Industrial process error estimation by machine learning

MATTHIEU LOUSTAUNAU
Industrial process error estimation by machine learning

MATTHIEU LOUSTAUNAU

Degree Project in Scientific Computing (30 ECTS credits)
Degree Programme in Engineering Physics (300 credits)
Royal Institute of Technology year 2015
Supervisor at Saint-Gobain Recherche S.A., Aubervilliers:
Thierry Kauffmann
Supervisor at KTH: Michael Hanke
Examiner: Michael Hanke

TRITA-MAT-E 2015:85
ISRN-KTH/MAT/E--15/85--SE
Abstract

Performing a set-up on a complex machine may be difficult. This problem arises frequently for the industry, especially when the relation between input and output data cannot be defined precisely. Heavy methods of optimization may be used to perform a set-up. This master thesis investigate the possibility to use a machine learning approach on a specific machine. We study the structure of the relation between input and output data. We show the variations are smooth. We define a set of tests to evaluate future models. We design and test several models on simulation data, and select the best one. We design a strategy to use data in the best possible way. The selected model is then tested on actual data in order to be optimized.
Referat

Feluppskattning i industriella process genom maskininlärning

Acknowledgement

I would like to thank my industrial supervisor Thierry Kauffmann for all his support during the master thesis, as well as Anis Djedidi and Valentina Pozzoli for all the help and work their provided during this project.
Chapter 1

Context

In this master thesis, we consider a set of $N$ samples $\{(P_j, Q_j)\}_{j \in \{1, \ldots, N\}}$. Let $m \in \mathbb{N}, n \in \mathbb{N}$ so that:

- $\forall j \in \{1, \ldots, N\}: Q_j = (Q_{j1}, \ldots, Q_{jm}) \in \mathbb{Q} = [0; 1]^m$
- $\forall j \in \{1, \ldots, N\}: P_j = (P_{j1}, \ldots, P_{jn}) \in \mathbb{P} = [-1; 1]^n$

For $j$ in $\{1, \ldots, N\}$, $P_j$ is a relative variation and is expressed in percent$^1$. For this work, $m$ is 40 to 50 times larger than $n$:

- $n = 7$
- $m \approx 400$

The aim of this master thesis is to investigate the feasibility of a machine learning approach to address the problem$^2$:

(map $\mathbb{Q}$ to $\mathbb{P}$, based on a small set of data ($N \approx 40$).

Due to the number of samples and the number of dimensions of $\mathbb{P}$ and $\mathbb{Q}$, this problem is complex to handle with standard methods. This is why we investigate a machine learning approach.

This report presents the methodology, the different tools used during this master thesis, as well as some important results. Based on simulated data we show that a machine learning approach can provide satisfying results considering the number of samples we can afford. We also investigate the influence of the distribution of the collected data. We also show how our algorithm performs on a preliminary test made on real-life data.

$^1$Detailed information regarding the origin of the data $\mathbb{P}$ and $\mathbb{Q}$ are confidential and will not be disclosed in this report.

$^2$In practice, this corresponds to an inverse problem and there is a methodology to build an operator mapping $\mathbb{P}$ to $\mathbb{Q}$. This operator will be used to generate simulated data. See Chapter 2 for more information about how we use simulated data.
Chapter 2 gives elements regarding the way this project has been handled. Concepts and tools regarding machine learning and the types of models we have investigated are provided in Chapter 3, as well as in the appendix. Chapter 4 defines the tools and methods used during the master thesis to design a reliable model and a strategy to process data. In Chapter 5, we design, evaluate and compare different models. We also give result of a preliminary test made on real-life data. Chapter 6 presents some perspectives for future work.
Chapter 2

Methods and resources used during the master thesis

In this chapter we describe the different steps of the master thesis, as well as the different resources used. It gives a quick overview of the organisation adopted during the whole project.

2.1 Planning

The project was divided into several phases, or work packages:

- WP0: readings and getting ready for the project
- WP1: study of the data structure
- WP2: evaluation and selection of models by simulation
- WP3: optimization and evaluation of the final model

Phase WP1 was designed to get a good understanding of the data structure. Methods related to data projection were applied. This was very helpful when designing or model and our strategy to train it. For instance we could observe that a linear model could be a good approximation of the relation between $Q$ and $P$.

In phase WP2, different types of models were trained, evaluated and compared on simulated data. Since data acquisition is expensive –regarding both time and resources– the number of samples we can use is low: we want to have results after training on 40 or 50 examples. This means that actual data cannot be used during the phases WP1 and WP2, as both are designed to give us a comprehensive understanding of the behaviour of the different models. Therefore all data for WP1 and WP2 are simulated.
CHAPTER 2. METHODS AND RESOURCES USED DURING THE MASTER THESIS

The best algorithm of phase WP2 is then optimized and tested on actual data in WP3. During this phase we also adapt the methods defined in WP1 and WP2. This report will cover WP1 and WP2, as well as a part of WP3 – WP3 is still an ongoing phase.

2.2 Python and Scikit-learn for machine learning and data visualization

We used Python 2.7 during the whole master thesis. It provides good speed and user-friendly tools, along with a strong and wide documentation. This leads to a code easy to modify rather than an optimized but heavy code.

Another reason for using Python is that it allows us to use scikit-learn library [4]. Scikit-learn includes major machine learning algorithms for both classification and regression problems. This library is implemented in C or C++ and uses a Python interface. This provides the computing speed of C language while enabling the developing speed of Python. All algorithms – for preprocessing, learning, cross-validation and prediction – come in a standardized way, which allow us to re-use our code. The consequence is that less time is spent on coding, which gives more time to analyse the results. Additional Python libraries, such as Theano, Pylearn2 and scikit-neuralnetwork are required to build and evaluate neural network algorithms.

This master thesis aims at finding an adequate model to map $Q$ to $P$, based on machine learning. This implies building and testing several models, as well as developing tools to process the data and visualize the influence of various types of parameters. Standards algorithms are already implemented in scikit-learn, which is time-saving. However we need visualization tools in order to design an accurate model, select adequate values for the parameters used by the algorithms, or simply visualize the performance of the model after training. Therefore we designed our own visualization routines based on matplotlib and bokeh libraries.
Chapter 3

Mathematical background

This chapter presents briefly the different concepts, tools and machine learning algorithms used during the master thesis. For a description of the tests and scores developed during the master thesis, see Chapter 4 and 5.

Section 3.1 presents the different notions used all along this work, based on definitions given in the recommended book *The elements of statistical learning : data mining, inference, and prediction* by Hastie, Tibshirani, and Friedman [2] and the reference course provided by Ng [3]. Key concepts are described according to Hastie, Tibshirani, and Friedman [2, chap.2] and Ng [3, lesson 1]. We give some elements of method to address our problem based on Hastie, Tibshirani, and Friedman, [2, chap. 3] and Ng, [3, lesson 4 and lesson 5]. This includes a general methodology, a description of the troubles that may arise, and some basic tools to enhance the performance of a machine learning algorithm. Eventually, we give a quick description of the different algorithms used during this master thesis. This work is based on Hastie, Tibshirani, and Friedman, [2, chap. 3 and chap. 8] as well as Ng, [3, lesson 4 and lesson 5]. For Support Vector Regression, we give a basic presentation in this chapter and slightly more detailed description in the appendix. All reference to Support Regression are based on a tutorial provided by Smola and Schölkopf [7, section 1, 2 and 3].

**Machine Learning** is a branch of science which allows computers to learn from data and make predictions. The key point is to allow computers to act without being explicitly programmed. The goal is to provide computers that can generate automatic data-driven decisions, with programs that can evolve when exposed to new data Ng, [3, lesson 1 and 3].

A machine learning algorithm learns some patterns from a set of data – a so-called **training set**. It develops programs able to give predictions based on new data – a **test set**. It can also extract information from that data to improve the algorithm’s own understanding. Thus the new data is not used as test then dis-
carded, it is added to the training set. The idea is that we constantly add data to the training set: the database used by the machine learning algorithm evolves, therefore the decisions taken by the algorithm must change accordingly.

This type of approach can be used for various types of problems. It relies on mathematics and statistics skills as well as programming skills. Contrary to traditional research, it does not need comprehensive expertise on the problem, but relies exclusively on patterns extracted from collected data.

