Efficient large scale kernel methods for high energy physics

Marco Letizia

MaLGa – University of Genova

- *Learning new physics efficiently with nonparametric methods*, M.L., Gianvito Losapio, Marco Rando, Gaia Grosso, Andrea Wulzer, Maurizio Pierini, Marco Zanetti, Lorenzo Rosasco, To appear in EPJC, arXiv: 2204.02317 [hep-ph].
- *Kernel methods through the roof: handling billions of points efficiently,* Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi,

  NeurIPS 2020, arXiv:2006.10350 [cs.LG]
- *FALKON: An Optimal Large Scale Kernel Method,* Alessandro Rudi, Luigi Carratino, Lorenzo Rosasco, NeurIPS 2017, arXiv:1705.10958 [stat.ML]

Motivations

- Attempt to understand NPLM model by exploring connections with more standard ML approaches.
- Find a way to reduce training time of NN implementations, $\mathcal{O}(hours)$ for each toy (d=1-5, $N = \mathcal{O}(10^5)$).
- Good playground to test Falkon.

Outline

- Goal
- Statistical foundations
- Kernel methods
- Falkon
- Applications:
  - NPLM
  - DQM
  - Validationd of generative models/DE

Goal

Establish the compatibility between a *reference model* and the *data*

Reference $S_0 = \{x_i\}_{i=1}^{\mathcal{N}_0}, \;\; x_i \sim p(x|0)$

Data $S_1 = \{x_i\}_{i=1}^{\mathcal{N}_1}, \;\; x_i \sim p(x|1)$

$$p(x|1) \approx p(x|0)?$$

Model-independence

Goal

SM distributions

LHC data

Establish the compatibility between a *reference model* and the *data*

Reference $S_0 = \{x_i\}_{i=1}^{\mathcal{N}_0}, \quad x_i \sim p(x|0)$

Data $S_1 = \{x_i\}_{i=1}^{\mathcal{N}_1}, \quad x_i \sim p(x|1)$

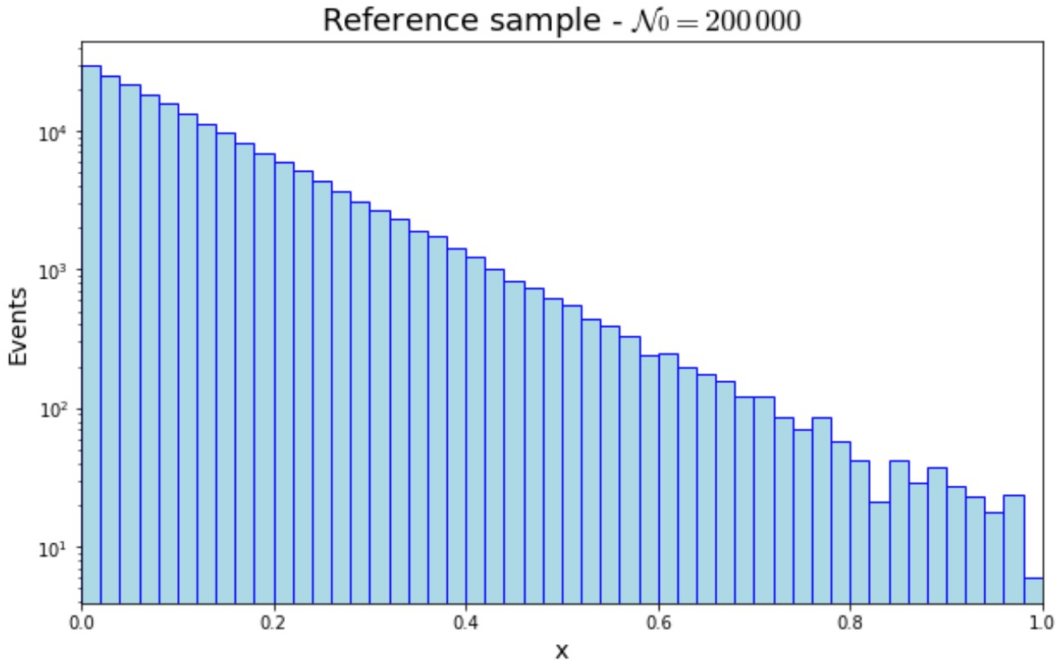
$$p(x|1) \approx p(x|0)?$$

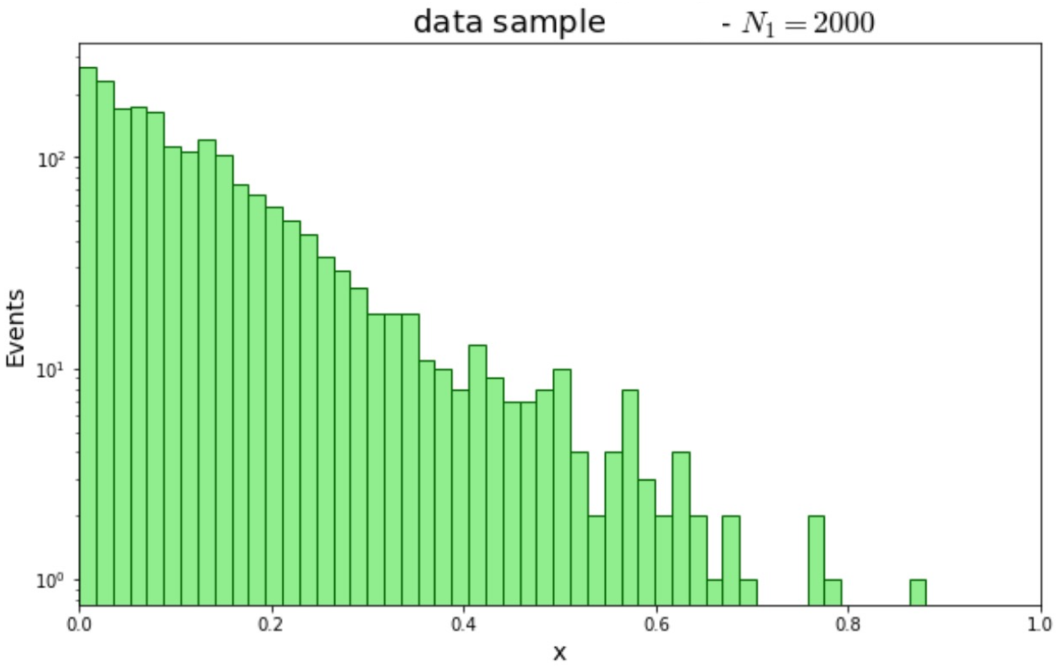
Model-independence

Goal

Goodness of fit via two-sample test:

compare S_0 and S_1 (with large $\mathcal{N}_0$) using machine learning.





Statistical foundations

Hypothesis testing based on likelihood ratio:
data sample S_1, hypothesis y

Likelihood

$$\mathcal{L}(S_1, y) = \frac{e^{-N(y)} N(y)^{\mathcal{N}_1}}{\mathcal{N}_1!} \prod_{x=1}^{\mathcal{N}_1} p(x|y) = \frac{e^{-N(y)}}{\mathcal{N}_1!} \prod_{x=1}^{\mathcal{N}_1} n(x|y)$$

$$n(x|y) = N(y)p(x|y), \qquad N(y) = \int n(x|y)dx$$

Statistical foundations

Parametrized alternative hypothesis

$$n(x|1) \rightarrow n_w(x|1)$$

Learn alternative hypothesis from data → machine learning

Ability of classifiers to (implicitly) model the data generating densities

→ logistic regression

Rich space of functions → kernel methods

Statistical foundations

Likelihood ratio

$$t_w(S_1) = -2\log\frac{\mathcal{L}(S_1,0)}{\mathcal{L}_w(S_1,1)}$$

$$= -2\left[N_w(1) - N(0) - \sum_{x=1}^{\mathcal{N}_1} f_w(x)\right], \qquad f_w(x) = \log\frac{n_w(x|1)}{n(x|0)}$$

Statistical foundations

Designing a classifier for hypothesis testing

Logistic regression

Data $(x_i, y_i)_{i=1}^{n}, \quad y = \{0,1\}$

Loss $\ell_{log}\big(y, f(x)\big) = (1-y)\log\big(1 + e^{f(x)}\big) + y\log\big(1 + e^{-f(x)}\big)$

Proxy of classification error/maximum likelihood principle

Statistical foundations

Given an instance x and the function $f(x)$ that the model is representing

$$\rightarrow P(1|x) = \sigma\big(f(x)\big) = \frac{1}{1+e^{-f(x)}}$$

$$\rightarrow P(0|x) = 1 - P(1|x) = \sigma\big(-f(x)\big) = \frac{1}{1+e^{f(x)}}$$

