Reusing ML tools and approaches for HEP

Andrey Ustyuzhanin\textsuperscript{1,2,3,4} on behalf of YSDA LHCb team

\textsuperscript{1}YSDA, \textsuperscript{2}NRC ”Kurchatov Institute”, \textsuperscript{3}HSE, \textsuperscript{4}YDF
Yandex School of Data Analysis

› non commercial private university
  https://yandexdataschool.com
› 450+ students graduated since 2007
› strong education focus on Data & Computer Science
› 25% of our students have physics background
› associate member of LHCb since Dec 2014
› interested in improving Physics results by Machine Learning
› organize bi-yearly international Machine Learning Conference
  https://yandexdataschool.com/conference/
Overview

› Motivation

› [Possible] way to go
  › Technological part
  › Sociological part
  › “yak shaving”

› HEP Challenges

› Some Results

› Conclusion
Case for ML for HEP

Higgs ML Challenge (http://bit.ly/1HysVUN, p28)

20-40% more data needed to get the same improvement
Space is really big. You just won’t believe how vastly hugely mindboggingly big it is. I mean you may think it’s a long way down the road to the chemist, but that’s just peanuts to space.

Douglas Adams,
Hitchhiker’s Guide to the Galaxy
ML out there

› Generic ML approaches
  › Classification
  › Clustering
  › Regression
  › Dimensionality Reduction
  › Feature Engineering
  › Model Selection (AutoML)

› Languages
  › R, Matlab, Python, Lua, Julia, ...

› Tools
  › https://github.com/josephmisiti/awesome-machine-learning
  › https://mloss.org/about/
  › https://www.kaggle.com/wiki/Software
  › ...
Popular ML problems in HEP

› Classification
  › Binary, Multi-class

› Model Selection
› Regression
› Pattern recognition
› ...
› http://bit.ly/1M0orCQ
Ingredients

› Shared datasets

› Technology
  › data
  › data format
  › algorithms accessibility

› People you can learn from
  › #DSLHC has reached its full capacity 3 weeks before start!
  › Yandex School of Data Analysis has joined LHCb in 2014
  › HEPML workshop in 2014
  › ALEPH workshop (http://yandexdataschool.github.io/aleph2015/)

› A bit of “yak shaving”
  › decompose of problems into ‘independent’ pieces
  › algorithm execution transparency
  › computationally rich communication platform
  › consistency checks
Technology part

Reproducible Experiment Platform (REP)
machine learning toolbox for humans

› unified classifier wrappers for
› Sklearn, XGBoost, uBoost, TMVA, ANN, Theanets, PyBrain, ...
› pluggable quality metrics
› support for interactive plots
› parallelized grid search strategies
› parallel training of classifiers classification/regression reports
› meta-algorithms ("REP Lego")

https://github.com/yandex/rep/
Scenarios

Installation

› manually (pip install ..., many dependencies!)
› virtual env
› docker
Scenarios

Installation

› manually (pip install ..., many dependencies!)
› virtual env
› docker

Deployment ‘modes’

› laptop
› remote server
› remote cluster
› cloud platform (docker-machine)
Accessing ROOT files (root_numpy)

data = root_numpy.root2array(
    "toy_datasets/random.root",
    branches=['X1', 'X2', 'sin(X1)*exp(X2)'],
    treename='tree', selection='X1 > 0')

root_numpy.array2root(data, filename='output.root',
                        treename='tree', mode='recreate')

