Empowering Knowledge Computing with Variable Selection

On Variable Importance and Variable Selection in Regression Random Forests and Symbolic Regression

Wouter Minnebo
Sean Stijven

Promotor:
Dr. Katya Vladislavleva

Universiteit Antwerpen

Universiteit Antwerpen

Empowering Knowledge Computing with Variable Selection

On Variable Importance and Variable Selection in Regression Random Forests and Symbolic Regression

Proefschrift
voorgelegd op 30 mei 2011 tot het behalen van
de graad van Master in de Wetenschappen,
bij de faculteit Wetenschappen,
aan de Universiteit Antwerpen.

Promotor:
Dr. Katya Vladislavleva

Wouter Minnebo
Sean Stijven

Acknowledgements

We express our gratitude to our promotor Katya Vladislavleva. Not only did she introduce us to the field of evolutionary computation and computational intelligence, she convinced us that it matters. Her desire to understand and learn inspired us to undertake the journey that is this thesis. During this journey we became aware of the prevalent issues in modeling and learned to appreciate elegant solutions to difficult problems. We enjoyed our discussions on how to create a better world where thinking is global and acting is local. Her helpful comments greatly improved this thesis. In addition we acknowledge her persistence as a driving force that guided and advised us to share our results with the world in a conference publication.

We thank Mark Kotanchek for providing licenses of the DataModeler package, which was extensively used and experimented with during the course of this thesis.

We thank prof. Serge Demeyer and the department of Computer Science and Mathematics for financial support and enabling our first international academic experience.

We thank our family and friends for their support during this turbulent year.

Wouter Minnebo

Sean Stijven

May 2011

Samenvatting

De focus van dit proefstuk is de kwantificatie van belangrijkheid van variabelen uitgaande van input-output data. We tonen aan dat interpretatie van de bijdragen van individuele variabelen enkel inzicht biedt in combinatie met een kwalitatief regressie model. Het opstellen van een regressie model wordt sterk bemoeilijkt door verschillende inherente eigenschappen van real life problemen. De problemen die we bestuderen zijn niet-lineair, en data sets kunnen mogelijk variabelen bevatten die niet relevant zijn. Het is waardevol een maatstaf te hebben om de belangrijke variabelen te kunnen beschrijven, zodat variabelen in eenzelfde model onderling vergelijkbaar zijn. Met deze informatie kunnen accuratere modellen gemaakt worden, en wordt meer inzicht geboden in het probleem. Tot op heden is er geen algemene methode om dit aan te pakken.

Om de belangrijkheid van variabelen na te gaan gebruiken we voornamelijk twee technieken, random forests en symbolische regressie. Beide technieken zijn in staat data van hoge dimensionaliteit te verwerken, maar ze verschillen sterk in hun aanpak en de modellen die ze produceren. Random forests combineert verscheidene classificatie en regressie bomen, terwijl symbolische regressie een populatie van algebraïsche expressies evolueert.

In Hoofdstuk 2 leggen we de belangrijkheid van een variabele vast en introduceren we eigenschappen die het mogelijk maken huidige en toekomstige methodes voor het berekenen ervan, onderling te vergelijken.

In Hoofdstuk 3 analyseren we random forests en het algoritme in detail. We bestuderen ook de parameter gevoeligheid van random forests aan de hand van intuïtieve voorbeelden.

In Hoofdstuk 4 wordt symbolische regressie gesitueerd in een ruimer kader van evolutionaire algoritmen. We illustreren deze techniek op eenvoudige problemen. Ook stellen we methoden op om belangrijkheid van variabelen te bepalen aan de hand van de expressie waar die in voorkomt. Dit leent zich tot integratie met symbolische regressie.

In Hoofdstuk 5 voeren we verschillende experimenten uit die meer inzicht bieden in het gedrag van de verscheidene technieken om belangrijkheid te bepalen. Ook onderzoeken we de impact van vaak voorkomende defecten in real life data sets.

In Hoofdstuk 6 passen we de vergaarde kennis toe op twee case studies. De eerste behandelt data verzameld door de Verenigde Naties, waarmee oorspronkelijk de Human Development Index bepaald werd. We modelleren verschillende variabelen en identificeren invloedrijke factoren door de belangrijkheid ervan te bepalen. De tweede case study behandelt een industriële data set van gas chromatografie metingen van een destillatie toren.

In Hoofdstuk 7 beschrijven we de software die ontwikkeld werd in het kader van deze thesis. In het bijzonder een C++ implementatie van random forests die via een grafische interface te bedienen is. Onze implementatie ondersteunt parallellisatie waardoor computers die beschikken over meerder cores de techniek sneller uitvoeren.

In de Appendices bespreken we vaak gebruikte technieken voor data analyse. Principale componenten analyse en neurale netwerken worden in respectievelijk Appendix A en B behandeld.

Summary

The focus of this thesis is to quantify the importance of variables starting from input-output data. We show that the interpretation of individual variable contributions is only meaningful in context of a regression model of sufficient quality. Constructing such regression model is hampered by several properties inherent to real life problems. The problems of interest are non-linear, and the data sets potentially contain irrelevant variables. It is desirable to find the driving variables by obtaining a measure relatively comparing variables from the same model. This information would allow the creation of more accurate models, and provide invaluable insight into the problem. At present no general method exists to acquire such measure.

We focus on obtaining variable importances using two modeling techniques, namely random forests and symbolic regression. Both techniques are capable of modeling high dimensional non-linear data but greatly differ in their approach and produced models. Random forests combines several classification and regression trees, while symbolic regression evolves a population of algebraic expressions.

In Chapter 2 we define variable importance and introduce properties by which existing and future methods for computing variable importance can be compared.

In Chapter 3 we analyze the random forest technique down to the algorithmic details. We also examine the sensitivity of the parameters of random forests using intuitive examples.

In Chapter 4 the symbolic regression technique is situated in the larger evolutionary computation framework. We apply symbolic regression on practical toy problems. In addition we propose methods for determining the importance of a variable directly from the algebraic expression. Such methods are of interest for integration in the symbolic regression framework.

In Chapter 5 we conduct several experiments on understandable problems to provide further insight in the behavior of different variable importance techniques. We also investigate the impact of common defects present in real life data sets.

In Chapter 6 we apply the knowledge gained in the previous chapters to two challenging case studies. The first one deals with data obtained by the United Nations for determining the Human Development Index. We model several output variables and derive importances to identify driving variables. The second case study deals with an industrial data set representing a gas chromatography measurement of a distillation tower.

In Chapter 7 we provide details about the software developed in context of this thesis. Notably the random forest technique is implemented in C++ and accessible through a graphical user interface. Our implementation also supports parallelization, enabling faster execution on multi-core machines.

In the Appendices we summarize background research on commonly used methods in data analysis. Principal component analysis and neural networks are covered in Appendices A and B respectively.

Contents

Chapter 1

Introduction

1.1 Motivation

Living in a technological age is a dream come true. We can accurately observe complex systems, and data collection is commonplace. Scientists have more data available than ever before and are looking for answers. Actually everyone is looking for answers to the same questions: Now that we have all this data, how can we make sense of it? What can the data tell us? What can we learn from it?

The amount of potential knowledge is astounding, but extracting this knowledge from the data is far from easy. The luxury of being able to measure almost everything poses interesting problems. For instance, interpreting data becomes increasingly difficult, the more data there is, as illustrated by the following quotation attributed to Lee Segall: 'A man with one watch knows what time it is, a man with two watches is never quite sure'.

A primary goal when observing complex systems is to describe the system in function of the observed factors. To ensure that this will be possible, one will typically measure too many factors rather than too few. The main problem is no longer what to measure but rather how to interpret many measurements, knowing that some factors might not be contributing to system understanding. This approach can yield vast data sets, which are seldom readily interpretable.

As the adage 'A picture is worth a thousand words' suggests, interpreting data is largely a matter of representation. In other words the aim is to first find a suitable representation of the data, and interpret that representation instead of the raw data. Creating such representation is one of the primary motivations for modeling. A model describes a transformation of the input parameters corresponding to observed factors to a response corresponding to system behavior. If the response of a model is truthful to the system under study, conclusions based on that model are valid for the system as well. We argue that it is the model that explains the data by making meaningful interpretations possible.

Most modeling techniques assume that all input parameters contain information, and include all input parameters in the resulting model. This assumption is not always realistic in an experimental setting where the observed process might not be well-understood. In order to apply these modeling techniques in such a setting, one must first perform feature selection.

Feature selection techniques extract a subset of the original variables, discarding variables that hold no useful information. For example multiple models could be build, using different subsets of input parameters until a satisfactory model is found. This relies on the fact that

a model of sufficient quality can not be build if an important variable is missing, and the model quality will deteriorate quickly if meaningless features are added. Remark how the usefulness of a variable is directly related to the modeling technique. In theory this approach is applicable to all modeling techniques. However, selecting the subsets to consider is a non-trivial practical issue. If no assumptions can be made, even the size of the optimal subset is unknown, leading to many possible variable combinations. Since considering a subset results in rebuilding the model, this incurs a serious computational investment. It is therefore of interest to strategically choose which subsets to consider. Two traditional approaches are *forward selection* and *backward elimination*. In forward selection the size of the subset is iteratively increased by one. Variables are added such that the subset produces the best model with dimensionality equal to the subset size, until no further improvement is observed. Backward elimination works the other way around, starting from the full variable set and iteratively removing the variable with the least impact on the response until the model quality is considered insufficient.

A related class of techniques are known as *dimensionality reduction* techniques. Here the goal is to find a lower dimensional representation of the original data. While this is different from feature selection, the two terms are sometimes used interchangeably in the literature. In this thesis we use the terms as described in this section to avoid confusion.

Many modeling techniques were designed to represent data compactly and deliver accurate models. Since we are interested in understanding the process under observation, it remains paramount to gain insight in the contributions of parameters, as to identify the driving factors of the process. We argue that even the most accurate model does not directly improve problem understanding if it can not provide an interpretable measure of importance for each of its variables. The uninterpretable black box that is the process would merely be replaced by an equally uninterpretable model.

Determining when a variable can be considered important in context of a given model is the core topic in this thesis. We will treat a variable as important if its presence or absence in the model matters, not unlike the feature selection techniques. However, some subtleties must be addressed, specifically inter-dependent variables. These are variables which only hold information if combined with other variables. In other words, such variables by themselves are unable to provide any predictive power, it is only by a combination they can provide insight. A traditional example of such inter-dependency is the XOR[1] problem, also discussed in [32]. For that problem a prediction based on any one input variable will not perform better than randomly selecting an output value. For feature selection algorithms such as forward selection, inter-dependent variables can be problematic. Using forward selection, any one variable of an inter-dependent set of variables will never be added to the subset, since it does not directly improve prediction results.

The knowledge gained from variable importance is key to process optimization. If after a series of exploratory experiments the variables' influences are discovered, future efforts can be focused on the meaningful aspects of the process. Not only does this catalyze further progress, but it can very well result in significant monetary savings or earnings. For instance, once variables are known to be unrelated to the process behavior, it will no longer be necessary to measure them.

[1]The XOR problem is also known as the two-bit parity problem.

1.2 Regression Techniques

Regression techniques are designed to solve so called regression problems. Any regression problem consists of describing a mapping of several input variables to one or more output variables. This is an instance of *supervised learning*, where it is known which variable to predict. This contrasts with *unsupervised learning*, were techniques aim to discover novelties or patterns in the data without the prior separation of input and output. Input variables are also known in the literature as predictors, features or attributes, while output variables are called responses. We will use these terms interchangeably. The mapping constructed by a regression technique is formalized into a model, and this model is able to predict the response(s) using the input variables.

In this thesis we focus on regression techniques which we will later use to discuss and analyze variable importance in different settings. The covered material is not meant to be exhaustive and we refer to the literature for other techniques [97]. The techniques we used will be informally introduced in the remainder of this section. Future chapters will cover each technique separately in more detail.

1.2.1 Variable Types

Not all variables can be treated equally. Intuitively a variable assuming values from the set $\mathcal{S} = \{Kitten, Puppy, Goldfish\}$ will be handled entirely different than a variable assuming values in the set of real numbers $\mathbb{R}$. In this thesis we only consider continuous variables, and we refer to Appendix C for a discussion on other variable types.

1.2.2 Notation

To make further formalizations possible we will first introduce some notational conventions followed throughout the thesis. We will denote a functional relation as:

$$f(\mathbf{x}) = y, \quad \text{where } \mathbf{x} \in \mathbb{R}^d,\ y \in \mathbb{R} \tag{1.1}$$

The data set is then represented as an $n \times d$ input matrix X, and an $n \times 1$ output vector Y. Variable i will be denoted by x_i for $i = 1, \ldots, d$. The jth observation is denoted by x^j for $j = 1, \ldots, n$. A specific data value in X will be denoted by x_i^j such that i is the variable index, and j is the observation number. Analogous conventions will be used for Y.

Now we can formalize regression as follows: given a data set $\{X, Y\}$ with an unknown relationship f such that $f(X) = Y$, construct a predictor $\hat{f} : \mathbb{R}^d \mapsto \mathbb{R}$ such that:

$$\hat{f}(\mathbf{x}) = \hat{f}(x_1, \ldots, x_d) = \hat{y}, \quad \text{where } \widehat{y^j} \approx y^j \quad \text{for } j = 1, \ldots, n \tag{1.2}$$

In other words the model prediction should approximate the original relation. The quality of this approximation is determined by an *error* or *loss function* $L(y, \hat{y})$, such that L is large when the prediction is poor. The squared error is commonly used as a loss function: $L(y, \hat{y}) = \|y - \hat{y}\|_2$. The mean sum of square errors on the data set estimates the expected value of this loss function:

$$MSE = \frac{1}{n} \sum_{j=1}^{n} (y^j - \widehat{y^j})^2 \tag{1.3}$$

A regression model will minimize the expected error, given a loss function. Remark that there could be more than one output variable, in which case the output is denoted by the vector $\mathbf{y} = (y_m, \ldots, y_m)^T$, and Y will be an $n \times m$ matrix. In this thesis we only consider the case with one output variable.

Ensemble methods will use a set of models for prediction, which we will denote by $\mathcal{M} = \{M_1, \ldots, M_s\}$. The prediction of an ensemble method is then formalized as a combination of the predictions by individual models:

$$\hat{f}(\mathcal{M}, \mathbf{x}) = \mathcal{C}(M_i(\mathbf{x})), \quad \text{for } i = 1, \ldots, s \tag{1.4}$$

where $\mathcal{C}$ is a function combining individual predictions. Often $\mathcal{C}$ will be a weighted sum, such that we can write:

$$\hat{f}(\mathcal{M}, \mathbf{x}) = \sum_{i=1}^{s} w_i M_i(\mathbf{x}) \tag{1.5}$$

1.2.3 Relating Regression to Classification

While regression deals with output variables with an infinite number of possible values, classification treats problems where the domain of the output variable is a finite set. In a classification context the output variable is referred to as the class label. In other words, the classification problem consists of assigning labels to input points, or inferring the class, given a set of attributes. Regression is sometimes seen as an extension from classification, in the sense that it would be a classification problem with infinite output classes. While this seems plausible at first, assumptions made by classification techniques do not always generalize well to regression problems. Most often, classification focuses on discovering decision boundaries separating the output classes, instead of a more direct approximation of the response such as found in regression.

1.2.4 Relating Regression to Optimization

While regression techniques construct models to approximate the underlying relation between input and output variables, optimization techniques find minima or maxima of arbitrary functions. We already stated that a regression model minimizes the expected error, such that $\hat{y^i} \approx y^i$. This allows regression problems to be reformulated as optimization problems given an error function. We note that apart from the MSE, the sum of the squared errors (SSE), and the root mean square error (RMSE) can also be minimization targets:

$$SSE = \sum_{i=1}^{n} (y^i - \hat{f}(x^i))^2 \qquad MSE = \frac{1}{n} SSE \qquad RMSE = \sqrt{MSE} \tag{1.6}$$

Most regression techniques define a class of models. Constructing an optimal model is then equivalent to minimizing one or more error functions over all models within this class. Remark that not all modeling techniques use the same error function, although the RMSE is widely used. This concept of optimization is applied once the model structure is known, and a set of model parameters is to be optimized.

Remark that the minimization of the prediction error can be a non-convex optimization problem, such that convergence to a global optimum is not guaranteed. In particular the search space may contain local optima, which some optimization techniques identify as the

solution, yielding suboptimal results. Navigating such search spaces is a challenge that calls for sophisticated optimization techniques. Issues prevalent in optimization are also of interest for regression. The specific issue of local minima will be revisited in Section 1.2.12.8.

1.2.5 Ensemble methods

While many modeling techniques provide a single model of high accuracy, another approach is to base predictions on a model ensemble, a collection of multiple models. Predictions of individual models are combined in a meaningful way to provide an accuracy expected to surpass any single model in the ensemble. For example in classification problems it is common to build several models, and let each model then 'vote' for a predicted class label. The class label which receives the most votes will then be accepted as the final solution. The advantage of combining multiple models is that not every model must be equally accurate in the whole input range, as long as the majority of models is of sufficient quality. Since creating models of lower quality is typically less computationally intensive than creating high quality models, this approach could potentially save computational resources. Remark that individual models in the ensemble are not required to approximate the whole input space with high accuracy. However, for this approach to work, individual models should at least be accurate in a region of the input space.

1.2.6 Overview Table

	Linear Regression	Neural Networks	Random Forests	Symbolic Regression
Knowledge about explicit model structure required	Yes	No	No	No
Parametric or Non-parametric	Param.	Param.	Non-param.	Non-param.
Possibility for Local Adaptation	No	No[(1)]	Yes	No
Model complexity depends on the # of data samples	No	No[(1)]	Yes	No
Potential to create compact models irrespectively of data size and structure	High[(2)]	Limited	Limited	High
Can final models provide insight into the problem, and increase system understanding	Yes	Hardly	Hardly	Yes
Complexity control possible	Yes	Yes	Yes	Yes
Danger to over-fit the data without explicit complexity control?	No	High	Limited	No
Danger of having insignificant variables in final models	Present	Present	Present	Not Present, or Heavily Reduced
Danger of NOT having significant variables in final models	Not Present	Present	Present	Present

Table 1.1: Overview of the properties of different modeling methods, as described in [116].
[(1)] Yes, for radial basis function neural nets.
[(2)] If the model structure is correct.

1.2.7 Linear Regression

A linear least square model predicts the output variable by considering a linear combination of the input variables, such that the SSE with respect to the original data is minimal. In other words fitting such a model is equivalent to the optimization of parameters w_i for $i = 0, \ldots, d$ using the model structure:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0$$

This can also be formulated as the solution to a linear set of equations, by introducing the so-called design matrix A:

$$A = (\mathbf{1}\ X), \quad \text{where } \mathbf{1} = (1, 1, \ldots, 1)^T$$

The set of linear equations, also called normal equations, is then given by:

$$A^T A\mathbf{w} = A^T y, \quad \text{where } \mathbf{w} = (w_0, w_1, \ldots, w_d)^T$$

Solving for $\mathbf{w}$ yields the coefficients of the linear model. Remark that this approach can be generalized further by using basis functions. For example it could also be used for fitting polynomial models.

1.2.8 Neural Networks

Neural networks are inspired by the operation of the human brain. The behavior of biological neurons is mimicked by simple artificial representations. Multiple neurons are interconnected and organized in a directional network, as shown in Figure 1.1. Each neuron performs an operation on a weighted combination of its inputs, this operation is known as the activation function. An activation function produces an output value, such that propagating input values through the network yields a prediction for the output variables. This transformation is defined by the combination of the network topology, connection weights and activation function, ie. the functional relationship between input and output is modeled by the network. This implies that an explicit analytical form for the model might not be readily available. Typically, the topology and activation function are chosen depending on model requirements such as non-linearity. Optimal connection weights are then determined such that the model prediction error is minimized.

We refer to Appendix B for a summary of our background research on neural networks.

1.2.9 Classification and Regression Trees

Classification and regression trees (CART) are closely related to decision trees. While a decision tree represents a series of questions branching for each meaningful answer, a CART inspects a property of the data and branches for each possible outcome. Without loss of generality we only discuss trees involving binary decisions. This restricts answers to questions in decision trees or CARTS to either 'yes' or 'no', as shown in Figure 1.2. While decision trees are intuitively interpretable, this property is less evident for a CART. In order to construct a good decision tree it is essential to ask the right questions. Analogously the construction of an optimal CART, with highest predictive accuracy, consists of finding optimal data properties. For more information on how to construct a CART we refer to Section 3.2.1.

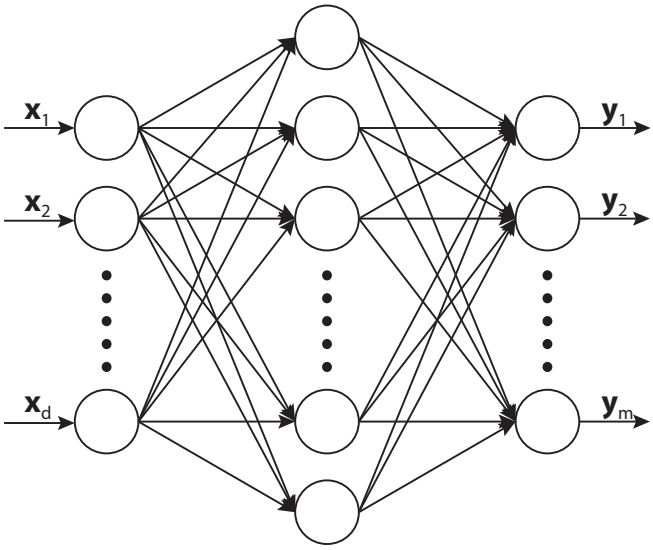


Figure 1.1: An example of a neural network with one hidden layer. Each arrow indicates a connection, with an associated weight. At every internal node a function is performed on the combination of incoming connections, producing one output value. Remark that the network does not need to be fully connected as depicted here.

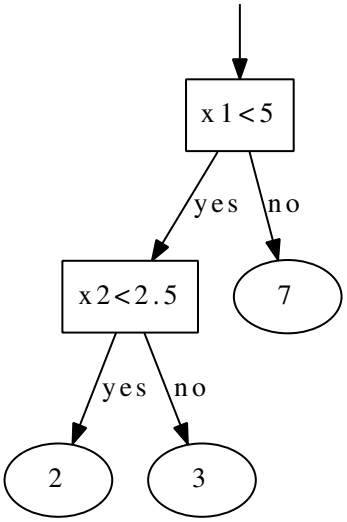


Figure 1.2: An example of a very simple CART. A prediction value is present in the leaf nodes, the prediction path through the tree is determined by data characteristics tested in each node.

1.2.10 Random Forest

From the simple regression tree shown in Figure 1.2 it is intuitively clear that a single tree does not approximate smooth functions well. Random forest (RF) improves the approximation of smooth functions by combining multiple regression trees in an ensemble. However, the construction of an optimal regression tree is deterministic, such that there would only be one tree for any given data set. However, suboptimal trees are not unique and individuals in an ensemble are not required to be optimal. By introducing a random component RF will create an ensemble of several suboptimal trees. This will enable the approximation of smooth functions, while remaining computationally feasible. In addition this technique can be used for both regression and classification tasks, and provides several variable importance measures.

We study this technique extensively in Chapter 3.

1.2.11 Symbolic Regression

Symbolic regression (SR) is a *genetic programming* technique where no assumption is made about model structure, and all models are explicit algebraic expressions. While the number of possible algebraic expressions for a given set of variables is infinite, SR aims to find optimal expressions. Algebraic expressions can be represented as a parse tree, as shown in Figure 1.3. In SR, a population of models is created, where each model is represented as a parse tree. This population evolves by strategically manipulating the parse trees, such that the model quality gradually improves. This involves the trade-off between model accuracy and model

complexity. Multiple models within a population can be optimal with respect to this trade-off, and a final solution consists of such optimal models. One can then either combine all optimal models, or a subset, in an ensemble.

We study this technique extensively in Chapter 4.

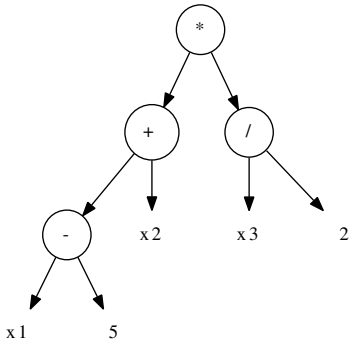


Figure 1.3: An example of a parse tree for the expression $(\mathbf{x}_1 - 5 + \mathbf{x}_2)\frac{\mathbf{x}_3}{2}$. Symbolic regression will construct a population of parse trees, which gradually evolve to optimal algebraic expressions expressing the functional input-output relationship of the data.

1.2.12 Common Issues in Regression

1.2.12.1 Model Expressiveness

This issue is inherent to any modeling approach, and not specific to any technique in particular. Since a model represents an unknown underlying relation between input and output variables, such a relationship is expected to exist in the first place. In other words the *'Garbage in, garbage out'* principle is certainly applicable in regression. While it would be interesting if the next winning lottery numbers could be predicted from ones shoe size, this does not make sense. For this trivial example it is obvious that the shoe size will not contain any valuable information, but this intuitive approach does not scale well.

The actual issue is that in order to construct a good model, the contributing factors should be known. In other words the relevance of variables is expected to be known in advance. This assumption does not hold in an experimental setting since the behavior of the system is *a priori* unknown. Even when a good model is available, understanding the variable importances of all inputs is of great interest since it could simplify future models and observations. In addition, variable importances are instrumental to gain insight into both the model and the process under study.

1.2.12.2 Model Structure

Many modeling techniques rely on the relationship between regression and optimization to construct the model. This readily implies that the model structure should be known in advance, otherwise it is not possible to formulate an explicit optimization problem in function of the model parameters. In reality the optimal model structure is often unknown, even with extensive domain knowledge. Also note the conflict of interest with respect to model structure complexity: It is desirable that a model is simple, so interpretation is straightforward, while

it should remain complex enough to capture the behavior under study. In other words, *the model should be as simple as possible but not simpler.*

1.2.12.3 Curse of Dimensionality

To see the impact of dimensionality on problem understanding, consider the input variables in a d-dimensional space. Now suppose we want this space to be sampled on a lattice with a certain density, ie. the number of points per unit volume. Let us start by sampling the unit interval with 10 equidistantly spaced points, and generalize to higher dimensionality. If we want to sample the two dimensional space with the same density, it would require 10^2 points. Analogously a three dimensional space, where sampling with the same density would require 10^3 points. Remark that the increase in required data is exponential with respect to the dimensionality, see also Figure 1.4 for a visual representation. The amount of data available can thus be counterintuitive, for example a data set of ten variables will still be a sparsely sampled input space when 10^5 records are available.

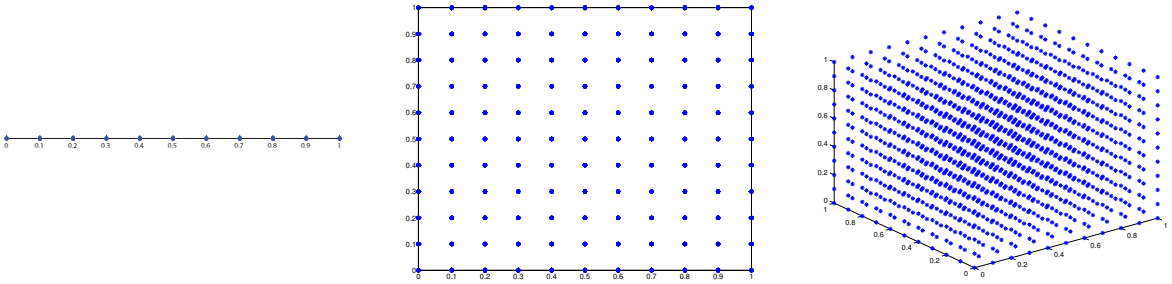


Figure 1.4: The unit interval, square and cube sampled with equal densities. The number of points needed to accomplish this increases exponentially.

Traditional metrics such as the Euclidean distance also behave counter-intuitively in high dimensional space [3]. In addition, the volume of the 'center' of the space decreases dramatically. This can be observed by generalizing the proportion of the volume of a circle inscribed in a square to the volume of that square as shown in Figure 1.5. In a d-dimensional space we compare the volume of a hypersphere with radius r to that of a hypercube with sides of length $2r$. The volume of the hypersphere is given by:

$$V_{\text{hypersphere}} = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}, \quad \text{where } \Gamma \text{ is the Gamma function.} \tag{1.7}$$

And the volume of the hypercube is given by:

$$V_{\text{hypercube}} = (2r)^d \tag{1.8}$$

It then becomes apparent that the hypersphere becomes an insignificant volume relative to the hypercube:

$$\lim_{d\to\infty} \frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} = 0 \tag{1.9}$$

In other words, any random point in a high dimensional space is more likely to be close to a boundary of that space than in the center of that space.

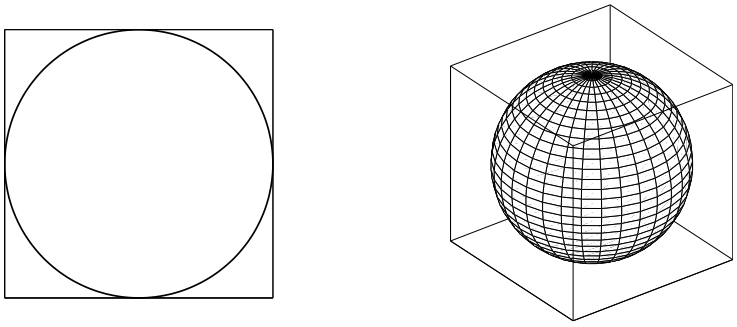


Figure 1.5: Visualization of the 'center' of a space in two and three dimensions, the circle and sphere respectively. The ratio of volumes of this 'center' compared to the full space will be of interest and is shown in Equation 1.9 to converge to zero with increasing dimensionality.

1.2.12.4 Overfitting the Training Data

Overfitting is the phenomenon occurring when a model has a low prediction error on data it has been trained with, but fails to generalize predictions to unseen input values. An extreme case would be a black box model that would explicitly store the data set, producing the known output if the input values are in the data set and an arbitrary value otherwise. Overfitting can be caused by several issues, for example an overly complex model structure.

1.2.12.5 Imbalanced Data Sets

Closely related to the curse of dimensionality is the issue of imbalanced data, when some areas of the input space are overrepresented such as shown in Figure 1.6. Since common error functions such as the SSE will sum the errors over all points, small errors in an oversampled area will be inflated due to large number of points in that area. In practice the resulting models will overfit in the oversampled areas, and accuracy will deteriorate in other areas. Remark that the problem is more complicated than sketched here, since it is, for example, desirable to sample proportional to the change in response in an area. A recent summary of available techniques in classification is given in [37, 16, 20], as well as a clear discussion on the problem. In [116] this topic is covered extensively in a regression setting.

A related and important research question is how to choose the sample points prior to the actual measurements. Although this topic is not elaborated further in this thesis we want to refer the interested reader to *design of experiments* [95] and *space-filling* designs [113].

1.2.12.6 Correlated Inputs

Formally the linear correlation between two random variables x_1 and x_2 is defined as:

$$\rho_{x_1,x_2} = \frac{E[(x_1 - \mu_1)(x_2 - \mu_2)]}{\sigma_1 \sigma_2}, \quad \text{with } \mu_i \text{ the mean and } \sigma_i \text{ standard deviation}$$

Conceptually correlated input variables occur when information is replicated across several variables, which is not limited to linear correlation. The problem is that the extra input dimensions they create are meaningless. For example if one variable is a multiple of another

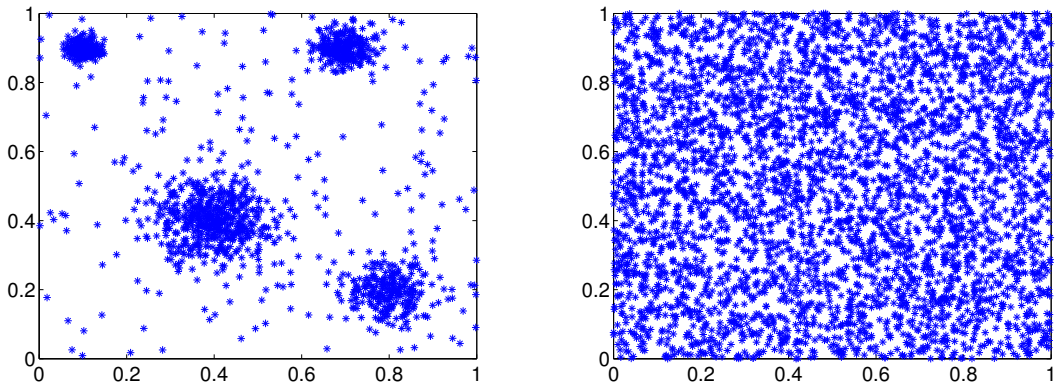


Figure 1.6: Both images contain the same number of data points. On the left an example of an unbalanced data set, on the right an example of a random uniform sampling. Remark that in some cases the data set on the left is preferred if the response behaves wildly in the oversampled areas.

variable, the same information could be represented by a single variable. Such meaningless dimensions increase the curse of dimensionality effect, which is to be avoided. In addition modeling techniques relying on optimization are affected as well. The optimization is considerably more difficult the higher the dimensionality, indicating that modeling a data set with highly correlated variables may produce poor models.

1.2.12.7 Missing Data Values

While already noted in Section 1.2.12.1 an underlying relation can only be adequately modeled if all contributing variables are present in the data set. Yet even if all variables are present, not all values might be available for all data records. For example suppose we are interested in five variables but are restricted to measuring only four variables at any given time due to external requirements. All data collected during such experiments will have an unknown value in one dimension. Unfortunately there is no universal solution to handle missing values. A statistical analysis of this problem is given in [71].

1.2.12.8 Local Optima in the Error Surface

In Section 1.2.4 we mentioned that regression problems can be reformulated as optimization problems, and local optima can be problematic when the solution to an optimization problem as found by an optimization technique is not a global optimum. Such solution is only optimal in a neighborhood surrounding it. The presence of local optima is dependent on the function to optimize, and the impact they have on the final solution depends on the optimization technique. To illustrate how this can be a problem we briefly explain the *steepest descent* method.

With steepest descent a function is optimized iteratively by following a trajectory in the input space, such that every step is advancing towards the optimum. Starting at a point input space, the gradient in that point is calculated. The next point to consider lies in the direction of the negative gradient, thus descending as fast as possible. It is not hard to imagine

a function where the steepest descent approach will be stuck in a local optimum, as shown in Figure 1.7. This illustrates that the choice of optimization technique will be instrumental in finding good models.

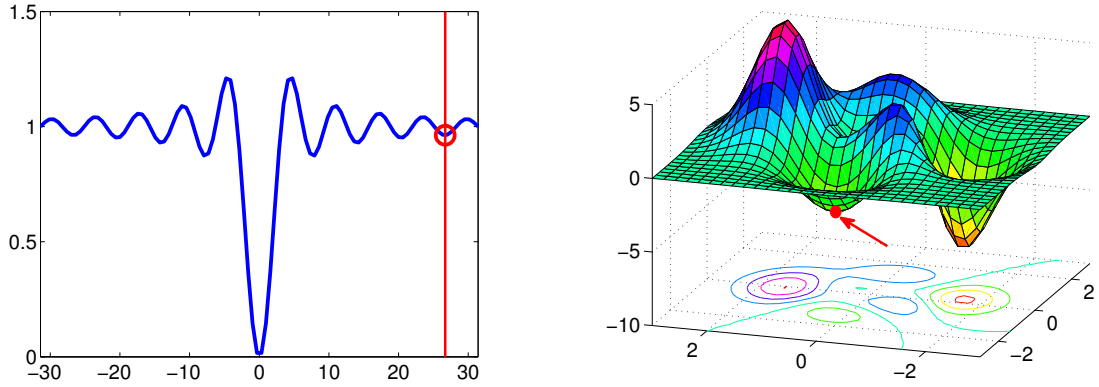


Figure 1.7: Two examples of a local minima where steepest descent would be stuck. In a local neighborhood this minima will be optimal, but the global optima is some distance away.

1.3 Goal of the Thesis

Our primary goal is to gain understanding about variable importance, guided by comparative experimental results. To that end we provide an overview with a selection of available methods. In particular we discuss how these methods differ conceptually and why it matters. We believe this to be fundamental for interpreting both models and variable importances obtained with these methods. While a formal discussion is beyond the scope of this thesis, we refer to the literature where applicable. So while not all aspects are discussed here, we provide useful references for the interested reader. We focus on two methods in particular: random forests and symbolic regression.

We can not stress enough that insight in a modeling technique and the resulting model(s) is paramount to understanding variable importances for the resulting model(s). To that end we present an extensive analysis of RF since this technique is widely used in many research fields for obtaining variable importance.

In summary we aim to raise awareness in important issues related to modeling and variable importance illustrated by experimental results, and inspire future researchers to continue the quest for knowledge.