Optimize negative log-likelihood

$$L = -\log \prod_{(x,y)} \sigma\big(f(x)\big)^{y} \sigma\big(-f(x)\big)^{1-y}$$

Statistical foundations

Each loss defines a goal via a target function

$$\ell_{log}(y, f(x)) = (1 - y)\log(1 + e^{f(x)}) + y\log(1 + e^{-f(x)})$$

$$L(f) = \int \ell_{log}(y, f(x)) p(x, y)\, dxdy = \int p(x) dx \int \ell_{log}(y, f(x)) p(y|x)\, dy$$

$$f^* = \arg\min_f \int \ell_{log}(y, f(x)) p(y|x)\, dy \rightarrow f^* = \log \frac{p(1|x)}{p(0|x\,)}$$

Statistical foundations

$$\ell_{log}\big(y, f(x)\big) = a_0(1-y)\log(1+e^{f}) + a_1 y \log(1+e^{-f})$$

$$\rightarrow f^* = \log\left(\frac{p(1|x)}{p(0|x\,)}\frac{a_1}{a_0}\right)$$

$$\frac{a_1}{a_0} = \frac{p(0)}{p(1)}\frac{N(1)}{N(0)} \rightarrow f^* = \log\left(\frac{n(x|1)}{n(x|0\,)}\right)$$

Statistical foundations

$$\frac{a_1}{a_0} = \frac{p(0)}{p(1)}\frac{N(1)}{N(0)} \approx \frac{\mathcal{N}_0}{\mathcal{N}_1}\frac{N(1)}{N(0)} \approx \frac{\mathcal{N}_0}{\mathcal{N}_1}\frac{\mathcal{N}_1}{N(0)} = \frac{\mathcal{N}_0}{N(0)}$$

$$\ell_{log}\big(y, f(x)\big) = \frac{N(0)}{\mathcal{N}_0}(1-y)\log(1+e^{f}) + y\log(1+e^{-f})$$

Statistical foundations

$$f_{\hat{w}} \approx f^* = \log \frac{n(x|1)}{n(x|0)}$$

$$N(1) = \int n(x|1) dx = \int n(x|0) e^{f^*} dx \to N_{\hat{w}}(1) = \frac{N(0)}{\mathcal{N}_0} \sum_{x \in S_0} e^{f_{\hat{w}}(x)}$$

$$t_{\hat{w}}(S_1) = -2 \left[\frac{N(0)}{\mathcal{N}_0} \sum_{x \in S_0} \left(e^{f_{\hat{w}}(x)} - 1 \right) - \sum_{x \in S_1} f_{\hat{w}}(x) \right]$$

The model is trained as a classifier, but we are not interested in typical classification metrics.

We now need a rich class of functions.

Kernel methods

regularization term

ERM $$\hat{f} = \arg\min_{f \in \mathcal{H}} \hat{L}(f) + \lambda\, R$$

empirical error $$\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(y, f(x)\big)$$

Select a space $\mathcal{H}$ of possible functions, e.g., linear functions

$$f_w(x) \;=\; w^T x$$

Most common models, not very expressive but nice properties.

Kernel methods

Nonlinear functions

- $f_w(x) = w^{\mathrm{T}}\Phi(x)$, kernels
- $f_w(x) =$ "$\sigma(w^T x)$", neural nets

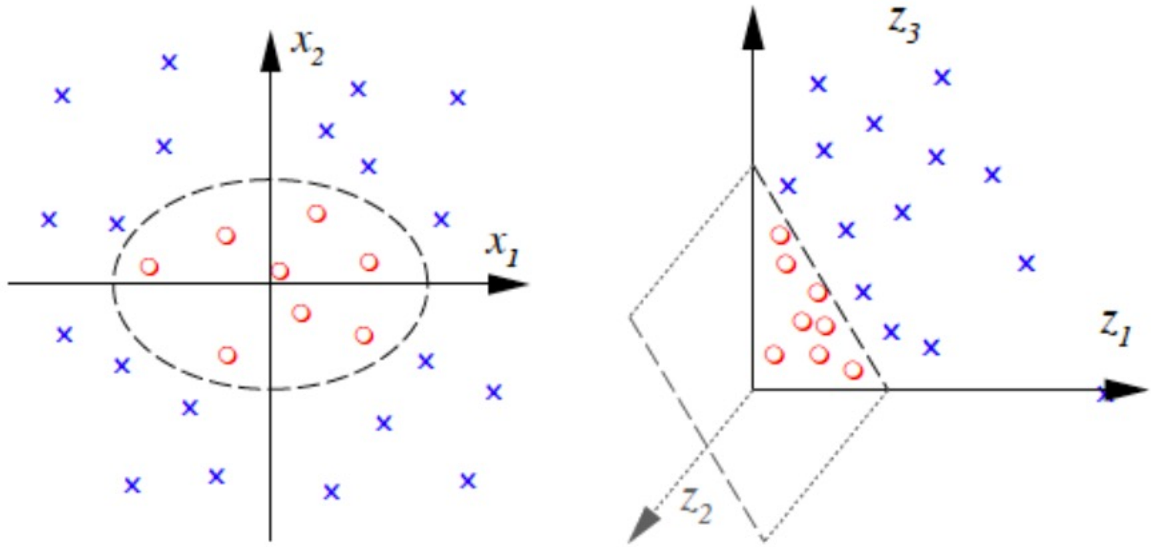
+ weights constraints, e.g., $||w|| < \lambda$.

Kernel methods

$f(x) = w^{\mathrm{T}}\Phi(x)$, *feature map* $\Phi: X \to F$,

Input (x_1, x_2) Feature map $\Phi(x) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)$

still linear



Kernel methods

One can consider inf dimensional feature maps if $k(x, x') = \Phi^T(x)\Phi(\mathrm{x}')$ can be computed.

The solution to the ERM problem can be written as

$$\hat{w}^T = \sum_{i=1}^{n} x_i^T c_i \Rightarrow f_{\hat{w}}(x) = w^T x = \sum_{i=1}^{n} x_i^T x \, c_i = \sum_{i=1}^{n} c_i \, k(x, x_i)$$

(representer theorem)

Kernel methods

$$f(x) = \sum_{i=1}^{n} c_i \, k(x, x_i)$$

Common kernels:

- Linear $k(x, x') = x^T x'$
- Polynomial $k_d(x, x') = (x^T x' + 1)^d$
- Gaussian $k_\sigma(x, x') = \exp - \frac{\|x - x'\|^2}{2\sigma^2}$

Kernel methods

Kernel methods are very flexible, they can approximate any continuous functions given enough data*.

They do not scale well: one must handle the kernel matrix $K_{nn} \in \mathbb{R}^{n \times n}$ with entries $k(x_i, x_j)$.
Hence, the computational complexity to determine the function is typically $\mathcal{O}(n^3)$ in time and $\mathcal{O}(n^2)$ in space and some approximation is needed.

*Andreas Christmann and Ingo Steinwart. Support vector machines. 2008

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. 2006

Falkon

A modern algorithm to efficiently extend kernel methods to large scale problems ($n = \mathcal{O}(10^7)$).

- Nyström approximation (subsampling)
- Iterative solvers
- Approximate preconditioning (w/ Nyström)
- Efficient (multi-)GPU implementation

Kernel methods through the roof: handling billions of points efficiently, Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi, arXiv:2006.10350 [cs.LG]

https://github.com/FalkonML/falkon

Falkon

It considers functions of the following kind (Nyström)

$$f(x) = \sum_{i=1}^{M} c_i \, k(x, x_i) \, ,$$

where $\{x_1, \ldots, x_M\} \subset \{x_1, \ldots, x_n\}$ are inducing points sampled uniformly at random, called *centers.*

Optimal statistical bounds can be obtained with $M = \mathcal{O}(\sqrt{n}) \;\rightarrow \mathcal{O}(n)$ cost in space.

Statistical (supervised) learning

Given $(x_i, y_i)_{i=1}^{n} \sim p^n$, find $\hat{f}$ with small

$$L(\hat{f}) = \mathbb{E}_p[\ell(\hat{f}(y), x)]$$

expected error

Statistical (supervised) learning

Error decomposition

excess risk

$$\mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right] - \min L = \mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right] - L(f_\lambda) + L(f_\lambda) - \min L$$

Statistical (supervised) learning

Error decomposition

excess risk

$$\mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right] - \min L = \mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right] - L(f_\lambda) + L(f_\lambda) - \min L$$

"test" error of ERM

best "test" error

ideal ERM error

Statistical (supervised) learning

Error decomposition

excess risk

estimation error/ variance

approximation error/ bias

$$\mathbb{E}[L(\hat{f}_\lambda)] - \min L = \mathbb{E}[L(\hat{f}_\lambda)] - L(f_\lambda) + L(f_\lambda) - \min L$$

Bounds $\lesssim \frac{\lambda}{n} + \frac{1}{\lambda}$

"test" error of ERM

best "test" error

ideal ERM error

$$\left(\lesssim g(\lambda, n) + h(\lambda)\right)$$

Statistical (supervised) learning

Optimization

$$\min_{w} L(f_w)$$

Gradient descent

$$\hat{w}_{t+1} = \hat{w}_t + \gamma_t \nabla \hat{L}\left(f_{\hat{w}_t}\right)$$

Statistical (supervised) learning

Computational error

$$\mathbb{E}\left[L\left(f_{\hat{w}_t}\right)\right] - \min L =$$

$$\underbrace{\mathbb{E}\left[L\left(f_{\hat{w}_t}\right)\right] - \mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right]}_{\text{Computational “test” error}} + \underbrace{\mathbb{E}\left[L\left(\hat{f}_\lambda\right)\right] - L(f_\lambda)}_{\text{variance}} + \underbrace{L(f_\lambda) - \min L}_{\text{bias}}$$

Falkon

Squared loss + L2 (kernel ridge regression)

Loss and penalty are quadratic → linear system $(K_{nn} + \lambda n I)\boldsymbol{c} = \boldsymbol{y}$.

