Karlsruhe University of Applied Sciences
Faculty of Computer Science and Business Information Systems

European Organization for Nuclear Research (CERN)

Evaluation of Machine Learning Methods for LHC Optics Measurements and Corrections Software

Author: Elena Fol

Supervised by: Dr. Rogelio Tomas Garcia, (CERN). Prof. Dr. Peter Henning (HSKa)

Submision date: August 29, 2017

Abstract

The field of artificial intelligence is driven by the goal to provide machines with human-like intelligence. However modern science is currently facing problems with high complexity that cannot be solved by humans in the same timescale as by machines. Therefore there is a demand on automation of complex tasks. To identify the category of tasks which can be performed by machines in the domain of optics measurements and correction on the Large Hadron Collider (LHC) is one of the central research subjects of this thesis.

The application of machine learning methods and concepts of artificial intelligence can be found in various industry and scientific branches. In High Energy Physics these concepts are mostly used in offline analysis of experiments data and to perform regression tasks. In Accelerator Physics the machine learning approach has not found a wide application yet. Therefore potential tasks for machine learning solutions can be specified in this domain. The appropriate methods and their suitability for given requirements are to be investigated. The general question of this thesis is to identify the opportunities to apply machine learning methods to find and correct the errors in LHC optics and also to speed up beam measurements.

Declaration by the candidate

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been marked.

The work has not been presented in the same or a similar form to any other testing authority and has not been made public.

August 29, 2017

Contents

1 Introduction

The Large Hadron Collider (LHC) is a double beam circular collider operated by the European Organisation for Nuclear Research and is used for High Energy Physics research. The Optics and Measurements Correction (OMC) software [1–3] targets to measure and correct several optics parameters of the LHC. As the machine has very tight tolerances, a good control of the optics is critical for machine protection and performance.

The purpose of the software is to analyze data measured during the LHC operation and compute optics corrections to get the best performance of the LHC. In order to compute the corrections, the measured data has to be compared with the design machine. The deviations from design have to be identified by special analysis methods and compensated by improved machine settings according to computed corrections.

The extraordinary optics control is a crucial factor to achieve the target luminosity at ATLAS and CMS experiments. In 2016 the achieved optics control exceeded the expectations of the design $\beta^* = 0.55$ m at 7 TeV. This success was the result of the improvements to the measurement, correction algorithms and technical instrumentation [4–6]. Note that providing the two main experiments with the same amount of luminosity and hence the same discovery potential is one of the most important targets of LHC operation.

Large part of the deviation from the design optics is caused by imperfections in magnets in experimental regions. Thus the biggest challenge for the optics correction is introduced by the requirement of squeezing the beam in interaction regions. Since 2016 more precise β-function measurements are performed using the gradient modulation in the quadrupoles as described in [7–9]. The implementation of this method is included into the OMC Software and it is used to calculate local corrections, which are used mostly to correct the optics around experimental regions. Local corrections reduced the β -beating to a peak of 20% [4], to reach a better optics control global correction approach was applied to correct the optics around the whole machine. This approach has been improved in 2016 by taking the measurements uncertainties into account as described in sections 4.2 and 4.3.3. A detailed overview on optics measurements and correction methods and their implementations in OMC software is given in chapter 4. Further improvements to the OMC Software developed as part of this thesis and used during the LHC commissioning in 2017 are presented in chapter 5 and possible machine learning application that are needed to reduce human intervention and increase the correction performance are presented in chapters 6 and 7.

Recently, machine learning methods have found application in high energy physics mainly in experiments data analysis. The machine learning regression methods are used to reconstruct the collisions and classification tasks are performed to find the event that might be of interest. Finding the exotic particles requires solving difficult signal-background classification problems. Application of Deep Learning algorithms allows to distinguish between signal and background more efficiently [10]. The Convolutional Neural Networks which are traditionally applied on image processing are also used for pattern recognition solutions in particle detectors. The first application of Convolutional Neural Networks in High Energy Physics to classify neutrino events is presented in [11].

In accelerator physics, neural networks have been applied to correction of orbit [12, 13] and for modeling and control of storage rings [14].

The technical challenges of applying machine learning methods in the domain of accelerator modeling and control are complex dynamics and massive amount of parameters to include into analysis, as well as wide range of interacting systems. Recently, optimization techniques such Simplex and random-value optimization have been applied in several storage rings [5]. Among others, the optimization methods using genetic algorithms have shown a great potential [15, 16]. In this regard, the combination of neural network and the genetic algorithm approaches could be a matter of interest for more efficient optimization systems. Certain techniques that combine these approaches such as *NEAT* and *NeuroGenesys* are presented in [17, 18].

In the past decades, applying machine learning methods to accelerator issues suffered from insufficient computational capability. Nowadays the computational power of modern machines has increased especially due to to the usage of GPUs which can handle computationally expensive algorithms. Bigger data sets can be obtained which are needed for training to solve complex classification and prediction tasks using neural networks. For example, at Fermilab a neural network model of the Radio Frequency Quadrupole is used in predictive control during operations [19].

Future accelerator projects, like HL-LHC and FCC will continue challenging optics control techniques in terms of accuracy, resolution, speed and instrumentation [5]. Deep Learning is a promising solution to accomplish these complex tasks since it allows faster performance of optimization or systematic studies. Deep Learning algorithms are suited to use parallelized computing methods, hence they have a potential to improve the performance of data analysis and reduce the demand on required resources.

The application of machine learning methods in current OMC Software could decrease the man power needed during the measurement process. Moreover the improvement in precision of corrections could be achieved leading to a higher machine performance. Possible applications of machine learning methods in optics measurements and correction are identification of bad beam position monitors data, noise reduction in measured data and corrections computing based on magnetic field error prediction.

The data given by past machine operation and performed corrections are a powerful basis for potential prediction of machine imperfections and beam behavior. Learning to predict beam availability times could provide better estimates and better planning. Since the data analyses based on machine learning approach require high abstraction level and data preprocessing, the development of new features in order to improve the data acquisition process of the optics measurements is needed. The automation of the measurements process and corrections computing is therefore among the aimed development targets of this thesis.

2 Introduction to Machine Learning Concepts

Machine Learning techniques are concerned with the target of how to build computer programs and algorithms that automatically improve with experience by learning from examples. In this context *learning* is defined by Mitchell [20] as follows:

> A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Machine Learning methods found their application in a wide range of areas, ranging from technology, science and medical diagnosis to commerce and social networks. The broad usage of Machine Learning techniques demonstrates their suitability for various problems and tasks.

Depending on problem domain and presence of learning examples, different approaches can be applied. If pairs of input and desired output are provided, an algorithm can generalize the problem from given examples and produce prediction for unknown input. Machine learning algorithms that learn from input/output pairs are called *supervised learning* algorithms because supervision is provided to the algorithms in the form of determined outputs for each given example.

In opposite to supervised learning approach, *unsupervised learning* algorithms solve the tasks where only the input data is known. For both kinds of learning algorithms, it is crucial to have a computer understandable, abstract representation of the data.

Depending on the desired prediction target - category or a quantity, the problem can be represented either as *classification* or *regression* task. Feature engineering provides techniques to mine the information from the data in order to simplify the model learning by discovering hidden relation in the data and building new features. As mentioned above, building an appropriate dataset is a crucial milestone for the further process as model selection and performance depend on the amount and quality of the given data. The performance of the model has to be evaluated using adequate quality metrics considering the data representation, desired result and model design.

Since the data is usually imperfect, certain preprocessing actions have to be taken before a model can learn from this data. Data preprocessing involves normalization and re-scaling of the data points, outliers elimination, encoding of categorical values and other data transformation techniques depending on the given data representation. The actual model training is followed by evaluation in order to measure the generalization ability

and prediction accuracy of the designed estimator. In the following a brief overview on fundamental Machine Learning concepts and methods is presented.

2.1 Regression and Classification

Machine Learning problems can be divided into two major types - *classification* and *regression*. The core of classification problems is to assign new input to one of a number of discrete classes or categories. In regression problems the output represents the values of a dependent variable. Both regression and classification problems can be seen as particular cases of function approximation. In the case of regression problems it is the regression function which should be approximated, while for classification the functions are the probabilities of membership of different classes expressed as functions of the input variables [21].

Linear Model

Linear models make a prediction using a linear function of the input features. For regression, the general prediction for the prediction output $\hat{y}$ of a linear model is formulated as follows:

$$\hat{y} = b + \sum_{i=1}^{j} w_i x_i \tag{2.1}$$

where x_i denotes the features of a single data point, j the number of features, w and b are the learned parameters of the model - weight of the input and the bias that adds noise to the linear relationship between input and output variable. The simplest linear regression model optimizes the parameters w and b by minimizing the mean squared error (MSE) between prediction $\hat{y}$ and the true regression target y on the training set size n as shown in Eq. (2.2).

$$MSE = \frac{1}{n} \sum_{n=1}^{n} (\hat{y} - y)^2 \tag{2.2}$$

This optimization approach is also called *Least Squares*.

Polynomial Regression Tasks

Some learning tasks require nonlinear predictors, such a fitting of one dimensional polynomial function of degree n, that is,

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \tag{2.3}$$

where $(a_0 ... a_n)$ is a vector of coefficients of size $n + 1$. One way to train the model for prediction of such problems is to reduce the problem to linear regression [22]. To

define a polynomial regression task as linear regression problem, a mapping such that $\psi(x) = (1, x, x^2, ..., x^n)$ is introduced and the problem reduces to

$$p(\psi(x)) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n = \langle a, \psi(x) \rangle \tag{2.4}$$

so that the optimal vector a can be found by minimizing MSE as in the case of linear regression.

Logistic Regression

Logistic regression is usually applied to classification tasks: the family of functions $h(x)$ in the interval $[0, 1]$ can be interpreted as the probability that the label of x is 1. The sigmoid function used in logistic regression is described as follows

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

The loss function in case of logistic regression is defined as a minimization problem for a given given input $(x_1, , ..., x_n)$ in form of

$$\min log(1 + exp(-(b + \sum_{i=1}^{n} w_i, x_i)) \tag{2.6}$$

as the loss function should define how bad it is to predict some function $h(x) \in [0, 1]$ given that the true label is $y = 1$. The label y is predicted as in the case of linear regression by $b + \sum_{i=1}^{n} w_i, x_i$.

Stochastic Gradient Descent

As shown above, both regression and classification involve optimization problem in form of minimizing the loss function during the model training. The optimization technique called *gradient descent* [23] founds on searching for the global minimum of a function $f(x)$ by moving x in small steps with the opposite sign of the derivative. Optimization algorithm may fail to find a global minimum in presence of multiple local minima or plateaus. The problem arises especially in case of multidimensional functions. However, learning algorithms often accept local solutions in case they correspond to significantly low values of loss function, even if they are not globally optimal.

In order to minimize $f(x)$, the gradient descent finds the direction in which f decreases fastest using directional derivative. The function $f(x)$ is then decreasing by moving in the direction of the negative gradient. Gradient descent proposes a new point

$$x_{i+1} = x_i - \epsilon \nabla_x f(x) \tag{2.7}$$

where ϵ is the *learning rate* determining the size of the step and $\nabla_x f(x)$ partial derivative. In context of learning, the function $f(x)$ is the loss function that can be expressed by the MSE as described above. During the training on a data set, gradient descent computes the average per complete training set in each step, which makes the solution computational expensive and time demanding.

In *stochastic gradient descent* algorithm, instead of averaging the gradient of the loss function over the complete training set, the loss function is computed for particular training sample. Thus, stochastic gradient descent approximates the solution and reduces the required computational resources and time significantly. Stochastic gradient descent is widely used in modern learning techniques. Several extensions of the algorithm are introduced during the past decades such *momentum* method [24], *AdaGrad* [25] and *Adam* [26].

2.2 Neural Networks

The first attempts of computations based on artificial neural networks (ANN) were performed already in 1943 [27]. Many ideas and concepts which define the field of neural network research were formulated between 1943 to 1968. A brief overview on the early era of neural network research can be found in [28].

ANNs are well suited for learning tasks in which data are represented by noisy, complex sensor signals and the target output function may consist of several attributes [20]. Single-layered networks with threshold activation function were introduced by Rosenblatt [29] and called *perceptron*. Basically, a perceptron calculates a linear combination of inputs and gives 1 or -1 as output y based on some threshold (activation function). Formally the computed output is described as

$$y = \begin{cases} 1 & \sum_{i=1}^{n} w_i x_i + b > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.8}$$

where w_i is weight of input x_i and b the bias that shifts the decision boundary away from the origin and is independent from any input. Formally a single-layer network with the output $\hat{y}$ is described as

$$\hat{y} = \varphi(\sum_{i=1}^{n} w_i x_i + b) \tag{2.9}$$

where φ is the *activation function* of the layer. A single-layer perceptron uses a step-function, however the most commonly used activation function is the sigmoid function (2.5) that adds non-linearity to the output. Figure 2.1 shows a sigmoid unit that describes a simple network with one output. The sigmoid unit can be understand as a combination of two components - first, it computes the linear output *net* and then, it applies the sigmoid

function to convert the output into a probability. Other option for activation function in ANN are hyperbolic tangent (TanH) and rectified linear unit (ReLU) [30] functions.

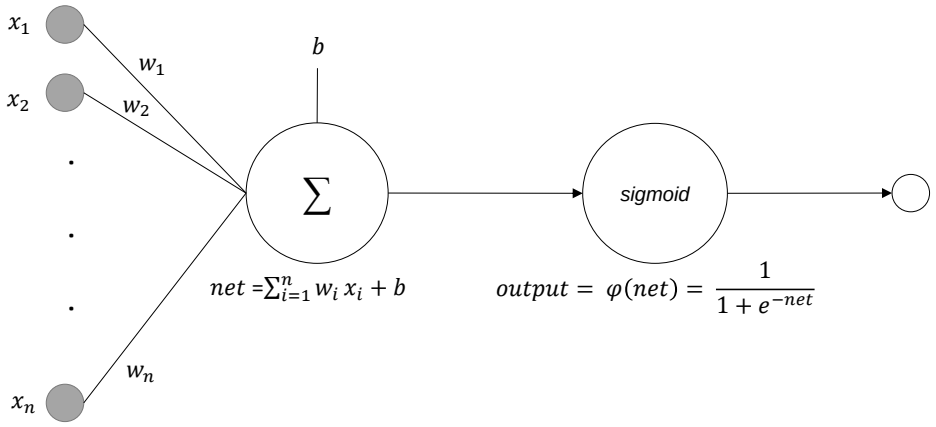

Figure 2.1: The sigmoid unit.

The multilayer feedforward neural networks, also called multi-layer perceptrons are the most widely studied and used ANN model. A multilayer network consists of one layer of input units, one or more hidden layers and one output layer with one or more output units. Multilayer ANNs are used more widely than single-layer perceptrons since they can solve more complex non-linear problems using the hidden layers.

Through the connections between the units (neurons), knowledge can be generated and stored as weights of the connections between different units. Hidden layers are the core elements for learning the patterns in the data and mapping the relationship between the input and output variables. Figure 2.2 shows a three layered network with one hidden layer. The role of the output layer is then to provide some additional information from the features to complete the task that the network must perform, e.g. to predict the label 0 or 1 of the given input.

To simplify the formal description of a multilayer network the bias can be seen as a weight connecting a neuron with the input $x_0 = 1$, so we can express the output of a single sigmoid unit o as

$$o = \varphi(\sum_{i=0}^{n} w_i x_i) \tag{2.10}$$

Then a three-layer ANN can be written as a model combining the layers of a network as follows:

$$\hat{y} = \varphi_2(\vec{w_2}\varphi_1(\vec{w_1}\vec{x})) \tag{2.11}$$

where φ_1 is the activation function of hidden layer and φ_2 is the activation function of the output layer. Note that instead of sigmoid function, any other activation function such TanH, ReLu or linear function can be used.

A multi-layered network performs learning using *backpropagation* algorithm [29] that

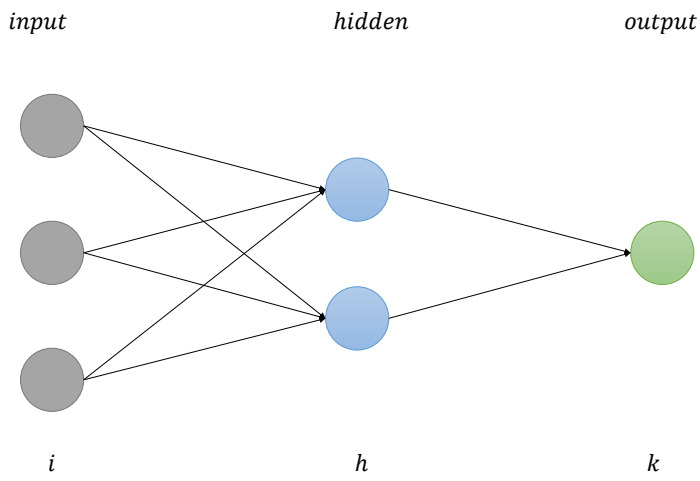

Figure 2.2: A simple multilayer network.

learns the weights given a multi-layer network with a fixed set of units and interconnections. Backpropagation computes the derivatives of the composed functions that describe the layers of the network to obtain the gradients with respect to all weights.

In backpropagation gradient descent is applied as optimization method in order to minimize the squared error (see 2.1) between the network output values and the target values for these outputs [20] by updating the connection weights. To describe the stochastic gradient descent version of backpropagation algorithm following notation is used:

$\vec{x}, \vec{y}$ training sample with network inputs $\vec{x}$ and target outputs $\vec{y}$

η learning rate

o_k output unit k

o_h output of hidden unit h

w_{kh} weight of the connection between output unit k and hidden unit h

x_{ji} input from any unit i to unit j

w_{ji} corresponding weight.

The algorithm propagates each sample input through the network, calculates the error of the network output for this sample using the target output, computes the gradient with respect to the error of this sample, then updates the weights in the network. This procedure is iterated until the network error reaches acceptable value. The algorithm is formally described as follows:

1. For each output unit k, calculate the error δ_k associated with the unit k

$$\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k) \tag{2.12}$$

with $o_k(1 - o_k)$ derivative of the sigmoid function.

2. For each hidden unit h, calculate the error

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k \tag{2.13}$$

3. Update each network weight w_{ij}

$$w_{ij} \rightarrow w_{ij} + \Delta w_{ij} \tag{2.14}$$

where

$$\Delta w_{ij} = \eta \delta_j x_{ij} \tag{2.15}$$

Since training samples provide target values y_k only for network outputs, no target values are directly available to indicate the error of hidden units values. The error term for hidden unit h is calculated by summing the error terms δ_k for each output unit influenced by h, weighting of the δ_k by w_{kh} - this weight characterize how the hidden unit h influence the error in the output unit k [20].

One interesting property of backpropagation learning is its ability to discover useful intermediate representation at the hidden layers inside the network by defining new hidden layer features that are not explicitly given in the input representation. Thus properties of input data that are most important to learn the target function can be discovered. This ability is a key feature of ANN learning that shows the advantage of this method in contrast to learning techniques that use only predefined features. ANNs with many hidden layers called *deep neural networks* are able to use fewer units per layer and have a better generalization ability, however the optimization of these networks is not trivial. The optimal architecture for a specific problem has to be found through experimentation using a validation data set to evaluate the prediction error.

2.3 Decision Trees and Ensemble Methods

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Considering the case of classification, decision trees sort down the input instances from the room to some leaf node. Usually, the splitting is based on one of the features of the input or on a specified set of rules. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and a tree itself to a disjunction of these conjunctions [20, 31, 32]. Note that a decision tree incorporates both nominal and numeric attributes. Each leaf is assigned to one class representing the most appropriate target value. Alternatively, the leaf may hold a probability vector indicating the probability of the target attribute having a certain

value. The estimation produced by the decision tree can be described as the estimation for a given partition of class probability from N_m examples in partition m and N_k examples in partition of class k:

$$p_{mk} = \frac{N_k}{N_m} \tag{2.16}$$

On one side, a single decision tree is simple to understand and requires small data sets. On the other hand using a single tree a model can overfit very fast, especially by increasing the tree depth. Moreover, over-complex models become instable to small variations in data. One possible solution to overcome this problem is to build ensembles of trees [33].

By training several slightly different models and taking the majority vote in case of classification or the average prediction in case of regression, the variance of the model can be reduced. One of the most commonly used ensemble algorithms for decision trees is *Random Forest* [34].

2.4 Generalization and Overfitting

In supervised learning, a model aims to be able to make accurate predictions on new, unseen data based on the training on the known input. The ability to predict on unexplored data is called *generalization*, which is the most important property of the model to be evaluated [21]. In most cases, training and test data sets have enough common properties and relations in the data, so that model can predict accurately also on the training test. However, in case of high-complex models, very high accuracy can be achieved on the training dataset, although a possible consequence is a very poor prediction on the test data. Too complex models tend to fit to the individual data points of the given training data too closely and hence perform insufficient on new data. Building a model that is too complex for the given amount of information is called *overfitting*.

One of the most successful methods for overcoming the overfitting problem is to provide a set of validation data to the algorithm in addition to the training data. The error is then monitored with respect to the validation set, while using the training set to drive the learning procedure [20]. The performance of the model measured on the training set should be nearly equal to the one measured on the validation set. If the model performance on the training set is significantly higher, the model is overfitting.

On the other hand, if the model is too simple, it will not be able to perform accurate prediction on the training data and to learn all the data aspects and its variability. The complexity of the model corresponds directly to the variation of inputs contained in the training data. Hence, it is crucial to control the size and the variation of the data in order to build an appropriate model.

2.5 Unsupervised Learning

2.5.1 Autoencoders

An autoencoder is a specific type of a neural network that is trained to attempt to reproduce its input as its output. The network consists of two parts: an encoder function $h = f(x)$ describing a hidden layer h and a decoder that produces a reconstruction $r = g(h)$ as illustrated in Figure 2.3. Autoencoders are usually restricted to copy only input that approximates the training data. Therefore, the model is forced to prioritize which aspects of the input should be reproduced and to learn useful properties of the data in this way [35].