1.4 Guide to the Thesis

We found that enumerating the chapters in sequential order is not optimal for representing the structure of this thesis. We reveal interconnections between chapters on a mind map, shown in Figure 1.8.

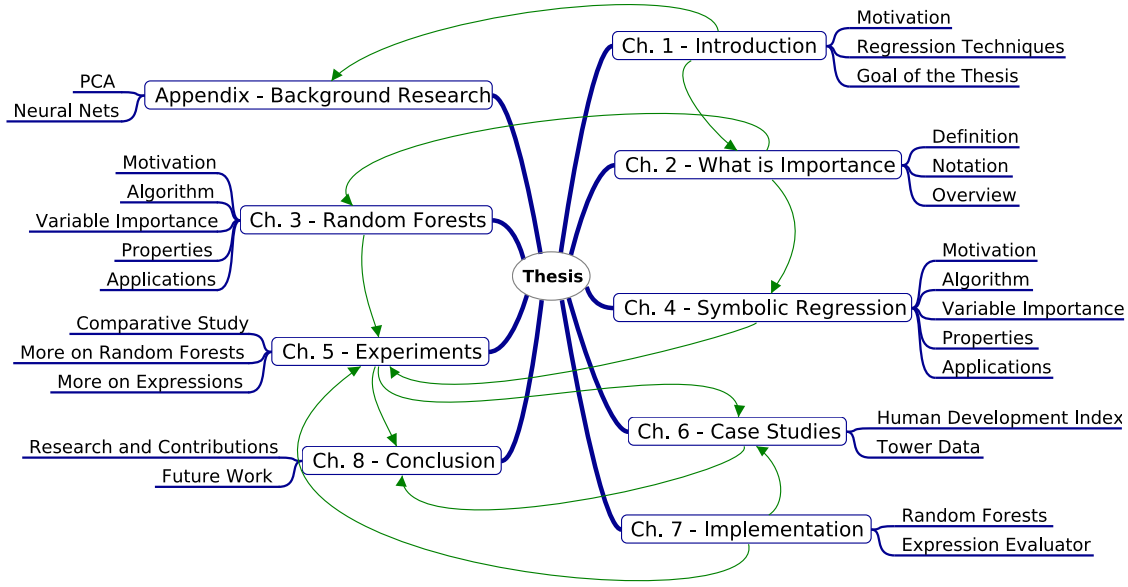


Figure 1.8: A mind map depicting the interconnections between different chapters of this thesis.

In the next chapter, the thesis addresses a core question: 'What is importance?'. We learn that importance in regression is only meaningful in context of a sufficient model. This is the motivation to examine two modeling methods in detail.

Both methods greatly differ in their approach and produced models. Random forests combines several classification and regression trees, while symbolic regression evolves a population of algebraic expressions. Random forests is analyzed in Chapter 3, and symbolic regression is studied in Chapter 4.

We also researched commonly used methods in data analysis, principal component analysis and neural networks. They are summarized in Appendix A and B respectively.

Several experiments that increased our understanding of the variable importance properties of random forests and symbolic regression are discussed in Chapter 5. Using the insight gained from these experiments we proceed to determine variable importances for two real-life problems in Chapter 6.

In order to perform both the experiments and case studies, we developed software components. We describe this implementation in Chapter 7.

This thesis concludes in Chapter 8 were the research and contributions are summarized, and future work is outlined.

Chapter 2

What is Importance

Using the term variable importance (VI) is not sufficient to express the different aspects of importance. This chapter explores different points of view, and provides context for later discussions.

Feature selection (FS) aims to extract a subset of features. Feature selection could be viewed as quantification of importance by assigning a score of either zero or one to variables, indicating whether to select them or not. But when we talk about VI we mean a quantification of importance such that scores of different variables are mutually comparable. Once importances are determined they could be used to perform FS, but it is assumed the quantification will be more informative than the subset presented by a FS technique. Even though the approaches are different, results should at least be consistent. In other words variables selected by FS should receive a high VI score, and conversely variables with highest VI score should be selected by FS.

We distinguish between methods computing either an *absolute* or *relative* importance. We use the term relative importance if the value is only interpretable in combination with importance values of other variables. In contrast, an absolute importance would also be interpretable in its own right, independently of the values for other variables.

In general variable selection algorithms come in three varieties: *filter*, *wrapper* and *embedded* methods [32, 55]. A filter method relies only on the characteristics or structure of the data and will determine importances without explicitly building any models. Wrapper methods are applicable when a sufficient model is already found and will estimate variable importance based on model evaluation. An embedded method determines variable importance during the model building process, typically resulting in a form of automatic feature selection guiding the modeling process.

We argue that filter methods can be used as preprocessing to remove some variables completely unrelated to the response. Although they will provide no further insight in the remaining variables since a meaningful interpretation of their contribution requires a model. While wrapper methods do use models, they handle it as if it were a black box simulation. Implying they are by definition constrained by the model they operate on, reflecting the limitations of the modeling engine. Embedded methods are beneficial since some VI information is used during model construction. Remark that this information will not be complete during model construction, but can serve as an indicator for the modeling technique. In addition embedded methods determine VI during model building. This removes the need to remodel excessively when the goal is to select a suitable variable subset, in contrast with

the forward selection or backward elimination FS techniques.

In summary, we distinguish methods either determining absolute or relative importances, optionally building and rebuilding models, as shown in Figure 2.1.

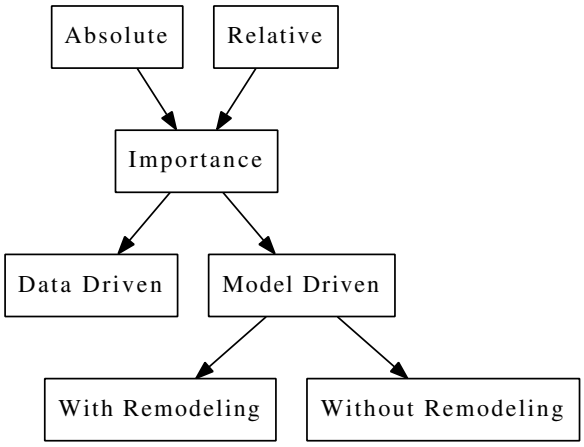


Figure 2.1: Different approaches of determining variable importances. Importance can be either absolute or relative, and is derived using either only the data or using models. If models are used, they can optionally be rebuild while determining the variable importances.

2.1 Definition

In our definition, something is important if its *presence or absence matters*. In other words the measure of importance is proportional to the change resulting from presence or absence. The importance of an input variable is determined by its power to both *explain and predict* the behavior of the system under study, i.e. the behavior of the response variable.

The importance of a variable can only be reliably interpreted given a functional relationship between the inputs and the response. A straightforward approach would be to determine the partial derivative of the model, and see how it varies across the input space:

$$\mathcal{I}_i(y, \mathbf{x}) = \frac{\partial \hat{f}}{\partial x_i}(\mathbf{x}) \tag{2.1}$$

An input variable is not important if it does not cause any change in the response, i.e. the partial derivative of the model over this variable (if it exists) is zero across the entire input space. Remark that the partial derivative is by definition a *local property* in case of nonlinear models. Consequently, variable importance should *also be treated as a local property*.

In practice, variable importance is estimated through *sensitivity analysis*. It is defined as the observed change in the response for when the value of that variable is varied while fixing the other variables to one of several configurations. The problem is that unless input-response models are given analytically as continuous differentiable functions, the sensitivity analysis must be performed by manually exploring the behavior of the response with little if at all quantification of importance. In this thesis we explore different approaches to sensitivity analysis in search for a method providing reliable importances.

Our aim is to build global models for a data set. Therefore, we are primarily interested in the variable importance over the whole input domain, given a model. In other words with

variable importance we want to express the global influence of a variable instead of the local influence. To that end local importances are accumulated:

$$\mathcal{I}_i(y) = \sum_{k=1}^{n} \left| \frac{\partial \hat{f}}{\partial x_i}(x^k) \right| \tag{2.2}$$

In addition we should confront another problem: are the variables present in optimal models that minimize the prediction error *per se* relevant to the problem? As it turns out not all such variables provide insight into problem understanding [55], and this distinction is highly dependent on the modeling technique. This can be an issue when using FS techniques, where the prediction error is the only available measure. The definition of importance we use will take the individual contribution to the response into account instead of only the total prediction error.

There is no universally acceptable method available to perform variable importance analysis. However, we identify several properties which would be present in a good VI method, and by which existing methods can be compared:

- **Interpretability**–Obtained importances should reflect the importances of the true input variables, without transformation.
- **Strictness**–Only variables relevant to describing the response should be allocated importances, so spurious variables should not appear important.
- **Conservativeness**–At intermediate stages of importance analysis all potentially interesting variables should receive importance.
- **Reproducibility**–A result can only be considered correct if it is reproducible.
- **Universality**–It is desirable for importances to be mutually comparable, and in addition to be problem independent.

2.2 Notation

More formally we will describe VI depending on which type of method was used to obtain it. In case of a filter methods:

$$\mathcal{I}_i = \mathcal{I}(\mathbf{x}_i \mid X, Y) \tag{2.3}$$

In case of a wrapper method:

$$\mathcal{I}_i = \mathcal{I}(\mathbf{x}_i \mid M, X, Y), \quad \text{where } M \text{ a model or model set } \mathcal{M} \tag{2.4}$$

Embedded methods can in general be described as:

$$\mathcal{I}_i = \mathcal{I}(\mathbf{x}_i \mid method, X, Y) \tag{2.5}$$

2.3 Overview

We compiled Table 2.1 to provide an overview of the properties discussed in Section 2.1 for different VI methods. Note that the table reflects conclusions from our experiments, detailed in Chapter 5.

	Interpretable	Strict	Conservative	Reproducible	Universal
Linear models	yes	yes	yes[(1)]	yes	yes
PCA	no	maybe[(2)]	maybe[(2)]	yes	no
Neural Networks	yes	maybe[(2)]	maybe[(2)]	yes	no
Random Forests	yes	no	no	yes	no
Symbolic Regression	yes	yes	yes	yes[(3)]	yes

Table 2.1: Overview of properties by different variable importance methods.
[(1)] If the model structure is correct.
[(2)] If the technique is applicable.
[(3)] Given enough time.

2.4 Relating Importance to Modeling

We already mentioned that the importance of variables can only be interpreted in function of a model. Perhaps surprisingly, variable importances are also valuable to the modeling process. Not only do they grant embedded methods guidelines for generating good models, but VI can also be instrumental in model verification. If extensive domain knowledge is available such that relevant variables are known in advance, even models achieving a low prediction error can be safely rejected if they do not incorporate these relevant variables. A situation where all relevant variables are known in advance is of course highly artificial and would remove the need to use VI methods. However, in practice it is not uncommon that a small subset of the variables is known to be relevant. Remark that if known relevant variables do not have a high VI score, this is either an indication that the modeling technique is not expressive enough, or the produces model is not sufficient, or that the expected relationship is simply not captured by the data.

Chapter 3

Random Forests

3.1 Motivation

The random forest (RF) technique was introduced by Breiman [13] as an ensemble of classification and regression trees (CART). We found RF to be an attractive method since it can handle large data sets of high dimensionality, and is applicable to either classification or regression problems. It is also one of the more accessible methods, in the sense that the algorithm and reasoning is rather intuitive. An accessible treatment is also provided in [99], for a formal treatment we refer to [9]. In addition RF is praised for its robustness, especially when dealing with data sets of very high dimensions such as gene expression data [5, 23]. This desirable quality is due to a VI scheme incorporated in the modeling technique, which we will examine as well.

3.2 Algorithm

3.2.1 Building a Tree

Once constructed, a CART represents a specific partitioning of the input space. Consider the root node to contain the full data set, and each child of the node to contain the partition of the data set satisfying the test imposed by that node. The construction of a tree starts at the root, where we have to determine a test suitable to partition the data in two groups. We only deal with binary trees, so we are restricted in this choice since the answer to this test should be either 'yes' or 'no'. Even with this restriction, tests such as $x_i = x_j$ or $x_i < x_j * x_k$ would be possible. Breiman also introduced RF techniques where the combination of variables is allowed [13], but we consider the version where a single variable is compared with a constant. In other words, the only splits allowed are of the form $x_i < c$ where c is a constant known as the decision value, and x_i is known as the decision variable. Even though this additional restriction is limiting, testing every variable x_i for $i = 1, \ldots, d$ remains infeasible when handling high dimensional data sets.

At every inner node j a candidate variable set $\mathcal{V}_j = \{x_k\}$ is selected randomly such that $k \in \{1, \ldots, d\}$ and $|\mathcal{V}_j| = K$ where K is typically much smaller than d. For each variable within this candidate set the optimal value for c is determined by consulting an information gain criterion $G(x_k, c, X, Y)$. The considered values for c are the midpoints of two sequential

values in the sorted values of variable k, formally c will be a member of set C:

$$C = \{\hat{x}_k^i + \frac{\hat{x}_k^{i+1} - \hat{x}_k^i}{2} \mid i \in \{1, \ldots, n-1\}\} \tag{3.1}$$

where $\hat{x}_k$ are the sorted values of x_k.

The split in that node will be determined by $x_k < c$ such that $G(x_k, c, X, Y)$ is maximal. The two child nodes will receive their share of the data partition according to the test result in the parent node, and will recursively partition it further.

This splitting process is applied until the partition is sufficiently small, or the information gain criterion signals that further splitting will not provide additional predictive power. This recursive partitioning can be seen as a special form of stepwise regression [8]. A node which is not split further is called a leaf node. The number of points remaining in a leaf is known as the leaf size, and is a parameter to the RF algorithm. Remark that a leaf can still contain more than leaf size elements in case the splitting criterion indicated splitting further is not useful. The impact of the leaf size will be inspected in Section 3.2.3.2. The construction of a tree is also shown in Algorithm 1.

Algorithm 1 ConstructTree(currentPartition)

```
if |currentPartition| > leafSize then
  V_j ← {x_k} such that k ∈ {1,...,d} and |V| = K
  for all x_k ∈ V do
    for all c ∈ C do
      G_{k,c} = G(x_k, c, X, Y)
  split ← max(G_{k,c})
  left, right ← Partition( CurrentPartition, split )
  ConstructTree(left)
  ConstructTree(right)
```

A crucial element in the tree construction as explained above is the information gain criterion G. It is by this measure that the node splits will be determined, thus affecting the whole tree structure. The criterion used in RF is computed as follows: Consider the partition P to split, and take the squared sum of the response variable divided by the size of P to be the baseline. For each split of a given variable x_i and constant c, compute the squared sum of each child divided by their size and add them. The best split is then defined as the split that differs most from the baseline, causing the newly created partitions to be as similar as possible in the response variable. More formally, let G be:

$$G(x_k, c, X_p, Y_p) = \frac{1}{|L|}\left(\sum_{i \in L} y^i\right)^2 + \frac{1}{|R|}\left(\sum_{i \in R} y^i\right)^2 - \frac{1}{|P|}\left(\sum_{i \in P} y^i\right)^2 \tag{3.2}$$

where $L = \{i \mid x_k^i < c\}$ and $R = \{i \mid x_k^i \geq c\}$. Remark that G is directly derived from the original Fortran implementation [11], also consistent with the latest version of the random forest package in R [69].

Now trees can be constructed, let us take a look at their predictions and how it is affected by the choices made at the splitting stage of the algorithm. The prediction of a tree is determined by starting in the root node of the tree, checking the decision variable and value.

The next node to consider is the node to which the input point would have been assigned during partitioning. This effectively creates a path from the root node to a leaf node. The prediction of a tree for any input point is the average of the training points in the leaf node reached by following such path. This prediction algorithm is also shown in Algorithm 2. From the restrictions on the splits in a node, we observe that the prediction of a single tree will be a multidimensional step function, as shown in Figure 3.1. Furthermore, each node will introduce a discontinuity in the dimension of the decision variable at the decision value. Also remark, that a prediction path is sequential, meaning that splits imposed at internal nodes apply to points satisfying all earlier splits. In other words, RF captures interactions between variables, because the value of one variable can influence future splits on other variables.

Interpretable visualizations of a tree prediction is only possible for problems of low dimensionality. Visualization of a tree prediction on a two dimensional problem is constructive for later discussions, and is provided in the examples in Section 3.2.4.

Algorithm 2 Prediction of a Tree

currentNode ← root
while (isNoLeafNode(currentNode)) **do**
  P ← currentNode$_{\text{partition}}$
  k ← currentNode$_{\text{decisionVariable}}$
  v ← currentNode$_{\text{decisionValue}}$
  low ← currentNode$_{\text{Child}}$, $\forall p \in P : p_k < v$
  high ← currentNode$_{\text{Child}}$, $\forall p \in P : p_k \geq v$
  if ($x_k < v$) **then**
    currentNode ← low
  else
    currentNode ← high
end while
return currentNode$_{\text{prediction}}$

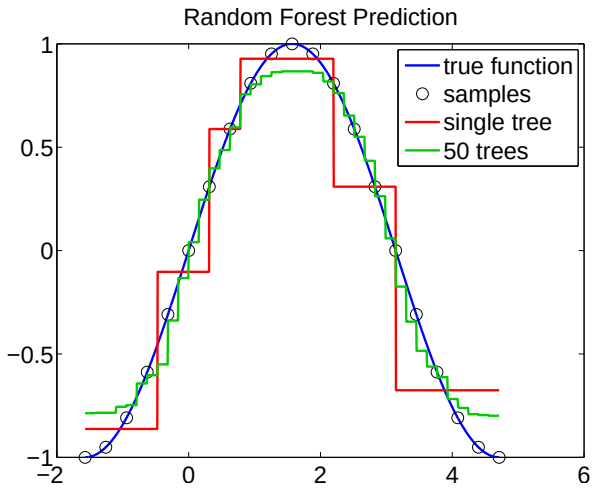


Figure 3.1: The smooth function is sampled at the dots. The coarse stepwise line represents the prediction of a single tree, while the finer stepwise line indicates the prediction of a forest of 50 trees.

3.2.2 Building a forest

Since RF is an ensemble method, the prediction of multiple trees will contribute to the model prediction. In Section 3.2.1 we observed that a single tree would be a poor approximation of a smooth function. By combining many step functions the approximation can be shown to converge to smooth functions [13].

In order to achieve this smoothness individual trees should be sufficiently diverse, implying their prediction should be different. This might be counterintuitive since each individual tree also models the data set, so why should their predictions be different? Consider what happens if very similar trees are combined: Since the decision values will be almost the same, the stepwise behavior of the prediction will be strongly encouraged. Consequently, smoothness would never be achieved. So the question becomes how to ensure tree diversity while not compromising model accuracy to a degree where individual trees become unusable. Breiman introduced a technique for this called bootstrap aggregating or bagging [12].

The idea behind bagging is elegant in its simplicity: Construct each model with a different subset of the data set. The input data is divided in an in-bag and out-out-bag set not unlike the training and test set used in neural networks. This division of the data set is different for each tree. The in-bag data is used to construct the tree, while the out-of-bag data is used to generate variable importances and estimate the accuracy of a tree. Important to note is that the in-bag data set is assumed to contain enough information to construct an adequate model. The individual trees will then be of reasonable quality, and the decision variables are not expected to be significantly different. The decision values in splits will be different though, preventing the steps in the predicted response to align. In that way combining predictions of multiple trees will create a smoother surface, as shown in Figure 3.1.

The algorithm for building a forest is summarized in Algorithm 3. The prediction of a random forest is the average prediction of all its trees, also shown in Algorithm 4.

Algorithm 3 Constructing a Forest

```
forestEnsemble ← {}
while ( |forestEnsemble| < # Trees) do
  inBag, outOfBag ← Bag( X, Y )
  singleTree ← ConstructTree( inBag )
  singleTree → forestEnsemble
```

Algorithm 4 Prediction of a Forest

```
totalContribution ← 0
for all singleTree ∈ forestEnsemble do
  Predict( singleTree, x ) / |forestEnsemble| → totalContribution
return totalContribution
```

3.2.3 Parameters

In this section the parameters of the RF method are presented. For a practical example where the impact of several of these parameters is shown, we refer to Section 3.2.4.

3.2.3.1 Random Seed

Any algorithm with a random component should be checked for the sensitivity to the random seed. In the RF algorithm there are multiple random factors, namely the division in in-bag and out-of-bag sets and the selection of variable candidate sets. When the other parameters allow the RF model to attain sufficient quality, the random seed does not influence the behavior of the forest significantly. Although the individual trees will differ, this effect is compensated by taking the average prediction of the ensemble.

3.2.3.2 Leaf Size

The leaf size will in essence determine the size of the steps in the predicted response of a tree. When leafs contain more data points the prediction values will be an average over more data points, decreasing the resolution of the predicted response. A larger forest will compensate for this loss in resolution though, and a larger leaf size can improve prediction accuracy in case of noisy data. The impact of this parameter will be shown on a practical example in Section 3.2.4.

Remark that a leaf size of one would cause a tree to overfit. In such a tree, every predicted response is a response value from the training set, not an average of multiple training responses. It is clear that a tree with leaf size one does not generalize well. A forest is in general robust against overfitting because the prediction is the average response of multiple trees.

3.2.3.3 Number of Trees

In Section 3.2.1 it was demonstrated that a single tree was not sufficient to provide adequate predictions. While more trees yield smoother results in general, it is of interest to know how many trees should be combined. Recall that the diversity within the ensemble is important as well for the predictive capabilities of the forest. This effect can be dramatic, such that better predictions can be obtained using a smaller highly diverse ensemble [27]. The impact of this parameter will be shown on a practical example in Section 3.2.4.

3.2.3.4 Candidate Subset Size

At every node in the tree the optimal candidate variable is selected for splitting. In order to provide a decent model, splits should occur on an important variable, since testing the data for an irrelevant attribute does not yield information. The information gain criterion will ensure this selection if important variables are present in the candidate set. However, consider the case when only a very small portion of the input variables are important. Then a small candidate subset size K results in a low probability of selecting an important decision variable. Yet the choice for K was also motivated by practical concerns since each candidate evaluation will use computational resources. The impact of this parameter will be shown on a practical example in Section 3.2.4.

3.2.4 Examples

3.2.4.1 Finding an Optimal Split in a CART

This example shows how to find the best split according to the Gini criterion, given a data set of input-response values. We use the following data set:

x_1	x_2	y
1	1	2
2	0	2
3	2	4
0	3	3
1	0	1

The first step in finding the best split is calculating the baseline, in other words the squared sum of the response variable divided by the size of the partition. For this data set this results in:

$$baseline = \frac{(2+2+4+3+1)^2}{5} = 28.8$$

To determine the optimal split, we check the information gain of all variables, as defined in Equation 3.2. We do this for all possible decision values, as defined by Equation 3.1. We start by sorting variable x_1, the resulting order is:

x_1	0	1	1	2	3
y	3	2	1	2	4

Now we determine the information gain for all values splitting the partition for x_1. We indicate the partition boundary by a double line.

x_1	0 ‖	1	1	2	3
y	3 ‖	2	1	2	4

$$gain = \frac{3^2}{1} + \frac{(2+1+2+4)^2}{4} - baseline = 0.45$$

Remark that it is impossible to split between two identical x_1 values.

x_1	0	1	1 ‖	2	3
y	3	2	1 ‖	2	4

$$gain = \frac{(3+2+1)^2}{3} + \frac{(2+4)^2}{2} - baseline = 1.2$$

x_1	0	1	1	2 ‖	3
y	3	2	1	2 ‖	4

$$gain = \frac{(3+2+1+2)^2}{4} + \frac{(4)^2}{1} - baseline = 3.2$$

We conclude that the best split for variable x_1 is between 2 and 3 since this results in the most information gain.

Analogously, we will consider variable x_2. We first sort the data set on x_2, resulting in the following order:

x_2	0	0	1	2	3
y	2	1	2	4	3

Since a split can only occur between two different values, the first partitioning that we consider is:

x_2	0	0 ‖	1	2	3
y	2	1 ‖	2	4	3

$$gain = \frac{(2+1)^2}{2} + \frac{(2+4+3)^2}{3} - baseline = 2.7$$

We continue checking the information gain for the other decision values:

x_2	0	0	1	2	3
y	2	1	2	4	3

$$gain = \frac{(2+1+2)^2}{3} + \frac{(4+3)^2}{2} - baseline = 4.04$$

x_2	0	0	1	2	3
y	2	1	2	4	3

$$gain = \frac{(2+1+2+4)^2}{4} + \frac{(3)^2}{1} - baseline = 0.45$$

We conclude that the best split for variable x_2 is between 1 and 2.

The best split is determined by the variable with the highest information gain. Variable x_2 scores higher with a score of 4.05, compared to variable x_1 with 3.2. This score for x_2 was obtained by partitioning in between values 1 and 2. Consequently, the best split for this example has x_2 as the decision variable, and 1.5 as the decision value.

3.2.4.2 Tree Structure and Prediction

To start with a basic 2d example, consider the function:

$$y = f(x_1, x_2) = x_2 \sin(x_1) \tag{3.3}$$

Let us generate a data set by sampling this function in 100 points placed on an equidistant grid, with ranges:

$$x_1 = 0 : \frac{\pi}{5} : 2\pi, \quad x_2 = 1 : 0.4 : 5 \tag{3.4}$$

The prediction of a forest of 20 trees is visualized in Figure 3.2. An explicit tree is shown in Figure 3.3, where the response to input $(5.5, 3.5)$ is predicted.

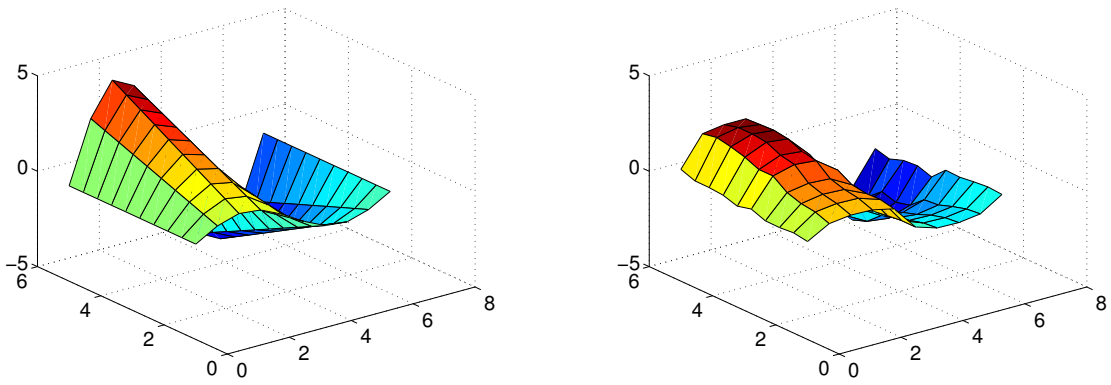


Figure 3.2: On the left the data as generated from Equation 3.3. To the right the prediction of a forest of 20 trees. Remark that the prediction of the forest approximates the original data but the response is less pronounced. This behavior is due to the averaging over input points taking place in a leaf. As for the other parameters the leaf size is 5, the in-bag data is 2/3 of the full data set sampled with replacement, the size of the candidate set was 1.

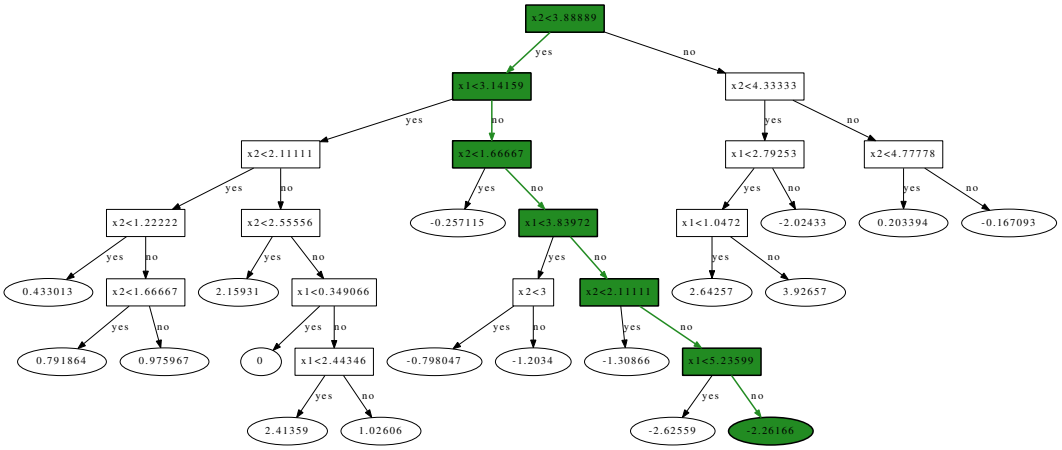


Figure 3.3: Predicting the response in a CART, part of the forest used in predicting the response as shown in Figure 3.2. Here the prediction path is shown in green for input $(5.5, 3.5)$. The resulting prediction is -2.26166 while the true value as obtained from the generating function given in Equation 3.3 is -2.47. While the error of a single tree can be rather large, this will be less of a problem in the forest prediction.

3.2.4.3 Partitioning the Input Space

To illustrate how a tree divides the input space in neighborhoods corresponding to the points in a leaf, we use the 2d-Salustowicz function as introduced in [94]:

$$f(x_1, x_2) = \left(e^{-x_1} x_1^3 \cos(x_1) \sin(x_1)(\cos(x_1) \sin^2(x_1) - 1)\right)(x_2 - 5) \tag{3.5}$$

The response of this function is also shown in Figure 3.4. This function is sampled on equidistant grids where the ranges of the variables are:

$$0.05 \leq x_1 \leq 10, \quad 0.05 \leq x_2 \leq 10.05 \tag{3.6}$$

such that a coarse grid consists of 10 distinct levels in each variable, and a fine grid consists of 100 levels. The division of the input space is shown on a colored height map of the 2d-Salustowicz function. The difference in prediction resulting from the resolution of the grid can be observed in Figures 3.5 and 3.6. Remark that for many modeling problems it is not optimal to sample on an equidistant grid, but we choose to do so in this example because the spatial division is very regular. The effect of not sampling on a grid is shown in Figure 3.4. The other parameters are fixed: a leaf size of 5, candidate set size of 2, no bagging (all data in-bag).

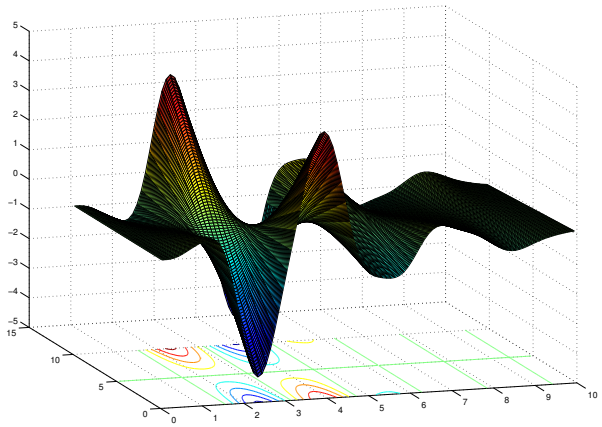


Figure 3.4: The 2d-Salustowicz function as described by Equation 3.5.

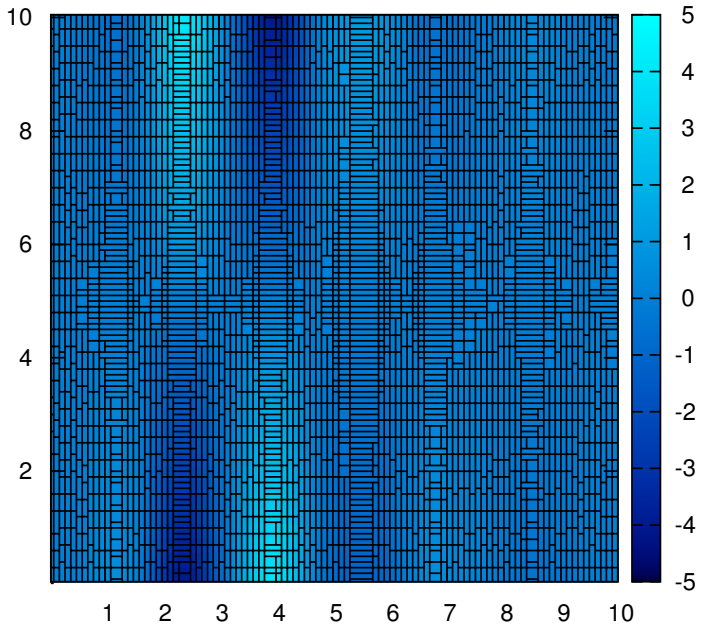


Figure 3.5: Spatial division by a tree for the 2d-Salustowicz function. The tree uses the full data set, two decision variable candidates per split, and the data is sampled on the fine input grid. Remark that the division is strongly influenced by the rate of change of the response.

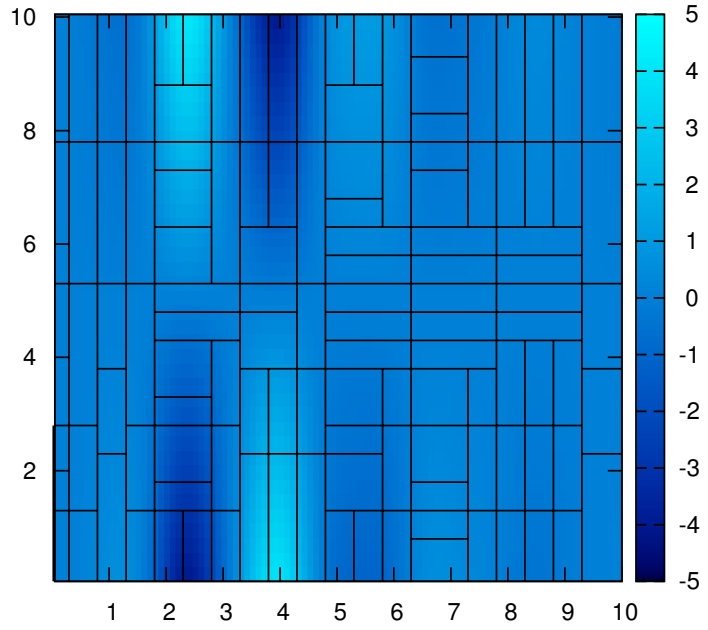


Figure 3.6: Spatial division of a tree for the 2d-Salustowicz function sampled on the coarse input grid. Remark that the prediction accuracy of the tree is significantly affected by the sparsity of the training data. This is indicated by the larger size of the regions, since all point in a region have the same predicted response.

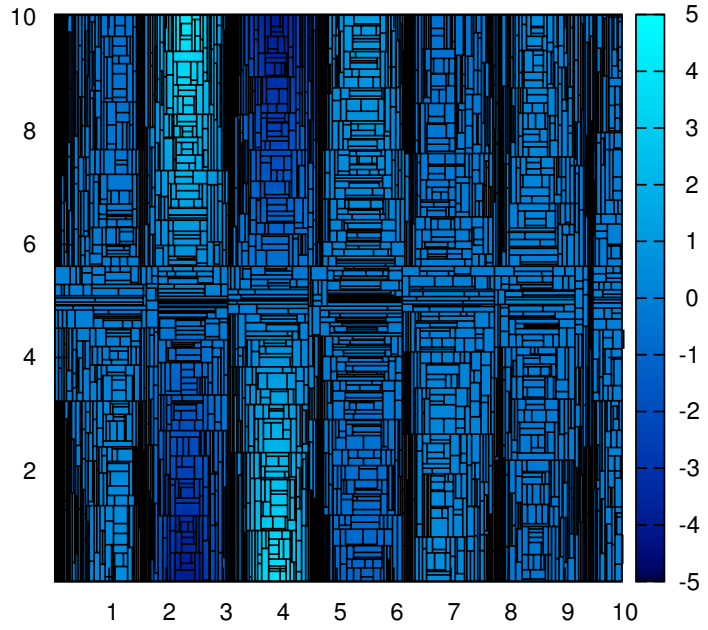


Figure 3.7: Spatial division of a tree for the 2d-Salustowicz function, where the training points are not on a grid but sampled randomly. Since splits do no longer 'cleanly' divide the input space, this allows for much more diversity.

3.2.4.4 Influence of the Random Forest Parameters

Visualizing the prediction is only possible in low dimensionality, so we opted to use an example with two variables. We use the response obtained from interpreting the grayscale values of the picture shown in Figure 3.8 as the response. We chose to use a picture because of the very non-linear response, and additionally the prediction defects will stand out and are more intuitive. The resolution of the picture was reduced to 150x150 pixels prior to modeling, to accelerate the modeling process. A preliminary test based on 1000 trees, 2/3 in-bag data, leaf size of 5, candidate set size of 1, as shown in Figure 3.8 indicates that RF is capable of dealing with this kind of response surface.

