Machine Learning and Colliders

Elena Fol

R. Tomás, G. Franchetti

CERN

Goethe-University Frankfurt

Part I. Introduction to Machine Learning

Teaching machines to learn from experience

- Tasks that are extremely easy and obvious for us are difficult to program in traditional ways
- Impossible to learn every possible rule to perform a task
  - ➢ learn from examples instead

Teaching machines to learn from experience

- Tasks that are extremely easy and obvious for us are difficult to program in traditional ways
- Impossible to learn every possible rule to perform a task
  - ➢ learn from examples instead

Teaching machines to learn from experience

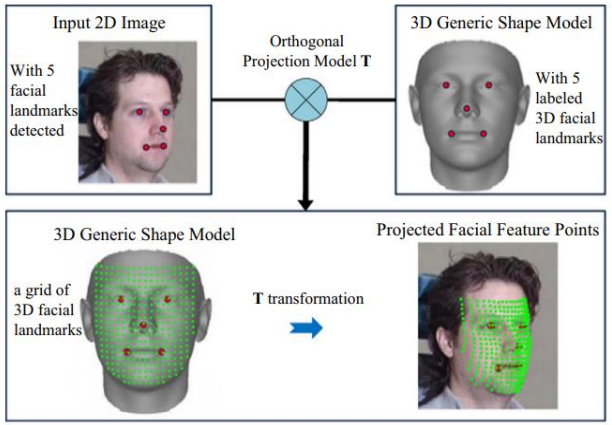
- Tasks that are extremely easy and obvious for us are difficult to program in traditional ways
- Impossible to learn every possible rule to perform a task
  - ➢ learn from examples instead

→ **Cat?**

MNIST handwritten digits dataset

http://yann.lecun.com/exdb/mnist/

Y. LeCun, et.al, "Gradient-based learning applied to document recognition"



D. Changxing, T. Dasheg, "Pose-invariant face recognition with homography-based normalization"

AlphaGo by Google

- First match against Go European champion in 2015,
  5:0 for AlphaGo
- In 2017 AlphaGo surpassed the performance of its previous versions and became the strongest Go player of all time *

* Silver, David et al. “Mastering the game of Go without human knowledge.” *Nature* 550 (2017): 354-359.

Face recognition and reconstruction

- Automatic detection of semantic regions
- Specific "layers" are sensitive to certain regions (e.g. eyes, nose, lips)

Medical Research: COVID-19

- > 1000 articles no *arxiv.org* related to ML applications to COVID19 research.
- Mostly image processing (x-ray images) and modeling of transmission.

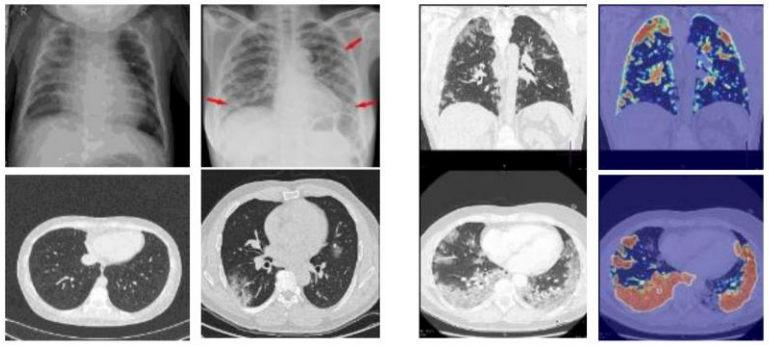
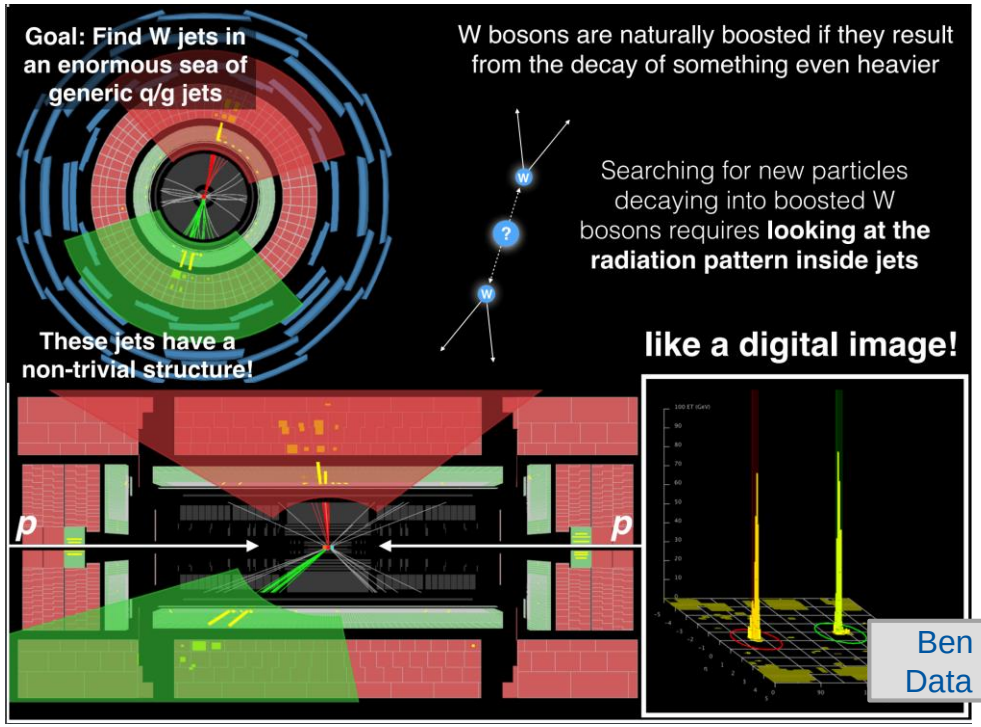


Figure 1: Example X-ray and CT images of normal and COVID-19 cases, and outputs of detection and infected region localization with NABLA-3 network.



High Energy Physics

- ML is used in dark matter search, jets recognition, particle tracking, neutrino classification, shower simulations

Ben Nachman, CERN
Data Science Seminars

Relevant ML concepts and definitions

*"... computer programs and algorithms that automatically **improve with experience** by **learning from examples** with respect to some class of task and performance measure, **without being explicitly programmed**." **

Supervised Learning

- Input/output pairs available
- Make prediction for unknown input based on experience from given examples

Object detection in computer vision, speech recognition, predictive control

Unsupervised Learning

- Only input data is given
- Learn structures and patterns

Anomaly detection, pattern recognition, clustering, dimensionality reduction

Reinforcement Learning

- No training data
- Interact with an environment
- Trying to learn optimal sequences of decisions

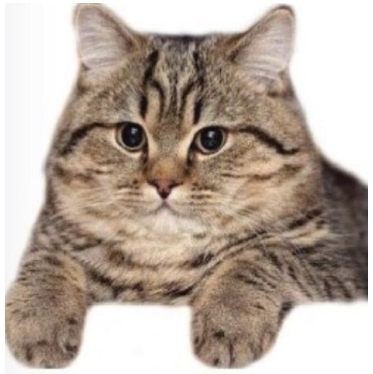
Robotics, industrial automation, dialog systems

* Thomas M. Mitchell. Machine Learning. McGraw-Hill, Inc., New York, 1997.

Supervised Learning

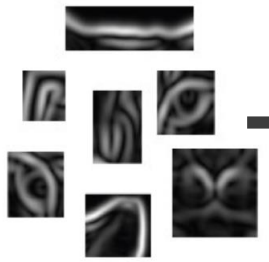
How does the learning work in practice?