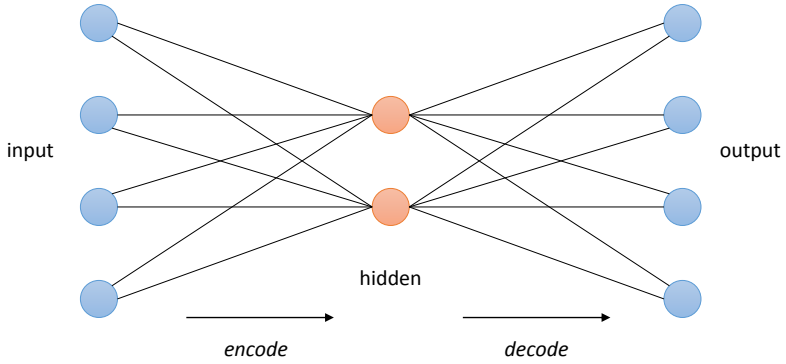

Figure 2.3: Autoencoder with single hidden layer.

In modern understanding autoencoders generalize the idea of encoder and decoder beyond deterministic functions to stochastic mappings $p_{encoder}(h|x)$ and $p_{decoder}(x|h)$. Initially, autoencoder concept was introduced in tasks of dimensionality reduction and feature learning. Recently, autoencoders are widely used in generative modeling. As autoencoder is considered as a special case of a feedforward network, they can use the same techniques for training, which are usually gradient descent methods for gradients computed by backpropagation. Apart from typical techniques autoencoders may learn using *recirculation* [36]. Recirculation is a technique based on comparison between activations of the network on the original input and the activations in the reconstructed input.

Undercomplete Autoencoders

Regarding the problem of obtaining useful features from data, an autoencoder built hidden layer of smaller dimension than the input can be used. This type of autoencoders is called *undercomplete autoencoders*. Since the representation is undercomplete, the autoencoder is forced to extract most important features of the training data. The learning process in

this case is described as minimizing a loss function

$$L(x, g(f(x))) \tag{2.17}$$

penalizing $g(f(x))$ for being dissimilar from x.

Regularized Autoencoders

Regularized autoencoders allow the training, choosing the code dimension and the capacity of the encoder and decoder based on the complexity of data representation to be learned. Using a loss function, regularized autoencoders control the model ability to copy its input to the output and provide other properties to the model such sparsity of representation and robustness to noise or missing data.

Sparse Autoencoders

A sparse autoencoder involves a sparsity penalty $\Omega(h)$ in its training criterion in addition to the reconstruction error. The training criterion is described as follows

$$L(x, g(f(x))) + \Omega(h) \tag{2.18}$$

where $g(h)$ is the decoder output and h is the encoder output defined as $h = f(x)$. The added penalty is simply a regularizer added to a network with the primary task to copy the input to the output.

Denoising Autoencoders

The denoising autoencoder is an autoencoder that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output [35]. Instead of adding a penalty to the loss function an autoencoder can be trained by changing the reconstruction error term. A denoising encoder minimizes the function $L(x, g(f(\tilde{x})))$, where $\tilde{x}$ is a copy of x that has been corrupted by some kind of noise. Denoising encoder eliminates this corruption instead of copying the original input to the output.

2.5.2 Association Rules

Association rule learning is a method to determine rule-based relations between variables. The purpose is the identification of strong rules that occur frequently in a given data set by measuring the significance of the rules [37]. A typical example of association rule mining application is the market basket analysis to identify which items are usually bought together to produce recommendations for customers or to understand the customers behavior.

Based on the concept of strong rules, Agrawal, Imielinski, and Swami [38] introduced the problem of mining association rules from transaction data in large databases as follows:

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items and T be a database of transactions. Each transaction t is represented as a binary vector, with $t_k = 1$ if item i_k is present in the transaction t and $t_k = 0$ otherwise. Let X be a set of some items in I. Hence, a transaction t satisfies X if for all items i_k in X, $t_k = 1$.

An *association rule* is an implication expression of the form $X \implies Y$, where X and Y are disjoint itemsets. The rule is satisfied with the confidence factor $0 \leqslant c \leqslant 1$ if at least fraction c of transactions in T that satisfy X also satisfy Y.

An important property of a dataset is *support count* $\sigma(X)$, which refers to the number of transactions in T that contain a particular itemset X. The rules have to satisfy additional constrains of two forms: syntactic constraints and support constraints. *Syntactic constraints* involve restrictions of items that can appear in the rule. For example only the appearance of specific item i_j can be of interest, either appearing in the consequent (right-hand-side) or rules that have the item i_j in the antecedent (left-hand-side) or in a combination of these constraints. The *support* for a rule is defined to be the proportion of transactions T that satisfy the union of items of the rule. While support $s(X \implies Y)$ corresponds to statistical significance and *confidence* $c(X \implies Y)$ determines how frequently items in Y appear in transactions that contain X,

$$\begin{aligned} s(X \implies Y) &= \frac{\sigma(X \cup Y)}{N} \\ c(X \implies Y) &= \frac{\sigma(X \cup Y)}{\sigma(X)} \end{aligned} \tag{2.19}$$

with support counts being $\sigma(X \cup Y)$, $\sigma(X)$ and N total number of transactions in T.

Support is used to eliminate unimportant rules since the rules with low support may occur basically by chance. Confidence measures the reliability of the inference made by a rule. For a given rule $X \implies Y$, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Confidence also provides an estimate of the conditional probability of Y given X [39].

The problem of rule mining can be decomposed into two subproblems:

1. **Frequent Itemset Generation**, which aims to generate combinations of items with a support above a certain threshold, called *minsupport*. Using syntatic constraints further limits for admissible combinations can be created. For example, if only rules involving a specific item i_k are of interest, then only those combinations that contain i_k are desired. To improve the efficiency of the frequent itemset generation, an important property called the *Apriori property* is used. This property implies that all subsets of a frequent itemset must also be frequent.

2. **Rule generation** for obtained frequent itemsets to extract all the high-confidence rules. Association rules are required to satisfy both a minimum support and a minimum confidence constraint at the same time.

2.5.3 Cluster Analysis

Cluster analysis includes methods of grouping or separating data objects into clusters, such that dissimilarity between the objects within each cluster is smaller than between the objects assigned to different clusters. Some of clustering methods also aim to build hierarchy of clusters by grouping clusters themselves that at each level of the hierarchy, clusters within the same group are more similar than those in different groups. The core of all types of clustering methods is the notion of the similarity degree between the individual objects in the data [40, 41].

Cluster analysis is used in a wide range of applications. Data clusters can be considered as a summarized representation of the data, such as group labels can provide description of patterns of similarities and differences in the data. Moreover, clustering can be used for classification prediction, such that classification of unseen data is performed based on knowledge about the properties of present data and by evaluating their similarity to the classifying data sample. For the case if all the variables are continuous, proximities between two individual data points p and q are defined by *distance metric*. The most popular metric for continuous features is the euclidean distance:

$$d(p, q) = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2} \tag{2.20}$$

The simplest and the most commonly used clustering algorithm is k-means [42], it starts with a random initial partition selection and builds clusters based on the similarity between cluster centers and the data points. Other popular clustering algorithms are Expectation Maximization [43], Nearest Neighbor [44] and Fuzzy clustering [45]. ANNs have also found application in cluster analysis in form of networks that use patterns (data points distribution) as the input associated with output nodes that represent clusters, the weights update procedure in this case is similar to classical clustering approaches [41] since the task of the network is to recognize the patterns in data.

3 Fundamentals of Beam Optics

3.1 Large Hadron Collider

The Large Hadron Collider is CERN's biggest accelerator and the most powerful machine in the world with the energy of proton beams collision up to 14TeV. The beams are pre-accelerated by the LHC injector chain which consists of several accelerators as shown in the Figure 3.1.

The beam sizes at the Interaction Point (IP) have to be squeezed in order to provide maximum number of collisions to the experiments. After the squeeze the beams are colliding at 4 different IPs. Bunches cross in average about 30 million times per second, so the LHC generates about 1 billion particle collisions per second.

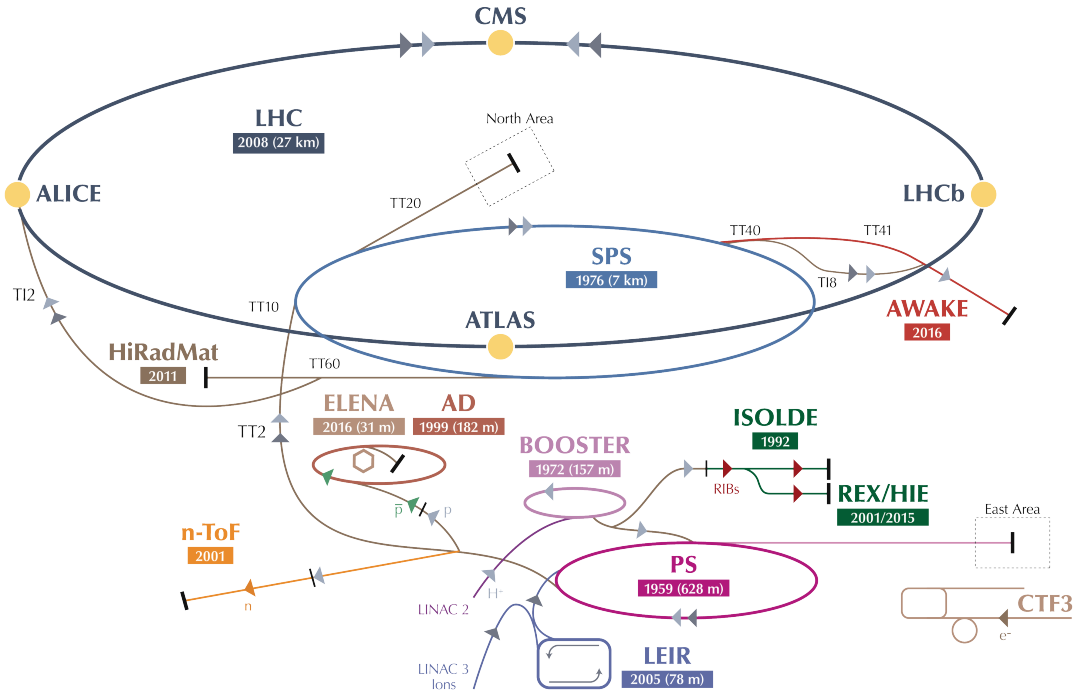

Figure 3.1: CERN accelerator complex.

The 4 main detectors of the LHC are indicated in Figure 3.1: ATLAS and CMS aim to explore new heavy particles, LHC beauty (LHCb) studies focus on the antisymmetries between matter and anti-matter and ALICE is specialized in heavy-ions collisions to study the properties of quark-gluon plasma.

The different experiments require different rates of collisions, therefore the beam is

squeezed more for the general purpose experiments (ATLAS and CMS) and less for LHCb and ALICE. The requested beam settings also depend on different modes of operation. This flexibility given by wide range of operation modes extends the physics potential of the LHC but leads to higher complexity. From the optics point of view this complexity implies the need of different optics settings which have to be tested and corrected.

3.2 Beam Focusing

While dipole magnets can be used to bend the beam's direction, focusing is done by quadrupole magnets [46]. Since quadruploles cannot focus in both planes at the same time, a special lattice design has to be used in order to focus the beam. The simplest possible strong focusing lattice is the *FODO cell* which represents a unit cell consisting of a pair of quadrupoles each followed by a drift space. The focusing is achieved by combining the focusing and defocussing quadrupoles which can be considered as lenses compared to light optics [47]. A beam line built on repeated FODO cells is shown in Figure 3.2.

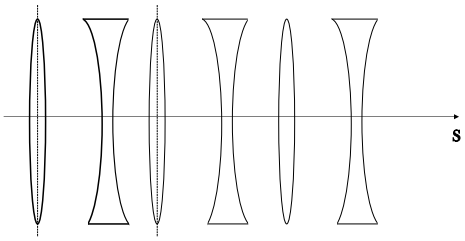

Figure 3.2: Beam line in a FODO cell consisting of focusing and defocussing magnets with drift space between the elements.

Replace the drift space with a dipole magnet to bend the beam, one obtains a FDBD cell. Another possible lattice design is a triplet, which contains three quadrupoles with the polarity of the center quadrupole opposite to the two quadrupoles on the outside of a cell. The Figure 3.3 shows the described lattice configurations.

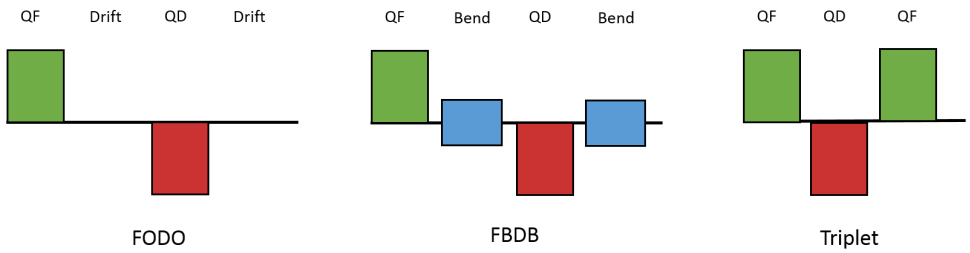

Figure 3.3: Different magnet configurations of a lattice design in particle accelerators.

In a high energy circular collider, FBDB cells are frequently used for transporting charged particle beams in the arc sections, while triplets are usually used in the interaction

region to strongly focus the charged particle beam into small beam sizes to facilitate collisions.

3.3 Beam parameters

β -function

The Betatron function is one of the main parameters to describe the beam optics. It defines the beam dimension together with the beam emittance which measures the area covered by particles in the phase space. In case of large β the beam is less focused, hence occupying a larger transverse space compared to locations with small β. Thus the beam envelope is defined as

$$e(s) = \sqrt{\beta(s)\epsilon}, \tag{3.1}$$

where ϵ defines the size of the space occupied by the beam particles.

The beating of the betatron function (β -beating) describes the ratio of the measured betatron function with respect to the nominal design function.

$$\beta - \text{beating} = \frac{\beta_{meas} - \beta_{model}}{\beta_{model}} \tag{3.2}$$

The β^* is defined as the β-function at the Interaction Point where the particles collide.

Betatron tune

The betatron tune gives the number of oscillation periods for one turn of the machine. The tune is related to the β -function as follows:

$$Q = \frac{1}{2\pi} \oint \frac{ds}{\beta(s)} \tag{3.3}$$

It affects the dynamics of the beam motion and at certain values it leads to beam instabilities. As in any oscillating system, the resonance conditions have to be avoided by keeping the frequency of the transverse motion not equal to (or an integer multiple of) the revolution frequency. It is desired to keep the tune away from fractional values with small denominators such as $\frac{1}{2}$, $\frac{1}{3}$.

Dispersion and Chromaticity

Dispersion is caused by the fact that the particles with different energies are bent differently. A particle with higher energy will be bent less compared to the one with lower energy. The dispersion is described as the deviation from the reference orbit $\Delta x(s)$ with respect

to relative momentum deviation $\Delta p/p$:

$$D(s) = \frac{\Delta x(s)}{\Delta p/p} \tag{3.4}$$

The deflection of a particle by a quadrupole magnet in the beam line depends on the trajectory of the particle as it enters the quadrupole, the length of the quadrupole, the magnetic fields in the quadrupole, and on the energy of the particle. The dependence of the particle motion on the energy of the particle will lead in particular to the variation of the tune with energy [47]. This variation is known as Chromaticity and is illustrated in the Figure 3.4 and described as

$$Q' = \frac{\Delta Q}{\Delta p/p} \tag{3.5}$$

where ΔQ is the tune change and $\Delta p/p$ is the relative momentum deviation. The natural chromaticities are always negative, which is to be expected since focusing is less effective for higher energy particles ($\delta > 0$).

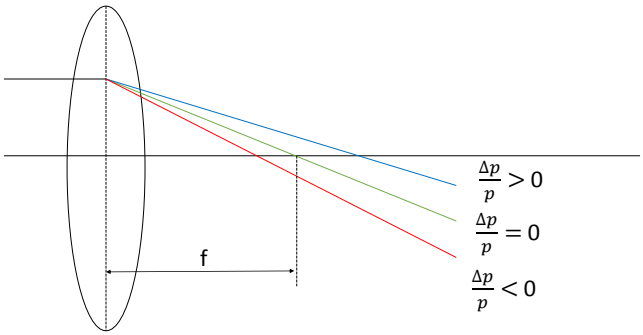

Figure 3.4: Higher energy particles are focused less than particle swith nominal energy and lower energy particles are over-focused.

Coupling

The horizontal and vertical planes are coupled in case of random and systematic skew-quadrupole errors in the main dipole and quadrupole magnets as well as in presence of offset in sextupole magnets. Normally, the preferred behavior in LHC is the uncoupled motion meaning that the transverse planes are independent. It is important to control the coupling since it disturbs the tune feedback and can push tunes into resonances [48].

3.4 Magnets Imperfections

3.4.1 Misalignments of Magnets

The movement of a quadrupole magnet by just a few micrometers can have a measurable effect on the beam. Quadrupole alignment errors are usually a significant source of closed orbit distortion in a storage ring. In the case of a vertically moved quadrupole, a particle moving along the reference trajectory sees a horizontal magnetic field. This causes a vertical deflection to the particle and the closed orbit is no longer exactly along the reference trajectory [47].

3.4.2 Magnetic Field Imperfections

Imperfections in the real accelerator appear in case of uncertainties or errors in magnets, power converter regulations, beam momentum, or Radio Frequency (RF) fluctuations. A large deviation from the model of the injection optics causes an increased emittance. When the beam emittance is too large, the betatron oscillations could cause losses, limiting their amplitude. In following a brief overview about the imperfections caused by the magnetic fields perturbations is given.

Dipole perturbation: closed orbit distortion

The closed orbit in a storage ring is defined as the trajectory that has periodicity equal to the turn. A magnetic dipole field causes the change in particle momentum and the closed orbit is no longer the reference trajectory. To be noticed is that the closed orbit at any point on a storage ring depends on the beta function at that point, as well as on the beta function at the location of the dipole field error: if the beta function is large, then a small deflection can lead to a large closed orbit distortion [47]. 3.3 describes the closed orbit resulting from distributed dipole perturbations θ_i

$$CO(s) = \frac{\sqrt{\beta(s)}}{2\sin\pi Q}\sum_i \sqrt{\beta_i}\theta_i \cos(\pi Q - |\phi(s) - \phi_i|) \tag{3.6}$$

where Q is the tune, $\beta(s)$ the β - function at the location s and $|\phi(s) - \phi_i|$ phase advance between the point s and the location of the dipole i.

Quadrupole perturbation: focusing errors

Focusing errors in a storage ring can easily arise from variations in the current flowing through the coils in the quadrupoles. The focusing errors lead to changes in the β -functions:

$$\frac{\Delta\beta(s)}{\beta} \approx \sum_i \frac{\Delta k_i \beta_i}{2\sin(2\pi Q)} \cos(2\pi Q - 2|\phi(s) - \phi_i|) \tag{3.7}$$

and the tunes:

$$\Delta Q_x \approx \sum_i \frac{1}{4\pi} \beta_{ix} \Delta k_i \qquad \Delta Q_y \approx \sum_i \frac{1}{4\pi} \beta_{iy} \Delta k_i, \tag{3.8}$$

where Δk is quadrupole strength error at the ith error source. Thus, the change in betatron tune depends on the change in focusing strength and on the value on the β -function at the location on the error.

Skew quadrupole perturbation: coupling

Coupling errors result from unwanted skew quadrupoles. A particle passing through a skew quadrupole experiences a horizontal deflection proportional to its vertical position. Although the ring in the absence of coupling is tuned so that the betatron tunes are the same, the presence of a skew quadrupole field leads to a split in the normal mode tunes [47].

Identifying and correcting the significant sources of machine errors is a major task during the commissioning of accelerator and machine developments. The good control of the optics in the LHC was a large contribution to the success in exploration of a new energy scale and important discoveries. In the next section the methods to measure and correct the optics of the LHC are discussed.

4 Optics Measurements and Corrections

Appropriate algorithms are required to identify sources of undesired beam behavior. The methods to measure and correct the beam optics in the machine as well as the dedicated software tools are discussed in this section.

4.1 Measurements

4.1.1 Exciting the Beam

In order to measure the optics, an excitation of the beam is needed. In the LHC the beams are excited with kicker magnets, usually an AC-Dipole is used to perform the 'kicks'. The AC-dipole is a fast oscillating magnet, which can be adiabatically turned on and off [49]. In this way it creates coherent oscillations of the beam particles without affecting the transverse emittance.

During the kick a distortion of the closed orbit is introduced, which allows to measure the oscillations [47]. The amplitude of the kick needs to be high enough to produce oscillations which can be recorded by Beam Position Monitors (BPMs). The BPMs record a sample of the beam position every turn (turn-by-turn data), that is used to reconstruct the optics by performing a spectral analysis on turn-by-turn data.

4.1.2 Optics Measurements

Measuring the β -function

The phase of measured betatron oscillation can be inferred from a harmonic analysis of the turn-by-turn data at BPMs. Using the three BPM method [50] through the phase advance between 3 consecutive BPMs the β in the position of the first BPM can be computed as follows:

$$\beta_1 = \frac{\left(\frac{1}{\tan\phi_{21}} - \frac{1}{\tan\phi_{31}}\right)}{\left(\frac{m_{11}}{m_{12}} - \frac{n_{11}}{n_{12}}\right)} \tag{4.1}$$

where the ϕ_{ij} is the phase advance between BPM_i and BPM_j and m_{ij}, n_{ij} the elements of the transfer matrix which are given by a local model between two BPMs. The graphical representation of the three BPM method is shown in Figure 4.1. The accuracy of described method depends not only on the knowledge of the optics model and the precision of the measured phase but also on the value of the phase advance between the BPMs.