(a) Original (b) Reduced (c) Predicted

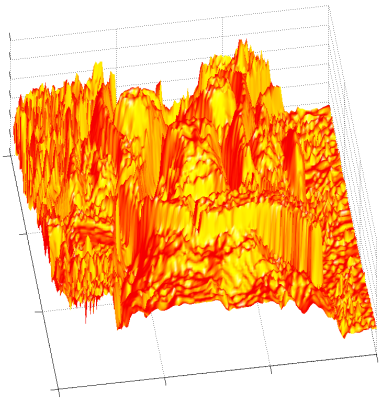
Figure 3.8: Feasibility test to see if RF is capable of dealing with this kind of response surface.

Before exploring the influence of different parameters, let us take a closer look at the response surface as shown in Figure 3.9. From both the height map and the contour plot it is apparent that the response behaves wildly. This is intuitively due to edges in the image, where the grayscale value changes dramatically fast in a direction, for example, the transition from the black collar to the white wall.

In the next sections we will investigate the influence of several parameters of the random forest algorithm. In each section one parameter will be varied while keeping the other parameters fixed. When a parameter is not specified, we used the value given in Table 3.1.

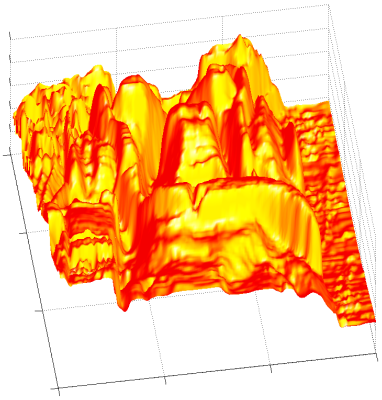
Parameter	**Value**
Number of trees	1000
In-Bag data	2/3
Leaf Size	5
Candidate set size	1

Table 3.1: The prediction examples from the next sections use these default parameter values.

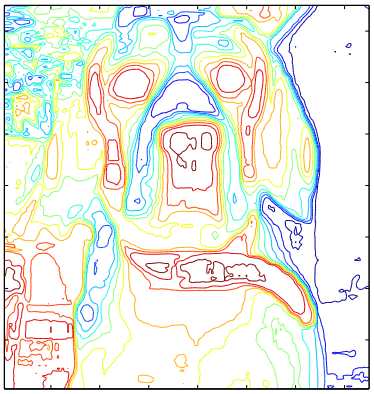


(a) Reduced - Height map

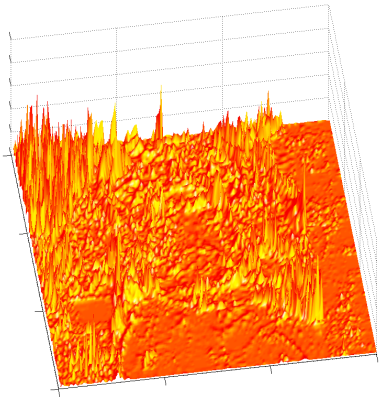
(b) Reduced - Contour plot



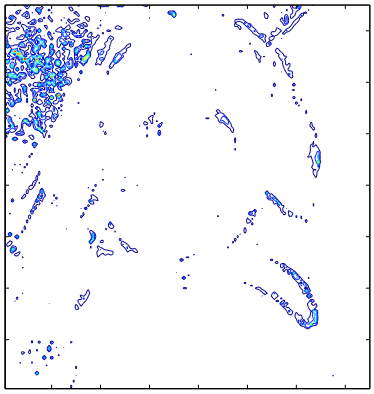
(c) Predicted - Height map



(d) Predicted - Contour plot



(e) Error - Height map

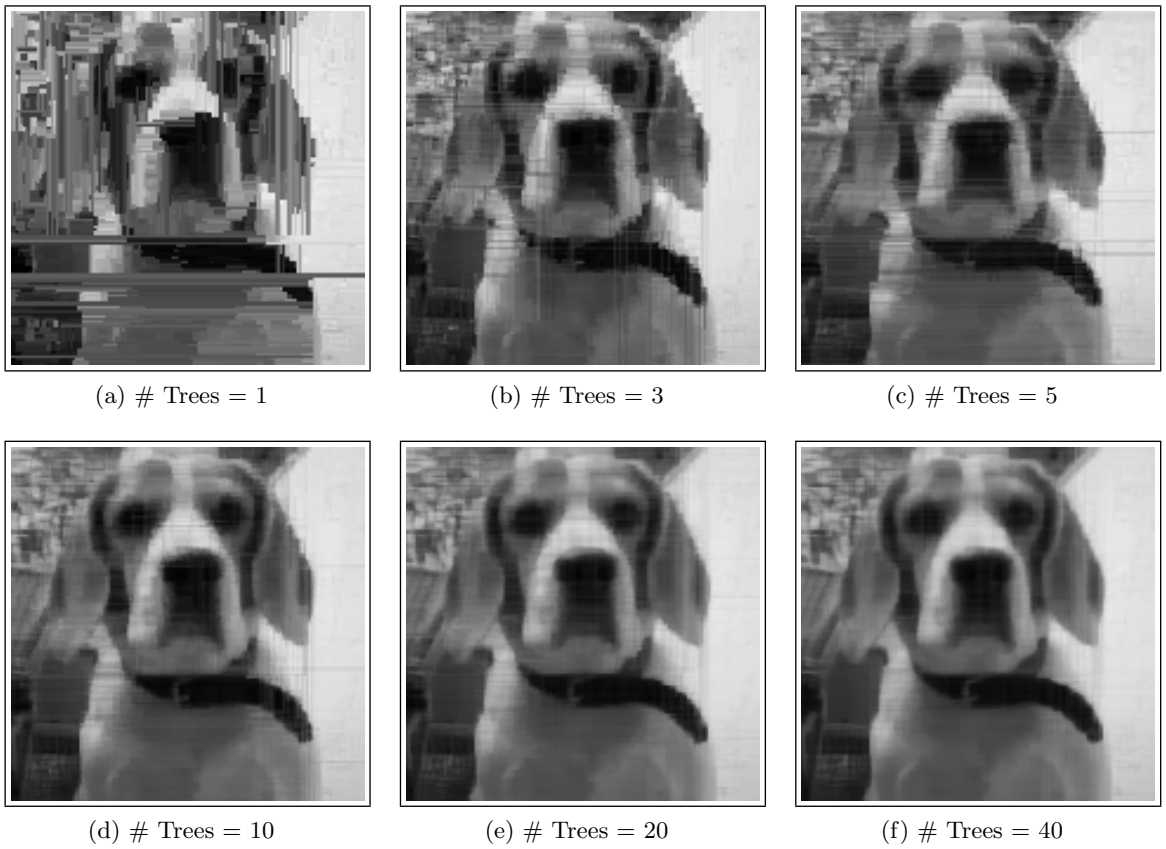


(f) Error - Contour plot

Figure 3.9: Height maps and contour plots corresponding to the original data, its prediction and error surface. Black corresponds to a value of one, while white is represented as zero. The random forest parameters for generating Figure 3.9c and 3.9d are given in Table 3.1.

3.2.4.4.1 Influence of the Forest Size

To see the impact on the prediction of a single tree and how a forest aggregates these predictions, the forest size is varied and the results shown in Figure 3.10. We observe that while a single tree does not provide a good prediction overall, the prediction is greatly improved by adding more trees to the forest.



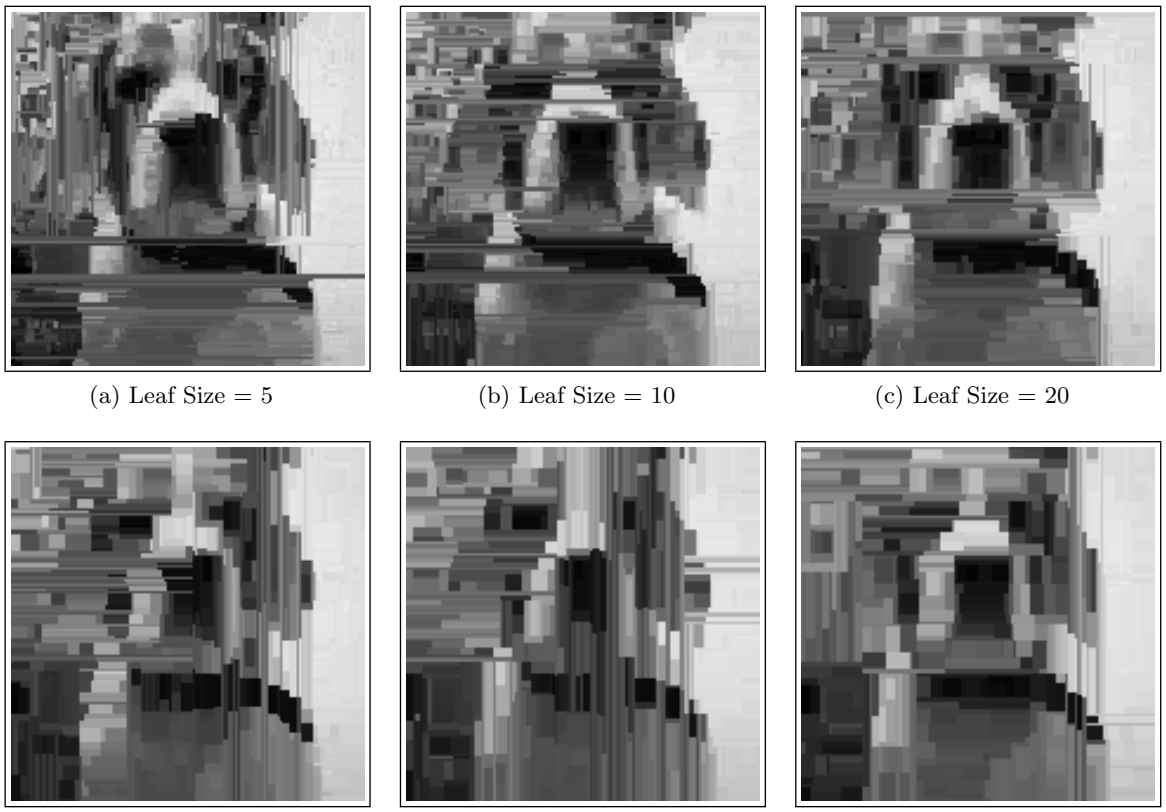
(a) # Trees = 1 (b) # Trees = 3 (c) # Trees = 5

(d) # Trees = 10 (e) # Trees = 20 (f) # Trees = 40

Figure 3.10: The influence of varying the forest size, with the other parameters fixed at the values given in Table 3.1. Remark that for this example after 20 trees, adding more trees does not improve the prediction much. In addition the increase in prediction accuracy from adding another tree diminishes with the size of the forest. In higher dimensionality this property is of particular interest.

3.2.4.4.2 Influence of the Leaf Size

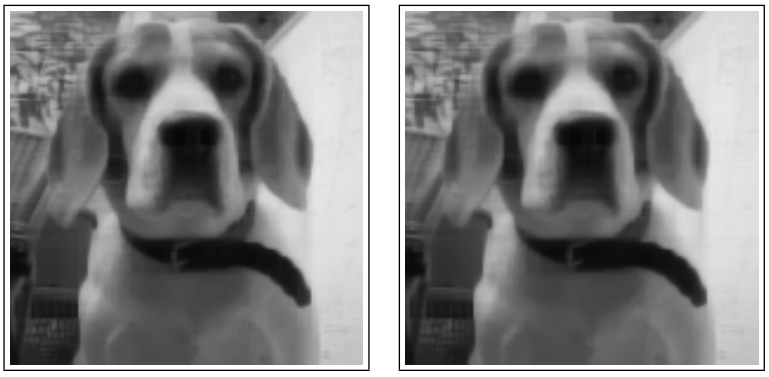
To see the impact on the prediction of a single tree, the leaf size is varied and the results shown in Figure 3.11. By increasing the leaf size the prediction is smoothed out more, in other words larger neighborhoods are assigned the same prediction value. How the forest aggregates this predictions is shown in Figure 3.12.



(a) Leaf Size = 5 (b) Leaf Size = 10 (c) Leaf Size = 20

(d) Leaf Size = 30 (e) Leaf Size = 40 (f) Leaf Size = 50

Figure 3.11: Illustrating the influence of the leaf size on the prediction of a single tree, with the other parameters fixed at the values given in Table 3.1. Remark that the size of rectangles with the same value increases with the leaf size.

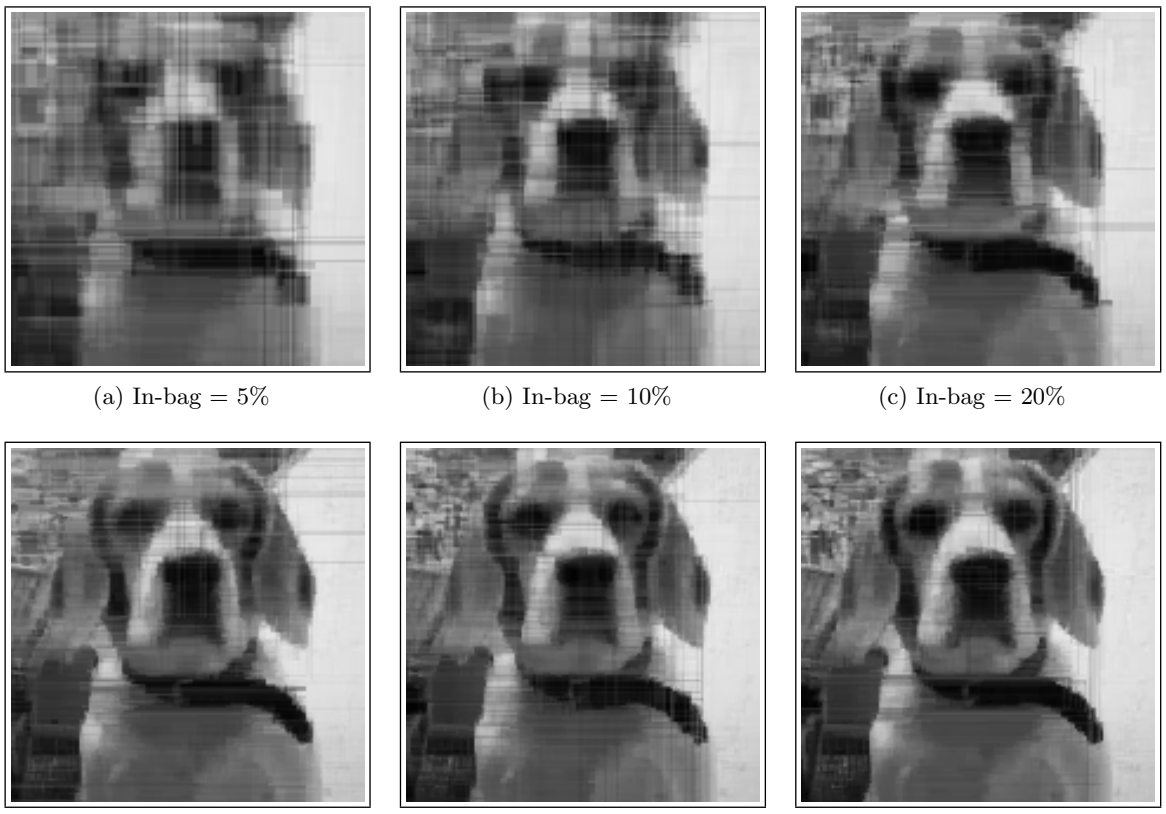


(a) Leaf Size = 1 (b) Leaf Size = 5

Figure 3.12: Prediction of a forest using the parameter values given in Table 3.1. Remark that having one candidate is the same as choosing a random decision variable.

3.2.4.4.3 Influence of Bagging

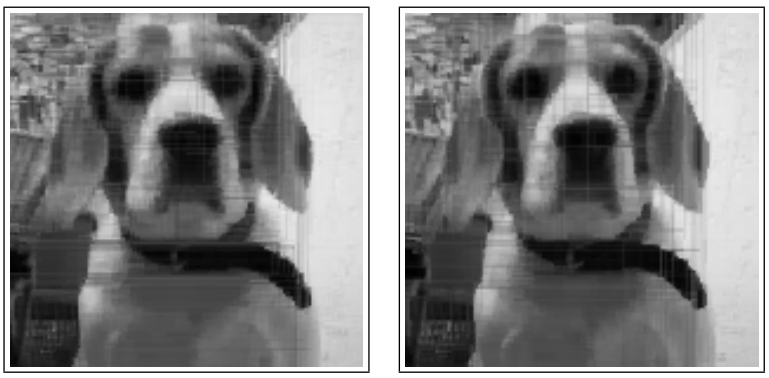
By bagging the inter-tree diversity is encouraged, since every tree is trained with only a subset of the data. Training every tree with a smaller random subset from the data will amplify this effect, as shown in Figure 3.13.



(a) In-bag = 5% (b) In-bag = 10% (c) In-bag = 20%

(d) In-bag = 66% (e) In-bag = 80% (f) In-bag = 100%

Figure 3.13: The prediction of a forest of 5 trees, with an increasing amount of in-bag data sampled with replacement, the other parameters fixed at the values given in Table 3.1



(a) With replacement (b) Without replacement

Figure 3.14: For our example using the picture of a dog, sampling with or without replacement does not change the outcome noticeably. This is largely due to the low dimensionality of the problem, and small forest size. The amount of in-bag data is 100% in these two figures, other parameters are the same as in Figure 3.13.

3.2.4.4.4 Influence of the Candidate Subset Size

An important source of randomness in RF is the candidate subset size. This will prevent trees from all choosing the same decision variable, thus preventing the steps in the response to align. This impact is shown in Figure 3.15. Remark that having only one candidate is the same as choosing a random decision variable. As expected the image where trees are more diverse is more blurry, indicating a smoother approximation.

(a) # Candidates = 1 (b) # Candidates = 2

Figure 3.15: Prediction of a forest using 20 trees, the other parameter values are given in Table 3.1. Since the dimensionality of this problem is only two, all trees used in the figure on the right will always split optimal. Although the crispness of the image indicates that the predicted response is not as smooth, due to the lack of diversity. This can also be an indication that the model is overfitting the training data.

3.2.4.5 Interpolation Example

To show RF handling input values within the domain of the training data, we revisit the problem of modeling an image as in Section 3.2.4.4. Here we scaled the image to 1500x1500 pixels by predicting the response on a finer grid in the input domain. As can be seen from Figure 3.16, this interpolation is reasonable.

Figure 3.16: Interpolation using a forest constructed with the default parameter values given in Table 3.1. No extra artifacts are introduced.

3.2.4.6 Extrapolation Example

To show RF handling input values outside the domain of the training data, we revisit the problem of modeling an image as in Section 3.2.4.4. Here we generated a model and sampled outside that range. The result is shown in Figure 3.17. This behavior was to be expected, knowing the prediction mechanism of a tree. Predictions outside the training ranges are the same as the prediction of the closest point within the training range.

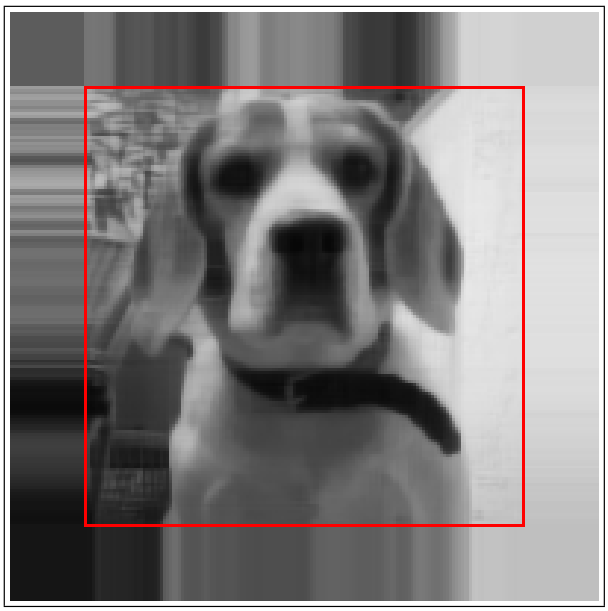


Figure 3.17: Extrapolation from an random forest model constructed with the default parameter values as given in Table 3.1. The predictions outside the training range are the same as the prediction on the boundary of the training range.

3.3 Variable Importance

Several approaches to VI in RF exist. Accumulating the Gini index was proposed originally, later complemented by a technique which we will call 'Shuffle'. For a comprehensive comparison of linear regression and RF with respect to VI we refer to [31].

Throughout this section we illustrate the different variable importance measures using an example. Consider the data set given in Section 3.2.4.1. If we finish the tree construction, using a leaf size of one, the resulting tree is as follows:

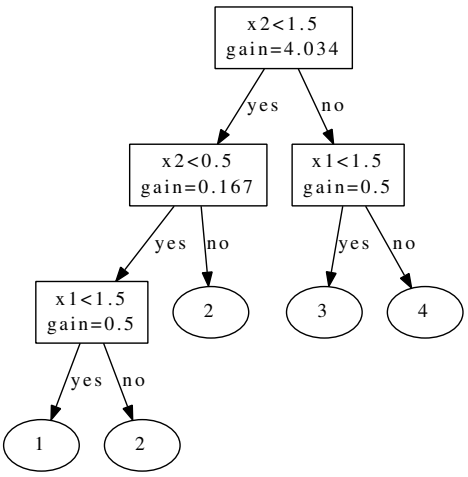


Figure 3.18: The tree generated for the data set given in Section 3.2.4.1. For this example a leaf size of one is used, so every response value is in a separate terminal node.

3.3.1 Accumulating the Gini Index

The method originally proposed was to use the value of the splitting criterion. Here the VI of variable i is the sum of G over all splits where i is the decision variable. Formally let $g^j_{k,c}$ be the value of criterion G for split j with decision variable k and decision value c. Then define the VI of variable i as follows:

$$\mathcal{I}_i = \sum_j g^j_{i,c} \tag{3.7}$$

We apply this importance measure on the tree given in Figure 3.18. Summing the information gain of each variable results in the following importances:

Variable	Gini Importance	
x_1	$0.5 + 0.5$	$= 1$
x_2	$4.034 + 0.167$	$= 4.2$

In Section 5.2, we compare the accumulated Gini index to other VI measures in a series of experiments.

3.3.2 Importance Estimation by Shuffling

An alternative method for determining VI in RF is by perturbing data records and determine the change in prediction error. Each out-of-bag data point is evaluated using the value of that variable from another out-of-bag point in the data set. The resulting point is also known as a perturbed data point. More formally consider out-of-bag data perturbed in variable i:

$$\widetilde{X_i} = (x_1, \ldots, \widetilde{x_i}, \ldots, x_d) \tag{3.8}$$

where $\widetilde{x_i}$ is a random permutation of x_i. The motivation for this perturbation is that important variables are split more often, thus influencing the prediction path more than irrelevant variables. In essence the contribution of a variable to the prediction path is estimated. Such that by perturbing the data in an important variable, the perturbed points are more probable to end

up in a leaf 'further away' from their original leaf as the importance of the variable increases. The importance of a variable can then be defined as the average change of the SSE per tree. Or formally the variable importance of a tree t for variable i with out-of-bag set O_t:

$$\mathcal{I}_i^t = \frac{1}{|O_t|} \sum_{j \in O_t} (\hat{f}(\widetilde{x^j}) - y^j)^2 \tag{3.9}$$

The VI as computed by a forest with a set T of trees can then be formulated as the average over all trees:

$$\mathcal{I}_i = \frac{1}{|T|} \sum_{t \in T} VI_i^t \tag{3.10}$$

Remark that this definition is *interpretable* as defined in Chapter 2. Note that other yet similar techniques are proposed as well [44]. In classification problems it has been shown that the variable importance is biased towards variables with many categories [106], so caution is advised when mixing categorical and continuous variables. In [28] different strategies are proposed for selecting variables either for explanation or prediction, in addition they also examine the influence of several parameters such as data dimensionality, number of data points, size of K and the number of trees.

This shuffle technique is a random procedure, and it is suggested to compute the VI several times using a different permutation of $\widetilde{x_i}$.

We apply this importance measure on the tree given in Figure 3.18. Let us estimate the importance of variable x_1 first. We shuffle all its values randomly and compute the square error for predictions of the perturbed data points:

x_1	x_2	y		x_1	x_2	$\hat{y}$	$(\hat{y}-y)^2$
1	1	2		**3**	1	2	0
2	0	2		**1**	0	1	1
3	2	4	$\rightarrow$	**0**	2	3	1
0	3	3		**1**	3	3	0
1	0	1		**2**	0	2	1

Summing the squared errors, we find that the importance score for x_1 equals 3. We repeat this procedure for variable x_2, We randomly order its values:

x_1	x_2	y		x_1	x_2	$\hat{y}$	$(\hat{y}-y)^2$
1	1	2		1	**0**	1	1
2	0	2		2	**3**	4	4
3	2	4	$\rightarrow$	3	**0**	2	4
0	3	3		0	**2**	3	0
1	0	1		1	**1**	2	1

Summing the squared errors, we find that the importance score for x_2 equals 10. In this example, variable x_2 is considered most important.

3.3.3 Counting the Number of Splits

Since the splits are effectively determining the tree, we propose to count the number of splits on a variable for a fast approximation of the VI. Since this information is available during

model building, it would not incur any extra computation such as with the aforementioned 'Shuffle' technique. Formally:

$$\mathcal{I}_i = |\{g^j_{k,c} | k = i\}| \tag{3.11}$$

Remark that this counting technique is not usable if the candidate set size is one. Since all variables have equal probability of being selected as candidate decision variable, all variables would be assigned the same importance.

We apply this importance measure on the tree given in Figure 3.18. Both variables are split twice, so they are considered equally important by this importance measure.

3.4 Properties

3.4.1 Adaptive Nearest Neighbors

As observed from the examples in Section 3.2.4, a tree prediction can be interpreted as a local property in a neighborhood. The information gain criterion favors nodes with similar values, so intuitively data points grouped in a leaf can be considered 'close' together. Remark this intuitive notion holds because the decision boundaries divide the input space in convex partitions. This behavior resembles a nearest neighbors prediction, a notion also discussed in [8]. Although we will not explore this property further, this similarity with nearest neighbors provides considerable insight into tree prediction. While we only mention this property informally, we refer to the literature for a formal treatment [70].

3.4.2 Model Extrapolation

The prediction of a tree is the average of a combination of responses present in the data Y. The prediction of a tree or forest will therefore never leave the observed response range. In addition the prediction is based directly on the locality of the new input point x compared to points in X. In case x is not in the domain of training points, the resulting prediction reflects the prediction at the boundary of the input domain.

3.5 Applications

Random forests are capable of both classification and regression tasks, but are also routinely used for variable screening. The field of bio-informatics extensively uses random forest for analyzing micro-array data in screening studies [5, 23, 75, 15]. Another interesting application of RF is the study on fault diagnosis [76, 17, 120]. Also for tasks closer related to pattern recognition the RF technique is attractive. For instance RF is used for face detection and recognition [7] and predicting the age of a person through face image analysis [80].

3.6 Summary

In this chapter we have shown how to build a classification and regression tree (CART) and how they can be combined in a forest. We observed that the prediction of a single CART is a multidimensional step function, and not a good approximator for smooth functions. A random forest (RF) is able to approximate smooth functions by averaging the predictions of several CARTs.

The influence of different parameters of RF was shown using an example where the gray-values of an image were approximated. We concluded that for this low dimensional problem a low number of trees was sufficient for a good prediction. Using a larger leaf size yielded smoother results, and this parameter is of interest for noisy problems. By increasing the candidate set size we observed a lower prediction error, but stressed the need for diversity in the population of trees. The diversity is also influenced by the bootstrap aggregation or bagging, and good results were observed with the default 2/3 in-bag data. In addition we have shown that RF interpolates well for our test problem, while extrapolation is not meaningful by the definition of the tree construction.

We have shown two variable importance measures for RF. A shuffle method based on evaluating out-of-bag points with perturbed values, and the accumulation of the splitting criterion for a variable called Gini index. The behavior of these measures will be investigated in Chapter 5.

Chapter 4

Symbolic Regression

4.1 Motivation

Symbolic regression (SR) differs from traditional regression since it does not rely on a specific *a priori* determined model structure. The only assumption made in SR is that the response surface can be described by an algebraic expression. Instead of the traditional approach where the model structure is fixed and the remaining free parameters are optimized, SR reformulates the regression problem as a search problem for the optimal model structure. Once a model structure of sufficient quality is found, traditional techniques can be used to find the optimal coefficients.

The goal of finding a global model described by an algebraic expression certainly is ambitious. Since the number of possible expressions is infinite, it is apparent that a powerful search technique will be required. This ability to efficiently find solutions in a vast search space is provided by genetic programming.

This approach to regression has been shown to perform well for several industrial data sets [102, 50, 6], where the data is typically noisy, of high dimensionality and exhibit strong non-linear behavior. A discussion on the advantages of evolutionary computation in industrial applications is found in [58]. This shows beyond doubt that this technique holds promise, and motivated us to provide an in-depth treatment.

4.2 Algorithm

The algorithm used for SR is best explained in a structured fashion, so all concepts are introduced in their proper context. To this end we explore genetic algorithms first, of which genetic programming (GP) is a specialization. Then GP is augmented to Pareto GP, by incorporating the concept of Pareto optimality. The theory of ordinal optimization is a valuable addition and results in Ordinal Pareto GP. We briefly mention an advanced technique called ESSENCE. We conclude this section with examples.

4.2.1 Genetic Algorithm

A genetic algorithm (GA) belongs to the larger class of evolutionary algorithms (EA). The goal of a GA is to solve an optimization problem, by evolving towards a solution, not unlike iterative refinement methods. Where this method differs is that the evolutionary process is

inspired by nature, in particular mechanisms of information propagation through reproduction and survival. An evolutionary process is not an endeavor undertaken alone, requiring any GA to maintain multiple intermediate solutions. We adopt the terminology used in the GA literature, reflecting the biological inspiration of the technique. The collection of intermediate solutions is referred to as the population, where an individual is a single intermediate solution.

Every individual is encoded by a chromosome, traditionally represented by a string of consecutive zero or one values. This could used for denoting presence or absence, for example in context of a FS technique. This encoding is not always practical, and in general the choice of encoding is problem dependent. For example GAs are also used to solve permutation problems, where an optimal order of a fixed set of values must be found [72].

A fitness function will determine the quality of an individual, and is a guiding force throughout the evolution. In other words the fitness function will reward and encourage successful individuals while penalizing unfit individuals.

The evolution to a solution takes the form of an iterative stochastic search, where each iteration is referred to as generation. Typically an initial population is composed of randomized individuals. Their fitness is evaluated at every generation, and a new population is composed by stochastically selecting individuals (also based on fitness) and recombining them in a meaningful way. This recombination is referred to as reproduction, where the original individuals or parents create a new population of children. To this end mutation and crossover operators are used. Similar genetic operators are also found in nature.

A mutation is an operation partially changing the chromosome of a single individual. Traditional crossover will combine the chromosomes of two parents, although multi-parent operators are also a valid option. Simple examples are shown in Figure 4.1 and 4.2, but many variants of these operators exist. When domain knowledge is available and the encoding allows it, special purpose operators can be defined to accelerate the evolution. The convergence to an optimal solution depends on the applicability of the reproduction operators to the problem. A general requirement for crossover operators is that there is a reasonable probability that the offspring is of higher fitness than the parents.

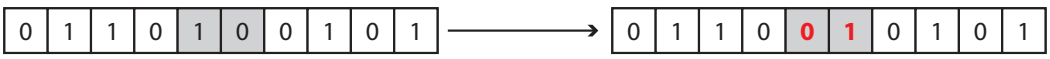
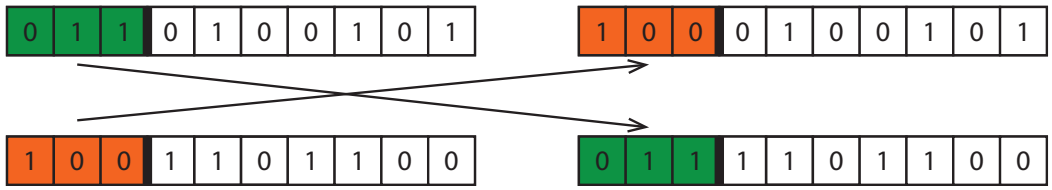


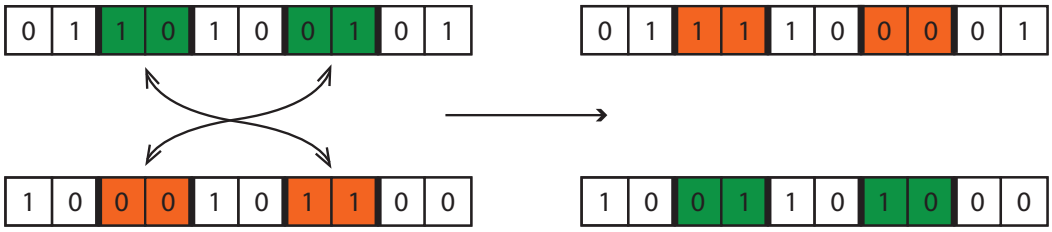
Figure 4.1: Example of a simple two-point mutation operator in GA.

In any search problem there is always a trade-off between exploring new areas and exploiting known areas. In the evolutionary search performed in GA this is no different. On the one hand it is desirable to converge if good solutions are found, but on the other hand sufficient exploration must be guaranteed in order to safeguard against convergence to local optima. In other words the population diversity must be preserved. Some reproduction operators might be specifically designed to create diversity, even if this means degrading the quality some individuals for this purpose. In addition the rates of mutation and crossover will influence this as well. Intuitively a mutation can bee seen a random jump made by an individual while offspring from crossover is expected to be a step in a reasonable direction. This implies that more mutations will slow down convergence, but there is more exploration.

There is no way to predict the result of an evolution except by observing it. This reformulates the original modeling problem from a search problem to a dynamic system problem. Thus it becomes a priority to gain insight in how this complex process is influenced



(a) One point



(b) Two Point

Figure 4.2: Examples of simple crossover operators in GA.

by the set of rules that define the evolution. Again the problem is largely a matter of representation. In other words the encoding of the chromosome and the operations performed on them should make sense. This is one of the reasons why a GA can be very efficient. Consider some of the traditional crossover schemes as shown in Figure 4.2. Intuitively they extract some building blocks from the parents and recombine them to produce the offspring. With a good encoding, the building blocks selected by this operators would represent logical entities such their combination is meaningful.

4.2.2 Genetic Programming

Genetic programming (GP) is a specialized form of GA, in the sense that an individual is explicitly assumed to be a parse tree instead of a string of values and therefore has a variable length.

These parse trees often correspond to algebraic expressions, and GP is primarily used for finding functional descriptions of input-output data, i.e. modeling. We refer to the literature for details [60, 61, 62, 63].

While it is more common to evolve expressions or formulae, computer programs can be evolved as well. As correctly observed in [2] : "To be evolutionarily effective, a fitness function that measures the functionality of a result must correlated positively with a similarity measure comparing the generated form with a fully effective form." Which is problematic when generating source code from scratch, since the optimal fitness function would already know the answer. For example in [53, 54] a sorting algorithm is evolved, where the fitness function is defined as how many swaps are needed to correctly sort the output generated by an evolved program. This fitness function expresses 'distance to a good solution' but requires understanding the problem, namely that a swap would be a logical measure for performance. In this thesis we will not explore the automatic program generation further, but will limit ourselves to expression trees representing an algebraic function.

When the fitness function in GP, constrained to algebraic expressions, is defined as a

regression error function, the resulting technique is known as symbolic regression (SR). In other words GP is the evolutionary framework, while SR imposes a fitness function so individuals will approximate the relationship in the data set. In [52] it is shown that the fitness function $1 - R^2$ where R^2 is the square of the scaled correlation between prediction and response, produces better results than the MSE. In essence this will measure the fitness of the model should it be optimally scaled. This allows for faster identification of interesting model structures compared to the MSE.

In expression trees we distinguish between operators and terminals. Operators are functions taking terminals as argument, for example plus or minus. Terminals are values or variables the operators can process, such as a constant 2 or variable x_1. Throughout the evolution care must be taken to always generate valid expressions.

Remark that in order to find good expressions modeling a data set, the relation must be captured by the functions available to the GP algorithm. For example data generated from an exponential function can not be found starting when using only the operators $\{+, -, *, /\}$. In other words an exact solution can only be found if the set of primitive functions is sufficient to express the underlying relationship.

A good strategy to select individuals to create offspring is a tournament selection. A tournament is organized for each participant in the reproduction, each tournament having a random subset of individuals as competitors. The winner is then chosen to be the individual with highest fitness. Remark how this method can be considered fair, since individuals with average fitness still have the chance to reproduce. If this behavior would be discouraged, the individuals with highest fitness would always produce the most offspring, leading to stagnation of the population. In other words, performing selection by holding tournaments does not jeopardize the diversity of the population.

In the simple crossover technique shown in the previous section, parents swapped a part of their chromosome. In GP this corresponds to swapping subtrees as shown in Figure 4.4. A one point mutation in GP corresponds to changing a single node in the parse tree as shown in Figure 4.3. Remark that in order to create acceptable offspring with mutation, an operator must be replaced with an operator of the same arity.

