Machine learning for accelerators
A physicist approach

Pierre Schnizer

Helmholtz-Zentrum Berlin (HZB), Germany

ICALEPCS 2021: 2nd Data Science and Machine Learning Workshop 15th of October 2021

Contents

Machine learning

- approaches
  - learn to reproduce: "supervised learning" (e.g. regression)
  - find structure: "unsupervised learning" (e.g. isolation forests)
  - learn environment: "reinforcement learning" (e.g. AlphaGo)
- concepts
  - Random variables, statistic distribution
  - prior, posterior, Bayes rule
  - additive models, trees networks
  - Bellman's equation

We are drowning in information and starving for knowledge

– Rutherford D. Roger

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

– Elena Fol, IPAC'20

Machine learning

- approaches
  - learn to reproduce: "supervised learning" (e.g. regression)
  - find structure: "unsupervised learning" (e.g. isolation forests)
  - learn environment: "reinforcement learning" (e.g. AlphaGo)
- concepts
  - Random variables, statistic distribution
  - prior, posterior, Bayes rule
  - additive models, trees networks
  - Bellman's equation

„Fliegen tut auch ein Scheunentor, wenn der Motor nur gut ist."[1]

Simon Brunnhuber

"With a big enough engine you can make a barn door fly"

Regularisation

- slow orbit feedback: Thikonov regularisation, finding orbit distortions, sklearn

Example

Orbit feedback

- Read orbit deviation
- Calculate correction → on diagonal → ridge

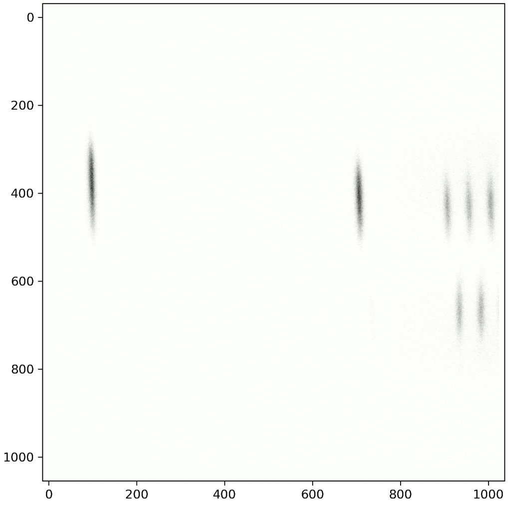
```
# bpm signals
b_A = np.array([x_bpm, y_bpm]).ravel()

# orbit response matrix
corr_mat = self.orb.get()
alpha = self.alpha.get()

# sci kit learn: regularisaton of the central diagonal
self.ridge = Ridge(alpha=alpha)
self.ridge.fit(corr_mat, b_A)
pos_corr = self.ridge.coef_
```

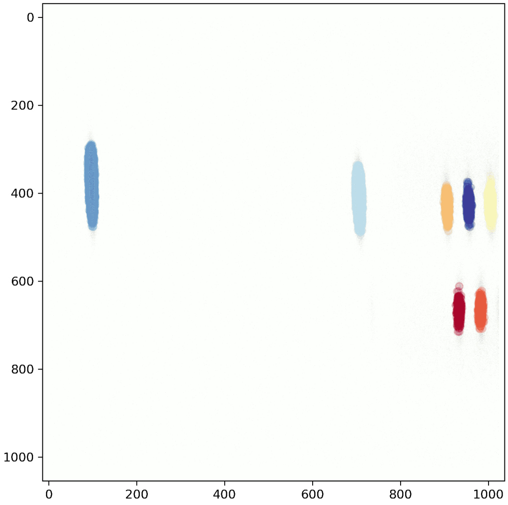
- Streak camera measurement
- Identify spots

```
x, y = np.fromfunction(lambda y,x: (x,y), im.shape)
val = im.ravel()
# Points above threshold
xr = x.ravel()[val>threshold]
yr = y.ravel()[val>threshold]
X = np.array([xr, yr]).T
# Make spots rather round ...
X_trans = X * np.array([1, .2])[np.newaxis, :]
y_pred = KMeans(n_clusters=7, random_state=0)
          .fit_predict(X_trans)
```



