Resolving Profile Distortion for Electron-based IPMs using Machine Learning

3rd IPM Workshop
J-PARC (Tokai, Japan)

D. Vilsmeier (GSI)
M. Sapinski (GSI)
R. Singh (GSI)
What is Machine Learning?

"Field of study that gives computers the ability to learn without being explicitly programmed."

- Arthur Samuel (1959)

"Classical" approach:

\[
\text{Input} + \text{Algorithm} = \text{Output}
\]

Machine Learning:

\[
\text{Input} + \text{Algorithm} = \text{Output}
\]
Machine Learning Toolbox

Supervised Learning:
- Artificial Neural Networks
- Decision Trees
- Linear Regression
- k-Nearest Neighbor
- Support Vector Machines
- Random Forest
- ... and many more

Unsupervised Learning:
- k-Means Clustering
- Autoencoders
- Principal comp. analysis

Reinforcement Learning:
- Q-Learning
- Deep Deterministic Policy Gradient
IPM Profile Distortion

**Ideal case**
Particles move on straight lines towards the detector

**Real case**
Trajectories are influenced by initial momenta and by interaction with beam field
Counteract via ...

**Increase of electric field**
Resulting in smaller extraction times and hence smaller displacements; limit is quickly reached

**Additional magnetic field**
Constrains the maximal displacement to the gyroradius of the resulting motion; usually an effective measure

\[ E_3 > E_2 > E_1 \]
Distortion without magnetic field

- Already observed in [W. DeLuca, IEEE 1969] (+ observation of focusing for electron collection)
- R. E. Thern "Space-charge Distortion in the Brookhaven Ionization Profile Monitor" PAC 1987
  - Simulations + Measurements
  - Good agreement for nominal extraction voltages
  - Disagreement at lower extraction voltages
- W. Graves "Measurement of Transverse Emittance in the Fermilab Booster" PhD 1994
  \[
  \sigma_{\text{beam}} = c_1 + c_2 \sigma_{\text{measured}} + c_3 N
  \]

\[
\sigma_m = \sigma + 0.302 \frac{N^{1.065}}{\sigma^{2.065}} \left( 1 + 3.6 R^{1.54} \right)^{-0.435}
\]

Fig. 4 Comparison of calculated and measured values of ion scatter as a function of electrode field strength.

+ other approaches, including non-Gaussian beam shapes via iterative procedures
Distortion with magnetic field

- More complex motion also due to the interaction with beam electric field
- Capturing effects as well as different electromagnetic drifts play a role
- Displacement from initial position can be mainly ascribed to three different effects:
  - Displacement of gyro-center due to initial velocities ($\Delta x_1$)
  - Displacement of gyro-center due to space-charge interaction ($\Delta x_2$)
  - Displacement due to gyro-motion above detector ($\Delta x_3$)

Final motion is determined by effects in the "space-charge region"
Electron trajectories

- The resulting motion strongly depends on the starting position within the bunch and hence on the bunch shape itself.

- Various electromagnetic drifts / interactions create a complex dependence of the final gyro-motion on the initial conditions.

Electron trajectories

- Polarization drift \( \left( \frac{d\vec{E}}{dt} \right) \)

- ExB-drift

Electron motion

"Pure" gyro-motion

Electron trajectory

Capturing

p-bunch
Gyro-radius increase

- This interaction effectively results in an increase of gyro-radii which consequently determines the profile distortion.

- The increase itself depends on the starting position and thus on the bunch shape. It prevents usage of simple description by other beam parameters (e.g. point-spread functions).
Profile distortion

Ideally a one-dimensional projection the transverse beam profile is measured, but...

- $E = 6.5\text{ TeV}$
- $N_q = 2.1 \times 10^{11}$
- $\sigma_x = 0.27\text{ mm}$
- $\sigma_y = 0.36\text{ mm}$
- $4\sigma_z = 0.9\text{ ns}$
Magnetic field increase

Without space-charge electrons at the bunch center will perform exactly N turns for specific magnetic field strengths.