→ $(K_{nM}^T K_{nM} + \lambda n K_{MM})\boldsymbol{c} = K_{nM}^T \boldsymbol{y}, \quad \boldsymbol{c} \in \mathbb{R}^M$.

Solvable directly in $\mathcal{O}(nM^2 + M^3)$ time and $\mathcal{O}(M^2)$ space.

Falkon

Squared loss + L2 (kernel ridge regression)

- Interative solvers, such as conjugate gradient.

- Approximate preconditioning

$$PP^T = (K_{nM}^T K_{nM} + \lambda n K_{MM})^{-1}$$
$$\rightarrow \tilde{P}\tilde{P}^T = \left(\frac{n}{M} K_{MM}^2 + \lambda n K_{MM}\right)^{-1}$$

Optimal bounds in $\mathcal{O}(n\sqrt{n}\log n)$ in time and $\mathcal{O}(n)$ in space.

Similar considerations are valid for *LogFalkon.*

Falkon

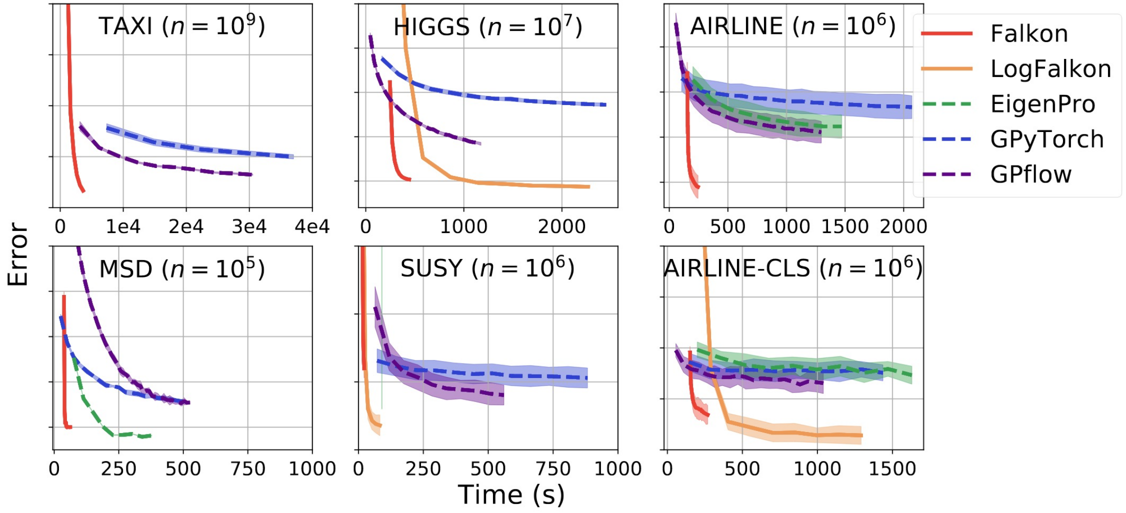
Kernel methods through the roof: handling billions of points efficiently, Giacomo Meanti, Luigi Carratino, Lorenzo Rosasco, and Alessandro Rudi, arXiv:2006.10350 [cs.LG]

	MNIST $n = 6 \cdot 10^4, d = 780$	CIFAR10 $n = 6 \cdot 10^4, d = 1024$	SVHN $n = 7 \cdot 10^4, d = 1024$
Falkon	10.9 s	13.7 s	17.2 s
InCoreFalkon	6.5 s	7.9 s	6.7 s
ThunderSVM	19.6 s	82.9 s	166.4 s

Table 2: Accuracy and running-time comparisons on large scale datasets.

	TAXI $n \approx 10^9$		HIGGS $n \approx 10^7$		YELP $n \approx 10^6, d \approx 10^7$	
	RMSE	time	$1 - \text{AUC}$	time	rel. RMSE	time
Falkon	**311.7±0.1**	**3628±2 s**	0.1804±0.0003	**443±2 s**	**0.810±0.001**	**1008±2 s**
LogFalkon	—		**0.1787±0.0002**	2267±5 s	—	
EigenPro	FAIL		FAIL		FAIL	
GPyTorch	315.0±0.2	37 009±42 s	0.1997±0.0004	2451±13 s	FAIL	
GPflow	313.2±0.1	30 536±63 s	0.1884±0.0003	1174±2 s	FAIL	
	TIMIT $n \approx 10^6$		AIRLINE $n \approx 10^6$		MSD $n \approx 10^5$	
	c-error	time	rel. MSE	time	rel. error	time
Falkon	32.27±0.08 %	**288±3 s**	**0.758±0.005**	**245±5 s**	$(4.4834\pm0.0008)\times10^{-3}$	**62±1 s**
EigenPro	**31.91±0.01 %**	1737±8 s	0.785±0.005	1471±11 s[1]	$\mathbf{(4.4778\pm0.0004)\times10^{-3}}$	378±8 s
GPyTorch	—		0.793±0.005	2069±50 s	$(4.5004\pm0.0010)\times10^{-3}$	502±2 s
GPflow	33.78±0.14 %	2672±10 s	0.782±0.005	1297±2 s	$(4.4986\pm0.0005)\times10^{-3}$	525±5 s
	AIRLINE-CLS $n \approx 10^6$		SUSY $n \approx 10^6$			
	c-error	time	c-error	time		
Falkon	31.5±0.2 %	**186±1 s**	19.67±0.02 %	**22±0 s**		
LogFalkon	**31.3±0.2 %**	1291±3 s	**19.58±0.03 %**	83±1 s		
EigenPro	32.5±0.2 %	1629±1 s[1]	20.08±0.55 %	90±0 s[2]		
GPyTorch	32.5±0.2 %	1436±2 s	19.69±0.03 %	882±9 s		
GPflow	32.3±0.2 %	1039±1 s	19.65±0.03 %	560±11 s		

[1] Using a random subset of 1×10^6 points for training. [2] Using a random subset of 6×10^5 points for training.

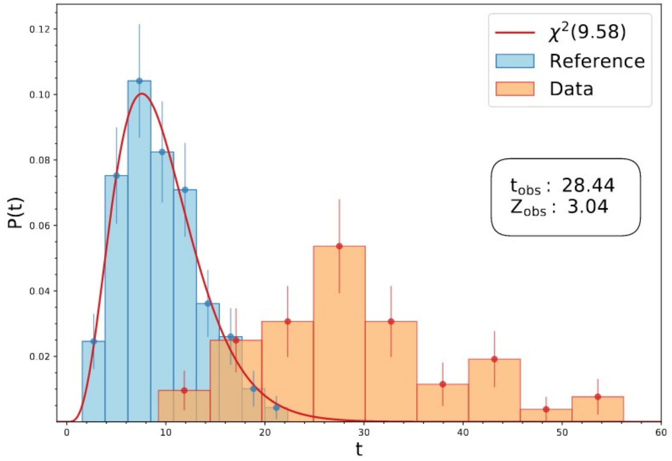


Falkon for NPLM

- Reference sample S_0 and data sample S_1
- (weighted) logistic loss to learn $\mathrm{f}_{\hat{w}} \approx \log \frac{n(x|1)}{n(x|0)}$ and compute $t_{\hat{w}}(S_1)$.
- Efficient algorithm based on kernel methods (Falkon).

Pipeline:

- Reconstruct the distribution under the null hypothesis with data coming from the reference.
- Compute the test statistics for the actual data sample.