Machine Learning is usually divided in two main domains: supervised and unsupervised learning. Supervised learning aims at inferring a function between a so-called features vector and a so-called target or label. Unsupervised learning is used to detect hidden structures or patterns in the data. Our goal is to map two spaces, and falls in supervised learning.

Supervised learning is usually also divided in two main domains: classification and regression. Classification means that the output space of the predictive model – the target or label space – is a discrete space. For instance this is used to classify mails as spam or non-spam. Regression means that the output space takes continuous values. This could concern a model aiming at predicting houses prices. Our problem is a regression problem and therefore must addressed as such.

**Conclusion:** In this master thesis we focus on a branch of machine learning called **supervised learning**. Each sample in the data provided to the algorithm contains features and targets. The features are the input of the model – hence, $Q$ –, the targets are the output of the model – $P$ – and the algorithm finds a pattern that maps the features to the targets.

### 3.1 Regression by machine learning

Let $(x^j, y^j)_{j \in \{1, ..., N\}}$ be a set of data we want to fit. Let $x^j \in E = \mathbb{R}^m, \forall j$ and $y^j \in \mathbb{R}, \forall i$. We can write $\forall J \in J$, $x^j = (x^j_1, x^j_2, ..., x^j_m) = (x^j_i)$

In order to **design** a model, we define a weights vector $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$. By convention we call $x_0$ the **intercept**. We take a class of function $h$ that can be parametrised by $\theta$. The designed model is denoted as $h_\theta$.

**Fitting** the data means that we try to find $\theta$ that minimizes a cost function $J : \theta \mapsto J(\theta)$. The cost function evaluates the error of prediction on the training set.

For instance, the machine learning algorithm may solve a least square problem to find the optimal $\theta$.

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} (h_\theta(x^j) - y^j)^2$$

(3.1)
3.1. REGRESSION BY MACHINE LEARNING

Figure 3.1 gives an example of a polynomial model trained by machine learning on a training set. We will use this example to illustrate some of the key concepts regarding regression by machine learning.

**Figure 3.1.** We try to fit the ‘true function’ \( x \mapsto \cos(1.5\pi x) \). **Top:** we collect data in a training set. **Bottom:** we want to build an estimator using a 4 degree polynomial function. The machine learning algorithm estimates the values of the different coefficients.

The machine learning algorithm estimates the true function without specific
knowledge. Although this could look like a ultimate and wonderful technology, one must remain careful. Fitting the data on the training set does not mean that the model will provide accurate predictions on new data. This is the purpose of section 3.1.1

3.1.1 Trade-off between bias and variance and the problem of underfitting and overfitting

This section presents the notion of bias and variance for a machine learning algorithm, then extends these concepts to get the notions of underfitting and overfitting. This parts is based on Hastie, Tibshirani, and Friedman [2, p.37, 'Model Selection and the Bias-Variance Tradeoff']. Consider an estimator $h_\theta$ that fits a set $(x_j, y_j)_{j \in \{1,...N\}}$. Let $f$ so that $\forall x \in E$, $y = f(x) + \epsilon$. $f$ is the actual model we approximate by $h_\theta$, and $\epsilon$ is the observation noise. We assume the noise is distributed according to a zero-mean Gaussian distribution:

$$E[\epsilon] = 0$$

One may derive the expected prediction error for new a prediction $x^*$ and a new observation $y^*$ as:

$$E[(h\theta(x^*) - y^*)^2]$$

It comes:

$$E[(h\theta(x^*) - y^*)^2] = E[h\theta(x^*)^2 - 2h\theta(x^*)y^* + y^{*2}]$$

$$= E[h\theta(x^*)^2] - 2E[h\theta(x^*)y^*] + E[y^{*2}]$$

Eventually, after developing and sorting all terms:

$$E[(h\theta(x^*) - y^*)^2] = E[(h\theta(x^*) - E[h\theta(x^*)])^2]$$

$$+ (E[h\theta(x^*)] - f(x^*))^2$$

$$+ E[(y^* - f(x^*))^2]$$

Three distinct terms can be identified:

- $E[(h\theta(x^*) - E[h\theta(x^*)])^2]$ : the variance of the estimator
- $(E[h\theta(x^*)] - f(x^*))^2$ : the square of the bias of the estimator
- $E[(y^* - f(x^*))^2]$ : the variance of the new test target – noise in observation

The last term does not depend on $h_\theta$ and is beyond our control. Generally, as the model complexity increases the variance tends to increase and the bias tends to decrease. Hence there is a bias-variance trade-off. We want to avoid two situations: underfitting and overfitting.
3.1. REGRESSION BY MACHINE LEARNING

**Underfitting** corresponds to a situation where the estimator has a high bias and low variance.

\[ E[(h_\theta(x^*) - E[h_\theta(x^*)])^2] < (E[h_\theta(x^*)] - f(x^*))^2 \]

It happens when the estimator is unable to fit the training data. This occurs for instance when the type of estimator we are looking for is 'too simple'. This is the case of the degree 1 estimator on Figure 3.2. In this case, large error in both training and test set can be observed. This may happen because the model is not adapted: we use a linear function to fit a cosine on a large interval. It may also happen because we do not use enough features.

**Overfitting** corresponds to a situation where the estimator has low bias and high variance.

\[ E[(h_\theta(x^*) - E[h_\theta(x^*)])^2] > (E[h_\theta(x^*)] - f(x^*))^2 \]

It happens when the estimator is 'too complex'. It fits the training data quite well, but cannot produce accurate prediction on test data. This may occur because the model involves too much variance, or because we use too many features. The example of a degree 15 estimator on Figure 3.2 illustrates this problem.

![underfitting and overfitting](image)

**Figure 3.2.** We want to fit the same function as in Figure 3.1. Firstly, we use a degree 1 polynomial. This is 'too simple' and cannot fit the training data properly. We face a problem of underfitting. We also use a degree 15 polynomial. It fits the training data well but cannot provide accurate prediction for new samples. The model is 'too complex' and we face overfitting problem.
During the master thesis, two tools were used to address underfitting and overfitting problems. We used regularisation to control the variance and the complexity of the estimator and we used cross-validation to ensure good choice of the parameters. We briefly present these two tools.

3.1.2 Control of the complexity of a model by regularization

Regularization is a way to tune and control the level of complexity of the model we want to train to improve prediction accuracy on new data. We usually introduce a penalisation term for $\theta$ in the cost function the machine learning algorithm must minimize. A strong or high regularization means that we force values in $\theta$ to be small. Increasing the regularization should reduce the variance of the model and overfitting. A weak or low regularization means that all coefficients in $\theta$ can take a large range of values. Reducing the regularization should increase the model variance and can be used against underfitting.

Figure 3.3 The key point is to find a value for the regularization that gives a good compromise in the trade-off between bias and variance. We introduce now the method that helped us dealing with this type of problems: cross-validation.

Figure 3.3. We use the same true function as in Figure 3.1. We fit this function with a degree 15 polynomial. To avoid overfitting, we use regularization : this force the different coefficients of the polynomial to remain small. This decrease the variance of the model and the effect of overfitting.
3.1. REGRESSION BY MACHINE LEARNING

3.1.3 Parameters optimization by cross-validation

During the master thesis we used cross-validation to tune the parameters of the fitting algorithm. It can be used to select the degree of a polynomial estimator, the value of the regularization parameter, or any other specific parameter of the algorithm.

Two methods have been considered two identify adequate values for the different parameters of the algorithm. The first option is to divide the training set in two subsets: one subset is used as training set and the other is used as ‘cross-validation’ set. We train the model on the train subset and see how it performs on the cross-validation subset. We determine the parameters of the algorithm that yields best accuracy on the cross-validation set. This works well on large datasets.

For small datasets, we use a generalized version of this method, called K-fold cross-validation. The training set is divided into K folders. The model is then trained K times. At each iteration, one of the folders is used for cross-validation while others are used for training, and each folder is used once and only once for cross-validation.

Using K-Fold cross-validation ensures accurate estimation of the performance of the estimator (Refaeilzadeh, Tang, and Liu, [5]).

However, K-fold can lead to several troubles. Dividing the training set into K folds means that we compare only small samples of performance estimation. Since the training sets are smaller, we may face high bias which makes comparison harder for model selection. The training sets defined by the folds overlap – each fold is used once for test and K-1 times for training. This can lead to an underestimation of the variance of the performance of the estimator (Refaeilzadeh, Tang, and Liu, [5]). Fortunately, these elements did not significantly affect the master thesis.