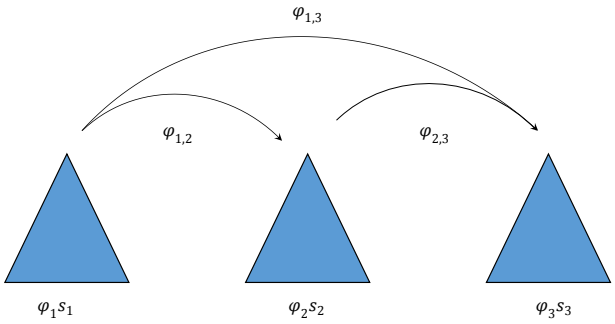

Figure 4.1: Illustration of the β -function measurement from phase.

The N-BPM method [51] allows to use more BPM combinations from a larger range of BPMs to increase the amount of information that is used in the measurement of β -function. A number of N BPMs is chosen close to the probed BPM as shown in Figure 4.2. The best estimate of the measured β -function out of M combinations of three BPMs is to be found by performing a least squares minimization of the function

$$S(\beta) = \sum_{i=1}^{m} \sum_{j=1}^{m} (\beta_i - \beta) V_{ij}^{-1} (\beta_j - \beta), \tag{4.2}$$

where β_i are the β - functions obtained from different BPM combinations and V_{ij} are the elements of the covariance matrix for the different β_i.

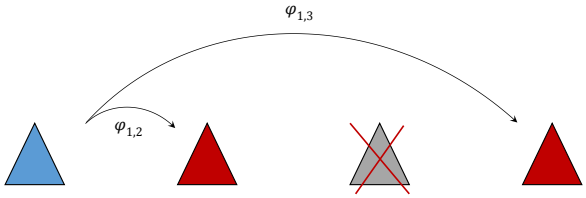

Figure 4.2: The blue BPM is probed, the gray BPM should be skipped and the two red BPMs are included into the measurement in order to obtain the optimum phase advance.

The measured β - function at the probed BPM position is then a weighted average of the m β - functions as shown in 4.3.

$$\beta = \sum_{i=1}^{m} w_i \beta_i \tag{4.3}$$

K-modulation

Measuring the β - function via K-modulation is based on the strength change of individually powered quadrupoles and the resulting tune change [8, 52]. This method is model

independent and offers an alternative for the measurements of β^* (β in the IPs). Moreover the method is used to obtain the β - function close to the quadrupoles in LHC Point 4.

Changing the strength of a quadrupole leads to a tune change corresponding to the change of strength and the average β - function in the quadrupole [52]. The β - function is calculated from the change in quadrupole strength and the change in tune following the formula

$$\beta = \frac{2}{l\Delta k}\left[\cot(2\pi Q) - \frac{\cos(2\pi(Q+\Delta Q))}{\sin(2\pi Q)}\right] \tag{4.4}$$

where l is the length of the quadrupole, Δk the strength change, ΔQ the tune change and Q the nominal tune.

4.2 Correction Methods

The correction methods can be divided into two types - global and local. The global correction is used to make a global fit around the ring and it is computed through a response matrix using the ideal model of the machine [4]. The perturbation of the optics functions at m BPMs due to the change in strength of n correctors is formulated into $m \times n$ response matrix ($\mathbf{R}$). This method is discussed in more detail in section 4.3.3.

The identification of local error sources and finding the correction is based on the Segment-by-Segment technique [53] and is used mostly around the IPs. The main idea of this technique is to consider a part of accelerator as beam line (see section 4.3.3). The optics parameters at the location of the segment are taken from the measurement and propagated with the ideal model and compared to the measurements. By comparing the propagated model with the measured optical parameters in the beam line local deviations could be observed. Corrections are then computed and applied to individually powered magnets.

4.3 OMC Software

The OMC Software is a large software package where many computer scientists and physicists have contributed. It consists of several analysis tools and Graphical User Interfaces (GUIs). The purpose of the software is to analyze data measured during the LHC operation and compute optics corrections to maximize the performance of the LHC. Basically the data is recorded during dedicated optics measurement sessions and the analysis codes are used to provide results online. In order to compute the corrections, the measured data has to be compared with the design machine. The OMC Software tools are presented in the this section with the focus on the Beta-Beating GUI and the Multiturn application.

4.3.1 Beta-Beating GUI

The Beta-Beating application offers a graphical representation of analysed data together with the ability to call external scripts in order to analyze the turn-by-turn data and compute optics corrections. The main goal of the Beta-Beating application is to allow fast and reliable evaluations of the optics errors and provide the user all the information and tools to compute corrections [1]. Figure 4.3 shows the common use case of the data analysis using the Beta-Beating GUI and external programs.

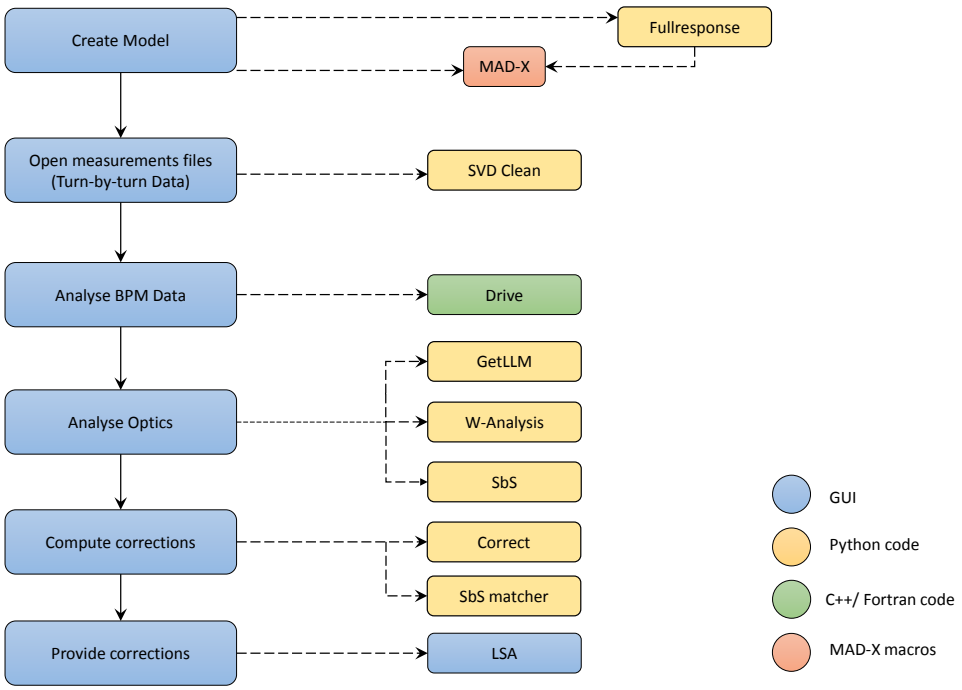

Figure 4.3: Program flow and connection between the Beta-Beating GUI and external scripts.

Besides the analysis of data taken on LHC the Beta-Beating GUI supports the handling of data from other accelerators such as SPS, PS Booster and RHIC. In the following the discussion focuses on the usage at LHC.

4.3.2 GUI Functionality

As the environment for Graphical User Interfaces used in CERN Control Center is predominant based on Java, the Beta-Beating GUI was programmed in Java too [1, 54]. The advantages of Java concerning the development of Graphical User Interfaces are the wide usage and code base, the platform independence, reliable libraries and simplicity of its use. The development was started in 2010 [54] and is being continued since then by contributions of OMC Team members. The GUI contains multiple views for the specific analysis steps. Start screen allows to choose the accelerator, the output and input directories for the data analysis as well as the path to the repository where the analysis

scripts are stored. It has to be mentioned that in the OMC Software the two LHC Beams are considered as two different accelerators due to the fact that the data obtained from the different beams has to be analyzed independently.

Model creation

The creation of the design model is a crucial step since it described the desired optics in the machine and is needed to compute corrections. External analysis and correction scripts use the model as input parameter. The model is a file generated by MAD-X [55] including the calculation of the response matrix for correction quadrupoles. The calculation of the response matrix and the model generation is presented more detailed in the section 4.3.3. The GUI allows to input all the needed parameters to create a model such tunes of the machine, optics settings and energy.

Data representation

Once a model has been created, the turn-by-turn data can be loaded into the GUI to show the graphical representation of the data. The GUI supports the loading of raw measurement files and provides information about the loaded data inside the BPM panel. The charts can display the turn number to the values for every BPM from the list or display the phase space as shown in Figure 4.4.

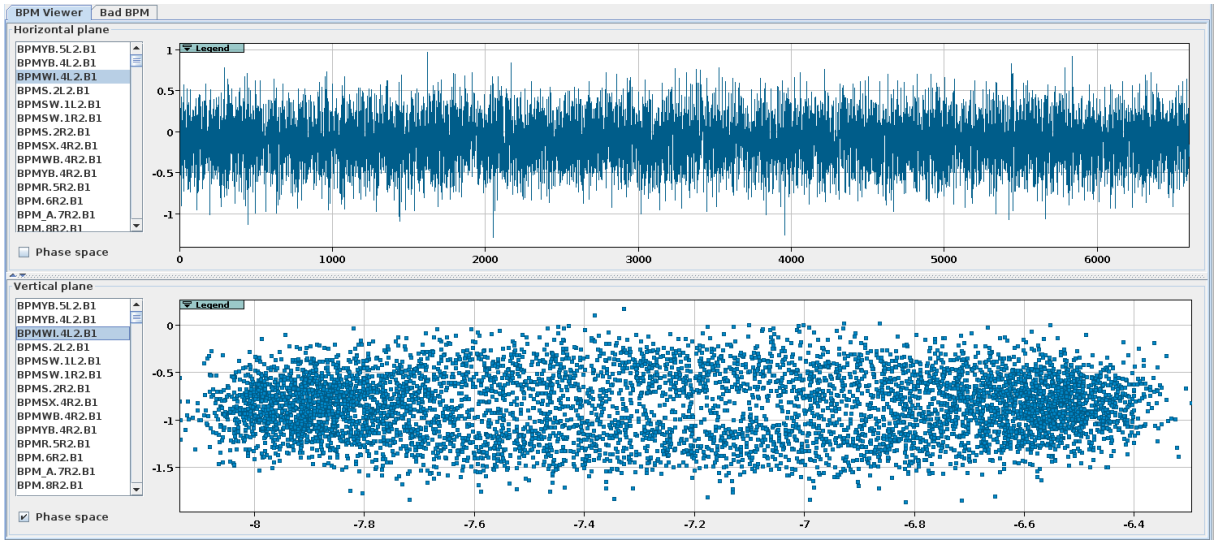

Figure 4.4: Graphical representation of a measurement.

Data analysis

In the next step the loaded, converted and cleaned data will be analyzed by Drive, which computes a refined Fourier transformation and writes frequencies, phases and amplitudes for all BPMs to output files in Table File System (TFS) format [56]. Analysis results are

shown in Figure 4.5: every column of a TFS file can be displayed in the chart, the given example is showing the horizontal and vertical tune values.

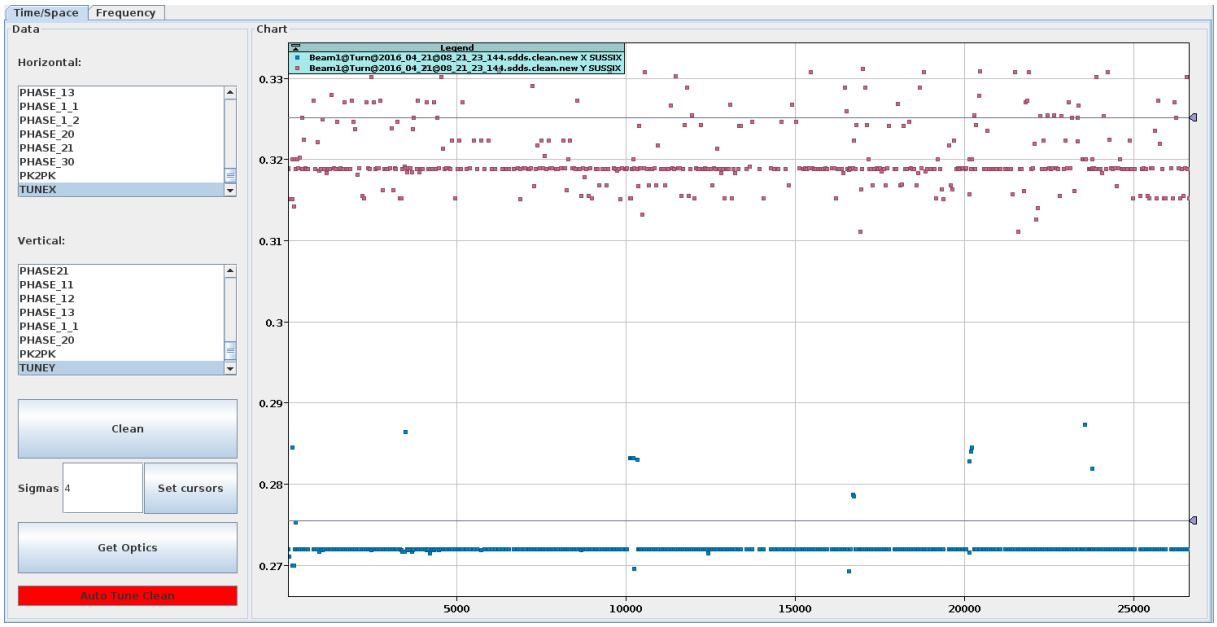

Figure 4.5: Graphical representation of a loaded measurement file before and after analysis with Drive.

The main components of the Analysis panel is the table containing file names and the relative momentum deviation ($\Delta p/p$) computed for each of files inside one of external scripts and the graphical representation of the analyzed data in form of interactive charts plotting the columns values of the analysis output files for each BPM. Different properties such tunes, peak values, amplitude and phase can be selected to be displayed in the charts. The interactive charts allow user to zoom and to clean the data manually using the limit cursors.

From this panel one can start the optics calculation using GetLLM (Get Linear Lattice function and More) for the analysis of on-momentum measurements or W-Analysis to compute the Montague functions [57] and chromatic coupling [58]. The results are presented in the Optics Panel which displays different measurements compared to each other or current measurement compared to the designed model as shown in Figure 4.6.

Corrections

The correction button from the optics panel can be used to calculate the optics corrections. The measured machine parameters have to be compared to the design model of the machine in order to identify differences that must be corrected.

The results are shown in the Correction Panel representing the needed strength of the correction magnets to correct the measured errors. The Knob Panel inside the Correction Panel allows the user to upload the corrections to LSA which is a high-level layer of the

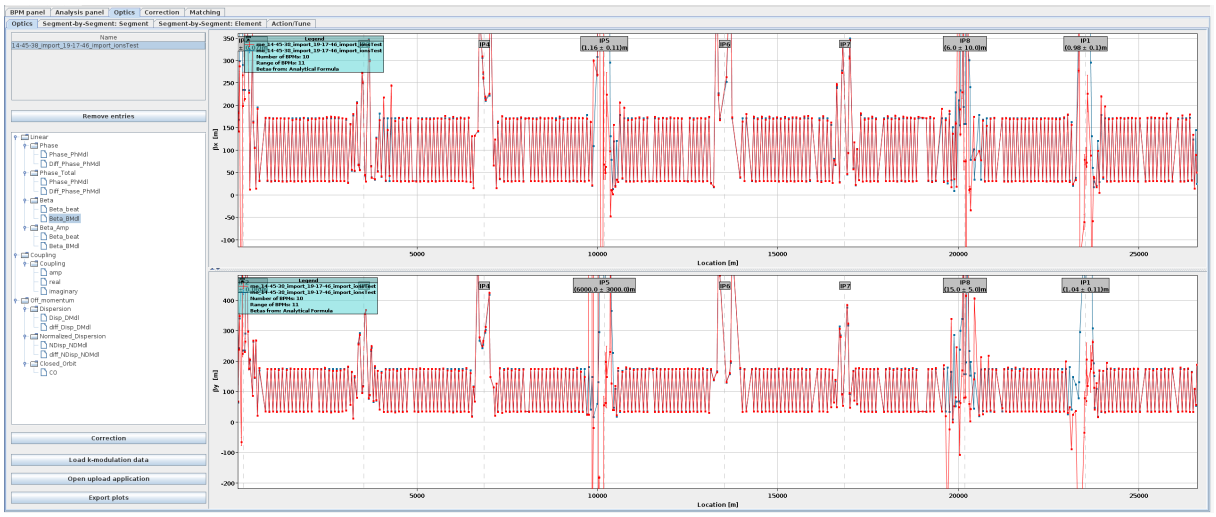

Figure 4.6: Optics panel showing the the measured beta-beating compared to model. On the left side different optics functions can be selected to be displayed in the chart.

LHC control system [59]. The corrections are applied then to the physical machine.

4.3.3 External programs

The following programs represent the core of the data analysis and correction computation. The understanding of the analysis concepts is crucial for the further research on the applications of machine learning methods.

Model creation and Fullresponse

The model creation is mainly implemented in Java by calling MAD-X using the settings defined by user input in the Beta-Beating GUI.

The Python implementation of the fullresponse computation produces the responses of the beta, phase and horizontal and vertical dispersion on the quadrupole strengths. In addition corresponding calculation for the coupling and the vertical dispersion is performed. These response matrices are used to calculate the global corrections.

The Fullresponse code is based on computation of the response matrix R that relates the signals measured at BPMs to changes in the corrector magnets strengths $\vec{k}_1$. It has to be noted though that we consider the strength in the quadrupole circuits and each circuit represents either a set of quadrupoles powered in series or an individually powered corrector quadrupole. The response matrix is computed using the ideal model of the machine providing the relation between phase-beating, beta-beating, dispersion-beating

and tune-beating and changes in the quadrupole strength as follow:

$$\vec{R_i} = \left(\frac{\sqrt{w_{\phi_{x,y}}} \cdot \frac{d\vec{\phi_{x,y}}}{dk_i}}{\vec{\sigma_{\phi_{x,y}}}}, \frac{\sqrt{w_{\beta_{x,y}}} \cdot \frac{d\vec{\beta_{x,y}}}{dk_i}}{\vec{\sigma_{\beta_{x,y}}}}, \frac{\sqrt{w_{ND_x}} \cdot \frac{d\vec{ND_x}}{dk_i}}{\vec{\sigma_{ND_x}}}, \frac{\sqrt{w_Q} \cdot \frac{dQ_{x,y}}{dk_i}}{\vec{\sigma_{Q_{x,y}}}} \right)^T \tag{4.5}$$

Furthermore the method allows the specification of quantity specific weight. Thus the strengths of the quadrupoles needed to perform the corrections can be computed as

$$\Delta\vec{k_1} = -R^{-1} \cdot \left(\sqrt{w_{\phi_{x,y}}} \left(\frac{\Delta\vec{\phi_{x,y}}}{\vec{\sigma_{\phi_{x,y}}}} \right), \sqrt{w_{\beta_{x,y}}} \left(\frac{\Delta\vec{\beta_{x,y}}}{\sigma_{\beta_{x,y}}} \right), \sqrt{w_{ND_x}} \left(\frac{\Delta\vec{ND_x}}{\sigma_{ND_x}} \right), \sqrt{w_Q} \frac{\Delta Q_{x,y}}{\sigma_{Q_{x,y}}} \right)^T \tag{4.6}$$

where $w_{\beta_{x,y}}$, $w_{\phi_{x,y}}$, $w_{Q_{x,y}}$, w_{ND_x} are the quantity specific weights. In the improved method the measurement uncertainties are taken into account as weights [4].

SVD Clean

SVD Clean is responsible for removing noise and bad BPMs from the turn-by-turn data files using the Singular Value Decomposition [60]. Singular Value Decomposition (SVD) is widely used in signal processing and statistics and is defined mathematically for a $m \times n$ matrix as:

$$M = U\Sigma V^T \tag{4.7}$$

In this definition U is an $m \times n$ matrix. Σ is an $T \times n$ diagonal matrix containing non-negative real numbers, which are the singular values. V^T is $n \times n$ matrix, defined as the conjugate transposed of V. In our case U has the size of $B \times n$ with B the number of BPMs and V the size $n \times T$, where T is the number of turns. Thus the transposed matrix U^T has the size $n \times B$ with rows containing the BPMs values, so SVD can be used to identify the principle components by maximizing the cross covariance between data. Together with identified faulty BPMs using the SVD technique, the SVD Clean program removes also the BPMs classified as "bad" based on following reasons:

- The measured signal contains exact zeros
- Spikes in the measured signal
- BPM with the same value for all turns (flat signal)

The identified faulty BPMs are written to an ASCII file together with known bad BPMs. Another ASCII output file contains the rest of the BPMs (*"good"* BPMS) which will be used as input for further analysis.

Drive

The methodology of analyzing the turn-by-turn data includes a Fourier analysis which allows to obtain the main frequencies of the measured signal. The analysis code called SUSSIX was initially implemented in FORTRAN77 [61]. The computation of Fourier transformation and writing of frequencies, phases and amplitudes is done with help of the wrapper written in C++ for the SUSSIX code called Drive.

By subtracting from the measured signal the identified amplitude peak in the spectral decomposition line which is in most cases the tune, we obtain a new signal of equal length which can be re-analyzed in the same way. This iterative procedure of identification and subtraction of the amplitude maximum provides the set of frequencies contained in the turn-by-turn data.

The execution of Drive can be started by calling it from the BPM Panel in the Beta Beating GUI. The Drive input is converted and cleaned by the SVD ASCII files. The output of Drive is given in TFS format, the results are displayed in the Analysis Panel.

Optics analysis

The optics analysis script called GetLLM (Get Linear Lattice function and More) contains the calculation of different optics functions described in 4.1.2 using the output of Drive. The results are written to files in TFS format and are automatically shown in the Beta-Beating GUI.

Segment-by-Segment

The Segment-by-Segment (SbS) technique [62] is used to correct the optics at regions of importance as interaction points through the identification of phase advance beating [2]. Another purpose of SbS is the propagation of optics observables from the BPM position to other elements, e.g. to obtain the β^* at IPs [63].

