Mixed-Integer Linear Programming Robust Regression with Feature Selection

by

Oleksii Omelchenko

B.Sc., Taras Shevchenko National University of Kyiv, Ukraine, 2010

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

APPROVAL

Name: Oleksii Omelchenko

Degree: Master of Science

Title: Mixed-Integer Linear Programming Robust Regression with Feature Selection

Examining Committee: **Chair**: Dr. Qianping Gu
Professor

Dr. Andrei Bulatov
Senior Supervisor, Professor

Dr. Cenk Sahinalp
Supervisor, Professor

Dr. Artem Cherkasov
Supervisor, Computing Science, Simon Fraser University, Adjunct Professor,
Department of Urologic Sciences, UBC, Associate Professor

Dr. Martin Ester
Internal Examiner, Professor

Date Approved: December 16, 2014

Abstract

We introduce a Mixed-Integer Linear Programming approach for building Regression models. These models can detect potential outliers and have a built-in Feature Selection technique. We demonstrate how to build a linear regression model as well as a multidimensional piece-wise linear regression model that can simulate non-linear models. We compare our techniques with the existing statistical approaches for building regression models with different feature selection algorithms by comparing the results of predictions for 3 real-world data sets. All experiments show that our approach is useful in case where the number of training instances is less than the number of predictors, more stable and provides better results than Stepwise regression, which is the most used linear regression technique in cases when we deal with too many features in the model while having fewer observations.
Keywords: linear regression, robust regression, mixed-integer linear programming, feature selection, QSAR.

To whomever whoever reads this!

``The Guide is definitive. Reality is frequently inaccurate.''

--- The Hitchhiker's Guide to the Galaxy, Douglas Adams*, 1978-80*

Acknowledgements

I wanted to say thank you to so many people. Unfortunately, the space on this page is limited so I can mention only a few people. However, I'm grateful to everyone who has influenced my life and thanks to whom I can do what I do.

First of all, I would like to thank Dr. Andrei Bulatov. This work is definitely not from his field of expertise but he gave me a chance to try myself in the field of drug design and computer-aided peptides design. Moreover, he was supporting me during my Master program progress and introduced to the world of Random graphs that I find somehow magical.

Second, I wanted to say thank you to Dr. Cenk Sahinalp for his introduction to bioinfomatics. It was definitely not the easiest class but I enjoyed it. And that was he who acquainted me with Dr. Artem Cherkasov at the Vancouver Prostate Centre.

Third, thank you Dr. Artem Cherkasov for letting me working a little bit on your QuaSAR project and introducing me to the world of computational drug design.

Next I wanted to mention and say thank you to my parents, brother and grandma. As people usually write, without support from their families their work would never be finished. And I completely agree with this statement. So thank you for your support and faith in me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Regression analysis is the most popular tool in predictive analysis and statistics [15]. Informally speaking, regression analysis tries to identify a functional relation between a dependent variable and one or more independent variables based on a limited number of available examples by learning this relationship from the provided training data set. Each regression analysis starts with hypothesizing the form of the relation. The regression model is simply a mathematical representation of the relation that consists of functional interactions between independent variables and the inner model's numerical parameters. In the most general sense, a model can be either linear or non-linear.

To construct a regression model means to estimate the values of the parameters of the model in order to produce a regression relation that would nicely "fit" the data we have. Many statistical tools have been developed to determine if the model is reliable and satisfactory [29]. If the model seems acceptable, the regression function can be used for predicting dependent variables given the values of the independent variables.

However, if the model doesn't seem to be predicting well even on the known instances from the training data set, there might be several reasons. First, the hypothesized form of the regression may be inadequate, second, the training data set may contain noisy observations that need to be excluded prior to regression analysis, and third, we may need to include additional independent variables that can enhance the model and repeat the modelling process [17]. Unfortunately, selecting of a new form of regression model, excluding of observations, which we call outliers, and altering the list of independent variables are rather intuitive steps than formal rules. The main reason behind this is that we try to operate with objects of *a priori* unknown nature, meaning we do not know in advance if there is indeed a good functional relationship between dependent and independent variables, what observations are representative for the process we study and what observations were received due to mistakes in measurements.

The problem of outlier detection is an old topic in statistics and it has been further advanced by Machine Learning studies. Outliers are the observations that somehow statistically differ from the majority of the training instances. Unfortunately, there is no rigorous mathematical definition of what constitutes an outlier; so labeling some observation as an outlier is a subjective decision rather than a formal rule. Many techniques have been developed to tackle the outlier detection problem [23]. In the most general sense, they all can be divided into 2 categories: purely statistical and machine learning related.

The statistical approaches were the first to be developed. Among them are the following statistical tests: Z score, Grubbs Test [18], Tietjen-Moore Test [50], Generalized Extreme Studentized Deviate (ESD) Test [6], and many more [22]. Usually, they make an assumption about the probability distribution of the underlying process first, and, based on this assumption, conclude about the probability for some observation to appear in a sample taken from the suggested distribution. The obvious disadvantage of such approaches is it is not always possible to suggest a correct probability distribution. Moreover, some statistical tests such as Tietjen-Moore Test and ESD accept as a parameter an exact number of outliers we expect to see in the data. Unfortunately, since there are no formal definitions of outliers it is hard to conclude how many outliers we expect to see.

On the other hand, Machine Learning based approaches do not make any prior assumptions about the data distribution. They rely on the distances between observations and if some observation is too far from the others, we may conclude that this is an abnormal observation [23]. The weak sides of these approaches are: 1) we can choose different distance metrics and there must be a clear justification why we decided to pick a specific one; 2) machine learning algorithms require much more computational efforts than regular statistical tests. Overall, there is no formal guide which technique to use to filter out the abnormal training instances.

Another aforementioned common problem that arises in regression analysis is choosing relevant independent variables [9]. In general, the more independent variables we add to the model, the more flexible it becomes, meaning the more complex interactions between variables it can account for. However, it might happen that we have included too many variables, making it possible to memorize all the observations instead of learning the true relation that lies behind the dependent and independent variables which is the goal of any regression analysis.

Some heuristics and statistical criteria have been suggested for tackling the problem of selecting relevant independent variables [19]. Unfortunately, first, no formal rules exist which would dictate how many variables we need to include to the model to make its predictive performance acceptable for our purposes and, second, results from these heuristics may contradict each other.

We faced all the above issues while trying to construct reliable and well behaving predictive models using the traditional regression techniques for a number of data sets provided by the Vancouver Prostate Centre. These data sets were collected as a first step of a bigger project aimed at discovery of immunomodulatory peptides, which can control or suppress inflammatory processes in living bodies. At the Vancouver Prostate Centre they believe that Quantitative-Structure Activity Relationship or QSAR studies can be used for predicting immunomodulatory properties of the peptides and by obtaining good predictive models we can prove this.

Quantitative Activity-Structure Relationship or QSAR models are predictive statistical or machine learning models used in biological and biochemical studies. The ultimate goal of modelling is to find a mathematical relationship between structural or physico-chemical properties of chemical compounds and their specific biochemical activities that describe beneficial or adverse effect of the compounds on living bodies.

Since living bodies are the complex biochemical systems, any imbalance in concentrations of the natural chemical substances may lead to a disease state. As a result, each disease can be associated with one or more compounds that we seek to suppress (inhibit) or to boost (catalyze) their production. So biochemical activity of a drug is expressed as a concentration of these disease-associated organic chemicals after applying a potential drug.

To be precise, QSAR modelling pursues two goals [26]. First, it aims to find a subset of numerical physico-chemical properties (in QSAR they are called descriptors or features of compounds) that would definitely or very likely detect the presence or absence of a specific biochemical activity. Second, it looks for such a mathematical model based on the selected descriptors that can approximately predict biochemical activity, given the computed descriptors of previously unknown chemical compound that we treat as a potential drug. Having a reliable predictive model, we can create a list of chemicals, compute their physico-chemical properties and assay their biochemical activity *in silico*. After that we can disregard compounds that were predicted not active enough, focusing only on the most active ones. This way, we save time and money needed for conducting experiments, since we do not synthesize and assay the non-active compounds.

Another positive side-effect of QSAR models is a set of relevant descriptors. They might give researchers and chemists a key for understanding why specific compounds have the activity we are interested in; thus, making their following analysis easier [47]. This is the reason why in QSAR a) it is highly recommended to use predictive techniques that are easy to interpret, and b) to avoid the reduction of the number of relevant features by their transformations, like feature clustering, PCA/SVD, LDA, etc., at any cost, since they may lead to difficulties with interpreting of the final model.

To construct a data set for QSAR modelling, chemists first build a small library of chemical compounds that are known to have the activity we are interested in. They can found these compounds either from literature or by conducting a small experiment to assay the activity of some chosen chemicals on the samples of the donor cells. At the end of this step we have a list of compounds and their measured activities.

In the second stage, we compute *in silico* all available physico-chemical properties of compounds in the library from the first stage. This is done by using a number of QSAR computer programs, like Molecular Operating Environment (MOE), Dragon, Tomcat, etc., that include algorithms to compute specific descriptors [26]. These descriptor range from as simple as overall molecular weight of the compound, the number of specific atoms or functional groups, that can be computed even on paper, to as complex as numerical quantum characteristic of chemicals, that would require several hours to compute even on the most modern personal computers for only one chemical compound [31]. However, we need to compute as many different physico-chemical properties as possible since we do not know in advance what properties must be present in the model to make accurate predictions.

Finally, compounds, their activity, and computed descriptors are aggregated into one data set that is used for modelling. In the data set the descriptors serve as independent variables and the activity is a dependent variable. However, since we measure biological activities, it is very common for QSAR data sets to have many outliers and noisy observations. So outlier detection becomes a crucial problem in any QSAR modelling. Another common problem is a huge number of features available for modelling, making it impossible to use many predictive tools without applying Feature Selection first [26].

We faced all the above mentioned problems while trying to build reliable predictive regression models for the provided data sets. First, we applied Stepwise regression, that is commonly used in QSAR studies [2], but it did not perform sufficiently well. Although, we could try some complex non-linear methods but we wanted to keep the model simple to interpret and easy to build without applying any transformations to the original descriptors except of scaling.

In this thesis we suggest and describe how to use the Mixed-Integer Linear Programming (MILP) approach for building regression models that can automatically detect and remove potential outliers from the model, bound and reduce the number of features used in the model. In this work we describe how to build MILP linear regression as well as MILP piece-wise linear regression that can be used for modelling even non-linear relationships, where regular linear regression fails to build a good predictive model. Besides being robust and having a built-in Feature Selection, MILP regression can suggest when an analyst should switch to another predictive technique without having to wait until the actual modelling is finished. This is possible, since we can calculate a lower bound of predictive performance of the **potentially best** regression model formulated as a Linear Program. Therefore, if we are not satisfied with its performance, we can switch to the MILP piece-wise linear regression. If again we cannot find acceptable model using the latter approach we need either to include additional independent variables and repeat the modelling process or to conclude that the linear approaches are not applicable in that case.

We tested our approaches against the traditional Stepwise regression, that uses the stepwise Feature Selection algorithm as an intermediate step in the process constructing a model. For comparing our approaches with Stepwise regression we used 3 real-world data sets and showed better performance of our methods relative to the traditional regression techniques. Our method was also able to build better models with fewer features while yielding much better predictive performance. Moreover, since the models we obtain are linear, they are easy to interpret and to use for future predictions.

Chapter 2

Theoretical Background

2.1 Regression Analysis

In regression analysis we try to find a relation between a *dependent variable* y (which we will call the *response* or the *response variable* interchangeably in this work) and a vector of *independent variables* $x = (x_1, x_2, ..., x_n)$ (we will also call them *features*, *attributes*, *regressors*, and *predictors*) by studying a limited number of observations of the response variable's values coupled with the corresponding values of the independent variables.

More formal definition of regression analysis [38] is the following:

Definition 1. *We are given a class of regression functions $\Lambda = \{\bar{f}_l(x, \beta)\}$ as well as a set of observations $D = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is a row-vector of corresponding values of independent variables and $y_i \in \mathbb{R}$ is an associated value of a dependent variable, n is the total number of training instances in the set, and d is the number of available independent variables.*

We would like to find a function $\bar{f}(x, \beta) \in \Lambda$, such that the response variable y can be approximated as a function of independent variables:

$$y \approx \bar{f}(x, \beta) + \epsilon \,, \tag{2.1}$$

where $\epsilon \in \mathbb{N}(0, \sigma^2)$ is random normally distributed noise and $\bar{f}(x, \beta) \in \Lambda$ is a function from Λ in x with parameters β.

Every model consists of 3 components:

1. a response variable y;

2. a function, denoted $\bar{f}(x, \beta)$ that defines a deterministic component of the model;

3. a non-deterministic noise term, usually denoted by ϵ.

The noise term that is included into the model make the relationship between dependent and independent variables statistical rather than fully deterministic. This term accounts for any unpredictable deviations in the response variable which we cannot predict and which we would like to minimize.

The response variable y is a quantity we expect to exploit and explain using the suggested function $\bar{f}(x, \beta)$. We assume that the dependent variable can be successfully explained as a function of one or more independent variables. However, this assumption is not always true, and one of the goals of modelling is to confirm or refute this assumption.

The function $\bar{f}(x, \beta)$ contains arguments of two types. These types are the predictor variables $x = (x_0, x_1, ..., x_d)$ and the parameters of the model $\beta = (\beta_0, \beta_1, ..., \beta_l)$. The predictor variables are measured together with the values of the response variable and they can be treated as constants, while the parameters β are the quantities that will be estimated during the construction of the model. The parameters and predictors are combined in different forms in the function $\bar{f}(x, \beta)$ to produce a function that we expect to explain the behaviour of the response variable.

The parameters β are estimated using a lack-of-fit function $L(\bar{f}(x, \beta), D)$, where $\bar{f}(x, \beta)$ is the functional part of the regression model and D is a training data set. Most commonly used lack-of-fit functions are (recall that the training data set is $D = \{(x_i, y_i)\}_{i=1}^{n}$):

- **Least Sum of Squares**. This method tries to minimize prediction errors in terms of the L_2-norm or more formally:

$$L(\bar{f}(x, \beta), D) = \sum_{i=1}^{n} (y_i - \bar{f}(x_i, \beta))^2. \tag{2.2}$$

  If this criterion is used, then linear regression is called Multivariate Least Squares or MLS regression.

- **Least Sum of Absolute Deviations**. This method minimizes errors in terms of the L_1-norm:

$$L(\bar{f}(x, \beta), D) = \sum_{i=1}^{n} |y_i - \bar{f}(x_i, \beta)|. \tag{2.3}$$

Although, these are the two most often used and best studied lack-of-fit functions, many other lack-of-fit functions exist that can be used under different circumstances or when some assumptions can not be verified [38].