1. Collect examples



Data sample

2. Preliminary processing



Input Features

3. Training, tuning, validation

Model

4. Prediction

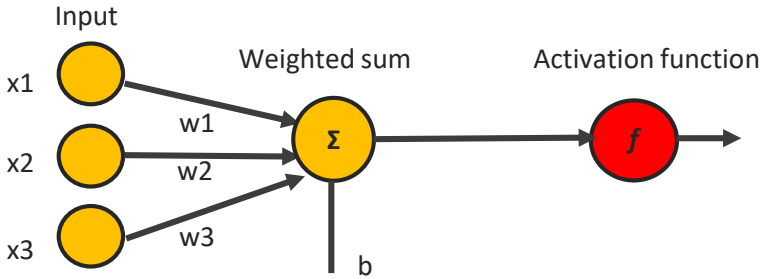
Cat

Output
(target variable)

Supervised Learning

Neural Network as an example:

- **Weights w** of the **inputs x**
- **Activation function f**
- **Output y** of a single neuron: $y = f(\sum x_i w + b)$

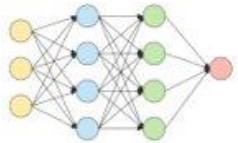


Universal Approximation Theorem: A **simple neural network** including only a single hidden layer can approximate any bounded continuous target function with arbitrary small error. *(Cybenko, 1989, for sigmoid activation functions)*

How does the learning work in practice?

example 1
example 2
example 3
.
.
.

Training input data



Function with **adjustable parameters** (weights, bias)

Model output

$$y = f\left(\sum x_i w_i + b\right)$$

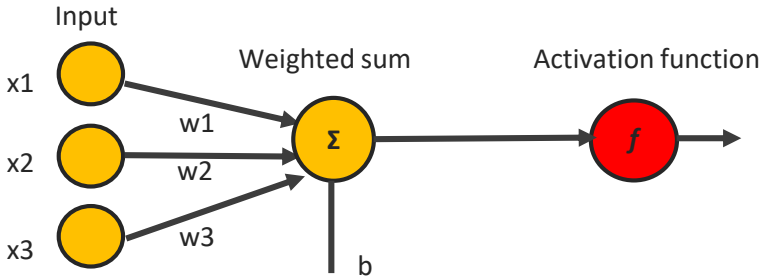
Compute the **loss** (approximation error):
e.g. MSE, MAE

Training output data

Supervised Learning

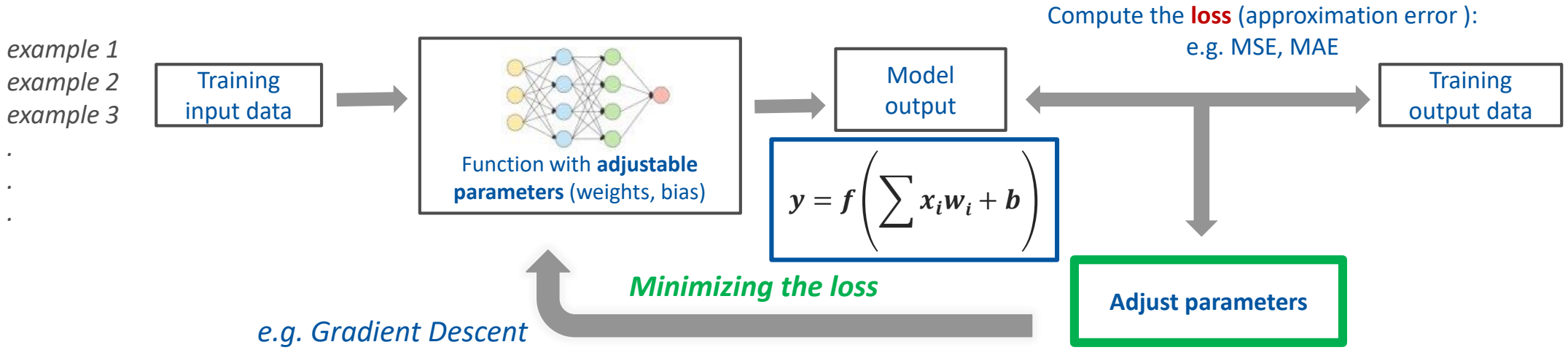
Neural Network as an example:

- **Weights w** of the **inputs x**
- **Activation function f**
- **Output y** of a single neuron: $y = f(\sum x_i w + b)$

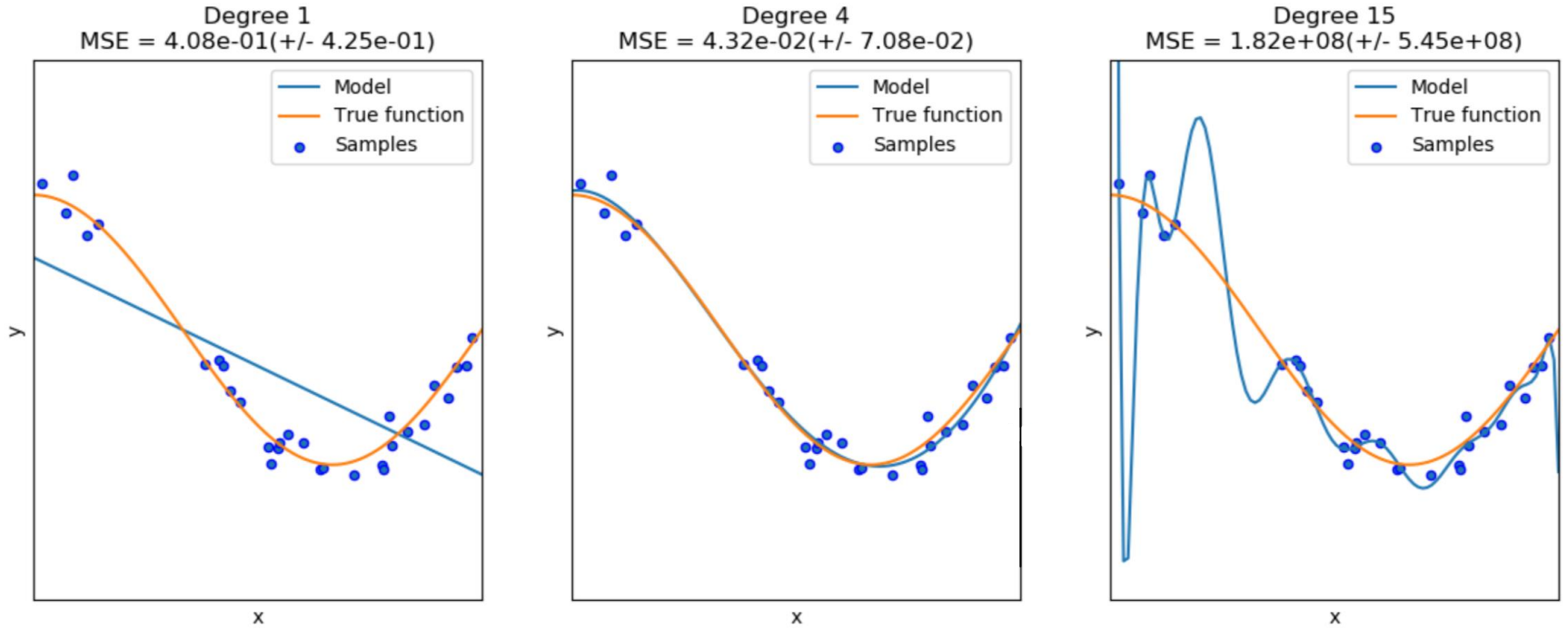


Universal Approximation Theorem: A **simple neural network** including only a single hidden layer can approximate any bounded continuous target function with arbitrary small error. *(Cybenko, 1989, for sigmoid activation functions)*

How does the learning work in practice?



Training and generalization: no perfect model needed!