The main idea of this technique is to divide the machine in a certain number of segments and treat them as beam lines. The phase advance is propagated between the BPMs using the reference model. The deviations found in the propagation are easier to correct taking into account the smaller segments than for the entire machine.

SbS program includes a calculation of correction of the phase advance between BPMs in a given segment. Using MAD-X, the phase deviation at each measurement point are matched using the least square minimization method on

$$\Delta\phi_{n+1} = \phi_{n+1}^{meas} - \phi_n^{meas} - (\phi_{n+1}^{mod} - \phi_n^{mod}) \tag{4.8}$$

representing the deviations in the phase advance ϕ between each BPM in the region of interest. After the phase deviation is reproduced with target accuracy, the sign of the

resulting magnet strength will be flipped and incorporated in the machine to eliminate this deviation.

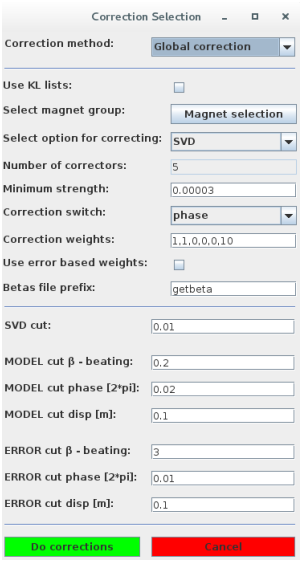

Figure 4.7: The input panel to specify the desired parameters for correction scripts.

Correction

The correction scripts use the output of GetLLM and Fullresponse to calculate correction strengths for the magnets. Local corrections are best suited for the IRs where the β -functions are large and there are independently powered quadrupoles [64]. First local errors are identified and corrected using segment-by-segment. When the optics is corrected to a level where there is no dominant error source left in the machine, a global correction approach can be applied to minimize the optics errors in the arcs. The global correction is based on response matrix technique. β-beat and horizontal dispersion is corrected using quadrupole magnets. Skew quads are used to correct coupling. As soon as correction strengths are computed for the selected groups of magnets with given parameters (see Figure 4.7), the corrections can be provided into the machine using the LSA service to generate the knobs, which is a list containing the correctors and their delta values.

5 Improvements towards Machine Learning Application

Before the implementation of the machine learning concepts can be started, the OMC software required several improvements in terms of automation and increasing the data abstraction level. In the following section the improvements to the Beta-Beating GUI and the Multiturn application are presented.

5.1 Beta-Beating GUI Developments

The observation of the measurement process demonstrated that the measurements are taken with high manual effort. The essential tasks which can be automatized have been identified and improved aiming a higher abstraction level of analysis data and output, as well as the automation of the measurement process.

5.1.1 Automatic chart zoom

The Beta-Beating GUI offers interactive charts to represent the data. As the manual zooming and data cleaning is time and human effort expensive, automatic methods had to be implemented to perform these methods.

The new implementation offers the zooming in steps, which are calculated based on deviation from the mean of the data to be displayed. In the first step the outliers are removed from the data to compute the trimmed mean. Then the mean and the standard deviation from the raw data are computed. The method implemented to reject the outliers from the raw data iterates through the samples and removes the values outside of the range between $\mu_{raw} + 3\sigma_{raw}$ and $\mu_{raw} - 3\sigma_{raw}$, where μ_{raw} and σ_{raw} are the average and the standard deviation of the uncleaned data. The 3σ tolerance window was introduced due to the assumption that data is normally distributed and according to the three-sigma-rule which considers an event as unlikely or insignificant if it lies in the region of values at a distance from the mean more than three times the standard deviation [65].

Then the upper and lower limits for the data to be displayed in the zoomed chart are computed as follows:

$$\begin{aligned} upper &= \mu_{trimmed} + n \times \sigma_{trimmed} \\ lower &= \mu_{trimmed} - n \times \sigma_{trimmed}, \\ n &\in \{3, 2, 1\}, \end{aligned} \tag{5.1}$$

Figure 5.1: The initial graphical representation of comparison between measured and modeled β -function in horizontal plane before and after applying automatic zoom. The important values can be observed and understand better with help of automatic zoom.

where $\mu_{trimmed}$ is the average and $\sigma_{trimmed}$ the standard deviation of the data without outliers.

The zooming can be performed by clicking on the chart frame. In first steps the chart will be zoomed to the range of 3σ , repeated clicks trigger the zooming to display the data in the range of 2σ and 1σ (see Figure 5.1). To zoom out in the steps of the same size, the user just has to press the right mouse button. The charts can be displayed for all the measured optics observables computed by GetLLM (see 4.3.3) including phase advance, β -function, dispersion and coupling. The measured functions can be compared to the dedicated model or to a different measurement. Hence the user can observe a wide range of charts types and modes where the zooming has to be performed in order to gather needed information from the displayed data. Considering this fact, the automatic zooming saves a significant amount of time by reducing the number of required user actions.

5.1.2 Noise reduction

The turn-by-turn data obtained during the measurements can contain values which are distant from the other observations even after applying the SVD technique to remove the noise from the measurement. These values can be specified as *outliers*. As defined in [66], an outlying observation is one that appears to deviate markedly from other members of the sample in which it occurs. In a sense, the definition of "markedly deviation" remains open and the decision about the reasonable specification is left to the user (or an automatic process).

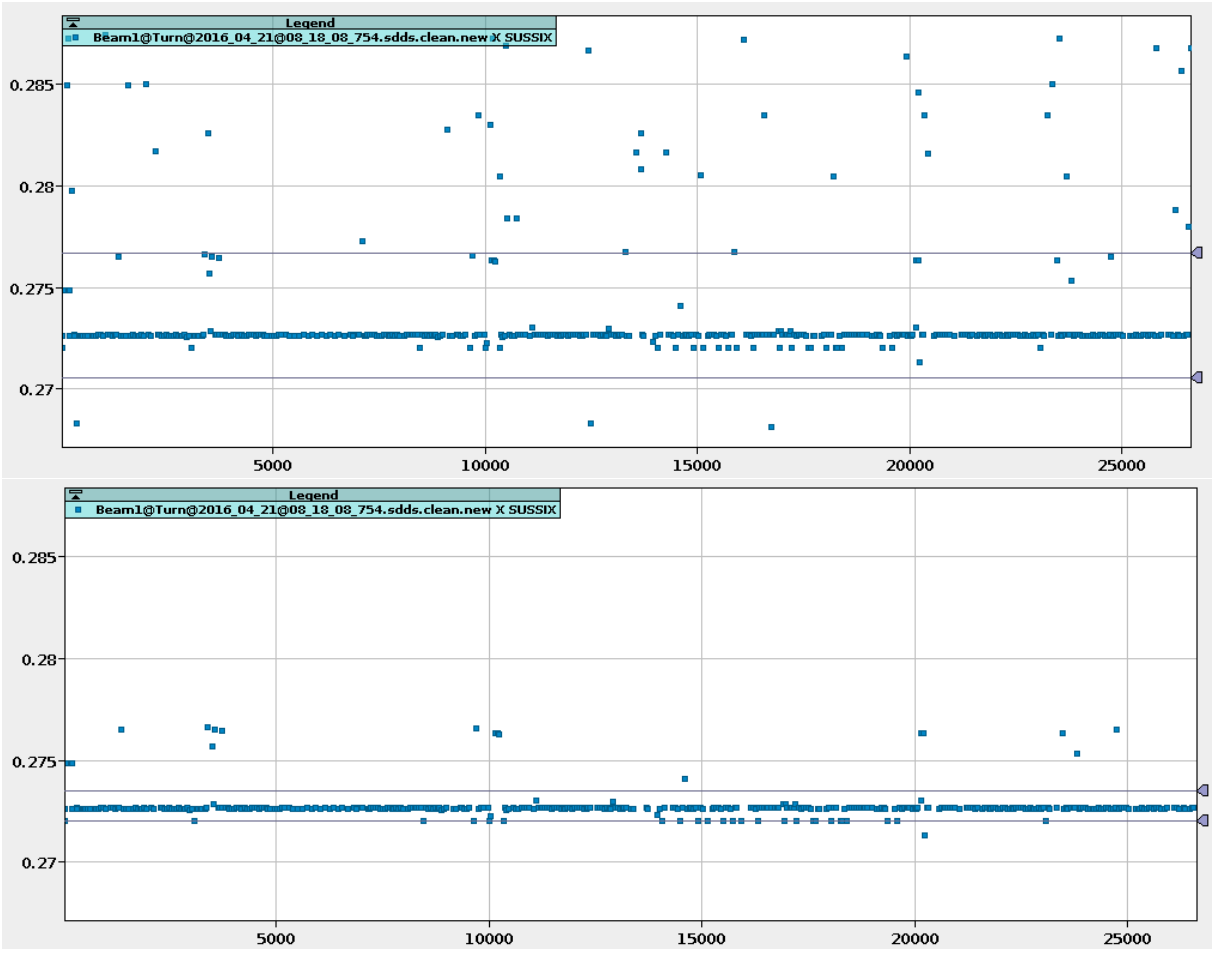

Figure 5.2: The top chart shows the uncleaned horizontal tune values with the cursor set automatically to 1σ. The bottom chart represents the cleaned data with the suggestion for the next cleaning step.

Considering the turn-by-turn data, the outliers have to be identified and removed before triggering the optics analysis. Since the analysis and the following correction computation aim high precision level, the outliers can affect the results significantly. The Analysis Panel in the Beta-Beating GUI offers a functionality to remove unreasonable values from the displayed charts manually. In this case the user can select between different measured

properties as e.g. tune, phase, orbit, peak-to-peak values and display the values of selected parameter for each BPM in the charts.

In the past, the user had to decide which values belong to outliers and remove them by locating the cursor in the chart to specify the cleaning cuts. The new functionality enables the automatic calculation of the cleaning cuts corresponding to desired deviation set by user. Thus, the user can determine the domain of acceptance by giving a number of sigmas and the data will be cleaned automatically without further user interaction to find the outliers.

The desired cleaning cuts are computed using the σ value set by the user. As the data presented in the chart will be used for correction computation it is crucial for the user to make sure that cleaning meets further analysis needs. Therefore the automatically computed cleaning limit will be first presented to the user by setting the limit cursors. In the next step the user can correct the cursor position or immediately click on the "Clean"-button which will trigger the removing of outliers.

5.1.3 Automatic computing of relative momentum deviation

The Beta-Bating GUI offers analysis methods for different types of data – data acquired on- or off-momentum. For the analysis of off-momentum data calculation of the relative momentum deviation ($\Delta p/p$) is needed.

The current $\Delta p/p$ value of the measurements files is computed inside the SVD clean script and it is written to the header of the converted and cleaned file. After the data have been analyzed with Drive, the $\Delta p/p$ value of each file will be shown in the Analysis panel. Before the introduction of the improvements, these values had to be grouped manually and the $\Delta p/p$ value to use for further optics analysis had to be calculated by user and put into the file table manually. The procedure consisted of the following steps:

1. Identify the $\Delta p/p$ values closest to zero
2. Calculate the mean of $\Delta p/p$ closest to zero
3. Make groups with $\Delta p/p$ with similar values
4. Calculate the mean of each group
5. For each group: Subtract the $\Delta p/p$ from the mean of the group
6. Put the results of subtraction to each file row in the table corresponding to the $\Delta p/p$ group.

Performing this procedure manually is not a trivial task and costs a significant amount of time and human effort. Moreover it has a potential to lead to mistakes. The manual procedure has been replaced with the fully automatic calculation of the relative momentum

deviation per file which can be triggered by pressing "Arrange $\Delta p/p$" – button in the Analysis Panel. Thus the manual calculation could be avoided to reduce the data analysis time and possibility of mistakes.

5.1.4 Further improvements to Beta-Beating GUI

Since the purpose of this chapter is to present the improvements needed for the future application of machine learning methods, following improvements that have been introduced among with the developments described above will not be presented in detail:

- Automatic test for the optics analysis tool "GetLLM"
- Set up of the commit control system using Travis CI [67]
- Plot export from the interactive charts in the Beta-Beating GUI
- Adaptation of the Beta-Beating GUI to enable the call of new functionalities of external programs
- Enable to load measurement or simulation files from other accelerators
- Requested bug fixes

Thus the Beta-Beating GUI was improved by reducing the human effort and potential mistakes risk via using basic statistics methods in order to automatize the tasks.

Improvements to K-modulation application

The K-modulation application [8,52] was improved in terms of extension of data extraction functionality. The K-modulation application offers precise measurement of β -function by changing the strength of quadrupoles and measuring the corresponding tune change as described in 4.1.2.

The K-modulation application allows the user to extract the tune change measurements data at specified region. This functionality was extended in order to extract the orbit data, which was needed to obtain the data for measuring misalignments and crossing angles. The results of this study are presented in [68].

Besides discussed changes, the improvements towards machine learning application have been introduced to the Multiturn application. The following section presents the achieved results and describes additional developments introduced to OMC Software in terms of automation and communication between different systems.

5.2 Automatic Coupling Correction

In this section a new software development for automatic coupling correction and automatic model generation for data acquisition extending functionality of the Multiturn application is presented. The section is based on the Machine Development Note 1988 published on February 2017 [69].

5.2.1 Coupling correction using AC-Dipole

Global coupling correction in the LHC has been based on three different methods: observing the BBQ (diode-based base-band-tune System) and test different settings of the coupling knobs [70], based on the injection oscillations to automatic calculate a correction [71] and corrections based on measurements from the excitation of the AC-dipole. The method relying on the BBQ is time consuming and has been observed to not be fully reliable, in particular for small beta star. The injection oscillations method works well but it is intrinsically limited to injection. The coupling corrections with the AC-dipole has been demonstrated to reduce the coupling to very low values [72].

The coupling correction using the AC-dipole required several steps between the acquisition of turn-by-turn data and obtaining the correction itself and two tools were required: the Multiturn application and the Beta-Beating GUI calling the external programs.

In the previous version of the Multiturn application, the turn-by-turn data acquisition produced only a binary file as output. The new implementation provides the data in ASCII format needed to perform the analysis.

The new functionality of the Multiturn application provides online automatic coupling correction on data acquisition along with full optics analysis. In order to enable this functionality, further improvements to the Multiturn application were required and therefore presented in this chapter.

5.2.2 Model generation

One of the requirements to enable automatic coupling correction computation is the presence of model files in order to run the analyzes on the measurement file.

The investigation on the codes capabilities showed that the automatic model generation for the current LHC optics settings should be possible since all required information such tunes values, current optics and energy settings can be obtained from the user input and using the online model [73]. The Figure 5.3 shows the Ac-dipole panel containing the fields for user input of tunes and model directory. The online model allows to obtain the integer part of the tune values and the energy settings in the machine, since they are stored in a measurement database and can be extracted using a Java API.

The model generation is handled in the same way as described in the section 4.3.3. The complication concerning obtaining the current optics settings in the machine has

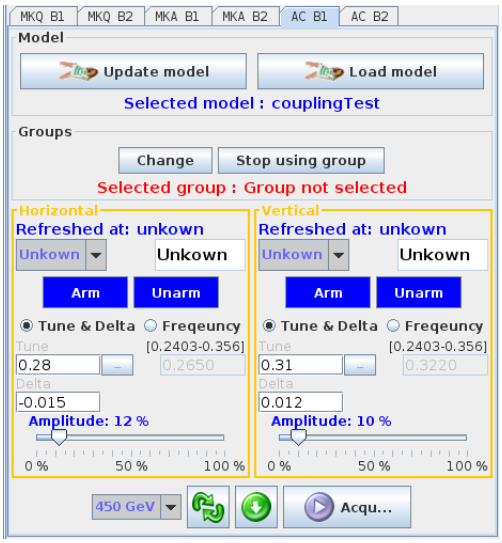

Figure 5.3: The AC-Dipole excitation panel for beam 1. The tunes and the model directory can be obtained directly from the panel.

been solved using the Model Extractor application. The Model Extractor provides a file containing the optics file name and the energy settings. This information is used to run required MAD-X scripts corresponding to optics settings and to select a correction file for the current energy.

As the target of the development is to compute the coupling correction, the Fullresponse has to be computed only for specific magnet groups which are used to correct the coupling. The external program Fullresponse enables to select an option to compute the Response matrix only for coupling magnets, so the running time is reduced significantly. The automatically created model can be loaded into the Beta-Beating GUI and used together with obtained measurement files for the further analysis.

Since the presence of a model is a crucial requirement to obtain the automatic coupling correction an additional functionality to provide a model had to be developed. The manually created model with help of the Beta-Beating GUI can be used for analyzes running from the Multiturn application as well. The directory where the model files created by the Beta-Beating GUI are stored can be easily selected in the AC-Dipole excitation panel.

5.3 Automation in Measurement Process

5.3.1 Measurements grouping

One of the most important steps of the development towards the application of machine learning methods on OMC Software is the automation of the measurement process. The manual effort has to be reduced and the obtained measurements have to be structured in order to simplify the automatic file analysis and correction prediction in the future.

In order to provide more structured data representation and to enable higher process automation level a concept of measurements grouping has been created. The concept of separated groups of measurements is needed due to the fact that during a typical measurement session several data acquisitions are performed.

On one side, multiple acquisitions are needed to improve the precision of measured optics observables by providing larger input to analyzes. Moreover, the measurements are performed to obtain different types of data such off-momentum data acquisition with different deviation of relative momentum. The measurement files have to be grouped considering the specific properties of data and the purpose of the measurement.

The central subject of this concept is called "Kick Group" which is represented by a TFS file containing the information about the measurements obtained on AC- Dipole excitations. The measurements are grouped by the user by selecting the option "Add measurement to kick group" after the measurements file appears in the current LHC fill directory.

Grouping files

The Grouping files describe the properties of the Kick Group and contain the information about the single measurements added to this group. The file format has to provide a clear structure and a possibility to easily obtain the needed information. As the TFS format is used for output files and the infrastructure to handle the files in that format is already partially provided, it is reasonable to use TFS format as well for the Grouping files.

The Beta-Beating GUI code includes a class to read the TFS files. However the writing of TFS files was not implemented in Java. Since until now only the output of external programs has been written in the TFS format, a TFS-Writer is provided inside of the external python scripts package. Based on the Python TFS-Writer implementation, a TFS-Writer was written in Java and included into the Beta-Beating GUI in order to write the Grouping Files and extract the needed information about the grouped measurements.

The header of the Grouping File contains the information related to a measurement session such beam name, date, chosen model and current optics settings. The table inside the file specifies the information about single measurement obtained on AC-Dipole excitation. In order to ease the further analysis and observation of the measurements, it was decided to include the parameters which are described in Table 5.1. As examples the values from optics commissioning on 25th of May 2017 are taken.

These parameters build the columns of the table. After every excitation a new row will be added to the table in case the measurement was accepted by user.

The introduction of Kick Grouping concept changed the measurement process from the user point of view. The initial process is presented in Figure 5.4.

The main goal of improvements of the measurement process is to automatize the loading of the measurements files into the Beta-Beating GUI, to establish communication between

Parameter	Description	Example
FILE	Name of the measurement file	Beam1@Turn@2017_05_25@19_10_34_547.sdds
FILL	Number of LHC Fill	5708
QX, QY	Horizontal and vertical tune	0.31 0.32
QDX, QDY	Driven horizontal and vertical tune	0.30 0.33
AMPX, AMPY	Amplitude of the excitation per plane	26.0% 23.0%
PK2PKX, PK2PKY	Difference between the maximum positive and the maximum negative per plane	1.4153mm 1.6013mm
DPP	$\Delta p/p$ value	-3.3084e-06
REAL, IMAG	Computed real and imaginary values for the coupling correction	-0.0021 -0.0039

Table 5.1: Parameters of a single measurement that are written to the Grouping File.

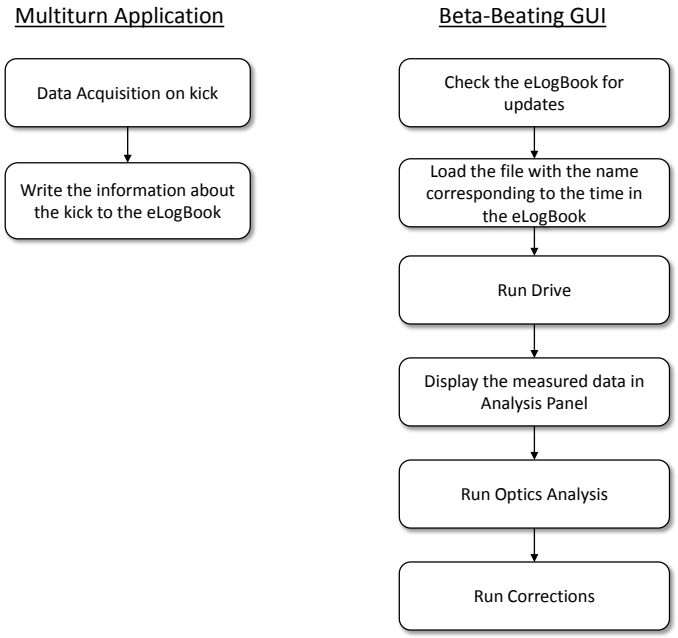

Figure 5.4: The schematic representation of the measurement process before introduction of Kick Grouping and automatic file loading to the Beta-Beating GUI.

involved applications and to reduce the time between data acquisition and correction. The implementation of the Kick Grouping concept and including the computation of coupling correction into the Multiturn application allowed to reduce the needed manual effort significantly as shown in Figure 5.5.

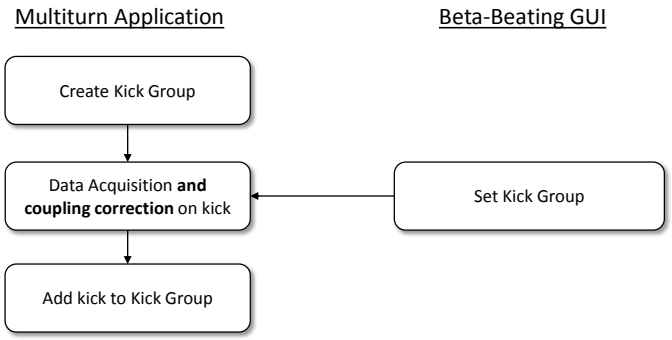

Figure 5.5: New improved process shows that the number of the manual steps performed by the user decreased significantly.

