RECONSTRUCTING SPACE-CHARGE DISTORTED IPM PROFILES WITH MACHINE LEARNING ALGORITHMS

D. Vilsmeier, M. Sapinski, R. Singh, GSI, Darmstadt, Germany
J. W. Storey, CERN, Geneva, Switzerland

Abstract

Measurements of undistorted transverse profiles via Ionization Profile Monitors (IPMs) may pose a great challenge for high brightness or high energy beams due to interaction of ionized electrons or ions with the electromagnetic field of the beam. This contribution presents application of various machine learning algorithms to the problem of inferring the actual beam profile width from measured profiles that are distorted by beam space-charge interaction. (Generalized) linear regression, artificial neural network and support vector machine algorithms are trained with simulation data, obtained from the Virtual-IPM simulation tool, in order to learn the relation between distorted profiles and original beam dimension. The performance of different algorithms is assessed and the obtained results are very promising with simulation data.

INTRODUCTION

Ionization Profile Monitors (IPM) are used for non-destructive measurements of the transverse beam profile. They make use of residual gas ionization by the particle beam and collect the ionization products via appropriate guiding fields. The distribution of those ions and electrons follows the charge distribution of the beam and the IPM measures a corresponding one-dimensional projection. In conditions when fast measurements are required, usually electrons are measured due to their shorter time-of-flights. An additional magnetic guiding field, collinear with the electric field, may be applied to confine the electron trajectories. In cases where the magnetic field is not sufficient, electrons might be registered at different locations than they were generated, leading to a deformation of the measured profiles with respect to the beam profile. While several attempts have been made to correct such effects for cases without magnetic guiding field, only a few approaches exist for cases with magnetic guiding field [1–4].

IPM PROFILE DISTORTION

In IPM, the deformation of measured beam profile occurs as a result of effects related to ionization, transport of electrons to the detector or detector-related effects such as non-uniform response of Multi-Channel Plate (MCP). Here we focus on correcting distortions which arise from ionization and transport-related effects. These include mainly two aspects; *ionization momenta*: the ionized electrons may have a velocity component which points in the direction along the measured profile. That means while electrons travel towards the acquisition system, they will also experience a displacement along the profile; *space-charge interaction*: the interaction of electrons with the electric field of the particle beam may significantly alter their trajectories, resulting in noticeable distortions.

The magnetic field, collinear with the electric field, typically has strength in range between 50 mT to 200 mT. For 50 mT the gyro-frequency is 8.79 GHz and the resulting gyro-radii are around 100 µm while time-of-flights are around 3 ns. Hence, electrons perform a few tens of gyrations before reaching the detector. However the space-charge interaction can become so strong, with fields of up to a few MV m^{-1} [2] (typical extraction fields are around 50 kV m^{-1}), that electrons are (a) trapped in the space-charge region since the beam electric field outweighs the electric guiding field, and (b) forced on significantly different trajectories, resulting in a vast increase of gyro-radii and hence a distortion of the measured profile.

USING MACHINE LEARNING

Domain knowledge describing IPM profile distortion under these conditions is available [1–4], but the problem is complex such that it is impractical to handcraft algorithms that allow for the correction of distorted profiles. A principal characteristic of Machine Learning (ML) is to implicitly deduce a set of rules from given data, mapping specific input to output, relieving the user from this tedious task.

Relevant for the presented problem are machine learning regression models which predict continuous variables (the beam profile or its second moment) from the given data (measured distorted profile, bunch length and intensity). In general a ML model represents an algorithm f, a mapping from input x to output y_p called *decision function*, which is specified by a set of parameters θ_i. While the structure of such an algorithm is fixed (e.g. the number of parameters $|\{\theta_i\}|$), the goal is to tune θ_i so that the corresponding function $f(x \mid \{\theta_i\})$ describes the output best. The quality of this description is typically assessed by a so called *loss function* $\mathscr{L} : (x, y, y_p \mid \{\theta_i\}) \mapsto \mathbb{R}^+$ where y is the expected output for input x. Minimizing this loss function yields optimal values for the predictor parameters θ_i.