Simple models underfit

- Derivate from data (high bias)
- Do not correspond to data structure (low variance)

We don‘t want „look up tables“ ⟷ Complex models overfit

- Very low systematical deviation (low bias)
- Very sensitive to data (high variance)

⟷ We don‘t want unreliable prediction

→ Bias-Variance tradeoff

ML is more than Neural Networks…

- **Regression and Classification Models:** *resolve correlation between input variables and dependent target variables*
  - Simple Linear Regression, Multivariate Regression, Logistic regression, Support Vector Machine
- **Dimensionality reduction techniques:** *reduce the number of independent variables (features) without significant decrease on prediction accuracy*
  - Independent Component Analysis, Principle Component Analysis, Features Importance Analysis
- **Decision Trees**: *split the input data based on a sequence of variables (thresholds) to estimate the target output value or to separate data points into regions*
  - Ensemble methods: Train several slightly different models and take majority vote/ average of the prediction
- **Clustering:** *grouping or separating data objects into clusters*
  - Identify hidden patterns in the data, similarities and differences

ML is more than Neural Networks…

- **Regression and Classification Models:** *resolve correlation between input variables and dependent target variables*
  - Simple Linear Regression, Multivariate Regression, Logistic regression, Support Vector Machine
- **Dimensionality reduction techniques:** *reduce the number of independent variables (features) without significant decrease on prediction accuracy*
  - Independent Component Analysis, Principle Component Analysis, Features Importance Analysis
- **Decision Trees**: *split the input data based on a sequence of variables (thresholds) to estimate the target output value or to separate data points into regions*
  - Ensemble methods: Train several slightly different models and take majority vote/ average of the prediction
- **Clustering:** *grouping or separating data objects into clusters*
  - Identify hidden patterns in the data, similarities and differences

Machine Learning is about learning from the data, not about application of a particular "intelligent" technique.

ML and accelerators: motivation

Accelerators

Limitations of traditional optimization and modeling tools?

ML is a powerful tool for prediction and data analysis

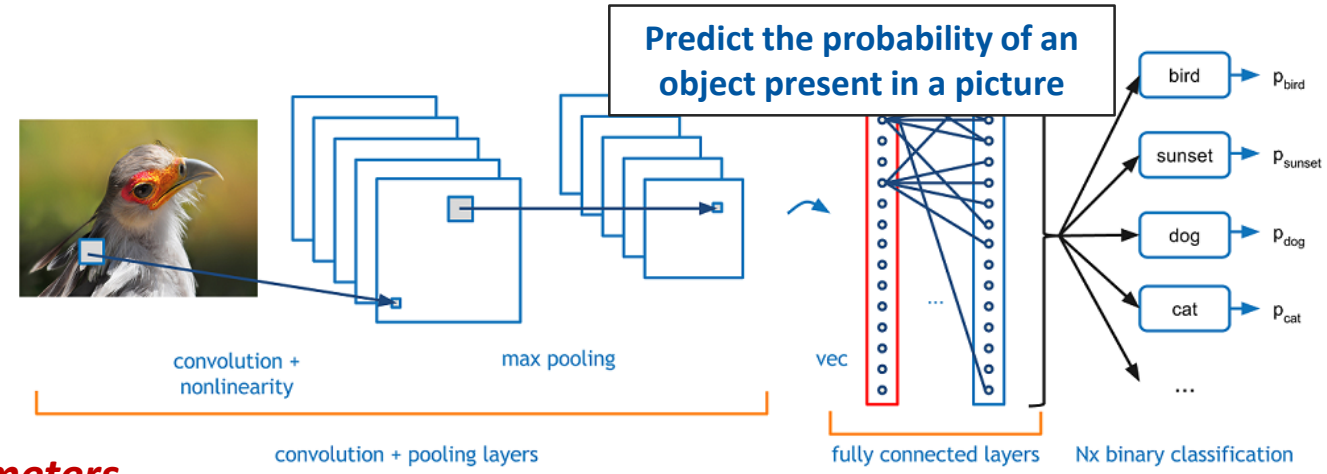
Which limitations can be solved by ML with reasonable effort?

- How to deal with previously unobservable behavior?
- Required computational resources for large amount of optimization targets
- Objective functions, specific rules and thresholds have to be known
- Non-linear interacting sub-systems, rapidly changing behavior.

Machine Learning methods can learn an arbitrary model from given examples without requiring explicit rules.

Adapting typical ML tasks to accelerator-specific problems

Image processing using **Convolutional Neural Networks** is a very common approach in ML research.



→ ***Image-based prediction of multiple beam parameters***

"First steps toward incorporating image based diagnostics into particle accelerator control systems using Convolutional Neural Network", A.L. Edelen et al. NAPAC16 (TUPOA51)

- CNN and fully-connected ANN are used to incorporate image-based and non-image-based data into the model to predict multiple beam parameters via regression.

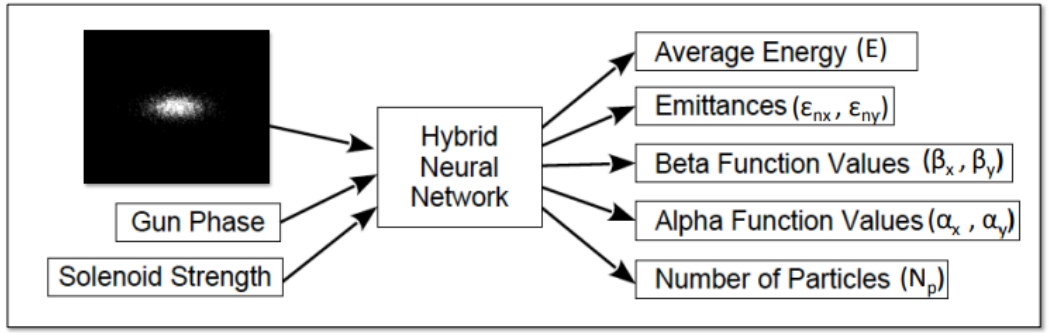
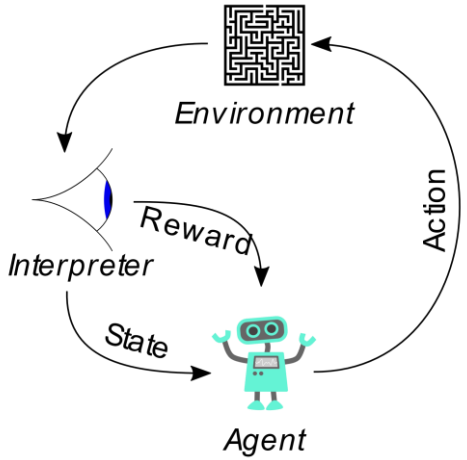


Figure 3: Neural network inputs and outputs.

Adapting typical ML tasks to accelerator-specific problems

Reinforcement Learning is widely applied in robotics in **control systems** in general.

Learn how to walk/ jump/ avoid obstacles



→ *Automatic sub-systems tuning to achieve optimal machine performance*

T. Boltz et al. "Feedback Design for Control of the Micro-Bunching Instability based on Reinforcement Learning", IPAC'19 (MOPGW017)

- Instabilities resulting from self-interaction of the bunch with its own radiation field limits stable operation.
- **Fast and adaptive feedback system** to stabilize the dynamics is required.

→ Reinforcement Learning model based on "actions" = modifications in RF and CSR signal as "reward".

(Coherent Synchrotron Radiation)

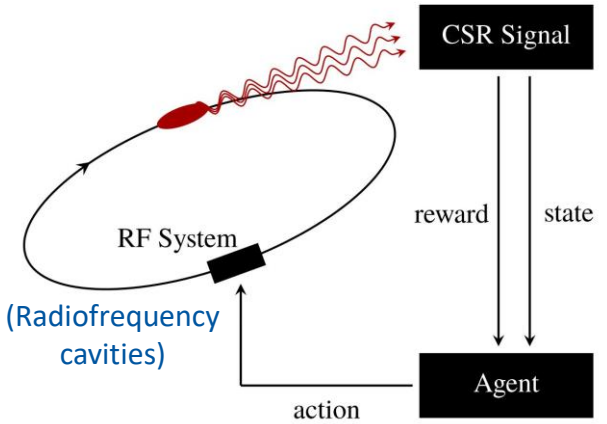
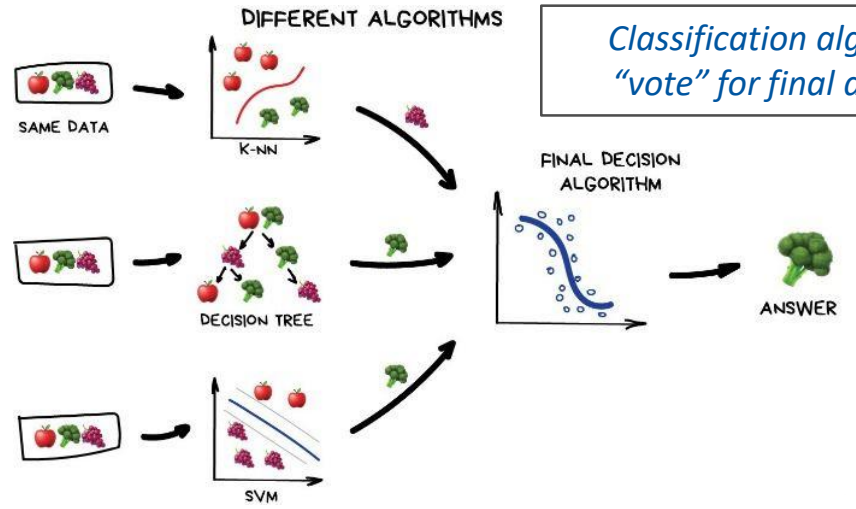


Figure 2: General feedback scheme using the CSR power signal to construct both, the state and reward signal of the Markov decision process (MDP).