5.3.2 Communication between OMC Applications

The communication between Beta-Beating GUI and Multiturn Application was achieved by implementation of monitoring of the selected Group File. The loading of measurement file is triggered when the size of the file corresponding to the selected group name has changed. The measurement files specified by name will be loaded from the measurement directory with the Fill number given in the TFS table in the grouping file. The Fill number is required in order to reconstruct the physical link to the measurement directory. The file and other information about the kick will be loaded into the Beta-Beating GUI including the group name as it will be needed to select the files which belong to one group for the optics analysis.

During a session different measurements might have to be taken. In this case the active Kick Group has to be changed. The creation of a new Kick Group has to be handled in the Multiturn Application. In the Beta-Beating GUI the active group can be changed using the Group Search Panel which provides dynamic filtering of Kick groups.

As soon as the measurement data are analyzed, the files are loaded and displayed automatically in the BPM and Analysis panels of the Beta-Beating GUI. Thus, the time expensive searching and loading of the measurements files and running Drive are avoided now, the increase of speed of the measurement process has been achieved. Moreover, the layout of the files tables in Analysis Panel has been changed in order to ease the further step, the optics analysis, by adding the information about the amplitude and the name of the Kick Group.

5.3.3 Automatic measurements logging

In the past, the data acquisition process from the user point of view included manually logging of the AC-dipole kick parameters which are identical to the parameters of a single kick in the Kick Group file (see 5.1). Furthermore, the user had to log the exact time of excitation, since it was necessary in order to find the files and to load them manually into the Beta-Beating GUI. The introduction of the "Kick Group" concept enables to read information about the kicks from the Kick Group files and provide it to the OMC E-Logbook automatically.

Creation of Logbook events is implemented using API which allows other applications to send data to the e-Logbook. The events can be edited after the creation with the help of the *eventID*, that is known since the method to create an event in the e-Logbook *addEvent()* returns the ID after the event has been created.

The editing of the events is needed as it is required to log the information about the performed AC dipole excitation right after the kick data has arrived to the system. Thus, the automatic logging routine creates new event in the OMC Logbook after the creation of a new Kick Group using the Multiturn Application. The Kick Group has a property *eventID*, so every Kick Group is associated with an event in the Logbook. Addition of a measurement file to the Kick Group triggers the request to edit the event with a corresponding *eventID* and the information about the performed kick is written to the Logbook event.

The automatic logging contributes not only to the automation of the measurement process and time effort minimization, but it also provides a structured, abstract representation of the information which can be potentially used to mine the data for predictions using machine learning methods or to collect statistical information about the measurements.

6 Potential Machine Learning Applications

The OMC Software has been improved in terms of automation and simplification of handling the data analyzes to take further steps towards application of appropriate machine learning algorithms. As it could be seen in chapters 4 and chapter 5 several steps of optics measurement and corrections on LHC rely on basic statistical methods such automatic cleaning using the standard deviation to find the cleaning limits or the outliers in optics parameters computed from the measurements. The Singular Value Decomposition which is considered as one of machine learning techniques has found its application in OMC Software in the form of an algorithm to identify BPMs with unacceptable data. Nevertheless, there is room for further improvements and investigation on appropriated data analysis methods considering different aspects of the measurement and correction process.

This chapter focuses on possible applications of machine learning methods dedicated to identification and analysis of faulty BPMs and corrections computing based on previous machine developments. In the following the conceptual solutions and appropriate methods for these problems are presented.

Application of association rules in OMC software

The concept of Association Rules could be applied in optics measurements and corrections for such problems as selection of correction magnets and extracting the common features of BPMs using the Apriori Algorithm.

The Apriori algorithm is an association rule mining algorithm for finding frequent itemsets using candidates generation controlling the candidate itemset growth based on support measure [74]. The algorithm employs a level-wise, breadth-first approach iterating through transactions as described in the pseudocode bellow with set of candidate k-itemsets C_k and set of frequent k - itemsets F_k.

The algorithm first finds all the frequent 1-itemsets in the dataset (steps 1 and 2). Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent (k-1) - itemsets found in the previous iteration (step 5). Using the subset function all the candidate itemsets in C_k that are contained in each transaction t are determined (steps 6-10). After counting their supports, the itemsets whose support does not satisfy the *minsupport* condition will be eliminated. The algorithm runs till no new frequent itemsets are generated.

The idea of association rules application on the transactions database can be applied

Algorithm 1 Frequent itemset generation of the Apriori algorithm

```
1: k = 1
2: F_k = i|i ∈ I ∧ σ(i) ⩾ N × minsup
3: repeat
4:     k = k + 1
5:     C_k = apriori − gen(F_{k−1})
6:     for each t ∈ T do
7:         C_t = subset(C_k, t)
8:         for each candidate itemset c ∈ C_t do
9:             σ(c) = σ(c) + 1
10:        end for
11:    end for
12:    F_k = c|c ∈ C_k ∧ σ(c) ⩾ N × minsup
13: until F_k = ∅
14: Result = ∪F_k
```

to the problem of investigation on faulty BPMs in following way: the database T can be equalized with a dataset of measurements, then the transaction t would be equivalent to the set of BPMs classified as "bad" and the item i_k in the itemset I corresponds to a single BPM. This approach can help to observe the relations between bad BPMs and potentially find the undiscovered reasons of appearance of bad signal from these BPMs. Moreover, the relations between properties of known bad BPMs can be investigated, in this case a BPM would be considered as an itemset and the properties of BPM as items. Based on information that around 10% of all BPMs are bad and assuming that bad BPMs have common properties which could be considered as "frequent itemsets", an alternative method for cleaning of measurement data could be investigated.

To correct the optics parameters the strength of specific magnets has to be changed. Depending on type of the desired correction method (local or global, see 4.2) and machine configuration, specific sets of magnets have to be taken into account. In currently used correction methods the classes of magnets to be used in the correction have to be chosen manually for each correction iteration. Through the analysis of magnets selection in previous corrections data the frequent magnet classes combinations could be found. The magnet classes combination can be considered as itemsets from a transaction dataset as described in Apriori algorithm. Once frequent magnets combination are collected, the same approach as for database transactions can be applied to gain the association rules for magnets selection.

6.1 Signal Classification

One of the problems appearing often in the measurement data are the outliers that indicate bad signal of a BPM. Sometimes they cannot be identified on early stages of analysis such SVD Clean and Drive and appear first in the optics analysis (GetLLM) results. Early

identification of outliers would help to obtain cleaner results and reduce human effort. One of the issues related to outliers identification is to detect an appropriate threshold definition. Another issue regarding the problem of faulty BPMs recognition is the fact that the properties of bad signal are often unknown which makes the identification of faulty BPMs problematic.

The relations in measured turn-by-turn data of a single bad BPM and relations between bad BPMs themselves can be discovered using the Apriori algorithm as discussed above. Moreover, the influence on the classification of a BPM as bad of each observable in the measurement can be investigated using feature analysis techniques [21]. This knowledge can provide more information for investigation on the reasons for faulty signal and help to develop new methods to avoid noise in the measurements data.

Regarding the identification of bad BPMs, one possible approach is to consider the classification as supervised learning problem since the BPMs can be labeled using the results of SVD-Clean analysis from the past measurements. However, in this case the performance of the trained model will be similar to SVD Clean. Therefore, unsupervised learning is a more appropriate approach since the aim is to reduce the number of faulty signal in the data for optics analysis. An *autoencoder* is a possible solution for this problem. It learns to reproduce the data representation of the given input. In the case if the model is trained with good BPM data, a bad BPM will be identified since its data representation will differ from the known good BPMs data. This approach has been prototyped and it is presented in 7.2.

Another solution to identify faulty signals is *clustering*. Defining the problem as a clustering task, the data has to be separated into minimum two clusters - good and faulty signal, which can be identified as outliers and considered as noise. The significant benefit of this approach is the possibility to apply clustering algorithm directly on the measurement data in opposite to the methods described above where a trained model is required in order to make a prediction.

6.2 Correction Prediction

The problem of finding corrections can be considered as a regression problem to fit the design β-function and to minimize beta-beating between measured and design parameters. The central concept of training a neural network is the optimization (trying to find the minimum) of the loss function which represents the error in prediction given by comparison between known and predicted output.

Combining these two ideas, a regression model can be applied to find correction values which result from fitting a given error function to minimize the deviation between design and measured optics of the machine. Based on previous correction computations, a model can learn to find magnetic field errors that have to be eliminated using the computed

deviation from the desired machine design. This approach is presented and evaluated in section 7.1.

Considering the whole measurement and correction process, there is more room for application of machine learning methods. Following problems can be potentially specified as machine learning tasks:

- Prediction of measurements start time using the machine parameters and communications in LHC Page 1
- Prediction of the possibility to dump the beam based on machine parameters, information in LHC Page 1 and automatic entries in OP Logbook.

The pipeline of a possible solution involves the process of structuring the information given as text, deriving patterns within the structured machine parameters data, semantic analysis and more preprocessing steps related specifically to text analysis and to combination of different representations of information. As these problems are not part of the core components of the actual measurement and correction process, they are not nearly investigated by building a prototype. The following chapter focuses on prediction-based correction computation and identification of bad BPMs.

7 Prototyping and Results

7.1 Prediction-based Correction Computation

In order to train a prediction model using a supervised learning approach, a dataset consisting of features and dependent variables is needed. As mentioned in 4.2 the correction of β-beating can be replaced with the phase beating correction since the phase advance correction leads to the same results. Hence, the measured phase deviation from the design is considered as feature (input parameter for the model). The knob, which is a list containing the corrector magnets names as variables and its delta values is the dependent variable, the desired output of the trained model.

7.1.1 First experiment

Data simulation

The training data is simulated using MAD-X and a simple Python script to call MAD-X code and store the generated data. To generate the samples an ideal model is modified by adding random uncertainties in order to simulate the deviation from the ideal design. The generation of samples is implemented in such a way that datasets creation is parallelized using different cores. Each generated sample is appended to a Numpy array which is then stored in a file in order to proceed with the training later. The generated MAD-X file containing the random magnetic field errors and the generated twiss files are removed after appending the data sample to the Numpy dataset in order to reduce the memory needed for data simulation. One data sample consists of deviation in horizontal and vertical phases between generated twiss and in the ideal twiss and the value of the magnetic field error introduced to the ideal twiss file in order to generate the perturbed model. Each twiss file contains 1026 BPMs (for both planes), therefore phase deviation is represented as a vector with size 1026 and each sample represents a multi-dimensional Numpy array. In first experiments, the used dataset contains 50000 samples generated choosing randomly one optics setting - either injection or $\beta^* = 40$ cm. Before performing the training and prediction, the input parameters for training and test dataset have been normalized to the norm of the input vectors.

Training

In order to train the estimators and obtain the prediction, the machine learning library Scikit-learn [75] for the Python programming language is used. It offers various classification, regression, clustering algorithms and is designed to consolidate with the Python numerical and scientific libraries NumPy and SciPy which are widely used in OMC Software.

Scikit-Learn Neural Network module offers models based on neural networks such Multi-Layer Perceptron models for supervised learning, Restricted Boltzmann machines which are unsupervised nonlinear feature learners based on a probabilistic model. Ensemble methods such Random Forest are available for classification and regression tasks, as well as linear regression models.

Considering the problem of predicting the magnetic fields errors, supervised learning approach applying one of regression models is used. Given a set of features $X = x_1, x_2, ..., x_m$ and target y, an estimator can learn a regression model or in the case of Multi-Layer perceptron, a non-linear approximator.

In first experiment the features are represented by phase deviations between ideal and perturbed twiss files and the target output is the magnetic field error, which is a vector containing 206 magnets. In this sense, a vector of 206 errors in magnetic fields should be predicted based on two given vectors of phase errors, each consisting of 512 BPMs measurements (horizontal and vertical plane). In order to obtain the correction values, the sign of the computed errors just has to be flipped.

Results

The generated dataset was splitted into training set containing phase errors and corresponding errors in magnetic field (80% of generated data) and test set (20%), which was used as input to predict the magnetic field errors based on given phase errors and to evaluate the performance of different estimators.

The most efficient training was performed using the Random Forest Regressor model. The results of prediction after the training using 50000 data samples is shown in Figure 7.1. The observable fit to the target errors shows a great potential for further investigation on obtaining the correction using regression models.

The performance of different estimators was compared by the mean absolute error (MAE) as it is more robust to the outliers than the mean squared error. The second figure of merit is explained variance (var_{exp}), that measures the proportion to which a model accounts for the variance of a given data set. The closer to 1 is the score of explained variance, the better is the model performance. The two figures of merit are computed as follows:

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}| \tag{7.1}$$

$$var_{exp}(y, \hat{y}) = 1 - \frac{var\{y - \hat{y}\}}{var\{y\}} \tag{7.2}$$

where y is correct target output, $\hat{y}$ estimated output, var the variance and N the number of samples in the test data set.

The model quality has been also reviewed in terms of overfitting. To disprove the overfitting of the model, the performance scores measured on a for the training and test data sets have to be approximately equal. Overfitting is not identified in any of the experiments described bellow.

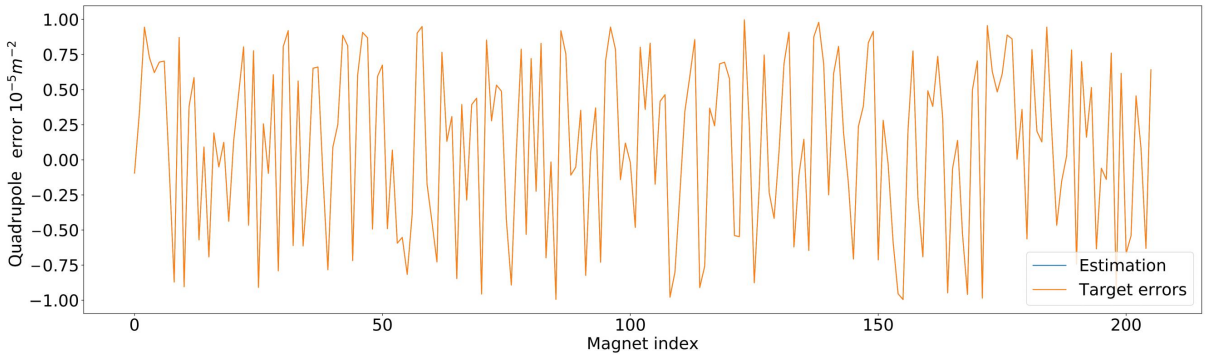

Figure 7.1: Prediction of magnetic errors after the training the Random Forest Regressor with 10 estimator trees, the prediction with MAE $0.07 \times 10^{-5} m^{-}2$ and explained variance 0.93.

Random Forest

A random forest is an ensemble method that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control overfitting [75]. As other models, forest classifiers and regressors have to be fitted with two arrays: a sparse or dense array X of size $n_{samples} \times n_{features}$ holding the training samples, and an array Y of size n_{sample} holding the target values in case of regression or class labels in case of classification for the training samples. In contrast to the original publication [34], the Scikit-Learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

The core element of random forest approach is that for each tree a random vector is generated independently from past random vectors, but with the same distribution. A tree is grown using the training set and the random vector generated for this tree, resulting in a classifier $h(x, Q_k)$ where x is a vector from input X and Q_k the random vector generated for the tree k.

The result of the introduced randomness is a minor increase of the forest bias (with respect to the bias of a single non-randomized tree). On the other hand, due to the averaging the forest also experiences a decrease of variance, which is greater than a value needed just to compensate the increase in introduced forest bias. Therefore, a better overall model can be achieved.

The main parameters to adjust when using Random Forest are number of trees and maximum number of features to consider when looking for the best split. The number of trees affects the quality of the model, but also the computation time. The results will not get significantly better beyond a critical number. Empirical good values for maximum number of features is the number of input parameters in dataset for regression tasks. For classification, the maximum number of features should be the square root of input size. The module also allows the parallel construction of the trees and the parallel computation of the prediction.

The comparison between the appropriated estimators for the correction prediction regression task is presented in this section. A model for correction estimation has to satisfy specific requirements which follow from the given conditions: since the task is a regression problem with predominantly linear dependencies between input and output variables, the following linear models are investigated:

1. Random Forest Regressor
2. Orthogonal Matching Pursuit
3. Multi-Layer-Perceptron Regressor

Orthogonal Matching Pursuit

Orthogonal Matching Pursuit algorithm was introduced by G. Mallat and Z. Chang [76]. The linear model OMP offered by Scikit-Learn is based on the implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit [77]. The K-SVD Algorithm is a highly effective method of training overcomplete (i.e. redundant) dictionaries and its implementation is optimized in terms of speed and memory consumption. The Batch-OMP implementation is specifically suited for sparsity-based techniques.

The basis model suggests that natural signals can be efficiently represented as linear combinations of prespecified *atom signals* , where most of coefficients are zero. The sparse approximation problem can be efficiently solved using several approximation techniques, including Orthogonal Matching Pursuit. The choice of the dictionary is a crucial question for the sparse approximation problem - the dictionary can be either derived from a predesigned transform or from example data using a training process as in the case of OMP Regressor in Scikit-Learn implementation.

The core of the used OMP algorithm is the selection of the atom with the highest correlation to the current residual at each step. After the atom selection, the signal is projected orthogonally to the span of selected atoms, the residual is computed and the process repeats.

The improvements in K-SVD algorithm are achieved by replacing the explicit SVD computation with a simpler approximation. The K-SVD algorithm meets only a local minimum, not a global one and it assumes a reduction of the target function value, not an optimal solution. Hence, the goal of K-SVD is to improve an initially given dictionary and an approximate solution can be used rather than the exact one - just as long as this approximation reduces the target function.

The Scikit-Learn implementation allows to control the sparsity of the solution by giving a desired number of non-zero entries in the solution. Moreover, maximum of residual can be set as a tolerance for the solution improvements and normalization of input data can be included into the training procedure. Figure 7.2 shows the prediction results from OMP model.

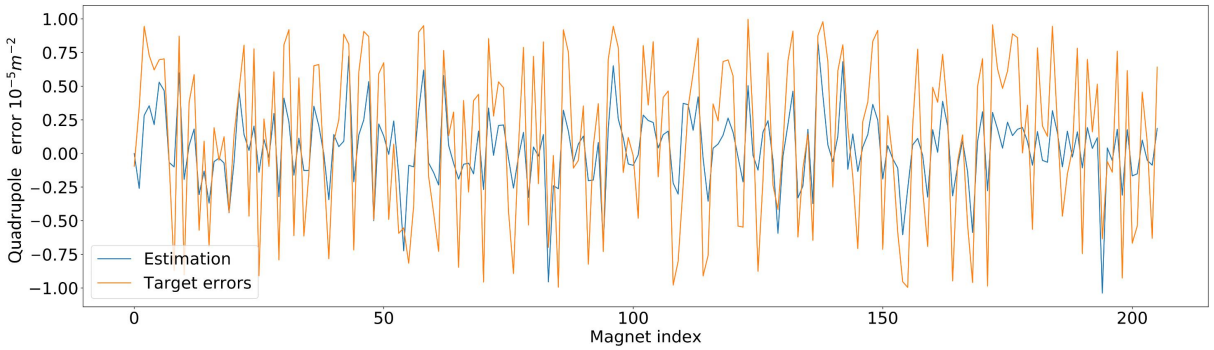

Figure 7.2: OMP showed second best performance with MAE $= 0.29 \times 10^{-5} m^{-2}$ and explained variance of 0.59.

Multi-Layer-Perceptron Regressor

The supervised-learning algorithm Multi-Layer Perceptron learns a non-linear function approximation for either classification or regression using a set of features $X = x_1, x_2, ..., x_m$ and desired target y. The training is performed using back-propagation with a specified activation function for hidden layers. Compared to logistic regression, which solves the tasks using input and output layer, in Multi-Layer Perceptron model can have one or more non-linear hidden layers. The clear advantage of this model is the capability to learn complex non-linear models, however the model uses non-convex functions to compute the losses of training iterations. Since non - convex functions are used, a problem of obtaining the global minimum arises.

The model trains using Limited-Memory BFGS which is an optimization algorithm to approximate Broyden-Fletcher-Goldfarb-Shanno [78] algorithm with smaller amount of computer memory needed to perform parameter estimation tasks. As alternative, Stochastic Gradient Descent or its optimized version *Adam* [26] can be used. Adam outperforms other methods in case of non-convex problems, thus Adam can be efficiently applied for multi-layer neural networks weight optimization as the objective functions are non-convex in this case.

The Multi-Layer Perceptron models require tuning a wide range of hyper-parameters such as number of hidden layers, neurons, iterations, activation function and solver selection. The result from Multi-Layer Perceptron Regressor

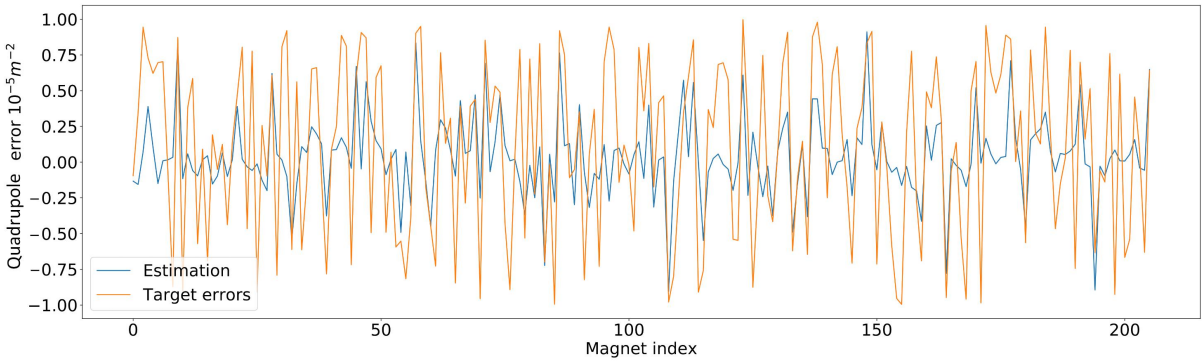

Figure 7.3: Multi-Layer Perceptron Regressor performed the prediction with the worst score giving MAE = $0.37 \times 10^{-5}m^{-2}$ and explained variance of 0.34.