Due to space-charge interaction only large field strengths are effective though.

N-turn B-fields
Using Machine Learning

1. https://pypi.org/project/virtual-ipm

2. \( N_p \rightarrow \) Physical process \( f \) (Virtual-IPM)
\( \sigma_{x,a} \rightarrow \sigma_{y,a} \rightarrow \sigma_t \)

Approximation of inverse process \( f' \)
\( \sigma_{x,p} \leftarrow \) Measured profile

3. Training
- Used to fit the model; split size ~ 60%.

Validation
- Check generalization to unseen data; split size ~ 20%.

Testing
- Evaluate final model performance; split size ~ 20%.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Step size</th>
<th>Protons</th>
<th>Consider cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunch pop. ([1e11])</td>
<td>1.1 -- 2.1 ppb</td>
<td>0.1 ppb</td>
<td>6.5 TeV</td>
<td>21,021 different cases</td>
</tr>
<tr>
<td>Bunch width (1(\sigma))</td>
<td>270 -- 370 (\mu m)</td>
<td>5 (\mu m)</td>
<td>4kV / 85mm</td>
<td></td>
</tr>
<tr>
<td>Bunch height (1(\sigma))</td>
<td>360 -- 600 (\mu m)</td>
<td>20 (\mu m)</td>
<td>0.2 T</td>
<td></td>
</tr>
<tr>
<td>Bunch length (4(\sigma))</td>
<td>0.9 -- 1.2 ns</td>
<td>0.05 ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Evaluated on grid data and randomly sampled data
Artificial Neural Networks

Inspired by the human brain, many "neurons" linked together.

Map non-linearities through non-linear activation functions.

**Perceptron**

\[ y(x) = \sigma (W \cdot x + b) \]

**Multi-Layer Perceptron**

- Apply non-linearity, e.g. ReLU, Tanh, Sigmoid.
**ANN Implementation**

```python
IDense = partial(Dense, kernel_initializer=VarianceScaling())

# Create feed-forward network.
model = Sequential()

# Since this is the first hidden layer we also need to specify
# the shape of the input data (49 predictors).
model.add(IDense(200, activation='relu', input_shape=(49,)))
model.add(IDense(170, activation='relu'))
model.add(IDense(140, activation='relu'))
model.add(IDense(110, activation='relu'))

# The network's output (beam sigma). This uses linear activation
model.add(IDense(1))

model.compile(
    optimizer=Adam(lr=0.001),
    loss='mean_squared_error'
)

model.fit(
    x_train, y_train,
    batch_size=8, epochs=100, shuffle=True,
    validation_data=(x_val, y_val)
)
```

Batch learning
- Iterate through training set multiple times (= epochs)
- Weight updates are performed in batches (of training samples)

Fully-connected feed-forward network with ReLU activation function


After each epoch compute loss on validation data in order to prevent "overfitting"
**Why ANNs?**

**Universal approximation theorem**

"Every finite continuous "target" function can be approximated with arbitrarily small error by feed-forward network with single hidden layer"

[corresponding Cybenko 1989; Hornik 1991]

\[ y = \sum_{j=1}^{n} w_j^o \cdot \sigma \left( \sum_{k=1}^{d} w_{jk}^h x_k + b_j \right) \]

- \( n \) hidden units
- activation function
- \( d \) - dimensional domain
- parameters to be "optimized"

Works on compact subsets of \( \mathbb{R}^d \)

Proof of existence, i.e. no universal optimization algorithm exists □ "No free lunch theorem"
Profile RMS Inference - Results

Very good results on simulation data below 1% accuracy

Results are without consideration of noise on profile data

Tested also other machine learning algorithms:
- Linear regression (LR)
- Kernel ridge regression (KRR)
- Support vector machine (SVR)

Multi-layer perceptron (= ANN)

Table 2: Resulting Scores for the Different Models. Values are given in units of 1 μm, 1 μm² respectively.