Adapting typical ML tasks to accelerator-specific problems

Classification models become more robust by combining several models (Ensemble Learning)

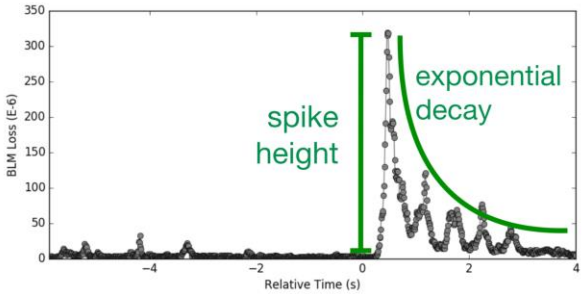
→ ***Robust classification of operational events***



Classification algorithms "vote" for final decision.

Automatic alignment of collimators (*G. Azzopardi et al., "Operational results on the fully automatic LHC collimator alignment", Phys. Rev. Accel. Beams 22, 093001 (2019)*)

- Collimators have to be realigned during operation due to beam parameter changes .
- If beam loss spike is above a pre-selected threshold, the collimator is stopped: **requires an expert** to determine if the collimator actually has touched the beam.



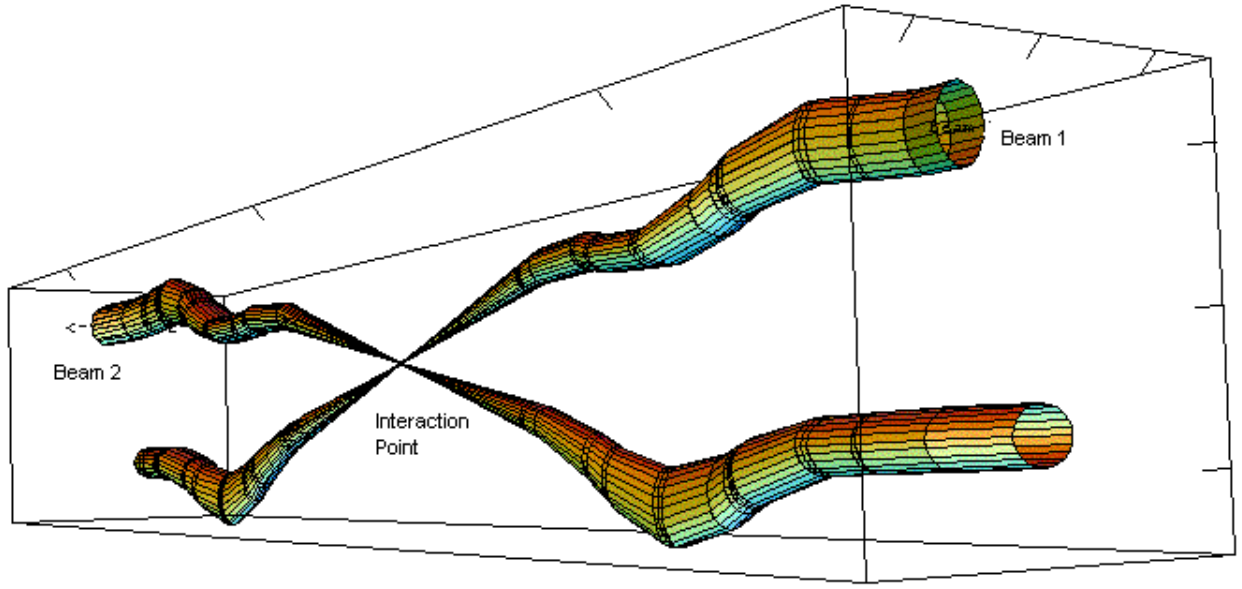
- **Input**: spike height, exponential decay and collimator position.
- **Output**: if collimator is aligned or not. (**classification**)

- Ensemble of several ML methods used – use the majority vote of all models.
- **Developed ML technique became standard for fully automatic LHC collimators alignment.**

ML for Beam Optics Measurements and Corrections at the LHC

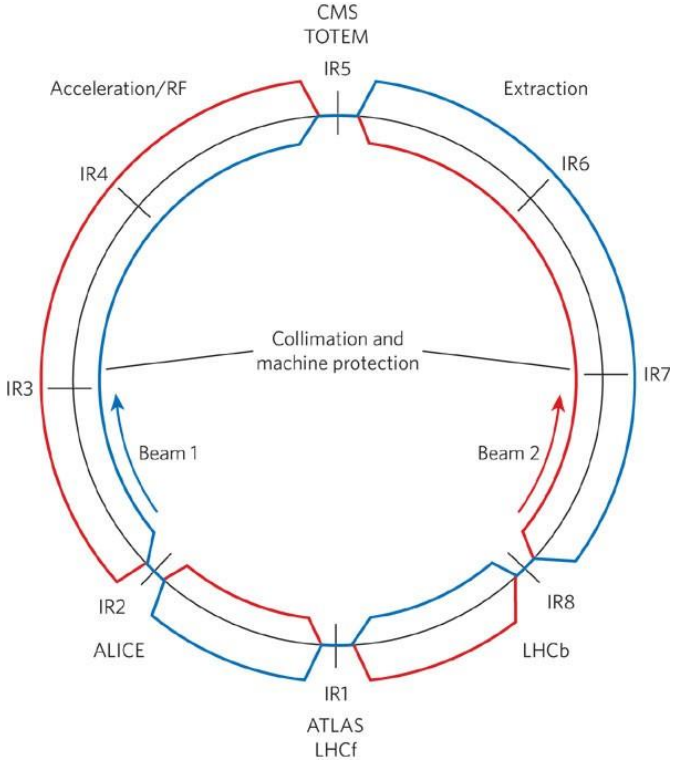
Importance of Beam Optics Control in Colliders:

- **Control of the beam size** in Interaction Points (IPs) to increase the chances of a collision.
  → **Luminosity:** the ratio of the number of collisions in a certain time to the interaction cross-section area.
- Beam Optics imperfections can lead to **machine safety** issues.



Relative beam sizes around IP1 (Atlas) in collision

Courtesy of J. Jowett



ML for Beam Optics Measurements and Corrections at the LHC

- Optics measurements are based on the **signal of Beam Position Monitors**: record the position of the beam at several thousands of turns.
- Corrections aim to **minimize the difference between the measured and design optics** by changing the strength of quadrupole magnets installed around the ring.

→ ***Tasks to be (potentially) solved using Machine Learning***

Detection of BPM failures

- Robust optics measurements rely on BPMs integrity.
- Applying traditional techniques, few faulty BPMs remain in the data.

→ Detection of BPM failures **prior to optics computation** using unsupervised learning.

Magnetic field errors

- The deviations from design optics are caused by magnetic field errors.

→ Estimation of **field errors currently present in the machine** based on measured optics.

Missing or noisy measurements

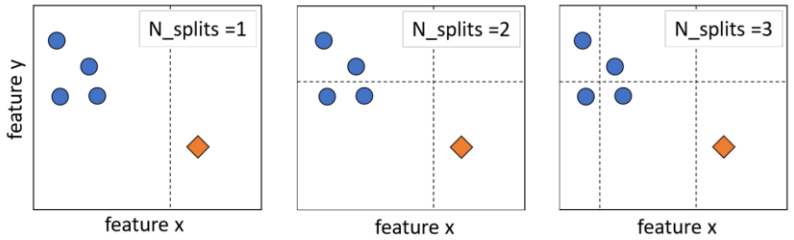
- In case of BPMs failures the signal and the optics function computation at the location is missing.