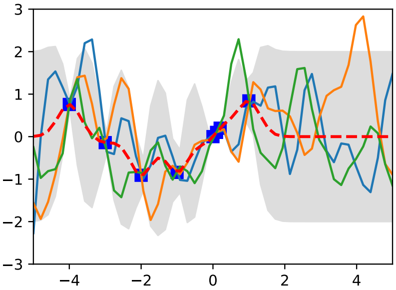
- Streak camera measurement
- Identify spots

```
x, y = np.fromfunction(lambda y,x: (x,y), im.shape)
val = im.ravel()
# Points above threshold
xr = x.ravel()[val>threshold]
yr = y.ravel()[val>threshold]
X = np.array([xr, yr]).T
# Make spots rather round ...
X_trans = X * np.array([1, .2])[np.newaxis, :]
y_pred = KMeans(n_clusters=7, random_state=0)
          .fit_predict(X_trans)
```



Gaussian process

Power of a simple idea

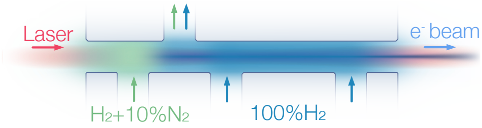


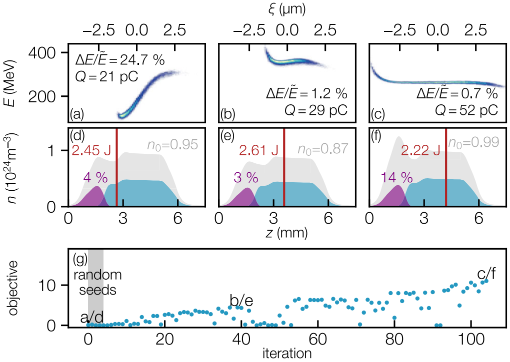
- Linear regression → minimum distance
- Bayesian approach: most probable line
- Idea taken further:
  - Set of probable functions
  - covariance between functions
  - characteristic length

  }

  - data points → most probable function
  - covariance → confidence bound
  - further probing: most probable one (add noise for exploration)

Gaussian process: laser plasma optimisation





Focus position: *target length 5 mm / 0.05 mm = 100x*

Doping: *0 to 100 % / 1 % = 100x*

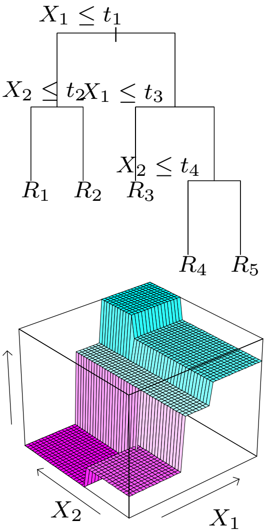
Gas density: *0.5x10^{18} to 1x10^{18} cm^{-3}/0.05x10^{18} cm^{-3}= 10x*

Laser energy: *2 J to 3 J / 0.1 J= 10x*

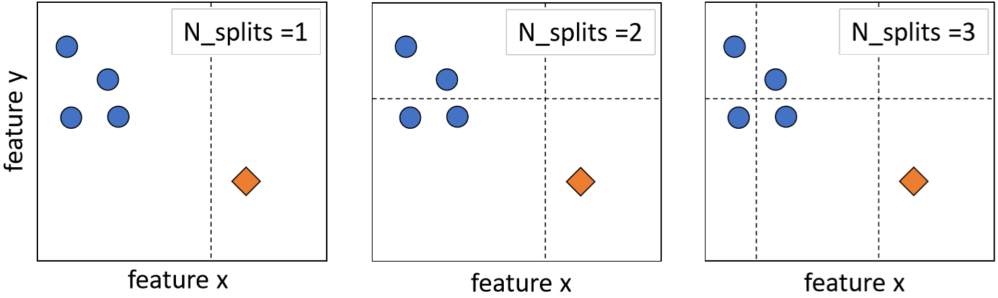
Sören Jalas et al., "Bayesian Optimization of a Laser-Plasma Accelerator" [1]