We used both approaches in practice. The first approach is used to have a quick understanding of the model and gives a first approximation for the optimal parameters, the second one is used when robust cross-validation is needed, namely during an optimization phase.

Regularization and cross-validation are key tools to control the variance of the estimator. Cross-validation can be used to select any kind of parameters of the estimator. The algorithm must learn from the data: we do not want to introduce much a priori knowledge of the problem nor force a specific coefficient to take a specific values. We only need knowledge to choose the class of functions used to build our predictive model.
We use the same true function as in Figure 3.1. We have kept away some data in a cross-validation set – CV set. We can fit our model on the training set and see how it behaves on the cross-validation data. This simple test tells us that a degree 4 polynomial should give best performance to estimate the true function.

3.2 Defining the different algorithms used during the master thesis

In this section we motivate the fact that we are looking for predictive models with low variance. Then we present the different machine learning algorithms used during the master thesis.

Correlation between $Q$ and $P$ : Based on simulation data, we did a preliminary study of the correlation between our features $Q$ and our targets $P$. Figure 3.5 links $P$ and relative variations of $Q$. In this test, variations for all parameters are equal to zero, except for parameter $P_1$. This shows that the relation between variations of $Q$ and $P_1$ can be approximated by a linear operator, although some non linear effects may arise for large variations – greater than 20%. Similar results can be derived for all other parameters – $P_2, \ldots, P_7$ and feature in $Q$. This gives us knowledge regarding the class of function to use for the predictive model: we must use a 'simple' predictive model. The variance of the model should remain small. For instance linear or polynomial estimators should work well for this type of situation.

We present now some the different types of estimators used during the master
3.2. DEFINING THE DIFFERENT ALGORITHMS USED DURING THE MASTER THESIS

Figure 3.5. All parameters are kept to zero except $P_1$. We plot the correlation between $P_1$ and some components of the variations of $Q$. We control $P_1$ so that the variations remain in $[-30\%; +30\%]$. The correlation between the variations of $P_1$ and $Q$ can approximated with a linear operator. Some non-linear effects seem to arise for variations greater than $20\%$. We can derive similar results for each feature in $Q$ and each target in $P$.

...thesis. When possible we specify to the machine learning algorithm that the predictive model called $h$ or $h_\theta$ should be linear or polynomial. We start with algorithms ensuring ‘simplicity’ i.e low variance, then we present more and more ‘complex’ algorithms, and eventually present an algorithm based on a multiple steps approach to give an estimate of $P$.

3.2.1 RIDGE regression

RIDGE regression resorts to regularization to control the complexity of the predictive model. It involves a specific cost function, defined as:

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} (h_\theta (x^j) - y^j)^2 + \alpha \|\theta\|^2$$

where $\|\|\|$ is the $L^2$-norm. The sum scores the error of prediction on the training set. The second term $\alpha \|\theta\|^2$ controls the amplitudes of the coefficients in $\theta$. The regularisation parameter is used to handle a trade-off between bias and variance and deal with underfitting or overfitting problem [2, p. 59, 'Ridge Regression']. In practise, a high value of $\alpha$ penalises large values in $\theta$. Thus, it drives coefficient...
in $\theta$ to remain small. This decreases the variance of the model and can be used as a leverage against overfitting. A small value of $\alpha$ will let the model fit the data perfectly: reducing $\alpha$ can help to solve a situation of underfitting.

Effective methods exist to find the adequate $\theta$. This model fits the data using all features, therefore it can fit small variations well. It ensures low mean error (low bias) in the training set. The main advantage of this machine learning algorithm is its 'simplicity': we can use a linear operator for $h$ and avoid complex operators. However, learning small variation can mean learning noise. This type of algorithm may not be robust to noise since. This is why cross-validation for parameters selection is crucial with this algorithm.

3.2.2 Lasso regression

Lasso regularisation – also called lasso regression – is similar to ridge regularisation. We simply replace the $L^2$-norm with the $L^1$-norm. The cost function becomes now:

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{m} (h_\theta(x^j) - y^j)^2 + \alpha \|\theta\|_{L^1}$$

Finding $\theta$ that minimizes $J$ can be written as an optimization problem with constraint. For a so-called 'shrink parameter' $t \in \mathbb{R}$:

$$\begin{cases} \min_{\theta} \frac{1}{2} \sum_{j=1}^{m} (h_\theta(x^j) - y^j)^2 \quad \text{(lasso)} \\ \|\theta\|_{L^1} < t \end{cases}$$

The constraint region has corners and edges. If the solution $\theta^*$ is in one of the corners, then it has at least one parameter equal to zero [2, p.64, "The Lasso"]. This is a major difference with ridge regression. Lasso regularisation can force some weights in $\theta$ to zero while ridge regularisation keeps all weights in $\theta$ to non-zero values. This means lasso regularisation can help for feature selection by dropping irrelevant features. The number non-zero coefficients of a lasso estimator can be considered as an estimate of the 'degrees of freedom' of the lasso. It can be controlled via the regularisation parameter $\alpha$ or the shrink parameter $t$ (Zou, Hastie, and Tibshirani, 2007[9], and Tibshirani and Taylor, 2012 [8]).

This type of algorithm allows us to use simple predictive models: we can use a linear model. To a certain extent, it performs a basic feature selection. However, forcing some features to zero may turn to be ineffective as we lose some information.

3.2.3 Support Vector Regression (SVR)

We use $\varepsilon$-Support Vector Regression during the master thesis. This machine learning algorithm is complex, we describe here briefly how our implementation works.
3.2. DEFINING THE DIFFERENT ALGORITHMS USED DURING THE MASTER THESIS

We give more details in the appendix. This section and the appendix are based on [7, p.1 to 3]; this tutorial is highly recommended to understand Support Vector Regression.

Suppose we are given a set of data \( \{(x_1, y_1), ..., (x_n, y_m)\} \) where the input belong to \( E = \mathbb{R}^n \) and y belongs to \( \mathbb{R} \). With \( \varepsilon \)-SV regression, we try to find a function \( f(x) \) with a small deviation – less than \( \varepsilon \) – from the target \( y \) for the whole training set. We also want this function to minimize the variance as much possible: \( f \) must remain 'simple' or 'flat'. Therefore we ignore error smaller than \( \varepsilon \) but strongly penalise larger errors. \( f \) is defined as:

\[
f(x) = \langle x, \theta \rangle + b, \quad \text{where } x \in E, b \in \mathbb{R}
\]

The dot product represents a so-called kernel function. It can be the usual inner product in \( E \), but can belong to various classes of functions [7, Smola, p.3].

Forcing \( f \) to remain flat can be done by solving a convex optimization problem:

\[
\begin{align*}
\min & \frac{1}{2} \|\theta\|^2 \\
\text{subject to } & |y_i - \langle \theta, x_i \rangle - b| < \varepsilon
\end{align*}
\]

The kernel function is used to remap the space of input to a features space so that the 3.4 becomes a convex optimization problem [7, Smola, p.2]. The SVR algorithm computes all the parameters of the estimator by solving the dual problem and using Lagrange multipliers.

Further explanation is given in A. For a more comprehensive understanding of SVR method, the interested reader may refer to Smola and Schölkopf, 2003 [7], Borchani et al., 2015 [1].

The implementation used during the master thesis includes regularisation. Therefore the algorithm has two main parameters: regularisation and tolerance.

The main advantage of SVR are 'the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by optimising the margin.' (Shawe-Taylor and Cristianini (2004)).

With our specific \( \varepsilon \)-implementation one may add its robustness to noise: using a tolerance parameter means that we ignore small variations – that we assume to be caused by some noise. It is related to the margin optimisation mentioned by Shawe-Taylor and Cristianini.

However, it may not be as efficient as ridge regression concerning small variations. It would require more samples to learn them – with a very small training set, small variations would be considered as noise and discarded. Is can also be sensitive to regularisation and tolerance set-up, and in some case is not able to solve the minimization problem.
3.2.4 Ensemble-based method

Ensemble-based method use a bunch of estimators. Each estimator gives a prediction value, and the average value is given as the final prediction. This should ensure that only relevant trends are taken into account, other trends being discarded thanks to the averaging. This concerns for instance random forest, or extratrees. Each algorithm builds a set of decision trees. They select features thanks to a score function –random forest– or randomly –extratree. Our implementation uses bootstrapping method to feed each estimator. Let \( N \) the number of samples available in the training set. For each estimator, we select randomly with replacement \( N \) samples from the training set and train the estimator – or tree – on this subset.