The lack-of-fit function estimates the prediction error produced by a given model. Since we would like to build a model that minimizes this error, we tune parameters β so that the chosen lack-of-fit function reaches its local or global minimum. In other words:

$$\bar{\beta} = argmin_{\beta} \; L(\bar{f}(x, \beta), D) \; . \tag{2.4}$$

In this formula $\bar{\beta}$ are the estimated values of the parameters β that produce a regression function $\bar{f}(x, \bar{\beta})$ providing a minimum for the lack-of-fit function $L(\bar{f}(x, \beta), D)$ computed on a training data set D. If the lack-of-fit function coupled with the regression function is a convex function of variables β, then we expect to reach its global minimum, meaning we can find the best possible values of the parameters of the model. However, if it is not, then we would like to find a local minimum as deep as possible.

Depending on the used class Λ of allowed approximating functions, regression analysis can be either linear or non-linear.

2.1.1 Linear regression

The simplest predictive model for regression is the one that involves only linear combinations of fixed functions of independent variables:

$$\bar{f}(x, \beta) = \sum_{i>0} \beta_i \phi_i(x) + \beta_0 \; , \tag{2.5}$$

where $\beta \in \mathbb{R}^{p+1}$ are the model's parameters and $\phi_i(x)$ are called *basis functions*. Although, the function (2.5) can be non-linear in the input variables x, any regression model of this form is called linear regression.

Many basis functions have been used to account for different non-linear interactions between independent variables. We will present only a few of the most commonly used basis function:

- Simple basis function

  $$\phi_i(x) = x_j \; ,$$

  where x_j is the j component of a row-vector x.

- Polynomial function

  $$\phi_i(x) = x_j^{a_j}$$

for some constant a^j. The simple basis function is a special case of polynomial basis functions, where all $a_j = 1$.

- Gaussian function
$$\phi_i(x) = \text{exp}\left(-\frac{(x_j - \mu_j)^2}{2s_j^2}\right) ,$$
where μ_j is the mean of variable x_j and s_j is its standard deviation.

- Sigmoidal function
$$\phi_i(x) = \sigma\left(\frac{x_j - \mu_j}{s_j}\right), \quad \text{where} \quad \sigma(y) = \frac{1}{1 + \text{exp}(-y)}$$

More examples of basis functions and suggestions when to use them can be found in [11].

Every linear regression analysis starts with accepting a number of assumptions about the process being studied:

- **Linearity**. We assume that a linear relation exists between the dependent and independent variables or we can account for substantial variability of the dependent variable by a linear combination of independent variables.

  A special quantity which is called the Coefficient of Determination R^2 is often used to calculate the fraction of the variability of the response variable predicted by linear regression:
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} , \tag{2.6}$$
  where $SS_{res} = \sum_i (y_i - \bar{f}(x_i, \beta))^2$ is the total sum of squared residuals and $SS_{tot} = \sum_i (y_i - \bar{y}))^2$ ($\bar{y}$ is the sample mean of the dependent variable) is the value called the total sum of squares, which is proportional to the total variance of the sample.

- **Normal noise**. We assume that the noise term is a normally distributed random variable with zero mean and some unknown but **constant** standard deviation.

- **Independence of observations**. Another assumption we make prior to performing regression analysis is that all measurements are independent of each other, and one observation does not influence another one. Thus, we expect a constant variance of residuals after analysis.

Linear regression possesses a number of useful advantages over other predictive methods, like interpolation with polynomials, non-linear regression, or artificial neural networks (ANN):

- **Empirical Model**. We can use linear model even in the case, when we do not know the true relation between a response variable and independent variables. Furthermore, if this relation can be approximated to some extent by a linear function, linear regression analysis will be successful.
- **Easy to model**. A linear model is easy to compute and that is the main reason why it is so wide-spread.
- **Easy to interpret**. Linear regression is simply a linear combination of independent variables and model's parameters. Each coefficient in the linear regression function represents contribution of the corresponding variable to the response variable. Moreover, for linear regression many statistical criteria exist that can assess the quality of the model.

At the same time, there is a serious limitation of linear regression that does not allow to use in many Pattern Recognition problems or in the applications where we deal with many independent variables with smaller number of available observations. The reason is that in any n-dimensional affine space containing less than n points, we can write infinitely many equations of hyperplanes that contain all the points. Thus, linear regression does not yield any useful knowledge, and so we need to switch to another predictive tool or to use Feature Selection to reduce the number of independent variables.

2.1.2 Non-linear regression and piece-wise linear regression

Linear regression is a powerful predictive tool that is easy to model and interpret, but it does not always perform well, especially when no clear linear relation exists between variables. In this case, non-linear regression can be used instead [38].

The definition of non-linear regression is given by Definition 1 with additional requirement that a class of approximating functions Λ must contain functions non-linear with respect to model's parameters β.

Since Λ can contain functions of any nature, non-linear regression is thought to be more flexible and powerful in predicting the response variable than regular linear regression. However, prior to applying non-linear regression, a researcher must clearly justify his selection of a class of approximating functions. Otherwise we can always deduce a function that would fit all the available training data points, and conclude that this functional dependence is the true underlying relation, while it is not. Such problem is especially common in the Artificial Neural Networks studies because ANN functions act as the universal approximators and there are no formal rules for deciding the number of model's parameters.

Another problem associated with non-linear regression is the inability to obtain a model with best predictive characteristics. We can imagine integrated predictive errors of a model as a function of model's parameters β. In linear regression, this function is concave, meaning the function has only one local minimum, which is also a global minimum. When we tune the parameters so that we reach this minimum, we can conclude that no other linear model would yield more accurate prediction. However, the majority of functions used in non-linear regression are neither concave nor convex, meaning no algorithm exists that would reach their global minimum in reasonable time; therefore, we may be unable to find the best model among all the possible functions in Λ.

Non-linear regression includes a term for random noise just like linear regression. However, in non-linear regression function noise components can mix with each other, leading to different than normal probability distributions. As a consequence, we can no longer use confidence intervals and other statistical estimations as we do in linear regression.

On the other hand, we can overcome all these limitations of non-linear regression by linearizing our model using piece-wise linear regression or segmented regression [27]. Most often piece-wise regression is used for modelling simple predictive problems, when we have only one independent variable that affects the response variable. However, the basic idea of segmented regression can be easily extended to any number of independent variables.

The basic idea of segmented regression is to partition the domain of independent variables into convex subdomains and then to approximate within each subdomain using linear regression; thus, allowing the relationship between the response variable and independent variables to vary from subdomain to subdomain.

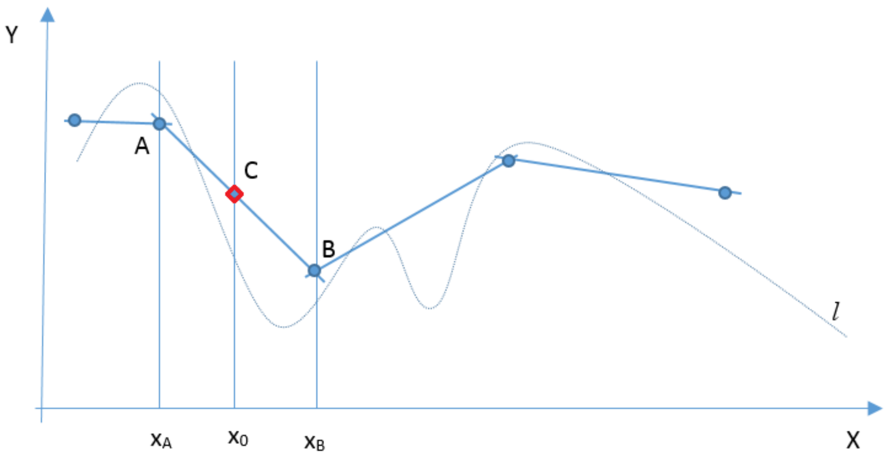


Figure 2.1: An example of segmented regression in the two dimensional space. The dotted curve l represents the "true" relation between the response Y and independent variable X. To approximate it, we build linear simplexes (bold straight lines) between each pair of consequent training data points depicted as circles. Note that due to errors made while measuring the observations, the training data points slightly deviate from the "true" relation curve

Figure 2.1 shows an example of how segmented regression is performed. We are given 5 training points depicted as circles that we will consider as the basis points for segmentation. These 5 points are measured with random normally distributed noise and so their values differ from the values of a "true" generating function that we need to approximate.

To approximate a value at a specific point of the independent variable we first find two closest basis points and then draw a straight line through them. This line will serve as an approximation of the "true" generating function l for all the points that lie in between the basis points.

In general, when we deal with points in $(n+1)$-dimensional spaces, where we have n independent variables and one dependent variable, we need exactly $(n+1)$ closest basis points to build an approximating convex simplex, which is a line in 2-dimensional case, triangular in 3-dimensional case, prism in 4-dimensional space and so on.

Segmented regression uses linear simplexes, and as a result, it has all the advantages of linear regression, as well as the ability to model non-linear relations. However, it also has a drawback that makes this approach inappropriate in some circumstances. Unlike regular linear regression, segmented regression does not yield a closed-shell functional representation of the relation between the response and independent variables. Thus, segmented regression can be used solely for approximations of a "true" generating function at specific points.

2.2 Outlier Detection

In statistics, *outliers* are observations that statistically differ from the majority of observations [18]. Outliers may appear for many reasons, but the most common ones are experimental errors and *a priori* unknown variability in the measured properties, which we are trying to explain and predict. Usually outliers are excluded before applying statistical analysis; however, abnormal observations should be investigated separately from good observations since some of them may carry additional knowledge useful for our analysis and provide us with a key for deeper understanding of the studied phenomena [12].

Many techniques exist that allow us to find potential outliers [23]. In general, they can be divided into two categories: statistical and machine learning techniques.

Statistical approaches were the first methods of outlier detection. The basic idea behind the statistical methods is that all the observations are sampled from some, possibly unknown, distribution and so, outliers are the values that are very unlikely to occur from that distribution. One of the earliest approaches was Grubb's method (Extreme Studentized Deviate) [18], which calculates a so called Z value of each observation. Z value is the difference between the mean of the response variable and the query value is divided by the standard deviation of the response variable, where the mean and standard deviation are calculated from all the observations including the query value. After computing the Z value is compared with 1% or 5% significance level. This method do not use any input parameters. However, it is sensitive to the number of observations and it assumes that the response variable is normally distributed, which is not always true.

Another simple and well-known statistical tool used for potential outlier detection is a method called boxplot [30] that uses graphical plots for revealing the outliers in a given sample. The boxplot is a diagram that visually represents the distribution of the sample we study, highlighting where the majority of values lie as well as shows abnormal observations that lie too far from the majority of points. To construct a boxplot we need to calculate the median M, first quartile $Q1$, the third quartile $Q3$ and the interquartile range $IQR = Q3 - Q1$.

Next, we draw a box of length IQR and we indicate the median M as a vertical line inside the box.

The next step is to define a range of values relative to the lower and higher quartiles that we will consider as a range of normal observations. Laurikkala et al. suggested a heuristic of $1.5IQR$ as the most common value for the range, since it is very unlikely to get an observation that lies lower than $Q1 - 1.5 \times IQR$ or higher than $Q3 + 1.5 \times IQR$. If some observation falls out of the above range, this means the observation seems abnormal and thus, we suspect it is an outlier.

In Figure 2.2 we can see an example of a boxplot diagram. The box represents the majority of observations that lie within an interquartile IQR range. The left side of the box is the value of the first quartile, while the right side of the box is the third quartile. The vertical black line inside the box is the median value of the observations and two horizontal lines stretching from the left and right sides of the box depict the range of "well-behaving" observations while 5 black points outside the range are potential outliers.

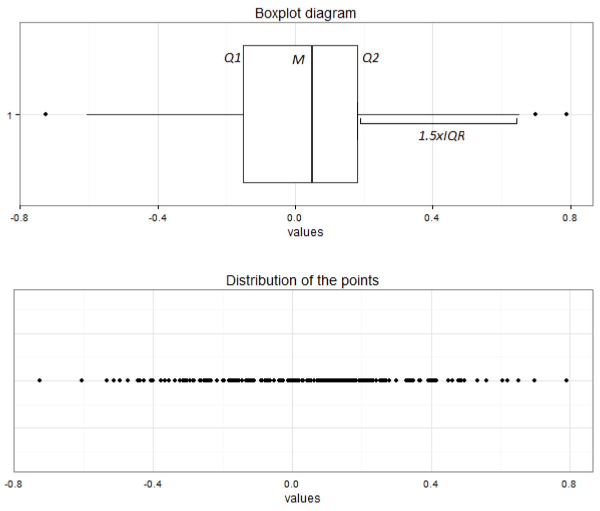


Figure 2.2: Example of the boxplot method

Boxplots make no assumption about the distribution of the data we study, however, the discovery of outliers is very subjective in this case, since the analyst decides what the range of "well-behaving" observations should be.

As for the outlier detection techniques that are based on machine learning algorithms, they rely on the idea that good observations are grouped into some clusters, while abnormal observations lie far from the clusters. These methods rely on proximity information from the data, like different clustering techniques and especially methods that are based on k-Nearest Neighbours. Proximity based techniques make no prior assumptions about the data distribution and usually are very easy to implement. However, there are two drawbacks in such approaches. First, there must be a clear justification which distance metric we should use to calculate the proximity between any two observations. And second, machine learning algorithms are very susceptible to the *Curse of Dimensionality*. While the most effective statistical techniques automatically focus on the suspicious observations, the machine learning methods need to compute the distance between any 2 points and so increase in the data dimensionality will lead to at least quadratic increase in time complexity. This is a common issue for the methods based on clustering or k-Nearest Neighbour algorithms. To overcome these problems, the data need to be preprocessed first to select the most relevant features and/or to reduce the number of observations leaving only the most representative [3, 43, 44].

All these techniques are general-purpose and can be applied before any data analysis routines. However, if we need to perform regression specifically, we have another option. We can use regression techniques that are less sensitive to the presence of outliers. These regression techniques are called robust. The most widely used Least Sum of Squares estimator is a well-known non-robust lack-of-fit function. Even the presence of only one abnormal observation may change the behaviour of the MLS regression [21].

On the other hand, another popular estimator, Least Sum of Absolute Deviations, is a known robust estimator since even the presence of several outliers does not change much its results comparing to results obtained on a data set without outliers. Many more robust estimators have been developed and are still being developed, like the general M-estimator [32], Least Trimmed Squares [41], S-estimator [40], MM-estimator [55] and their variations [56] to account different assumptions, conditions and models.

Nevertheless, despite of having many tools for detecting potential outliers, they still pose a problem for regression modelling. The reason is that no formal definition of an outlier exists, and so different techniques most likely will label different observations as outliers.

2.3 Feature Selection

Feature selection reduces the dimensionality of data by selecting only a subset of measured features to create a model. Subset of features is evaluated using criteria that usually involve minimization of a specific measure of predictive error. Feature Selection algorithms search for a subset of predictors that optimally model the response variable, subject to constraints such as required or excluded features and the size of the subset.

Feature selection is preferable to feature transformation when the original units and meaning of features are important and the modeling goal is to identify an influential subset. When categorical features are present, and numerical transformations are inappropriate, feature selection becomes the primary means of dimensionality reduction.

It might seem that adding more variables to the predictive model can make the model utilizing more information; thus, making it more reliable. However, the more variables we include into the model, the more flexible it becomes, and so we increase the overall probability for the model to overfit [9]. So the final goal of Feature Selection is to keep the number of parameters used in the model as low as possible while keeping its predictive performance on an acceptable level. This way we reduce the probability of overfitting the model.