The comparison of different models described above is presented in Table 7.1. The results are obtained by performing the training and prediction test on the generated dataset with two different optics settings, 50000 samples are spitted randomly in training (80%) and test (20%) datasets.

Estimator	MAE $[10^{-5}m^{-2}]$	Explained σ^2
Random Forest	0.07	0.93
OMP	0.29	0.59
MLP Regressor	0.37	0.34

Table 7.1: Comparison between different estimators performing prediction of magnetic errors for mixed optics.

7.1.2 Prediction of errors in phase

Initial idea of training the regression models aims to obtain the prediction of magnetic errors using the errors in phase as input. However, flipping the model and hence, predicting

the phase errors showed an interesting result.

Prediction of phase errors demonstrates that more precise prediction is obtained when the input is based on the 40cm optics while prediction of phase errors for injection optics was insufficient especially in particular regions. Prediction results of two different optics applying Random Forest model are compared in Figure 7.4.

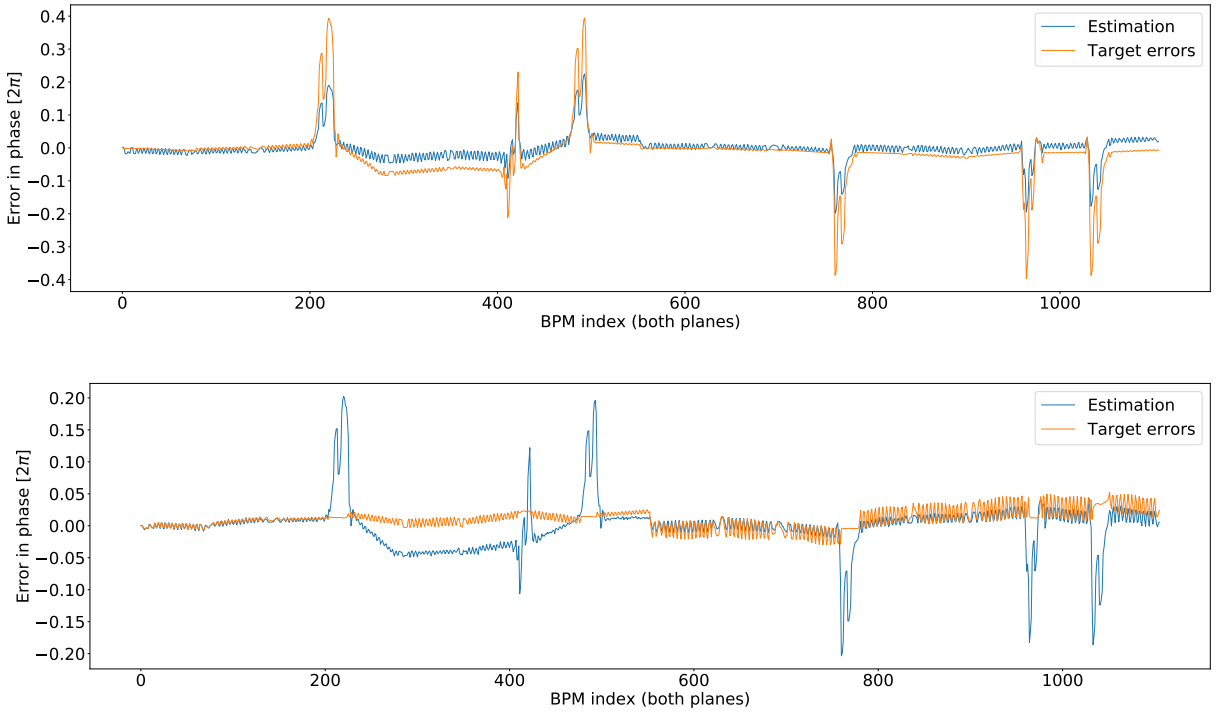

Figure 7.4: The top chart shows the distribution of phase errors in the machine for the 40 cm optics and the bottom chart for the injection optics. The best prediction of phase errors is achieved by applying Random Forest Regressor (MAE = $0.02 \times 2\pi$, explained variance 0.55) on a mixed optics dataset.

The reason for this behavior is the training of the models on a mixed dataset. A model aims to fit the errors in both optics settings which are completely different in IRs. However, the model underfitting in the IRs is an important observation as it shows that the model mimics the design of LHC by recognizing the positions of the IRs and optics specific behavior of measured phase error distribution over the ring.

To achieve better results in prediction of magnetic field errors, in the next experiment the estimators are trained on two separate datasets for injection and $\beta^* = 40$cm optics using the phase errors as input. Moreover, as this problem is predominantly linear, the Linear Regressor was applied in addition to the estimators described in previous section. In following the results of training Orthogonal Matching Pursuit, Multi-Layer Perceptron, Random Forest and Linear Regressor are presented.

7.1.3 Results for Injection Optics

The prediction by Multi-Layer Perceptron Regressor remains unexpectedly insufficient as shown in Figure 7.5. The poor performance of this regressor can be caused by low complexity of the regression problem since Multi-Layer Perceptron usually gives better results compared to other estimators when solving non-linear problems. One possible solution is tuning of hyperparameter such activation function, number of nodes and layers and optimization method. Another possible approach to increase the prediction result is training on a smaller dataset in order to increase the complexity of the problem.

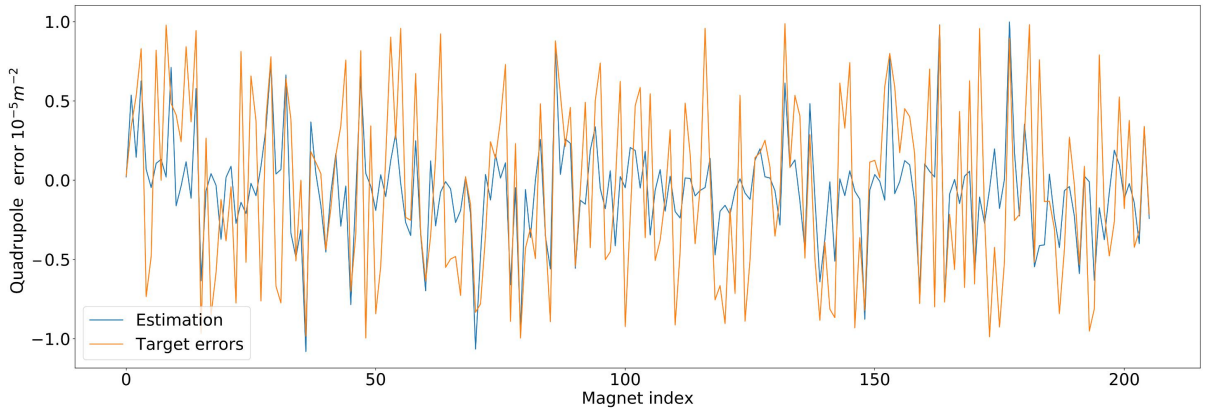

Figure 7.5: Prediction of magnetic errors using Multi-Layer Perceptron Regressor with MAE=$0.35 \times 10^{-5}m^{-2}$ and explained variance of 0.38.

The Orthogonal Matching Pursuit Regressor showed better result in optics specific prediction compared to the prediction based on the training on mixed dataset. The result is presented in Figure 7.6.

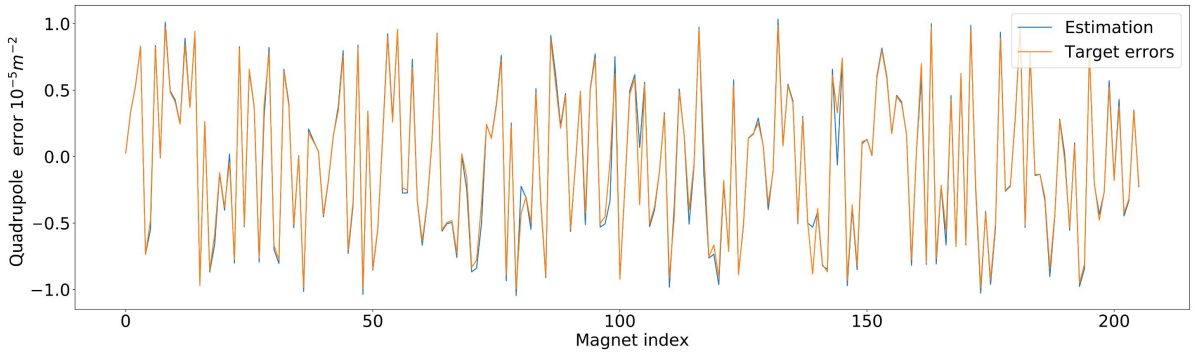

Figure 7.6: Prediction of magnetic errors using Orthogonal Matching Pursuit Regressor with MAE = $0.04 \times 10^{-5}m^{-2}$ and explained variance of 0.97.

The second best prediction was achieved using Linear Regression. From the implementation point of view, this is just plain least squares approach applied to fit the given data.

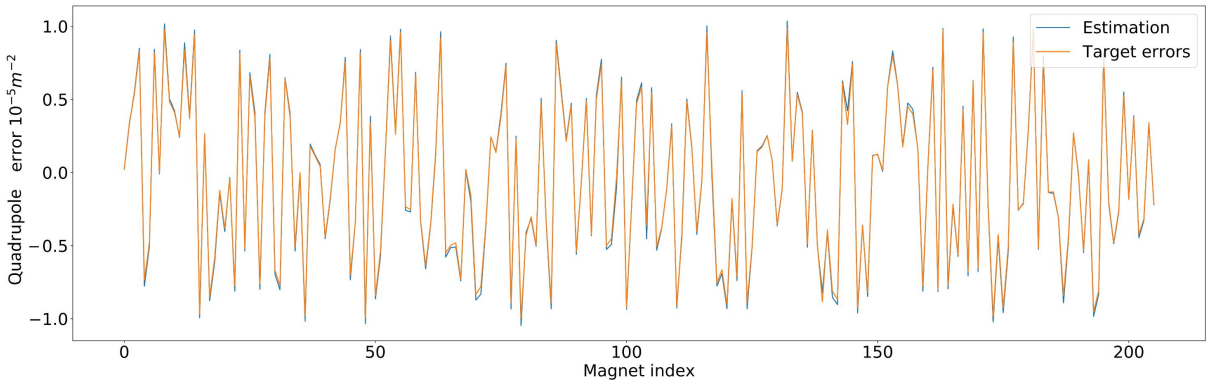

Figure 7.7: Prediction of magnetic errors for injection optics with Linear Regression with MAE $= 0.02 \times 10^{-5}m^{-2}$ and explained variance of 0.99.

The best prediction scores are obtained applying Random Forest model as shown in Figure 7.8. To note is the fact that both Linear Regressor and Random Forest models explain the variance of the data equally well, however the MAE produced by Random Forest model is significantly smaller compared to the MAE of the Linear Regressor. The results of magnetic error prediction using the described estimators are summarized in Table 7.5.

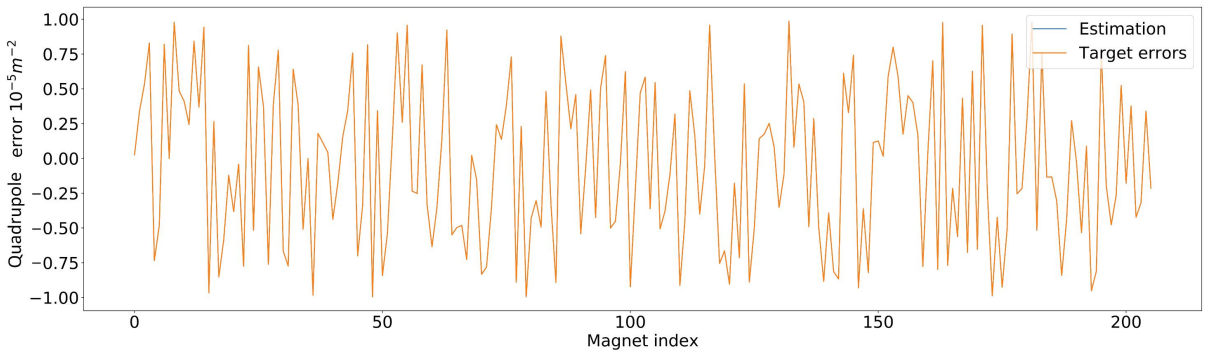

Figure 7.8: Random Forest performs the best prediction with scores MAE $0.005 \times 10^{-5}m^{-2}$ and explained variance 0.99.

7.1.4 Results for β^* = 40 cm

To predict the magnetic field errors for the $\beta^* = 40$ cm optics, the same estimators as for injection optics have been applied on the generated data set with 50000 samples. The

Estimator	MAE $[10^{-5}m^{-2}]$	Explained σ^2
Random Forest	0.005	0.99
Linear Regressor	0.02	0.99
OMP	0.04	0.97
MLP Regressor	0.35	0.38

Table 7.2: Comparison between different estimators predicting magnetic errors for injection optics.

performance of estimators is identical to the results from prediction of magnetic fiend errors of injection optics. The best prediction is achieved by Random Forest as shown in Figure 7.9.

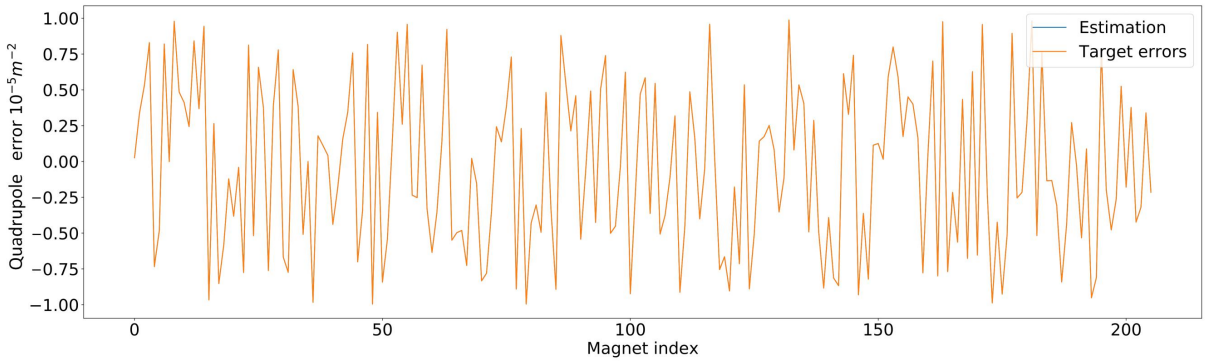

Figure 7.9: Random Forest Regressor prediction for $\beta^* = 40$ cm optics with scores MAE $0.005 \times 10^{-5}m^{-2}$ and explained variance 0.99.

The second best prediction is performed by the Linear Regression model, however the model could not achieve as high score of explained variance for the $\beta^* = 40$ cm optics as for injection optics. Due to the fact that machine design for $\beta^* = 40$ cm optics is more complex than for injection, the error prediction becomes less linear. The result of Linear Regressor prediction is shown in Figure 7.10.

An unexpected difference between the predictions by OMP for injection and $\beta^* = 4$ 0cm optics appears in the results. This difference could be explained with the higher redundancy and sparsity in the input of injection optics data set since the measured phase errors in IRs contain values close to 0. As mentioned in section 7.1.1, OMP Regressor is based on the implementation of K-SVD algorithm, which is suited for training redundant dictionaries. The prediction result is presented in Figure 7.11.

The MLP Regressor showed slightly better results compared to prediction of magnetic field errors of injection optics. While the MAE remains approximately the same, the explained variance score increased from 0.38 to 0.47. Although the score is insufficient low,

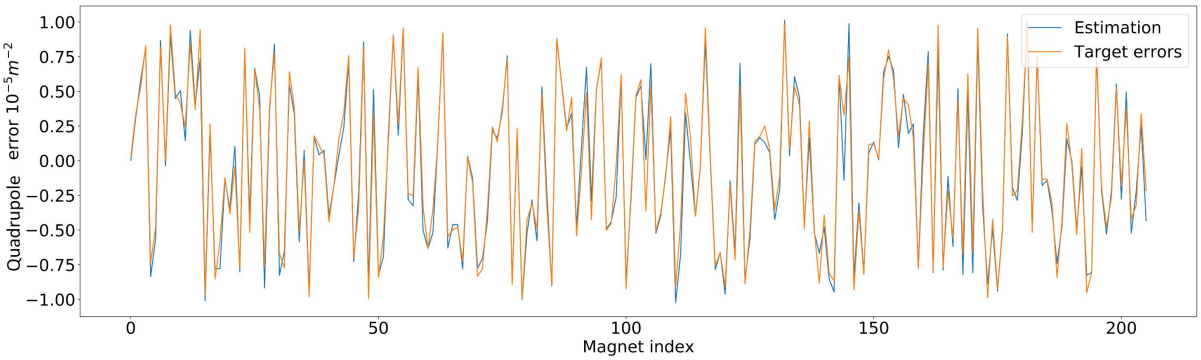

Figure 7.10: Linear Regressor performed with MAE = $0.16 \times 10^{-5}m^{-2}$ and explained variance of 0.86.

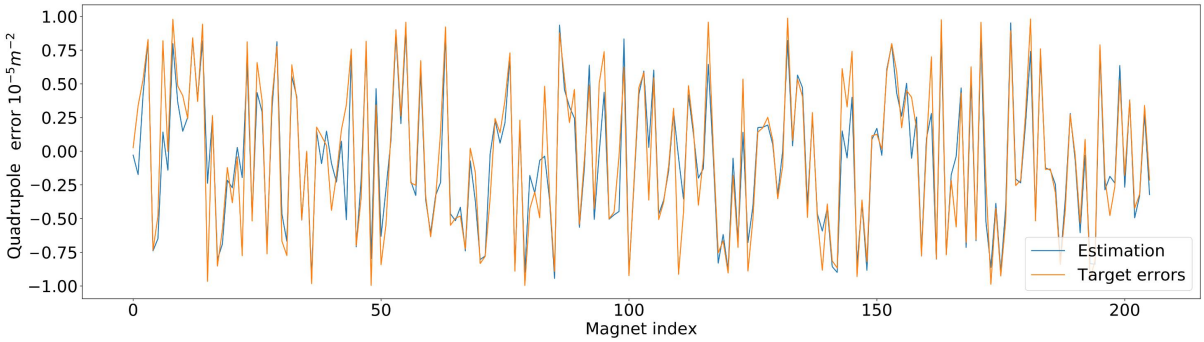

Figure 7.11: OMP Regressor achieved prediction of magnetic field errors with MAE = $0.21 \times 10^{-5}m^{-2}$ and explained variance of 0.76.

the increase shows that the MLP model performs better on less linear task. The prediction result for $\beta^* = 40$ cm optics obtained from MLP Regressor model is shown in Figure 7.12.

The results of magnetic field error prediction for $\beta^* = 40$cm optics are summarized in Table 7.3.

Estimator	MAE $[10^{-5}m^{-2}]$	Explained σ^2
Random Forest	0.005	0.99
Linear Regressor	0.16	0.86
OMP	0.21	0.76
MLP Regressor	0.33	0.47

Table 7.3: Comparison between different estimators predicting magnetic errors for β^*=40 cm optics

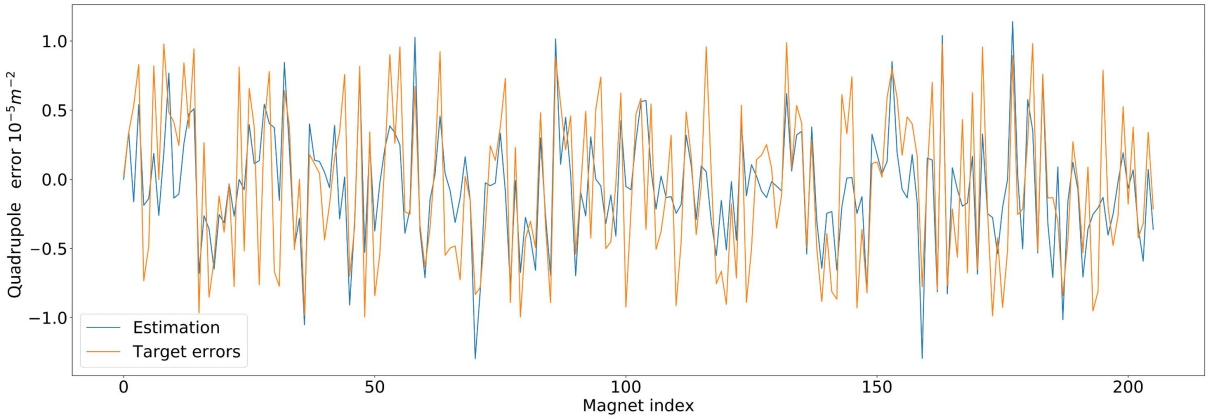

Figure 7.12: MLP Regressor predicted the magnetic field errors with MAE = 0.33 × $10^{-5}m^{-2}$ and explained variance of 0.47.

7.1.5 Conclusion

The results of the prototyped correction prediction clearly show the great potential of the prediction-based correction computation. The problem lies definitely in the domain of regression tasks and can be solved successfully applying well-known machine learning techniques.

Comparison of the results on the different optics settings and data sets demonstrated that Random Forest Regressor achieves the best scores in magnetic error prediction. Consequentially, more powerful models can be built based on this method, e.g. applying boosting techniques.