![Diagram of ensemble-based method](image)

**Figure 3.6.** With an ensemble based method, a bunch of estimators – for instance trees – are built. The predictive model takes the average prediction of all estimators and returns it as final prediction.

The main advantage of this type of machine learning estimators is that absolutely no prior knowledge of the problem is required. The algorithms extract all the patterns that seems relevant. However, it may require a lot of data to show good accuracy of prediction. A comprehensive study of this type of estimators is beyond the scope of the master thesis, and the interested reader may refer to Hastie, Tibshirani, and Friedman [2, chap. 9, ‘Additive Models, Trees, and Related Methods’].
3.2. DEFINING THE DIFFERENT ALGORITHMS USED DURING THE MASTER THESIS

3.2.5 Q(P) estimator

![Diagram](image)

**Figure 3.7.** With this model, we build an estimator to map \( \mathcal{P} \) to \( \mathcal{Q} \). We use then a minimization function when we need to find the value of \( \mathcal{P} \) corresponding to a value \( Q_{\text{target}} \).

This estimator investigates an alternative approach to map \( \mathcal{Q} \) to \( \mathcal{P} \). This approach is based on the method developed before this master thesis: we want to find a new way to build an operator between \( \mathcal{Q} \) and \( \mathcal{P} \).

Firstly, we build a predictive model mapping \( \mathcal{P} \) to \( \mathcal{Q} \) and call this predictive model \( G \). Then, we consider a target value for \( \mathcal{Q} \) and call it \( Q_{\text{target}} \), and we define the optimization problem:

\[
\min_{\mathcal{P}} \| Q_{\text{target}} - G(\mathcal{P}) \|_2^2 \tag{3.5}
\]

By solving this optimization problem, we link \( Q_{\text{target}} \) to a certain \( \mathcal{P} \), and can provide an estimate for the relation between \( \mathcal{Q} \) and \( \mathcal{P} \).

In this approach, the inverse problem ‘mapping \( \mathcal{Q} \) to \( \mathcal{P} \) is handled via optimization techniques. Our predictive model is an approximation of the direct function.

\(^1\)Note that the assumptions made to ensure this problem can be solved, as well as the optimization strategy used are not discussed in this report.
To build this estimator, we look for polynomial models with a degree smaller than or equal to 3 and we use regularization techniques. In this report, we present results for a multi-task lasso regularization. This type of regularization is very similar to lasso. It simply considers all outputs at once, whereas for a standard lasso we consider that all outputs independent from each other. The main consequence is that for a multi-task lasso, we use the exact same features to compute the target.

The main advantage is that similar techniques are currently used in real-life. We know this type of method can give satisfying results. Yet, the call to an optimization function may be computationally expensive. Contrary to other machine learning algorithms considered during this master thesis, we have here more targets than features. This means that we may need more than $\approx 40$ samples to avoid overfitting.
Chapter 4

Design of a scoring strategy to analyse the performance of a machine learning algorithm

In this chapter, we present all the methodology used during the simulation phase – WP2. All results and conclusion will be given in chapter 5.

During the phase of simulation different models are built, trained and compared. As specified in chapter 2, most of the classical algorithms are implemented in an efficient way in scikit-learn. Therefore, most time is spent on the possible ways to get good results for each algorithms.

We tried to cover very different types of situations: small or big variations, with or without noise. The point is to find a reliable model but also a strategy to train it: is it better to use small or big variations? How much should we fight against the noise?

Eventually, the most promising algorithm is selected in order to be optimized and trained on actual data. This means we may have to deal with some trade-off and define what matters the most for the algorithm.

As far as we are concerned, the most important criterion is the accuracy. We need an algorithm able to make good predictions – basically a low mean error of prediction. The second aspect is the robustness, namely the ability of the algorithm to remain stable under different conditions – we do not want a large error variance. Eventually, time performance is also taken into account, as we also want fast predictions.

We build evaluation routines to confront the different algorithms and keep the model that guarantees the best standards. We present now some parameters used to generate data during the whole simulation phase. Then we present the different tools used to score and evaluate the resulting predictive models.
CHAPTER 4. DESIGN OF A SCORING STRATEGY TO ANALYSE THE PERFORMANCE OF A MACHINE LEARNING ALGORITHM

Control of simulated data: Simulation phase gives us some control on the data used to feed and test our algorithms. We can generate \( P, Q \) couples according to a specific distribution in \( P \). In our case we use different parameters to control the distribution of the data or to denote a specific set.

We could control the size of the different sets, that is the number of samples each set could receive.

Another useful parameter is the maximum amplitude of a set. We define it as a bound value \( C \) for \( P \). It is the limit of variations relative to \( P_{ref} \) allowed in the set, where \( P_{ref} \) is the set of parameters ensuring we have the desired value \( Q = Q_{ref} \). The maximum amplitude is expressed in percent as it is a relative variation.

We also carried tests on different types of distribution, as we do not know a priori the distribution of data to be collected. This parameter though was only briefly investigated, as it is hard to carry on a comprehensive investigation on its influence.

4.1 Accuracy performance scoring strategy

Accuracy is by far our main concern. In order to be selected, an algorithm needs to provide accurate results. A good way to check this is to compute the mean error of prediction. Since the number of sample is low, it is relevant to understand if we benefit from adding some samples to the training set. Therefore learning curves are of great use here. We plot the mean error of the prediction along each axis of \( P \) – i.e 7 curves – for different sizes of training sets.

Training sets are generated after a uniform distribution, where all parameters \( P_1, \ldots, P_7 \) are independent and in identically distributed, with a maximal amplitude of 15%. The number of samples and the value of the maximum amplitude correspond to the data we expect to collect in real life. The main uncertainty lies in the uniform distribution, and will be discussed briefly in 5.3. We control the size of the training sets. The sizes of the different training sets belongs to \{20, 30, 40, 50, 60\}. The test set is also generated with an iid uniform distribution. The typical size for the test set is 100. We investigate two types of test sets:

- 'small' variations: the set has a maximal amplitude of 5%
- 'large' variations: the set has a maximal amplitude of 30%

Sets labelled with 'large variations' have also samples with a small amplitude. They simply cover a wider range of possibilities for variations values, although it represent a coarse mesh.

Note that in each case, cross-validation is used to improve model performance. We use a grid of parameters and keep the combination giving best results. The parameter grid remains quite coarse: we make sure the model is close to its best
4.2 ROBUSTNESS PERFORMANCE SCORING STRATEGY

but it still leaves room for improvement – especially for WP3 phase. The point is to find a trust region for the model parameters.

Learning curves are given and discussed in 5.1

4.2 Robustness performance scoring strategy

A second criteria that need to be checked is the robustness of the algorithm. We want to know if the fitted model can be embedded in a algorithm to find the solution to the general problem ?? . One method is to apply successive corrections suggested by the model until we reach desired quality standards. We may not be able to reach these standards for all initial errors, therefore we estimate a trust region.

The diameter of this region is given by a coefficient called limit of convergence. For starting values such that all components in Q are in the range [-limit of convergence; limit of convergence], the algorithm should find a way to the optimal values – $P_{ref}, Q_{ref}$ with a rate of success greater than 80%.

4.3 Influence of the maximum amplitude in the training set

The performance of the algorithm highly depends on the distribution of the data in the training set. This is also investigated in order to be sure our model performs to its best. In our case, we have two parameters to generate a training set: the type of distribution and the maximal amplitude. The two parameters have been investigated. Different types of distribution have been tested but no significant improvement were noticed. The main element to investigate is therefore the maximal amplitude.

The limit of convergence coefficient gives some perspectives, as it provides a trust region for the model and for different values of the maximal amplitude. However, it does not tell anything about the number of iterations needed, or the norm of the residual error.

We compute the ratio between the mean error of prediction and the mean value of the initial error.

$$\text{error_amplification} = \frac{\text{mean squared error of prediction}}{\text{mean square initial error}}$$

It shows how fast an error can be reduced.