There is a number of other potential benefits from Feature Selection: simplification of data visualization, decrease in storage requirements, reduction of training time, defying the curse of dimensionality to improve the overall prediction performance.

Unfortunately, Feature Selection is an NP-hard problem, meaning no effective algorithm exists to solve it, unless $P = NP$. Moreover, even a formal definition of the feature's relevancy does not exist. We understand feature's relevancy as its ability to increase predictive performance of the chosen model. However, as it was mentioned earlier, the more features we add to the model, the more degrees of freedom it gets, leading to two conflicting consequences. First, the model is capable of memorizing more and more underlying features of the objects we investigate, and second, it can reach the point when it memorizes all the data points instead of extracting useful knowledge from them.

Currently, Feature Selection is solved by a number of different approaches [19] that can be divided into the following three categories:

- **Ranking the features**. A number of ranking statistical criteria has been developed that measure the relative usefulness of every feature for making predictions. After ranking, we can select the best k features and build a model using only the selected attributes.

  Formally speaking, we have a data set of n observations $(x_i, y_i)_{i=1}^n$ each consisting of d independent variables $x_i \in \mathbb{R}^d$ and one output dependent variable y_i. Let x_{ik} be the *k*th component of the vector x_i. We would like to select at most $\lambda \leq d$ highly relevant features.

  We use a scoring function $S(k) \in \mathbb{R}$ for $k = 1..d$ computed from the *k*th feature x_{ik} and y_i for every $i = 1..n$. We will assume that a high score indicates high relevancy of the variable in the sense of the used scoring function. So after calculating $S(k)$ for every $k = 1..d$ we sort all features in the decreasing order of $S(k)$ and select the top λ independent variables.

  There are other techniques for selecting an optimal subset of features [24, 20]. The main idea of these techniques is to introduce a random probe in the data by adding artificial independent variables to the original data set that consist purely of gaussian noise. After sorting the features, we disregard independent variables that have lower relevancy than the newly constructed random variables since they potentially would provide less information for a predictive model than regular random noise.

  The most well-known and widely used scoring functions are Bayesian Information Criterion (BIC), Akaike information criterion (AIC), Correlation (R) and Coefficient of Determination (R^2) [9]. However, the major drawback of ranking the features is that each criterion can suggest a different list of the best attributes.

- **Wrappers**. Methods of this type use a predictive machine learning algorithm as a perfect black box to evaluate the quality of a subset of attributes. To assess relevancy of a subset of the features, we need to select a predictive tool first, like a regression model or a classification algorithm, then we compute predictive performance of the selected model that uses only the subset of selected features. Next, if we are not satisfied with the quality of prediction, we modify the subset by following a predefined search strategy. We repeat this procedure until we get acceptable predictive performance or until we reach some stop criterion, like a maximum number of iterations or inability to modify the subset of features in two consecutive iterations.

A wide range of search strategies can be used for selecting the best subset of features, among them are: sequential search, simulated annealing, genetic algorithms, tabu search, etc. Actually, any general purpose heuristic can be used as a search strategy.

Using a learning machine algorithm as a black box, wrappers are universal and simple to implement. Moreover, wrappers not only result in a good subset of relevant features but they also produce a trained predictive model. These methods are believed to be the most accurate; however, some criticize them as simple "brute force" methods since they require a massive amount of computation [38].

One of the most common search strategies that is used in conjunction with linear regression is sequential search. This type of regression called *Stepwise linear regression*. There are 3 variations of sequential search:

1. **Forward selection**. The regression is initialized with only one feature added into the model that is the most correlated with the response variable. Then it searches for a feature that reduces a mean squared or absolute error the most. If this decrease of the mean error is greater than some threshold, we continue adding new features until new independent variable does not decrease the mean error for more than the threshold value.

2. **Backward elimination**. This variation of regression starts with all the independent variables added into the model. Then it disregards one feature at a time that does not increase mean error for more than the threshold. We stop when we can not remove any additional feature from the subset of chosen features without increasing the predictive error for too much. This process is opposite to Forward selection. Unfortunately, we can not use this method alone, if the number of observations we have is less than the number of features.

3. **Bidirectional search**. This method is a combination of the above two. We start with only one feature correlated with the response variable, then we apply Forward selection until no more additional features can be added into the model.

After that the Backward elimination procedure starts, until no more features can be removed from the model without increasing the mean error too much. We continue repeating these 2 search heuristics, until we reach a subset of independent variables that can not be alternated by any of the two procedures.

In this work we will be using Stepwise regression coupled with the Bidirectional search strategy since it is the most commonly used search strategy of Feature Selection [28].

- **Embedded Methods**. Embedded methods are machine learning and statistical algorithms that have a built-in Feature Selection. Usually they include a regularization term that penalizes the objective function of the method if it adds too many features into the model (like in LASSO [49], Multivariate adaptive regression splines (MARS) [16] or Random multinomial logistic methods [37]) or any method that uses pruning as an intermediate step in the model building process for making the model less complex, like in Decision Trees [19].

2.4 Mixed-Integer Linear Programming

Linear programming (**LP**) is an optimization technique, where the objective function we need to maximize or minimize is expressed as a linear combination of positive real variables with additional linear constraints. Formally speaking, Linear Programming is an optimization problem given as follows:

$$\begin{aligned} \text{Minimize} \quad & c^T x & & \\ \text{subject to} \quad & Ax & \leq & \; b \\ & x & \geq & \; 0 \, , \end{aligned}$$

where $c^T x$ is a linear objective function with coefficients c and problem's variables x, and $Ax \leq b$ are the constraints. The goal is to minimize the value of the objective function for all the possible values of the variables x that satisfy constraints $Ax \leq b$.

Thus, we can reformulate LP as:

Given a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, find a vector $x \in \mathbb{R}^n$ that minimizes the following optimization problem:

$$\min_x \{c^T x : Ax \leq b \text{ and } x \geq 0\} . \tag{2.7}$$

The first algorithm for solving Linear Programming problems was introduced by Dantzig in 1947. The algorithm was later called the Simplex method. The complexity of the algorithm is exponential in the number of variables and constraints, meaning that in the *worst case* it will not produce results in reasonable time. However, it was proved to produce an optimal result for majority of the instances of Linear Programming in polynomial number of steps [46], and so the Simplex method has become one of the most widely used algorithm for solving Linear Programming problems [51].

Mixed-Integer Linear Programming or MILP is a modification of the Linear Programming problem. Informally, MILP is the Linear Programming problem with an additional constraint that some or all the variables must be integers or they must take values from some countable domain:

$$\begin{array}{rrcl} \text{Minimize} & c^T x + d^T y & & \\ \text{subject to} & A_1 x + A_2 y & \leq & b \\ & x & \geq & 0 \\ & y & \in & \mathbb{D}^n , \end{array}$$

where $\mathbb{D}$ is some countable set, like the set of binary digits $\{0, 1\}$ or the set of integers $\mathbb{Z}$.

Mixed-Integer Linear Programming is computationally harder than Linear Programming. In fact, MILP is a known NP-hard problem [48].

Contemporary LP-solvers use many heuristics as subroutines in their optimized search, that accelerate the search for an optimal solution; thus, allowing us to solve problems even of substantial size in reasonable time [7]. MILP has been successfully used

Most studies on MILP deal with planning problems with limited number of resources, like vehicle route planning [13], trajectory planning for aircrafts [25], scheduling satellite launches [10], etc. In several cases the MILP approach is used for machine learning applications, like modelling SVM [54] or clustering [35] problems. Only a few papers have been published where Mixed-Integer Programming was used specifically for regression and Feature Selection, see [56] and [8]. Unfortunately, the suggested models use Quadratically Constrained Quadratic Programming (QCQP) that we cannot solve at present time, since QCQP is a hard computational problem and there are no algorithms that can produce near-optimal results.

Chapter 3

Methods

In this chapter we describe Mixed-Integer Linear programs we developed for finding robust regression models with automatic Feature Selection. We start the chapter by introducing notation we use for describing the MILP programs, then we introduce our MILP linear regression and in the last section we present a description of the MILP piece-wise linear regression.

3.1 Preliminaries

We are given a training data set D, which can be described as a finite set of pairs $D = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_n \in \mathbb{R}^d$ is a row-vector of corresponding values of independent variables and $y_i \in \mathbb{R}$ is an associated value of a dependent variable, n is the total number of training instances in the set and d is the number of available independent variables. By x_{ij} we will denote the j-component of vector x_i.

Let λ be a maximum number of features that predictive model can utilise. By limiting the total number of predictors we can create linear regression even for data sets, where the number of features is greater than the number of training instances, which is impossible in the traditional linear regression analysis.

Let $nz(\beta)$ be the function that accepts a vector $\beta = (\beta_0, \beta_1, ..., \beta_n)$ and returns the number of non-zero components of vector β. In other words, $nz(\beta) = \sum_i sign(\beta_i)^2$.

By γ we will denote the maximum number of outliers that can be detected by the model. Let $D' \subseteq D$ be a subset of "good" observations from the training data set D. Then, $|D - D'| \leq \gamma$.

Our goal is to build an approximating function $\bar{f}(x, \beta)$, such that:

$$y = \bar{f}(x, \beta) + \epsilon \ , \tag{3.1}$$

where $\epsilon \in \mathbb{N}(0, \sigma^2)$ is random normally distributed noise and

$$y_i \approx \bar{f}(x_i, \beta) + \epsilon \ \text{ for all } (x_i, y_i) \in D' \ . \tag{3.2}$$

Using Linear Programming for constructing linear regression models is not a new approach. Parameters of linear regression with the least absolute deviations (LAD) lack-of-fit function are most often calculated using Linear Programming [4] [42, 53]. We improve regular LAD regression by including into a regression linear program binary variables that can greatly enhance the model and make it possible to detect outliers and relevant features or independent variables. To do so we combine the ideas from [56] for detecting outliers and the idea from [36] for bounding the total number of independent variables used in the model, and apply them to linear regression and piece-wise linear regression with LAD lack-of-fit function to get robust predictive models computed using MILP.

3.2 Linear Regression using MILP

The optimization problem of the regular linear regression with the LAD lack-of-fit function can be stated as:

$$\min_{\beta}\{ \sum_{(x_i,y_i)\in D} |y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij}|\} \ . \tag{3.3}$$

Informally, the above problem seeks for a combination of vector β that would minimize the total absolute deviation between the predicted value and the actual response, measured on all the observations in data set D. However, we would like to enhance the above problem by adding additional constraints on the total number of utilised attributes λ and the maximum number of outliers γ our model can disregard. More formally, we would like to solve the following optimization problem:

$$\min\{ \sum_{(x_i,y_i)\in D'} |y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij}| \ : \beta \in \mathbb{R}^d, D' \subseteq D \ such\,that\, nz(\beta) \ \leq \ (\lambda+1) \ and \ |D - D'| \ \leq \ \gamma\} \ . \tag{3.4}$$

This problem is identical to the original optimization problem of the LAD linear regression (3.3), except that we added two additional components. First, the constraint $nz(\beta) \leq$

$(\lambda + 1)$ bounds the total number of non-zero components of vector β by $\lambda + 1$. Since parameters β define a regression linear relation, any $\beta_i = 0$, where $i \neq 0$, would mean that the attribute number i has no effect on the regression hyperplane. Parameter β_0 is not related to any attribute and so we don't need to bound its value. This is the reason why we use $(\lambda + 1)$ as a bounding term and not simply λ.

The second component is a bound on the total number of outliers $|D - D'| \leq \gamma$. Recall, that by D' we denote a subset of the original observations that we consider as "good" observations. So the observations that are not in D' will be considered as outliers and will not affect the model.

This way we defined three components of robust MILP linear regression with Feature Selection:

- **Component** $\mathbb{A}(D)$ defines the objective function.
- **Component** $\mathbb{B}(D, \lambda)$ defines the constraint on the total number of features used by the model.
- **Component** $\mathbb{C}(D, \gamma)$ defines the constraint that bounds the total number of potential outliers.

$\mathbb{A}(D)$ *component*. To model the component $\mathbb{A}(D)$ as a linear program, we will use the method developed in [1]. They showed, that LAD linear regression

$$\min_{\beta} \sum_{(x_i, y_i) \in D'} |y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij}| \tag{3.5}$$

can be rewritten as a linear program in the following form:

$$\begin{aligned}
\text{Minimize} \quad & \textstyle\sum_i s_i && (3.6)\\
\text{subject to} \quad & y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij} \leq s_i && (3.7)\\
& y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij} \geq -s_i && (3.8)\\
& s_i \geq 0 && (3.9)
\end{aligned}$$

for every $(x_i, y_i) \in D$.

$\mathbb{B}(D,\lambda)$ *component*. To model the component $\mathbb{B}(D,\lambda)$ by linear constraints, we introduce binary variables A_i for $i = 1..d$:

$$A_i = \begin{cases} 1, & \text{if a feature } i \text{ is used} \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

Having these variables, we can limit the number of used features by adding the following constraints:

$$\sum_{i=1}^{d} A_i \leq \lambda \tag{3.11}$$
$$-MA_i \leq \beta_i \tag{3.12}$$
$$MA_i \geq \beta_i \text{ ,} \tag{3.13}$$

where M is some big positive constant.

Since all A_i's are binary variables, constraint (3.11) makes sure that we do not use more than λ features in the final model. Constraints (3.12) and (3.13) bound values of coefficients β_i. When $A_i = 0$, meaning the attribute i should not be used in the model, the corresponding coefficient β_i will be bounded to take only one value which is zero. In other words, in linear regression linear program (3.7)-(3.8) the ith attribute will not contribute anything into the model. However, if $A_i = 1$, then β_i can take any value from the range $[-M, M]$.

$\mathbb{C}(D,\gamma)$ *component*. To implement the component $\mathbb{C}(D,\gamma)$ which incorporates outlier detection into our model making it more flexible, we will add another collection of binary variables O_i for $i = 1..n$, where

$$O_i = \begin{cases} 1, & \text{if observation } i \text{ is an outlier} \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

To bound the number of outliers that our model can exclude from the training process, we will add the following inequality:

$$\sum_{i=1}^{n} O_i \leq \gamma \text{ .} \tag{3.15}$$

If some observation i is considered as an outlier, we assume that we made some mistake while measuring it and, since we do not know what is the ``true'' value of the observation, we can not penalize our regression model on this training instance. To simulate this in MILP we modify constraints (3.7-3.8) in the following way:

$$y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij} \leq s_i + MO_i \tag{3.16}$$

$$y_i - \beta_0 - \sum_{j=1} \beta_j x_{ij} \geq -s_i - MO_i \ , \tag{3.17}$$

where M is some big constant.

To explain how the last two constraints work we consider two cases. First, the simplest case is when observation i is a ``good" observation. This means that $O_i = 0$ and constraints (3.16-3.17) become identical to constraints (3.7-3.8). Second, if $O_i = 1$ then the ith observation is considered as an outlier. Keeping in mind that objective function (3.6) tries to minimize each positive s_i as much as possible, we can conclude that no matter what the value of the ith residual is, while it stays in the range $[-M, M]$, s_i is restricted to be zero. So we do not penalize our regression function. This way, the observation *i* belongs to the data set of "well-behaving" observations D' if and only if $O_i = 0$.

