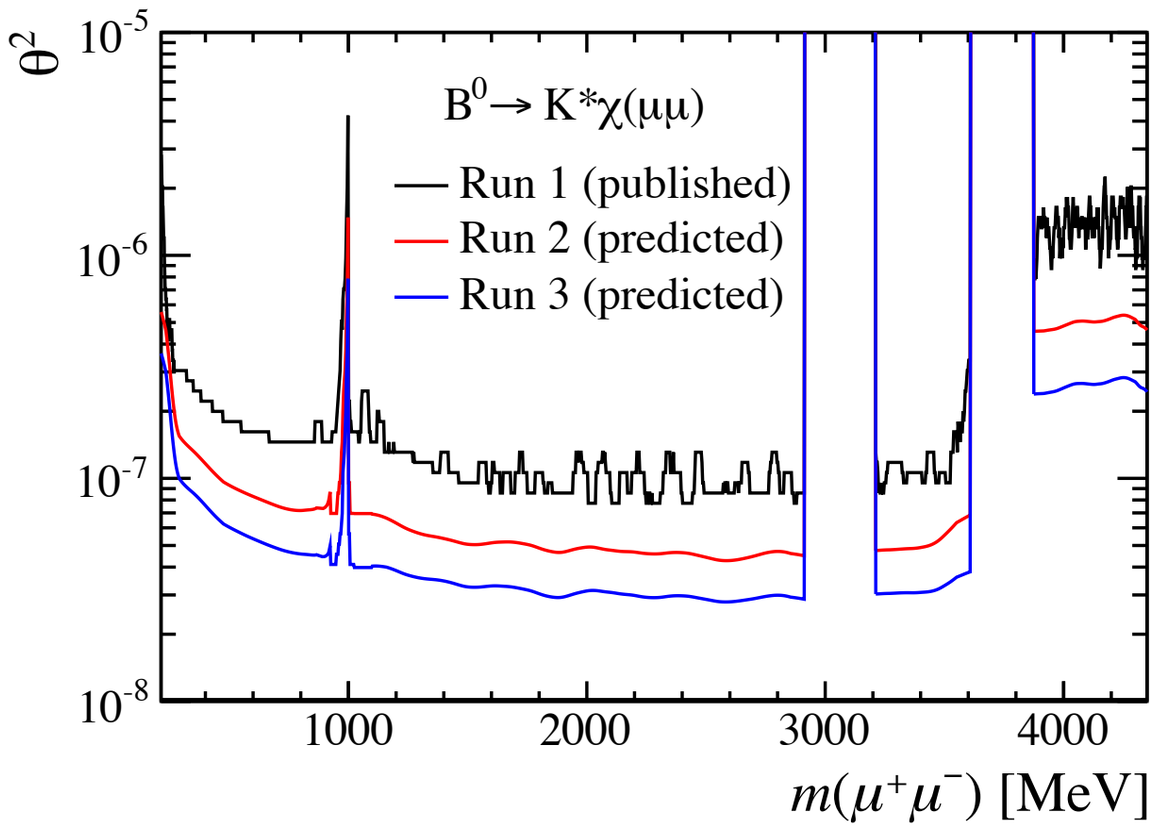
Unfortunately, we found no evidence for such a boson, and model-independent limits were set. Impact on a few models shown below.

PRL 115 (2015) 161802

Freytsis, Ligeti, Thaler [arXiv:0911.5355]

PeV scale!





The uBDT concept is simply to encode in the loss function the domain-specific part of the classification -- then makes analysis simple. In this case, we consider non-local information for each candidate to solve a specific class of problems.

The future?

Some thoughts on areas for progress...

HEP-Specific ML

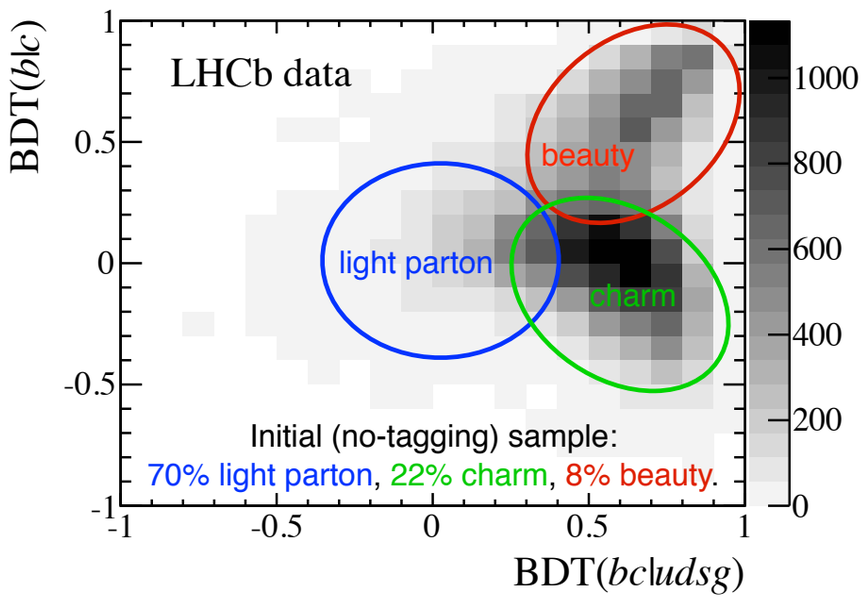
Following on from the success of the uBDT, can we formalize what it is we really want out of a ML algorithm in cases where it simply isn't some FOM(S,B)?