This ratio is computed for different values of maximal amplitude for both training and test sets. This provides us information in order to design a strategy to collect real data.
CHAPTER 4. DESIGN OF A SCORING STRATEGY TO ANALYSE THE PERFORMANCE OF A MACHINE LEARNING ALGORITHM

We will also discuss briefly about the impact of the noise on our data. It is not a criteria of selection regarding the model, as robustness to noise is similar for all models considered here, but remains relevant regarding the way we collect data.

4.4 Time performance

Eventually, our algorithm has to perform learning and prediction steps in a 'decent' time. This criteria is rather loose, and is mainly used as indication for user. Average performance for each algorithm will be given for both training and prediction steps. Typically, we use 40 examples to generate the training set and 100 for the test set.

4.5 Models and algorithms taken into account

We tried different ways to design a model that could fit the data. Since a linear approximation was used before the beginning of the master thesis to solve the general problem, we start by designing a linear model. We present results for the two types of regularisation –ridge and lasso – used to fit the data.

Then, we use Support Vector machine Regression with a polynomial kernel, in order to catch non-linear effects on large variations.

Ensemble-based algorithms have also been tested, as way to fit the data without assuming much about the type of relation between $P$ and $Q$.

A model designed via an estimator mapping $P$ to $Q$ is also presented.
Chapter 5

Analysis of the performance of different algorithms

Sections 5.1, 5.2, 5.3, 5.4 dedicated to simulation results. We present results regarding accuracy and make a first selection of algorithms. We analyse the robustness of the selected models. This is followed by an analysis of the distribution of data. We give some recommendation in order to process actual data. We also provide some hints about computational time involved during the training/predicting process.

Section 5.5 is dedicated to an analysis of the first results obtained on real-life data.

In this chapter, we consider sets of data \((P^j, Q^j)_{j \in \{1, \ldots, N\}}\), where \(\forall j \in \{1, \ldots, N\}; P^j \in \mathcal{P} = \mathbb{R}^n, Q^j \in \mathcal{Q} = \mathbb{R}^m\). A set is defined in this work by three parameters:

- the number of samples : \(N\)
- the maximum amplitude : \(\forall j \in \{1, \ldots, N\}, \|P^j\|_\infty < \text{maximum amplitude}\)
- the type of distribution of the set : uniform, Gaussian, ...

We want to build a relation between \(\mathcal{Q}\) and \(\mathcal{P}\). We use simulated data to evaluate the different models. In practice we will use the predictive algorithm in an iterative way. Starting from an initial guess, we use the corrections \(P\) provided by the predictive model to find a set \(P\) corresponding to a value \(Q_{\text{target}}\)

5.1 Analysis of the accuracy

Accuracy is evaluated on every model we have taken into account. We use learning curves to analyse and compare the behaviour of each model.

\(^{1}\)We recall that mapping \(\mathcal{Q}\) to \(\mathcal{P}\) is in our case an inverse problem : we look for a value \(P\) so that \(Q(P) = Q_{\text{target}}\)
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

We give results for a linear model, fitted with two kinds of regularisation – ridge and lasso. Then we give results for a model fitted using support vector machine and a polynomial kernel. Learning curves are also provided for a random forest algorithm. We tried different ensemble-based methods, where results were the same as in the case of random forest. An alternative model, the so-called ‘Q(P) estimator’, where we fit $Q$ as a function of $P$ is also investigated.

Note that for all learning curves presented here, we used training sets with the same maximum amplitude: 15%. This type of set corresponds to the data we expect to collect in real life. We compare if the designated algorithm can perform on new data sets where the maximum amplitude is equal to 5% and 30%\(^2\). We used the expected squared error as score and used absolute error. For a test set with a maximum amplitude of 5%, we want the mean error of prediction to remain smaller than 1% on average. For a test set with a maximum amplitude of 30%, the accuracy criterion is loose. We do not expect predictions to be as accurate as for small variations, and simply take the best models.

5.1.1 RIDGE regression

We use a linear model and a RIDGE regression. The regularisation parameter is set using K-Fold cross-validation. Figure 5.1 shows that we can achieve good results on a test set with a maximum amplitude of 5% – small variations – even if the training set is small. Adding data to the training set is a good way to improve the accuracy of our model.

The learning curves plotted on Figure 5.2 for a test set with a maximum amplitude of 30% gives a lot of information about the behaviour of this type of algorithm.

Since the designed model is linear, it cannot generalize all non-linear effects for such variations. The error of prediction on test data here is high compared to the error of prediction on training data. We have low bias and high variance. Increasing the value of the regularisation parameter should decrease the variance. However, doing so could lead to a loss of accuracy on small variations, for instance on a test set with a maximum amplitude of 5%. There is actually a trade-off between accuracy on small variations and accuracy on large variations.

In this case, we can simply consider adding a lot more elements in the training set. Increasing our database to have a bigger training set should help for generalisation on test data.

\(^2\)The so called ‘training error’ which is the mean error on the training set is only provided as a way to understand if the machine learning algorithm is able to fit the training data. Comparison of the training and the test error presented here may not be relevant to conclude that we face underfitting or overfitting situation: we are not exactly comparing the same thing. See appendix A for a concrete example of this situation.
5.1. ANALYSIS OF THE ACCURACY

Figure 5.1. learning curve for a RIDGE, with maximum amplitude of 5% in the test set. It shows good results even for a very small training set (≈ 40 samples). The training set has a maximum amplitude of 15%. The machine learning algorithm fits data in the training set and is able to provide accurate predictions on the test set.
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

Figure 5.2. learning curve for RIDGE. The test set has a large maximum amplitude (30%). The expected error of prediction is larger than what can be observed on Figure 5.1. The mean error on training data is low. This situation shows that the machine learning algorithm fits training data, where the maximum amplitude is 15%, but cannot give accurate predictions for large variations.
5.1. ANALYSIS OF THE ACCURACY

5.1.2 Lasso regression

![Figure 5.3. Learning curve for lasso. The test set has a maximum amplitude of 5%. Parameter $P_5$ cannot be well-predicted compared to other parameters. This can be explained by the fact that on the training set with a maximum amplitude of 15%, the lasso algorithm has some troubles fitting data relative to $P_5$. One may also notice that the training error seems larger than the test error for some of the parameters such as $P_4$, $P_6$. It simply means that the model fitted on the training samples is a lot more effective on small variations than on large variations. This is confirmed by Figure 5.4.](image)

We design a linear model and use lasso regularisation to fit the data in the training set. Lasso regularisation implies a feature selection. Since $Q \in \mathbb{R}^m$ and $P \in \mathbb{R}^n$, this could be a good way to discard unnecessary features and avoid overfitting. Some parameters seem hard to predict – the prediction error on the test set does not decrease quickly. In the case of small variations – with a maximum amplitude of 5% – this happens for parameters $P_3$ and $P_5$. For a maximum amplitude of 30% this happens for the parameter $P_4$. 

27
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

Figure 5.4. learning curve for lasso. The test set has a maximum amplitude of 30%. The training set has a maximum amplitude of 15%. Lasso regression can fit the training data properly. However, the range of values taken by $P$ i.e the maximum amplitude in the test set has a major effect. We cannot provide accurate predictions on large variations with a lasso. For 40 samples the expected error of prediction on new data is not satisfying. Adding samples in the training set improves the error, although it is not enough.

5.1.3 Support Vector Regression

Support Vector Regression – SVR – allows us to use the kernel trick. We used different types of kernel. Two kernels showed interesting performance: linear and polynomial. The two kernel are quite close. Therefore results are almost the same for small variations, but a polynomial kernel is more effective when large variations are involved, as it can take non-linear effects into account.

This type of algorithm gives good results with a training set so that the maximum amplitude is 15%. On a test set with a maximum amplitude of 5%, we reach the criterion of 1% of error on average as desired. However, we cannot reach very good accuracy on larger variations, as it appears on Fig. 5.6.