The size of the above Mixed-Integer Linear program is the following:

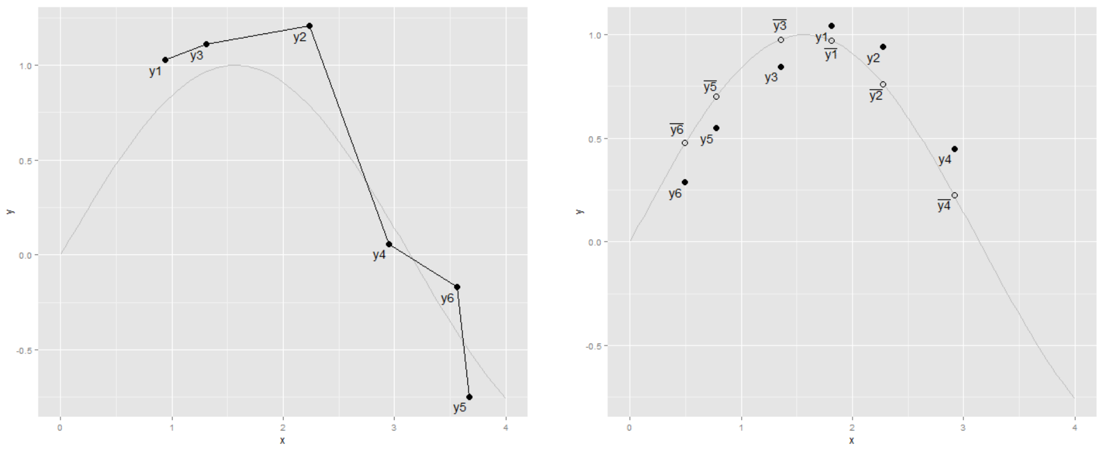
- **Binary variables**. The total number of binary variables is $n+d$, where n is the number of observations in the training data set D and d is the number of available attributes in D.
- **All variables**. The total number of all variables used in the program is $3n + 2d + 1$, since we introduced n instances of s_i and y_i variables, as well as $d + 1$ variables of β_j and includes the $n + d$ binary variables O_i and A_j.
- **Constraints**. The total number of constraints is $5n + 2d + 2$.

Theorem 1. The MILP program constructed above performs robust linear regression with automatic Feature Selection (3.4) that can tolerate at most γ outliers and utilises at most λ attributes.

Proof. By the above construction of the components $\mathbb{A}(D)$, $\mathbb{B}(D, \lambda)$, and $\mathbb{C}(D, \gamma)$. ■

3.3 Piece-wise linear regression using MILP

To model piece-wise or segmented linear regression as a MILP program we will slightly modify the basic idea of segmented regression, described in Section 2.1.



(a) (b)

Figure 3.1: Examples of the difference between regular segmented regression and MILP piece-wise regression. The black points represent the observations, grey curve in the background is the ``true'' generating function. The unfilled circles on the right represent the ``true'' value of each observation, that we would receive if there were no errors made during the collection of the training data set.

Recall, that in segmented regression we use observations as basis points for building approximating simplexes. However, we assumed that each observation is measured with some normally distributed error and so we should keep in mind that approximations using this technique are calculated with errors from all the basis points.

In our method we try to minimize these errors by learning not only what the most relevant features are but also the best possible position of the basis points. To visualize the difference between approaches, we give two artificial examples shown in Figure 3.1. Figure 3.1(a) shows an example of regular segmented regression. There are 6 consequent observations that are basis points on which approximating lines are built. However, since the observations have error terms they contribute this error into the approximating simplexes.

In Figure 3.1(b), we see how our approach tries to deal with the errors of the training instances. Instead of relying on the raw observations $\{y_1, ..., y_6\}$, MILP piece-wise regression tries to find what is the hypothetical ``true'' value of each observation $\{\bar{y_1}, ..., \bar{y_6}\}$, that would be measured under ideal conditions without any deviations from the underlying generating function and simplexes are built on these hypothetical ``true'' points as on the basis points (see Figure 3.2).

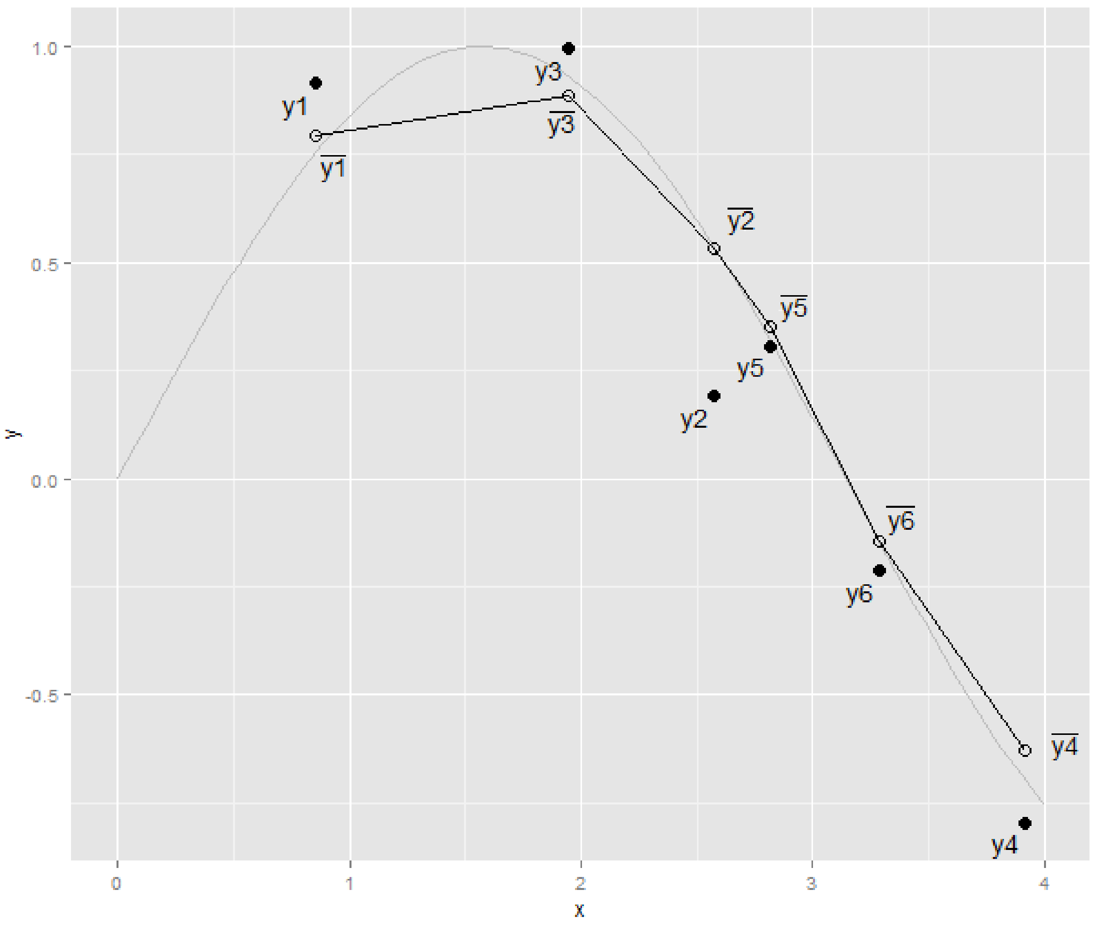


Figure 3.2: Example of the final result of MILP piece-wise regression applied to some artificial data set that consists of 6 training points. The ``true'' values of the observations depicted as unfilled circles do not lie on the generating function (grey curve in the background), but since they are ``squeezed'' as tight as possible to the generating function, they should serve as better basis points for approximation simplexes than the raw observations.

To find these hypothetical ``true'' values for each training instance $p_i = (x_i, y_i) \in D$ we associate two real valued variables: $\bar{y}_i$ will serve as a hypothetical ``true'' value of the dependent variable without a noise component, meaning $y_i = \bar{y}_i + \epsilon$, and the second variable y_{pi} will serve as a predicted value for the point p, estimated using segmented regression.

We assume that the underlying function can be approximated using only the local information and so we would like to build a model that for each training point $p_i \in D'$ minimizes the distance between the predicted value y_{pi} and the ``true'' response value $\bar{y}_i$, since we would like to build as tight approximating functions as possible, as well as the distance between the measured value $y_i$ and the ``true" value $\bar{y}_i$, because we assume that the ``true" value should lie not too far from the real, measured value of response variable:

$$\min_{\bar{y}_i, y_{pi}} \{\sum_i |\bar{y}_i - y_{pi}| + |\bar{y}_i - y_i|\} \ . \tag{3.18}$$

The result of our MILP piece-wise regression is a set of linear approximations defined on each subdomain that are bounded by the convex region of the basis points.

To write a MILP program that can compute piece-wise regression, we add the following variables:

1. Binary variables A_i. They serve as the indicator variable for deciding if the feature number i ($i = 1..d$) is used in the model or not. More formally,

$$A_i = \begin{cases} 1, & \text{if a feature } i \text{ is used} \\ 0, & \text{otherwise.} \end{cases} \tag{3.19}$$

2. Binary variables O_i. They serve as the indicator variable for deciding if the observation number i ($i = 1..n$) is considered as an outlier:

$$O_i = \begin{cases} 1, & \text{if an observation } i \text{ is an outlier} \\ 0, & \text{otherwise.} \end{cases} \tag{3.20}$$

3. Binary variables C_{ijk}. These indicator variables shows if the training point p_j is the kth closest point to the point p_i ($i, j = 1..n$ and $k = 1..(\lambda + 1)$). We will use these variables to detect subdomain regions.

4. Real variables d_{ij}. These variables hold the information about the squared euclidian distance between points p_i and p_j and they depend on the selection of features used in the model (determined by A_i).

5. Real variables δ_{ik}. We use these variables to track what is the kth closest distance from the point p_i to other training points.

6. Real variables β_{ij}. These variables define the linear coefficients of the jth approximating hyperplane, where $i = 0..d$ and $j = 1..n$.

7. Real variables s_i. These variables represent the distance between predicted value y_{pi} and the ``true'' value of the response variable in the ith observation for $i = 1..n$.

8. Real variables t_i. These variables represent the distance between y_i and $\bar{y}_i$ for $i = 1..n$.

We divide the MILP piece-wise linear regression program into 5 components:

- **Component** $\mathbb{A}(D)$ defines the objective function.
- **Component** $\mathbb{B}(D, \lambda, i)$ defines the constraint on the total number of features used for predicting the point p_i.
- **Component** $\mathbb{C}(D, \gamma)$ defines the constraint that bounds the total number of potential outliers.
- **Component** $\mathbb{D}(D, \lambda, i)$ defines the component that detects exactly $(\lambda + 1)$ closest neighbours of the point p_i.
- **Component** $\mathbb{E}(D, i, C = (C_{ijk}), A_l)$ defines the approximating simplex for the training point $p_i \in D$. It utilises information from a matrix of binary variables C that indicates if some point p_j is considered as a basis point for p_i, and also makes sure that the approximating hyperplanes do not rely on features that are not used in the model.

$\mathbb{A}(D)$ *component*. To implement component $\mathbb{A}(D)$ we will use the same idea as in MILP linear regression to rewrite the optimization problem (3.18) as a linear program:

$$\begin{aligned}
\text{Minimize} \quad & \sum_i s_i + t_i & & & (3.21)\\
\text{subject to} \quad & y_{pi} - \bar{y}_i & \leq & \; s_i & (3.22)\\
& y_{pi} - \bar{y}_i & \geq & \; -s_i & (3.23)\\
& \bar{y}_i - y_i & \leq & \; t_i & (3.24)\\
& \bar{y}_i - y_i & \geq & \; -t_i & (3.25)\\
& s_i & \geq & \; 0 & (3.26)\\
& t_i & \geq & \; 0 & (3.27)
\end{aligned}$$

for each $(x_i, y_i) \in D$. In this program, s_i is the distance between the predicted value and the ``true'' value of the corresponding observation and $t_i$ is the absolute deviation between the ``true'' value of the observation i and its measured value y_i.

$\mathbb{B}(D, \lambda)$ *component*. To make sure the number of features used by the model is equal to λ, we introduce the following constraint:

$$\sum_i A_i = \lambda \ . \tag{3.28}$$

Since each A_i is a binary variable, their sum gives the exact number of the features used in the model and so the constraint (3.28) constraints the model to use exactly λ independent variables in the final model.

$\mathbb{C}(D, \gamma)$ *component*. To find potential outliers we can use the same idea we used in MILP linear regression by adding constraints like (3.15) and (3.16)-(3.17) into our model.

To bound the number of outliers that our model can exclude from the training process, we will add the following constraint:

$$\sum_{i=1}^{n} O_i \leq \gamma \ . \tag{3.29}$$

If some observation i is considered as an outlier, we assume we made a significant error while measuring it, it is better for the model to disregard the observation than to try to estimate its ``true'' value $\bar{y}_i$. To simulate this in MILP we modify constraints (3.22)-(3.25) in the following way:

$$\begin{aligned} y_i - \bar{y}_i &\leq s_i + MO_i && (3.30)\\ y_i - \bar{y}_i &\geq -s_i - MO_i && (3.31)\\ y_{pi} - \bar{y}_i &\leq t_i + MO_i && (3.32)\\ y_{pi} - \bar{y}_i &\geq -t_i - MO_i \ , && (3.33) \end{aligned}$$

where M is some big constant.

$\mathbb{D}(D, \lambda, i)$ *component*. After adding component $\mathbb{B}(D, \lambda)$, we know that (λ+1) is the total number of dimensions used by the model and for each point $p_i \in D$ we need to find its $(\lambda + 1)$ closest points, build a hyperplane that crosses all these basis points and evaluate $\bar{y}_i$ using the hyperplane.

We introduce variables d_{ij} to store the squared euclidian distance from the points p_i to all other points:

$$d_{ij} = \sum_{k=1}^{d} (x_{ik} - x_{jk})^2 A_k \ . \tag{3.34}$$

Since we do not know in advance what features will be eventually used in the model, the above constraint makes sure the MILP program calculates the correct distances for any set of used attributes.

To find the first closest point to the point p_i we need to obtain the index j that provides the minimum possible distance d_{ij}. This index j defines the closest to p_i point p_j. To do so we add the following inequalities to the MILP program:

$$\begin{aligned} \delta_{i1} &\leq d_{i1} \\ \delta_{i1} &\leq d_{i2} \\ &\dots \\ \delta_{i1} &\leq d_{in}\ . \end{aligned} \tag{3.35}$$

These constraints force variable δ_{i1}, which should hold the information about the first closest distance to the point p_i, to be not greater than the minimum possible distance d_{ij} from the point p_i to any other point in the training data set. However, being not greater than the minimum distance doesn't mean it will be equal to the minimum distance. To force variable δ_{i1} to take the value of the minimum distance, we first introduce auxiliary variables a_{ijk} that we force to be equal to $C_{ij1} \times d_{ij}$. Although this relation is non-linear, it can be simulated using the following constraints:

$$a_{ij1} \leq MC_{ij1} \tag{3.36}$$

$$a_{ij1} \leq d_{ij} \tag{3.37}$$

$$a_{ij1} \geq d_{ij} - M(1 - C_{ij1}) \tag{3.38}$$

$$a_{ij1} \geq 0\ , \tag{3.39}$$

where M is some positive big constant.