<table>
<thead>
<tr>
<th></th>
<th>$\mu(\text{res})$</th>
<th>$\sigma(\text{res})$</th>
<th>R2</th>
<th>EV</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.012</td>
<td>0.449</td>
<td>0.99976</td>
<td>0.99976</td>
<td>0.201</td>
</tr>
<tr>
<td>KRR</td>
<td>0.005</td>
<td>0.340</td>
<td>0.99986</td>
<td>0.99986</td>
<td>0.115</td>
</tr>
<tr>
<td>SVR</td>
<td>0.006</td>
<td>0.349</td>
<td>0.99985</td>
<td>0.99985</td>
<td>0.121</td>
</tr>
<tr>
<td>MLP</td>
<td>0.232</td>
<td>0.370</td>
<td>0.99977</td>
<td>0.99984</td>
<td>0.190</td>
</tr>
</tbody>
</table>
RMS Inference with Noise

Linear regression model

Linear regression amplifies noise in predictions if not explicitly trained

Multi-layer perceptron

MLP amplifies noise; bounded activation functions could help; as well as duplicating data before "noising"
Full Profile Reconstruction

So far:

Instead:

Machine Learning Model

Compute beam RMS

$\sigma_x$

Compute beam profile

18/09/2018
3rd IPM Workshop, D. Vilsmeier
Gaussian bunch shape

MLP Architecture

- 2 hidden layers, 88 nodes
- tanh activation function

Performance measure

- Mean squared error (MSE)

\[
\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_{p,i} - y_{i})^2
\]

\[
\begin{align*}
\text{mean} &= 0.0024 \\
\text{std} &= 0.0045
\end{align*}
\]

\[
\begin{align*}
\text{mean} &= 0.1231 \\
\text{std} &= 0.0808
\end{align*}
\]
Generalized Gaussian bunch shape

\[
\frac{\beta}{2\alpha \Gamma(1/\beta)} e^{-\left(\frac{|x-\mu|}{\alpha}\right)^\beta}
\]

Gen-Gauss used for testing while training (fitting) was performed with Gaussian bunch shape

\[\beta = 3\]

\[\beta = 1.5\]

Smaller distortion in this case

ANN model generalizes to different beam shapes
Q-Gaussian bunch shape

$$\frac{\sqrt{\beta}}{C_q} \left[ 1 - (1 - q) \beta x^2 \right] \frac{1}{1-q}$$

Q-Gauss used for testing while training (fitting) was performed with Gaussian bunch shape

ANN model generalizes to different beam shapes

No distortion for this case

$q = 0.6$

$q = 2.0$
Model prediction uncertainty

- Could train multiple models with different initialization and different data presentation. Ensembling predictions
- Emulate multi-model ensemble by using "Dropout" layers (also for predictions)

MLP shows very small standard deviation in predictions. The fitting converged well, small model uncertainty.
Sub-resolution measurements

- Understanding or (machine) "learning" beam profile deformation (and how to revert it), this information could be used to measure beams that are smaller than the resolution of the detector (by provoking a deformation / blow-up, then reverting it)
- Example: SwissXFEL, 5.8 GeV electrons, 230 pC bunch charge, 21 fs bunch length, 5-7 μm transverse size
- Bunch size is 1/10-th of detector resolution however the deformed profile is well above and strongly depends on the bunch size
- Alternative to R. Tarkeshian et al. Phys. Rev. X 8, 021039 (reconstruction based on ion energies)
Summary

- Successful beam RMS reconstruction with various machine learning models
- Reconstruction of complete profiles with multi-layer perceptron model
  - The mapping generalizes to different beam shapes
  - Model seems to "learn" the distortion mechanisms rather than specific beam shapes
- Model uncertainty estimates show small variations
- These methods could potentially be used to measure sub-resolution beam profiles