with root_open('test.root', mode='a') as myfile:
    c = numpy.array(mycolumn, dtype=[('new', 'f8')])
    root_numpy.array2tree(c, tree=myfile.tree)
    myfile.write()
Training a classifier (SKlearn GBDT)

```python
train_data, test_data, train_labels, test_labels = 
    train_test_split(data, labels, train_size=0.5)
variables = list(data.columns[:26])
from rep_estimators import SklearnClassifier
from sklearn.ensemble import GradientBoostingClassifier
sk = SklearnClassifier(GradientBoostingClassifier(),
    features=variables)
sk.fit(train_data, train_labels)
```
Training a classifier (TMVA)

```python
train_data, test_data, train_labels, test_labels = train_test_split(data, labels, train_size=0.5)
variables = list(data.columns[:26])
from rep.estimators import TMVAClassifier
tmva = TMVAClassifier(method='kBDT', NTrees=50,
                      Shrinkage=0.05, features=variables)
tmva.fit(train_data, train_labels)

https://github.com/yandex/rep/blob/develop/howto/01-howto-Classifiers.ipynb
```
Making predictions

```python
prob = tmva.predict_proba(test_data)
print 'ROC AUC:', roc_auc_score(test_labels, prob[:, 1])
```
from sklearn.ensemble import AdaBoostClassifier

# Construct AdaBoost with TMVA as base estimator
base_tmva = TMVAClassifier(method='kBDT', NTrees=15, Shrinkage=0.05)
ada_tmva = SklearnClassifier(AdaBoostClassifier(base_estimator=base_tmva, n_estimators=5), features=variables)
ada_tmva.fit(train_data, train_labels)
Meta-ML, Factories

```python
defactory = ClassifiersFactory()
defactory.add_classifier('tmva', tmva)
defactory.add_classifier('ada', ada)
defactory['xgb'] = XGBoostClassifier(features=variables)
# training
factory.fit(train_data, train_labels, features=variables, parallel_profile='IPC')
# predict
factory.predict_proba(some_data, parallel_profile='IPC')
```

https://github.com/yandex/rep/blob/develop/howto/02-howto-Factory.ipynb
Visual Comparison

```python
from rep.report.metrics import RocAuc
report = factory.test_on(test_data, test_labels)
learning_curve = report.learning_curve(RocAuc(),
    metric_label='ROC AUC', steps=1)
learning_curve.plot()
```
More links

› Neural Networks: https://github.com/yandex/rep/blob/develop/howto/06-howto-neural-nets.ipynb
› add new estimator: https://github.com/yandex/rep/wiki/Contributing-new-estimator
› add new metric: https://github.com/yandex/rep/wiki/Contributing-new-metrics
› and other stuff...

“Sociological” part

Cooperation + Competition = Coopetition
[Old] cooperation model
Coopetition model
Education & communication

› ALEPH 2015
  http://yandexdataschool.github.io/aleph2015/
› Heavy Flavour Data Mining, Feb 2016, Zurich
  https://indico.cern.ch/event/433556/
› DS @ LHC 2016
› Connecting the dots workshop,
› Competitions
  › Kaggle
  › OpenML
  › CodaLab
  › ...

Andrey Ustyuzhanin
“Yak shaving” part

Any apparently useless activity which, by allowing you to overcome intermediate difficulties, allows you to solve a larger problem. https://en.wiktionary.org/wiki/Talk:yak_shaving

› How to share data?
› How to share knowledge?
› How to reproduce results?
› How to share the code?
› How to share the environment?
› What are communication practices?
› How to make sure the whole analysis stays in place with every commit?
Common practices

› dropbox
› use github
› continuous integration (travis)
› wiki
› ticketing
› agile
One more step to lower the borders

› Algorithm containerization
› Cross-language experience exchange
› Analysis preservation
› Docker-like technology, containerization
Everware

› Get familiar with Docker
› Try it: https://everware.rep.school.yandex.net
› Examples
  › https://github.com/everware/everware-dimuon-example
  › https://github.com/arogozhnikov/hep_ml
  › https://github.com/yandexdataschool/manchester-cp-asymmetry-tutorial
  › https://github.com/yandexdataschool/flavours-of-physics-start

› Discussion about integration is ongoing with
  › CERN open data portal
  › http://openml.org

https://github.com/everware/everware/everware
Selected HEP Challenges

› LHCb Topological Triggers
› LHCb Flavour Tagging
› Rare decay search ($\tau \rightarrow 3\mu$)
› COMET track reconstruction
› Data Placement Strategy
› Online Data Processing Anomaly Detection
LHCb Topological Triggers, Data