→ **Denoising and reconstruction** of optics measurements using Neural Networks.

I. Detection of faulty Beam Position Monitors

Presence of remaining faulty signal can be observed only in the last analysis step – optics reconstructed from BPM signal: manual cleaning and repeating optics computation are required.

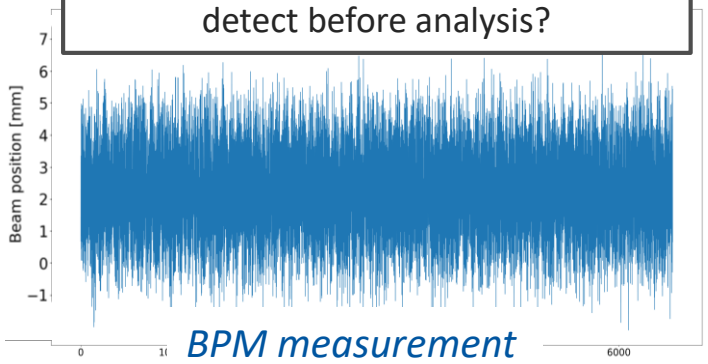
→ **Unsupervised Learning using Isolation Forest (Ensemble of Decision Trees)**



- **Random splits aiming to "isolate" each point.**
- The less splits are needed, the more "anomalous".
- Expected proportion of outliers is a parameter of the algorithm.

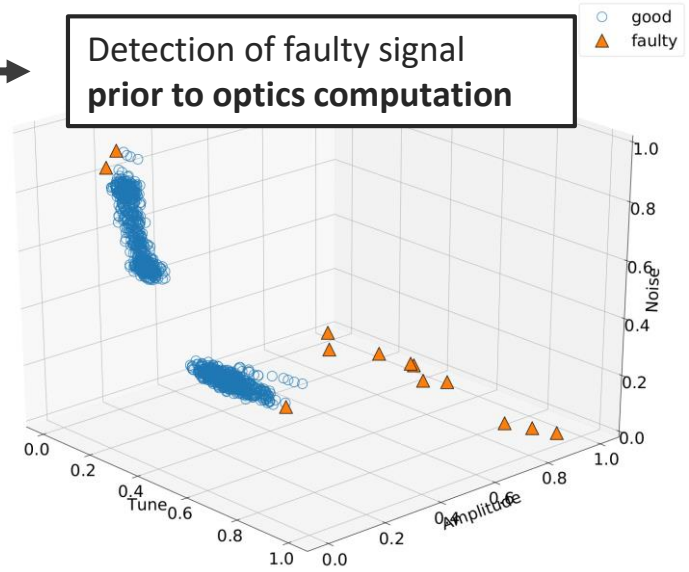
→ ***Ability to identify anomalies without predefined thresholds or rules.***

Causes unphysical values in reconstructed optics, but how to detect before analysis?



BPM measurement

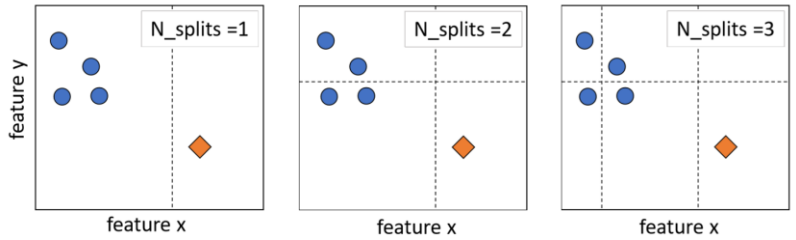
Detection of faulty signal **prior to optics computation**



I. Detection of faulty Beam Position Monitors

Presence of remaining faulty signal can be observed only in the last analysis step – optics reconstructed from BPM signal: manual cleaning and repeating optics computation are required.

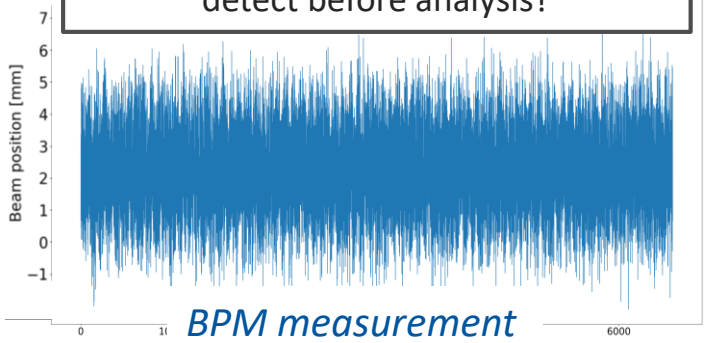
→ **Unsupervised Learning using Isolation Forest (Ensemble of Decision Trees)**



- **Random splits aiming to "isolate" each point.**
- The less splits are needed, the more "anomalous".
- Expected proportion of outliers is a parameter of the algorithm.

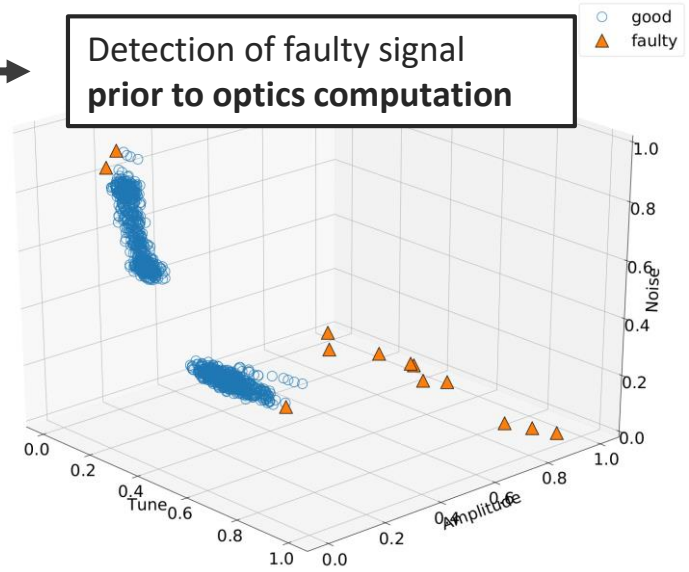
→ ***Ability to identify anomalies without predefined thresholds or rules.***

Causes unphysical values in reconstructed optics, but how to detect before analysis?

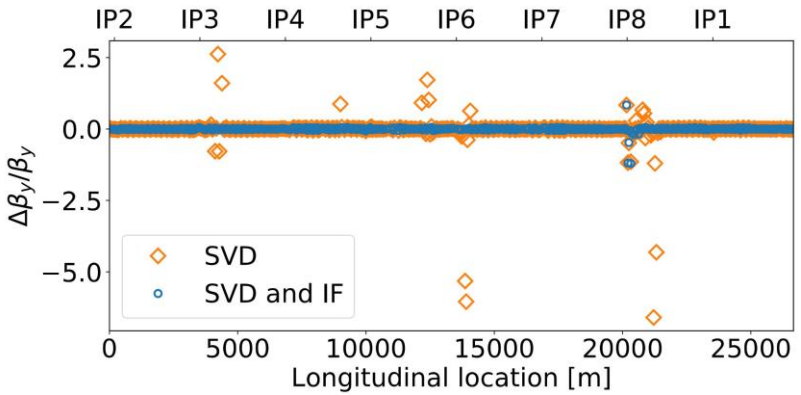


BPM measurement

Detection of faulty signal **prior to optics computation**



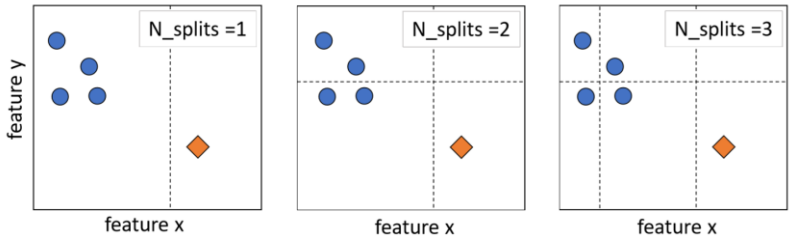
Avoid the appearance of erroneous optics computation



I. Detection of faulty Beam Position Monitors

Presence of remaining faulty signal can be observed only in the last analysis step – optics reconstructed from BPM signal: manual cleaning and repeating optics computation are required.