The parameters for the algorithm are set by K-Fold cross-validation on a grid of parameters. This step is crucial in our case. A wrong set of parameters (regularisation, tolerance) could be a disaster. In some cases, the algorithm may not
5.1. ANALYSIS OF THE ACCURACY

Figure 5.5. learning curve for SVR. The test set has a maximum amplitude of 5%. It shows good results with 40 samples in the training set: for all parameters but $P_7$, the expected error on new data is smaller than 0.5%, and for 50 samples the same result applies for $P_7$. Increasing the number of elements significantly improves the quality of the prediction.

be able to simply fit the training data and give predictions. Yet, one can define a 'trust region', where the accuracy of prediction remains the same. Figure 5.5 and 5.6 show that we benefit a lot from improving the size of the training set.
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

Figure 5.6. learning curve for SVR. The test set has a maximum amplitude of 30%. The quality of the prediction is not the same for all parameters but remain in the same order of magnitude. As for all algorithms investigated so far, the difference of expected error between training and test set does not necessarily indicate overfitting. The maximum amplitude of the training set is 15%, the machine learning infer non linear effects on larger variations.

5.1.4 Ensemble-based algorithms

Algorithms based on ensembles have also been tested. Considering the size of the training set used to train the model, we expect algorithms based on decision trees to be poorly adapted to fit a function 'close to linear'. They would require a very large training set to fit the relation between $P$ and $Q$.

Several algorithms based on ensemble have been tested: extratree, random forest, gradient boosting. As all these algorithms are very similar, conducting the test was quick. The results are the same for the whole class of algorithms, therefore we give learning curves only for the case of random forest. Figures 5.8 and 5.8 show large error of prediction on the test set, and match our expectations regarding this class of algorithms.
5.1. ANALYSIS OF THE ACCURACY

Figure 5.7. learning curves for a random forest regressor, The test set has a maximum amplitude of 5%. While all other algorithms tested so far could reach at 1.5% of mean error or better, this algorithm cannot perform well with a low number of elements in the training set. One may notice that the error in the training set is larger than for other machine learning algorithms. This type of algorithms is simply not well-adapted to our problem.


**Figure 5.8.** learning curves for a random forest regressor. The test set has a maximum amplitude of 30%. As expected, this type of estimator is not adapted to our problem and performs poorly. The machine learner algorithm does not fit training data properly and therefore cannot issue accurate predictions. It would require a lot more samples in the training set to perform to its best.
5.1. ANALYSIS OF THE ACCURACY

5.1.5 Q(P) estimator

This model maps $P$ to $Q$ – whereas all other models here map $Q$ to $P$. The designed function presented here is a degree 2 polynomial, as it shows good compromise between accuracy and size of the training set. Using a degree 3 polynomial turns effective with more than 100 elements in the training set. Using higher degree polynomials would only lead to overfitting considering the number of samples we can afford. Lower degree polynomial cannot catch no linear effects and turns less effective than a degree 2 polynomial.

Since we use now $P$ as input, we have 7 'linear' features and 28 'quadratic' features. Adding the intercept means we try to map a space of dimension 36 to a space of higher dimension – $\approx 400$. We use 'multi task' lasso regularisation. This ensure that the same features are used to fit and predict all output, which gives more coherence to the model and improve the quality of prediction.

Two elements must be taken into account: the quality of the estimate provided for $Q$, but also the quality of the solution given when solving the problem ???. We show first a grid used to visualise the influence of the regularisation parameter – $\alpha$, then learning curves in the same fashion as for other models.
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

Figure 5.9. A grid used to visualise cross-validation error for a Q(P) learner. One axis represents the size of the training set, in the same fashion as what is done with a learning curve. The other axis represent the magnitude of the regularization parameter, called $\alpha$. We analyse here the expected error of the predictive model mapping $P$ to $Q$. The test set used here has a maximum amplitude is equal to 15%.

5.9 shows a grid where the size of the training set as well as the value of the regularization parameter change. Notice that we give the error of prediction for the estimate of $Q$ with a logarithmic scale. This grid gives major information. It shows that the error for such estimate is huge for a low number of elements in the training set, and that regularisation cannot improve this significantly. However with more than 40 elements in the training set, the error drops and we may expect better results when using this estimator as an oracle to solve the problem ??.

Learning curves presented on Figure 5.10 and 5.11 show strange behaviours. Two elements must be taken into account. The first one is the accuracy of the prediction of $Q$. As we have said, this accuracy is terrible if the training set contains less than 40 elements. Therefore we use an optimization routine on a poor approximation, ending with large errors of prediction. The second element is the fact that we use cross-validation and that parameters such as the regularisation parameter do change with the size of the training set. This explain most of the behaviour observed for the error in the training set. Considering the size of the training set, this algorithm cannot be used in practice.
5.1. ANALYSIS OF THE ACCURACY

Figure 5.10. Learning curves for a Q(P) estimator fitted with lasso regularisation. The test set has a maximum amplitude of 5%. We use a degree 2 polynomial to map \( P \) to \( Q \). Since we have 36 weights to optimize, this model does not show efficiency for training sets with a lower number of samples. The shape of the learning curve may seem unusual, it is due to the fact that we perform cross-validation for each training set, therefore the regularization term takes different values and the evolution of the prediction error is no longer smooth.
CHAPTER 5. ANALYSIS OF THE PERFORMANCE OF DIFFERENT ALGORITHMS

Figure 5.11. learning curve for a Q(P) estimator. The test set has a maximum amplitude of 30%. The quality of the prediction is not the same for all parameters, and does not show any stability. We cannot predict large variations well. The shape of the learning curve may seem unusual, it is due to the fact that we perform cross-validation for each training set, therefore the regularization term takes different values and the evolution of the prediction error is no longer smooth. As for other learner, it seems hard to infer non linear effects on large variations if the training set has ‘only’ a maximum amplitude of 15%. Moreover, the shape of the learning curve indicates a kind of instability in the behaviour of the predictive model: the test error is larger for 50 samples than for 40, but decreases for training sets of sizes 60. This cannot be used in practice.
5.2. ANALYSIS OF THE ROBUSTNESS

**Model selection**  This first step shows that RIDGE and Support Vector Regression with polynomial kernel provide the best estimators. Some important results and questions arise. 
Firstly, using 40 samples can provide good accuracy for small variations – test set with a maximum amplitude of 5%.
Secondly, adding a few samples to our training set can improve accuracy significantly. It also turns out that the accuracy of the prediction decreases when we want to predict large variations – test set with a maximum amplitude of 30%. We must find a way to ensure predictions as accurate as possible in every situation: we must investigate robustness.

5.2  **Analysis of the robustness**

We give here an estimation of the trust region for both ridge and SVR via the limit of convergence coefficient. It is given for different maximum amplitudes in the training set. For an initial error lower than this coefficient, we have a good rate of success regarding convergence.

It appears that using a set mixing evenly large and small variations – for instance with a maximum amplitude of 30% – is irrelevant. The diameter of the trust region falls to 0: no point is ‘safe’ regarding convergence.
Secondly, it appears that the value of the limit of convergence is rather the same for a large range of maximum amplitude, while SVR show an interesting optimum for a maximum amplitude of 15%. Using this optimum should ensure good robustness, and gives a good advantage to SVR.
Figure 5.12. limit of convergence coefficient for a ridge, computed for different values of the maximum amplitude in the training set – filled with 40 samples. This coefficient gives a trust region where the estimator can be used. Using amplitudes larger than 25% in the training set can turn the model to be useless in practice. We added some noise in the test data – 0.1%
5.2. ANALYSIS OF THE ROBUSTNESS

Figure 5.13. limite of convergence coefficient for a SVR, computed for different values of the maximum amplitude in the training set – filled with 40 samples. Using amplitudes larger than 25% in the training set can turn the model to be useless in practise. We added some noise in the test data – 0.1%. The optimal value for the maximum amplitude used in the training set is approximately 15%.
5.3 Analysis of the influence of the distribution of data in the training set

We analyse the influence of the maximum amplitude via the error amplification coefficient. As mentioned in 4.1, this coefficient compares the error of the prediction with the initial error. On Figure 5.14, a grid of values for the maximum amplitude in both training and test sets is prepared and the coefficient is computed for each case.

Regions corresponding to small values of the error amplification are region where the error is decreased drastically. For instance using a training set with small variations to predict small variations should lead to excellent results.

Regions where the error amplification coefficient is 'large'– for instance greater than 0.7 – are region where the model may not be able to fit the data properly. This can explain some observations made during about the limit of convergence. We want to avoid two situations.