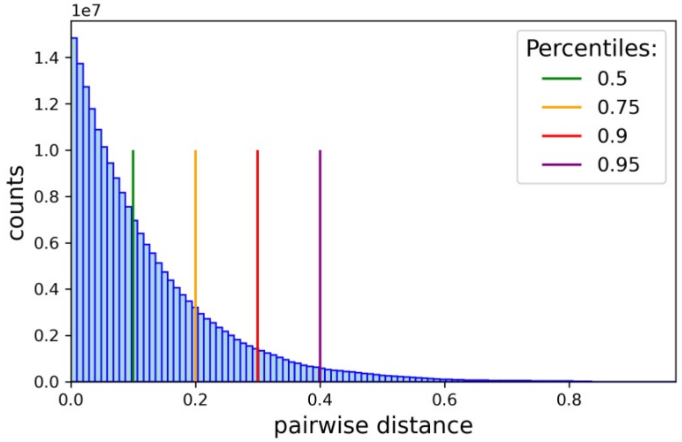
Falkon for NPLM

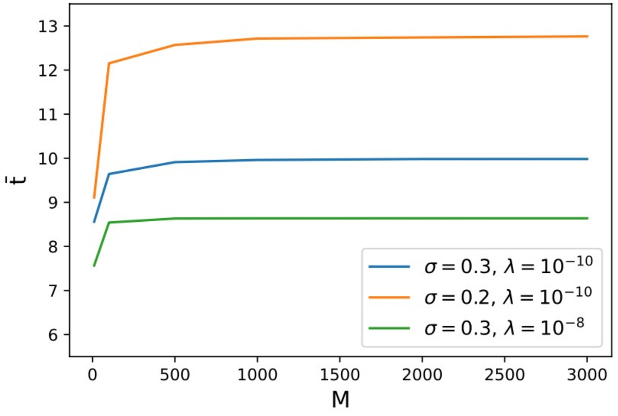
Falkon has three main hyperparameters (M, σ, λ)

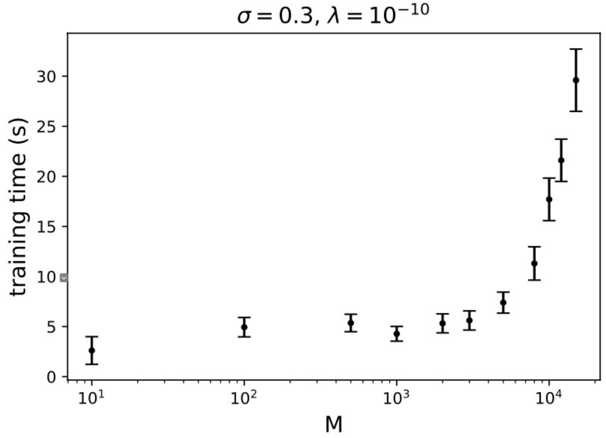
Typically, they can be tuned using cross-validation.

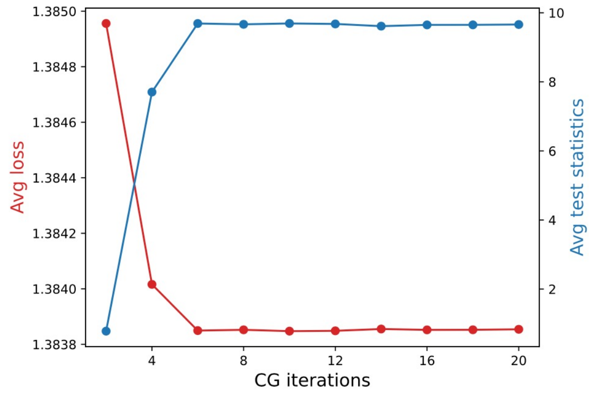
In NPLM applications we do not do that to preserve model-independence.

→ mix of heuristics, statistical considerations and efficency



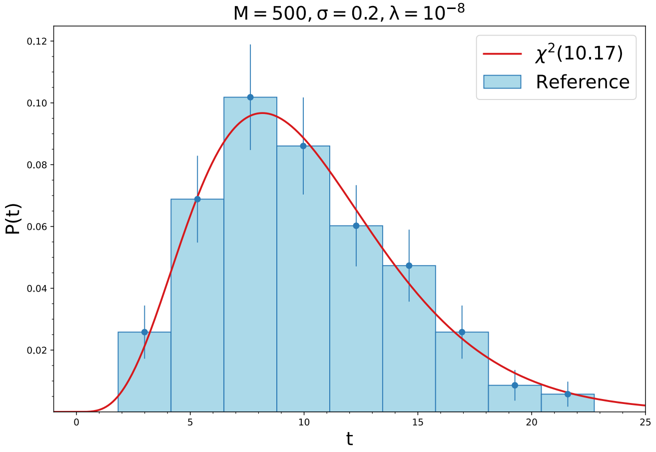


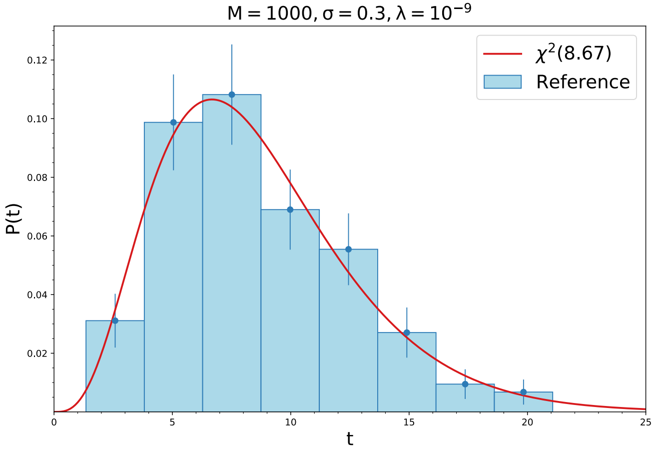


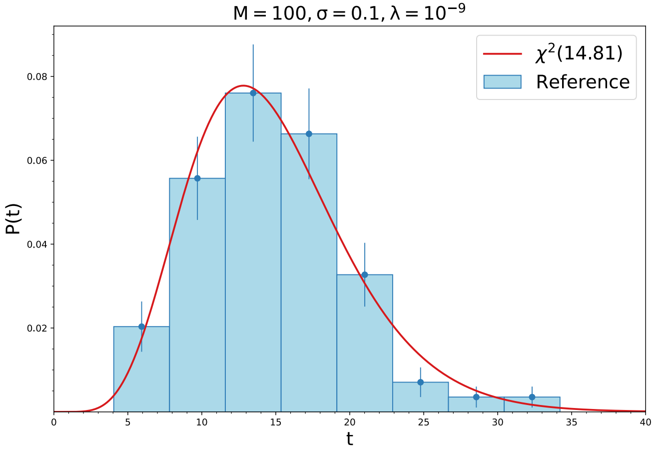


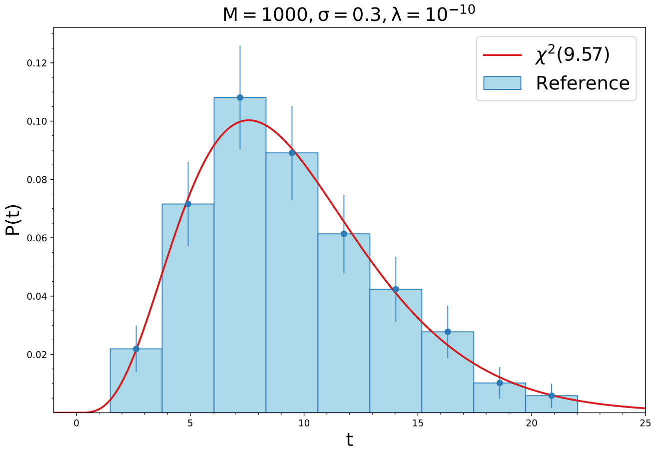
Falkon for NPLM

Compatibility of the null distribution with a χ^2.

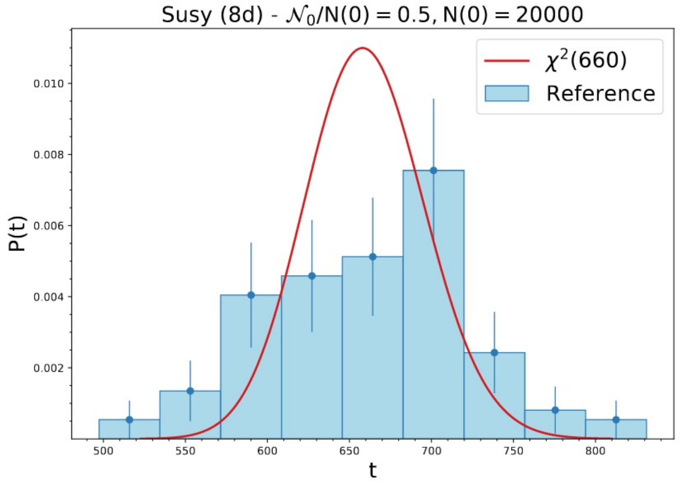








It breaks down if reference is too small.