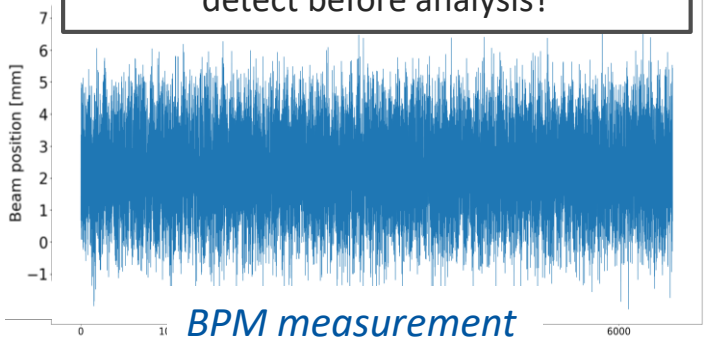
→ **Unsupervised Learning using Isolation Forest (Ensemble of Decision Trees)**



- **Random splits aiming to "isolate" each point.**
- The less splits are needed, the more "anomalous".
- Expected proportion of outliers is a parameter of the algorithm.

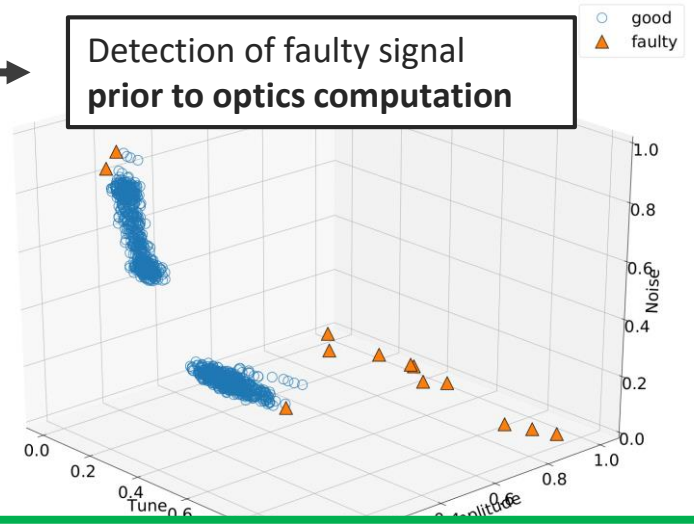
→ ***Ability to identify anomalies without predefined thresholds or rules.***

Causes unphysical values in reconstructed optics, but how to detect before analysis?

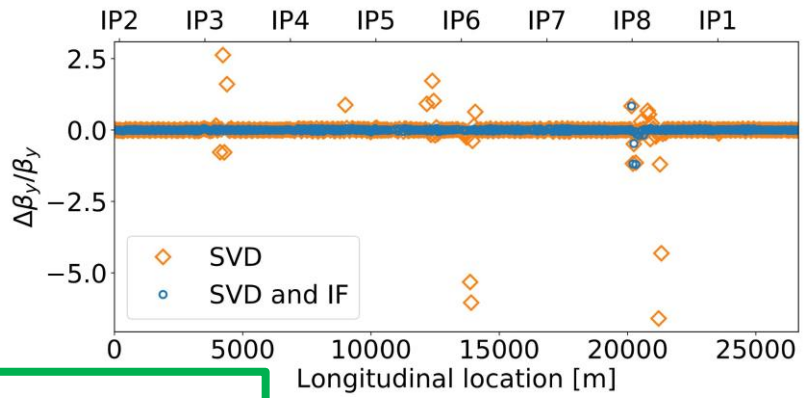


BPM measurement

Detection of faulty signal **prior to optics computation**



Avoid the appearance of erroneous optics computation

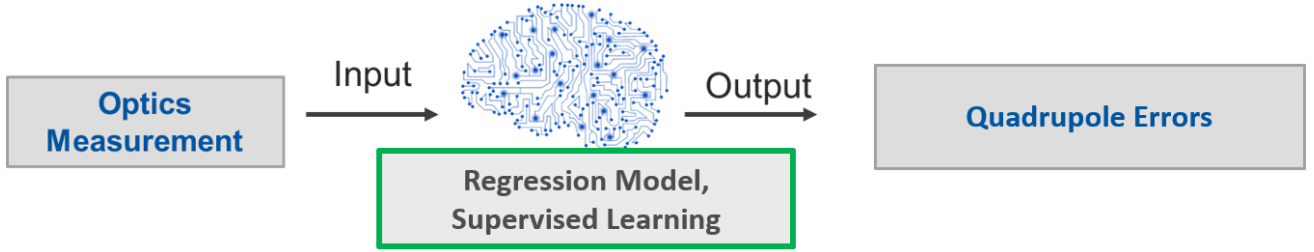


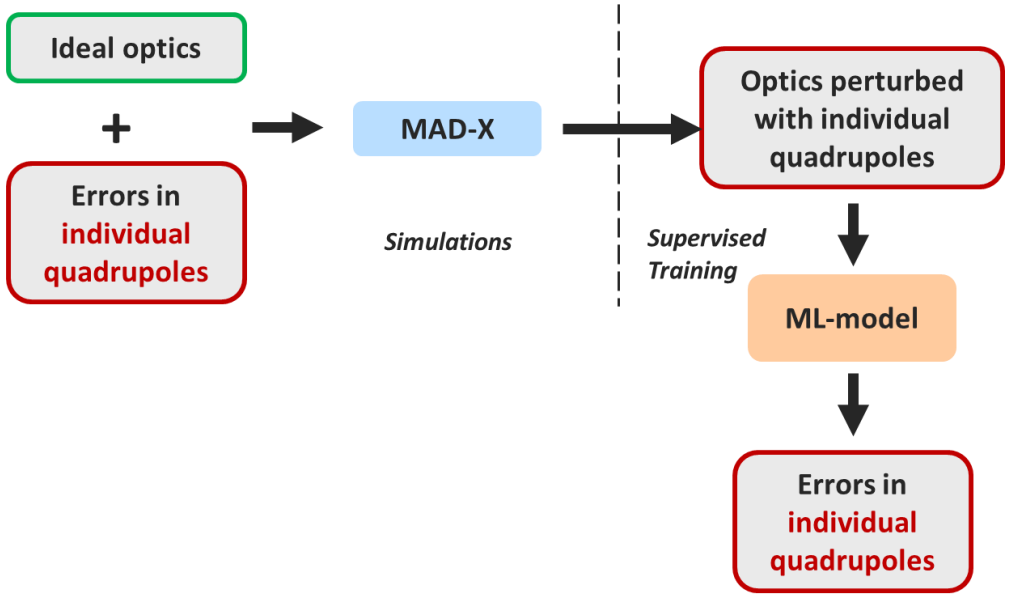
- ✓ Fully integrated into optics measurements at LHC
- ✓ Successfully used in operation under different optics settings.

II. Estimation of quadrupole errors

How to get the entire set of currently present magnet errors in one step?

→ Train **supervised regression model** to predict magnet errors from optics perturbations caused by these errors.
→ Large dataset is needed in order to train a regression model: simulations!





Training ML- regression model:

- **Input:** simulated optics functions at different BPM locations (adding **realistic noise** estimated from the measurements).
- **Output:** randomly generated magnet errors which produce the perturbation in the simulated optics used as model input
- Using **Linear Regression as baseline model** (fast to train and easy to interpret).