However, to be noted is that this high performance was achieved by performing training and prediction on generated data. The real data is more complex and requires potentially a bigger training dataset in order to learn the representation of the real LHC optics. Particular circumstances such powering of the magnets and different optics settings have to be taken into account in order to generate more realistic dataset. Alternatively, real measurements and correction data from the past can be used to train the prediction model.

7.2 Faulty BPMs Recognition

In order to find an appropriate method for identification of faulty BPM signals in data, the past measurements have been analyzed. The BPMs which have been identified as "bad" by SVD Clean are collected from all measurements starting from 2015. The information about identified bad BPMs is important for further analysis of bad BPMs features. The BPMs with highest rates of identification as "bad" are listed in the tables bellow:

The analysis is performed on 3742 measurements taken in the time between 2015 and 2017. The Figures 7.13 - 7.16 show how frequently each of BPMs has been identified as

BPM	Ratio [%]
BPMSE.4R6.B2	33
BPM.9R6.B2	25
BPMWC.6L7.B2	24
BPMR.6L7.B2	23

Table 7.4: Horizontal plane

BPM	Ratio [%]
BPM.10R4.B2	13
BPM.28L3.B1	12
BPM.34R6.B1	11
BPM.24L5.B1	10

Table 7.5: Vertical plane

bad by SVD Clean. In this analysis, only the BPMs identified as "bad" in more than 50 measurements are considered. The observation shows that significantly larger number of bad BPMs appear in the horizontal plane. Comparing beam 1 and beam 2, more BPMs have been identified as bad in beam 2 measurements. These facts have to be taken into account for the feature importance analysis, (e.g using the Association Rules analysis), larger weights can be assigned to the BPMs, which appear as "bad" in a significant number of measurements.

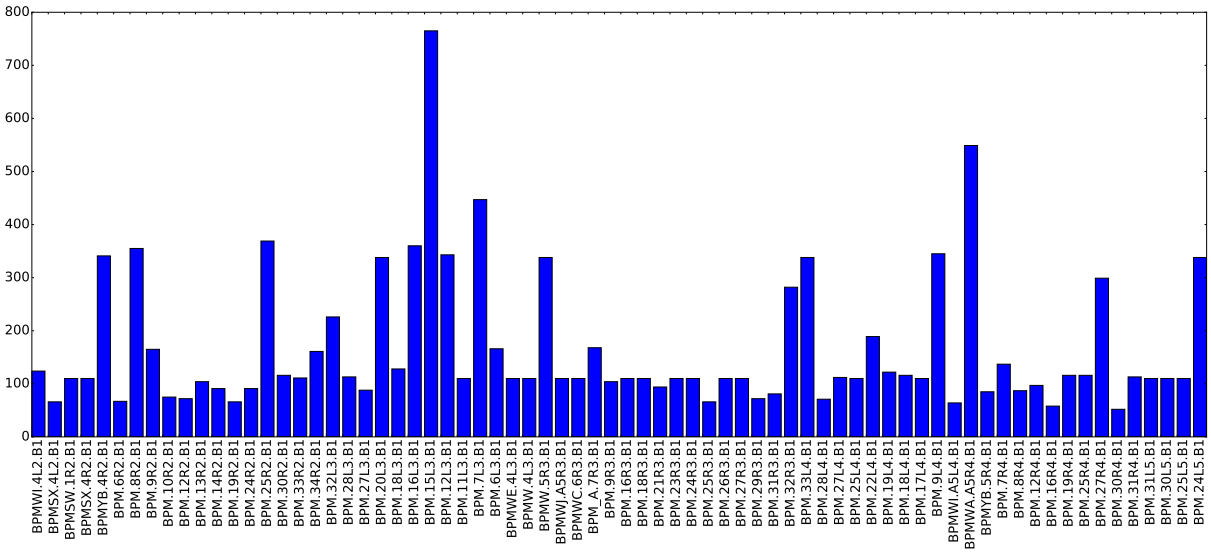

Figure 7.13: Bad BPMs identified for Beam 1 measurements in horizontal plane.

7.2.1 Autoencoder for identification of bad BPMs

Currenty, the identification of bad BPMs is performed by SVD Clean (see 4.3.3), which includes Singular Value Decomposition to cut weak signal in turn-by-turn data and removes BPMs with flat signal, as well as known bad BPMs. Since in further analysis steps remaining unphysical signals are observed, potentially the denoising procedure can be improved. Statistical analysis of bad signal detection using SVD Clean shows, that predominant amount of BPMs are good, therefore the presence of bad BPM signal in turn-by-turn data can be seen as an anomaly.

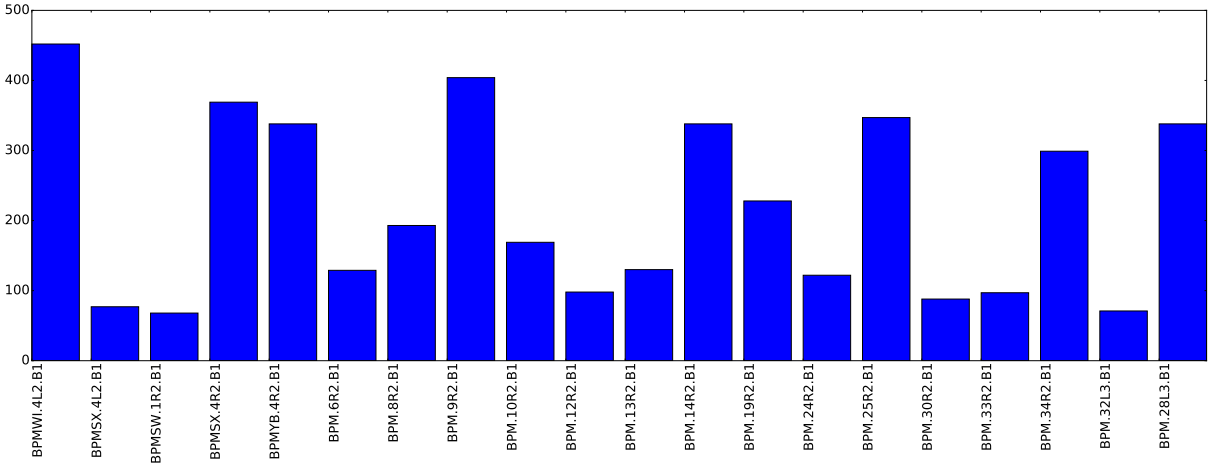

Figure 7.14: Bad BPMs identified for Beam 1 measurements in vertical plane.

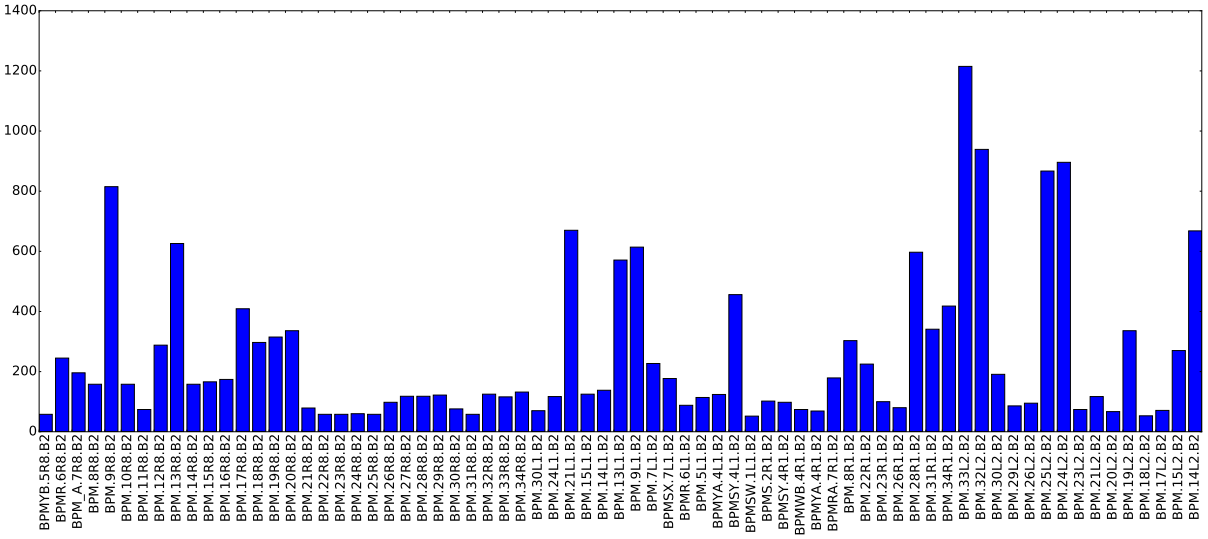

Figure 7.15: Bad BPMs identified for Beam 2 measurements in horizontal plane.

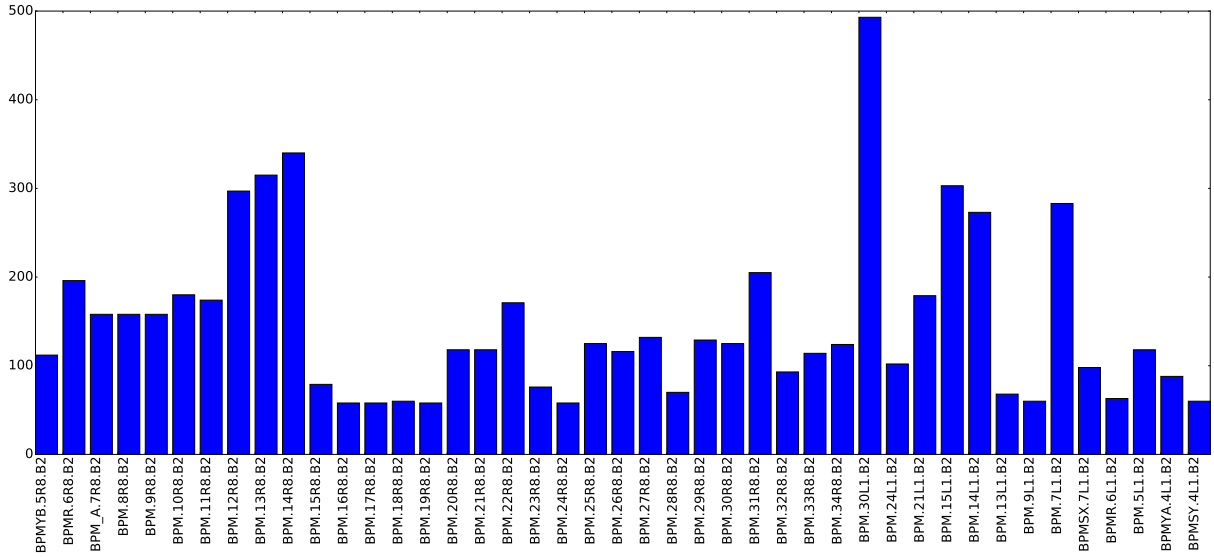

Figure 7.16: Bad BPMs identified for Beam 2 measurements in vertical plane.

Training data

The previous measurements build a sufficient base to train and evaluate classification model. In order to obtain more realistic data, only turn-by-turn data from the recent measurements (starting from May, 2017) are collected and used for training. As each measurement includes as well SVD Clean results as the uncleaned data file, the results from the trained model can be compared to SVD Clean results.

Each measurement file consists of 1024 samples (512 BPMs per plane) and since the measurement is taken for 6600 turns, each BPM has 6600 measured signals which are considered as features of a BPM. The target of the training is the binary classification of turn-by-turn data measured at given BPM - identification as good or bad.

Concerning the size of the dataset and the number of features, the samples are divided into a smaller size of 2200 turns. Consequently, the number of features is reduced and a bigger training dataset could be obtained. The number of turns is an adjustable measurement parameter, although 6600 is the default value and most of the measurements are taken at 6600 turn, the turn number can vary for different measurements. One possible solution is padding with zero in case of lower turns number, however it can affect the classification results. The separation into smaller samples makes the dataset more reliable.

Training

Autoencoders follow a typical feed-forward neural network architecture with the difference that the output and input layers have exactly same number of neurons and the input dataset itself is used as its target. Using an autoencoder the input data can be reproduced as the output of the model. The learned representation can be then used for further binary classification (see 2.5.1).

As Scikit-Learn library does not include auto-encoder implementation, TensorFlow library [79] was used to build an autoencoder consisting of multiple layers for decoding and encoding.

A TensorFlow computation is described by a directed graph, which is composed of a set of nodes. The graph represents a dataflow computation in which graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them [79]. TensorFlow enables GPU support to speed up the computations. Building the models using TensorFlow allows application on more specific problems as the model architecture is more flexible than in Scikit since the numbers of nodes in the layers, connections between them, activation functions and weights can be defined explicitly.

The first part of the network encodes the given data and the second part decodes the data in order to obtain the original input. During the training, the loss which is computed as RMS error has to be minimized by backpropagation. After the desired loss value is

achieved, the RMSE values of each data point has to be computed in order to identify the anomalies. As the RMS measures the difference between the learned pattern and the anomaly value, the BPMs with highest loss values can be considered as anomalies and classified as bad. In the following results of this approach are presented.

Results

After performing the training of the autoencoder model built with TensorFlow, the results have been evaluated using one of the past measurements that contains bad BPMs identified by SVD Clean. The losses of BPMs are shown in Figure 7.17. The histogram shows,

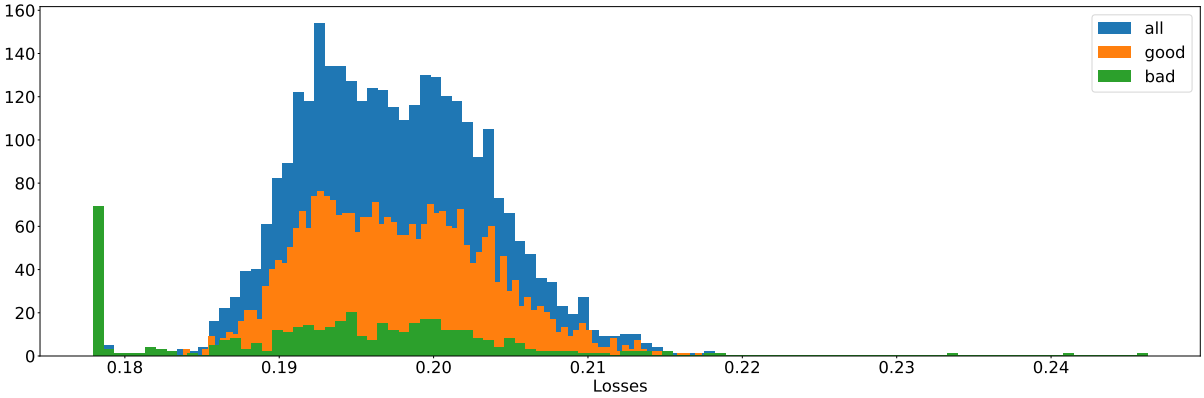

Figure 7.17: Computed losses per BPM in the test measurement.

that some bad BPMs could not be identified as anomalies as they have identical losses as good BPMs. But more interesting is the fact, that bad BPMs that have not been identified by SVD Clean with default settings could be found. To investigate this fact more detailed, SVD Clean with different setting has to be executed on this test measurement. The observation of turn-by turn data measured by the BPMs with losses higher than *0.21* demonstrates that the signal at these BPMs is faulty in fact as shown in figure 7.18.

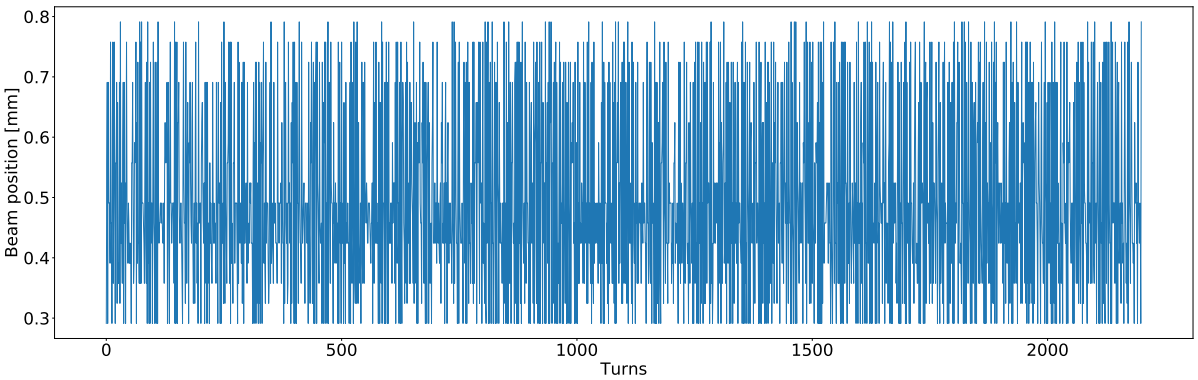

Figure 7.18: Turn-by-turn signal measured at BPM classified as bad using auto-encoder. SVD Clean could not recognize this as faulty signal.

Conclusion

The classification method using the autoencoder can be used in addition to SVD Clean. In the current state the model cannot be used as an alone-standing method for noise recognition. The results have to be studied more carefully, especially the fact that significant amount of bad BPMs corresponds to low losses.

Additional constraints to compute the losses can be used in order to support the classification results e.g. using the statistical results presented above. Weights can be assigned considering the probability of a BPM to be bad based on information about the regions where the BPMs, that are recognized as bad more often are placed.

7.2.2 Clustering

Since the identification of bad BPMs can be seen as a problem of separating the data, clustering algorithms can be applied to solve this task. In order to find an appropriate clustering algorithm, certain facts have to be taken into account. The data demonstrates a significantly high rate of outliers that affect the optics analysis results and therefore have to be eliminated.

The algorithms based on nearest neighbor search such K-means [42] require a definition of desired number of clusters. The clusters are built by identifying the mean (centroid) of each cluster and searching for nearest data points. These algorithms assume convex clusters and perform poorly on the elongated data sets. Due to this limitations, nearest neighbor search algorithms are not suitable for our problem, since the possible number of clusters is unknown and the clusters shape is not necessarily convex. The *DBSCAN* [80] algorithm assumes clusters of arbitrary shape and views clusters as areas of high density separated by areas of low density, instead of looking for the centroids. Therefore, DBSCAN is better suited for the problem of identification of bad BPMs. In the following a brief introduction to DBSCAN algorithm is given.

DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) was proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996 as a data clustering algorithm for large spatial databases. The algorithm relies on density-based notion of clusters which is designed to discover clusters of arbitrary shape. The notion of clusters apply as well to 2D and 3D Euclidean space as to high dimensional feature space.

The main idea of the algorithm is the concept of core samples, which are samples in areas of high density. A cluster is therefore a group of core samples which have to contain a minimum number of points in the neighborhood, which is computed by the chosen distance metric. The cluster also includes non-core samples that are in the neighborhood of a core sample, but are not core samples themselves (as there are not enough points in

the neighborhood). The Eps-neighborhood is defined by Martin Ester [80] as

$$N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\} \tag{7.3}$$

for data points p and q in data set D. For every data point p in a cluster C there is a point q in C so that p is inside of the Eps-Neighborhood of q and $N_{Eps}(q)$ contains at least a defined minimum number of points *MinPts*. This condition indicates a data point as *directly-density-reachable*, so that the requirement for a core-point is fulfilled. The non-core points in a cluster are *density-reachable*, which means that between two points p and q is a chain of points such that p_{i+1} is directly density-reachable from p. Consequently, the input of the algorithm is the neighborhood *Eps* and the minimum number of data points *MinPts*. The data points in the data set not belonging to any of clusters are defined as noise.

To build a cluster, DBSCAN starts with a random point p and retrieves all points density-reachable from p wrt. Eps and MinPts. If p is a core-point, this procedure returns a cluster. In case p is a non-core point, no points are density-reachable from p and DBSCAN continues with the next point in the data set. If two clusters are very close to each other, a point that belongs to both clusters as a non-core point might exist. In this case, the point will be assigned to the cluster discovered first.

Results

In order to evaluate clustering as a method for faulty BPM identification, one of turn-by-turn measurements taken on 19 May, 2017 is used as a data set. Before applying DBSCAN on the data, following features have been extracted: orbit, tune and peak-to-peak value from the measurement. The computed values are scaled to the range from 0 to 1.

Depending on the tuning of input parameters of DBSCAN algorithm, one or more clusters can be built. Figure 7.19 demonstrates the result applying DBSCAN with Eps = 0.075 and MinPts = 5. Building several clusters of good BPMs is potentially interesting to discover the properties of good BPMs and the relations between the features. To indicate bad BPMs we can consider one cluster.

Due to the large variance of the measured values, the orbit can not be used as a reliable feature. Instead, in further experiments the orbit (the average of measured beam position) was subtracted from the data. The result obtained by DBSCAN using Eps = 0.075 and MinPts = 100 using only tune and peak-to-peak value as features is presented in Figure 7.20. The algorithm discovers significant amount of BPMs that have been recognized as bad by SVD clean, but also bad BPMs that could not be identified by SVD Clean. In Figure 7.21 an example for turn-by-turn data identified as bad by DBSCAN, but not by SVD Clean is shown. In the presented results, Euclidean metrics is used as a distance metrics between the data points.

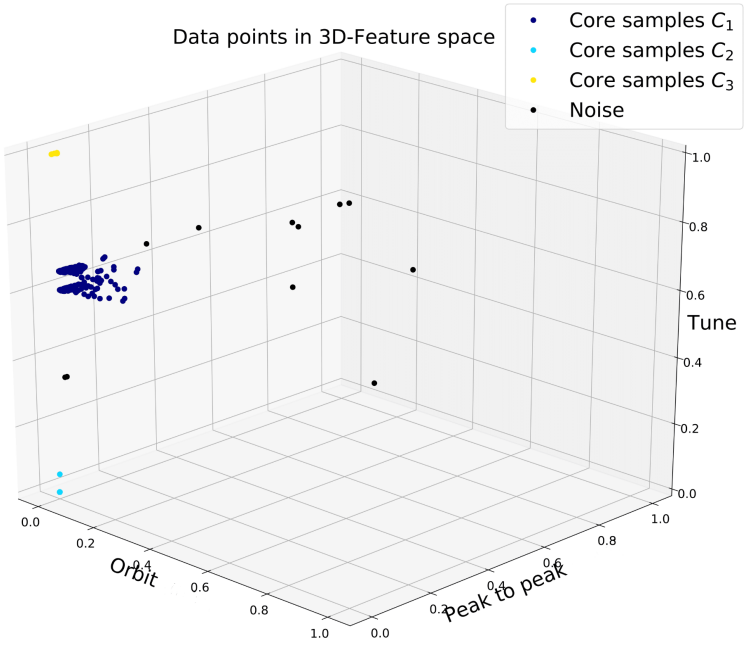

Figure 7.19: Bad BPMs identified as outliers and 3 clusters of good BPMs.