The first application of machine learning to the presented problem was limited to artificial neural networks [5] while here we compare a whole range of algorithms with increasing level of complexity as discussed in the next section.

Linear Regression and Ridge Regression

Linear Regression (LR) uses a linear approximation for modelling the relationship between the scalar dependent variable y_p and one or more explanatory variables x. This corresponds to the following decision function:

$$y_p = W^T x + b \tag{1}$$

where W is an array of coefficients ("weights") and b is a bias term, both being adjusted during the fitting procedure. Mean squared error (MSE) is typically used as the loss function for linear regression:

$$\mathscr{L} = \arg\min_W \sum_k \left(y_k - y_{p,k}\right)^2 \tag{2}$$

where the sum is taken over the various training samples k which are used to fit the model.

Ridge regression (RR) uses a similar decision function as linear regression while the loss function contains a regularization term for the magnitude of the weights:

$$\mathscr{L} = \arg\min_W \sum_k \left(y_k - y_{p,k}\right)^2 + \lambda ||W||^2 \tag{3}$$

where the regularization term $\lambda||W||^2$ represents the L2-norm and λ is a scalar which defines the strength of regularization. Hence the fitting procedure will try to improve the quality of predictions while keeping the predictor weights as small as possible, balancing these two factors. If a predictor has small influence on improving the quality of predictions its weight will be decreased in favor of other predictors. The regularization term makes the ridge regression a more robust model in comparison to the simple linear regression.

Kernel Ridge Regression

Kernel Ridge Regression (KRR) differs from the ridge regression in the structure of the decision function. The input features are (implicitly) transformed into a higher dimensional feature space described by a kernel mapping $\Phi : x \mapsto x'$ which allows to describe non-linear relations in the initial space. The weights are then optimized in that higher dimensional feature space:

$$y_p = W^T \Phi(x) + b \tag{4}$$

Explicitly computing such transformations may be computationally expensive. This issue is circumvented by applying the "kernel trick" which exploits the fact that fitting and predicting only involves dot products between samples. The trick is to express feature space dot products, $K(x_1, x_2) = \Phi(x_1) \cdot \Phi(x_2)$ (the "kernel") in terms of the original predictors x_1, x_2 such that it is not necessary to compute the mapping Φ in the first place. Commonly used kernels involve radial basis function kernel (RBF), polynomial kernel (Poly) and linear kernel. Note that for the linear kernel KRR is similar to RR if a similar loss function is employed. The decision function is:

$$y_p = \sum_k \alpha_k K(x_k, x) = \sum_k \alpha_k \, \Phi(x_k) \cdot \Phi(x) \tag{5}$$

where K is the considered kernel and α_k are the Lagrange multipliers used during the fitting step; they act as weights for the training samples with index k.

Support Vector Machine Regression

In a nutshell, Support Vector Machine Regression (SVR) differs from KRR in its loss function which causes a subset of the training data to be selected (the *support vectors*) for usage with the decision function. Only certain training samples for which $|y - y_p| > \epsilon$ contribute to the loss function. Once an optimal solution is found the corresponding samples constitute the support vectors. The decision function is:

$$y_p = \sum_k (\alpha_k - \alpha_k^*) K(x_k, x) + b \tag{6}$$

where α_k, α_k^* are Lagrange multipliers corresponding to the constraints of the optimization and x_k are the selected support vectors. One has to note that both kernel based algorithms allow choices for the kernel as well as for several hyper-parameters which require either domain knowledge or exhaustive search in hyper-parameter space.

Multi-Layer Perceptron

A fully-connected feed-forward Multi-Layer Perceptron (MLP) is a specific architecture for artificial neural networks (ANN) which is represented by consecutive layers of nodes where all nodes of two consecutive layers are connected to each other. Each node sums all its weighted inputs and transforms the result using an *activation function* g. The activation function should be non-linear in order to represent non-linearities in the data and it must be differentiable in order to comply with the fitting procedure. Typically used activation functions are *sigmoid* or *ReLU*. An MLP with at least one hidden layer is known to be a universal approximator [6] and can be shown to represent any function given a sufficient number of nodes. The decision function for a single hidden layer perceptron is described by

$$o_n = g\left(\sum_{m=1}^{M} W_{nm} \cdot g\left(\sum_{l=1}^{N} W_{ml} \cdot x_l + b_m^h\right) + b_n^o\right) \tag{7}$$

For MLPs with multiple hidden layers the propagation of inputs repeats in a similar fashion.