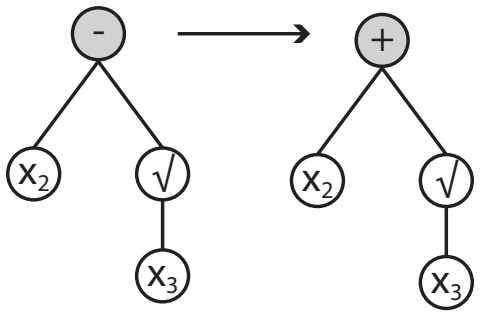


Figure 4.3: One point mutation example in GP. The operator minus is replaced by another operator of the same arity.

4.2.3 Pareto GP

In GP we already represented an individual as a parse tree, but skipped over some important issues. For example, the depth and complexity of individual trees. Consider what happens

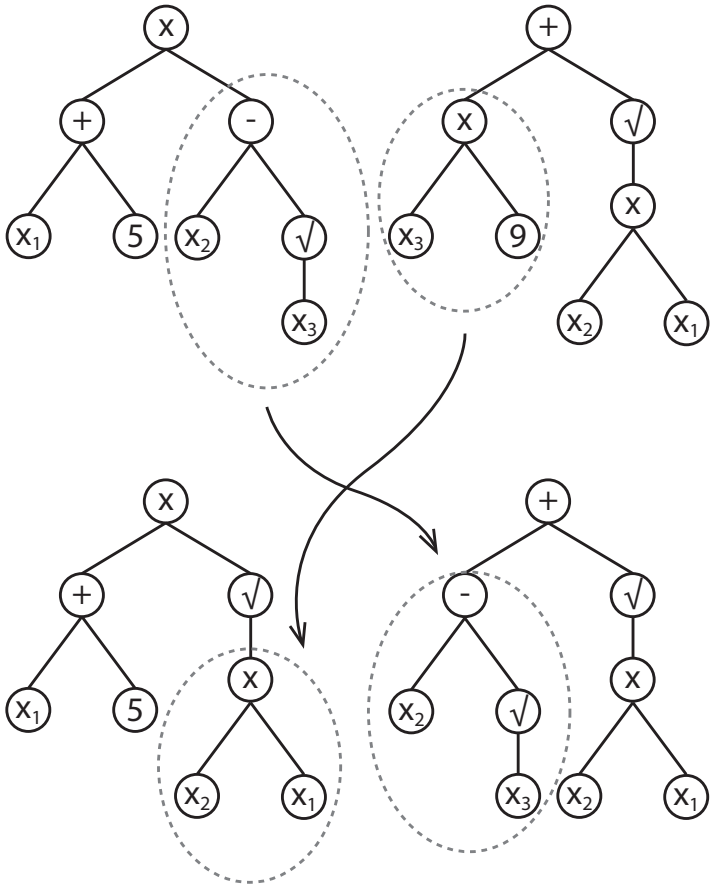


Figure 4.4: One point crossover example in GP. Two individuals effectively swap subtrees, resulting in two new expressions.

when using the simple crossover operators as shown in Figure 4.4. Since the mutation points are chosen randomly the newly generated expressions could very well be larger than the parent, and have a deeper tree. While this behavior could be explicitly controlled by correcting the situation as it occurs, it would be desirable to have a generic concept within the rules of the evolution to diminish the impact of a possibly unlimited increase in expression complexity.

In addition we must recognize that most of real-world optimization problems do not have a single objective function, and optimal decisions involve competing trade-offs. An example is price versus quality. While everyone would like to have high quality products, another objective is not to spend more resources than necessary. The solution results in finding an acceptable balance in quality-price. This principle is also applicable in GP, and proves to be very useful.

In GP the goal is to find high quality models, which should be as simple as possible but not simpler. In other words when given the choice between two models having the same predictive properties, the simpler is preferred. This is formalized using the notion of Pareto optimality. In the context of conflicting objectives, a solution is Pareto optimal if it is not dominated by another solution. A solution is known to be dominating another solution if it performs better on at least one objective while not scoring less in any other objective. This is also shown in Figure 4.5.

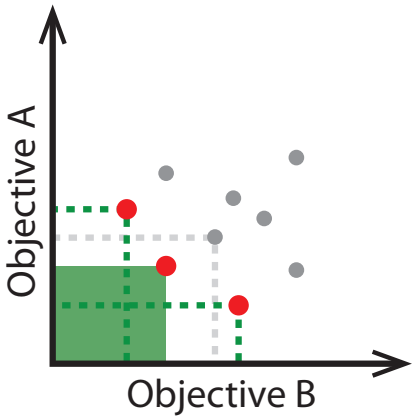


Figure 4.5: Example of a Pareto optimality when minimizing two objectives. The slightly larger red dots are part of the so-called Pareto front and indicate Pareto optimal solutions. They are optimal since they are not dominated by any other solution. For example, the shaded area indicates the space dominated by the point in its upper right corner. For several other solutions the dominated space is indicated using dotted lines.

By introducing the concept of Pareto optimality, the search problem for good solutions is augmented with a powerful tool to handle multiple objectives. Remark that while we only mention complexity as an additional objective, Pareto optimality is also applicable in case more than two objectives are involved.

A problem previously left untouched lies in the creation of the new generation. Consider what would happen to an individual scoring very good on the fitness criteria. If the new generation consists entirely out of offspring created by mutating and recombining existing individuals, such good performing solutions might be lost. This is undesirable, and to avoid this loss of knowledge an elitist strategy is applied. An archive of the set of current optimal individuals is maintained. Remark that optimal refers to the Pareto optimality, or an alternative niching strategy in a space of selected objectives. This quality preservation strategy (elitism) is instrumental in the parent selection since it can be expected that the probability to create offspring with good fitness will be higher if one of the parents was an elite solution. Niching is instrumental to preserve the diversity of population.

Another issue previously left untouched is the stagnation of the population. When the population converge to a solution, all members of the population will become similar. In fact all individuals can be exactly the same in an extreme case. The population might be converging to a local optima, and after diversity is lost there would be no way to escape this local optima. To this end the population is reinitialized periodically while retaining the archive, where the lifetime of a population is known as a cascade.

Pareto GP with a suitable fitness function is known as Pareto-aware Symbolic Regression, or SR via Pareto GP [116, 101] and is implemented in [24].

4.2.4 Soft Ordinal Pareto GP

In traditional GP systems, the fitness of individuals is determined at every iteration using the full data set. In practice SR would use all input data for predicting the performance of all current models, determine the approximation error. In general the most computationally

intensive part of any GP is this fitness evaluation. A fast approximation of the fitness is of interest, as this has a significant impact on the performance.

We will only briefly mention the concepts of ordinal optimization (OO). We refer the interested reader to the paper introducing OO [38], and to [39] for a recent work which theoretically validates the intuitive assumptions of OO.

The two fundamental ideas in OO:

- 'Order' is easier to determine and much more robust against noise than 'Value'.
- Insisting on getting the 'Best' is too expensive, be willing to settle for the 'Good Enough'.

In other words the two ideas of OO are to focus on fitness ranking in combination with softening the goal. An example from [39] explains the difference between an ordinal and cardinal approach elegantly:

> Imagine you hold two identically looking boxes with unknown content in your two hands. You are asked to judge which one is heavier than the other – and 'ordinal' question. Almost all of us can correctly tell the answer, even if there is only a very slight difference between the weights of the two boxes. But if we are asked to estimate the difference in weight between the two boxes – a 'cardinal' question, we will have a hard time.

In context of SR this would translate to model evaluation using only a subset of the full input data. Details are given in [110], and we refer to [100] for an application of this approach to Pareto aware SR.

This happens to be not only more efficient in terms of time spent on evaluation, but also guarantees more diversity in the population. With this approach even average individuals can rank higher on some subset evaluation, increasing its chance to create offspring. By using a different subset at each generation the quality of the population will be preserved, while at the same time encouraging more exploration.

By the end of the evolution the individuals should model the entire data set. This is achieved by linearly increasing the subset size during the evolution. In effect the initial populations will only have very limited partial representation of the data set, which allows more exploration. By increasing the subset size the focus is gradually shifted from exploration to exploitation in later generations.

This strategy opens an interesting question with respect to updating the archive. To be fully consistent the archive should be re-evaluated on the data subset in use by the current generation, so individuals between population and archive are comparable. As shown in [116, Ch. 6] it turns out re-evaluating the archive at each generation is not strictly necessary to provide good results. Re-evaluating the archive at every cascade will make sure that lower quality individuals will be removed, although they can survive within a cascade. This allows for more diversity and exploration during a cascade, while only the truly good individuals will survive multiple cascades. This approach is called Soft Ordinal Pareto GP.

4.2.5 ESSENCE algorithm

In Soft Ordinal Pareto GP, the idea was introduced of evaluating individuals on only a subset of the data. This strategy is further enhanced using the ESSENCE algorithm expected to perform Effective Search Space Exploration via Nested Content-based Evolutions [116, Ch.

7]. Here the data subsets are nested, meaning that subsets used later in the evolution also contain all points used previously. By using nested data subsets the periodic re-evaluation of the archive will require less computational resources since only the newly added points since the last evaluation need to be computed.

In addition the subsets are chosen to be maximally informative, such that good global models could also be found from a small subset. Instead of increasing the subset size linearly, the information in the subset is increased linearly. Remark that this may result in a non-linear increase in subset size.

To elaborate on how the subsets are constructed would involve treating the issue of data balancing briefly mentioned in Section 1.2.12.5. This is not within the scope of this thesis, and for more information on the ESSENCE algorithm and empirical results we refer to [116, Ch. 7]. We summarize that the ESSENCE algorithm is able to consistently achieve higher quality solutions in comparison with Soft Ordinal Pareto GP, given that the data set does not contain too many outliers and noise.

4.2.6 Examples

To show the capabilities of SR in practice we show two toy problems of limited dimensionality. Experiments in Chapter 5 will use real life data sets of much higher dimensionality, but for now we prefer low dimensional problems which are more intuitive.

4.2.6.1 Newton's Universal Gravitation Law

As an example of the capabilities of SR, we generate a 3 dimensional data set of 50 points following the universal gravitation law:

$$F = G\frac{m_1 m_2}{r^2} \tag{4.1}$$

Where F is the force attracting two masses m_1 and m_2 over a distance r. The constant G is the gravitational constant:

$$G = 6.67428 \times 10^{-11}, \quad \text{with unit } N(m/kg)^2 \tag{4.2}$$

The data was generated uniformly in ranges:

$$0 \leq m_1 \leq 1, \quad 0 \leq m_2 \leq 1, \quad 1 \leq r \leq 2 \tag{4.3}$$

The univariate plot of all inputs and response is shown in Figure 4.6. The data set was modeled using three independent runs of 150 seconds, and the resulting models are shown in Figure 4.7.

Table 4.1 shows optimal solutions as found by SR, confirming that algebraic expressions modeling the data set can be found. Remark that these models differ in their approximation of the constant G. The model with best prediction error found the correct value for the gravitational constant, such that the leading constant term is negligible.

Expression	$1 - R^2$	Complexity
$3.656 \times 10^{-27} + \dfrac{m_1 m_2 (6.674 \times 10^{-11})}{r^2}$	0.000	26
$1.067 \times 10^{-12} + \dfrac{m_1 m_2 (7.830 \times 10^{-11})}{r^3}$	0.043	25
$-4.843 \times 10^{-13} + \dfrac{m_1 m_2 (5.041 \times 10^{-11})}{r}$	0.064	23

Table 4.1: A selection of Pareto optimal, nearly exact solutions as found by SR in three independent runs of 150 seconds.

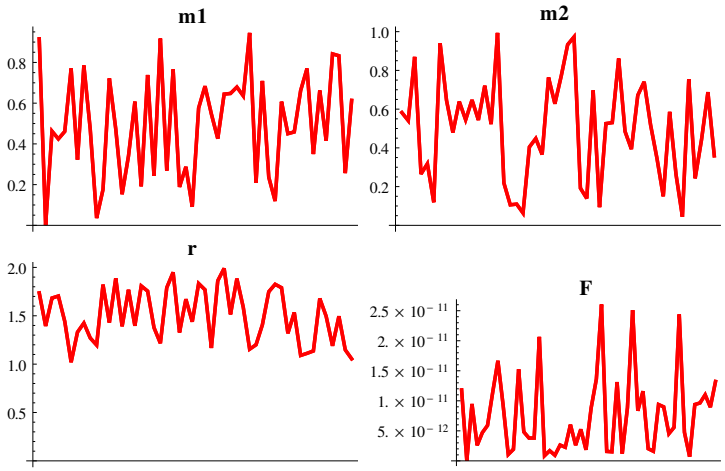


Figure 4.6: Univariate plot of all variables in the data set.

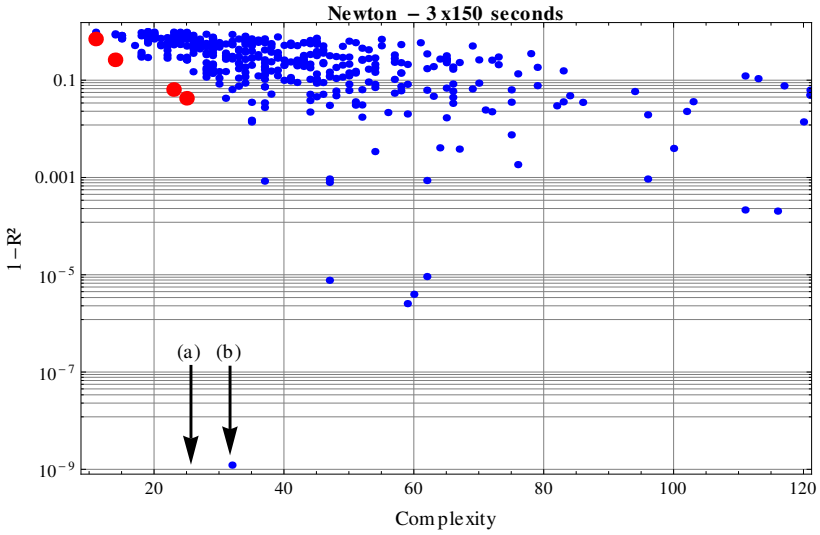
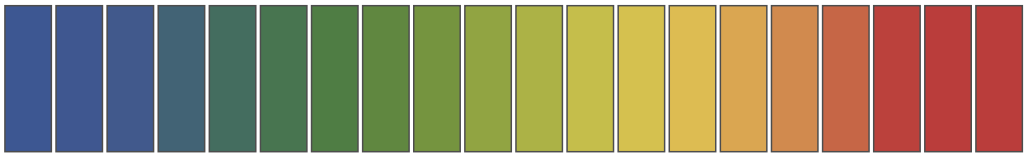


Figure 4.7: Pareto front of models obtained in three independent runs of 150 seconds. The Pareto optimal solutions with respect to prediction accuracy and model complexity are indicated by slightly larger dots. Remark that (a) indicates the optimal solution as shown in Table 4.1, such that $1 - R^2 = 0$, with complexity 26. This will dominate the model indicated by (b).

4.2.6.2 Defining Color Schemes

In later experiments we present results using bar charts. We wanted to use a visually pleasing color scheme such as the 'DarkRainbow' scheme in Mathematica, yet we generated many plots using Gnuplot [1]. The Gnuplot program is flexible enough to specify the colors of the bars independently, but does not have a default color scheme similar to the 'DarkRainbow' style.

To achieve consistency in the color scheme we proceed to model the color output in function of the bar index. We create the initial data set by sampling selected colors of a barplot in Mathematica using RGB values, as shown in Figure 4.8. The goal is finding a good model for each RGB component based on the index of a bar, where each RGB component is modeled independently.

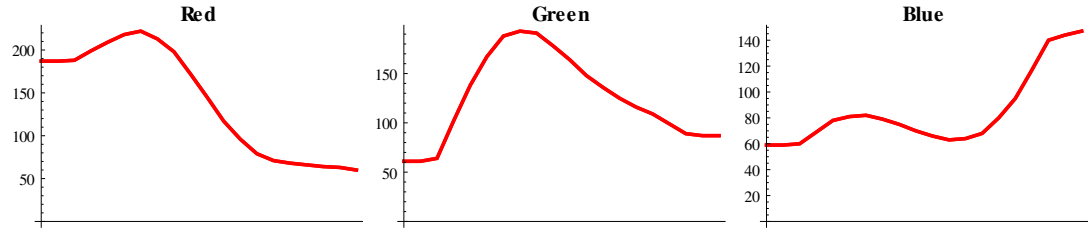


(a) Original Samples

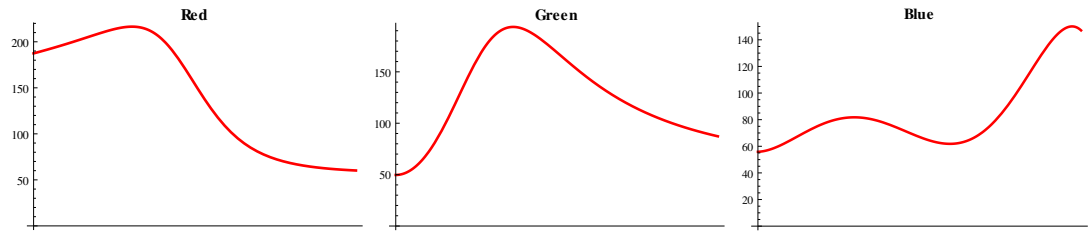
(b) Predicted Gradient

Figure 4.8: The color spectrum was sampled at the 20 discrete points shown in Figure 4.8a. Using those data points we modeled the RGB components using SR. The final model can be sampled over the full spectrum, resulting in the smooth gradient shown in Figure 4.8b

The relation of each component with respect to the index is shown in Figure 4.9. As can be observed there is not enough information to determine the exact model structure a priori.



(a) Original Samples



(b) Predicted Gradient

Figure 4.9: The RGB components plotted versus the bar index. Observe that models created by SR result in smooth functions.

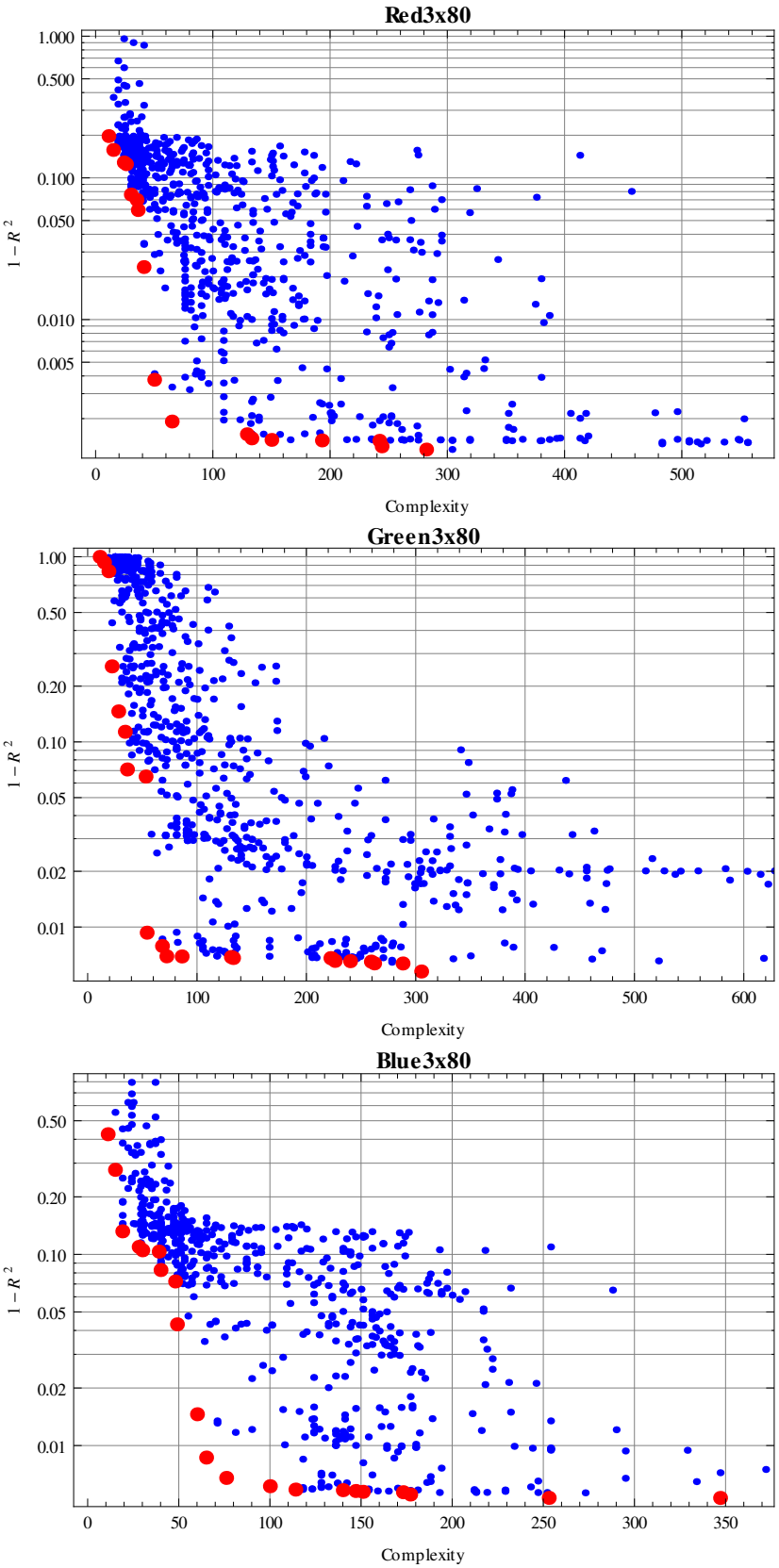


Figure 4.10: Pareto front of models obtained in three independent runs of 80 seconds. The Pareto optimal solutions with respect to prediction accuracy and model complexity are indicated by slightly larger dots.

All models produced in three independent runs of 80 seconds are shown in Figure 4.10. Since the Pareto front indicates optimal trade-offs, we select one model from the Pareto front for each component as the final solution, shown in Table 4.2. We find these models have a simple structure while still achieving very good prediction accuracy. The difference between the original color scheme and the solutions shown in Table 4.2 is on average two color values. This error is hardly noticeable by the human eye. While there are models with a higher accuracy, their model complexity is significantly higher. Remark that longer runs might provide even more accurate models with lower complexity, but this solution was sufficient for the purpose of providing a color scheme.

As Figures 4.8 and 4.9 confirm, colors based on these formulas not only match the original data points, but also allows for smooth interpolation.

Color	**Expression**	$\mathbf{1-R^2}$	**Complexity**
R =	$57.291 + \dfrac{181.657}{-i + 1.396 + i^6 62.633}$	0.004	50
G =	$49.667 + \dfrac{i^2 20669.400}{540i^4 + 9.536}$	0.009	54
B =	$55.856 + 10.002\left(i + 63.125i^2(2i - 1.202)^2\left(-i^2 + 1.202\right)\right)$	0.006	100

Table 4.2: The approximation for the red-green-blue component in function of the bar index i. We selected these models as solution based on a three independent runs of 80 seconds each.

4.3 Variable Importance

4.3.1 Population Based VI

Since SR uses a population of models a sensible approach is to examine the presence of a variable in the population. Models incorporating variables that are irrelevant to the problem will introduce extra complexity without achieving a better prediction error. Due to the Pareto optimality such individuals will have less chance to create offspring, which is an effective strategy to avoid irrelevant variables. Therefore, the presence of a variable in a sufficiently evolved population will provide an indication on whether that variable is relevant for describing the response. Remark that this is an *interpretable* measure as defined in Section 2.1.

We present presence-weighted and fitness-weighted VI as defined in [117]. Let us first define a function indicating whether a variable x_i is present in model M_j:

$$\delta(x_i, M_j) = \begin{cases} 1 & \text{if } x_i \subset M_j \\ 0 & \text{if } x_i \not\subset M_j \end{cases} \tag{4.4}$$

4.3.1.1 Presence-Weighted VI

The presence-weighted VI for any variable x_i with $i \in \{1, \ldots, d\}$ in a set of models $\mathcal{M} = \{M_1, \ldots, M_s\}$ is computed as the fraction of models containing x_i. Such definition provides

a robust estimation of relevance if the models in set $\mathcal{M}$ are of high quality and sufficiently diverse, i.e. obtained using several independent runs.

$$\mathcal{I}_i^{(PW)}(\mathcal{M}) = \sum_{j=1}^{s} \frac{\delta(x_i, M_j)}{m}, \tag{4.5}$$

4.3.1.2 Fitness-Weighted VI

The fitness-weighted VI eliminates the need for high quality in $\mathcal{M}$ (but does not eliminate the need for diversity) by weighting the presence indicator with the fitness of each model. It also takes into account the complexity of the model, by dividing the fitness equally over all variables present in a model:

$$\mathcal{I}_i^{(FW)}(\mathcal{M}) = \sum_{j=1}^{s} \frac{fitness(M_j)}{\sum_{i=1}^{d} \delta(x_i, M_j)} \delta(x_i, M_j) \tag{4.6}$$

Remark that in the extreme case of a population of perfect models, the fitness-weighted VI is equivalent with the presence-weighted variable importance. For the remainder of the thesis we will use the normalized fitness-weighted VI as defined by:

$$\mathcal{I}_i^{(NFW)}(\mathcal{M}) = \frac{\mathcal{I}_i^{(FW)}(\mathcal{M})}{\sum_{i=1}^{d} \mathcal{I}_i^{(FW)}(\mathcal{M})} * 100\% \tag{4.7}$$

4.3.2 Individual Based VI

Previously we described techniques to determine VI given a population of sufficient quality. However, we are also interested in measures to compute VI given a single model. To this end we explore several options, some only applicable to algebraic expressions such as provided in SR.

4.3.2.1 Derivation of the Expression

In our first discussion about importance in Chapter 2, we noted that importance is a local property. Furthermore in any point of the input domain the local importance of any variable would be the partial derivative over that variable. If we are given a model consisting of a differentiable function, we recall Equation 2.2:

$$\mathcal{I}_i(y, x^k) = \sum_{k=1}^{n} \frac{\partial \hat{f}}{\partial x_i}(x^k) \tag{4.8}$$

In Section 5.3 we will examine the behavior of this measure in several experiments.

4.3.2.2 Substitution by the Mean

It could be that a variable in the model actually assumes the role of a constant, thus never changing value. Then judging the importance using presence or absence as indicator would be flawed. The prediction error will change, but models of the same accuracy could also be build without this constant disguised as a variable. Replacing this variable by its mean would give an impression of the approximation error should this variable be removed from the model. In Section 5.3 we will examine the behavior of this measure in several experiments.

4.3.2.3 Variable Elimination

We defined importance as the change resulting from the presence or absence of a variable. A literal approach is to eliminate the variable from the expression. Let us introduce an elimination operator $\ltimes(x, expr)$ which eliminates variable x from expression $expr$ and yields a new expression.

In order to eliminate a variable we remove that variable and operators involving that variable. Consider addition and multiplication:

$$\ltimes(a, a + b) = b, \qquad \ltimes(a, a * b) = b \tag{4.9}$$

Remark that eliminating variables this way is curious, since variable 'a' is actually replaced by a value depending on the operator involved. In case of operator plus or minus, elimination as defined here is equivalent with replacing by zero. While in case of multiplication of division, the variable is replaced by one.

For unary operators such as sine and cosine, the operator is also eliminated such that:

$$\ltimes(a, \sin(a) + b) = b \tag{4.10}$$

Now consider a more difficult case of exponentials:

$$\ltimes(a, a^b + c) = c, \qquad \ltimes(b, a^b + c) = a + c \tag{4.11}$$

This choice is not obvious, since eliminating variable 'a' would remove variable 'b' from that subexpression as well. This illustrates that an elimination technique is rather crude. In Section 5.3 we will examine the behavior of this measure in several experiments.

4.3.2.4 Importance Estimation by Shuffling

It is also of interest to examine the primary VI technique used by RF, described at length in Section 3.3.2, when applied to expressions. In Section 5.3 we will examine the behavior of this measure in several experiments.

4.4 Properties

4.4.1 Model Structure

A key strong point of SR is that is does not make assumptions about the model structure, instead only the set of primitive functions must be specified. An optimal model will then be searched for, in the space of all possible models using these primitive functions. However, choosing an ill-fitted set of primitive functions will have consequences. On the one hand, when choosing the set of primitive functions too small, the problem can not be solved exactly and one can only hope that an approximation can be provided. On the other hand, if the set of primitive functions is too large, the search space is increased unnecessary.

4.4.2 Stopping Criteria for the Evolution

Since the SR algorithm is a stochastic process, it is not clear when the evolution should be stopped. The model building process is ideally a continuous evolution, and while convergence

is assumed, there is little information available about the speed of convergence or the proximity to a acceptable solutions.

A pragmatic approach is allocating a certain computational budget to SR, in terms of function evaluations or time in seconds. In practice models are accepted as final solution if their performance is good, and SR can not find better solutions in runs that are 'long enough'. For this reason it remains important to look for robust algorithmic configurations that ensure the discovery of models of sufficient pre-defined quality.

4.4.3 Prediction Confidence

The final solution produced by SR is no single model, but rather a population of models. If all these models are of sufficient quality, their disagreement can be used as confidence intervals for predictions. This idea is illustrated in Figure 4.11

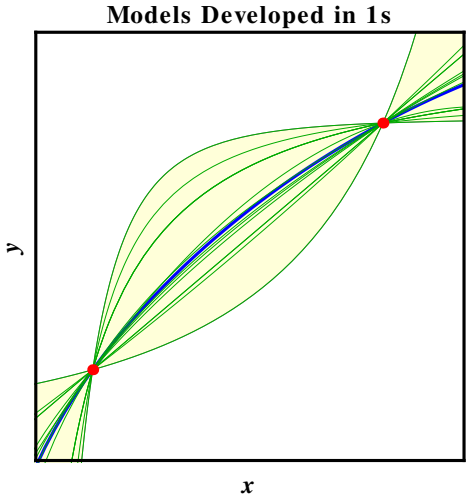


Figure 4.11: Response curves of models found by SR fitting the two points. The model disagreement can be used a measure for confidence.

4.4.4 Robust Model Building

As we will see in the later experiments, SR proves to be capable of handling irrelevant variables. In addition correlated variables and data imbalance are less of a problem compared with other techniques we studied. This does not imply that SR is insensitive to these problems, the model building is greatly facilitated if variables are not correlated and the data is balanced. However, we empirically observed that the performance of SR remains acceptable even in presence of all the aforementioned issues.

4.4.5 Computational Intensity

The reformulation of the regression problem to a search problem with infinite search space is not always favorable. When the regression problem can be solved using traditional modeling methods, these techniques are typically faster than SR. However, when traditional techniques prove insufficient, the heuristic exploration of an infinite search space is more attractive than giving up on solving the problem.

4.4.6 Model Interpretability

Although the convergence SR to a global optimal solution can be slow, approximate model structures as used in intermediate generations already provide insight into the problem. For example if preliminary runs indicate that the model structure is probably linear, traditional regression techniques might be used to solve the problem efficiently.

In addition all created models are an explicit algebraic expression and are readily interpretable. This full insight into the model is invaluable for domain experts. For example the domain expert might recognize a part of a known physical law, and the model can be easily adapted to incorporate this knowledge. This contrasts with models produced by RF or NN.

4.5 Applications

Symbolic regression is applied in sensory science, for modeling panel data. This is data obtained through experiments where a panel of subjects assigns a liking score to a sensory observation. When considering sensory information for taste it is of great interest to find optimal recipes and identify ingredients that drive liking. For background on why mustard comes in many varieties and ketchup does not we refer to [30]. Interpreting sensory panel data poses a challenge since flavors interacts non-linearly and the data is typically sparse. In addition panel members will have personal preferences, thus reacting differently to the same flavor. Segmenting the panel members according to personal preferences can be of commercial interest as well, since deciding how many different versions of a product to bring to the market will typically be founded on a sensory study to determine the potential interest. A study on such data set provided by Givaudan Flavors Corporation is presented in [117].

Developing soft sensors is a key application in an industrial setting. A soft sensor is a model to predict or infer the outcome of a process, implying that models need to be adaptive to changes in the process. The prediction of such models is for example used to control the process they describe, and optimize the process parameters in real-time. In [59] ensembles of SR predictors are successfully applied to the biomass estimation for the growth phase in a batch fermentation process.

Interval arithmetic [51] and affine arithmetic [91] can be used to analyze the output ranges of generated models during evolution. This allows for undesirable models to be rejected without doing the costly fitness evaluation. For instance models containing asymptotes can be detected this way. It is shown that applying such static analysis significantly improves the generalization performance of SR, while reducing over-fitting.

The prospects of auto-constructive evolution is discussed in [105], where operators for aspects such as reproduction are themselves encoded in the individuals. This will in effect co-evolve the problem solver as well as the solution. The results are preliminary, but it is speculated that auto-constructive evolution will extend the problem-solving power of GP.

4.6 Summary

This chapter situates symbolic regression (SR) in the context of evolutionary computation. While a genetic algorithm (GA) searches for solutions to an optimization problem, genetic programming (GP) searches for optimal parse trees describing functional input-output relationships. Pareto GP is a generalization of GP, where the fitness of an individual is determined

by the optimality of multiple objectives. Soft Ordinal Pareto GP applies goal softening by evaluating individuals on a subset of the data set, allowing more exploration and diversity. The ESSENCE algorithm also uses this approach, and increases success by strategically choosing the subsets and their size.

In Pareto-aware SR only parse trees of algebraic expressions are considered, and their quality is judged by their prediction error on a data set. A second objective is the minimization of the expression complexity, such that parsimonious models are preferred over complex models. This encourages SR to find models that are as simple as possible but not simpler. Examples illustrate the effectiveness of SR, rediscovering Newton's law of universal gravitation and deriving a smooth color gradient from a limited number of samples.

This chapter also discusses several variable importance measures for SR, either population- or individual-based. The behavior of these measures is investigated in Chapter 5.

Chapter 5

Experiments

It is of interest to study the techniques presented in the previous chapters in practice. This chapter provides selected results on experiments designed to increase understanding of the properties of each VI technique and their relation with the model structure.

First, the RF and SR methods are compared, followed by additional tests with alternative VI measures as introduced in Section 3.3 and 4.3.2. The same data set is used for both algorithms, and variables are uniformly sampled between zero and one unless stated otherwise.

All RF runs use the implementation detailed in Chapter 7, and we verified the results with available implementations. In all RF runs, a forest of 1000 trees was built, considering candidate subsets of the default $|d|/3$ size to determine the best split. The in-bag data was $2/3$ of the data, sampled without replacement. The shuffle method was used to compute VI, with 5 random permutations for each of the 20 independent runs. The mean normalized variable importances are plotted in bar charts, using the 25-50-75 percentile points as confidence interval.

All SR runs use the DataModeler package [4] in Mathematica. In all SR runs, the fitness function for prediction accuracy was defined as $1 - R^2$ where R^2 is the square of the scaled correlation between prediction and response. The sum of subtree sizes in an expression was used as complexity measure. Model age was used as the secondary complexity measure. Variable importance is computed over several independent runs. The mean normalized variable importances are plotted in bar charts, using the 25-50-75 percentile points as confidence interval.

Recall the desirable VI properties from Section 2.1. We will use these properties when discussing experimental results.

- **Interpretability**–Obtained importances should reflect the importances of the true input variables, without transformation.
- **Strictness**–Only variables relevant to describing the response should be allocated importances, so spurious variables should not appear important.
- **Conservativeness**–At intermediate stages of importance analysis all potentially interesting variables should receive importance.
- **Reproducibility**–A result can only be considered correct if it is reproducible.
- **Universality**–It is desirable for importances to be mutually comparable, and in addition to be problem independent.

5.1 Comparative Study

In this section we empirically compare RF and SR with respect to VI on diverse problems. For RF the Shuffling method is used as it is the most commonly used VI technique for RF, see also Section 3.3.2. For SR the normalized fitness-weighted VI is used, see also Section 4.3.1.2.

5.1.1 Linear Model

In linear models a logical measure for VI would be the coefficients since they have all the ideal properties as shown in Table 2.1. We argue that in these experiments the coefficients of a linear model are readily interpretable as a measure for variable importance, since all variables have the same range. The coefficients of a linear model are also strict and conservative. When considering traditional methods, linear models are also reproducible. We argue that the coefficients are also a universal measure, since the influence to the response is described problem independent.

When presented with a large data set, is is common that most of the variables are redundant, and it is desirable to remove them. To see if irrelevant variables will be identified we will consider a generating function:

$$y = 10x_1 + 10x_2 + 5x_3 + 1x_4 + 0x_5 + \ldots + 0x_{10} \tag{5.1}$$

Observe that only the first four variables are relevant to the response, and we added six spurious variables with zero coefficient. This way the methods are tested for *strictness*, as defined in Section 2.1. Optimal VIs are expected to correspond to the normalized coefficients, as shown in Figure 5.1.