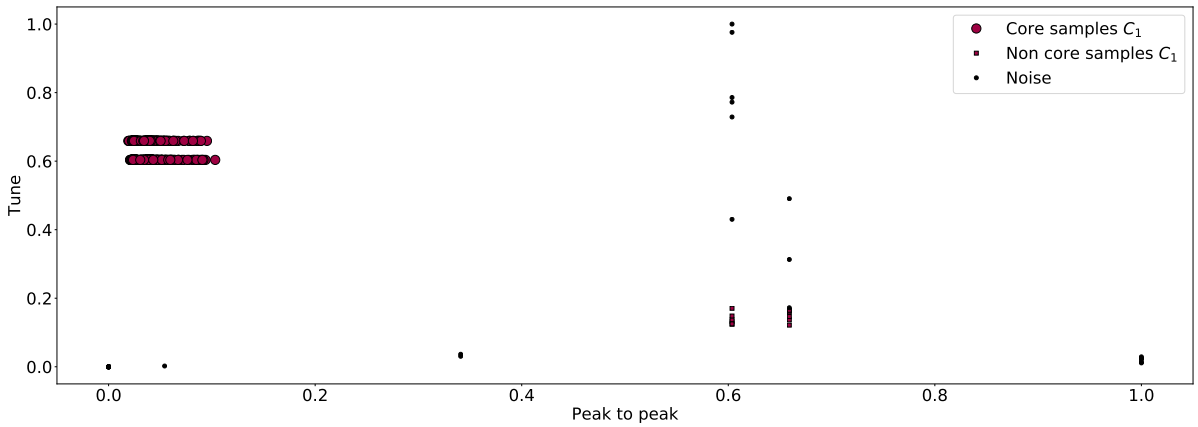

Figure 7.20: The result of applying DBSCAN on turn-by-turn measurement using tune and peak-to-peak values.

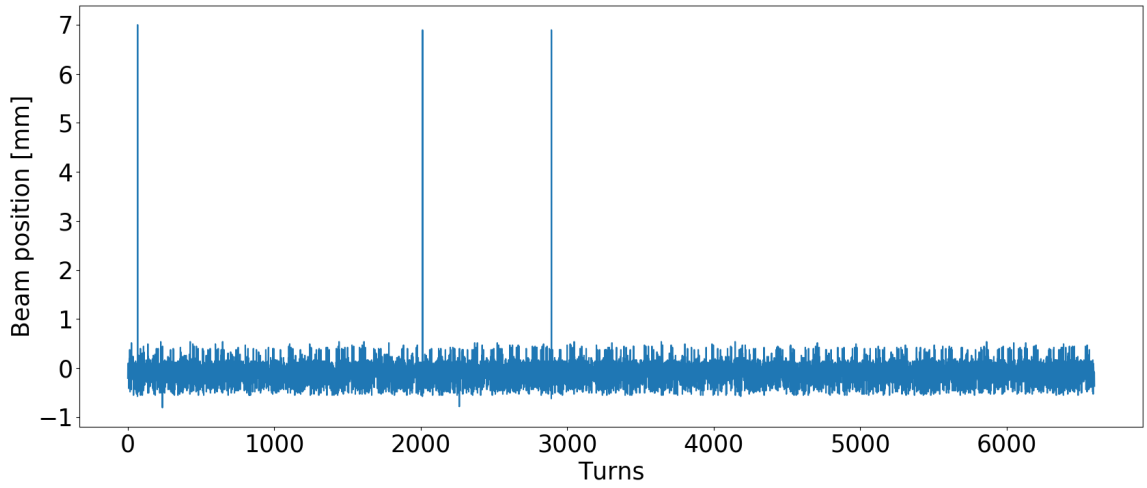

Figure 7.21: The BPM identified as bad by DBSCAN.

The Euclidean distance is described as follows

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{7.4}$$

with data points p and q represented by euclidean n-dimensional vector. Different metrics yield different results for the same algorithm input. In Figure 7.22 clustering using euclidean, cosine, and mahalanobis distance is presented. The most reliable noise identification regarding to the observation of turn-by-turn data is obtained using the euclidean distance.

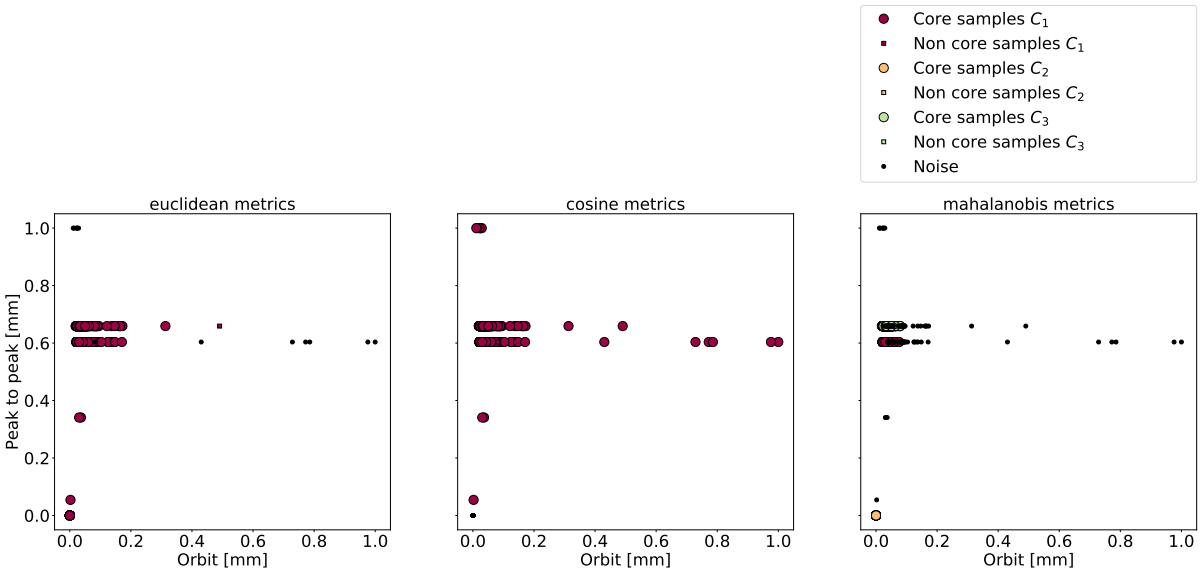

Figure 7.22: DBSCAN applying different metrics with Eps = 0.3 and MinPts = 5.

Conclusion

Clustering approach can be applied in addition to SVD Clean, potentially it can replace this method if higher precision can be achieved e.g. through including more features into the data representation or combining different clustering algorithms. The method has to be tested with data acquired from measurements with different settings and the BPMs classified inconsistently with SVD Clean have to be analyzed in detail. Applying clustering on a bigger data set containing of several past measurements, the properties of BPMs clustered together can be investigated which will lead to better understanding of noise presence in the analysis data.

Apart from identification of bad BPMs, clustering using DBSCAN can be applied as an universal data cleaning method on different stages of the data analysis during the measurements such the cleaning of Drive output files containing the values for optics observables. As the algorithm is robust against the outliers, it can be applied to eliminate them. Although different input parameters for the algorithm might be required for specific applications as the range of data is not the same for different features. Alternatively, the data can be normalized to one range as it was done in the presented prototype.

8 Conclusions and Outlook

This thesis studies potential application of machine learning in optics measurements and correction at LHC aiming to reduce the human effort, speed up the process and improve the correction accuracy. For this, improvements in terms of automation and communication between software applications have been introduced to the OMC software. These improvements lead also to better structured data representation with a higher abstraction level, since it is needed before the application of machine learning.

The study of the optics measurement and correction process demonstrated the potential for improvements using machine learning techniques. Different aspects of the process ranging from signal artifacts recognition to correction computation can be defined as machine learning tasks. The main practical applications studied are recognition of bad BPMs and prediction of quadrupole errors.

The prediction of quadrupole errors is based on the idea of defining optics correction as a regression task that can be solved with the help of a trained model. The presented prototype of prediction based correction shows that this approach is possible and produces meaningful and consistent results. Moreover, prediction of phase errors shows that a regression model can also learn the behavior of the machine for different optics settings. In prediction of phase errors based on given magnetic fields errors the same pattern as in the real measurements can be observed.

Excellent precision could be achieved in correction prediction from given phase deviation between measurement and design for injection and $\beta^* = 40$ cm optics. The mean absolute error $0.005 \times 10^{-5} m^{-2}$ and explained variance of 0.99 are obtained applying the Random Forest regressor on the generated test data. This result solidly proves the potential of the prediction-based correction approach.

For further improvements on prediction quality, different model boosting techniques [81] which construct ensembles with higher capacity than individual models have to be investigated. It would be also interesting to apply models built with such frameworks as TensorFlow [79] and Keras [82] which allow to design more complex models especially concerning the ANN architecture, since the multi-layer perceptron offered by Scikit-Learn library yields insufficient results. More realistic data needs to be generated in order to train the model appropriately and to ultimately produce prediction that could be used in production.

Very promising results have been also obtained regarding the identification of bad BPMs. Both autoencoder and clustering methods discovered bad BPMs that could not be

identified by SVD Clean. The autoencoder approach requires additional improvements in terms of complexity of the network architecture - more layers might be needed and different types of layers should be combined. Besides anomaly detection, autoencoder model can help to extract new features in measurements and analysis data. DBSCAN clustering can be easily integrated into current OMC applications as a complementary method to SVD Clean.

A possible extension of the cleaning functionality using clustering is to integrate the Fourier spectrum information features as it contains optics observable values. The clustering methods allow to clean in a space of arbitrary dimensionality and thus an arbitrary number of optics parameters can be selected to perform the cleaning procedure.

This will allow to observe the relation between particular observables and the patterns in the clustered data. The fact that particular good BPM data points can be separated in different clusters depending on the algorithm settings is of interest for further investigation. A deeper analysis of clustering patterns could cast light on unobserved properties of BPMs.

The suitability of machine learning methods has been clearly shown in the performed experiments. Considering future accelerator projects like HL-LHC and FCC that will require more challenging optics control, more powerful analysis methods will be needed - machine learning techniques might become the key technology to implement fast powerful data analysis and discover new useful relations and features in the data. The positive results presented in this thesis open more space for further investigation and deeper analysis of appropriate approaches.

Bibliography

[1] T. Bach and R. Tomás. Improvements for Optics Measurement and Corrections software. Report No. CERN-ACC-NOTE-2013-0010, August 2013.

[2] J. Coello de Portugal, F. Carlier, A. Langner, T. Persson, P. Skowroński, and R. Tomás. OMC Software Improvements in 2014. In *Proceedings, 6th International Particle Accelerator Conference (IPAC 2015): Richmond, Virginia, USA, May 3-8, 2015.*

[3] V. Maier. Software Quality Improvement in the OMC Team. CERN-Thesis-2014-028, February 2014.

[4] T. Persson, F. Carlier, J. Coello de Portugal, A. Garcia-Tabares Valdivieso, A. Langner, E. H. Maclean, L. Malina, P. Skowronski, B. Salvant, R. Tomás, A. C. García Bonilla. LHC optics commissioning: A journey towards 1% optics control. *Phys. Rev. Accel. Beams*, **20**(6), 2017.

[5] R. Tomás, M. Aiba, A. Franchi, and U. Iriso. Review of linear optics measurement and correction for charged particle accelerators. *Phys. Rev. Accel. Beams*, **20**(5), May 2017.

[6] F. Calier J. Coello de Portugal E. Fol A. Garcia-Tabares Valdivieso M. Gasio A. Langner E.H. Maclean L. Malina J. Olexa P. Skowronski R. Tomás D. Valuch A. Wegscheider T. Persson, G. Baud. Optics control in 2016. `https://indico.cern.ch/event/578001`. Evian Workshop.

[7] F. Carlier and R. Tomás. Accuracy and feasibility of the β^* measurement for LHC and High Luminosity LHC using k modulation. *Phys. Rev. Accel. Beams*, **20**(1), January 2017.

[8] R. Calaga, R. Miyamoto, R. Tomás, and G. Vanbavinckhove. β^* Measurement in the LHC Based on K-modulation. Report No. CERN-ATS-2011-149, September 2011.

[9] M. Kuhn, B. Dehning, V. Kain, R. Tomás, G. Trad, and R. Steinhagen. New Tools for K-modulation in the LHC. Report No. CERN-ACC-2014-0159, June 2014.

[10] P. Baldi, P. Sadowski, and D. Whiteson. Searching for Exotic Particles in High-Energy Physics with Deep Learning. *Nature Commun.*, 5, 2014.

[11] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. D. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, and P. Vahle. A Convolutional Neural Network Neutrino Event Classifier. *JINST*, 11(09), 2016.

[12] E. Bozoki and Friedman A. Neural network technique for orbit correction in accelerators/storage rings. *AIP Conference Proceedings*, 315(1), 1994.

[13] E. Meier, Y. R. E. Tan, and G. S. LeBlanc. Orbit correction studies using neural networks. In *International Particle Accelerator Conference 2012*, New Orleans, 2012. Proceedings of IPAC2012.

[14] A. L. Edelen, S. G. Biedron, B. E. Chase, D. Edstrom, S. V. Milton, and P. Stabile. Neural Networks for Modeling and Control of Particle Accelerators. *IEEE Trans. Nucl. Sci.*, 63(2), 2016.

[15] K. Tian, J. Safranek, and Y. Yan. Machine based optimization using genetic algorithms in a storage ring. *Phys. Rev. ST Accel. Beams*, **17**(2), 2014.

[16] L. Yang, D. Robin, F. Sannibale, C. Steier, and W. Wan. Global optimization of an accelerator lattice using multiobjective genetic algorithms. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 609(1), 2009.

[17] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2), 2002.

[18] Steven A. Harp, T. Samad, and A. Guha. Advances in neural information processing systems 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[19] A. Edelen, S. Biedron, D. Bowring, B. Chase, J. Edelen, S. Milton, and J. Steimel. Neural Network Model Of The PXIE RFQ Cooling System and Resonant Frequency Response. In *Proceedings, 7th International Particle Accelerator Conference (IPAC 2016): Busan, Korea, May 8-13, 2016*, 2016.

[20] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[21] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[22] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[23] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25, 1847.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. MIT Press, Cambridge, MA, USA, 1988.

[25] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12, July 2011.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[27] W. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 1943.

[28] Jack D. Cowan. Neural networks: The early days. In *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990.

[29] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

[30] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405, 2000.

[31] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[32] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[33] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, London, UK, 2000. Springer-Verlag.

[34] L. Breiman. Random forests. *Mach. Learn.*, 45(1), October 2001.

[35] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. 2016. Book in preparation for MIT Press.

[36] Geoffrey E. Hinton and James L. McClelland. Learning representations by recirculation. In *Neural Information Processing Systems*. American Institute of Physics, 1988.

[37] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In *Knowledge Discovery in Databases*. AAAI Press, 1991.

[38] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 1993. ACM.

[39] T. Pang-Ning, M. Steinbach and K. Vipin. *Association Analysis: Basic Concepts and Algorithms*. Addison-Wesley, 2005.

[40] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York,USA, 2001.

[41] Anil K. Jain, M. Narasimha Murty, and P. J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3), 1999.

[42] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28, 1982.

[43] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1977.

[44] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1), September 2006.

[45] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3), 1973.

[46] Richard L. Taylor. Magnetic measurements of quadrupoles focusing magnets at SLAC. *the American journal of Physics*, 1991.

[47] A. Wolski. *Beam Dynamics in High Energy Particle Accelerators*. Imperial College Press, London, 2014.

[48] Tobias Hakan Bjorn Persson, Piotr Skowronski, Rogelio Tomas, and Thomas Nilsson. *Beam-Based Error Identification and Correction Methods for Particle Accelerators*. PhD thesis, CERN-THESIS-2014-324, June 2014. Presented 10 Jun 2014.

[49] J. Serrano and M. Cattin. The LHC AC Dipole system: an introduction. (Report No. CERN-BE-Note-2010-014), May 2010.

[50] P Castro. *Luminosity and beta function measurement at the electron-positron collider ring LEP*. PhD thesis, CERN-SL-96-070-BI, 1996. Presented on 25 Nov 1996.

[51] A. Langner and R. Tomás. Optics measurement algorithms and error analysis for the proton energy frontier. *Phys. Rev. ST Accel. Beams*, **18**(3), 2015.

[52] M. Kuhn, V. Kain, A. Langner, and R. Tomás. First K-Modulation Measurements in the LHC During Run 2. In *Proceedings, 4th International Beam Instrumentation Conference, IBIC2015*, 2016.

[53] M. Aiba et al. First beta-beating measurement and optics analysis for the CERN Large Hadron Collider. *Phys. Rev. ST Accel. Beams*, **12**, 2009.

[54] G. Vanbavinckhove, R. Tomás Garcia, L. R. Evans, and J. J. Engelen. *Optics measurements and corrections for colliders and other storage rings.* PhD thesis, Amsterdam U., CERN-THESIS-2013-022, January 2013.

[55] MAD-X. `http://cern.ch/mad`.

[56] H. Grote. TFS file format. `http://mad.web.cern.ch/mad/madx.old/Introduction/tfs.html`, 2002.

[57] B. W. Montague. Linear optics for improved chromaticity correction. LEP Notes - 1979, Jul 1979.

[58] T. H. B. Persson, Y. Inntjore Levinsen, R. Tomás, and E. H. Maclean. Chromatic coupling correction in the Large Hadron Collider. *Phys. Rev. ST Accel. Beams*, **16**(8), 2013.

[59] D. Jacquet, R. Gorbonosov, and G. Kruk. "LSA - the high level application software of the LHC - and its performance during the first three years of operation". March 2014.

[60] R. Calaga and R. Tomás. Statistical analysis of RHIC beam position monitors performance. *Phys. Rev. ST Accel. Beams*, **7**, 2004.

[61] R. Bartolini and F. Schmidt. "SUSSIX: A computer code for frequency analysis of nonlinear betatron motion". In *"Nonlinear and stochastic beam dynamics in accelerators: A challenge to theoretical and computational physics. Proceedings, Workshop, Lueneburg, Germany, September 29-October 3, 1997"*, 1997.

[62] M. Aiba, S. Fartoukh, A. Franchi, M. Giovannozzi, V. Kain, M. Lamont, R. Tomás, G. Vanbavinckhove, J. Wenninger, F. Zimmermann, R. Calaga, and A. Morita. First β-beating measurement and optics analysis for the CERN Large Hadron Collider. *Phys. Rev. ST Accel. Beams*, **12**, August 2009.

[63] A. Langner, J. Coello de Portugal, P. Skowroński, and R. Tomás. Developments of the Segment-by-Segment Technique for Optics Corrections in the LHC. CERN-ACC-2015-331, 2015.

[64] R. Tomás et al. Record low beta beating in the LHC. *Phys. Rev. ST Accel. Beams*, 15, 2012.

[65] N. W. Smirnow and I. W. Dunin-Barkowski. *Mathematische Statistik in der Technik. (Transled from Russian)*. Deutscher Verlag der Wissenschaften, Berlin, 1969.

[66] F. E. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1), 1969.

[67] Travis CI GmbH. Travis CI. `https://docs.travis-ci.com/`.

[68] L. van Riesen-Haupt, J. Coello de Portugal, E. Fol, A. Seryi, R. Tomás, et al. K-modulation developments via simultaneous beam based alignment in the lhc. In *8th Int. Particle Accelerator Conf.(IPAC'17), Copenhagen, Denmark, 14 -19 May, 2017*. JACOW, Geneva, Switzerland, 2017.

[69] E. Fol, J. Coello De Portugal, T. Persson, and R. Tomás. LHC MD 1988: Automatic coupling correction test. Report No. CERN-ACC-NOTE-2017-0010, February 2017.

[70] A. Boccardi, M. Gasior, R. Jones, P. Karlsson, and R.J. Steinhagen. First Results from the LHC BBQ Tune and Chromaticity Systems. Technical Report CERN-LHC-Performance-Note-007, Jan 2009.

[71] T. Persson and R. Tomás. Improved control of the betatron coupling in the Large Hadron Collider. *Phys. Rev. ST Accel. Beams*, **17**(5), 2014.

[72] E. H. Maclean, F. Carlier, S. Fartoukh, T. Persson, P. Skowronski, R. Tomás, and D. Wierichs. Demonstration of coupling correction below the per-mil limit in the LHC. (Report No. CERN-ACC-NOTE-2016-0053), Aug 2016.

[73] T. Persson, J. Coello de Portugal, M. Fjellstrom, L. Malina, J. Moeskops, G. Roy, P. Skowroński, and A. Szczotka. Review of LHC On-line Model Implementation and of its Applications. Report No. CERN-ACC-2016-243, 2016.

[74] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th international conference on Very Large Data Bases (VLDB'94)*. Morgan Kaufmann, 1994.

[75] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[76] S.G. Mallat and Zhifeng Zhang. Matching Pursuits with Time-frequency Dictionaries. *Trans. Sig. Proc.*

[77] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Technical report, 2008.

[78] David F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111), 1970.

[79] Martin Abadi et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016.

[80] M. Ester, H. Kriegel, and X. Sander, J.and Xu. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96. AAAI Press, 1996.

[81] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[82] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.