Applications

- Learning new physics
- Data quality monitoring
- Validation

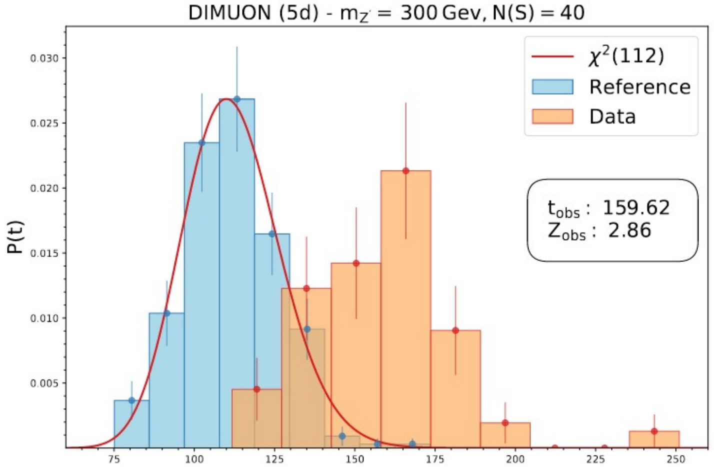
Learning new physics

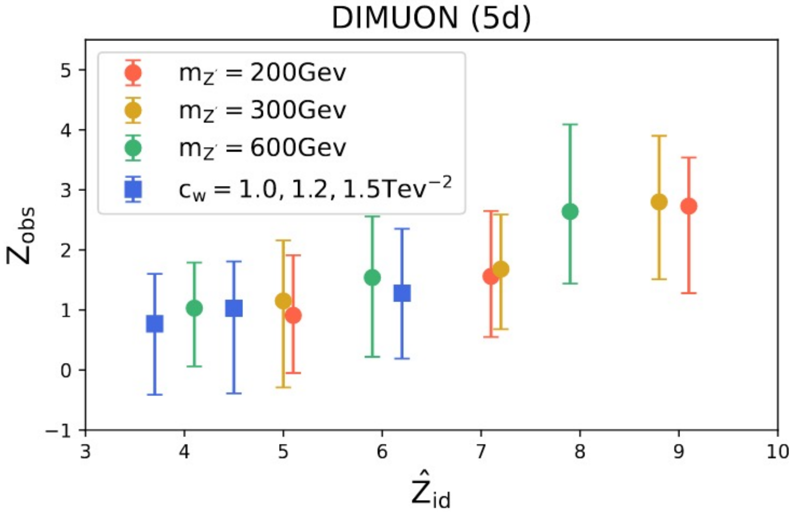
DIMUON :

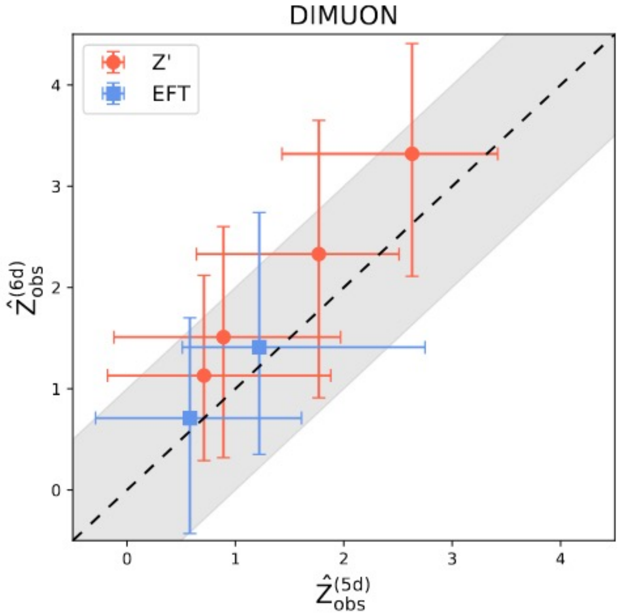
$pp \to \mu^+\mu^-,$

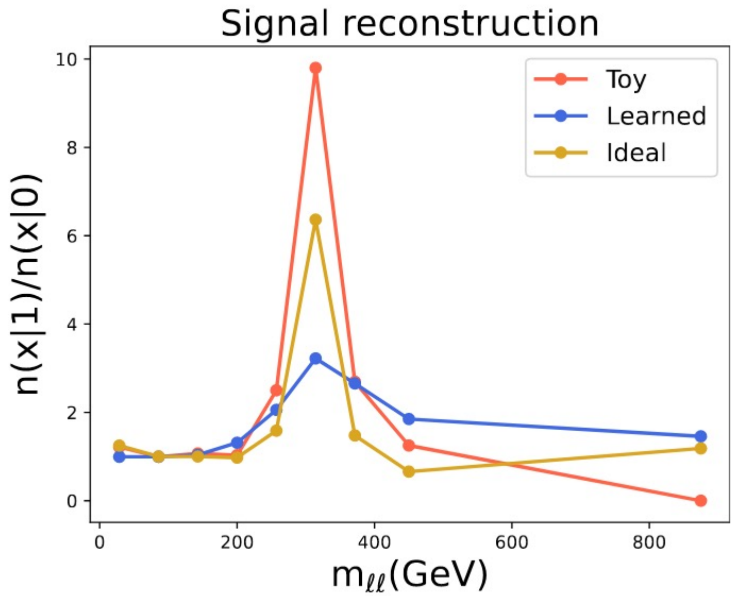
$x = [p_{T1}, p_{T2}, \eta_1, \eta_2, \Delta\phi],$

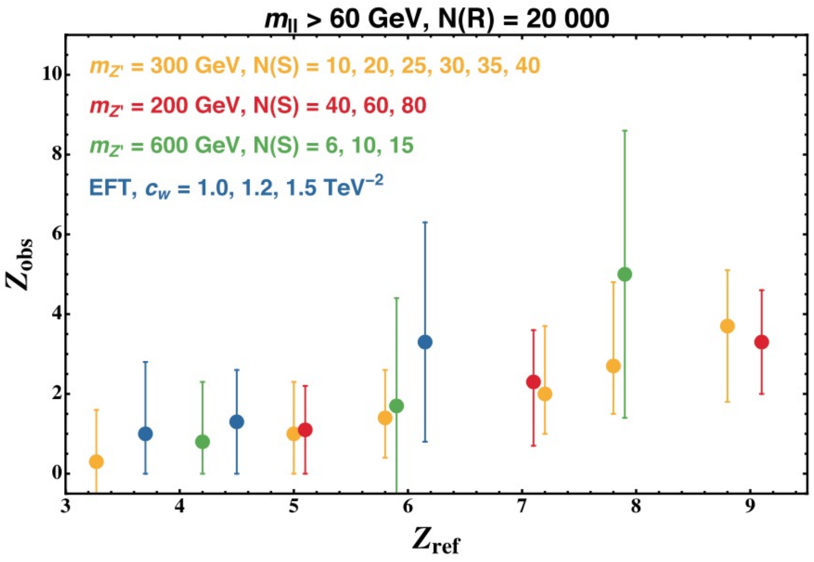
$m_{Z'} = 200{,}300{,}600$ GeV
EFT $c_w = 1.0{,}1.2{,}1.5$ TeV^{-2}