II. Estimation of quadrupole errors: validation

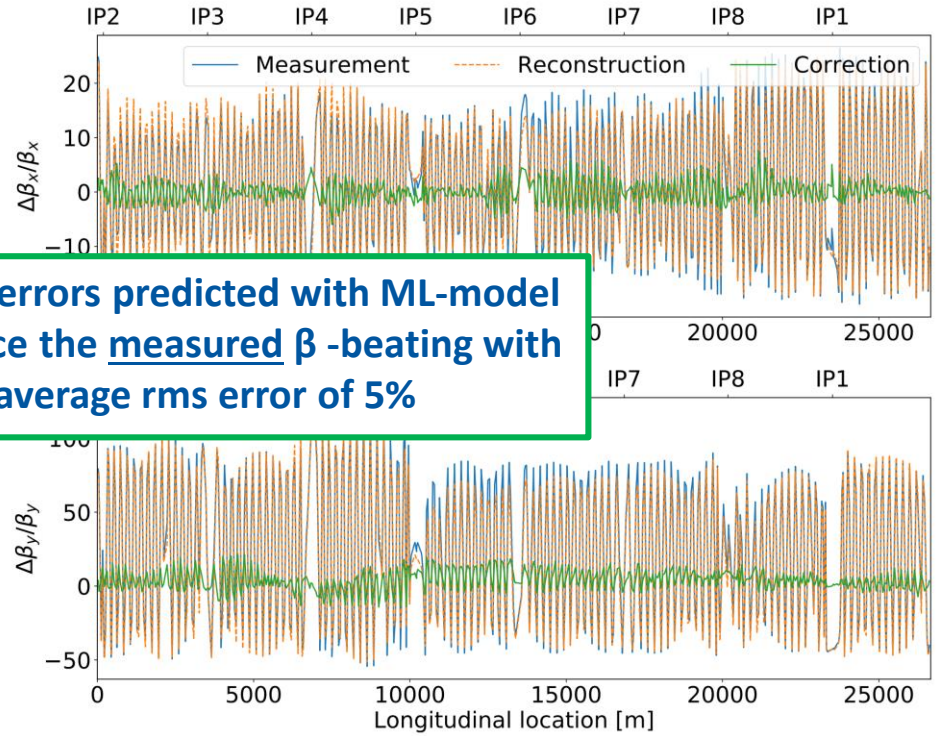
*Prediction of **simulated** individual magnet errors.*

- 75 000 samples from **simulations** (80% training, 20% test)
- **Test data**: Comparing true simulated magnet errors and model output.
- Systematic error of prediction **16%,** random error ~30%

Supervised Learning allows to determine realistic quadrupole errors directly from measured optics.

Test on LHC optics measurements, uncorrected machine

True magnet errors are unknown:
→ **Simulate optics** perturbation **with predicted errors**
→ **Compare to measurement** used as input.

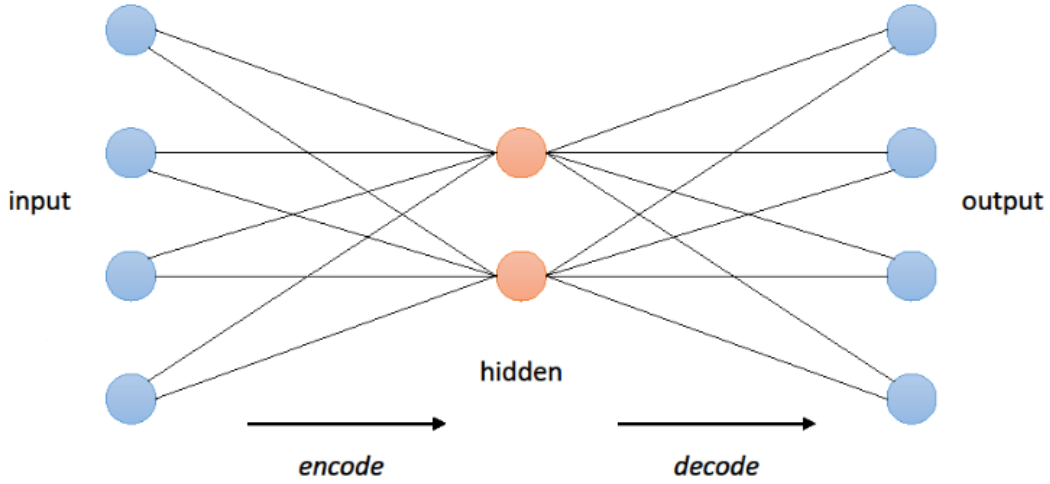


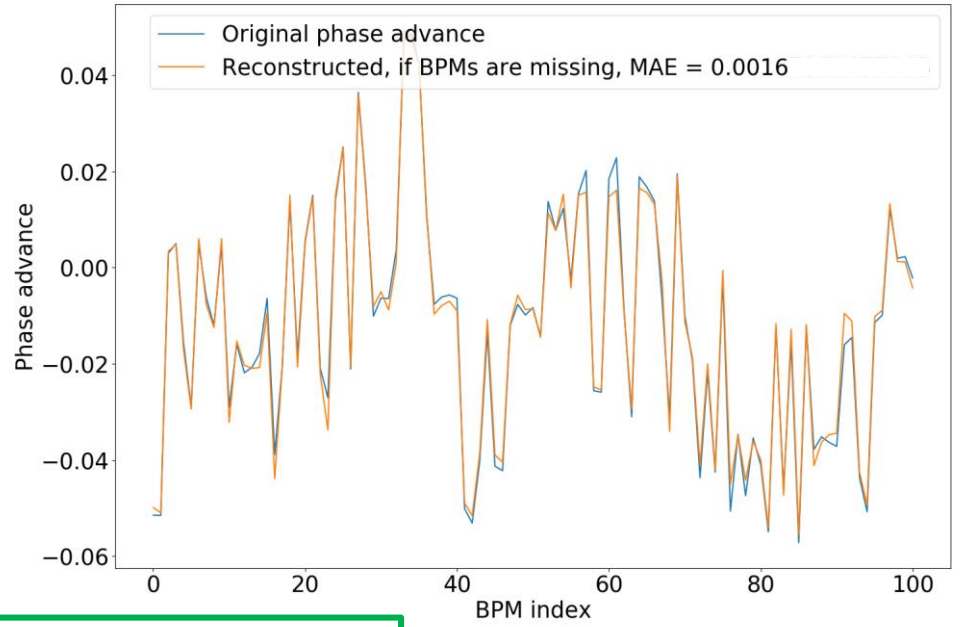
Magnet errors predicted with ML-model reproduce the measured β -beating with average rms error of 5%

III. Denoising and Reconstruction of phase measurements

- Phase advance between the BPMs is the main signal property for the optics computation.
- Quality of magnet errors prediction depends on **noise in the input data** and the number of available BPMs:

→ **Autoencoder Neural Network to denoise and reconstruct phase measurements.**





First results on simulations:
→ Phase advance measurements at simulated missing BPMs reconstructed with 1% accuracy.
→ Noise is reduced by a factor of 2.

Potentially useful, but not (widely) used yet

Some ideas…

- **Transfer Learning**

  - Train model on one problem domain, apply on another task after short re-training,
  - Data set required for re-training can be much smaller than data set used in initial training,
  - Real-time application (e.g. re-training using data recorded in operation),
  possibility to take advantage from previous efforts.

- **Inverted Models**

  - Train to predict a set of output targets from a set of input parameters.

  - Invert the model and use the learned correlation to predict the “input” parameters from targets,

  e.g. to predict settings from desired beam properties.

- **Text processing**

  - Logbooks contain a lot of unstructured information, which can be relevant to build automation/control ML tools.

  - Extract relevant information automatically by analyzing text entries using e.g. Ontology Learning.

  - Use extracted information to build models for machine components failures prediction, to automatize operation, etc.

Practical advice

- **Feature engineering** is highly important! Rescaling, denoising, outlier elimination...
  - ➢ data vizualisation can help
- Start with **simple models** (increase the model complexity (e.g. applying Neural Networks) only if really needed.
- Well **structured** data, **extendable architecture** of existing frameworks
  → possibility for the integration of ML tools.
- Estimate model generalization (split into training, test and validation sets)

Frameworks to use:

- Prototyping, fast and easy implementation (very good documentation): http://scikit-learn.org/
- High-level package for Neural Networks: – https://keras.io/
- Deep Learning, specific complex model architectures: https://www.tensorflow.org/ http://deeplearning.net/software/theano/
- Reinforcement Learning: OpenAI Gym https://gym.openai.com/

Summary: Where can we use ML in accelerators?