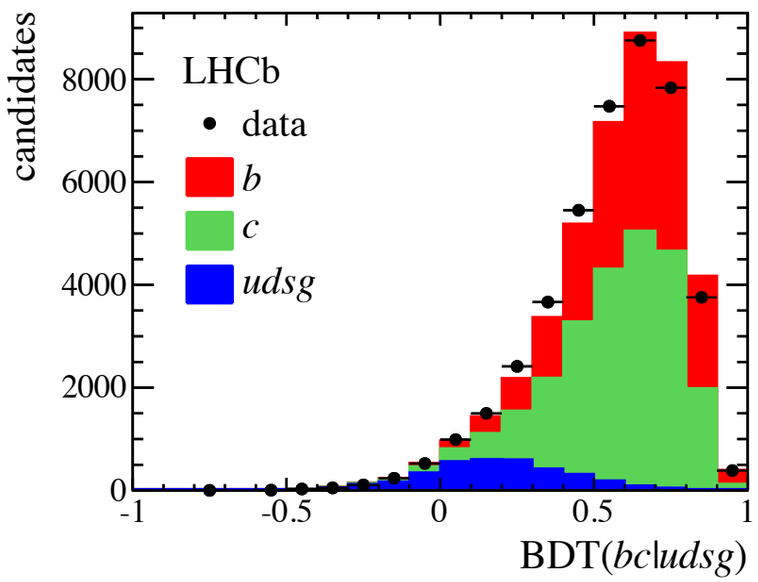
Can we incorporate (a priori unknown, but obtainable in data-driven ways) systematic uncertainties into the optimization? E.g., can the entire analysis -- including classification -- be carried out by the ML algorithm to obtain a full systematic uncertainty for any selection? If so, the ML could try many selections and take the one that produces the minimum total error.

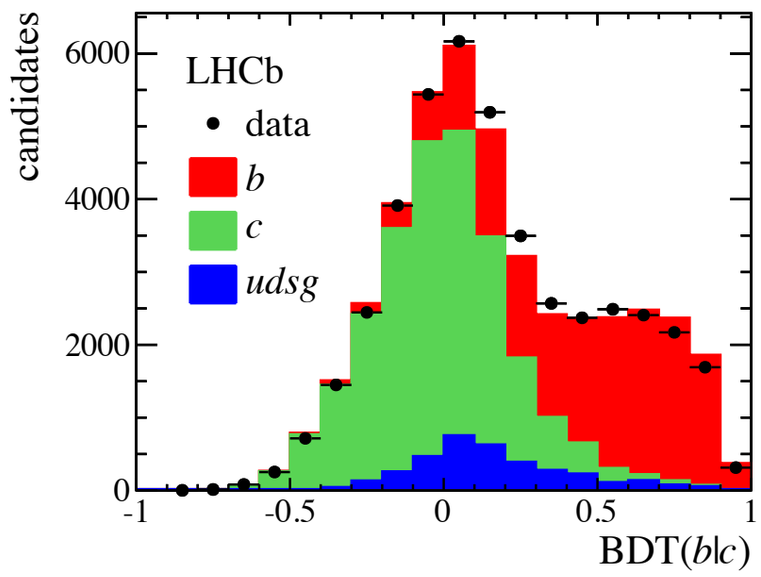
Multi-Class Problems

In many cases there isn't really "signal" and "background" -- there are multiple classes of events, and many (possibly all) are interesting. E.g., consider jet tagging where b-jets and c-jets are both signals.

JINST 10 (2015) P06013
LHCb-PAPER-2015-016







Shouldn't a 3-class ML algorithm be able to beat two 2-class ones? What about charged particle ID (5-class problem)? Etc. I'm currently studying this with Yandex.