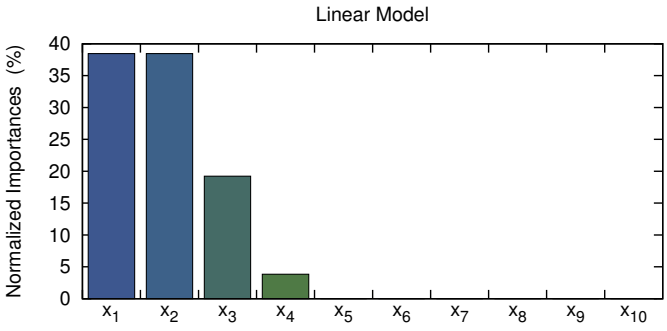


Figure 5.1: The normalized coefficients from the function given by Equation 5.1. We argue that for this linear problem VI should reflect this proportions.

5.1.1.1 Sampling Uniformly

For this test we used a data set of 500 points. The shuffle method by which RF computes the variable importance could capture the same relation, and results are shown in Figure 5.2. Remarkable is that the x_4 variable's importance is underestimated. This suggests that RF is *strict* but not *conservative*.

Because the discussed variable importance for SR only reflects a variable's presence, even though fitness-weighted, it is expected that the relative importances will not be well preserved. Remark that a population of perfect models would consider all variables equally important.

As can be seen from the results in Figure 5.2 the variable importance of both x_3 and x_4 is overestimated. This more *conservative* behavior safeguards against removing relevant variables, but might cloud the relative importances. The small importance score for the spurious variables is due to lower quality models. Their contribution is far less since they are fitness weighted, but will not be zero.

When adding many extra variables with zero coefficient these conclusions still hold.

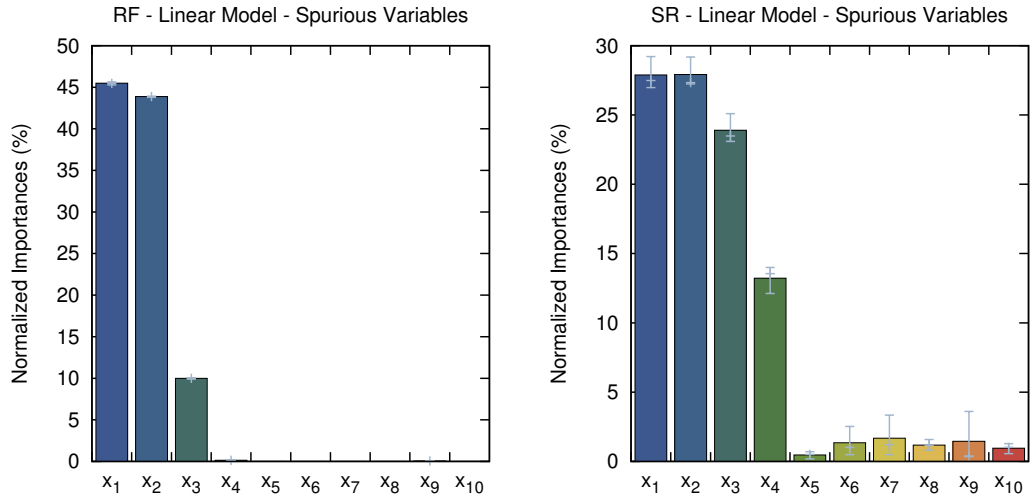


Figure 5.2: Comparing the results from RF and SR on data generated uniformly according to Equation 5.1. Observe that RF underestimates variable x_4, which was allocated an importance score close to zero. Also observe that SR overestimates the importance of variable x_3 and x_4, and the relevant variables are distinguishable from the spurious variables.

5.1.1.2 Imbalanced Data Set

In real problems the available data might not represent the input space well, as also discussed in Section 1.2.12.5. We mimic this situation by oversampling a variable such that its distribution is no longer uniform, here variable x_1 is sampled much denser in the interval $[0,\ 0.1]$. The generating function is given by Equation 5.1. A data set of 900 points was used, where the value of variable x_1 was sampled uniformly in $[0,\ 1]$ for 100 points, and in the interval $[0,\ 0.1]$ for the remaining 800 points.

The result obtained with RF is shown in Figure 5.3. We observe that the importance of variable x_1 is underestimated by a large margin.

Since more data points have an x_1 value in the interval $[0,\ 0.1]$, any perturbed data point is highly likely to be assigned a value from this denser area. And if the point already had a value for x_1 within this interval, little change in the response will be observed resulting in a low importance score. These findings agree that RF is not *conservative*.

This contrasts with the findings for SR, also shown in Figure 5.3. It is observed that SR is less susceptible to changes in variable importance due to an unbalanced data set. We argue that since SR builds global models and the underlying model is unchanged, the local difference in density will only slightly influence the variable presence.

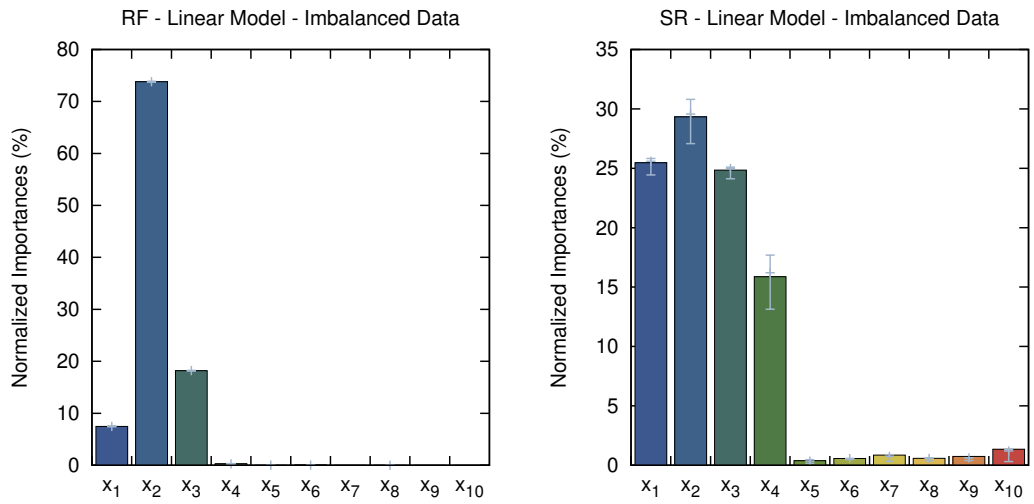


Figure 5.3: Comparing the results of RF and SR using an unbalanced data set sampling the function described by Equation 5.1. Remark that the importances obtained with RF for the densely sampled variable x_1 are significantly lower than when using a uniformly sampled data set such as in Figure 5.2. While SR also allocates less importance to x_1 than before, the same important variables can be identified as with the uniformly sampled data.

5.1.1.3 Relatively Weak Variables

Here it is examined how both methods deal with variables which are considerably less important than others, yet not redundant. This test verifies whether the methods are *conservative*, as defined in Chapter 2. A data set of 500 points was generated using the function given by Equation 5.2:

$$y = 10x_1 + 10x_2 \dots 10x_8 + 1x_9 + 0x_{10} \tag{5.2}$$

The variable importance produced by RF are shown in Figure 5.4. We observe that the variable with relatively low coefficient (x_9) has an importance close to zero, supporting the observation that RF is not *conservative*. In addition the relative importance of the variables x_1 through x_8 vary considerably, implying that RF is not *universal*. Since most variables are of equal importance neither will consistently provide a better split than its peers, but still always better than the variable with a low coefficient. The potential splits on the equally important variables differ only slightly according to the splitting criterion. They will reflect the distribution of the input variables which might exhibit slight non-uniformities due to the limited sample size. We argue that the splitting criterion amplifies these slight variations in density.

The same situation is better handled in SR, of which results are shown in Figure 5.4. While the importances are not exactly the same, this is expected since the population can still yield suboptimal models causing a small variation in the importances, even though the variables are have equal coefficients.

Remark that the importance of a variable with lower coefficient is overestimated. In the specific case that many significant variables' contribution is approximately equal, Latent Variable Symbolic Regression (LVSR) performs better than SR with respect to *strictness* (see [77]).

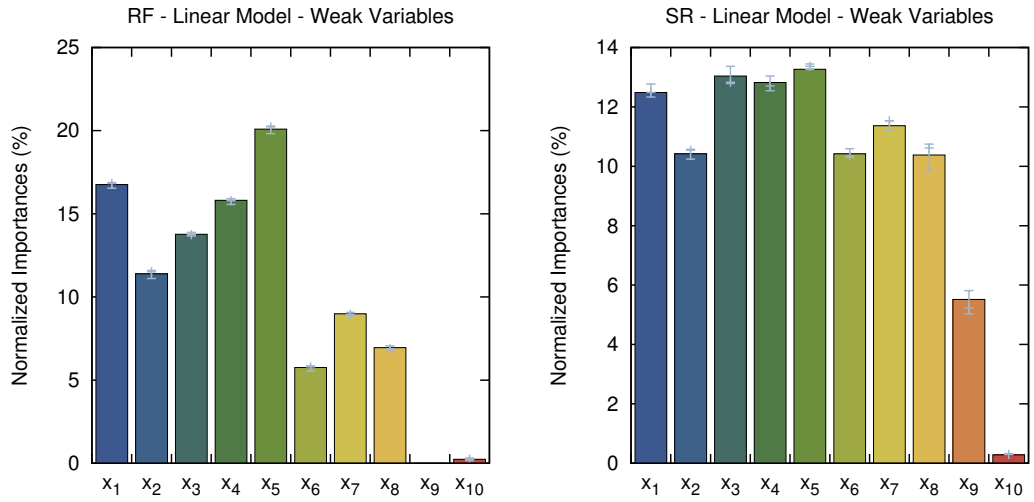


Figure 5.4: Comparing the results of RF and SR on data generated uniformly according to Equation 5.2. While variables x_1 through x_8 have the same coefficient in the generating function, we observe significant differences in the importance scores from RF. In addition a variable (x_9) with a small coefficient compared to the rest, is not distinguishable from x_{10} which is irrelevant noise. Variable x_9 is still identified as influential by SR, while the differences in VI for x_1 through x_8 is less pronounced.

5.1.1.4 Correlated Irrelevant Variables

Frequently not all variables in a data set are independent, and input variables are correlated. Such data set was created by adding new variables related to a true variable. In this experiment variables x_i for $i = 1, \ldots, 10$ satisfy Equation 5.1, while variables x_j for $j = 11, \ldots, 20$ are constructed by

$$x_j = 0.9 * x_1 + 0.1 * Noise \tag{5.3}$$

where $Noise$ is uniformly distributed in [0, 1]. We used a data set of 500 points.

In RF the variable importances change dramatically as illustrated in Figure 5.5. Because variables correlated with x_1 provide splits of comparable quality they can also be selected as decision variables instead of the true variable x_1, even though this correlated variable might have coefficient zero in the generating function. This causes x_1 to lose its influence over the prediction path through the tree, resulting in a lower VI score. These findings support the notion that RF is not *conservative*.

The best models obtained with SR do not contain the correlated noise variables. But since those correlated variables can still serve as a surrogate for the true variable to build rough approximations, they will remain active in the population. Consequently, SR will always allocate some importance to the noise variables, as can be observed in Figure 5.5. This indicates that SR is *conservative*. We argue that this is not conclusive about the *strictness* of SR, since even with the overestimated VI of the correlated variables the true important variables are distinguishable.

We conclude from this experiment that it is possible that the importance of a variable is artificially low when using RF, if irrelevant variables correlate with it. Those irrelevant variables will obtain an importance that might even be larger than the true variable. We

explain this by relating the shuffle score to the Gini score and candidate set size in Section 5.2.1.3.

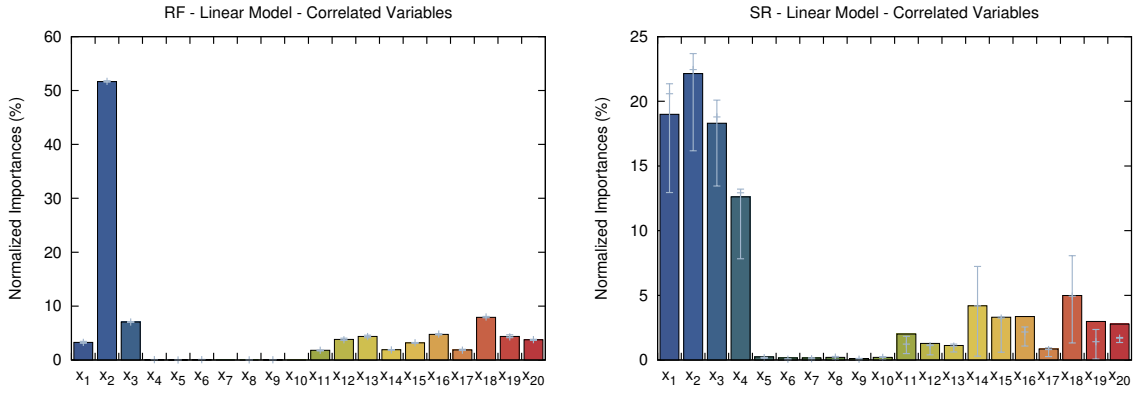


Figure 5.5: Comparing the results of RF and SR using correlated variables, the generating function is described by Equations 5.1 and 5.3. Observe that all variables x_{11} through x_{20} are correlated with x_1 and receive a non-zero importance score. While RF actually distributes the importance of x_1 over the correlated variables as explained in the text, importances obtained by SR can still identify all relevant variables.

5.1.1.5 Correlated Relevant Variables

In an experimental setting there is the realistic possibility that the researcher is not fully free to choose the input values to a process, due to external constraints. To mimic this situation we correlated relevant input variables. A data set of 500 points is generated using the function given by:

$$y = 10x_1 + \ldots + 10x_6 + 1x_7 + 0x_8 + \ldots + 0x_{10} \tag{5.4}$$

Where variables are constructed as follows:

$$x_i = 0.5 * x_3 + 0.5 * Random, \quad \text{for } i \in \{2, 4, 5, 6\} \tag{5.5}$$

Where *Random* is uniformly distributed in $[0,\ 1]$. Remark that x_1 is independent of any other variable.

Results are shown in Figure 5.6. We observe that RF scores x_3 very high compared with other variables with coefficient 10 in the generating function. In addition x_7 is not recognized as important analogously to the experiment from Section 5.1.1.3. The importances obtained by SR are consistent with earlier experiments. The importance of x_7 is overestimated by SR, but variables with the same coefficient in the generating function have approximately the same importance score. This more *conservative* ensures that even variables with a lower influence will not become extinct in the population.

We conclude from this experiment that it is possible that the importance of a variable is artificially high when using RF, if other relevant variables correlate with it. This can result in other independent variables receiving a much lower importance even though they have the same coefficient in the generating function. We explain this by relating the shuffle score to the Gini score in Section 5.2.1.4.

The results of RF are consistent with the literature [107], where a conditional VI is proposed which greatly improves this behavior.

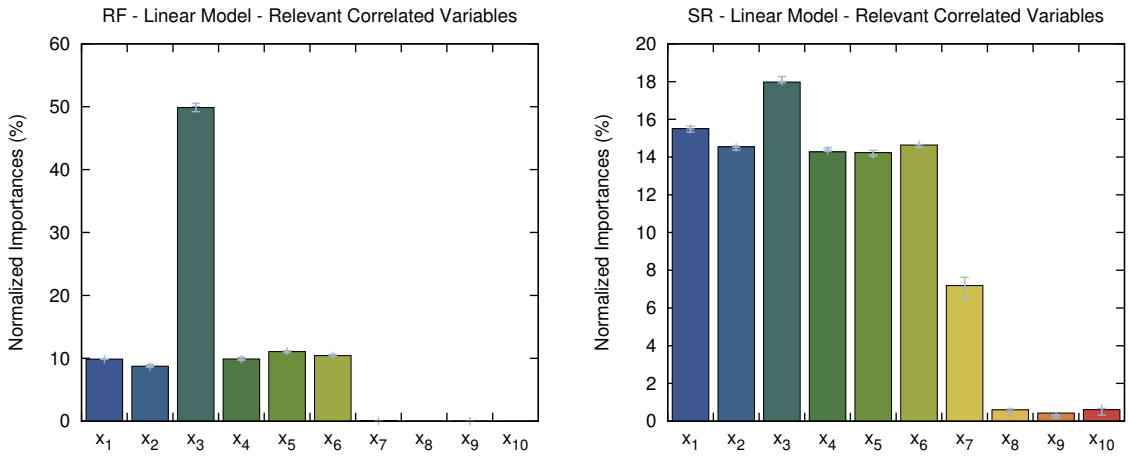


Figure 5.6: Comparing the results of RF and SR on a 500 point data set constructed according to Equation 5.4 and 5.5. Variable x_3 is severely overestimated by RF, while x_7 is not recognized as important. In contrast, importances obtained using SR overestimate x_7, and variables with the same coefficient in the generating function have approximately the same importance score.

5.1.1.6 Noisy Variables

When performing real life experiments it is common for measurements to be corrupted by noise, such that the true input-output behavior is obfuscated. We mimic such situation by first creating a 500 point data set by sampling the function given by Equation 5.1. After computing the exact values for y, we create noisy variables $\{\widetilde{x_1}, \ldots, \widetilde{x_{10}}\}$ by adding noise to the true input variables as follows:

$$\widetilde{x_i} = 0.5 * x_i + 0.5 * Noise \tag{5.6}$$

Where $Noise$ is uniformly random in the interval $[0,\ 1]$.

The results are shown in Figure 5.7. Perhaps surprisingly, the results of RF do not differ significantly from the earlier noise-free experiment in Section 5.1.1.1. However, this is explained by the experiments in Section 5.1.1.4. They showed that for finding splits RF does not really distinguish between a true variable and a variable to which it correlates. This results in a much coarser interpretation of the data, where only the general trend will matter.

In contrast, the noisy variables pose a challenge to SR, which tries to find an optimal global model. However, his relationship is obfuscated due to the added noise. While the variables with highest coefficients in the generating function are still easily identified, the variable with lowest coefficient (x_4) is not distinguishable from noise. This could be an indication there is room for evolving better models. However, we remark that even the generating function would have a significant error, and many other models will have a comparable error. This results in a very high diversity in the population, where models also using irrelevant variables can attain a relatively high fitness.

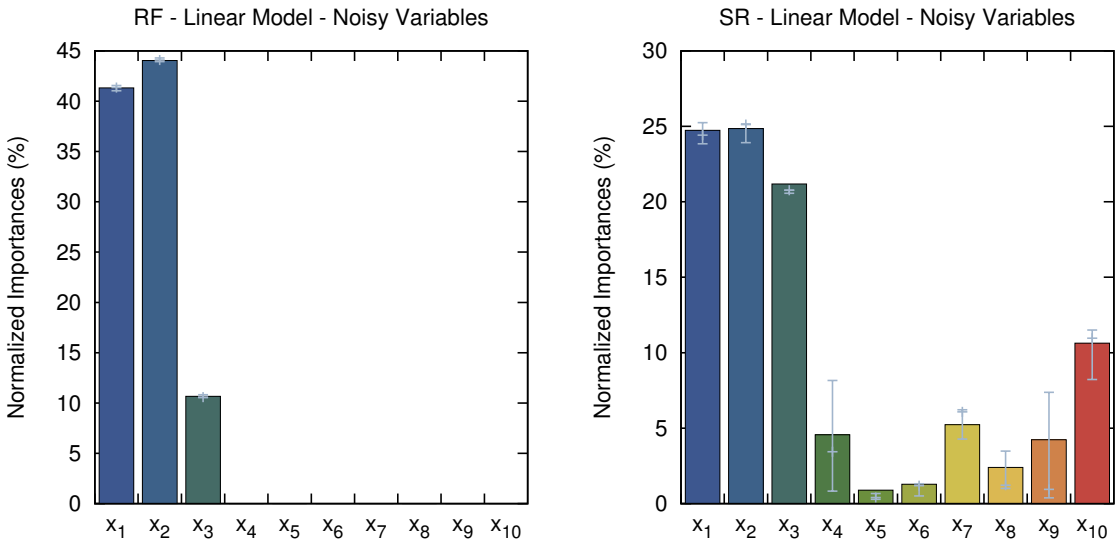


Figure 5.7: Comparing results for RF and SR using a noisy data set for samples from the function given by Equation 5.1. While RF yields the same results as the noise-free experiment in Section 5.1.1.1, SR is looking for a model obfuscated by the noise. The variable with lowest coefficient is not distinguishable from noise.

5.1.2 Newton's Universal Gravitation Law

Now the differences between RF and SR have been shown on data from a simple linear function, we investigate this differences further using Newton's universal gravitation law. Recall from Section 4.2.6.1:

$$F = G\frac{m_1 m_2}{r^2} \tag{5.7}$$

Where F is the force attracting two masses m_1 and m_2 over a distance r. The constant G is the gravitational constant:

$$G = 6.67428 \times 10^{-11}, \quad \text{with unit } N(m/kg)^2 \tag{5.8}$$

The data was generated uniformly in ranges:

$$0 \leq m_1 \leq 1, \quad 0 \leq m_2 \leq 1, \quad 1 \leq r \leq 2 \tag{5.9}$$

As a preliminary test we determine VI using the same data set of 50 points as in Section 4.2.6.1, containing only the three relevant variables. Results are shown in Figure 5.8. Observe that RF determines a much lower importance for r than SR.

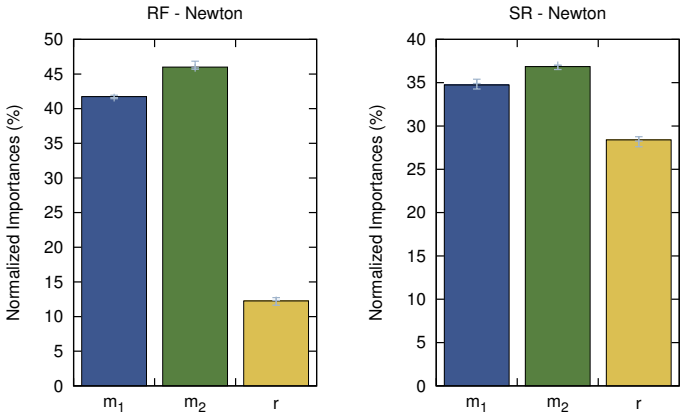


Figure 5.8: Comparing results from RF and SR on the 50 point Newton data set with only relevant variables.

5.1.2.1 Spurious Variables

Analogously to our experiments on a linear model we add many irrelevant variables. Fifty spurious variables are added sampled uniformly in the interval [0, 1], they do not contribute to the response in any way. The obtained importances are shown in Figure 5.9. Remark that only a selection of variables is shown, for RF the 14 most important variables are shown. Variables not shown on Figure 5.9 all have very low importance scores. Results for SR identify that variables m_1, m_2 and r are surely more important than all others. From VI as computed by RF the importance of variable r is no longer clear, and surprisingly some noise variables consistently score higher than r.

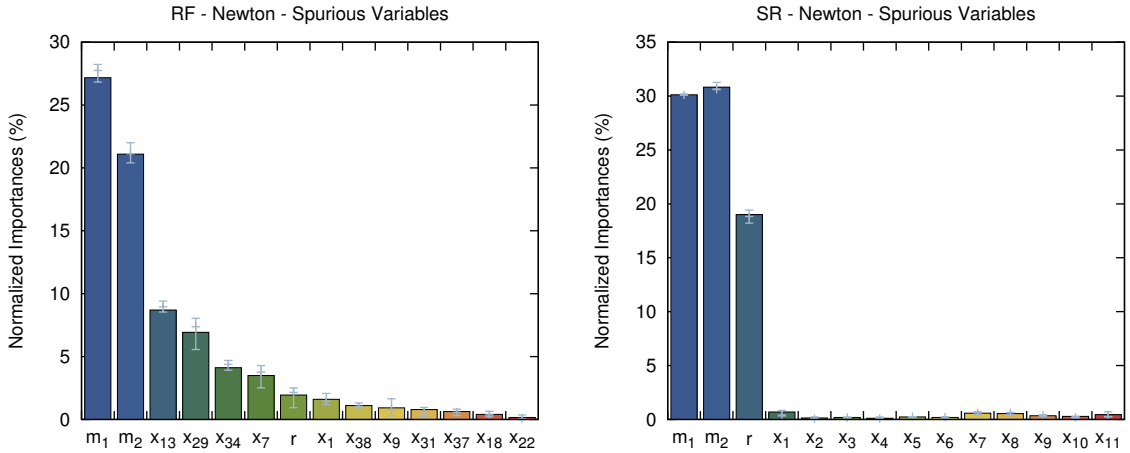


Figure 5.9: Comparing results from RF and SR on the Newton data set with 50 spurious variables. The importance score from SR correctly identifies the driving variables, while RF scores some variables that are not relevant to the response higher than the driving variable r.

Different parameter settings of leaf size, candidate set size did not improve the results for RF. However, increasing the data set size from 50 to 500 did improve the results, as shown in Figure 5.10. In this case RF does identify only the driving variables, and allocates zero

importance to any spurious variable. This indicates that RF is not suited for very small data sets.

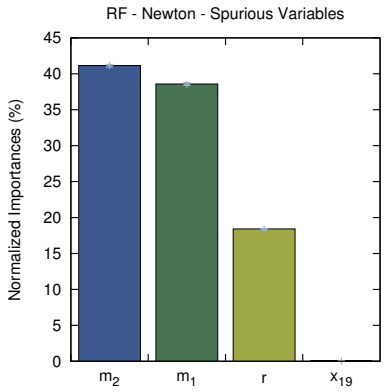


Figure 5.10: Variable importances obtained with RF on the Newton data set with 50 spurious variables, using 500 sample points. Remark that only the variables relevant to the response obtain a significant importance score. All variables that are not shown have an importance equal to or less than variable x_{19}.

5.1.2.2 Imbalanced Data Set

Analogously to the experiments on a linear model we examine the impact of using an imbalanced data set. We used a data set of 250 points with 10 spurious variables, where variable m_1 is oversampled in the interval $[0.45,\ 0.55]$.

The results are shown in Figure 5.11 and are consistent with results from Section 5.1.1.2. The data distribution makes the importance of m_1 appear much lower for RF, however, the driving variables are still clearly distinguishable from the spurious variables. With SR the importance of m_1 is lower than in the experiment from Section 5.1.2.1, but the drop in importance is not as dramatic as observed for RF.

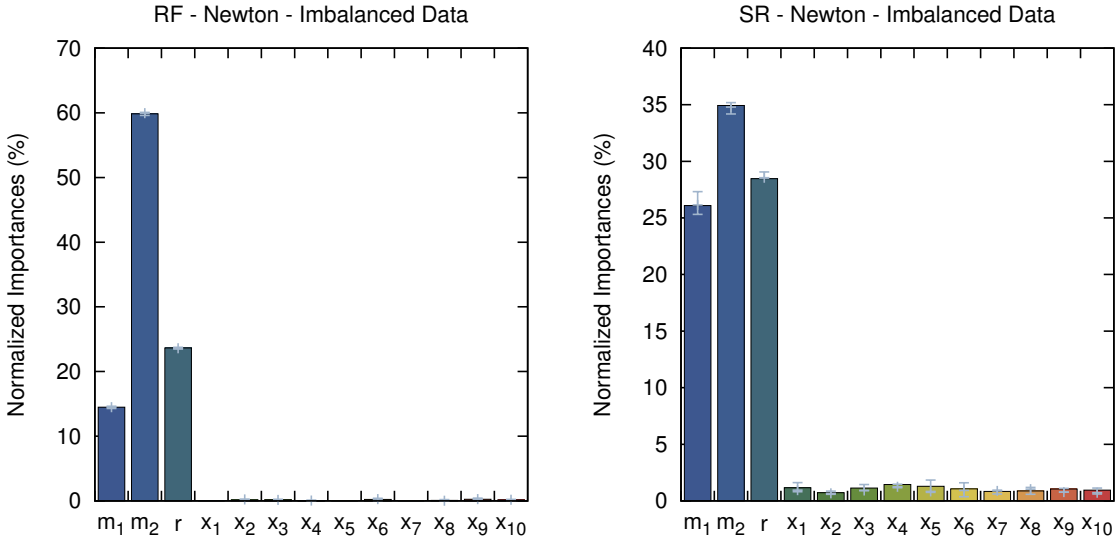


Figure 5.11: Comparing results for RF and SR using a 250 point data set with 10 spurious variables, sampling Equation 5.7 where m_1 is oversampled in the interval $[0.45,\ 0.55]$. The importance of m_1 is severely underestimated when using RF, whereas the loss of importance is not that pronounced when using SR.

5.1.2.3 Noisy Variables

Analogously to the experiments on a linear model we created a noisy data set of 200 points with 10 spurious variables, where noisy variables $\widetilde{m_1}$, $\widetilde{m_2}$ and $\widetilde{r}$ are created as follows:

$$\begin{aligned} \widetilde{m_1} &= 0.7 * m_1 + 0.3 * Noise \\ \widetilde{m_2} &= 0.7 * m_2 + 0.3 * Noise \\ \widetilde{r} \quad &= 0.7 * r \quad + 0.3 * Noise \end{aligned} \tag{5.10}$$

Where $Noise$ is uniformly distributed in the interval $[0,\ 1]$.

The results are shown in Figure 5.12. Similar to the experiment in Section 5.1.1.6, results do not differ significantly compared with noise-free data for RF. We observe that SR allocates more importance to spurious variables since the overall quality of the population is lower, however, the driving variables still score the highest.

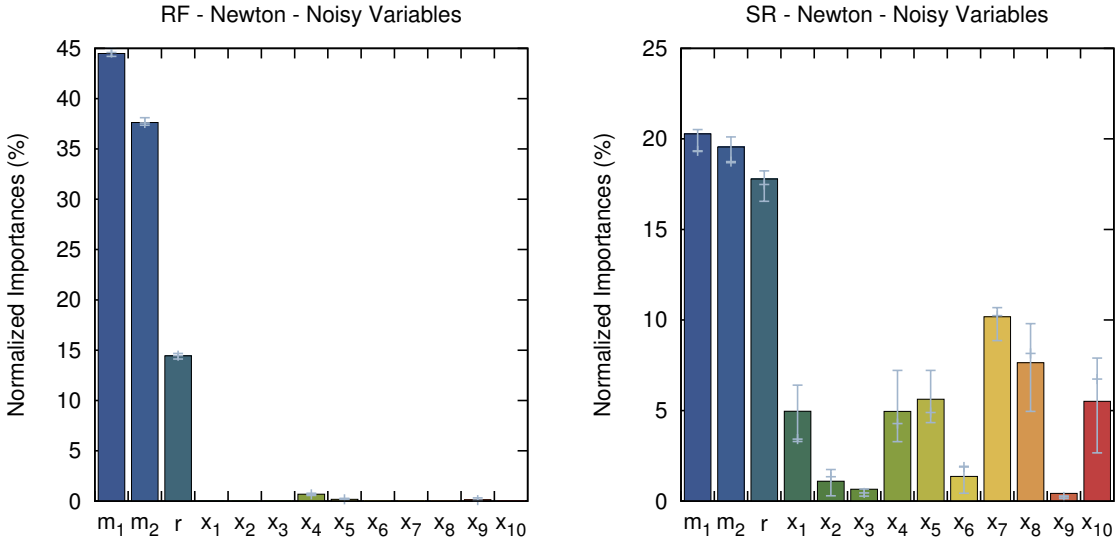


Figure 5.12: Comparison of results for RF and SR using a 200 point Newton data set, with 10 spurious variables and noise added to the driving variables. The importances of RF do not differ significantly from the noise-free experiment. Model building is more difficult for SR resulting in more spurious variables being present in the population, resulting in a higher importance score for spurious variables.

5.1.2.4 Correlated Irrelevant Variables

Here we add 30 variables correlated with the driving variables. A data set of 150 points was used, where the correlated variables were generated as follows:

$$\begin{aligned} x_i &= 0.7 * m_1 + 0.3 * Noise, \quad \text{for } i = 1, \ldots, 10 \\ x_j &= 0.7 * m_2 + 0.3 * Noise, \quad \text{for } j = 11, \ldots, 20 \\ x_k &= 0.7 * r \quad + 0.3 * Noise, \quad \text{for } k = 21, \ldots, 30 \end{aligned} \tag{5.11}$$

Where $Noise$ is uniformly random in the interval $[0,\ 1]$.

Results are shown in Figure 5.13. Similar to our observations on a linear model the importance of relevant variables is less pronounced for scores obtained with RF, since correlated

variables will receive high scores as well. In this experiment some noisy variables even obtain a score higher that the true variable. With SR the true variables are identified, even though some correlated variables also score relatively high.

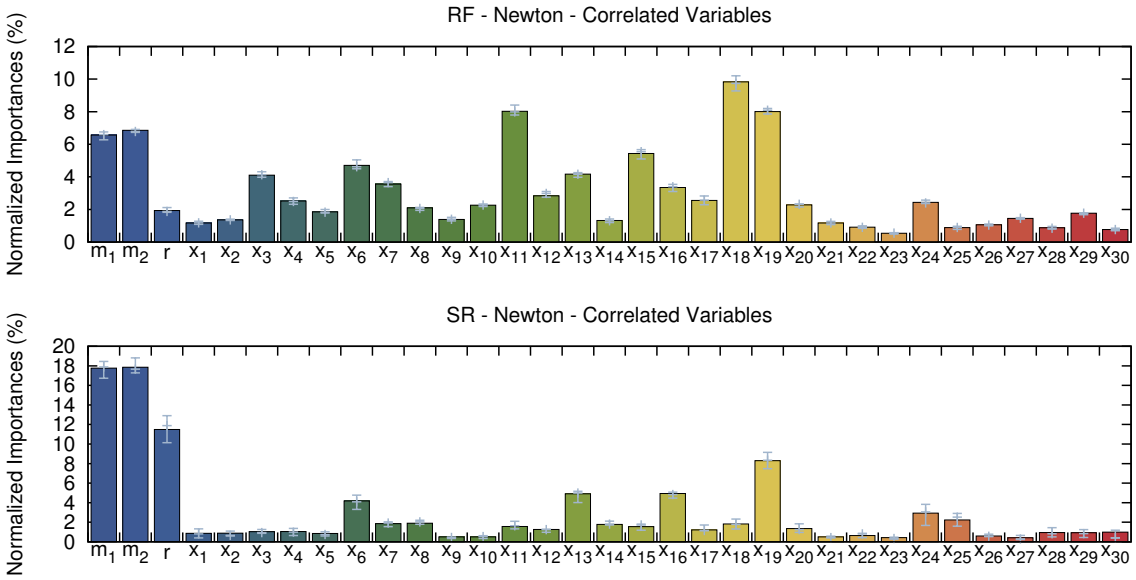


Figure 5.13: Comparing results from RF and SR on the Newton data set with 30 variables correlated with the driving variables. Importances obtained by RF do not allow for clear identification of the driving variables m_1, m_2 and r, while this is not a problem with SR.

5.1.3 Unwrapped Ball Function

We found the five dimensional unwrapped ball function another interesting non-linear problem. This function was introduced in [116] and is given by:

$$y = \frac{10}{5 + \sum_{i=1}^{5}(x_i - 3)^2} \tag{5.12}$$

The variable importances of this function are of interest, since the expression implies that all variables should be equally important. This is in a sense a non-linear version of our experiment in Section 5.1.1.3.

We created a data set of 1024 points where points were uniformly sampled in ranges:

$$0.05 \leq x_i \leq 6.05, \quad \text{for } i \in \{1, \ldots, 5\} \tag{5.13}$$

Results are shown in Figure 5.14. We observe that RF and SR agree on the order of importance. Although the variables should be equally important judging by the generating formula, neither RF nor SR score all variables equally. We attribute this to random sampling fluctuations, and this will be confirmed in Section 5.3.3.

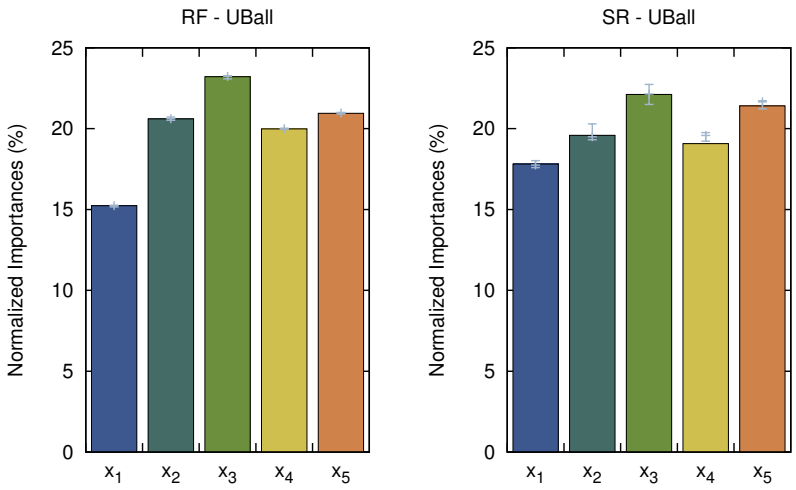


Figure 5.14: Comparing results from RF and SR on the five dimensional unwrapped ball data set. We observe that neither RF nor SR score all variables equally, but they agree on the order of importance. For this function this indicates that the data sampling is not perfectly uniform, as will be demonstrated in Section 5.3.3.