Figure 5.14. We plot the 'error amplification' coefficient $\frac{\text{mean squared error of prediction}}{\text{mean squared initial error}}$. We used a grid of values for the maximum amplitude in the training set and the test set. All training sets have 50 samples. Large values – greater than 0.7 – of the coefficient correspond to situations we want to avoid.

First we can analyse the case where the training set is only made of small vari-
5.3. ANALYSIS OF THE INFLUENCE OF THE DISTRIBUTION OF DATA IN THE TRAINING SET

ations. If the test set contains only small variations, the error amplification is very low. Applying the correction suggested by the model should give accurate values for $P$. However, it would not be able to make accurate predictions for large errors. This explains why the limit of convergence is low for training set based on small variations.

On the other hand, a training set with a maximum amplitude of 30% leads to a predictive model which cannot predict small variations very well considering the size of our training set. The grid of values of $P$ described by the set is too coarse for small variations. We cannot use the fitted model to find the desired values for $P$ by applying the suggested corrections one after another. If the prediction is made for a large error, the error would be decreased. However, for a small error, the amplitude of the error would remain the same, or may even be increased. We would go back and forth between areas close to the reference and areas corresponding to large variations. We cannot ensure fast convergence. This explains the observation made in 5.2: no point could be declared as 'safe' regarding convergence.

**Type of distribution** : the maximum amplitude in the training set has a major influence on the performance of the predictive model, but we have also investigated the types of distributions of the data provided in the training set. We tried distributions that could favour either small or large variations. In the first case we used Gaussian based distribution. In the second case we forced variations to be on the 'corners' of the training set : we select a value for the maximum amplitude of the set, and each sample has its component is equal in absolute value to either zero or the maximum amplitude.

We tried to set a balance between small and large variations. We did not investigate the type of distribution further, as we just wanted to have an idea of the type of distribution that could help the algorithm to perform to its best. It is also a way for us to analyse and understand the performance of a predictive model.

From all observations, it seems that the best way to ensure convergence in practice is to use a training set with an iid uniform distribution and a maximum amplitude between 10% and 15%. This is still a very general result. Note that it only tells us what we can expect from available data, and it is not designed to be used as it is as a design of experiment.

**Noise** : the influence of the noise has not be discussed yet. In practice, it is true that measurement errors may happen. It could alter our model by preventing it to fit effectively to the provided data. Fortunately, we do not expect the noise to reach large amplitudes – it should not exceed 1% of $P$. For a machine learning algorithm such as the Ridge, a small proportion of noise would alter the results,
but this is not significant. Algorithms such as SVR are even more robust to noise: our implementation includes a tolerance parameter, which means that the model is able to ignore small variations – i.e. which could be considered as noise. Figure 5.15 shows that for a SVR we need a proportion of noise larger than 0.5% to observe a change in accuracy: this proportion of noise is considered as ‘large’ for this work.

![SVR (polynomial kernel)](image)

**Figure 5.15.** We add some noise to our data: between 0.1% and 1% of relative noise. The maximum amplitude is set to 15% in the training set and 30% in the test set. Considering the range of values of the noise, it does not alter accuracy significantly.

### 5.4 Time performance

As mentioned in the previous chapter, time is used as indication for the user. The only constraint is to make sure a user will not have to wait too long to get the results – by too long we mean a few seconds. All models respect this constraint. This is why time is not really a sharp criterion, but is used as indication for the user.

We give here the time requested to fit and predict a set of data. We denote several types of ‘conditions’, each one referring to a specific value of the maximum amplitude parameter, and a specific value of the noise.

- In condition 1, the maximum amplitude is set to 10% and the noise proportion to 0 in both training and test sets.
- In condition 2, the maximum amplitude is set to 30% and the noise proportion to 0.2% in both training and test sets
In condition 3, the maximum amplitude is set to 5% in the training set and 30% in the test set. The noise proportion is set to 0.5% in both sets.

Table 5.1. average time for training (s)

<table>
<thead>
<tr>
<th>Model</th>
<th>condition 1</th>
<th>condition 2</th>
<th>condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR (polynomial kernel)</td>
<td>1.1</td>
<td>0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>Q(P) learner (multi-task lasso)</td>
<td>0.3</td>
<td>0.4</td>
<td>0.07</td>
</tr>
<tr>
<td>Random Forest</td>
<td>7.5</td>
<td>9.2</td>
<td>11.7</td>
</tr>
<tr>
<td>RIDGE</td>
<td>0.005</td>
<td>0.004</td>
<td>0.006</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.4</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Since random Forest uses a lot of estimators it requires more time than other learners to perform a training step. The time required to perform training is more or less the same for all conditions: some differences appears, but they are not significant enough to make a real difference.

Table 5.2 provides results on prediction steps for the conditions described in section 4.4

Table 5.2. average time for prediction (ms)

<table>
<thead>
<tr>
<th>Model</th>
<th>condition 1</th>
<th>condition 2</th>
<th>condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR (polynomial kernel)</td>
<td>0.64</td>
<td>0.69</td>
<td>0.67</td>
</tr>
<tr>
<td>Q(P) learner (multi-task lasso)</td>
<td>864</td>
<td>875</td>
<td>990</td>
</tr>
<tr>
<td>Random Forest</td>
<td>50</td>
<td>62</td>
<td>77</td>
</tr>
<tr>
<td>RIDGE</td>
<td>0.309</td>
<td>0.318</td>
<td>0.312</td>
</tr>
<tr>
<td>Lasso</td>
<td>0.3223</td>
<td>0.3216</td>
<td>0.4125</td>
</tr>
</tbody>
</table>

The longest time to get a prediction comes from the model dealing with the direct problem: the so-called Q(P) learner. This model requires a call to minimization function. Solving the optimization problem takes more time than computing a function, but the requested time is still below our bounds. RIDGE regularisation in this case is not as good as lasso regularisation. In both cases the model is way slower than all the other models. It still remains in the bounds of the time constraint: ‘getting predictions within a few second’.

**5.5 Analysis of real-life data and preliminary results**

From simulation data, we could derive that two machine learning algorithms give good results regarding accuracy and robustness. The two algorithms must be tested...
on real-life situations.

We use data collected in real-life made on the actual machine. First we analyse the structure of the data, in particular its distribution. Then we split our data into a training set and a test set and compare predictions provided by our estimator to real-life values.

5.5.1 Analysis of the real-life data

We analyse the structure and distribution of data to determine if the recommendations made in section 5.3 can be applied on actual data. We discard data corresponding to extremely large variations of \( P \), that could be considered as outliers. Then we analyse the distribution of the data. Major differences with simulation arise.

Firstly, we used a \( iid \) uniform distribution in a compact of \( \mathbb{R}^n \) – the data is bounded – to generate \( P \) data during simulation. This means that the matrix containing all examples of \( P \) would be a full matrix. However, the matrix built after the actual data is sparse. Only one or two dimensions of \( \mathbb{R}^n \) are activated for each example: this is a major difference of structure.

Secondly, Figure 5.16 shows that the recommendations issued in 5.3 does not apply. Most of the provided samples correspond to small variations, and the distribution of data is unbalanced. We can state that our model will not provide the same performance as observed during the simulation phase. To do so, it would require more samples with variations of \( P_1, \ldots, P_7 \) close to \(-15\%\) or \(15\%\).
5.5. ANALYSIS OF REAL-LIFE DATA AND PRELIMINARY RESULTS

Figure 5.16. Distribution of variations $P$ in the actual data. All variations equal to 0 have been removed. The distribution is unbalanced and provides a lot of small variations. In order to match our recommendations, we should collect some extra samples for variations in $[10\%; 15\%]$ or $[-15\%, -10\%]$.

5.5.2 Preliminary tests on real-life data.

We show here a preliminary test performed on real-life data. We split data into a training set and a test set. The training set contains more than 100 samples. However, each sample in the actual data corresponds to a variation of only one or two parameters $P_i$. This means that for each parameter there is only between 15 and 40 samples with non-zero variations. This is somehow a source of trouble.