Weights are usually randomly initialized and then iteratively updated during the fitting procedure in order to minimize the selected loss function (e.g. MSE). In order to do so, the gradient of the loss function with respect to the MLP's weights and biases is calculated using an appropriate optimizer (e.g. gradient descent) at each iteration and then applied with a previously chosen learning rate. This gradient computation involves *backpropagation* of the error to previous layers of the MLP. There are several variants for calculating the gradients during backpropagation [7].

SIMULATIONS

The heuristics mentioned in the previous section rely on the availability of reasonably accurate IPM simulation models [8] which provide the data for model fitting. The Virtual-IPM simulation tool [9] has been used for simulating the movement of electrons inside the IPM region. Table 1 shows the beam parameters, typical for LHC, used for the simulations. Single bunches are modeled by three-dimensional Gaussian charge distributions. The electric field of bunches is computed via an analytic formula for a two-dimensional Gaussian charge distribution [10] in the transverse plane while the longitudinal dimension is taken into account by rescaling the field with the particular line density. The longitudinal field component is neglected. Both the electric and the magnetic field of the beam are taken into account. The external electric and magnetic guiding fields are modeled to be uniform. The initial velocities of electrons are generated according to a double differential cross section for a Hydrogen target [11]. The passage of only a single bunch is simulated because the extraction times for electrons are only a few nanoseconds while the bunch spacing is 25 ns.

Table 1: Simulation Parameters. The bunch population, length, width and height are randomly varied within the given ranges, resulting in a total of 21 021 data points.

Parameter	**Values**
Beam particle type	Protons
Energy	6.5 TeV
Bunch population	1.1×10^{11} to 2.1×10^{11}
Bunch length (4σ)	0.9 ns to 1.2 ns
Bunch width (1σ)	270 µm to 370 µm
Bunch height (1σ)	360 µm to 600 µm
Electrode distance	85 mm
Applied voltage	4 kV
Magnetic field	0.2 T
Number of sim. particles	1 000 000
Time step size	0.3125 ps

Benchmarking of the simulation against LHC IPM data was not possible, therefore measurements on SPS IPM were performed. First, the IPM profile obtained with 200 mT magnetic field, for which no distortion is expected, was compared with wire scanner (WS) measurement. It was found that the IPM profile suffers from a broadening which can be described as a large, 520 µm point-spread function (PSF). Applying this PSF to the simulated profiles shows good agreement with the measurements for reduced magnetic fields down to 16 mT as presented in Fig. 1.

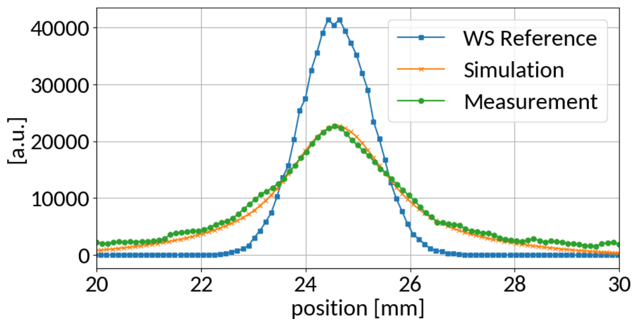

Figure 1: Measurement with SPS IPM Horizontal at 16 mT magnetic field strength, compared to simulation data.

RESULTS

The simulated profiles were binned with 100 µm, corresponding to the resolution of the acquisition system. Together with the bunch length σ_z and the bunch intensity (number of protons per bunch) they were used as predictors for the target output σ_x. Before fitting the ML models the data was processed by dropping predictors that have zero variance (profile points where no signal was recorded) and scaling the remaining predictors to have zero mean and unit variance $x \rightarrow (x - \mu)/\sigma$.