5.2 More on Random Forest

In Section 3.3 we mentioned several measures of VI in RF, which we briefly recall here:

Shuffle – The change in prediction error introduced by perturbing the out-of-bag data.

Gini – The accumulation of the splitting criterion for a variable, giving an impression of how well a variable can split the data.

Splits – The number of splits on a variable is not a good measure of importance on its own, but will be instrumental to our discussion.

We revisit some experiments from the comparative study in Section 5.1 and explain in the behavior of the VI in detail.

5.2.1 Linear Model

Recall that this experiment was determining VI using a uniformly sampled data set for the generating function given by Equation 5.1, shown again here:

$$y = 10x_1 + 10x_2 + 5x_3 + 1x_4 + 0x_5 + \ldots + 0x_{10} \tag{5.14}$$

The Gini measure and number of splits is shown in Figure 5.15. Here we observe that variables with a lower coefficient in the generating function will have a lower Gini index. The effect is that variable x_4 will only be the preferred variable to split on if all other candidates are spurious variables, since the variables x_1, x_2, and x_3 will always outperform x_4. Remark that x_4 is not split on more than an irrelevant variable.

We also observe that variable x_3 has a low importance score even though it has more splits compared to an irrelevant variable. Using a similar argument, splits on x_3 will only happen if

neither x_1 or x_2 are in the candidate set. Except for nodes lower in the tree, closer to leaf nodes. However, such splits do not alter the prediction path much when shuffling the data in that variable, explaining the lower importance given by the shuffle techniques.

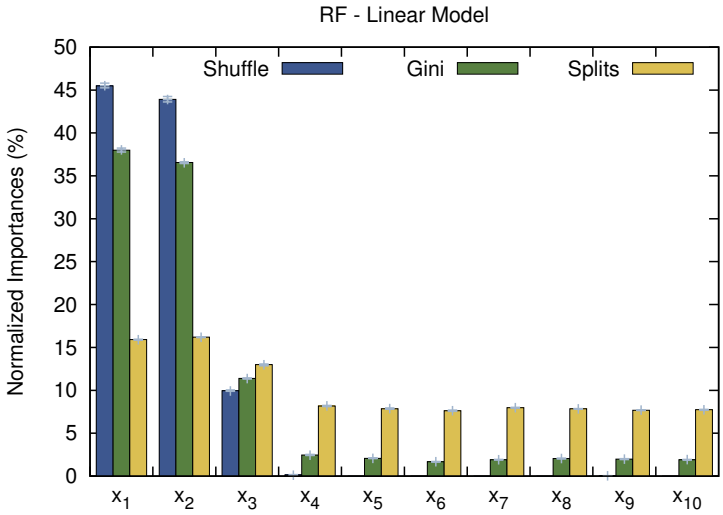


Figure 5.15: Other measures explaining the behavior of RF on a linear model given by Equation 5.14. Observe that variables with a lower coefficient in the generating function will be split less. Variable x_3 will only be chosen as a decision variable in the lower areas of the tree where the influence on the prediction path is lower, resulting in a lower shuffle score.

5.2.1.1 Imbalanced Data Set

Recall that this experiment used the same generating function as the previous section, but used an imbalanced data set where variable x_1 was sampled with much higher density in a small interval.

The Gini measure and number of splits is shown in Figure 5.16. We observe that splits on x_1 are far less common than when using a uniformly sampled data set. Although x_1 is still being split more that a spurious variable, implying that the prediction model is not per se faulty.

Consider the values of variable x_1. After the cluster is identified with a few initial splits, little information is gained by splitting x_1 further. This will allow the other variable with high coefficient (x_2) to always provide splits superior to any other variable, as indicated by its Gini score.

By the design of the shuffle technique, x_1 will not achieve a high importance score. Since more data points have an x_1 value in the interval [0, 0.1], any perturbed data point is highly likely to be assigned a value from this denser area. And if the point already had a value for x_1 within this interval, little change in the response will be observed resulting in a low importance score. We refer to a later experiment in Section 5.3.1.1 where we confirm this behavior using a perfect model.

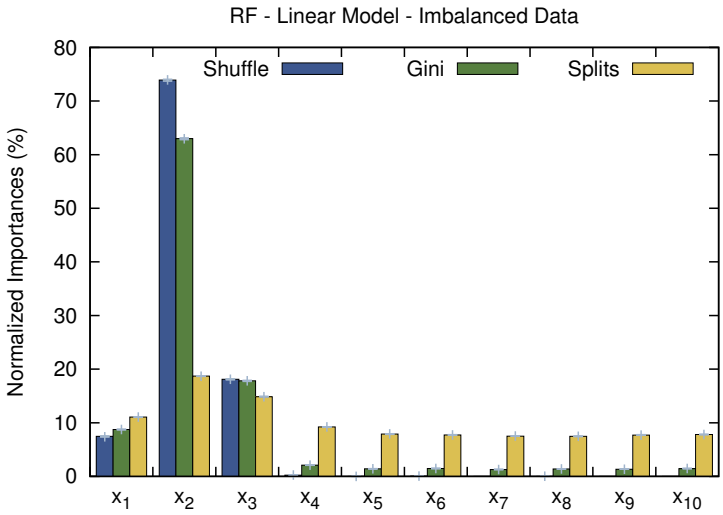


Figure 5.16: Other measures explaining the behavior of RF on a linear model given by Equation 5.14, using an imbalanced data set. Remark that variable x_1 no longer provides good splits despite its high coefficient in the generating function, as indicated by its low Gini score. The lower number of splits on x_1 lower the influence on the predictive path, resulting in a lower shuffle score.

5.2.1.2 Relatively Weak Variables

Recall that this experiment used the generating function given by:

$$y = 10x_1 + 10x_2 \dots 10x_8 + 1x_9 + 0x_{10} \tag{5.15}$$

The Gini measure and number of splits is shown in Figure 5.17. We observe that some variables deliver better splits than others, even though their coefficient in the generating function is the same. We attribute this to small structural defects in the random sampling of the input space.

The decision variable is chosen to be the variable that delivers the best split, irrespectively of how much the split is better compared to other candidates. This allows small structural defects to consistently favor the same variables early in the model construction, even though the difference with other variables is minimal. A variable consistently obtaining early splits influences the prediction path of perturbed points more, resulting in a higher shuffle score.

5.2.1.3 Correlated Irrelevant Variables

Recall that this experiment used the same generating function as the previous section, but variables x_{11} through x_{20} where added. These extra variables are correlated with x_1.

The Gini measure and number of splits is shown in Figure 5.18. We observe that splits on x_1 are far less common compared to the previous section. The correlated variables will provide good splits, as indicated by their higher Gini index compared with the irrelevant variables x_5 through x_{10}.

The candidate set size will be 7 in this experiment, which is the default $|d|/3$ value. The correlated variables are numerous, implying that most of the time at least one of them will be

in the candidate set. In case x_1 is not present in the same candidate set, one of the correlated variables has a high probability to be the decision variable. In effect they will compete with variable x_1. Since the correlated variables are numerous and deliver the same information as x_1, variable x_1 will be at a disadvantage.

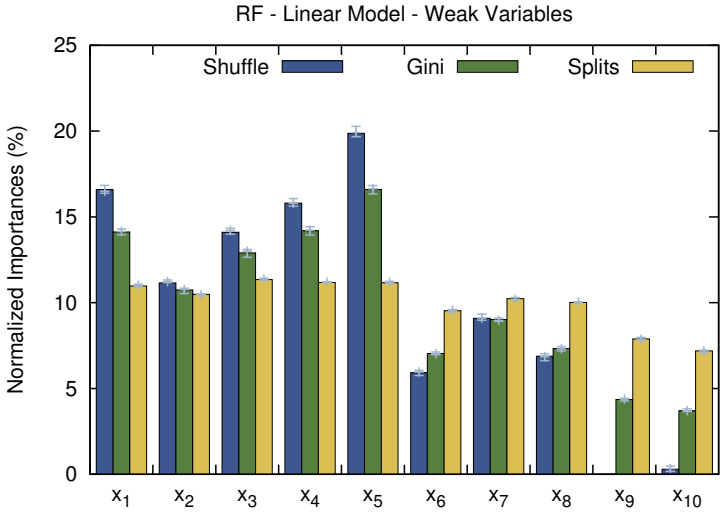


Figure 5.17: Other measures explaining the behavior of RF on a linear model given by Equation 5.15, where the coefficients of x_1 through x_8 are the same. Even though the number of splits is not that different, the Gini index is. This implies that some variables will be preferred, causing them to split the data early in the tree – close to the root. This will result in a greater difference in the prediction path, resulting in a higher shuffle score.

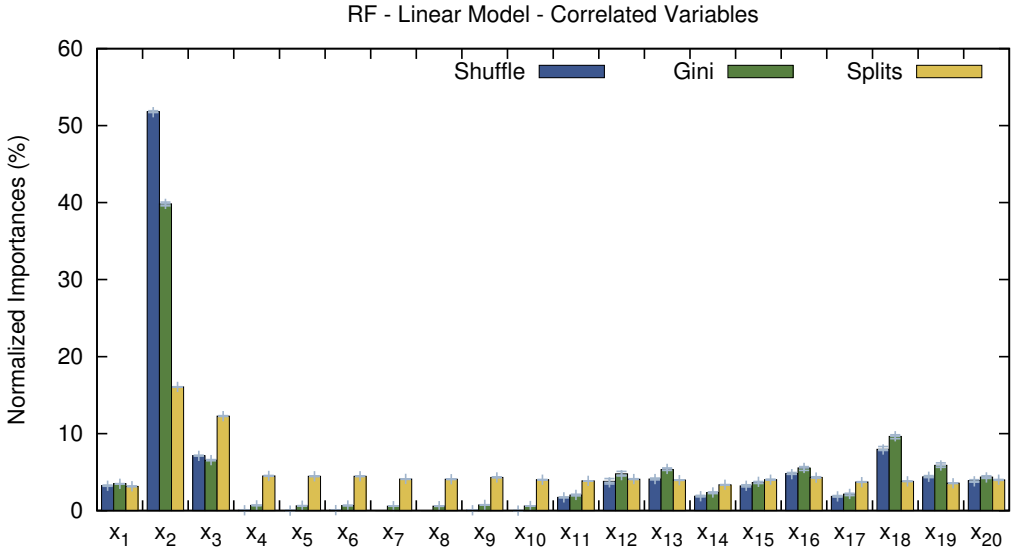


Figure 5.18: Other measures explaining the behavior of RF on a linear model using correlated variables. The correlated variables will compete with variable x_1 for the same splits. Since the correlated variables are numerous and deliver the same information as x_1, variable x_1 will be at a disadvantage.

5.2.1.4 Correlated Relevant Variables

Recall that this experiment used the generating function given by:

$$y = 10x_1 + \ldots + 10x_6 + 1x_7 + 0x_8 + \ldots + 0x_{10} \tag{5.16}$$

Where variables x_2, x_4, x_5, and x_6 are correlated with x_3.

The Gini measure and number of splits is shown in Figure 5.19. We observe that the number of splits on x_3 does not differ from the splits on a variable with equal coefficient (x_1) in the generating function. This implies the difference in importance score is due to other factors.

We observe that the Gini score for x_3 is much higher than any other variable. This is explained by the correlation of the other variables with x_3. This will in effect inflate its coefficient, resulting in the information gain criterion always preferring x_3 and assigning a high Gini score.

The prediction path of a perturbed data point is interesting. If the new value of x_3 causes a different decision to be made at any node where x_3 is the decision variable, the new prediction will differ significantly. Since the correlated variables are similar to x_3 in terms of input-output behavior in the original data, a decision on x_3 describes the influence of all those variables combined. So, a different prediction path will predict a very different area of the input space resulting in a high shuffle score.

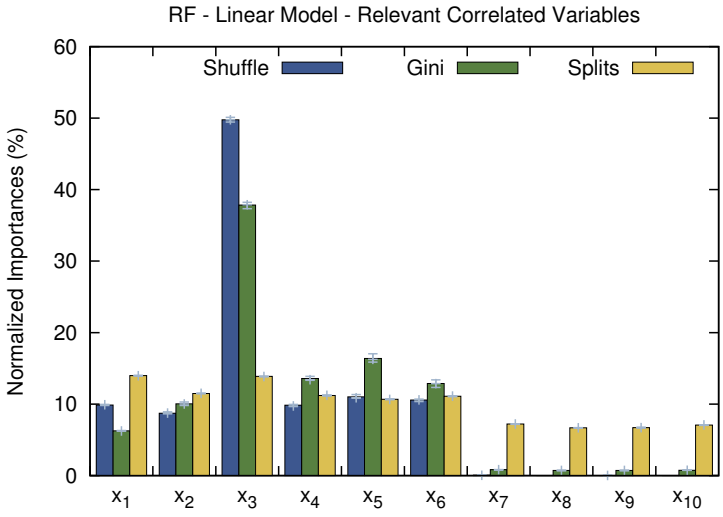


Figure 5.19: Other measures explaining the behavior of RF on a linear model given by Equation 5.16, where other relevant variables where correlated with x_3. Remark that x_3 outperforms any other variable even though its coefficient in the generating function is the same as variables x_1 through x_6.

5.2.2 Newton's Universal Gravitation Law

Observations similar to the previous sections can be made for the Newton problem, so results for correlated variables, imbalanced data are not shown. Recall the generating function:

$$F = G\frac{m_1 m_2}{r^2} \tag{5.17}$$

Where F is the force attracting two masses m_1 and m_2 over a distance r. The constant G is the gravitational constant:

$$G = 6.67428 \times 10^{-11}, \quad \text{with unit } N(m/kg)^2 \tag{5.18}$$

The Gini measure and number of splits is shown in Figure 5.20, obtained with the same data set as Figure 5.8. We observe the number of splits is equal, which is to be expected since the default candidate size is $|d|/3$, hence one in this experiment.

We speculate that variable r provides less opportune splits due to its limited domain. The combination of m_1 and m_2 have the potential to cause a much more change to the response compared to r^2, due to the difference in sampling domain.

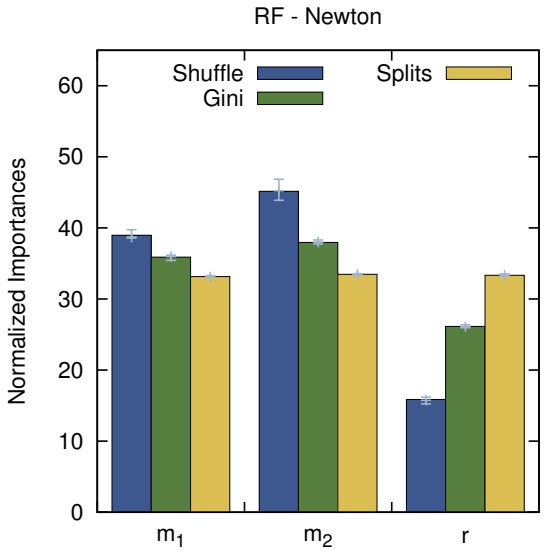


Figure 5.20: Other measures explaining the behavior of RF on the Newton problem. The difference in sampling domain of r compared with m_1 and m_2 prevents r from being identified as at least as important.

5.3 More on Expression Trees

In Section 4.3.2 we mentioned several measures that could help identifying important variables when applied to algebraic expressions. This section will further investigate the impact of each measure on several problems. We briefly recall the measures applicable on expressions:

Derive – The accumulated partial derivative over the available data points.

Mean – The prediction error made by substituting a variable with its mean.

Eliminate – The prediction error made when eliminating a variable from the expression.

Shuffle – The prediction error when shuffling the data in one dimension.

We revisit some experiments from the comparative study in Section 5.1 and find out how these measures perform.

5.3.1 Linear Model

Recall that this experiment was determining VI using a uniformly sampled data set for the generating function given by Equation 5.1, shown again here:

$$y = 10x_1 + 10x_2 + 5x_3 + 1x_4 + 0x_5 + \ldots + 0x_{10} \tag{5.19}$$

We first use a uniformly sampled data set, results for all measures are shown in Figure 5.21. We observe all measures agree on the relative importances, indicating that further experiments are needed to uncover their differences.

While the results for 'Shuffle' in Figure 5.21 correspond to results obtained with RF, we note that RF uses the SSE as error function. In order for the importances to be comparable to the coefficients of the generating function, this error function should be the RMSE as shown in Figure 5.22. We observe that using the RMSE in RF would be undesirable since the importance of an irrelevant variable with a small score would be inflated. The square root affects high importance scores more than low importance scores, such that variables with a higher importance score would be harder to distinguish from variables with a low importance score. We note that the square root is order preserving, thus selecting k variables with highest importance would yield the same result, irrespectively of whether the SSE or RMSE was used. In order for the 'Shuffle' to be comparable with earlier experiments, we opt to use the SSE in the rest of this section.

5.3.1.1 Imbalanced Data

Recall that this experiment used the same generating function as the previous section, but used an imbalanced data set where variable x_1 was sampled with much higher density in a small interval.

Results for all measures are shown in Figure 5.23. Remark that the 'Derive' measure is not influenced by the data distribution, since the partial derivative of this linear model will be a constant exactly corresponding to the coefficient in the generating function. All other measures are influenced by the data distribution.

We observe that the 'Shuffle' and 'Mean' method agree on the importance but underestimate variable x_1. In the previous experiments with RF on this data we already observed this, but

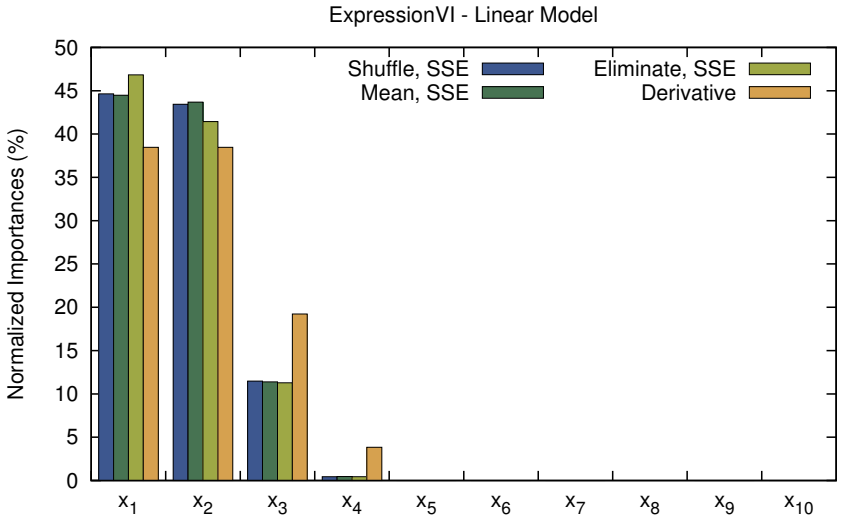


Figure 5.21: Comparing the results for different expression based metrics on a linear model given by Equation 5.19, using uniformly sampled data. We observe all measures agree on the relative importances, indicating that they all could be potentially interesting. However we note that the derivative is expected to be the optimal importance for this problem, while the other measures differ. This is a result of using the SSE error function. Using the RMSE would be a better match with the coefficients, as shown in Figure 5.22.

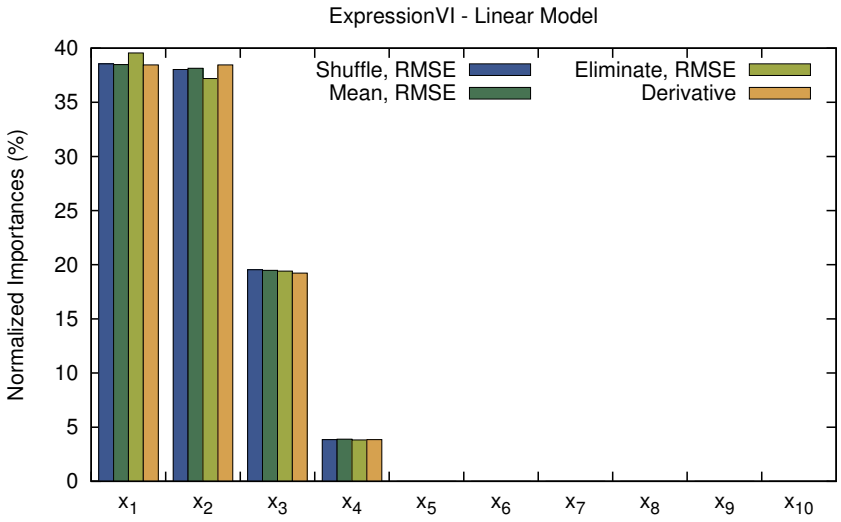


Figure 5.22: Using the same data as for Figure 5.21, but now using the RMSE as error function. We observe that the importances are all in agreement with the 'Derive' measure, which would be the optimal importance measure for this problem. In the text we provide reasons not to use the RMSE in conjunction with RF.

the use of the expression makes all the difference. By using the generating function directly, it is as if we use a perfect model. Hence we can be sure that this behavior is due to the definition of the 'Shuffle' procedure and not a modeling artifact introduced by RF.

The 'Eliminate' technique will overestimate x_1, since eliminating x_1 in this expression according to the rules introduced in Section 4.3.2.3 would be equivalent to substituting the value one. Considering the mean of x_1 is very low due to the high density in the interval $[0,\ 0.1]$, substituting by one introduces a larger error compared with the other variables whose mean is 0.5 since they are uniformly sampled in $[0,\ 1]$.

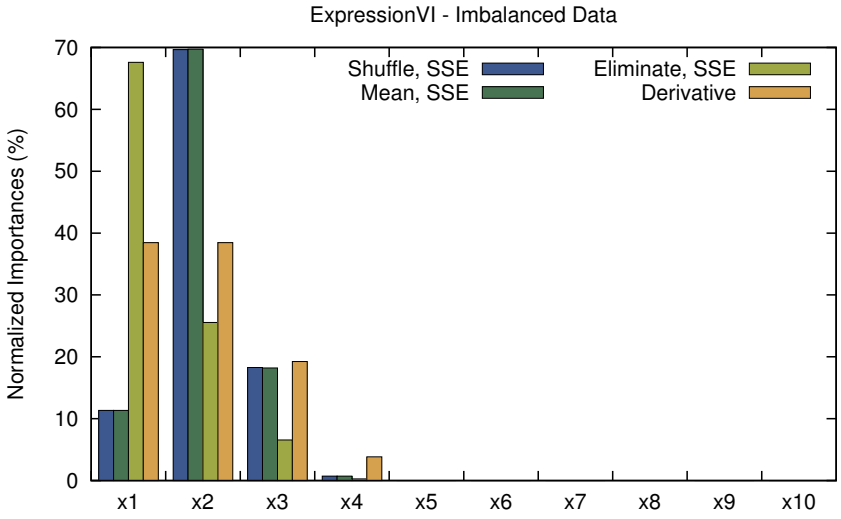


Figure 5.23: Comparing the results for different expression based metrics on a linear model given by Equation 5.19, using an imbalanced data set. We observe that 'Shuffle' yields the same results as when used in conjunction with RF, indicating that the underestimation of x_1 is not a modeling artifact of RF.

5.3.1.2 Correlated Relevant Variables

Recall that this experiment used the generating function given by:

$$y = 10x_1 + \ldots + 10x_6 + 1x_7 + 0x_8 + \ldots + 0x_{10} \tag{5.20}$$

Where variables x_2, x_4, x_5, and x_6 are correlated with x_3.

Results for all measures are shown in Figure 5.24. Remark that the 'Derive' measure is not influenced by variable correlations, since the partial derivative of this linear model will be a constant exactly corresponding to the coefficient in the generating function.

We observe again that the 'Shuffle' and 'Mean' measures agree on the importances. However, in contrast to experiments with RF in Section 5.2.1.4, we observe that the 'Shuffle' scores now make more sense. Here the importance of x_1 and x_3 is equal, as expected by their equal coefficients in the generating function. While all variables x_1 through x_6 have the same coefficient, the correlated variables can be considered less important because the information they add is not independent.

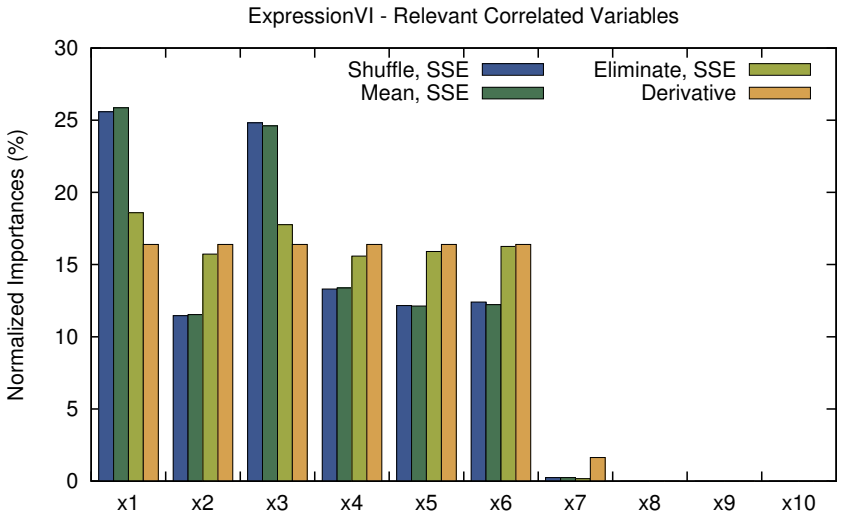


Figure 5.24: Comparing the results for different expression based metrics on a linear model given by Equation 5.20, using an imbalanced data set. We observe that 'Shuffle' yields the same results as when used in conjunction with RF, indicating that the underestimation of x_1 is not a modeling artifact of RF.

5.3.2 Newton's Universal Gravitation Law

Observations similar to the previous sections can be made for the Newton problem, so results for correlated variables, imbalanced data are not shown. Recall the generating function:

$$F = G\frac{m_1 m_2}{r^2} \tag{5.21}$$

Where F is the force attracting two masses m_1 and m_2 over a distance r. The constant G is the gravitational constant:

$$G = 6.67428 \times 10^{-11}, \quad \text{with unit } N(m/kg)^2 \tag{5.22}$$

Results for all measures are shown in Figure 5.25. We observe again that the 'Shuffle' and 'Mean' measure agree on the importances.

For this problem, eliminating any variable according to the rules as introduced in Section 4.3.2.3 would be equivalent to substituting the value one. For variable r this results a denominator of one, while substituting the mean would result in a denominator of 1/4. Consequently, the elimination will introduce a larger error than substituting the mean, so the 'Eliminate' score is higher than the 'Mean' score.

5.3.3 Unwrapped Ball Function

Recall that this experiment used the generating function given by:

$$y = \frac{10}{5 + \sum_{i=1}^{5}(x_i - 3)^2} \tag{5.23}$$

And that variable importances are expected to be equal. In our earlier experiment in Section 5.1.3 we observed that the importance score was not equal for all variables. We will first use

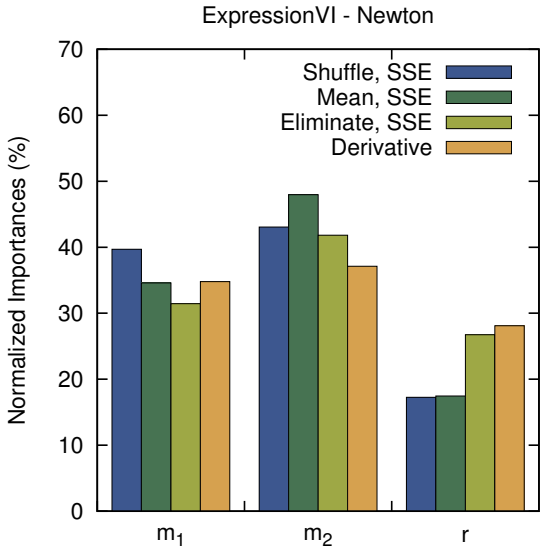


Figure 5.25: Comparing the results for different expression based metrics on the Newton problem. We observe that 'Shuffle' yields the same results as when used in conjunction with RF, indicating that the underestimation of r is not a modeling artifact of RF. The lower score of 'Derive' also indicates the lower score of r is to be expected for these input ranges.

the same data set and compute expression based importances. Results for all measures are shown in Figure 5.26. Even with the exact expression, the fluctuations in score is still present.

To show that the data distribution is influential, we repeat the experiment with a data set sampling on an equidistant grid. Results for all measures are shown in Figure 5.27. We observe that all measures report exactly the same importances. This confirms that the fluctuations in importance were due to the sampling distribution and not the underlying model.

5.4 Discussion

From experiments on expression trees we identified that replacing a variable by its mean, is equivalent with the shuffling technique. However, in RF replacing a variable by its mean would be too restrictive. This would effectively disable some region in a decision tree, since the outcome of any comparison against the decision value for the variable under study will be known a priori. By using the shuffling method, the comparisons can still differ for another permutation of values. Moreover this allows the permuted value for that variable to vary per tree, which increases diversity in the prediction.

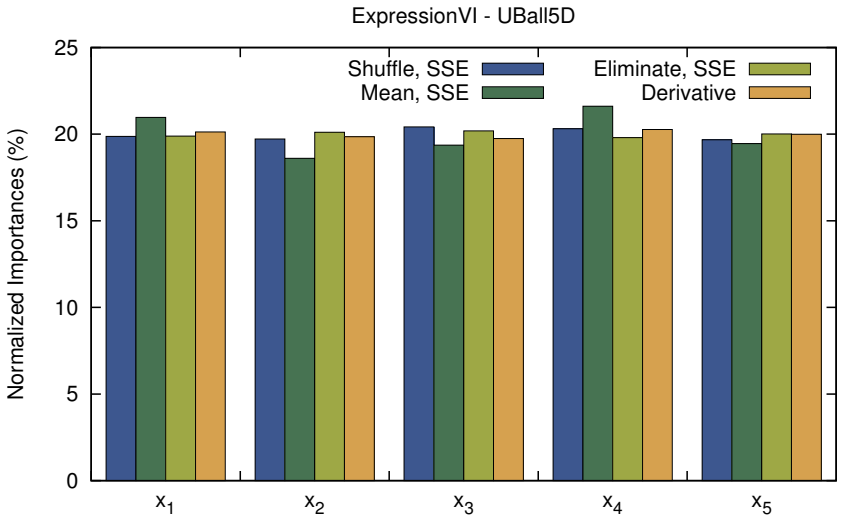


Figure 5.26: Comparing the results for different expression based metrics on the UBall problem using a randomly sampled data set. We observe that fluctuations in the importance are present, although all variables are expected to be equally important. This is due to the randomly sampled data set.

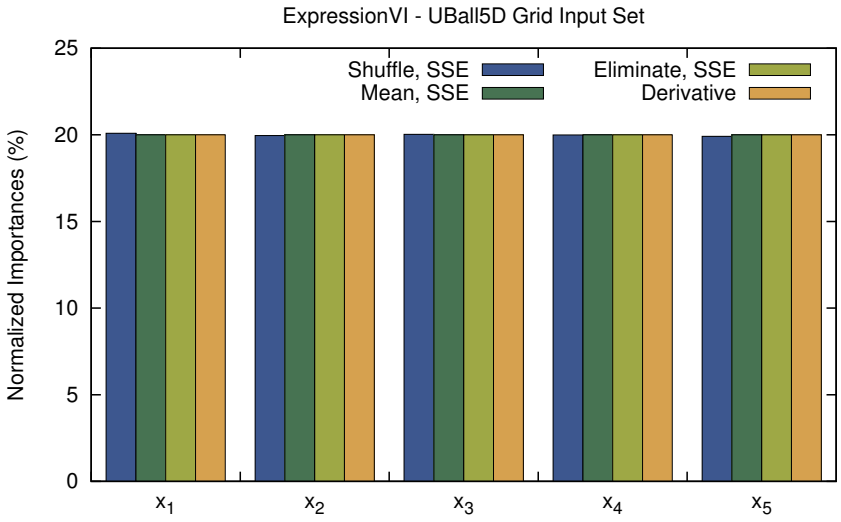


Figure 5.27: Comparing the results for different expression based metrics on the UBall problem using a data set sampled on an equidistant grid. We observe that all importances are equal, confirming that the fluctuations were due to the random sampling.

5.5 Summary

In this chapter we have shown how different importance measures react on different problem configurations, specifically correlated variables and imbalanced data. We have compared symbolic regression (SR) and random forests (RF).

In general SR performed well throughout all tests. We observed that SR favors a *conservative* approach, in other words variables will be rather overestimated than underestimated. This was most notable for the test with a noisy data set, where SR could identify most important variables when using a noisy data set but one variable with low importance could not be distinguished from noise. The noise causes models not resembling the generating function to be assigned a higher fitness. This allows irrelevant variables to survive, and those variables receive a higher importance score. While RF did not identify that variable either, RF is robust to noise in the data set.

We have observed that RF showed several curious properties when handling correlated variables, such that a variable can be either over- or underestimated depending on the correlation structure of the data. In some cases this was explained by the split criterion used for tree construction, in particular relevant correlated variables were favored by the splitting criterion thus inflating its importance score. This property was not present when testing with the same data set and using the exact expression, confirming that this is caused by the tree construction and not by the shuffle technique.

We have shown that RF uses the SSE error function, whereas the RMSE is a better option when comparing importances with coefficients in a linear generating function. However, in general it is preferable to use the SSE in RF, as to clearly distinguish between important and irrelevant variables. We note that the importance ranking remains the same when using either SSE or RMSE.

We have shown that a variable is underestimated by RF when using an imbalanced data set, but that this is by design of the shuffle technique and not the model creation.

Chapter 6

Case Studies

6.1 Human Development Index

Human development is an important and interesting subject, more so because it concerns every human on the planet. Many social, economical and personal factors are combined in a single number known as the Human Development Index (HDI) [83]. This index is an indicator expressing the general quality of life of the human population but not limited to material well-being. It uses three dimensions of development: health, knowledge, income. Health is measured by life expectancy at birth. Knowledge is measured by combining the expected years of schooling for a school-age child in a country today with the mean years of prior schooling for adults aged 25 and older. Income is measured in purchasing power adjusted per capita gross national income. These three dimensions are then combined using the geometric mean.

The HDI is not perfect, but the data collected in the process surely holds information and is subject of active research [84].

In this case study the variable importances obtained by RF and SR are compared, when estimating several parameters from the HDI data set. The initial data set is available at [83] and contains many missing values.

While RF can produce a proximity matrix by which missing values could be filled in, it seemed prudent to pre-process the data set independent of the methods to be compared. The data set was pre-processed as follows: If a value was missing for a specific variable in more than 50 countries, the variable was removed. If a country had more than 30 missing values on the remaining variables, the country was removed. The remaining missing values were replaced by the average value over countries within the same development group (low, medium, high, very high developed country). In addition the variable 'antenatal care' was removed because no data was available for any country in the very high developed group.

The variables 'Total Satisfaction Freedom of Choice', 'Total Purposeful Life' are modeled because any relationship between the development of a country and perceived freedom and purpose would be interesting. For both these variables data was collected through questionnaires were participants were asked whether they were satisfied with their freedom of choice, or whether they find their lives purposeful, respectively. The percentage of the population answering 'yes' to either question is the response variable. Remark that the data set provides both a total percentage and a percentage for the female population to make comparison across genders possible. Only the total percentage is modeled and the female percentage is removed *a priori*, since this variable holds exactly the same information.

The 'GDP per Capita' was also modeled to compare both techniques on a non-linear problem. The Gross Domestic Product (GDP) is the value of all products and services produced within a country in a year. This is divided by the population to make international comparison possible, considering that more people are able to produce more value. While the explicit formula is known in this case and the true variables GDP and population are present in the data set, the difficulty lies in extracting the two relevant variables from the many irrelevant but correlated variables. The variable GNI per Capita is removed a priori, this variables by itself approximates GDP per Capita.