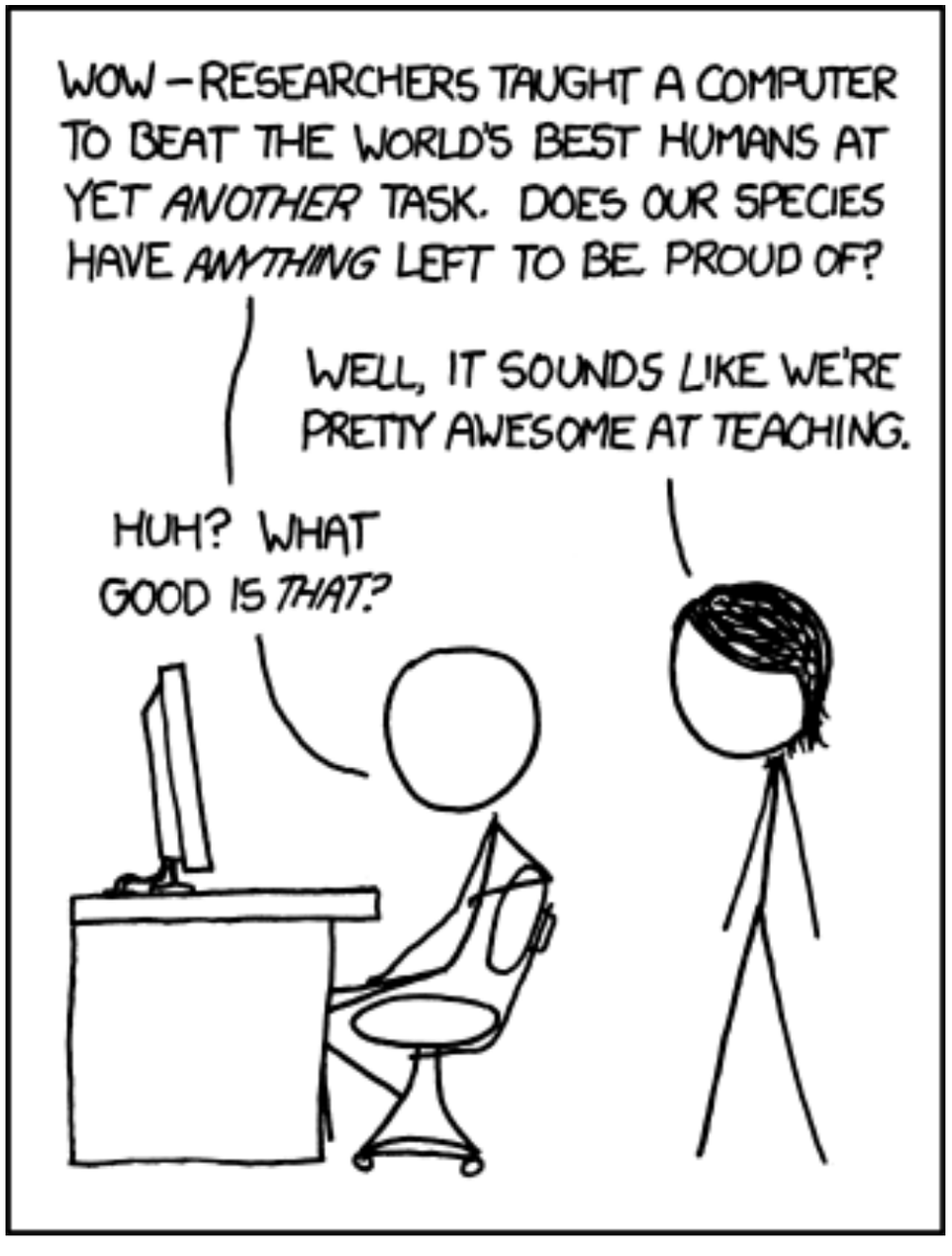
Machine Teaching?

What about situations where we don't actually understand what nature does? E.g., emergent properties of QCD are difficult to predict.

Can't we just let ML algorithms try and learn "interesting" things about the data and then tell us what they learned?

E.g., from data samples of jets produced in association with various other particles, tell me what is different about these jets machine!

Beyond this, humans learn by doing -- if ML replaces us, can it learn enough to design better detectors, etc?



Summary

ML has made a huge positive impact on HEP in the past decade. We're a long way from spending our days on the beach while our algorithms do all the work -- but I do expect a lot of progress in the near future, especially if HEP and ML scientists work together!

Machine, search for SUSY.

Not found.

Hmm, black holes?

Not found.

Eh, I'm going to take a nap…can you look for interesting differences in features between these jet samples?

Sure.