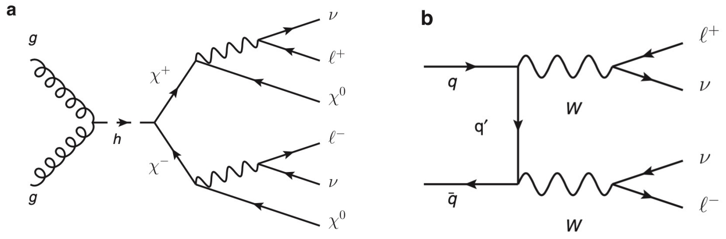




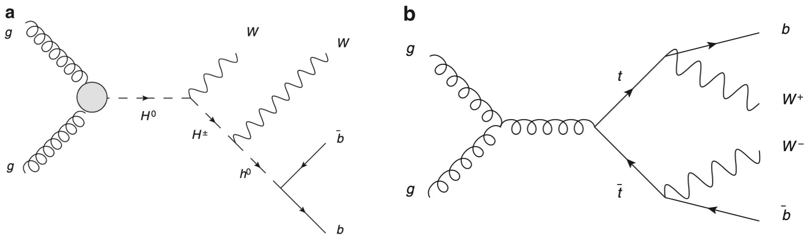


Learning new physics

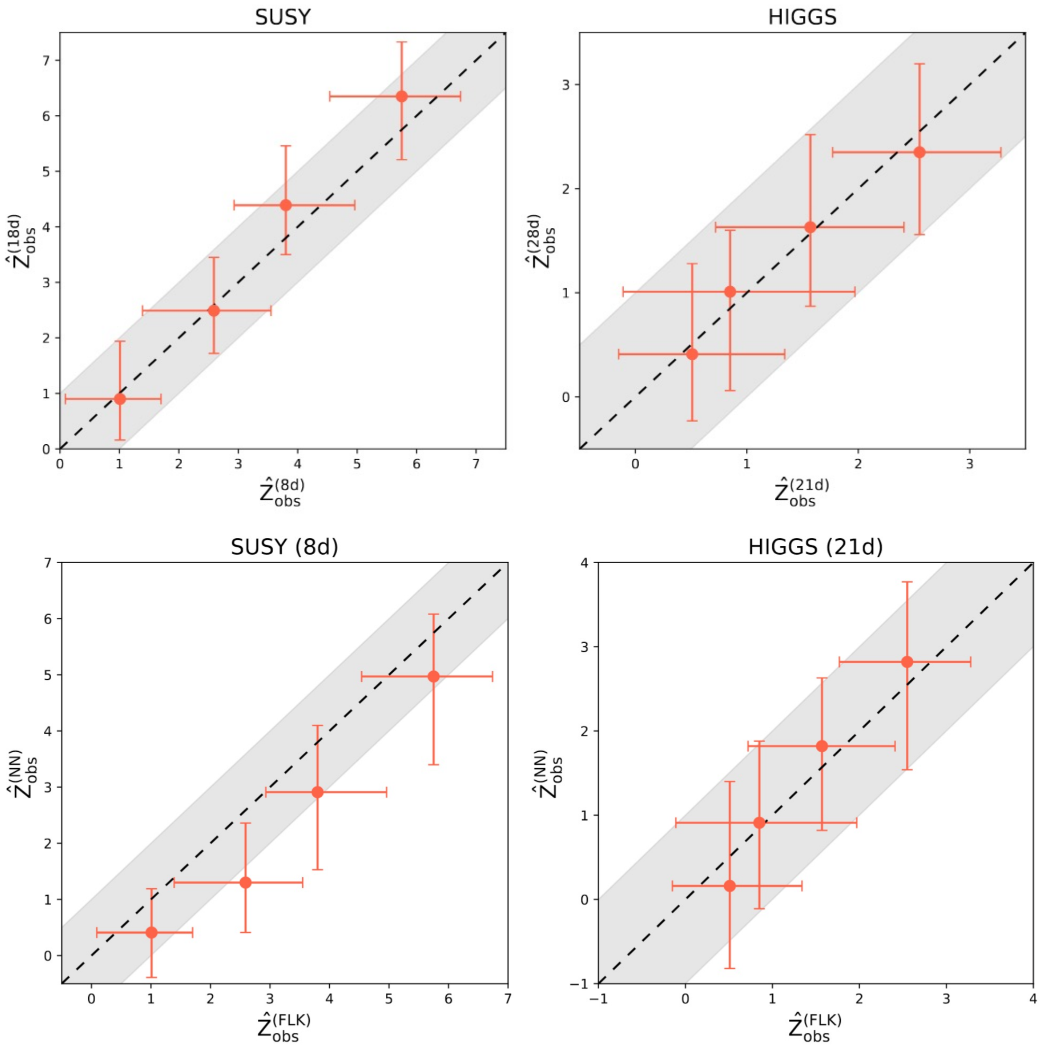
SUSY (8d):



HIGGS (21d):



Baldi et al, arXiv:1402.4735[hep-ph]



Learning new physics

Model	DIMUON	SUSY	HIGGS
FLK	$(\mathbf{44.9 \pm 3.4})$ **s**	$(\mathbf{18.2 \pm 1.2})$ **s**	$(\mathbf{22.7 \pm 0.4})$ **s**
NN	(4.23 ± 0.73) h	(73.1 ± 10) h	(112 ± 9) h

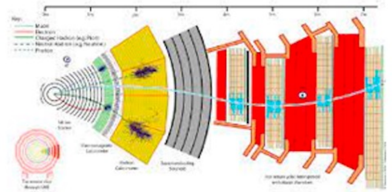
For details about the NN models (architectures, training,…), see the following papers

Raffaele Tito D'Agnolo and Andrea Wulzer. *Learning New Physics from a Machine*. Phys. Rev. D, 99(1):015014, 2019. arXiv:1806.02350 [hep-ph]

Raffaele Tito D'Agnolo, Gaia Grosso, Maurizio Pierini, Andrea Wulzer, and Marco Zanetti. *Learning multivariate new physics*. Eur. Phys. J. C, 81(1):89, 2021. arXiv:1912.12155 [hep-ph]

M.L., Gianvito Losapio, Marco Rando, Gaia Grosso, Andrea Wulzer, Maurizio Pierini, Marco Zanetti, Lorenzo Rosasco, *Learning new physics efficiently with nonparametric methods,* To appear in EPJC, arXiv: 2204.02317 [hep-ph]

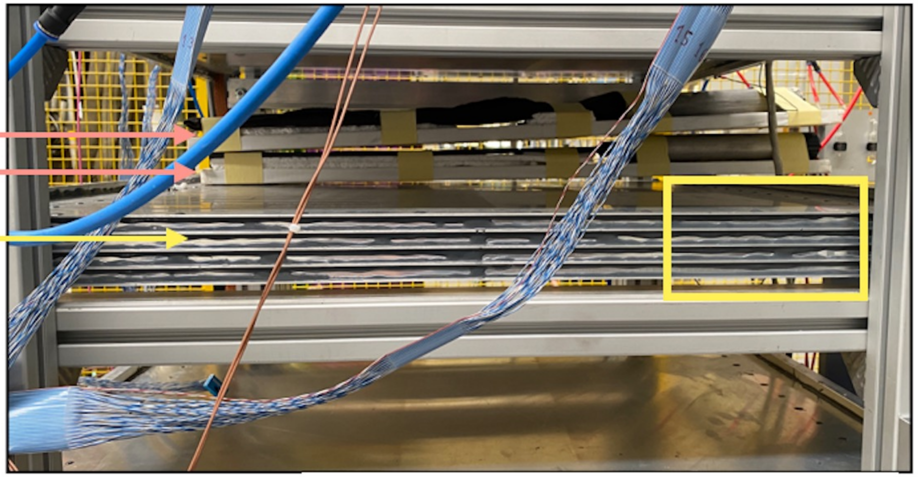
*n*D DQM (from Gaia's presentation)



Online monitoring of a DT chamber:

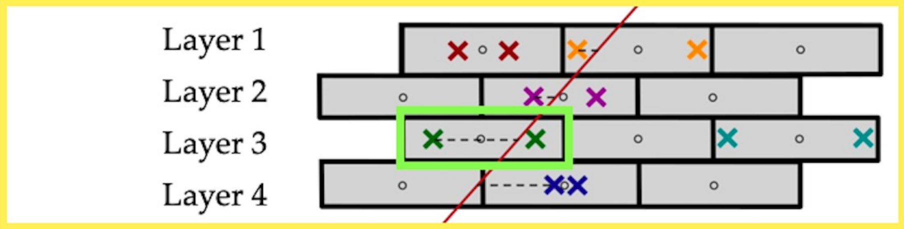
Setup (Legnaro INFN national laboratory):

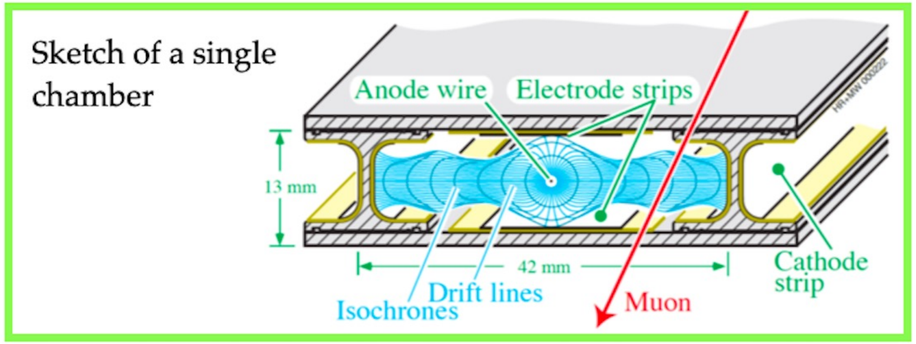
- 2 scintillators as signal trigger
- 1 drift tube chamber: 4 layers 16 wires each (16x4=64 wires)
- Source of signals: cosmic muons (triggered rate ~3 MHz)
- **Event**: muon track reconstructed interpolating 3/4 hits (one per layer)



Observables (6D problem):

- 4 drift times $[t_{\text{drift},1}, t_{\text{drift},2}, t_{\text{drift},3}, t_{\text{drift},4}]$: time for the ionised electrons to reach the wire from the interaction point (v_{drift} = cm/s) .
- θ: reconstructed track angle
- N_{hits}: average number of hits per time window ("orbit")





Dipartimento di Fisica e Astronomia Galileo Galilei
UNIVERSITÀ DEGLI STUDI DI PADOVA
UniGe | MaLGa