Trees

- Splitting fields in "common areas"
- Learning: when to split using which variable
- Boosting: chain trees: have next one predict residuals from the previous ones
- Random forests: tress split randomly, combine results → outliers Isolation forests



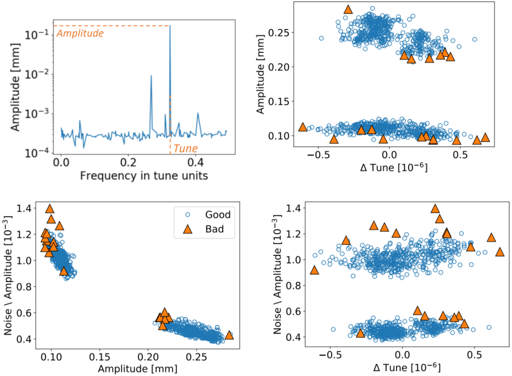
Example: BPM cleaning at LHC



- Target: identify "broken" BPMs
- Forest: split up
- Isolation forests: randomly split many times → outliers split of early

See E. Fol et al. "Detection of faulty beam position monitors using unsupervised learning", [2] [3]

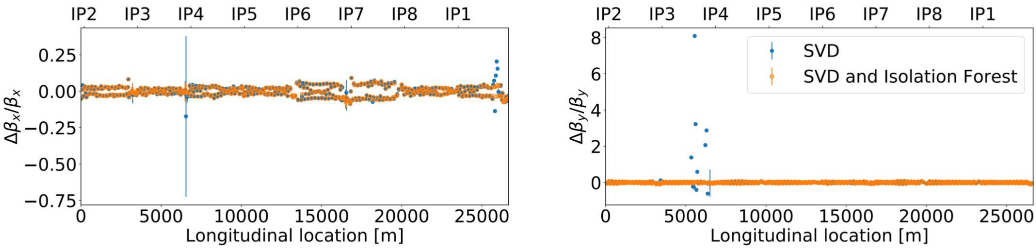
Example: BPM cleaning at LHC



- Target: identify "broken" BPMs
- Forest: split up
- Isolation forests: randomly split many times → outliers split of early

See E. Fol et al. "Detection of faulty beam position monitors using unsupervised learning", [2] [3]

Example: BPM cleaning at LHC



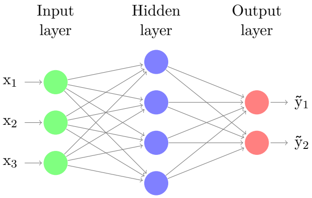
- Target: identify “broken” BPMs
- Forest: split up
- Isolation forests: randomly split many times → outliers split of early

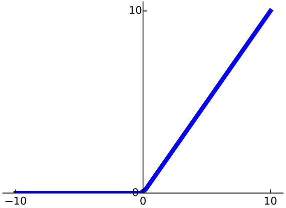
See E. Fol et al. “Detection of faulty beam position monitors using unsupervised learning”, [2] [3]

Network I/II

- Simple basis functions with weights
- network of functions → complex task
- inference (prediction) → forward pass
- learning (“fit network to data”) → backward pass: networks with gradient
- convolution: learn to combine data
- long short term memory: learn from history, but also when to ignore history!

- Layers of neurons: processing units
- activation functions: select range, weight
- build automatically gradient
- flexible, large set of parameters
- trained on large datasets → processed in batches