The estimator would learn to fit the value zero more than any other value. Figure 5.17 compares real and predicted values on new data. Our model cannot predict variations equal to zero properly. However it can predict non-zero variations. This first test shows that we still have some work, but gives us hope for future development.
Figure 5.17. We compare real and predicted value of $P_5$ on real-life data. Due to the structure of the data we have collected, the machine learning algorithm learns on a lot of small variations. The predictive model seems to have troubles predicting variations equal to zero. However, results are reasonably good when it comes to predicting non-zero variations.
Chapter 6

Future work and conclusion

From simulation data, we investigate the behaviour and performance of various algorithms on a very specific problem, where the dimension of the features vectors is significantly larger than the number of targets. Our main constraint is the size of the training set we can collect in reality. Fortunately, different algorithms – RIDGE, SVR with polynomial kernel – give satisfying results with small training sets. In order to ensure accuracy in various situations, we investigate the influence of the data collected to train our algorithm. Since we cannot afford a lot of training samples we want them to as useful as possible. As a result, we could issue some recommendation when collecting data in reality.

The machine learning algorithms giving best results for our problem are RIDGE and Support Vector Regression with polynomial kernel. They should be trained on sets with an iid uniform distribution and a maximum amplitudes of variations of $\pm 15\%$ to ensure a satisfying level of accuracy considering the number of samples we can afford (about 40).

We could collect dome real-life data. The number of samples we could afford is quite low : for each parameter we have less than 40 samples involving an actual variation of the considered parameter. The distribution of data does not follow our recommendation, and our predictive model presents some deficiencies during preliminary tests. It is not able to predict a variation of parameter equal zero. It can give good estimation of non-zero variations though. Our model must be improved, and this first result gives us hope.

6.1 Suggestions for future work on real-life data

The WP3 phase, which enables us to work with actual data has just been started. We could analyse the data structure and observe they do not exactly match all the recommendation we issued based on simulation results. This can explain why we do not have good results yet.
CHAPTER 6. FUTURE WORK AND CONCLUSION

However, working on simulation could give us some clue about the way to improve performance, and the different observations made so far can help us to build an effective strategy to fit the data properly. Some features will be added to our process in order to improve both accuracy and robustness of our estimator. We see different axis of development.

Collected data leave room to irrelevant measures, which makes a huge difference with simulation. A detection of outlier is about to be provided. This should ensure that all the examples used are useful for the estimator.

An other possibility is to tackle the problem of sparsity: we could build examples so that all parameters are activated. This should help the estimator to learn to give accurate predictions.

We are also investigating features selection on real-life data. We use methods such as Principal Component Analysis Smith, 2002 [6] as it may help us retrieving relevant features and improve accuracy or the estimator.

Another step is to make actual comparisons of performance between the current method and methods based on machine learning.

Eventually, we must say that we try to let some flexibility to our model. This is the purpose of a machine learning approach: our database is expected to change in the future. According to the available data, the estimator may evolve: some parameters may have to be adapted. Getting more data helps us training the model and catch non linear effects more efficiently. It may also happen that the machine behaviour changes a little, and we must be able to adapt the estimator. A solution to this may be to train the model with weighted data: recent data should be trusted more than old data. Such strategy will be explored in future development of the project.

6.2 Conclusion

In this master thesis we tackled a problem of multi-output regression based on a small training set. We first investigated the structure of the data and found out that a linear estimator could be used as a good approximation for the relation between input and output. We designed several types of estimators based on linear and polynomial models. We used classical machine learning algorithms based on scikit-learn library. We developed our own strategies and tools to evaluate accuracy and robustness on each estimator. We scored and compared all the estimators. RIDGE and SVR estimator based on polynomial kernel turned to give the best results: the first one due to its simplicity and the second one due to its robustness to noise.
6.2. CONCLUSION

We issued some recommendation about the optimal distribution of data. This can compensate the fact that we cannot afford a lot of samples in a short amount of time.

Working on simulation data has provided numerous hints and tips about the best way to handle actual data. We did preliminary tests regarding the structure of the data. This gave us some directions for future development and we can expect significant improvements regarding accuracy of the estimator. In the future, more data will be collected. This should improve the accuracy of the estimator.

Considering the size of the training set we can afford we can conclude that a RIDGE or a SVR with a polynomial kernel should provide good accuracy and robustness when used to fit a multi-output problem that is almost linear.
Appendix A

Tools and materials

A.1 Support Vector Regression

We describe the basic idea behind the $\varepsilon$-SV regression used during the master thesis. This section and the appendix are based on Smola and Schölkopf, 2003 [7], which is highly recommended to understand basic concepts of regression using support vectors.

Suppose we are given a set of data $\{(x_1, y_1), ..., (x_n, y_m)\}$ where the input belongs to $E = \mathbb{R}^n$ and $y$ belongs to $\mathbb{R}$. With $\varepsilon$-SV regression, we try to find a function $f(x)$ that has less than $\varepsilon$ deviation from the target $y$ for the whole training set. We also want this function to minimize the variance as much possible: $f$ must remain 'flat'. Therefore we ignore error smaller than $\varepsilon$ but strongly penalise larger errors.

For the sake of simplicity, we design $f$ as:

$$f(x) = \langle x, \theta \rangle + b,$$ where $x \in E, b \in \mathbb{R}$$

$\langle ., . \rangle$ is the standard dot-product in $E$. Forcing $f$ to remain flat can be done by solving a convex optimization problem:

$$\begin{cases}
\min_{\theta} \frac{1}{2} \|\theta\|^2 \\
|y_i - \langle \theta, x_i \rangle - b| < \varepsilon
\end{cases} \quad (P\text{-convex})$$

We make the assumption that the optimization problem is feasible. Under this assumption one can take the dual problem. We can build the Lagrange function:

$$L = \frac{1}{2} \|\theta\|^2 - \sum_{i=1}^{n} \alpha_i (\varepsilon - y_i + \langle \theta, x_i \rangle + b) - \sum_{i=1}^{n} \alpha_i^* (\varepsilon + y_i - \langle \theta, x_i \rangle - b)$$ \quad (A.1)

where $\alpha_i, \alpha_i^*$ are Lagrange multipliers. The dual variables must be positive. One can show that this function has a saddle point with respect to primal and dual variables. In particular it comes:

$$\begin{align*}
\partial_b L &= \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0 \\
\partial_\theta L &= \theta - \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i = 0
\end{align*} \quad (A.2)$$

51
APPENDIX A. TOOLS AND MATERIALS

We can write the dual problem

\[
\begin{align*}
\text{maximize} \quad & \left\{ \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) < x_i, x_j > \\
& - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) \right\} \\
\text{with} \quad & \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0
\end{align*}
\]

(P-dual)

(A.3)

It comes

\[
\theta = \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*) x_i
\]

(A.4)

The function depends entirely on the training data: its complexity is not a consequence of the dimensionality, it is a direct consequence of the number of input.

SVR can also use a method called the 'kernel trick'. It aims at replacing the dot product by a so-called kernel function. This is equivalent to remapping the space of input to a feature space where we can solve a problem similar to the convex optimization problem P-convex. For a more comprehensive of SVR method, the interested reader may refer to [7], [1].

The implementation used during the master thesis also includes regularisation. Therefore the algorithm has two main parameters: regularisation and tolerance.

A.2 Comparison of training and test error

As mentioned in section 5.1, a small error in the training set and a large error in the test set does not always mean overfitting. We give here a short example to illustrate this. It has to be compared to the situation exposed in section 5.1, where the so-called maximum amplitude is not the same in training set and test sets.

We consider the same true function as in section 3.1. On Figure A.1 we consider a training set restricted to \(0.3 < x < 0.7\). We use a degree one polynomial to fit this training set. As shown in section 3.1, this function cannot fit non-linear effects and leads to a situation of underfitting. Here, the function fits the training set: the error between real and predicted value is mainly due to noise. However, if we compare predicted and real value on the interval \([0, 1]\), the expected value on new data – i.e on a test set – would be quite large. This is usually classified a situation of overfitting. The same thing happens in section 5.1.
A.2. COMPARISON OF TRAINING AND TEST ERROR

Figure A.1. Top: we collect data in a training set. A predictive model is built, using a degree one polynomial. This type of function cannot fit all variations on the interval [0;1]. Bottom: we plot the corresponding learning curve, where the model is trained on a subset of the training set. The learning curve would indicate a problem of overfitting – low error in the training set, large error on the test set. However, the difference of error between training and test sets is mainly due to the fact that the training set covers only a part of the problem, and the test set covers a wider range of values. The same problem arises during the master thesis: our training set covers a certain range of values defined by a maximum amplitude, but the test set may have a larger maximum amplitude and therefore may cover a wider range of values.
Bibliography