The validity of these constraints can be checked by examining the possible cases of values that variable C_{ij1} and d_{ij} can take. If C_{ij1} is equal to zero, then the constraints (3.36) and (3.39) force the variable a_{ij1} to be equal 0. However, when $C_{ij1} = 1$, then from the inequalities (3.36) and (3.37) variable a_{ij1} can take any value from the range $[0, .., d_{ij}]$. At the same time, constraint (3.38) forces variable a_{ij1} to be not greater than d_{ij}. Combining the two allowed ranges, a_{ij1} is forced to take exactly the value of d_{ij}. So by adding the above constraints we make variable a_{ij1} to simulate C_{ij1} and d_{ij}.

Next we need to make sure that there is only one closest point to the point p_i:

$$\sum_{j=1}^{n} C_{ij1} = 1 \tag{3.40}$$

and finally to bound δ_{i1} to be not less than the minimum distance by using variables a_{ij1}:

$$\delta_{i1} \geq \sum_{j} a_{ij1}\ . \tag{3.41}$$

Since there is only one non-zero C_{ij1}, from the constraints (3.36) - (3.39) there will be only one non-zero a_{ij1}, and constraint (3.41) forces δ_{i1} to take the value of the minimum distance.

To find second, third or *k*th neighbour (where $k = 1..(\lambda + 1)$), we need to modify the above constraints so that we do not select a previously found point as the next closest neighbour. Since we store all the found basis points in variables C_{ijk}, it is easy to control which points we can select on the next round and which we cannot:

$$\begin{aligned}
\delta_{ik} &\leq d_{i1} + M \sum_{t=1}^{k-1} C_{i1t} && (3.42)\\
\delta_{ik} &\leq d_{i2} + M \sum_{t=1} C_{i2t} && (3.43)\\
& \dots && (3.44)\\
\delta_{ik} &\leq d_{in} + M \sum_{t=1} C_{int} && (3.45)\\
\delta_{ik} &= \sum_{j} a_{ij1}\\
a_{ijk} &\leq MC_{ijk}\\
a_{ijk} &\leq d_{ij}\\
a_{ijk} &\geq d_{ij} - M(1 - C_{ij1})\\
a_{ijk} &\geq 0\\
\sum_{j=1}^{n} C_{ijk} &= 1\ . && (3.46)
\end{aligned}$$

Now, if the model selects some point p_j as a closest on the previous step, it does not select it again, since the modified constraints (3.42 - 3.45) do not bound the value of variable δ_{ik} by distance variable d_{ij} if point p_j was previously selected as a basis point for the point p_i.

So after adding component $\mathbb{D}(D, \lambda, i)$ we have a list of basis points for approximating point p_i all stored in C_{ijk}.

$\mathbb{E}(D, i, C = (C_{ijk}), A_l)$ *component*. Component $\mathbb{E}(D, i, C = (C_{ijk}), A_l)$ utilizes the information from C_{ijk} to calculate the values of parameters β of the approximating hyperplane:

$$\beta_{i0} + \sum_{k=1}^{d} \beta_{ik} x_{jk} \leq M(1 - C_{ijk}) \tag{3.47}$$

$$\beta_{i0} + \sum_{k=1}^{d} \beta_{ik} x_{jk} \geq -M(1 - C_{ijk})\ . \tag{3.48}$$

On the left hand side of the two above constraints we see a linear relation for defining hyperplanes; however, on the right side they will be equal to zero if and only if $C_{ijk} = 1$, meaning that the point p_j is a basis point for the point p_i. If $C_{ijk} = 0$, then the linear relation in (3.47) -(3.48) can take any value from the range $[-M, M]$; therefore, these constraints are void. These 2 constraints make sure that only the closest basis points define the approximating hyperplane.

At the same time, since we do not use all the available features we need to bound the values of β_{ik} parameters to be equal 0 if the attribute k is not used. To do so we include the following constraints for each $k = 1..d$:

$$-MA_k \leq \beta_{ik} \leq MA_k\ . \tag{3.49}$$

So if $A_k = 0$, meaning the feature k is not used, then $\beta_{ik} = 0$ for all the approximating hyperplanes. However, if the model uses the attribute k, then $\beta_{ik} \in [-M, M]$

Theorem 1. The MILP program constructed above performs robust piece-wise linear regression with automatic Feature Selection that can tolerate at most γ outliers and utilises exactly λ attributes.

Proof. By the above construction of the components $\mathbb{A}(D)$, $\mathbb{B}(D, \lambda, i)$, $\mathbb{C}(D, \gamma)$, $\mathbb{D}(D, \lambda, i)$ and $\mathbb{E}(D, i, C = (C_{ijk}), A_l)$. ■

The size of the above Mixed-Integer Linear program is the following:

- **Binary variables**. The total number of binary variables is $n^2\lambda + d + n$, where n is the number of observations in the training data set D and d is the number of available attributes in D.

- **All variables**. The total number of all the variables, including binary variables, in the above program is $n\lambda + +2n^2\lambda + 2n^2 + 5n + 2d$.

- **Constraints**. The total number of constraints is $n\lambda + +2n^2\lambda + 2n^2 + nd + 7n + 3$.

As we can see, this model has a number of advantages highlighted in Section 2.1, however, the increased number of binary variables in the model makes it computationally much more intensive than MILP linear regression and so it can be used only for small or medium size data sets at the present time. Another drawback is the need to normalize the data sets prior to modelling, since the model relies on euclidian distances and so all the attributes must be measured in identical scales. We can achieve this by standartization or by scaling all independent variables to the range $[0, 1]$ if possible.

One more disadvantage is the iterative nature of the above approach. If we are not satisfied with the current selection of λ and γ, we need to select the new values and repeat the calculations.

Nevertheless, MILP piece-wise regression gives more flexibility in creating predictive models. First, it can approximate non-linear relations where linear regression fails. Second, since a piece-wise regression model is linear, it is easy to interpret and so if an analyst can not obtain acceptable predictive performance using traditional regression approaches, he/she can use suggested MILP piece-wise linear regression instead of relying on machine learning algorithms that act as ``black boxes''.

Chapter 4

Experiments & Results

To compare the suggested approaches with the existing statistical predictive tools, we use 3 data sets provided by the Vancouver Prostate Centre. They consist of a list of peptides with computed *in silico* physico-chemical properties and assayed biological activities. The goal is to find a regression function based solely on the provided training data sets that could predict biochemical activities of previously unseen peptides.

These data sets were ideal as an experimental benchmark, due to a number of reasons. First, the number of independent variables, which are computed physico-chemical properties, is much higher than the number of observations in each data set. Second, since we deal with results of biochemical experiments, the data sets carry very volatile activity values and so it is hard to distinguish good observations from outliers. Third, after finding a good regression function we can conclude which physico-chemical properties are relevant for predicting activities we are interested in; thus, providing some insight for understanding the deeper biochemical mechanisms of these biological activities.

4.1 Diagnostic Plots

Since we are using linear models for predicting linear and non-linear relations, we can use well-developed statistical tools of residual analysis to compare predictive models.

We will present 4 plots that are widely used in the linear regression analysis [15]:

- **Actual vs Fitted**. This plot shows how well the model fits the data. We would like our model to predict the dependent variable as close as possible to its actual value and if the predicted value is plotted against its corresponding actual value we expect all

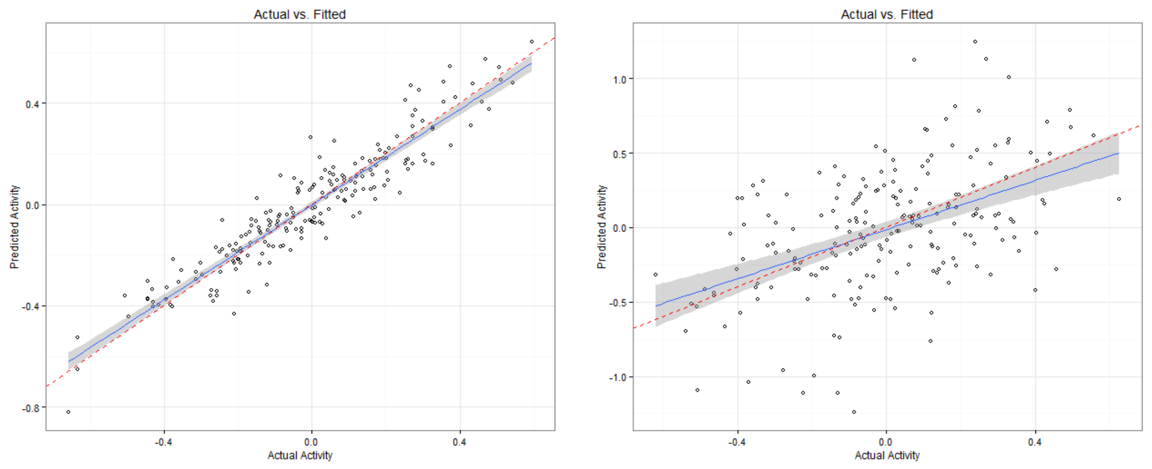


Figure 4.1: Two examples of the Actual vs. Fitted plots. On the left we have the plot that we expect to get after performing regression analysis, while the plot on the right show regression that failed to build a good model

the points to lie approximately on the line $y = x$. The closer all the points are plotted to this line the better the model.

Figure 4.1 shows two artificial examples of the Actual vs. Fitted plots. In these figures we see a set of points that deviate around the dashed line depicting $y = x$. In an ideal predictive model, all the predictions would be identical to actual values of the response variable; thus, all the points would lie on the dashed line. However, in the real-world predictive model we expect predictions to deviate slightly around the line $y = x$. The more substantial this deviation, the less reliable the model. Additionally, we plot a solid line of simple linear regression between the predicted and actual values. The closer its slope and intercept to the same characteristics of the dashed line, the better the model. This regression line also allows us to show the confidence interval of predictions. It is depicted in Figure 4.1 as a grey area around the regression line.

The picture on the left shows a plot we expect to get after any regression analysis. All the points deviate slightly around the $y = x$ line, indicating that the model was able to capture a good relation between response variable and predictors.

The Actual vs. Fitted plot on the right indicates that the corresponding regression model didn't find a good predictive relation. We can see that the deviation of residuals is comparable with the deviation of the original data points, because in average the magnitude of residuals (that is the distance from each point to the line of ideal regression) is close to the magnitude of changes of the response variable.

- **Fitted vs. Residuals**. This plot shows residuals plotted against predicted values of a dependent variable. Since we expect residuals to behave like normally distributed random variables, then we should not see any systematic components in the residuals distribution, they should not form an increasing or decreasing sequence. Otherwise, we can conclude that the assumption about constant variance of the error term is violated. ``Well behaving'' residuals are expected to oscillate around the zero level of the plot.

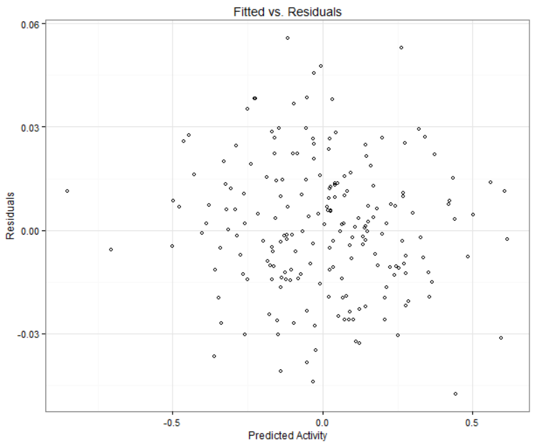
  In Figure 4.2 we present four examples of Fitted vs. Residuals plots of models of different quality. The first plot (Figure 4.2(a)) is a plot we expect to receive after conducting successful regression modelling. The residuals deviate around the zero value and they do not form any trends or patterns in their distribution. Even just from this plot we can conclude that the residuals are more or less normally distributed.

  On the other hand, Figure 4.2(b) shows not so good distribution of points. Although, they seem randomly distributed but all of them are above zero value meaning that all the training observations are located on one side of the corresponding regression hyperplane. This could happen if we stop Stepwise regression too early or if there was a significant outlier that shifted the hyperplane towards itself.

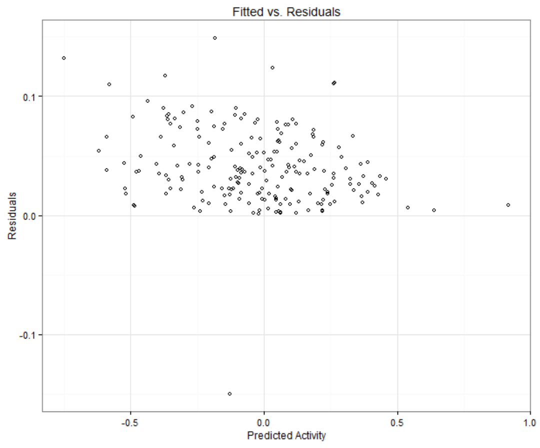
  Another bad example of the Fitted vs. Residuals plot is shown in Figure 4.2(c). There one can see that the variance of residuals increases as the predicted value grows. This may happen when we failed to capture all the relevant independent variables or there is a non-linear interaction between predictors.

  Finally, plot 4.2(d) shows noticeable functional behaviour of the residuals. Most probably, such pattern suggests that there should be a quadratic component included into the model, that can account for pair-wise interactions between predictors.

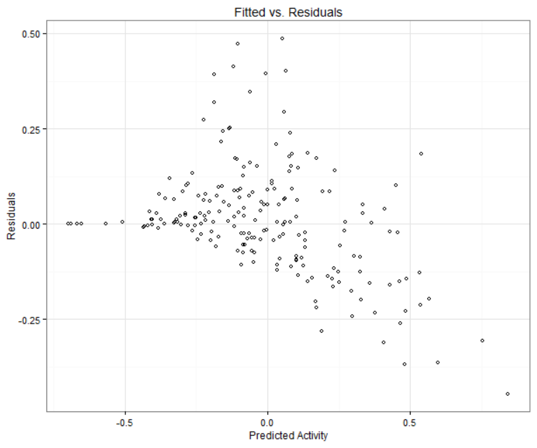
  To summarize, the well behaving predictive regression models should not form any patterns in the distribution of points in the Fitted vs. Residuals plot. Otherwise, we should suspect that some assumption is violated or we did not capture all the relevant variables.