ReLU

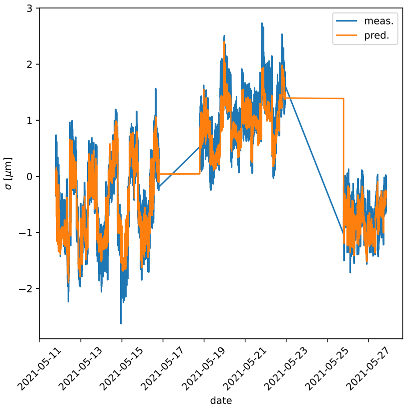
LSTM for beam size

- Light source: beam size → tight control, beam size influenced by emittance exchange horizontal → vertical ALS [4]:
- target: insertion device change → feed forward
- data showed: effect delayed → LSTM

```
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dropout, Dense
from tensorflow.keras.optimizers import Adam

inputs = Input(shape=(60, 152))
x = LSTM(128, return_sequences=True)(inputs)
x = Dropout(0.1)(x)
x = LSTM(64, return_sequences=True)(x)
x = Dropout(0.1)(x)
x = LSTM(32, return_sequences=False)(x)
x = Dropout(0.1)
outputs = Dense(1)

model = Model(inputs=inputs, outputs=outputs, name='vertical_beam_size')
```



Courtesy: Alexander Schütt

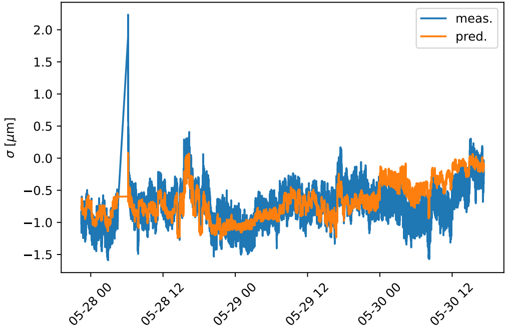
LSTM for beam size

- Light source: beam size → tight control, beam size influenced by emittance exchange horizontal → vertical ALS [4]:
- target: insertion device change → feed forward
- data showed: effect delayed → LSTM

```
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dropout, Dense
from tensorflow.keras.optimizers import Adam

inputs = Input(shape=(60, 152))
x = LSTM(128, return_sequences=True)(inputs)
x = Dropout(0.1)(x)
x = LSTM(64, return_sequences=True)(x)
x = Dropout(0.1)(x)
x = LSTM(32, return_sequences=False)(x)
x = Dropout(0.1)
outputs = Dense(1)

model = Model(inputs=inputs, outputs=outputs, name='vertical_beam_size')
```



Courtesy: Alexander Schütt

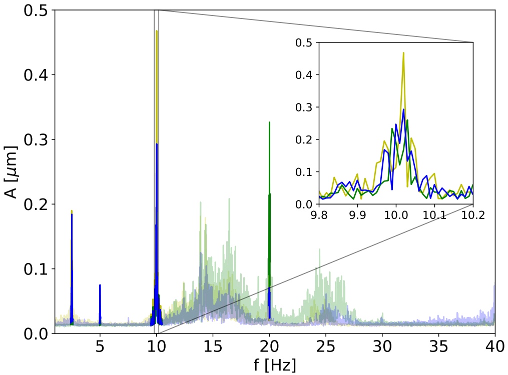
Reinforcement learning: motivation

- Markov decision process: current state → predict future (e.g. chess game)
  - actor (controller) → applying actions (control signal)
  - environment (plant) → state
  - rewards
- Learning
  - actor: policy → how to play (e.g. chess: next move)
  - value function: long term reward for actions (e.g. chess: move more likely to win)
- learning: long term outcome, estimate variation
- solution found: Bellman's equation → sufficient each step optimal
- control learned "playing games": good nerves when operating on beam → explainable AI

Orbit optimisation

- Harmonic orbit distortion
- no action: yellow
- classical: steerers → orbit response matrix → SVD → correction
- Here: SVD + reinforcement agent

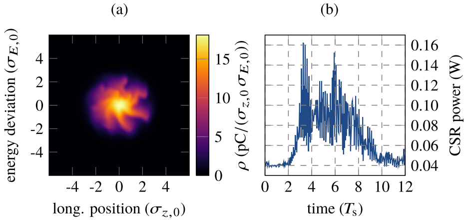
See Luis Vera Ramirez THAL01

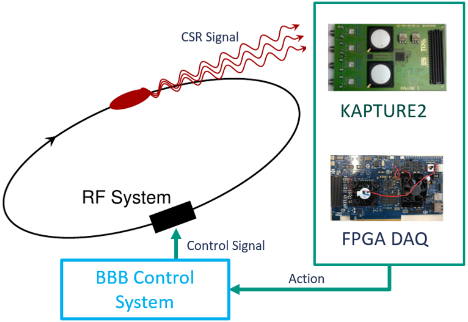


Coherent synchrotron radiation control

Development KIT/KARA

- Synchrotron light sources:
  - deflected electrons → emit light
  - short bunches → coherent light emission → ↑ Power → destabilises bunch
  - mitigation: RF control ← reinforcement learning
  - dedicated hardware → ≈ 10 μs [5]