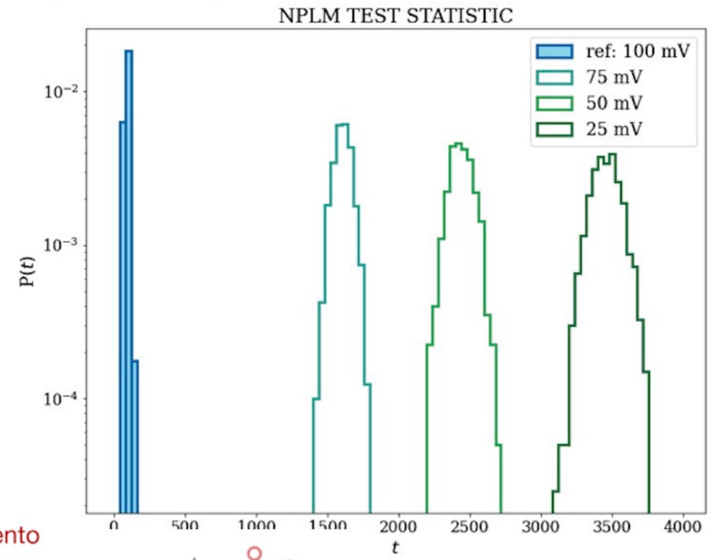
nD DQM (from Gaia's presentation)

Online monitoring of a DT chamber:

- **Reference sample:** long run in optimal conditions
- **Anomalous samples**: short runs acquired in presence of a controlled anomaly in the value of the **threshold tension** of the DT chamber

- Result of the test statistics

Complete separation of the distributions!



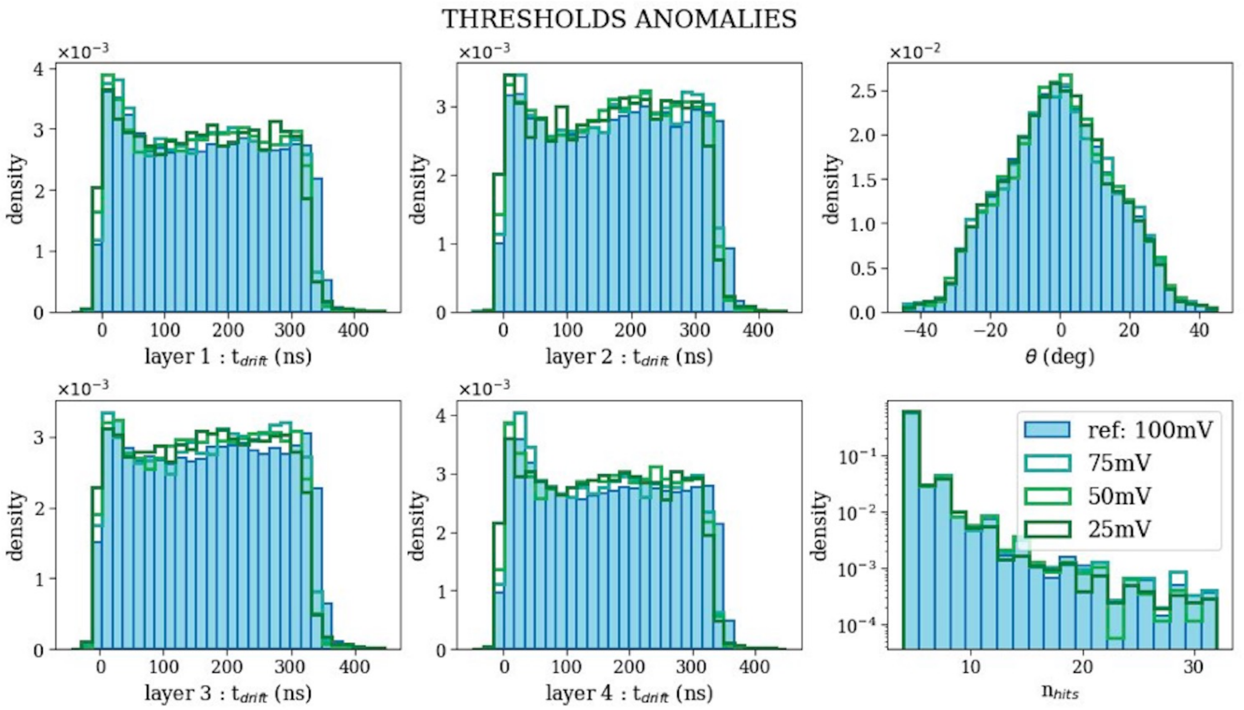
NPLM with Falkon

$M = 50, \sigma = 4.84, \lambda = 10^{-7}$

$N(D) = 5000$

$N_{\text{ref}} = 200\,000$

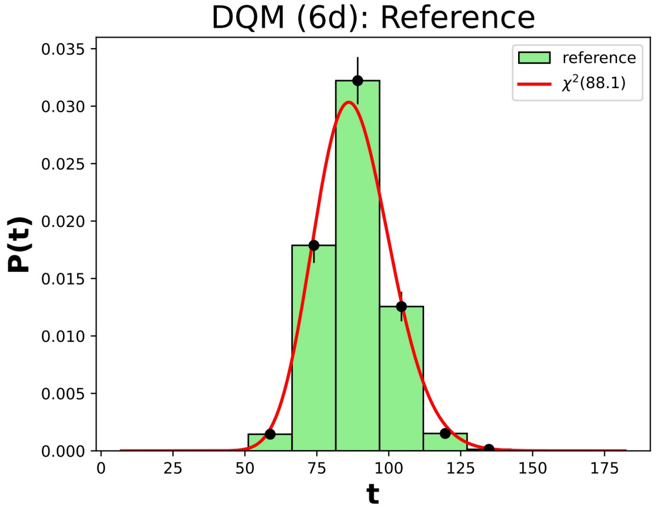
Execution time: ~ 1.5 s

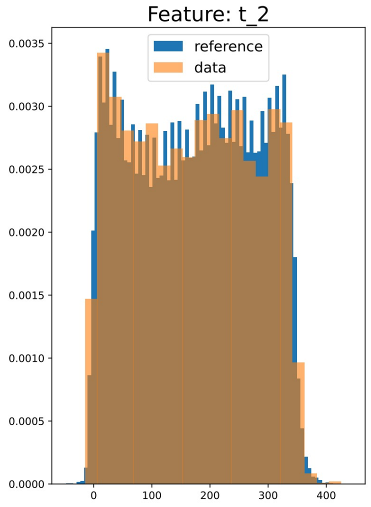


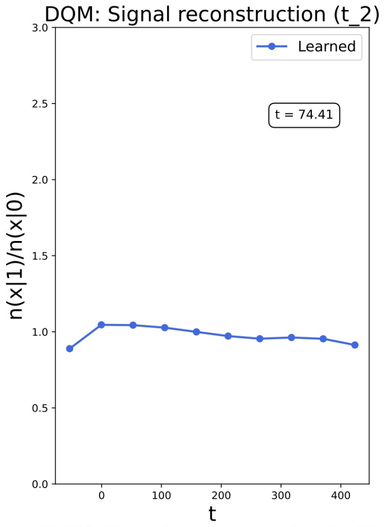
Distribution of the observables at different values of the threshold tension

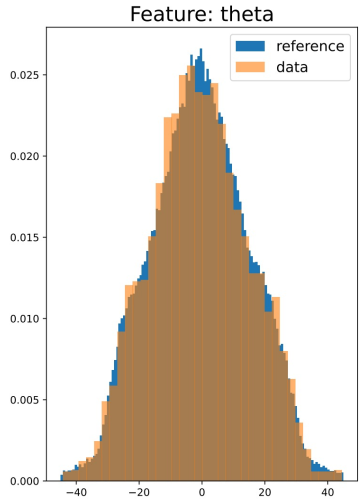
DQM

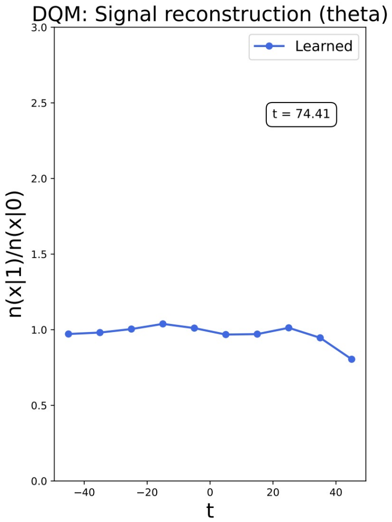

WORK IN PROGRESS
CHECK BACK SOON!