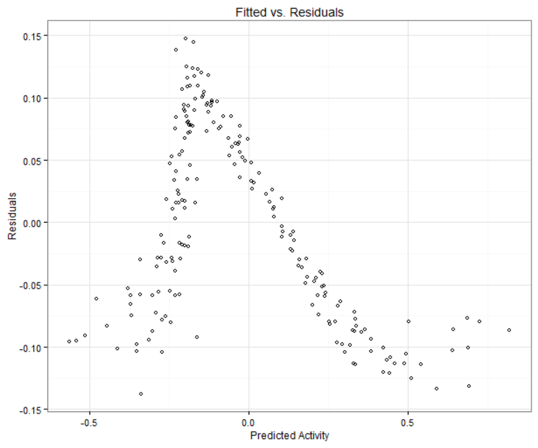
(a) A well behaving plot



(b) The plot shows only positive residuals



(c) Residual's variance increases as predicted values grow



(d) Residuals form a noticeable pattern in their distribution

Figure 4.2: Examples of the Fitted vs. Residual plots.

- **Actual vs Residuals**. This plot is similar to the previous one, except that the residuals are plotted against the actual values of the dependent variable. This plot can also be used for detecting violation of the constant variance assumption. In other words, if we notice that residuals form an increasing or decreasing sequence, we can assume that the dependent variable has some sort of inner memory; thus, linear regression model can not be applied.

  We expect this plot to behave the same very way as Fitted vs. Residuals plot.

- **Residuals Normal Q-Q plot**. This plot compares quantiles of the residuals distribution with quantiles of the two normal distribution. If two distributions are alike, then all the points should lie approximately on the line $y = x$. Since we assume that the noise term is normally distributed, then we expect all the residuals to lie around $y = x$.

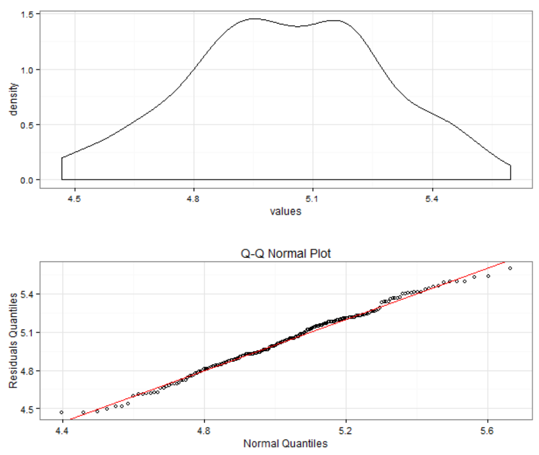
  In Figure 4.3 we see two examples of random processes. On the left hand side, Figure 4.3(a) shows a probability density function that looks similar to the Gaussian distribution. If we take a sample of random variables from this distribution and plot a Normal Q-Q plot for the sample, we will receive a graph similar to the one depicted in the bottom half of Figure 4.3(a). All points lie close to the line $y = x$ and this is a good sign that the sample was produced by a distribution similar to the normal distribution.

  However, in Figure 4.3(b) we can see an example of a random process that is far from being normally distributed and the corresponding sample, taken from such distribution, plotted on a Q-Q Normal plot. The difference between the two distributions is quite noticeable.

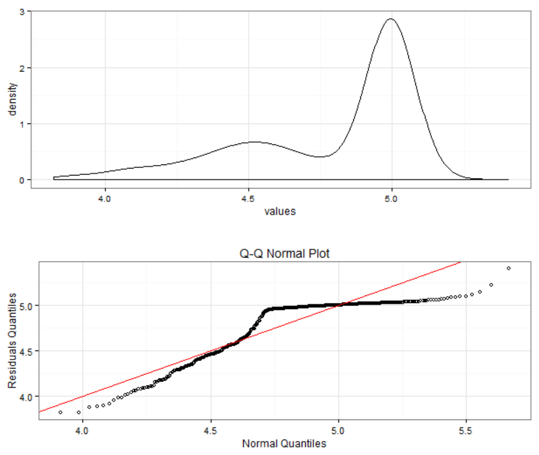
By visually studying these 4 diagnostic plots we can answer the following questions about the quality of regression: a) How good does the model fit predicted values to the original values of the response variable? b) Are outliers normally or close to normally distributed? c) Is there any skewness in the distribution of the residuals? d) Does the variance of the residuals remain constant among all the observations?

4.2 Data Overview

To compare our models with the existing statistical approaches we used 3 real-world data sets provided by the Vancouver Prostate Center. Each data set consists of 108



(a)



(b)

Figure 4.3: Examples of the Q-Q Normal plots.

peptides with measured immunomodulatory biological activity and 652 computed physico-chemical properties (or descriptors). Biological activities are measured in concentrations of a specific chemical compound called MCP1 cytokine after applying each peptide from the corresponding data set to a culture of donor cells. Units of measurements of the response variables are picograms per milliliter (pg/ml). Physico-chemical properties include a wide range of molecular characteristics, like molecular weight, total charge, total negative/positive charge on the surface, total area of charged molecule's surface, etc.

These data sets were collected as a part of the bigger project. The goal of which is to find a set of chemicals that potentially can destroy biofilms in the living bodies and control the inflammatory processes. It is believed, that biofilms cause more than 80% of all infections, and they are hard to treat due to their natural resistance against many known antibacterial drugs and antibiotics, since biofilms are structural formations rather than living organisms [34]. In living bodies, many bacteria grow on a surface rather than floating in planktonic form. They form colonies protected by biofilms that serve as a protective layer between the bacterial colony and aggressive host environment. Moreover, biofilms contribute to better microenvironment for bacteria growth.

Although biofilm itself is not dangerous for health, it promotes bacteria spread in the host organism; thus, resulting in worsening infectious state. Another threat from biofilms is that they might be the reason of sepsis, since they can constantly produce small amounts of bacterial cells which are killed either by the immune system of the host organism or by antibiotics. However, constant immune reaction might lead to increase of proinflammatory cytokines like MCP1 that, after reaching some level, can cause sepsis [14].

So having a medicine that can control production of these cytokines can greatly reduce the number of sepsis cases.

In our experiments we used peptides derived from three parental peptides called 1018, 1002, and HH2. It is known, that the peptide 1018 can potentially prevent and destroy the biofilms formed by multidrug-resistant pathogens [39] and control production of MCP1 cytokines, while peptides 1002 and HH2 were only suspected to have similar activities. All these peptides are 12 amino acid long and contain of 9 different amino acids. These 9 amino acids are believed to be highly relevant for the peptide to have the antibiofilm and immunomodulatory activity.

To create the initial library of compounds, chemists used these parental peptides as a basis for constructing new sequences of amino acid chains by replacing each amino acid in the parental peptide with one of the amino acids from the list of relevant amino acids; thus, from each parental peptide they received a list of 108 peptides that they assayed to have the immunomodulatory activity.

After the peptides were experimentally evaluated in their ability to eradicate biofilms, we began computing their physico-chemical properties. For that purpose, we used Molecular Operating Environment (MOE) computer program. Each peptide from the prepared library of compounds was virtually modelled using molecular mechanics technique and their properties were computed *in silico* using MOE. This way we collected a total of 652 descriptors that we used for modeling the regression function for approximating immunomodulatory activity.

4.3 Experiments

We are interested in building a predictive model that would allow us to select the most relevant physico-chemical properties for a peptide to be considered as active and to be able to predict the immunomodulatory activity for any other peptide without conducting wet lab experiments.

By having such a model, we can achieve two goals. First, since we can predict activities of peptides virtually, we do not spend time and money for synthesizing and assaying non active peptides. Second, if we know what physico-chemical properties are relevant for a peptide to be considered as active, it can give us a key for understanding the biochemical mechanisms of immunomodulatory peptides.

In drug design field and QSAR, the most often used predictive tool is Multivariate Linear Regression with Least Squares lack-of-fit estimator coupled with Stepwise Feature Selection algorithm [2]. This method is called Stepwise Linear Regression. The algorithm is believed to be the best predictive tool among linear models when we have more features than training instances (see [52] for examples of applying Stepwise Regression to real-world analysis problems).

Before applying Stepwise Regression, we used the box plot method [5] with $IQR = 1.5$ to exclude some observations that we marked as outliers. We did not remove any observations prior to applying MILP Regression approaches. We also standardized all the descriptors and the response variable in each data set so that they all have a mean value of 0 and standard deviation equal to 1.

Each data set has its own name that indicates a special identification code of the parental peptide from which all the peptides in the corresponding data set were derived. Here we will also use these names for convenience.

To show differences in predictive performances between regression approaches we will rely on examining the diagnostic plots described in the beginning of the Section, as well as on numeric criteria such as Coefficient of Determination (R^2) and the number of the used features.

First data set consisted of 108 peptides derived from a parental peptide named 1018. After applying Stepwise Regression, we obtained a predictive model that used 30 features and the Coefficient of Determination R^2 was 0.47. Figure 4.4 shows the model's diagnostic plots.

Stepwise regression wasn't been able to pick the correct relation between dependent and independent variables. First concern is with the Actual vs. Predicted plot. It clearly shows, that the predictions do not approximate the actual activity of peptides. Because of that all the other plots show some problems as well, like obvious trending on the Actual vs. Residuals plots.

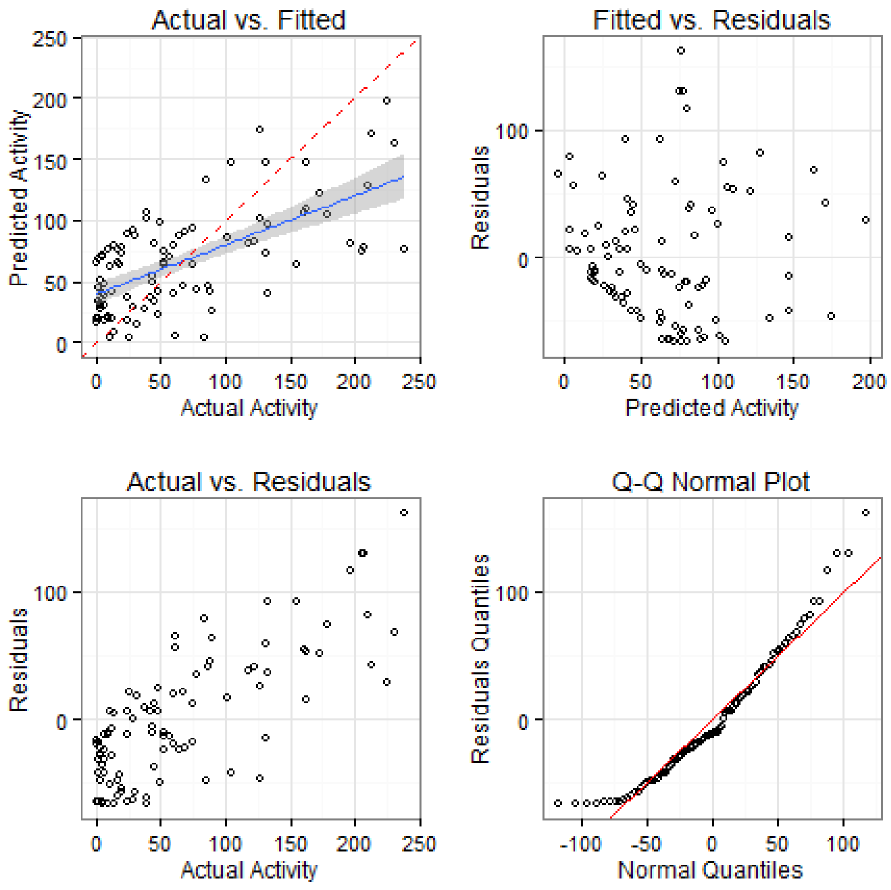


Figure 4.4: Stepwise Regression results for 1018 data set. These plots reveal serious problems of the linear model constructed via Stepwise regression. The model wasn't been able to find any good relation between response and independent variables. Moreover, we can see a pattern in distribution of residuals against observed values of the dependent variable.

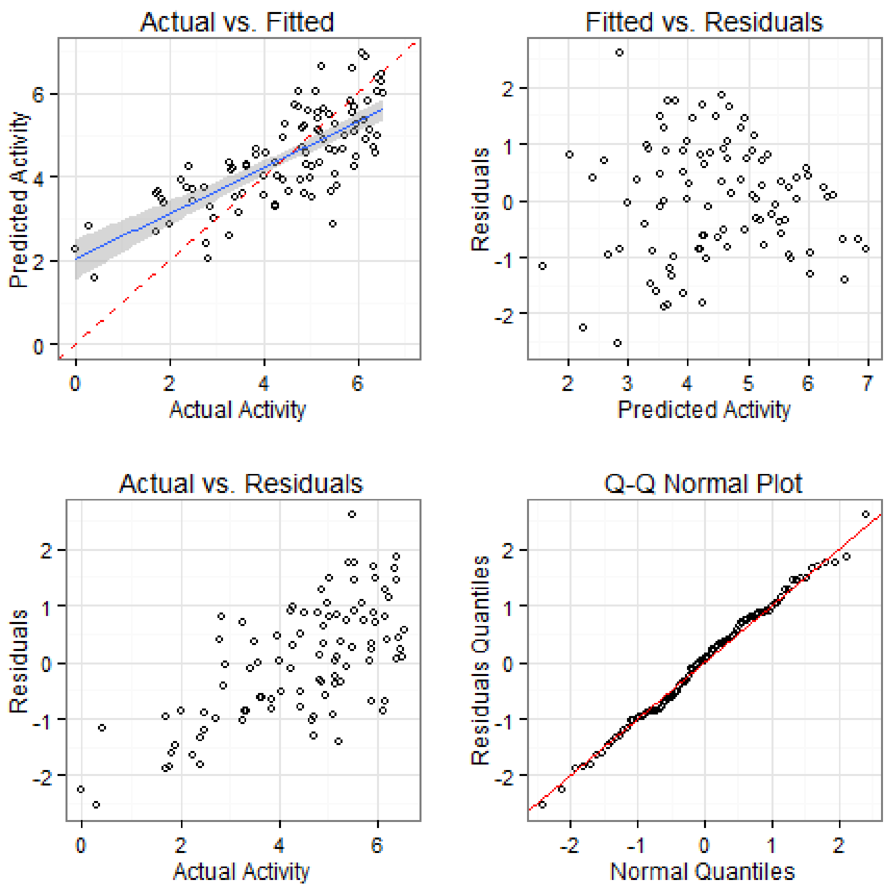


Figure 4.5: Stepwise Regression results for 1002 data set. On these plots we see the same issues as in the previous plots.