Explainable machine learning

Accelerators: expensive device → beam experiment access: precious resource: e.g. broken beam position monitor → effect on controller

- Network learns problem → black box
- Explore black box

- find simplified model: e.g. statistical learning one
- train model using original data and trained network
- understand network's behaviour

Methods

- Local interpretable model-agnostic explanations (LIME)[6]: behaviour local area in neighbourhood
- SHAP(ly) values[7] → prediction of cooperative players → influence of different features

Non learning part

Life cycle

- Data access
- Algorithm
- Deployment
- Monitoring

Time

- learning: data sources: consistent time
- interaction: minimize delays
- new installations: steady state machines $\rightarrow$ transient ones

Data quality

- constient access (archivers)
- metadata, calibration
- data cleaning, scaling

Deployment

- accelerators: lifespan $>$ 30 years
- software stack: technical debts $\downarrow$: minimise dependencies e.g. notebooks $\rightarrow$ input output controllers
- versioning, roll back

Conclusion

- (Short) overview over methods and fields
- More details: following talks today, conference
- Let data science experiment $\rightarrow$ reveals new insight
- Domain $\leftrightarrow$ data science: closely work together: exchange
  - knowledge
  - experience
  - best practise

Acknowledgements

- Thanks to Luis for tutoring, introduction materials
- Tom discussion on statistical methods
- Support of BESSY II
- all not even mentioned

For new bees

- Just try it! → first solutions applying examples to your data e.g. scikit-learn
- Networks: published solutions → your data → useable solutions
- well documented libraries: ← “optimisers” self implemented Netwon, Brent?
- contribute: your results, your success stories

Sören Jalas, Manuel Kirchen, Philipp Messner, Paul Winkler, Lars Hübner, Julian Dirkwinkel, Matthias Schnepp, Remi Lehe, and Andreas R. Maier.
Bayesian optimization of a laser-plasma accelerator.
Phys. Rev. Lett., 126:104801, Mar 2021.

E. Fol, G. Franchetti, and R. Tomás.
Machine Learning Techniques for Optics Measurements and Corrections.
In Proc. IPAC'20, number 11 in International Particle Accelerator Conference, pages 61–66. JACoW Publishing, Geneva, Switzerland, 10 2020. https://doi.org/10.18429/JACoW-IPAC2020-WEVIR12.

E. Fol, R. Tomás, J. Coello de Portugal, and G. Franchetti.
Detection of faulty beam position monitors using unsupervised learning.
Phys. Rev. Accel. Beams, 23:102805, Oct 2020.

S. C. Leemann, S. Liu, A. Hexemer, M. A. Marcus, C. N. Melton, H. Nishimura, and C. Sun.
Demonstration of machine learning-based model-independent stabilization of source properties in synchrotron light sources.

Phys. Rev. Lett., 123:194801, Nov 2019.

Weija Wang, Michele Caselle, Tobias Boltz, Edmund Blomley, Miriam Brosi, Timo Dritschler, Andreas Ebersoldt, Andreas Kopmann, Andrea Santamaria Garcia, Patrick Schreiber, Erik Bründermann, Marc Weber, Anke-Susanne Müller, and Yangwang Fang.
Accelerated deep reinforcement learning for fast feedback of beam dynamics at kara.
IEEE Transactions on Nuclear Science, 68(8):1794–1800, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin.
"why should i trust you?": Explaining the predictions of any classifier.
In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

Scott M Lundberg and Su-In Lee.
A unified approach to interpreting model predictions.

In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.