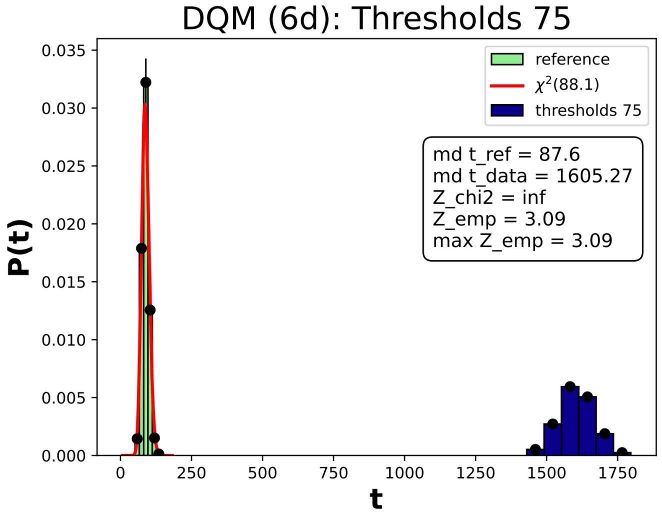

DQM (6d): Reference
reference
$\chi^2(88.1)$
P(t)
t

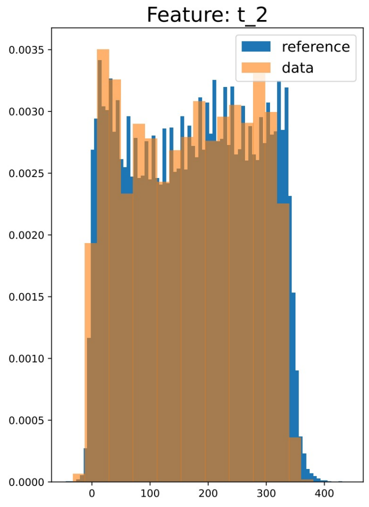

Feature: t_2
reference
data

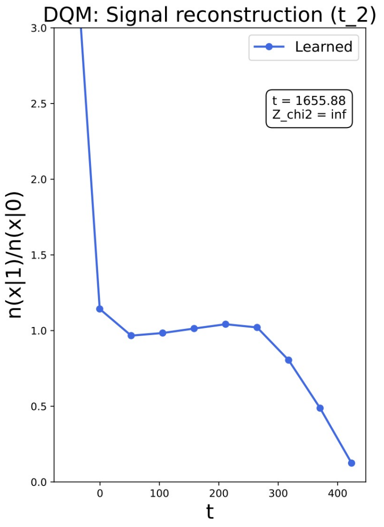

DQM: Signal reconstruction (t_2)
Learned
t = 74.41
n(x|1)/n(x|0)
t

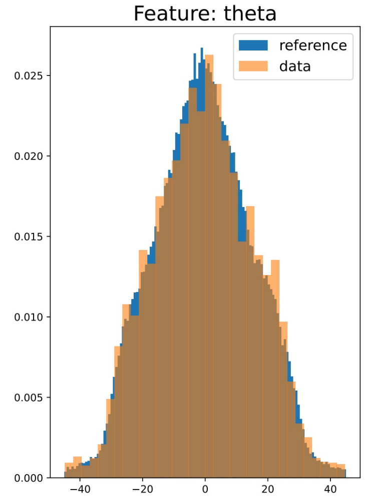

Feature: theta
reference
data

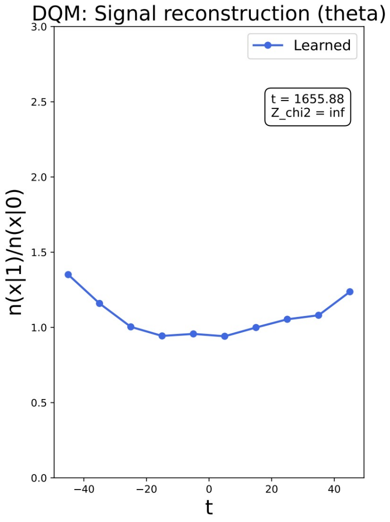

DQM: Signal reconstruction (theta)
Learned
t = 74.41
n(x|1)/n(x|0)
t


DQM (6d): Thresholds 75
reference
$\chi^2(88.1)$
thresholds 75
md t_ref = 87.6
md t_data = 1605.27
Z_chi2 = inf
Z_emp = 3.09
max Z_emp = 3.09
P(t)
t


Feature: t_2
reference
data


DQM: Signal reconstruction (t_2)
Learned
t = 1655.88
Z_chi2 = inf
n(x|1)/n(x|0)
t

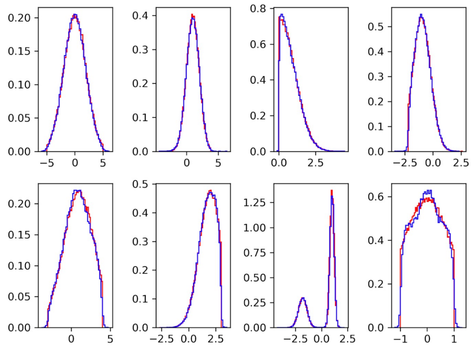

Feature: theta
reference
data

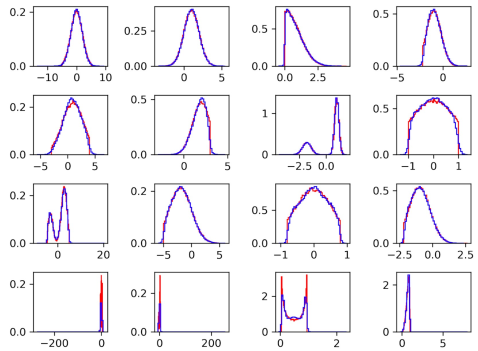

DQM: Signal reconstruction (theta)
Learned
t = 1655.88
Z_chi2 = inf
n(x|1)/n(x|0)
t

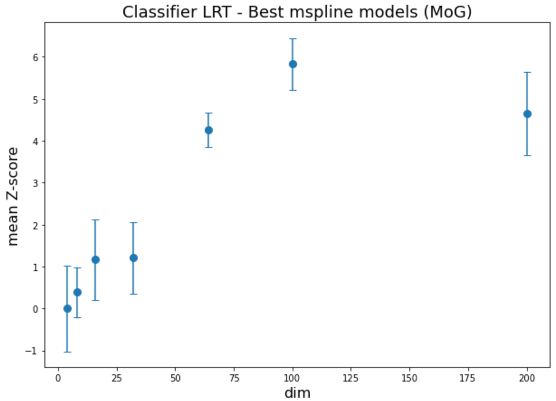
Validation of generative models

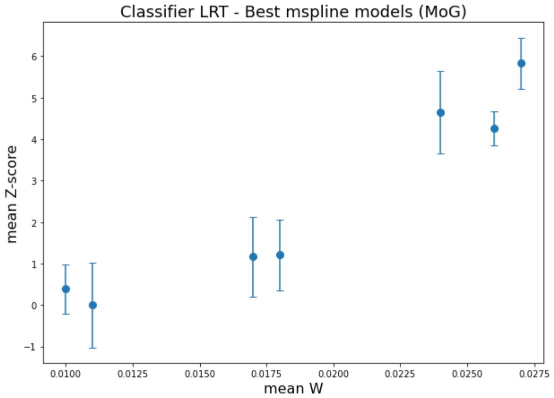
With Riccardo Torre and Humberto Reyes (Unige/INFN).

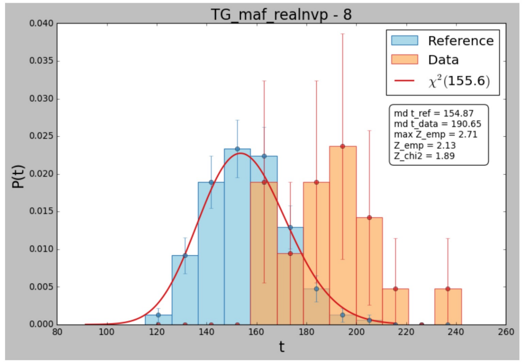
Normalizing flows in high dimensions (up to $d = 200$).

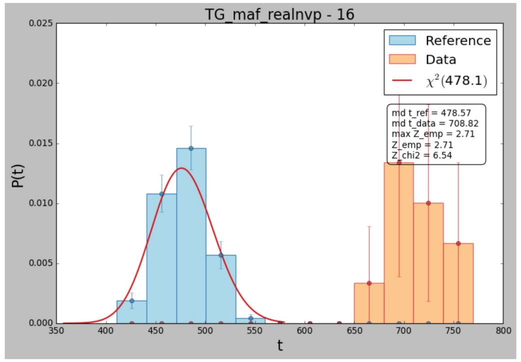












Testing the boundaries: Normalizing Flows for higher dimensional data sets, Humberto Reyes-Gonzalez, Riccardo Torre, ACAT 2021, arXiv:2202.09188 [stat.ML]

What is coming and to do list

- Comparison among AD models
- In-depth analysis of ML driven GoF tests

- Systematic uncertainties
- Hyperparameter tuning
- Selection of centers