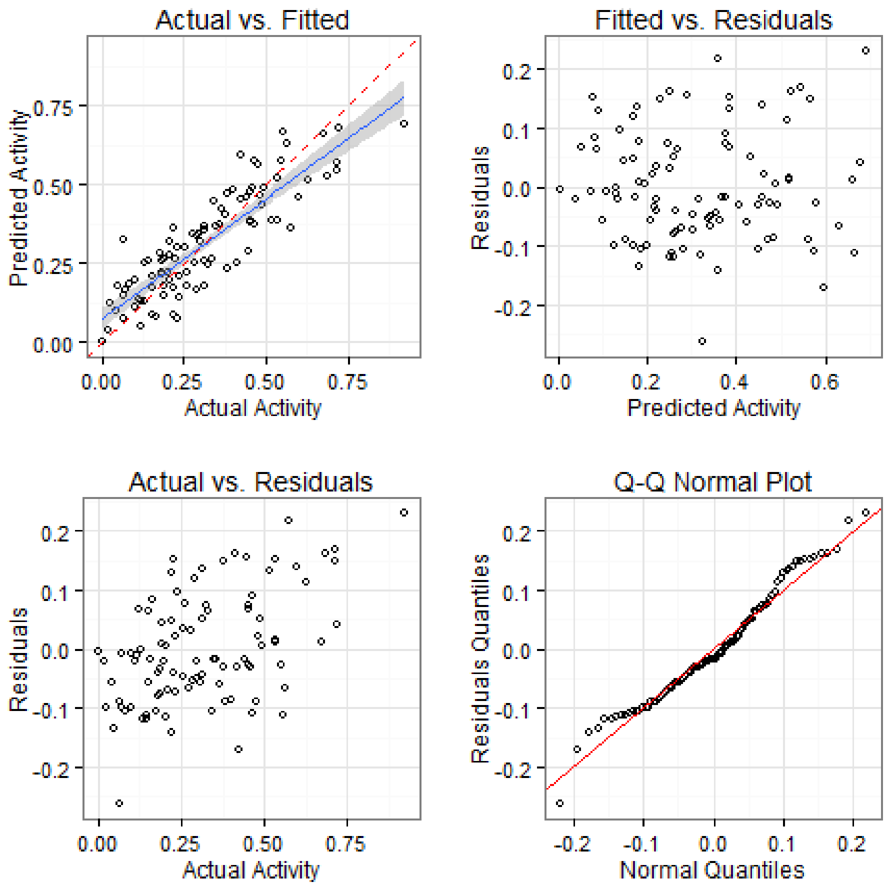


Figure 4.6: Stepwise Regression results for HH2 data set. This model seem good. We have more or less normally distributed residuals and predicted values lie not far from line $y = x$ line.

Somewhat similar pictures we obtained for the 1002 data set (Figure 4.5) too. It seems like Stepwise Regression wasn't successful in finding a good and relevant subset of features in this case as well and so it produced a model with bad predictive performance if we look at the Actual vs. Predicted plot, as well as the clear trending in the distribution of residuals on the Actual vs. Residuals. For 1002 data set the Stepwise Regression found a model with 18 descriptors that resulted in $R^2 = 0.62$.

The performance of the model on the HH2 data set is significantly different. Here we see a well-behaving model that was able to capture all the required linear relations between the response variable and descriptors, resulting in the model with 23 descriptors and $R^2 = 0.82$.

For computing MILP programs we used the industry-leading LP-solver CPLEX 12.4 by IBM. We set up CPLEX on 10 machines with 4 cores 3.2 GHz Intel® Xeon® with 64Gb of memory each. During the calculations of the MILP program, CPLEX was able to harness all the cores of a machine. We limited the running time of the CPLEX to 6 hours per data set for MILP linear regression and to 12 hours per data set for MILP piece-wise regression. We left all other CPLEX settings unchanged.

In Figures 4.7, 4.11 and 4.10 one can see the diagnostic plots for MILP Linear regression. Altogether these plots show reliable predictive models with minimum Coefficient of Determination $R^2 = 0.78$ for 1002 data set. Moreover, besides being able to pick better models than the Stepwise regression, our approaches have built models that use fewer descriptors.

In Figure 4.7 we can clearly see that the regression line (dashed line) on the Actual vs. Fitted plot has similar slope with the line of ideal prediction $y = x$. Additionally, majority of residuals have magnitude less than 30 (Actual vs. Residuals plot in Figure 4.9), while the Stepwise regression model built for the same data set results in prediction errors that have magnitudes comparable with the response variable. The Q-Q Normal plot indicates almost normal distribution of the residuals.

As for the MILP piece-wise regression for the 1018 data set(Figure 4.11), we obtained results similar to MILP linear regression with two noticeable distinctions. First, the magnitude of errors is much lower even comparing to MILP linear regression. Second, if we look at the Q-Q Normal plot we can notice that the distribution of residuals does not look like normal. However, such deviation is not significant and we can accept this model.

For 1002 and HH2 data sets we were able to receive good models as well. The magnitude of residuals of the MILP piece-wise linear model of 1002 data set is twice as low as in the MILP linear regression model of the 1002 data set and 4 times lower than in Stepwise regression of the corresponding model! Moreover, in Stepwise regression residuals seem to form some kind of functional relation (see Figure 4.5), while MILP based models produce residuals that do not show any obvious trends.

Similar results we received for HH2 data set (see Figures 4.11 and 4.12). Although, Stepwise regression was able to capture a good linear model but MILP linear regression and MILP piece-wise regression have produced models that have even lower residuals magnitude while having fewer features used in the models, comparing to the model received after applying Stepwise regression. At the same time all residuals seem normally distributed which indicates that we have obtained good linear predictive models.

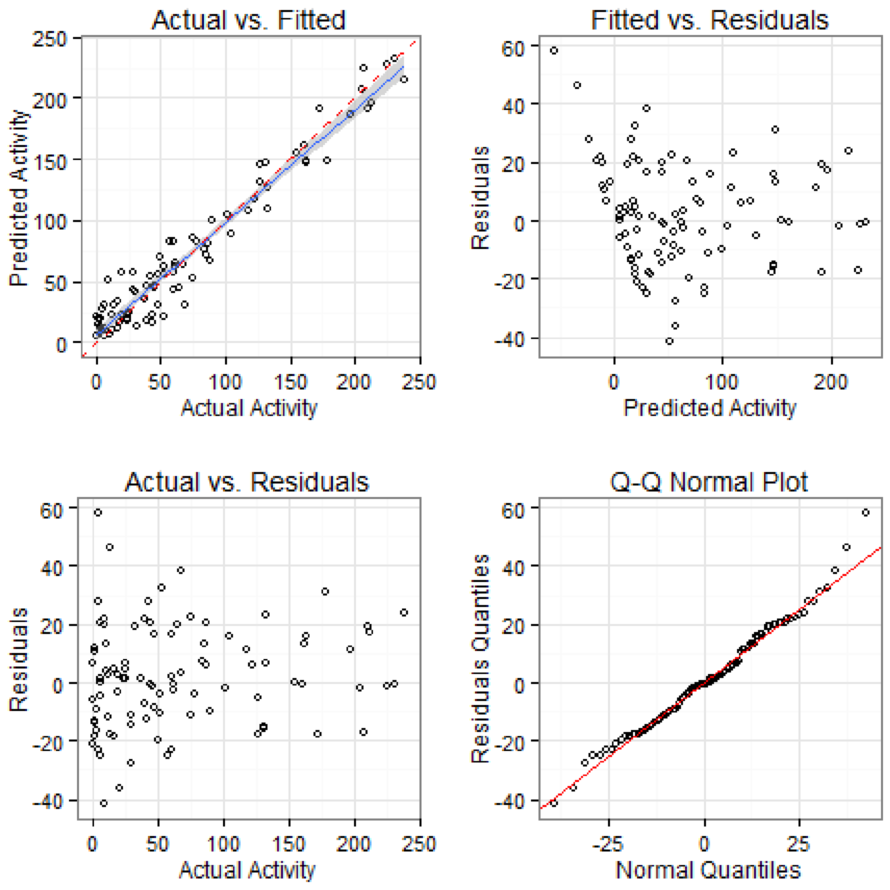


Figure 4.7: MILP Linear regression results for the 1018 data set

In the Table 4.1, we summarized numerical statistics for each model and data set we used.