Results were obtained using the Python machine learning package scikit-learn [12] as well as Tensorflow [13] with Keras [14] interface. The following hyper-parameters were used for the different models. KRR: degree = 2, $\alpha = 10^{-3}$; SVR: $\gamma = 10^{-4}, C = 10^3, \epsilon = 10^{-8}$; MLP: 4 hidden layers (200, 170, 140, 110 nodes), ReLU activation function, Adam optimizer ($\eta = 0.001$), MSE loss, 100 epochs with a batch size of 8. Detailed description of these hyper-parameters can be found in the respective software package.

In order to assess the quality of a particular model we compared the mean and the standard deviation of the residuals $y - y_p$ as well as the R2-score, explained variance (EV) and mean squared error (MSE) [12]. The results for the different ML models are depicted in Table 2.

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

	μ**(res)**	σ**(res)**	**R2**	**EV**	**MSE**
LR	0.012	0.449	0.99976	0.99976	0.201
KRR	0.005	0.340	0.99986	0.99986	0.115
SVR	0.006	0.349	0.99985	0.99985	0.121
MLP	0.232	0.370	0.99977	0.99984	0.190

Linear regression already provides a very good estimate of input beam profile width from the simulated output profile. KRR, SVR and MLP show incremental improvement. The performance of these algorithms has been tested in the presence of additive Gaussian noise and the estimated profile uncertainty increases linearly with noise [5]. For a full profile reconstruction, ANNs might be advantageous over other methods due to the possibility of a full mapping of measured profile to the actual profile. The study will be continued using the measured data.

SUMMARY

A novel method for resolving IPM profile distortion under the influence of magnetic guiding fields based on machine learning is presented. The first investigations, using simulated data, yield promising results. Next steps include estimation of influence of error sources on predictions, optimization of model selection and application of the method to measured data. The method has a potential to extend usability and reduce cost of IPMs for high brightness beams.

REFERENCES

[1] D. Vilsmeier *et al.*, "Investigation of the effect of beam space charge on electron trajectories in Ionization Profile Monitors", in *Proc. HB2014*, East Lansing, MI, USA, November 2014.

[2] D. Vilsmeier, "Profile distortion by beam space-charge in Ionization Profile Monitors", CERN-THESIS-2015-035, 2015.

[3] M. Patecki *et al.*, "Electron Tracking Simulations in the Presence of the Beam and External Fields", in *Proc. IPAC2013*, Shanghai, China, May 2013.

[4] M. Patecki, "Analysis of LHC Beam Gas Ionization monitor data and simulation of the electron transport in the detector", CERN-THESIS-2013-155, 2013.

[5] R. Singh *et al.*, "Simulation Supported Profile Reconstruction With Machine Learning", in *Proc. of IBIC2017*, DOI: `https://doi.org/10.18429/JACoW-IBIC2017-WEPCC06`, 2018.

[6] K. Hornik, "Approximation Capabilities of Multilayer Feedforward Networks", *Neural Networks*, 4(2), 251-257, (1991).

[7] K. P. Murphy, "Machine Learning: A Probabilistic Perspective", *The MIT Press*, 2012.

[8] M. Sapinski *et al.*, "Ionization Profile Monitor Simulations - Status and Future Plans", in *Proc. IBIC2016*, Barcelona, Spain, Sep. 2016, DOI: `https://doi.org/10.18429/JACoW-IBIC2016-TUPG71`, 2017.

[9] D. Vilsmeier *et al.*, "A Modular Application for IPM Simulations", in *Proc. of IBIC2017*, East Lansing, MI, USA, DOI: `https://doi.org/10.18429/JACoW-IBIC2017-WEPCC06`, 2018.

[10] M. Bassetti and G. A. Erskine, "Closed expression for the electrical field of a two-dimensional Gaussian charge", CERN-ISR-TH/80-06, 1980.

[11] A. B. Voitkiv *et al.*, "Hydrogen and helium ionization by relativistic projectiles in collisions with small momentum transfer", *J.Phys.B: At.Mol.Opt.Phys*, vol. 32, 1999.

[12] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] Martín Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from `tensorflow.org`

[14] François Chollet *et al.*, "Keras", `https://keras.io`, 2015.