Detection of instrumentation failures

Beam control and lattice imperfection corrections

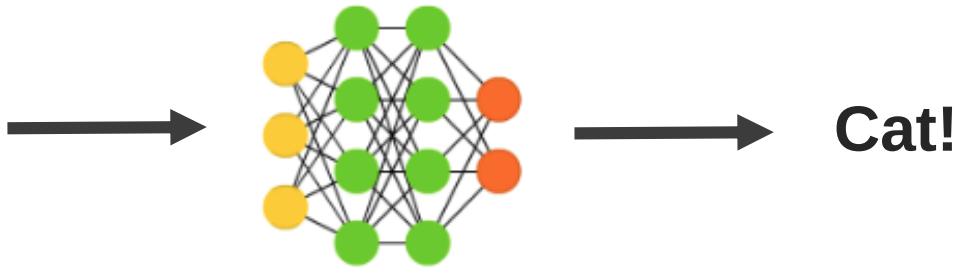
Optimization and operation automation

Virtual Diagnostics

- Defining a **narrow task** (optimization of specific parameters rather than the entire machine)
- **Performance measure** of selected model (beam size, pulse energy, …)
- e.g. when no analytical solution is available, rapidly changing systems, no direct measurements are possible.

Important to identify where ML can surpass traditional methods

How much effort is needed to implement a ML solution? Is appropriate infrastructure for data acquisition available? Enough resources to perform the training?



Thank you for your attention!

ML in accelerators: summary

Accelerator Problem	ML methods	Benefits	To be considered
• Automation of particular components	Supervised techniques for classification: Decision Trees, SVR, Logistic Regression, NN	Saving operation time, reducing human intervention, preventing subjective decisions	Dedicated machine time usually required to collect training data and to fine tune developed methods.
• Online optimization of several targets which are coupled • Unexpected drifts, continuous settings readjustment needed to maintain beam quality	Reinforcement Learning, Bayesian optimization, Gaussian Process, Adaptive Feedback	Simultaneous optimization targeting several beam properties, automatically finding trade-off between optimization targets, allows faster tuning offering more user time.	Ensuring that all important properties are included as optimization targets.
• Detection of anomalies	Unsupervised methods: clustering, ensembles of decision trees (e.g. Isolation Forest), supervised classification, Recurrent NN for time-series data.	Preventing faults before they appear, no need to define rules/ thresholds, no training is needed and can be directly applied on received data	In unsupervised methods, usually no “ground truth” is available → methods can be verified on simulations.

ML in accelerators: summary

Accelerator Problem	ML methods	Benefits	To be considered
• Computationally heavy, slow simulations • Reconstruct unknown properties from measurements	Supervised Regression models, NN for non-linear problems	Learning underlying physics directly from the data, faster execution	100% realistic simulations are not possible → the model performance will be as good as your data is.
• Reduction of parameter space e.g. for optimization	Clustering, Feature Importance Analysis using Decision trees	Speed up of available methods, simpler defined problems, easier to interpret	Parameter selection and combination (feature engineering) can have significant impact on ML methods performance
• Missing or too noisy data	Autoencoder NN	Robust models, data quality	Significant information should not be removed from the signal.

Regression Models

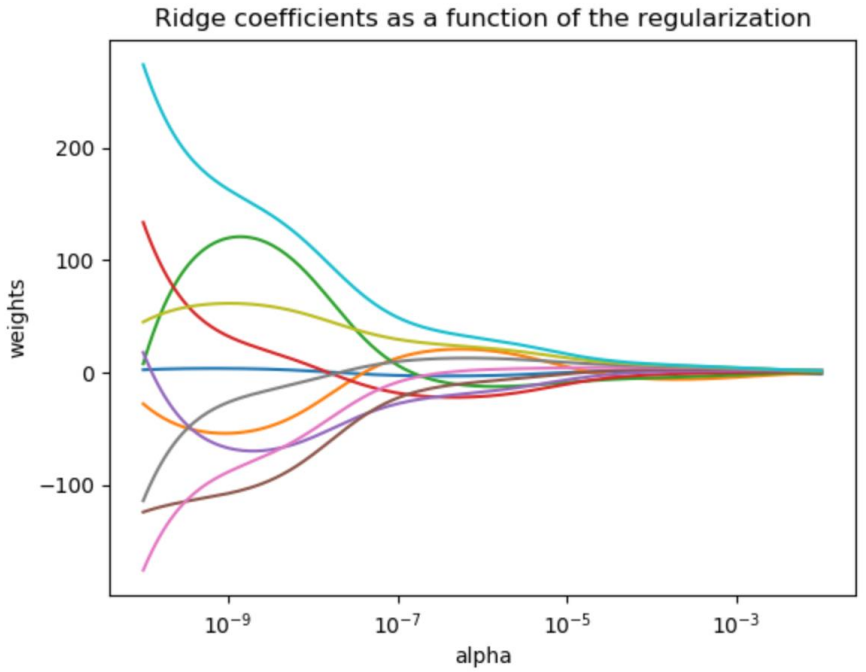
- Linear model for *input X / output Y pairs*, *i* – number of pairs (training samples): $\boldsymbol{f(X, w) = w^T X}$
- Squared Loss function for model optimization: $\boldsymbol{L(w) = \frac{1}{2}\sum_i \left(Y_i - f(X_i; w)\right)^2}$
- Find new weights minimizing the Loss function: $\boldsymbol{w^* = \arg min_w L(w)}$

→ **Update weights for each incoming input/output pair.**

Too much "flexibility" in weights update can lead to *overfitting*

→ **Regularization** places constraints on the model parameters (weights)

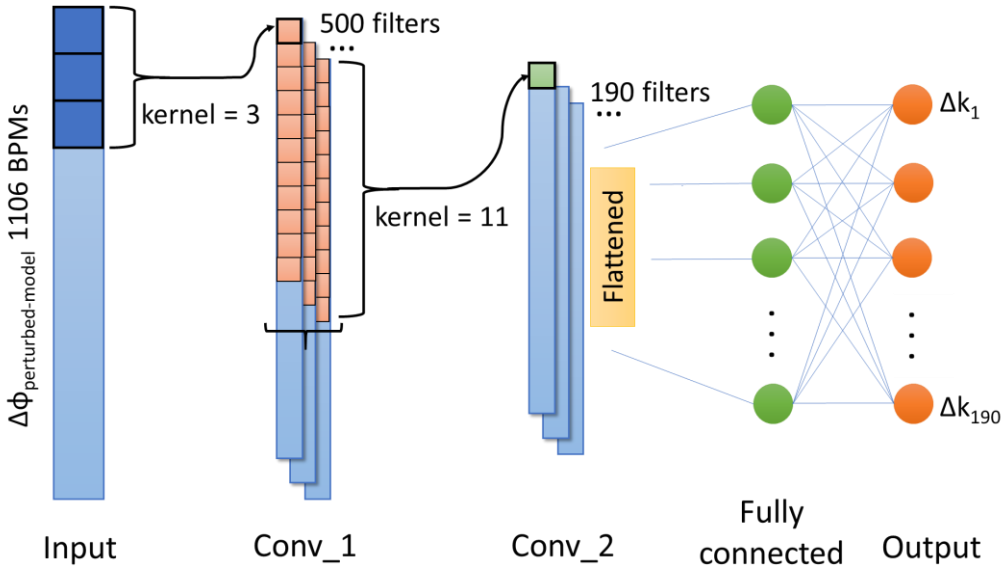
- Trading some bias to reduce model variance.
- Using L2-norm: $\boldsymbol{\Omega(w) = \sum_i w_i^2}$, adding the constraint $\boldsymbol{\alpha\Omega(w)}$ to the weights update rule
- The larger the value of $\boldsymbol{\alpha}$, the stronger the shrinkage and thus the coefficients become more robust.



Convolutional Neural Network

Convolutions: shared weights of neurons, but each neuron only takes subset of inputs.

- Used for image processing
- Looks for **spatially depended** features → optics is concerned by phase advance between **neighboring** BPMs
- Different deep layers look for different features
- Keras with TensorFlow backend



The composition of CNN implemented to predict 190 correctors values to correct the optics perturbed by individual quadrupole errors.