- Sample: one proton-proton bunches collision (40MHz)
- Event consists of the secondary vertices (SVR), where particles are produced
- Features: SVR and decay products physical characteristics reconstructed (momentum, mass, angles, impact parameter)
ML problem statement

- Output rate is fixed, thus, false positive rate (FPR) for events is fixed
- Goal is to improve efficiency for each type of signal events
- We improve true positive rate (TPR) for fixed FPR for events
Random forest for SVRs selection

- Train random forest (RF) on SVRs
- Select top-1, top-2 SVRs by RF predictions for each signal event
- Train MatrixNet on selected SVRs
Bonsai BDT, online processing

- Features hashing using bins before training (equivalent to features binarization in MatrixNet)
- Converting decision trees to N-dimensional lookup table
- Table size is limited in RAM
Post-pruning

› Train MatrixNet with several thousands trees, then reduce this amount of trees to a hundred by
› greedy selection of trees from the initial ensemble to minimize a modified loss function (exploss for BG and logloss for SIG)
› update leaves
Results

N-Body trigger Performance Comparison
(bars correspond to trigger efficiency for different decay modes)

https://github.com/yandexdataschool/LHCB-topo-trigger
Other Results

› LHCb Triggers — up to 60% relative efficiency increase https://github.com/yandexdataschool/LHCb-topo-trigger
› Rare decay search \((\tau \rightarrow 3\mu)\) — 6% significance increase in upper limit estimation
› COMET track reconstruction improved ROC AUC from 88.3% to 99.9% https://inclass.kaggle.com/c/comet-track-recognition-mlhep-2015
ML-inspired tools for HEP (HEP_ML)

› https://github.com/arogozhnikov/hep_ml
Reweighting use-case for particle physics is to modify output of Monte-Carlo (MC) simulation to reduce disagreement with real data (RD). Let’s assign new weights to MC such that MC and RD distributions coincide.
Traditional approach

multiplier_{bin} = \frac{w_{target, \bin}}{w_{original, \bin}}

Caveat Emptor!

› no good test for N-dimensional case
› becomes unstable as bins get small number of events
› no good reason to compare distributions per se (complex N-D objects)
BDT-reweighter intuition

Can my classifier spot any difference between MC and RD?

\[ \chi^2 = \sum_{\text{bin}} \left( \frac{w_{\text{bin, original}} - w_{\text{bin, target}}}{w_{\text{bin, original}} + w_{\text{bin, target}}} \right)^2 \]
BDT-reweighter approach

1. build a shallow tree to maximize symmetrized $\chi^2$
2. compute predictions for leaves:

$$\text{leaf}_{\text{pred}} = \frac{w_{\text{leaf, target}}}{w_{\text{leaf, original}}}$$

3. reweight distributions:

$$w = \begin{cases} 
  w, & \text{if event from target (RD) distribution} \\
  w \times e^{\text{pred}}, & \text{if event from original (MC) distribution}
\end{cases}$$

4. Repeat
Result illustration

Andrey Ustyuzhanin
Two major problems addressed by ML: distribution comparison and reweighting

Things to remember:

› check distributions by classifier used in analysis
› check reweighter on a holdout
Instructions

```python
from hep_ml.reweight import GBReweighter
gb = GBReweighter()
gb.fit(mc_data, real_data,
       target_weight=real_data_sweights)
gb.predict_weights(mc_other_channel)

> https://arogozhnikov.github.io/2015/10/09/
    gradient-boosted-reweighter.html
```
Outro

› [Open] Data
› Jupyter, REP to improve computational communications
› Things to improve team coopetition
   › Continuous Integration
   › Everware

› Communication channels
   › ALEPH Workshop, Dec 2015
   › Heavy Flavour Data Mining Workshop, Feb 2016
   › DS @ LHC 2016
   › MLHEP summer school
   › ML Challenges platforms
Make Science, Not War
Thank you!