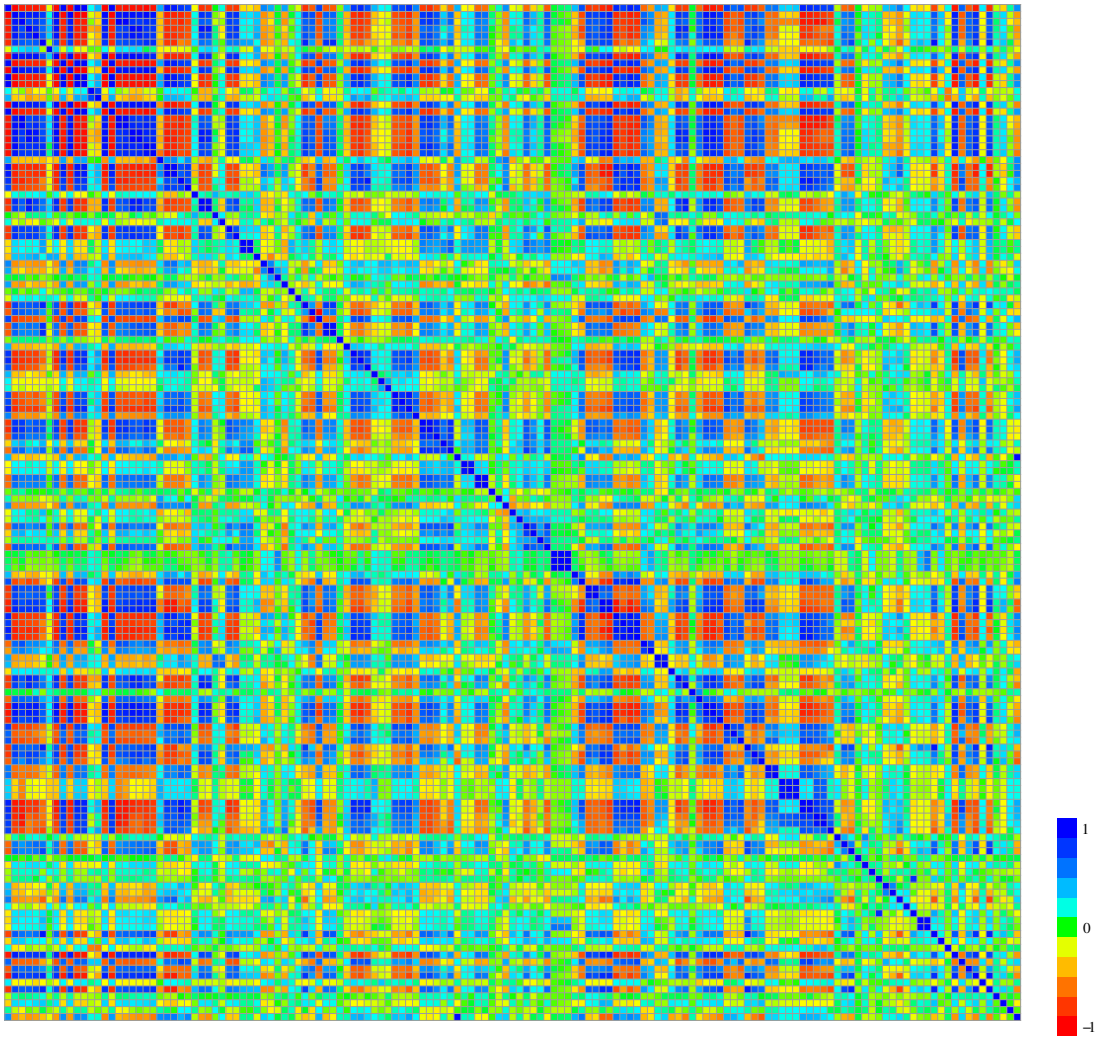


Figure 6.1: The correlation matrix of the HDI data set. Every square represents the correlation between two variables i and j where (i, j) is the index of that square in the matrix. We observe that many variables are correlated.

All modeling was performed on a quad-core i7 machine operating at 3Ghz, with a memory capacity of 12 GB. Other parameters are given in the sections discussing the results for the corresponding response variable.

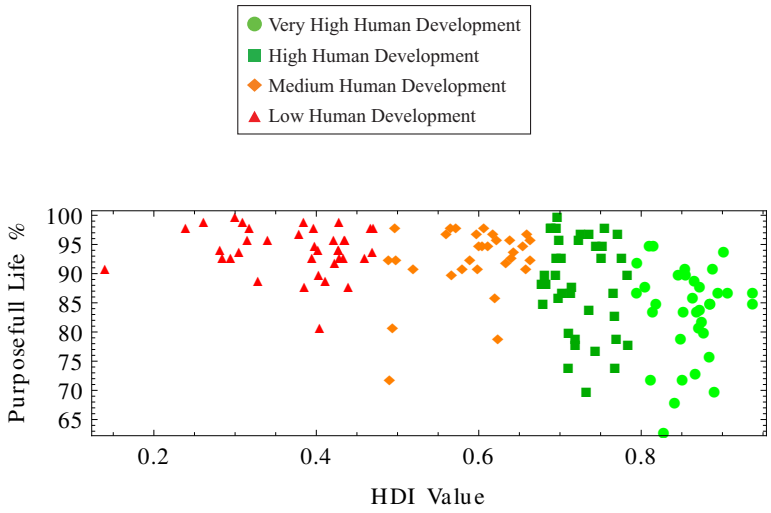


Figure 6.2: The HDI plotted versus Purposeful Life. Countries with a higher HDI score less on average due to higher variance in Purposeful Life.

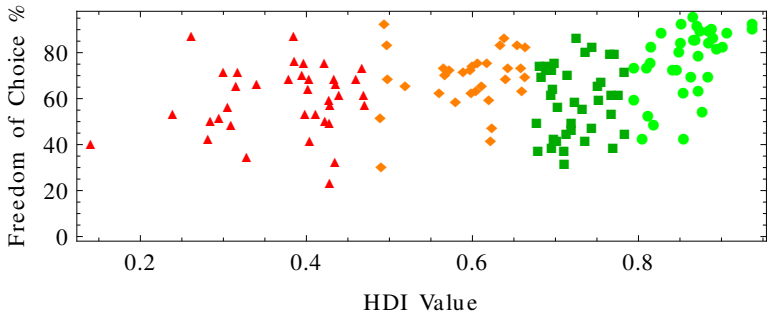


Figure 6.3: The HDI plotted versus Freedom of Choice Satisfaction. The higher developed countries score higher. The variance is large for all development levels.

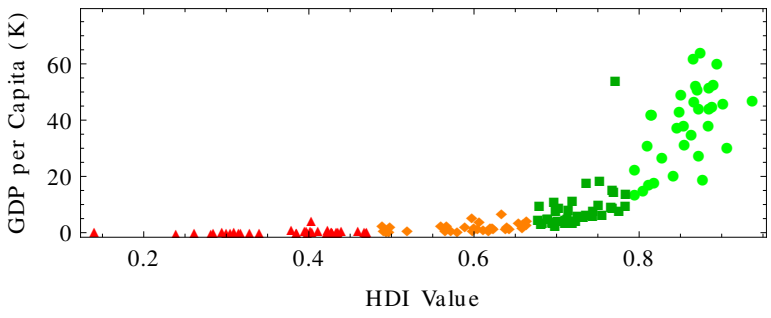


Figure 6.4: The HDI plotted versus the GDP per Capita, which is closely related to the IncomeIndex, one of the three dimensions of the HDI.

RF Parameter	Value	SR Parameter	Value
Number of trees	1000	Independent Runs	4
In-Bag data	2/3	Time Constraint/ Run	1 hour
Leaf Size	5	Accuracy Measure	$1 - R^2$
Candidate set size	$\|d\|/3$	Complexity Measure	sum of sub-tree sizes

Table 6.1: Parameter values used for modeling the Purposeful Life and Freedom of Choice response variables.

6.1.1 Purposeful Life

The relation between country development and Purposeful life is illustrated in Figure 6.2. Remark that countries with a higher HDI exhibit a larger variance in the Purposeful Life.

Importances obtained by SR displayed in Figure 6.6, models of good quality showed a linear relation between variables. The SR variable importances suggest that the most important variables determining whether life is perceived as purposeful are: HospitalBeds, Fertility, Age, MaternityLeave, Undernourishment. We roughly relate these variables to health, children and food. We find these factors not unreasonable in order to enjoy a purposeful life.

Importances obtained by RF are shown in Figure 6.6. It is observed only the most important variables are similar to those identified by SR, and variables with a lower importance are incomparable. Since the underlying model was suggested to be linear by SR the remarks of experiments conducted in the previous section are applicable here.

The parameter values from Table 6.1 were used to model Purposeful Life.

6.1.2 Freedom of Choice

The relation between country development and Satisfaction Freedom of Choice is presented in Figure 6.3. While the higher developed countries score higher, a large variance is observed.

Importances obtained by SR are shown in Figure 6.5, models of good quality showed a linear relation between variables. The SR variable importances suggest that the most important variables determining the satisfaction with ones freedom of choice are: Standard of Living Satisfaction, Healthcare Satisfaction, Respect, Employment, Air and Water Quality. So this one satisfaction variable aggregates information of different aspects in life. Again these variable importances intuitively make sense.

Importances obtained by RF are visualized in Figure 6.5. The most important variables agree with those found by SR with variations in relative importance, but variables with lower importances are different. For example the Income Gini Coefficient has a relatively high importance score, while this variable is not present at all in the top 20 important variables obtained with SR. In the absence of domain expertise this can have a significant impact on a decision making process. This illustrates caution is warranted when performing variable selection, and consulting multiple methods is advised.

The parameter values from Table 6.1 were used to model Freedom of Choice.

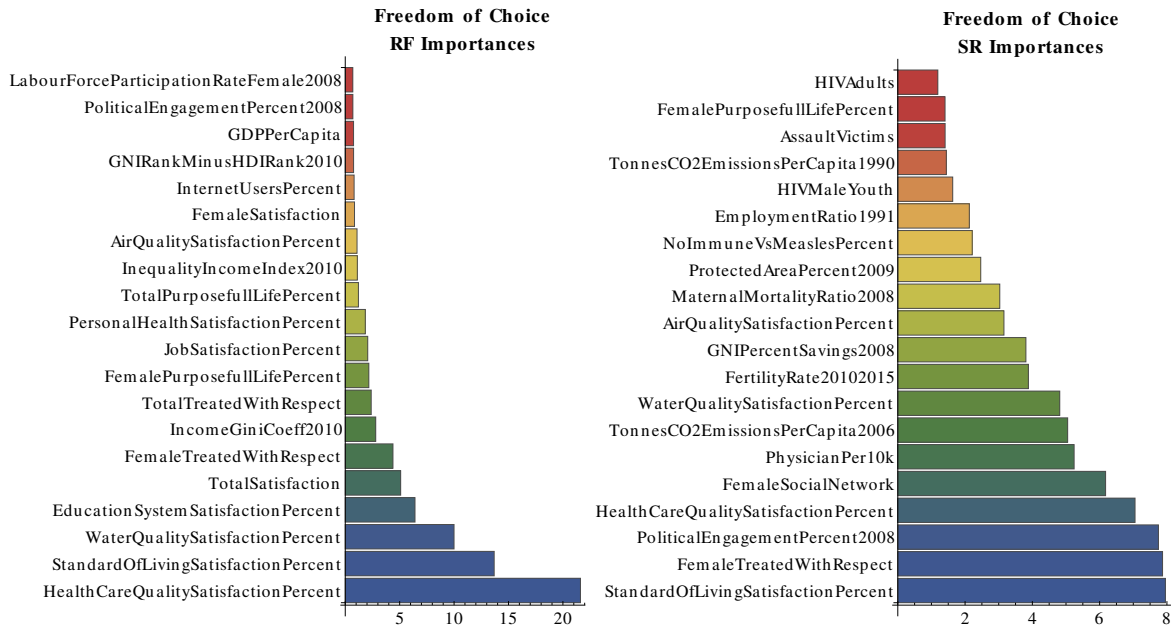


Figure 6.5: Importances obtained with RF and SR modeling Freedom of Choice, using the parameter values given in Table 6.1.

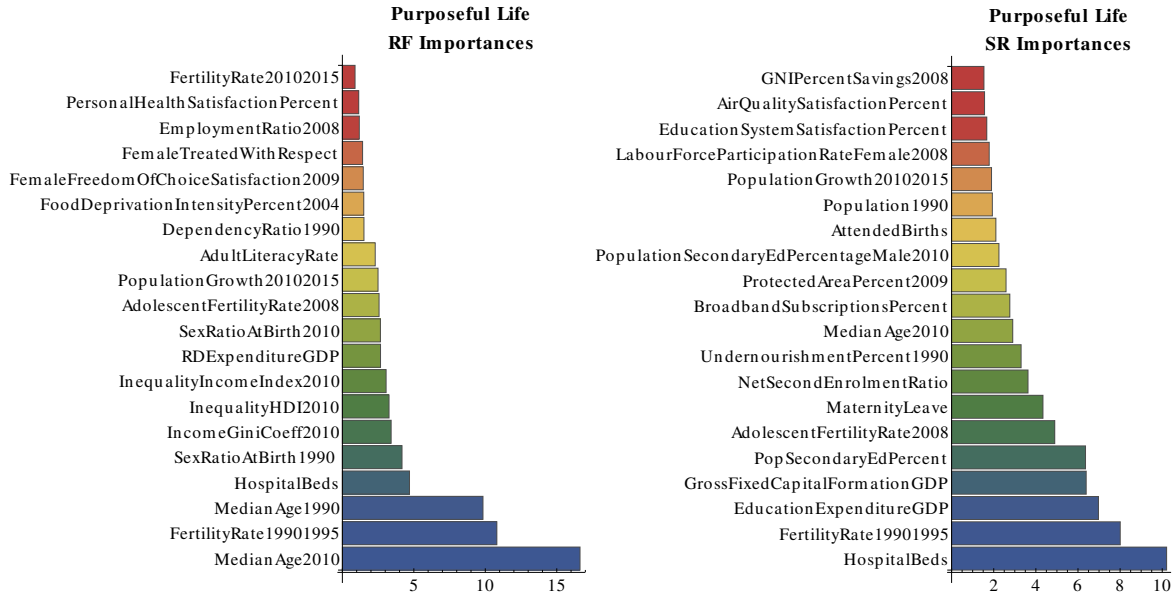


Figure 6.6: Importances obtained with RF and SR modeling Total Purposeful Life, using the parameter values given in Table 6.1.

6.1.3 GDP per Capita

The relation between country development and GDP per Capita is shown in Figure 6.4.

Importances obtained by five SR runs of two minutes are presented in Figure 6.7. While neither GDP nor Population scores high in importance, it is noted that the explicit formula $GDP/Population$ was retrieved in a few models. But since the majority of the population achieved a much lower level of quality, they will obscure these variables even though the importance is fitness weighted. This is an example of why models of sufficient quality should be used to estimate importances, and highlights the consequences of not doing so.

Importances obtained by RF are displayed in Figure 6.7. Neither the GDP nor Population variable is present in the top 20 most important variables, as was the case with SR. However, we observe that correlated variables dominate other variables. Many variables with 'HDI' in their name are correlated, and are also correlated with the GDP per Capita. This agrees with findings in other research regarding RF variable importance with correlated variables [107].

Importances obtained by five SR runs of twenty minutes are presented on the left graph in Figure 6.7. Both GDP and Population score high in importance, and it is observed that the majority of models are of high quality. By considering only members of the Pareto front, instead of the whole population, the importances shown on the right graph in Figure 6.7 are obtained. The two most important variables are now the GDP and Population, as is desirable in this problem setting. This demonstrates that when models are of sufficient quality, the estimated variable importances are more reliable.

RF Parameter	**Value**	**SR Parameter**	**Value**
Number of trees	1000	Independent Runs	5
In-Bag data	2/3	Time Constraint/ Run	20 minutes
Leaf Size	5	Accuracy Measure	$1 - R^2$
Candidate set size	$\|d\|/3$	Complexity Measure	sum of sub-tree sizes

Table 6.2: Parameter values used for modeling the GDP per Capita response variable. For symbolic regression we explore two series of runs, where the time constraint per run is either two or twenty minutes.

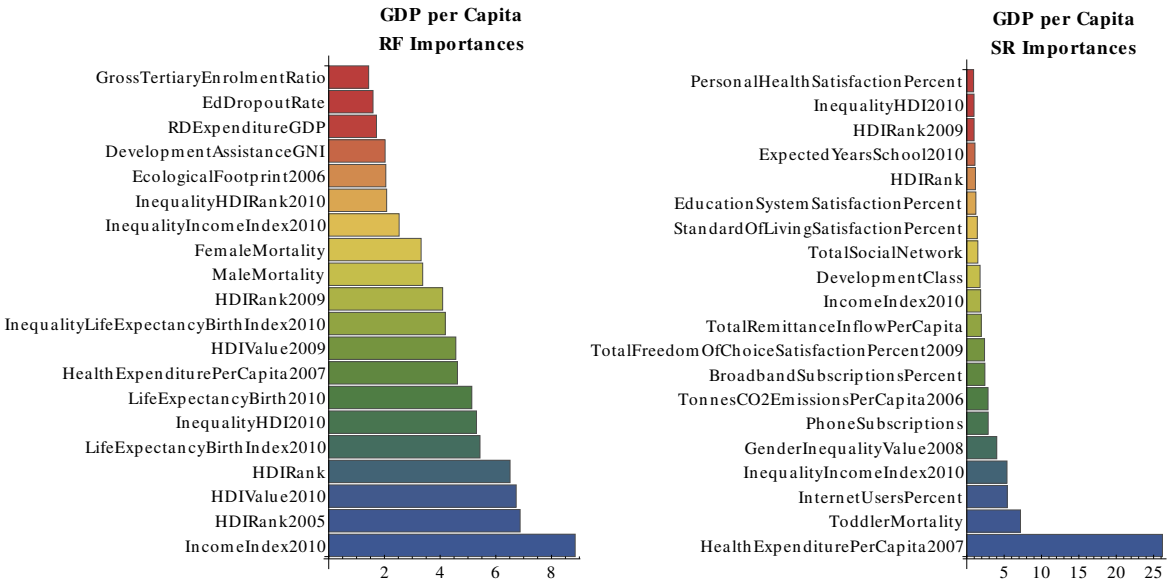


Figure 6.7: Importances obtained with RF and SR modeling GDP per Capita, using the parameter values given in Table 6.2. For SR these results correspond to five independent runs of two minutes.

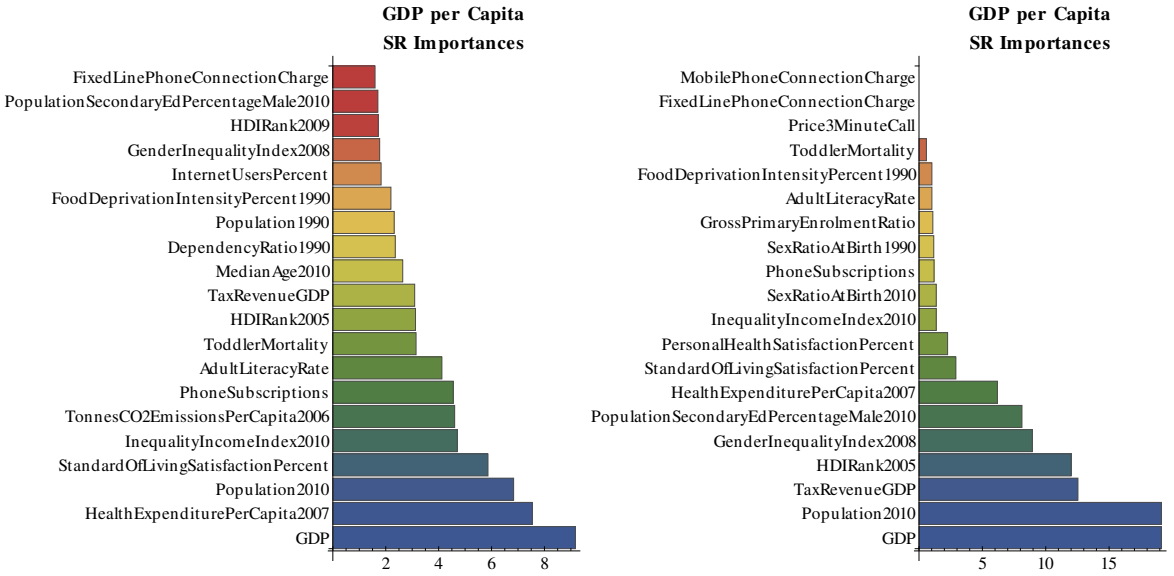


Figure 6.8: Importances obtained with RF and SR modeling GDP per Capita, using the parameter values given in Table 6.2. For SR these results correspond to five independent runs of twenty minutes. On the left graph, variable importances are estimated using the whole final population. On the right graph, variable importances are estimated using only the Pareto optimal models in the final population.

6.2 Tower Data

In this case study we examine an industrial data set available at [115], representing sensor measurements related to a distillation tower. Specifically the response variable is propylene concentration at the top of the distillation tower while the inputs correspond to temperatures, flows, and pressures related to the tower. Problems of this kind are known as inferential sensor development, and have many applications in the chemical industry. We refer to an excellent study on the application of SVMs for developing robust inferential sensors [49]. We provide a general context for this problem without delving into the details, summarizing [49, Ch. 1]

In an industrial setting it is extremely important to detect processes that are not running according to specification. Taking corrective action in time can avoid plant shutdowns, environmental contamination or hazard to human life. In other words, significant productivity is to be gained by monitoring such processes in real-time. Effective process monitoring is possible by measuring a wide variety of properties with hardware sensors. However, such sensors are not always practical or affordable. The solution is to develop a so-called soft sensor or inferential sensor, in other words develop a *mathematical model* instead of hardware.

Soft sensor development has several specific problems, we give an overview of the model requirements:

Complexity control – Inferential sensors are often prone to under-fitting or overfitting the learning data. This may seriously affect its reliability as well as its lifespan.

Using high-dimensional data and spaces – Industrial data sets typically contain many variables, and exhibit non-linear behavior. Overcoming the curse of dimensionality will be a key issue.

Robustness – Inferential sensors must be robust to noise in the measurements, and to outliers produced by faulty measurements. In addition the inferential sensor must be robust against changing operating conditions.

Good generalization capabilities – The learning data will typically be sparse such that the model must be able to extrapolate well. Graceful degradation of the model is preferable as compared to instable and erratic behavior outside the training domain.

Performing data compression and outlier detection – Data sets are so large that compression is essential. In addition outliers can contain useful information on abnormal behavior of the process.

Incorporating prior knowledge – As processes become more understood it is desirable to incorporate this knowledge into the model, as opposed to relying only on empirical data.

Adaptivity – The processes under study are dynamic, and it is expected that the model will adapt to gradually changing operating conditions.

Self-diagnostic capabilities – An inferential sensor must be able to evaluate its own predictive performance in order to be trustable.

From the requirements we observe that RF would not be a good modeling technique for soft sensor development, specifically the extrapolation is problematic as we demonstrated with an example in Section 3.2.4.6.

The univariate plot of the response variable is shown in Figure 6.9. To identify correlated variables we show the correlation matrix of the Tower data in Figure 6.10. We observe that many variables are correlated.

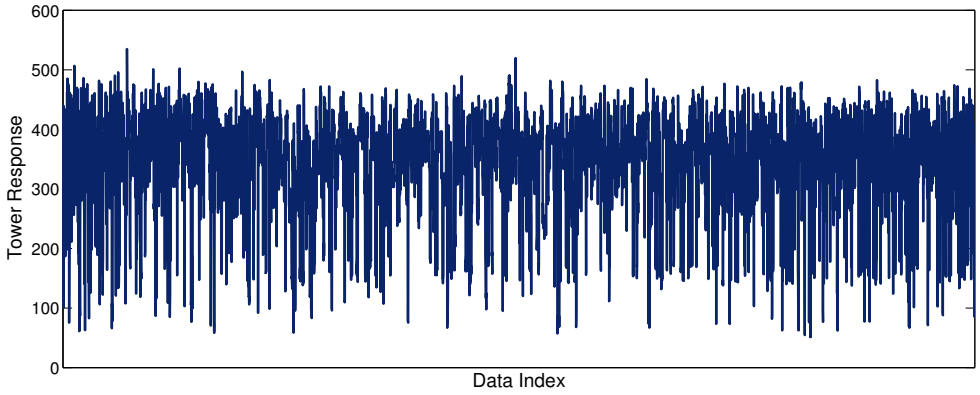


Figure 6.9: Univariate plot of the response variable in the Tower.

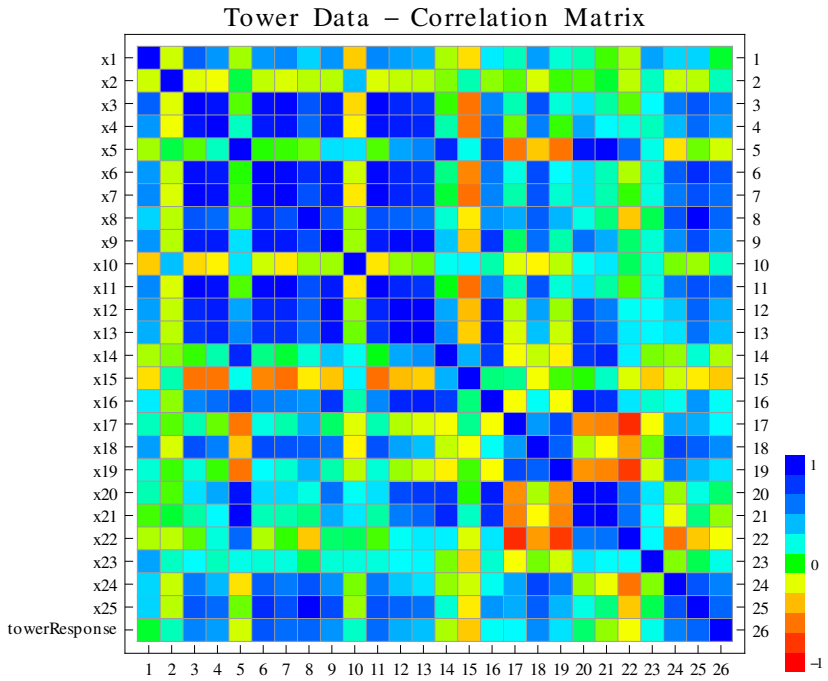


Figure 6.10: The correlation matrix of the Tower data set. Every square represents the correlation between two variables i and j where (i, j) is the index of that square in the matrix. We observe that many variables are correlated.

All modeling was performed on a quad-core i7 machine operating at 3Ghz, with a memory capacity of 12 GB. Other parameters are given in Table 6.3.

RF Parameter	Value	SR Parameter	Value
Number of trees	1000	Independent Runs	3
In-Bag data	2/3	Time Constraint/ Run	1 hour
Leaf Size	5	Accuracy Measure	$1 - R^2$
Candidate set size	$\|d\|/3$	Complexity Measure	sum of sub-tree sizes

Table 6.3: Parameter values used for modeling the Tower response.

Importance scores obtained with both RF and SR are shown in Figure 6.11. We observe that the methods do not identify the same variables. In addition we note that results obtained with SR are in agreement with similar studies in the literature [116]. We further observe that good quality models in SR only use the variables, ranking highest in the importance scores: x_1, x_4, x_6, x_{12}, x_{23}.

If we assume that the importances obtained by SR are reasonable, the results from RF make more sense. For example consider x_6 which scores twice as important as any other variable with RF, and from the correlation matrix in Figure 6.10 we observe that x_6 is correlated with several other input variables. Similar behavior was also observed in the experiment with correlated relevant variables in Section 5.2.1.4. In addition, our experiment on a non-linear model using correlated irrelevant variables from Section 5.1.2.4, showed that many correlated variables can cause fluctuations in importances even if the correlated variables are not part of the true input-output relationship. This was related to the Gini score and candidate set size in Section 5.2.1.3. We conclude that the Tower data contains properties of both experiments, and importance scores obtained by RF are likely to over- or underestimate depending on the correlation with other variables. In addition, the number of important variables can not be determined using the importances obtained with RF, as shown in Figure 6.12.

To asses the quality of produced models with both techniques we randomly chose 10% of the data to be a test set. Both techniques created new models using the 90% training data. The RMSE on the test set was computed and is shown in Table 6.4. To see how variable selection would influence model accuracy, we remodeled the Tower data using the same training and test sets, while using only the five most important variables as identified by SR. The RMSE of models created using only five variables is also shown in Table 6.4.

Since the RMSE does not change dramatically when using only five inputs to create models, we conclude that this variables indeed contain most of the information present in the data set. We also note that since the test set is chosen randomly, most test points will be in the training domain. However, for this Tower problem we stress that extrapolation was an explicit design requirement, which is not possible with the model created by RF. In contrast, models created by SR for this problem have been shown to behave reasonable under extrapolation [116].

To visualize the prediction errors we sorted the test data according to the response, in ascending order. Figures 6.13 and 6.14 show the prediction error on the test set. We observe that RF has a smaller prediction error compared with SR. Recall that RF averages training points 'close' to the point to predict, such that the prediction error will be reasonable in the training domain. In contrast, SR will search for global models with generalization capabilities. The RMSE is not a good indication of the generalization capability of a model, and SR will have a higher RMSE compared with RF. However, in context of the Tower problem, the models produced by SR are preferable to the models produced with RF.

Number of variables in the data set	RMSE RF	RMSE SR
25	15.59	31.08
5	17.35	29.48

Table 6.4: RMSE on a random test set with 10% of the data set. We observe that the RMSE does not change dramatically when creating models using only the five most important variables as identified with SR. This indicates that these variables contain most information present in the original data set.

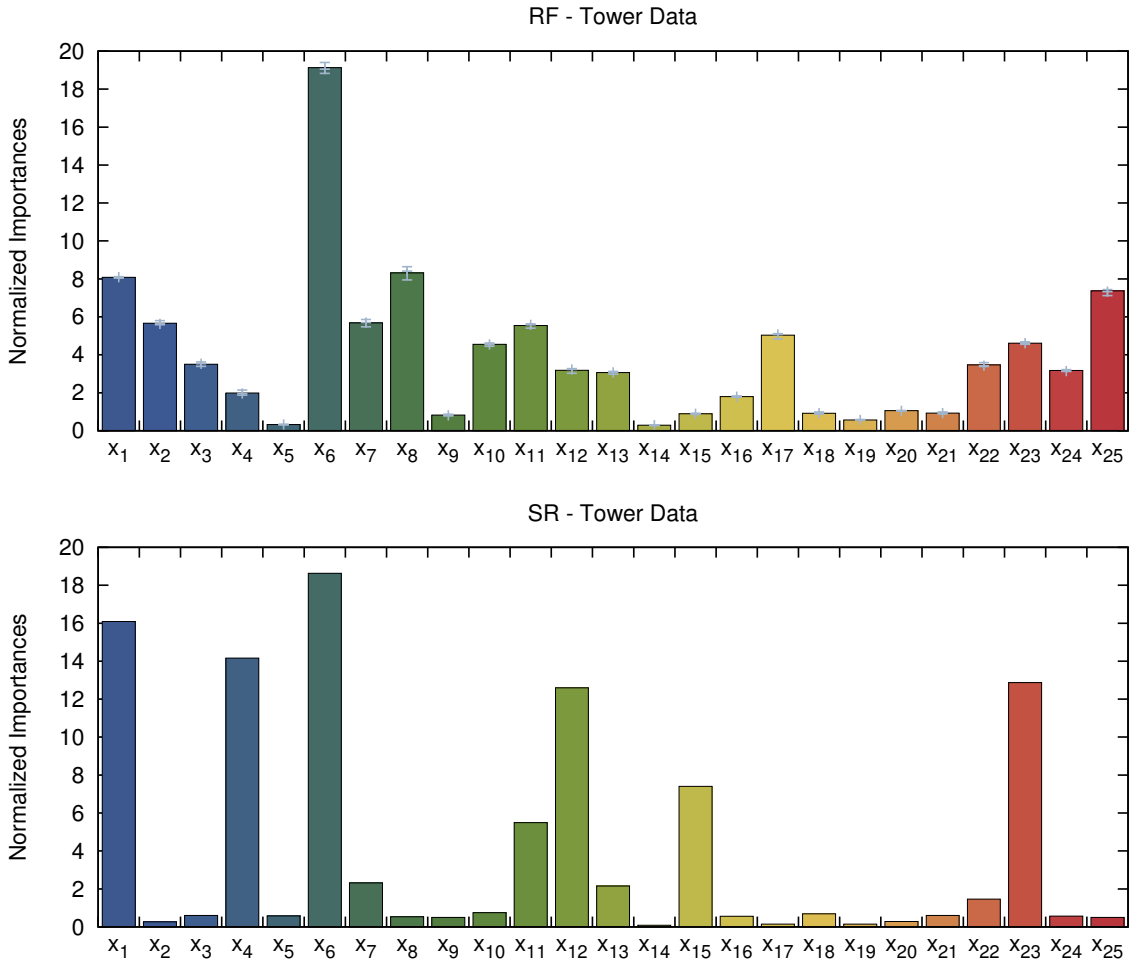


Figure 6.11: Comparing the importances obtained by RF and SR on the Tower problem. Observe that the methods do not agree. Results for SR are consistent with the literature[116], and those variables have been shown to produce good models. If we assume that variables identified by SR are a superset of the driving variables, we observe from the correlation matrix in Figure 6.10 that both relevant and irrelevant variables are correlated. We explain the behavior of RF in the text, analogously to our earlier experiments with correlated variables, observing that properties from both experiments are present simultaneously.

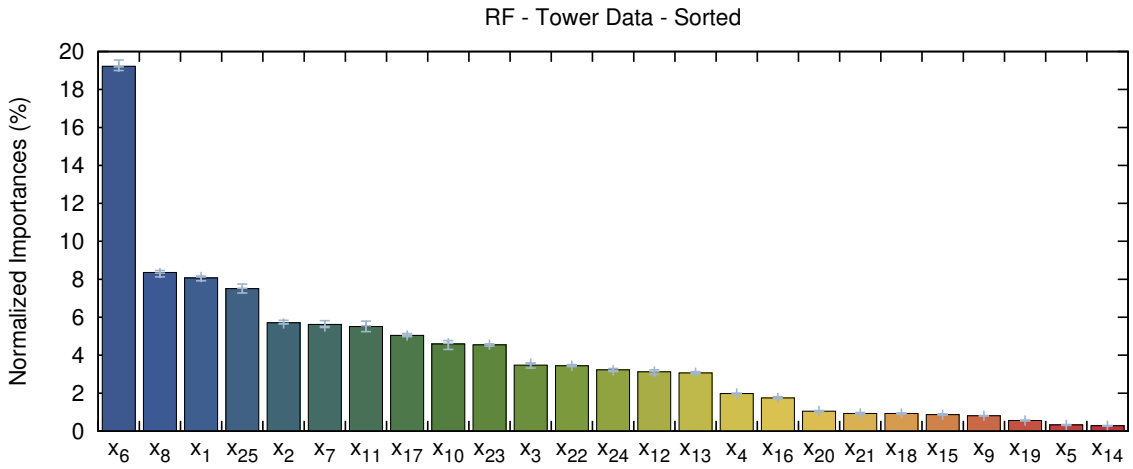


Figure 6.12: The sorted variable importances as computed by RF. From this importance scores the number of important variables can not be identified. This contrasts with importances obtained by SR as shown in Figure 6.11.

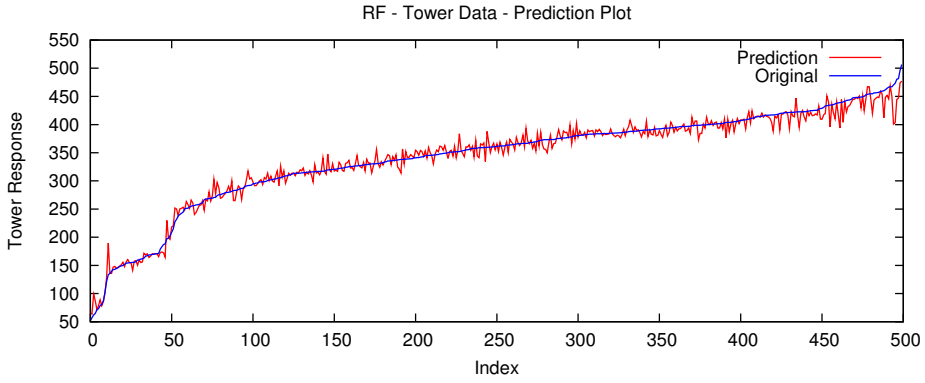


Figure 6.13: Visualization of the prediction error by RF on the Tower problem, using all variables for training. Note that the x-axis represents the index in the data set, and the points have been sorted according to the response, in ascending order. In other words the line marked 'Original' is not a response surface.

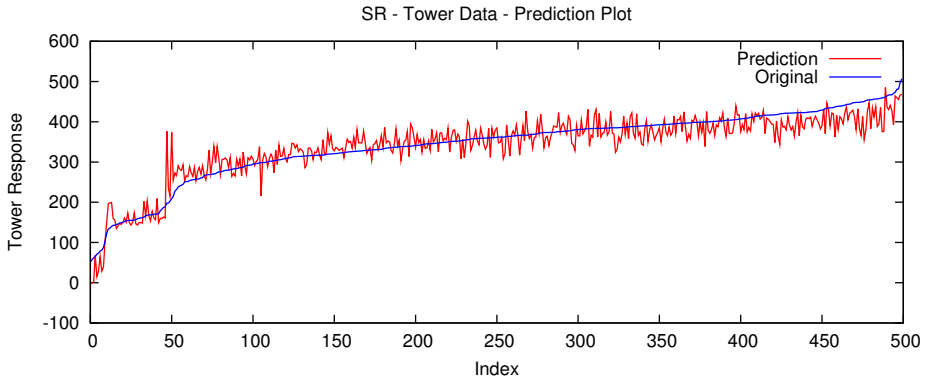


Figure 6.14: Visualization of the prediction error by SR on the Tower problem, using all variables for training. Note that the x-axis represents the index in the data set, and the points have been sorted according to the response, in ascending order. In other words the line marked 'Original' is not a response surface.