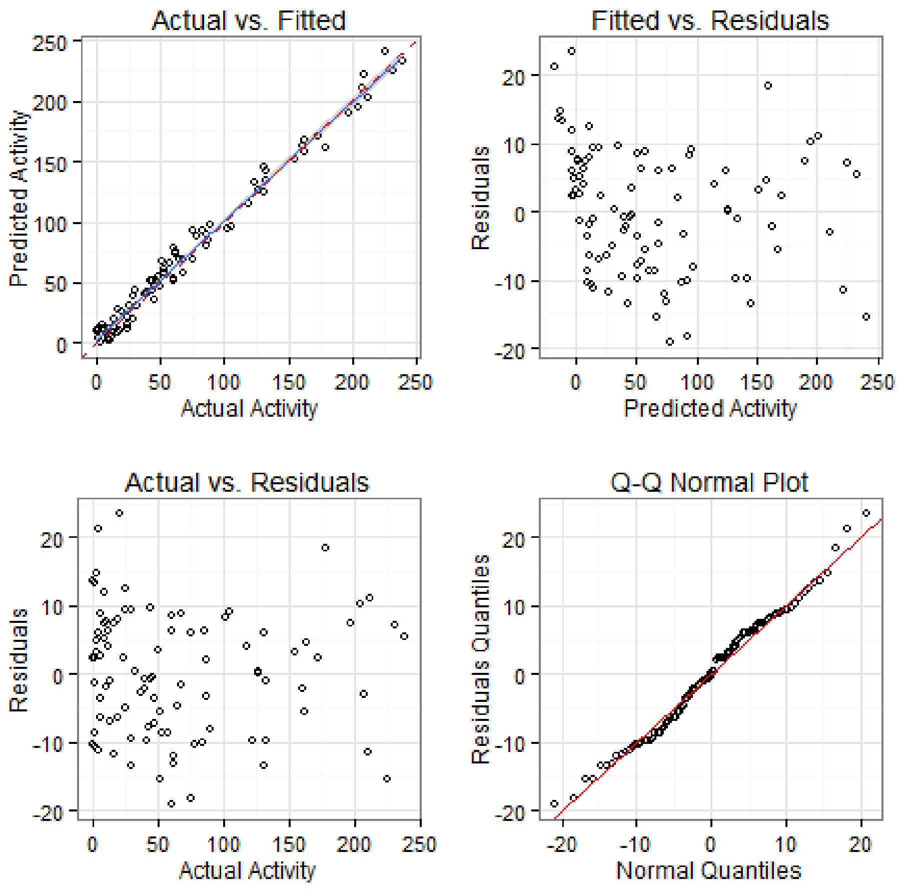


Figure 4.8: MILP Piece-wise regression results for the 1018 data set

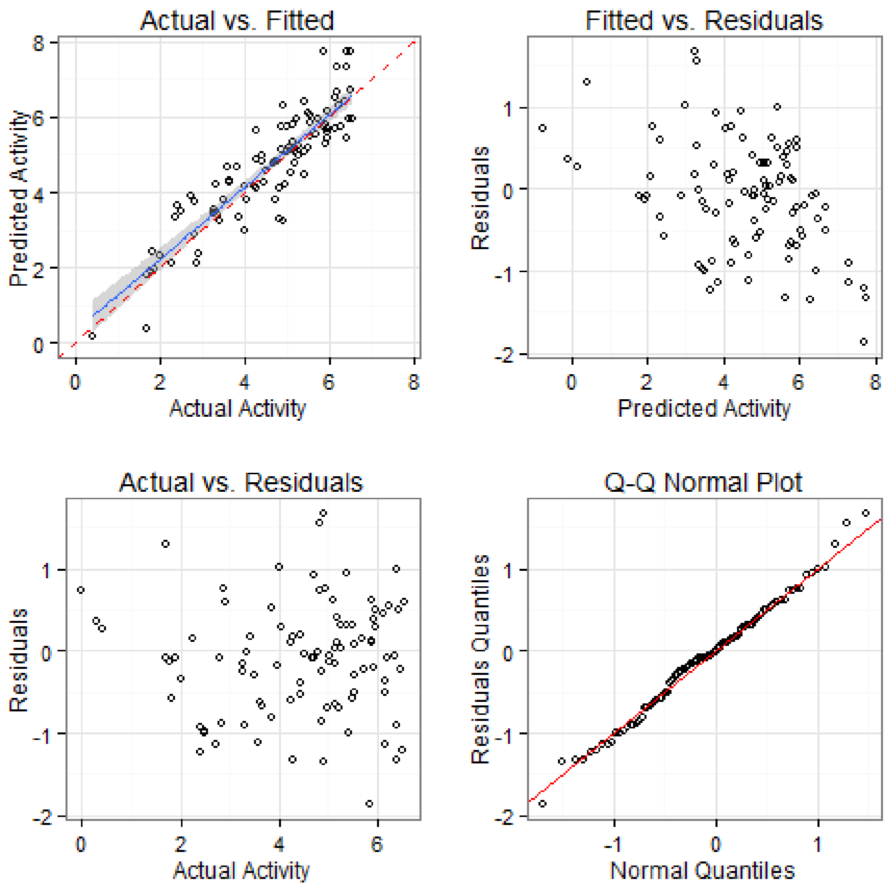


Figure 4.9: MILP Linear regression results for the 1002 data set

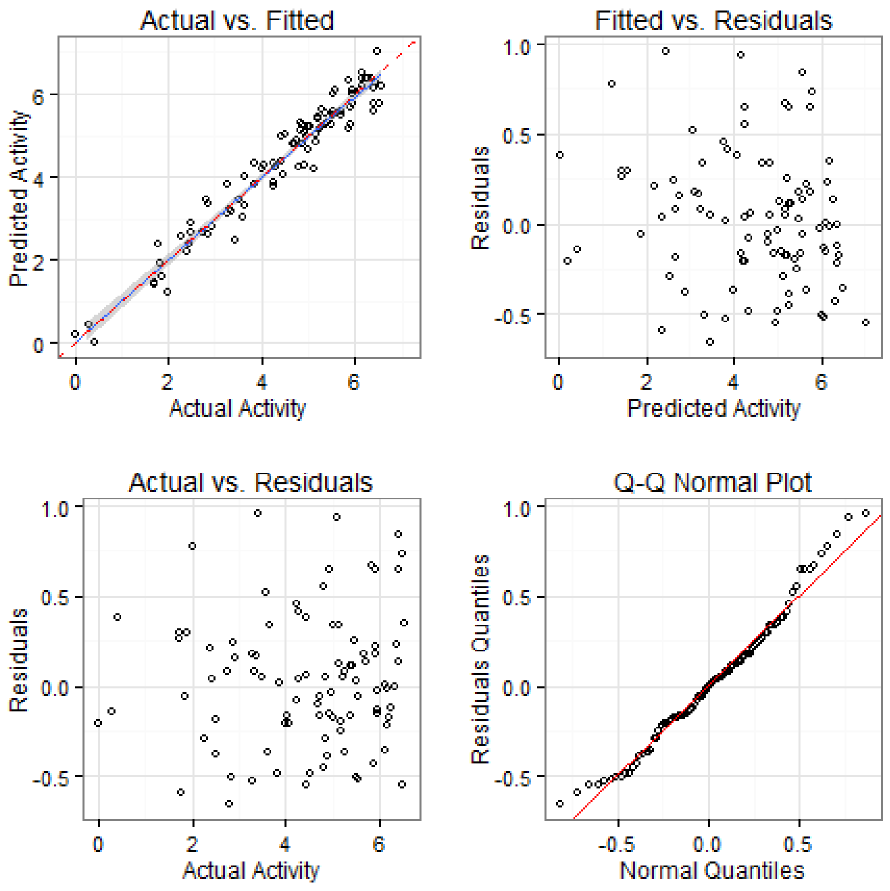


Figure 4.10: MILP Piece-wise regression results for the 1002 data set

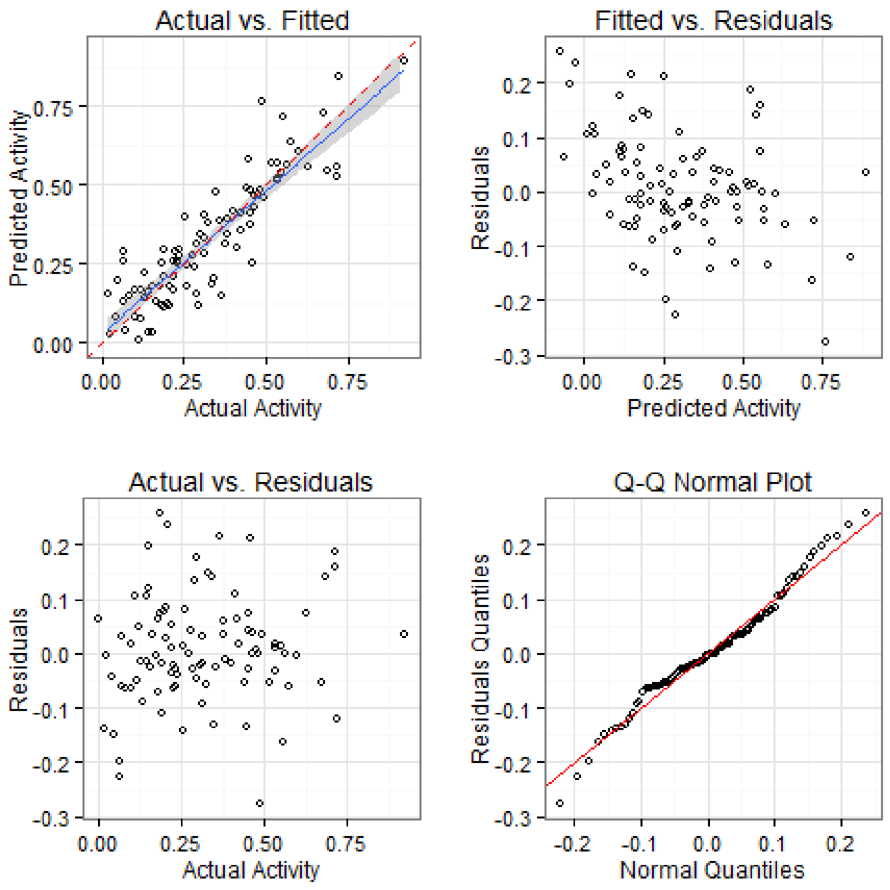


Figure 4.11: MILP Linear regression results for the HH2 data set

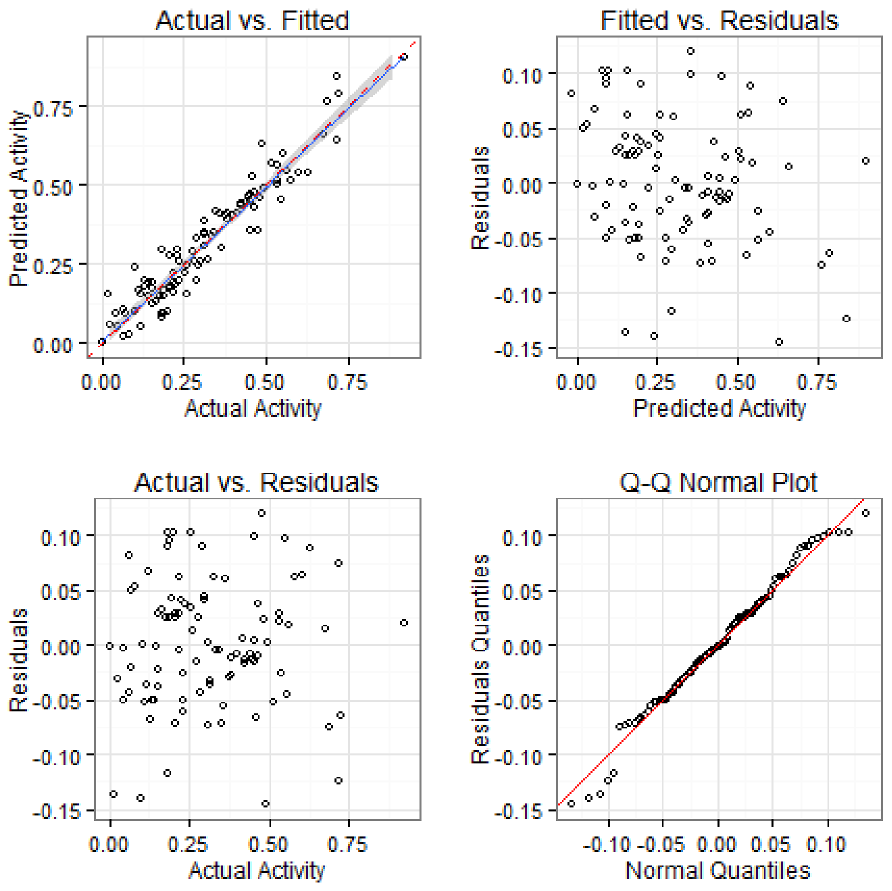


Figure 4.12: MILP Piece-wise regression results for the HH2 data set

Table 4.1: Comparative statistics for each predictive model used in the thesis

Method	Data set	Outliers Detected	# of Features used	R^2
Stepwise Regression	1002	5	18	0.60
	1018	3	30	0.47
	HH2	7	23	0.82
MILP Linear Regression	1002	5	15	0.77
	1018	4	15	0.88
	HH2	4	10	0.80
MILP Piece-wise Regression	1002	4	14	0.87
	1018	4	13	0.90
	HH2	3	9	0.88

Chapter 5

Conclusions

In this work we presented two new Mixed-Integer Regression approaches for building predictive models. We described the linear model as well as the novel piece-wise linear model, the latter is capable of capturing even non-linear relations. Moreover, the suggested MILP piece-wise linear regression model is fully empirical and, thus, it can act as universal approximator. Additionally, since all the models are linear, they are easy to interpret which is crucial especially in QSAR studies where predictive models are used not only as quantitative tools but also as exploratory to understand the inner mechanisms of biochemical reactions.

The suggested models include automatic Feature Selection, which is especially important for Pattern Recognition problems, and potential outliers detection, which is useful for every data analysis problem.

To compare our approaches against the traditional regression tools we used 3 data sets provided by the Vancouver Prostate Center and we showed that commonly used Stepwise Regression was not able to yield good predictive models or to predict dependent variable's response. Our methods showed much better results, they not only yielded reliable predictive models but also successfully reduced the number of used features to 13-15 in average from a list of 652 features.

Additionally, since the MILP approach can give us an optimal or near-optimal solution, then if we are not satisfied with the MILP Linear regression results we can switch to more powerful but much more time consuming MILP Piece-wise regression; thus, the problem of deciding which model to use becomes easier to deal with.

Bibliography

[1] W.W. Cooper A. Charnes and R. Ferguson. Optimal estimation of executive compensation. *Management Science*, 2:138--151, 1955. 24

[2] L. Alaei A. Maleki, H. Daraei and A. Faraji. Comparison of QSAR models based on combinations of genetic algorithm, stepwise multiple linear regression, and artificial neural network methods to predict K_d of some derivatives of aromatic sulfonamides as carbonic anhydrase II inhibitors. *Russian Journal of Bioorganic Chemistry*, 40:61--75, 2014. 5, 43

[3] D. W. Aha and R. B. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. 1994. In: Proceedings of the AAAI-94 Workshop on Case-Based Reasoning. 15

[4] I. Barrodale and F. D. K. Roberts. An improved algorithm for discrete L1 linear approximation. *SIAM Journal on Numerical Analysis*, 10:839--848, 1973. 23

[5] Yoav Benjamini. Opening the box of a boxplot. *The American Statistician*, 42:257--262, 1988. 43

[6] Rosner Bernard. Percentage points for a generalized ESD many-outlier procedure. *Technometrics*, 25:165--172, 1983. 2

[7] Timo Berthold. Primal MINLP heuristics in a nutshell. *Informationstechnik*, 2013. 20

[8] Dimitris Bertsimas and Romy Shioda. Classification and regression via integer optimization. *Operations Research*, 55:252--271, 2007. 20

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2007. 2, 16, 17

[10] Paolo Brandimarte. *Scheduling satellite launch missions: an MILP approach*, volume 16. Journal of Scheduling, 2013. 20

[11] Martin D. Buhmann. *Radial basis functions: theory and implementations*, volume 12. Ebrary Academic Complete, 2003. 9

[12] R. Dennis Cook. Detection of influential observations in linear regression. *Technometrics*, 19(1):15--18, 1977. 13

[13] M.G Earl and R D'Andrea. *Iterative MILP methods for vehicle control problems*, volume 4. IEEE Conference on Decision and Control (CDC), 2004. 20

[14] Cristina Ciornei et al. Biofilm forming p. aeruginosa induces an enhanced inflammatory response in human monocytes. *Critical Care*, 11, 2007. 42

[15] David A. Freedman. *Statistical Models: Theory and Practice*. Cambridge University Press, 2005. 1, 36

[16] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1--67, 1991. 19

[17] P. I. Good and J. W. Hardin. *Common Errors in Statistics (And How to Avoid Them)*. International Encyclopedia of Statistics. New Jersey:Wiley, 3 edition, 2009. 1

[18] F. E. Grubbs. Procedures for detecting outlying observations in sampless. *Technometrics*, 11:1--21, 1969. 2, 13

[19] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 2003. 3, 16, 19

[20] R. Dubois et al. H. Stoppiglia, G. Dreyfus. Ranking a random feature for variable and feature selection. *JMLR*, 3:1399--1414, 2003. 17

[21] Peter. J. Huber. *Robust Statistics*. Wiley, 1981. 15

[22] Boris Iglewicz and David Hoaglin. *How to Detect and Handle Outliers*, volume 16 of *The ASQC Basic References in Quality Control: Statistical Techniques*. 1993. 2

[23] Hodge V. J. and Austin J. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85--126, 2004. 2, 13

[24] M. Embrechts et al. J. Bi, K. Bennett. Dimensionality reduction via sparse support vector machines. *JMLR*, 3:1229--1243, 2003. 17

[25] W.A Kamal, Da-Wei Gu, and I Postlethwaite. *Real Time Trajectory Planning for UAVs Using MILP*. IEEE Conference on Decision and Control (CDC), 2005. 20

[26] Mati Karelson. *Molecular descriptors in QSAR/QSPR*. John Wiley & Sons, 2000. 3, 4, 5

[27] Hyune-Ju Kim, Michael P. Fay, Binbing Yu, Michael J. Barrett, and Eric J. Feuer. Comparability of segmented line regression models. *Biometrics*, 60(4):1005--1014, 2004. 11

[28] R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97:273--324, 1997. 19

[29] William H. Kruskal and Judith M. Tanur. *Linear Hypotheses*, volume 1 of *International Encyclopedia of Statistics*. Free Press, 1978. 1

[30] Juhola M. Laurikkala J. and Kentala E. Informal identification of outliers in medical data. 2000. 13

[31] K. Leonard M. Hechinger and W. Marquardt. What is wrong with quantative-structure property relations models based on three-dimensional descriptors? *Journal of Chemical Information and Modeling*, 52:1984--1993, 2012. 4

[32] C.L. Mallows. *On Some Topics in Robustness. Unpublished memorandum, Bell Telephone Laboratories*. New Jersey:Murray Hill, 1975. 15

[33] I Naseem, R. Togneri, and M. Bennamoun. Linear regression for face recognition. *Pattern Analysis and Machine Intelligence*, 32:2106--2112, 2010.

[34] Michael Givskov et al. Niels Hoiby, Thomas Bjarnsholt. Antibiotic resistance of bacterial biofilms. *International Journal of Antimicrobial Agents*, 35:322--332, 2010. 41

[35] DG Oliva, G Guillen-Gosalbez, JM Mateo-Sanz, and L Jimenez-Esteller. *MILP-based clustering method for multi-objective optimization: Application to environmental problems*, volume 56. Computers & Chemical Engineering, 2013. 20

[36] Martine et al. Perez, Juan; Labbe. Feature selection for support vector machines via mixed integer linear programming. *Information Sciences*, 279:163--167, 2014. 23

[37] Friedman J.; Hastie T.; Tibshirani R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33, 2010. 19

[38] John O. Rawlings, Sastry G. Pantula, and David A. Dickey. *Applied Regression Analysis: A Research Tool*. Springer texts in statistics. Springer, 2 edition, 1998. 6, 7, 10, 18

[39] Mansour S. Reffuveille F., de la Fuente-Nunez C. and Hancock R. A broad-spectrum antibiofilm peptide enhances antibiotic action against bacterial biofilms. *Antimicrobial agents and chemotherapy*, 58:5363, 2014. 42

[40] P. J. Rousseeuw and V.J. Yohai. Robust regression by means of s estimators in: Robust and nonlinear time series analyses. *Lecture Notes in Statistics No. 26*, pp. 256--272, 1984. 15

[41] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871—-880, 1984. 15

[42] E. J. Schlossmacher. An iterative technique for absolute deviations curve fitting. *Journal of the American Statistical Association*, 68:857--859, 1973. 23

[43] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. pp. 293--301, 1994. In: Machine Learning: Proceedings of the Eleventh International Conference. 15

[44] D. B. Skalak and E. L. Rissland. Inductive learning in a mixed paradigm setting. pp. 840--847, 1990. In: Proceedings of the Eighth National Conference on Artificial Intelligence. 15

[45] R.R. Sokal and F.J. Rohlf. *Biometry: The principles and practice of statistics in biological research*. International Encyclopedia of Statistics. New York:W.H. Freeman, 3 edition, 1995.

[46] Shang-Hua Spielman, Daniel; Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. pp. 296--305, 2001. Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing. ACM. 20

[47] David T Stanton. On the physical interpretation of QSAR models. *Journal of chemical information and computer sciences*, 43:1423, 2003. 4

[48] Eva Tardos and Jon Kleinberg. *Algorithm Design*. Pearson Education, 2006. 20

[49] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society*, 58:267--288, 1996. 19

[50] Tietjen and Moore. Some grubbs-type statistics for the detection of outliers. *Technometrics*, 14:583--597, 1972. 2

[51] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*, volume 114 of *International Series in Operations Research & Management Science*. Springer Verlag, 3 edition, 2008. 20

[52] T. D. Wallace. Efficiencies for stepwise regressions. *Journal of American Statistical Association*, 59:1179--1182, 1964. 43

[53] G. O. Wesolowsky. A new descent algorithm for the least absolute value regression problem. *Communications in Statistics Simulation and Computation*, B10:479--491, 1981. 23

[54] Gang Xu and Lazaros G Papageorgiou. *A mixed integer optimisation model for data classification*, volume 56. Computers & Industrial Engineering, 2009. 20

[55] V.J. Yohai. High breakdown-point and high efficiency robust estimates for regression. *Annals of Statistics*, 15:642–--656, 1987. 15

[56] Georgios Zioutas and Antonios Avramidis. Deleting outliers in robust regression with mixed integer programming. *Acta Mathematicae Applicatae Sinica*, 21(2):323–--334, 2005. 15, 20, 23