6.3 Summary

This chapter applies random forests (RF) and symbolic regression (SR) on real-life problems.

Several variables from the data set used to compute the Human Development Index (HDI) are modeled. Importance scores obtained by both RF and SR agree on some important variables, but differ on variables with lower importance. In addition the importance scores are interpretable in the application domain.

For the second case study an industrial data set of a distillation tower is modeled. Here we observe that variable importances obtained by RF are strongly influenced by correlated variables and SR identifies other driving variables. We confirmed that the prediction error does not increase significantly when the data is modeled using only the variables with highest importance score as obtained by SR. This indicates that SR is more applicable on large scale problems with many correlated variables.

Chapter 7

Implementation

7.1 Random Forest

Implementations of RF are freely available in several languages. We mention the original implementation in Fortran or C [11], which has been ported to R [69]. The R version has also been ported back to C with a Matlab interface [46]. However, for several experiments we wanted more control than was practical with any of these packages, and they were not easily extendable. In particular visualization and an intuitive graphical user interface (GUI). Furthermore, parallelization of the tree building process resulted in a near linear speedup per additional core, which was greatly appreciated by the authors when running the many experiments.

The focus of our implementation is to be flexible, and speed was only a secondary objective. Nevertheless we aimed for a practical balance. We opted to implement RF in C++ since we are familiar with the language and know it has the potential to be very efficient. In particular the use of templates enabled us to achieve high flexibility and remain efficient.

Importing data sets into a program is easier said than done. Not all data sets are perfect, and values can be empty, unknown, or not numeric. We wanted to have one representation available throughout the program. We opted to first read the data from a CSV[1] file and represent it as a matrix of cell of different types such that all validation could be performed on this representation. Only after validation is the data converted into a proper matrix type. This resulted in a more maintainable solution, since the actual algorithm is not dependent on the external data formats.

The class diagram of the main classes is shown in Figure 7.1. Consequent with the logical structure of RF we have an `RFEnsemble` which contains a vector of instances of `RFTree`. An `RFTree` has the root node as data member of type `RFNode`, and each node has two child nodes. These classes all have the same template parameters, namely `float_type` and `policy_type`.

The `float_type` allows for compile-time switching between single and double precision floating point operations. Since the RF algorithm does not perform intensive calculations where floating point errors would be a problem, using single precision will result in faster program execution without negative impact on the results.

The `policy_type` contains several algorithmic parameters, for example which split criterion to use. Since we used a uniform interface to represent any split criterion, other criteria besides the Gini index could be investigated as well. Details on how to handle the random generator

[1] Comma Separated Value

and parallelization details are also covered by `policy_type`. Incorporating other variable importance measures could also be a future option.

A tree is constructed recursively. As can be seen from the static method `RFTree.train`, a tree will be constructed from a data matrix. That data will be split into in-bag and out-of-bag data. A type as specified by `policy_type::DataInterface` will be used to contain the in-bag data. This type has the necessary manipulator methods available, such that a node will use this container to find optimal splitting values and create child nodes recursively.

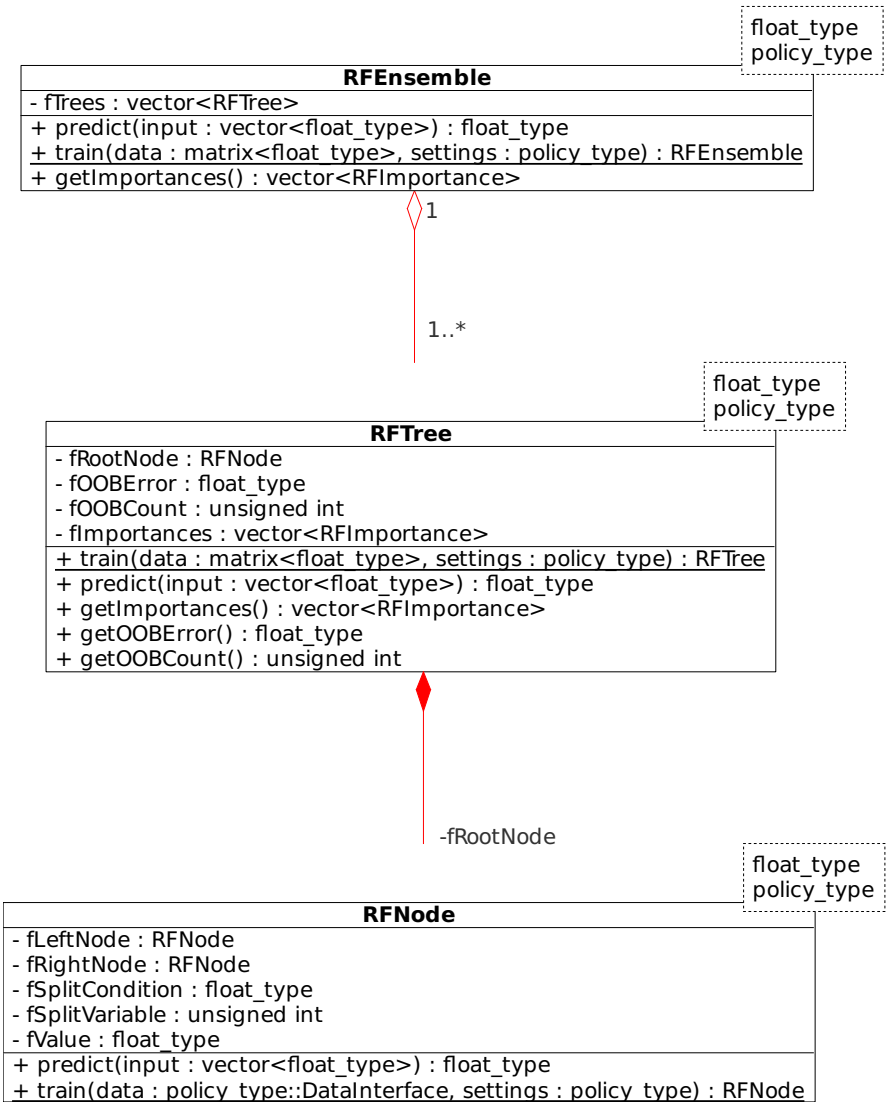


Figure 7.1: Class diagram of the most important classes for random forest. Only the most important fields and methods are shown.

It is the responsibility of the `policy_type::DataInterface` to maintain an internal structure to facilitate future data requests. For instance, the computation of the Gini criterion requires that the data of one variable is sorted, and the response is traversed in that sorted order. By implementing a generic interface the internal storage could be altered. In particular, we speculated that by using a clever internal organization it might be possible to avoid sorting

for every candidate, for every split. We tested an indexing approach, where the sorted indices are always available, but concluded this did not improve the performance. The extra work to maintain the internal structure and the penalty of accessing the data indirectly proved to be a slower approach than sorting the current partition. Since the partition size is expected to decrease rapidly, we find that only the first few splits on large data sets will spend a relatively long time to sort the data. For smaller partitions the sort operation is not as expensive. The current storage technique is to keep partitions in contiguous memory, and not explicitly split the data set.

The prediction is performed by iterating over the nodes, instead of the more intuitive recursive approach. Such recursive approach could build up the program stack, and since the tree depth is dependent on the data set size we preferred not to.

We provided a GUI for our implementation, primarily out of practical considerations. A screenshot of the GUI is shown in Figure 7.2. While the GUI is rather dense, this allowed us to easily adjust all parameters when performing experiments.

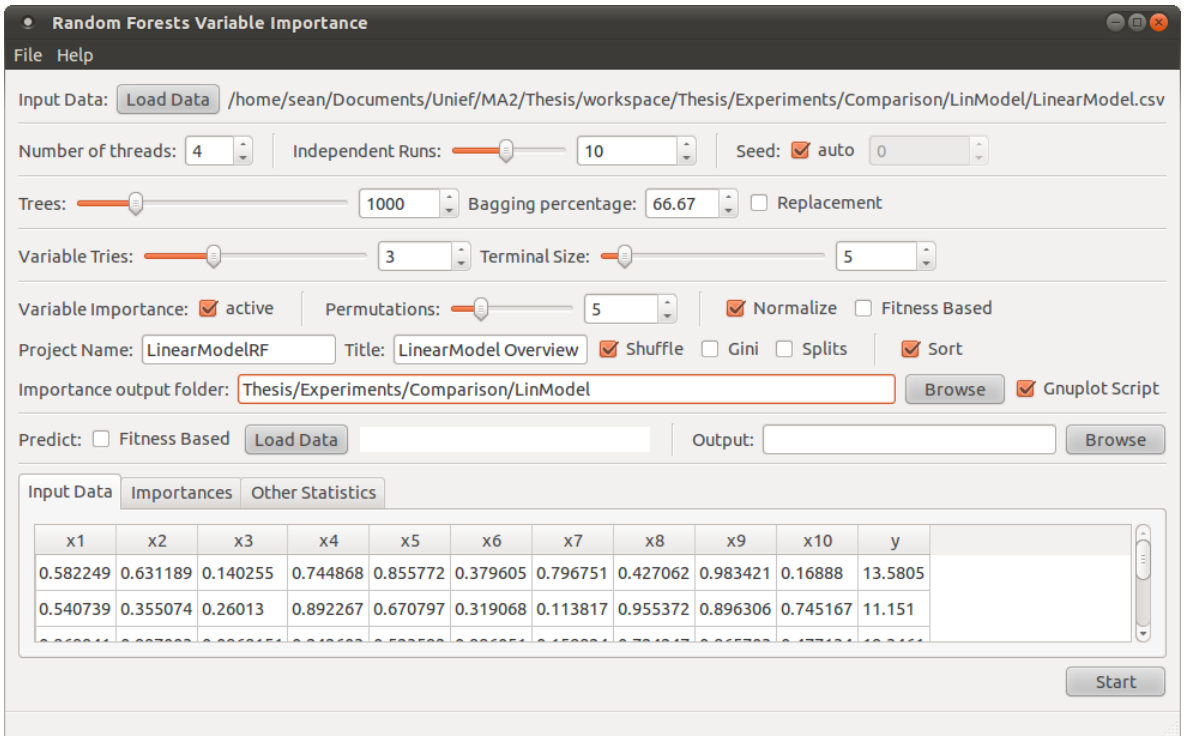


Figure 7.2: A screenshot of the GUI of our RF implementation. While the presentation is dense, all relevant parameters are easily accessible.

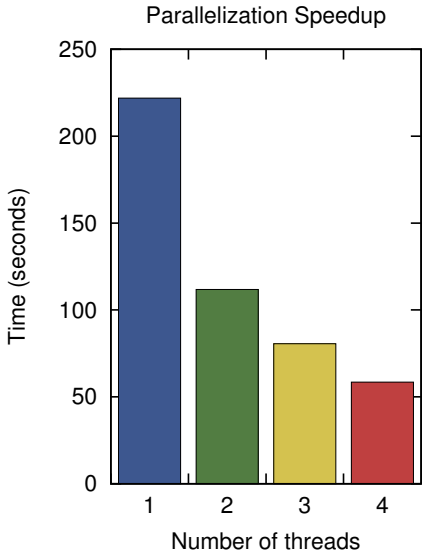


Figure 7.3: The time in seconds for constructing a forest of 1000 trees using a data set of 500 points in 10 dimensions, and determining the variable importance of the data set. Remark the linear speedup by increasing the number of threads.

Since the construction of all trees is independent, we opted to parallelize the tree construction and the computation of variable importance. The in-bag data of all parallel constructions are required to fit into main memory, but we did not find this a practical constraint. We used Intel Thread Building Blocks [43] (TBB) to manage and schedule threads. We found this library accessible and recommend its use in parallel applications. To test the effect of parallelization, we timed the execution of constructing a forest of 1000 trees using a data set of 500 points in 10 dimensions, and determining the importances using 10 independent permutations. All tests were performed on an Intel Core i7 quad core machine, the measurements are shown in Figure 7.3. We observe a near linear speedup in the number of threads.

7.2 Expression Evaluator

We wanted to have a tool for determining the variable importance from expressions, such as measures mentioned in Section 4.3.2. To this end we implemented a generic expression evaluator, focusing more on flexibility than speed.

Plain text representations of expressions are parsed with Bison [22] and Flex [87], such that each symbol is represented as a node in a parse tree. We opted for an object-oriented design where the type of nodes is dependent on the operation they represent. This allows the addition of extra operators in the future, in particular arbitrary complex operators such as separate programs.

The class diagram for the main classes is shown in Figure 7.4.

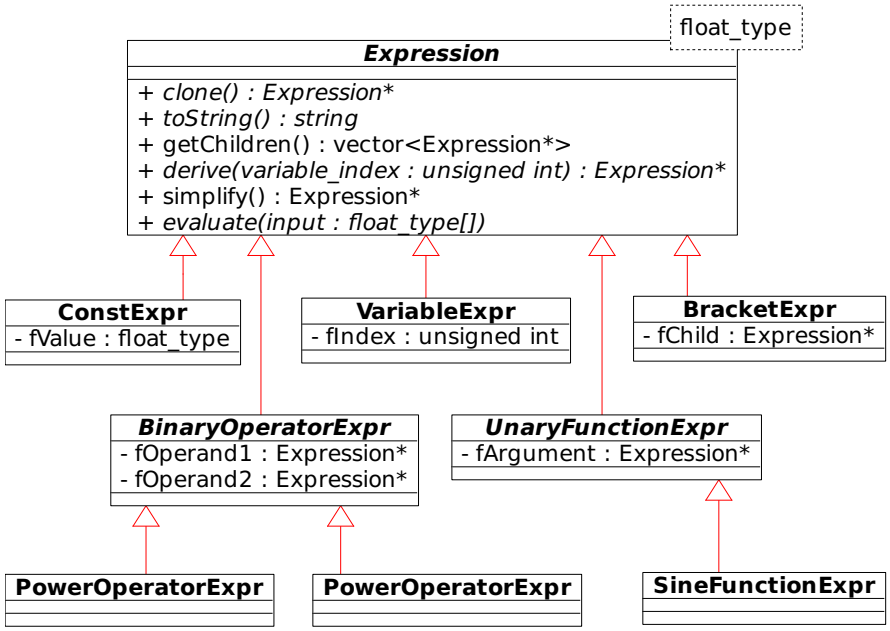


Figure 7.4: Class diagram for the main classes of the expression evaluator. Only the most important methods are shown.

The derivation of an expression is handled recursively, where every operator provides the knowledge of how to derive it. This design allows all current and future operators to be handled uniformly, which is a practical advantage. The elimination of variables is handled analogously.

We provided a uniform interface for variable importance measures, as shown in Figure 7.5. This facilitates the development of new measures. Currently we support the Mean, Shuffle, Derive, Eliminate measures as described in Section 4.3.2.

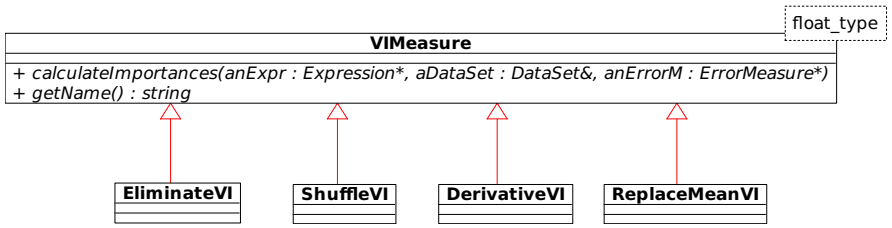


Figure 7.5: Class diagram for classes determining the variable importance. We chose to use a uniform interface for all importance measures.

Error functions are also handled through a uniform interface, such that any error function can be used without influencing other logic. Currently we support the sum of square errors, and $1 - R^2$.

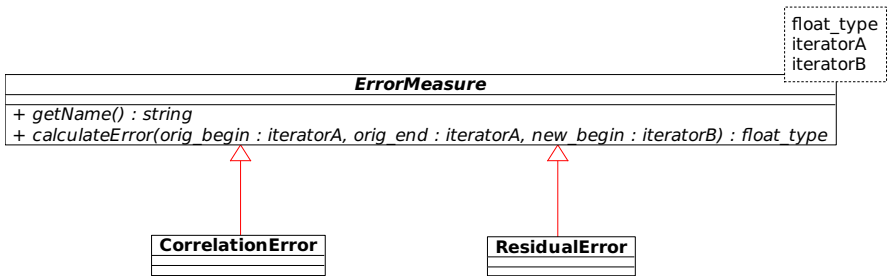


Figure 7.6: Class diagram for classes calculating a prediction error on a range specified by iterators.

We consider this implementation to be a prototype that could later be incorporated in an SR implementation, such that variable importances in expressions could be computed on individuals during an SR run.

Chapter 8

Conclusion

8.1 Research and Contributions

In this thesis we have studied regression problems, while showing the strong relation with classification, optimization and search problems. We identified several challenges that are present in real life data sets in the Introduction, in particular the quantification of the contribution of individual variables to the response. We stressed that models describing the data set are invaluable to interpret the importance of variables. We defined variable importance recognizing that it is a local property in Chapter 2, and introduced desirable properties of a good method for computing these importances, shown again here:

- **Interpretability**–Obtained importances should reflect the importances of the true input variables, without transformation.
- **Strictness**–Only variables relevant to describing the response should be allocated importances, so spurious variables should not appear important.
- **Conservativeness**–At intermediate stages of importance analysis all potentially interesting variables should receive importance.
- **Reproducibility**–A result can only be considered correct if it is reproducible.
- **Universality**–It is desirable for importances to be mutually comparable, and in addition to be problem independent.

In this thesis we focused on two specific techniques, namely random forests (RF) and symbolic regression (SR). Random forests is a commonly used technique for variable screening, and is based on an ensemble of classification and regression trees. Symbolic regression has its roots in evolutionary computation. It creates multiple models represented by explicit algebraic expressions, and has been shown to perform well on industrial data sets where traditional methods are no longer sufficient. These techniques are discussed at length in Chapter 3 and 4 respectively, and include algorithmic details. We illustrated the behavior of both techniques using intuitive examples. In particular in Section 3.2.4.4, we studied the parameter sensitivity of RF using a data set generated by interpreting the gray-values in an image. This results in a hard modeling problem where the gray-value is to be reconstructed using only the pixel coordinates. Since the model prediction could be also represented as an image, we were able to

intuitively approach the influence of RF parameters. To our knowledge such intuitive examples are not readily available in the literature.

In Chapter 5 several experiments relevant to real life data sets highlight the differences in variable importance between RF and SR. After examining and comparing the results we concluded that the variable importance measures of RF are prone to defects in the data set, while those of SR are more robust against such defects. We provided explanations on why RF exhibits this behavior in Section 5.2. We did not encounter a discussion from this point of view in the literature. In addition further experiments show that the sensitivity of RF to defects is due to the technique used to compute the variable importances as described in Section 3.3.2, and not an artifact of the generated models. From the experiment in Section 5.1.2.1 we conclude that RF can underestimate the importance of variables in a non-linear model given a small data set.

In addition we identified that it is of interest to determine variable importances given an explicit algebraic expression. Such importances could be readily incorporated in a SR algorithm, since models are represented as expressions. We presented preliminary experiments in Section 5.3.

Both RF and SR are applied to two real life case studies in Chapter 6. We concluded that SR is capable of delivering more interpretable results than RF.

For the RF component of both case studies and experiments we used software developed in context of this thesis. This software is described in Chapter 7. The RF implementation is notable for its extensive visualization capabilities. In addition we parallelized the model construction and variable importance computation. We have shown this approach scales linearly with the number of processors. This software is controlled using a graphical user interface providing access to all parameters discussed in this thesis. To our knowledge a prior parallel implementation of RF for regression was not available.

For experimenting with variable importance in expressions such as described in Section 4.3.2 we developed software, detailed in Chapter 7. Notable is the derivation of algebraic expressions and elimination of variables from algebraic expressions.

In the Appendices we provide a summary of our background research regarding variable importance, feature selection and dimensionality reduction. Principal component analysis and artificial neural networks are commonly used for such problems and are discussed in detail in Appendix A and B respectively. The popularity of these techniques is readily observed from references to recent applications.

8.2 Future Work

The model building process of SR is ideally a continuous evolution. But while convergence is assumed, there is little information available about the speed of convergence or the proximity to acceptable solutions. We observed that a population of insufficient quality will yield unreliable variable importances, so model quality must be verified when drawing conclusions. Therefore, it remains of great interest to look for robust algorithmic configurations that ensure the discovery of models of sufficient pre-defined quality.

A more formal framework to establish the importance of variables given an expression tree would be instrumental in guiding the evolutionary process. Future research could expand on the preliminary results obtained in this thesis.

As we observed from the evolutionary approach by SR, it is beneficial to incorporate

variable importance estimation in the modeling technique, since it allows the identification of interesting areas of the search space. It is of interest to investigate whether other techniques offer similar opportunities and how importance information discovered during a modeling run can be actively used within the same modeling run.

We are confident that RF can provide useful prediction models if extrapolation is not a requirement, based on observations from the examples in Section 3.2.4, and the Tower case study. We speculate that a better variable importance technique can be developed by incorporating knowledge about the model structure, in contrast to the current 'Shuffle' method.

We suspect that some problems with the importances derived by RF, could be solved by introducing a special kind of randomness in the trees. In particular using a varying candidate set size depending on the current depth in the tree. When this set starts only with very few candidates and gradually grows in size, a single variable would be prevented from dominating early on.

An interesting feature of RF is to produce a proximity matrix, defining a distance measure on the data depending on which points were together a leaf partition. Although this topic was not treated in this thesis, we think this can provide a very useful tool to fill in missing values in the data by interpolation.

While we compared importances obtained by RF and SR in this thesis, comparison with regularized learning techniques, such as lasso[111] and elastic net[125], is of interest.

We see great potential in the analysis of variable importance, and stress the need to understand the modeling technique as well. However, intuitive and user friendly tools are not readily available to the researcher, resulting in many different implementations. It would benefit all parties to develop a consistent framework.

Appendices

Appendix A

Principal Component Analysis

A.1 Motivation

Principal component analysis (PCA) is a multivariate statistical technique invented by Karl Pearson in 1901 [89]. The original purpose of the method was finding lines and planes that best fit a set of points in a p dimensional space. Thirty-two years later Hotelling devised the method as a technique to find a smaller 'fundamental set of independent variables (...) which determine the values' of the original data set [40, 45]. Hotelling called such fundamental independent variables 'components'. If the components are chosen such that they maximize their successive contributions to the total variance of the original variables, they are called 'principal components' [48], hence the term principal component analysis.

At present PCA is a standard tool in statistical data analysis and is used for dimensionality reduction and pattern recognition in diverse fields. Section A.5 gives an overview of the applications of PCA.

Much literature has been dedicated to PCA, we refer to [45, 48] for an extensive treatment. This section is loosely based on material from [98, 45, 48, 109] and will introduce the concepts of PCA and some basic properties.

A.2 Algorithm

PCA operates on multivariate data, that is data having multiple continuous attributes. The data is represented by the $m \times n$ matrix D.

$Attr_1$	$Attr_2$	$\dots$	$Attr_n$
$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$
$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \dots & d_{m,n} \end{bmatrix} \quad \text{(A.1)}$$

The goal of PCA is to find a new orthogonal basis such that the data is expressed in a more meaningful way. In the process we will discover hidden structures and have the opportunity to discard components that do not make a noteworthy contribution to structure in the data.

Each sample $\mathbf{d}^i$ of the data set can be interpreted as the vector $(d_{i,1}d_{i,2}\dots d_{i,n})^T$ in n-dimensional space, spanned by some orthonormal basis. By definition every sample $\mathbf{d}^i$ can be represented as a linear combination of this orthonormal basis. Actually each data record

naturally expresses a linear combination of the canonical basis. Therefore, the canonical basis is the most logical choice for the basis spanning the original data set:

$$\begin{bmatrix} e_1 & e_1 & \dots & e_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} = I$$

With the notion that each data point is expressed by a linear combination of a basis, the goals of PCA can be reformulated: PCA will construct a new orthogonal basis by linearly combining the original basis, such that the new basis expresses the data in an optimal way. Remark that PCA limits the transformation to the new basis to be linear, which will vastly simplify the process of finding a new basis. However, this property might be limiting for many data sets.

Let the linear transformation from the original basis to the optimal basis be expressed by the $n \times n$ matrix P. Then this transformation can be used to create a new $m \times n$ data set Y containing the transformed data records:

$$Y = DP, \quad \text{where } P = \begin{bmatrix} p_1 & p_2 & \dots & p_n \end{bmatrix} \tag{A.2}$$

The columns of P denote the new basis vectors $\{p_1, \dots, p_n\}$. The change of basis expressed by P in Equation A.2 can be interpreted geometrically by stretching and rotating the vectors associated with $\mathbf{d}^i$. PCA will construct each vector p_i such that it is orthonormal with respect to any other vector p_j for all $i \neq j$. Otherwise the set $\{p_i\}$ would not be an orthogonal basis.

Once the optimal transformation P is found, the principal components of D are exactly the vectors $\{p_1, \dots, p_n\}$. The question remains how to chose a good basis for our data. Let us first examine what the desired properties of Y are, in order to gain insight in the properties of P as well.

It is expected that a data set contains irrelevant information, for example due to imprecisions during data collection or a limitation in the data representation. Analysis techniques only work when the amount of useful information is sufficient, compared to the irrelevant information. To use a term from signal processing, this balance between relevant and irrelevant information is expressed as the *signal-to-noise* ratio. Where a signal denotes relevant information, and noise denotes irrelevant information. This ratio is expressed by:

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{A.3}$$

Where σ^2 is the variance. Measurements of high quality will have a high SNR, while a low SNR is an indication of poor data. It follows that if the $SNR > 1$, the direction with the largest variance will contain the most relevant information. In PCA, the data is assumed to be of sufficient quality, such that the directions exhibiting the largest variance in the n-dimensional space are assumed to contain the most interesting characteristics of the data set.

The directions exhibiting the largest variance do not necessary coincide with the original basis. As an example we generate a two dimensional data set consisting of 100 records according to the following rules:

$$U = rand(0, 2), \quad X_1 = 2U + rand(0, 0.5), \quad X_2 = U + rand(0, 0.5) \tag{A.4}$$

Where $rand(a, b)$ generates uniform random numbers between a and b. The resulting data is visualized in Figure A.1, and it is apparent that the original basis does not express the data well. The arrow σ^2_{signal} indicates the direction of the largest variance, intuitively this direction would be a more suitable basis vector. To complete the new basis the second basis vector is constructed perpendicular to the first one, indicated by the σ^2_{noise} arrow on Figure A.1. The directions indicated by the arrows conceptually represent directions of signal and noise. The generating function from Equation A.4 indicates that both variable X_1 and X_2 contain the same information, expressed by their correlation. It is not hard to see that they could be replaced by a single variable u. Moreover the variable u will be defined as linear combination of both variables.

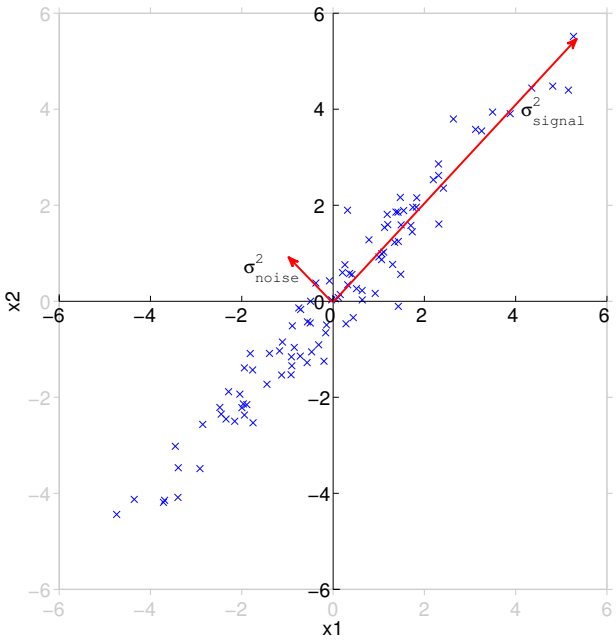


Figure A.1: Plot of the data set generated by Equation A.4. The σ^2_{signal} indicates the direction of highest variance which will chosen as the first principal component. The σ^2_{noise} arrow indicates the variance of the noise compared to the first principal component.

This example illustrated the central idea behind the dimensionality reduction of PCA: correlated variables can be replaced by a single variable by performing a linear transformation. While this assumption simplifies most problems considerably, it is crucial to see that it only holds for linearly correlated variables.

Finding the direction with the highest variance in a 2-dimensional space corresponds to finding the best-fit line of the data points. For higher-dimensional problems this concept has to be generalized. To this end we introduce variance and covariance formally. For a vector $A = (a_1 \dots a_n)$ the variance is defined as:

$$\sigma^2_A = \frac{1}{n}\sum_{i=1}^{n}(a_i - \mu_A)^2, \quad \text{where } \mu_A = \frac{1}{n}\sum_{j=1}^{n} a_j \tag{A.5}$$

In case μ_A is zero this simplifies to:

$$\sigma_A^2 = \frac{1}{n}\sum_{i=1}^{n} a_i^2 \tag{A.6}$$

Consider two vectors A and B with $\mu_A = 0$ and $\mu_B = 0$, the covariance of A and B is then defined as:

$$\sigma_{AB}^2 = \frac{1}{n}\sum_{i=1}^{n} a_i b_i \tag{A.7}$$

Remark that the variance σ_A^2 is actually the special case of the covariance of vector A with itself, or alternatively: $\sigma_A^2 = \sigma_{AA}^2$. The covariance expresses the linearity of the relationship between two vectors [98]. Variables are correlated if the covariance is nonzero, where the magnitude indicates the degree of redundant information. We can reformulate Equation A.7 using vector notation for A and B, where A and B are column vectors from matrix D. The covariance is then defined using the dot product of vectors d_i and d_j:

$$\sigma_{d_i d_j}^2 = \frac{1}{n} d_i^T d_j \tag{A.8}$$

This definition of covariance naturally generalizes to an arbitrary set of column vectors. The covariance matrix S of D can now be defined as

$$S = \frac{1}{n} D^T D, \quad \text{where } D = [d_1\ d_2\ \ldots\ d_n] \tag{A.9}$$

Every element $s_{i,j}$ of the covariance matrix S contains the covariance between the ith and jth attribute of D. It follows that S must be a symmetric matrix, since $\sigma_{AB}^2 = \sigma_{BA}^2$. Remark that the elements on the diagonal contain the variance of each attribute. Since the covariance reflects the correlation and redundancy, large values on the diagonal of S correspond to interesting structures in the corresponding attribute. Large values outside of the diagonal indicate that variables corresponding to its indices are correlated and contain redundant information.

By defining the transformed attributes y_i such that they have zero correlation and maximal variance we impose the following structure on the covariance matrix T of Y:

- Elements that are not on the diagonal must be zero, since we require every new attribute to be uncorrelated.
- Elements on the diagonal must have decreasing values, since we require principal components to be ordered by descending variance.

It follows that finding the principal components of D is equivalent with finding the diagonalization of S. There are multiple methods to diagonalize S which will be explained shortly. We found that the following algorithm for finding principal components explains the method intuitively:

1. Choose the normalized direction p_1 in which the variance in D is maximized and let this direction be the first principal component.
2. Obtain another direction p_i that maximizes the variance and that is orthonormal with respect to all previous components.

3. Repeat until n components are selected.

While this algorithm is intuitive, it is certainly not the most efficient algorithm to find the principal components. By using eigenvector decomposition or singular value decomposition we can obtain the solution analytically.

A.3 Example

Figure A.2 shows a data set with correlated variables x_1 and x_2. The result of applying PCA to this data set is shown in Figure A.3. To illustrate that correlated data can hide significant information, PCA is also applied to the data shown in Figure A.4. The resulting data set after transforming to a new basis is shown in A.5.

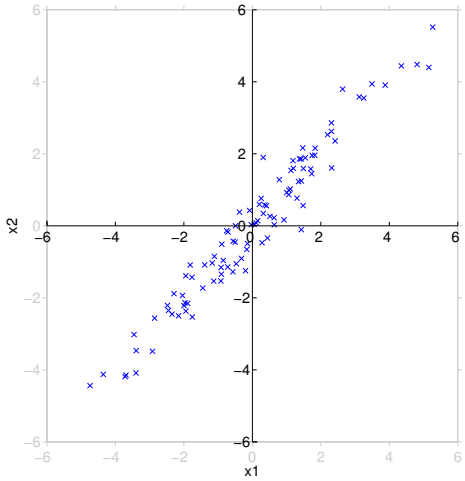


Figure A.2: The values of two correlated variables x_1 and x_2.

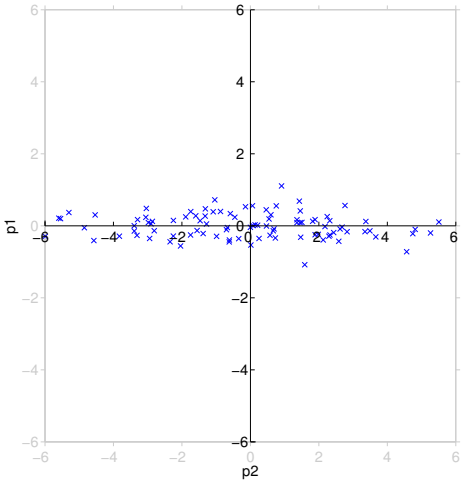


Figure A.3: Result from applying PCA on the data shown in Figure A.2. Variables p_1 and p_2 are linear combinations of x_1 and x_2, removing the correlation.

A.4 Variable Importance

In Chapter 2 we stressed that models are crucial for problem understanding, and variable importances should be determined from a suitable model. In the approach of PCA there is no distinction between input data X and output data Y. Consequently, PCA is not concerned with functional relationships between input and output, and operates on a data matrix D.

The literature refers to PCA as a feature extraction technique, with a curious definition of extraction. When a new feature is created by combining original features, this new feature is said to be 'extracted'. Such feature extraction should not be confused with feature selection.

Even though PCA is sometimes described as a feature selection technique, this is different from what we understand as feature selection. The transformed features are combinations of potentially all original features, so technically no features are selected. This alternative use of

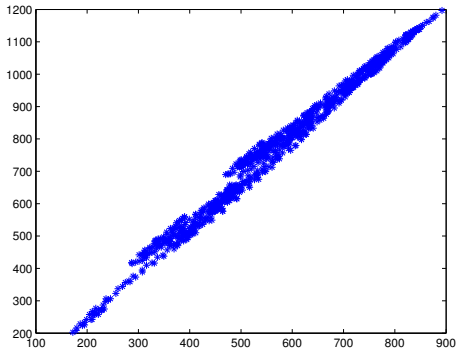


Figure A.4: A data set suspiciously similar to data shown in Figure A.2.

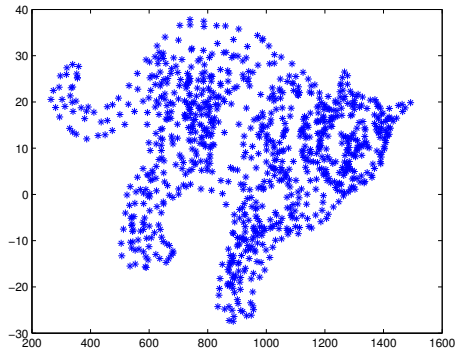


Figure A.5: Result from applying PCA on the data shown in Figure A.4. This information was previously obfuscated by the correlated variables.

the term feature selection can be confusing, especially since no functional relationships are considered. Hence the transformed features do not provide a good indication of the importance of the original features.

Variable importances are often determined by using PCA in conjunction with partial least squares regression and variable importance in projection. This is useful when linear regression is applicable, but assumes explicit knowledge of the model structure. This assumption does not hold for several problems studied in this thesis.

A.5 Applications

PCA is used for dimensionality reduction as a preprocessing step in tandem with other classification techniques for medical diagnosis [64, 92, 18]. The difference between the principal components and the actual discriminatory features for medical diagnostics is emphasized in [90], discussing issues of interpretability of the principal components as well.
In financial applications PCA is used to pre-process multi-collinear data, for instance to predict stock prices [108].
Other fields where PCA is employed extensively include image processing [21, 121, 124] and signal processing [78, 85, 42, 35]. As a specific case of image processing, PCA finds many applications in face recognition [41, 74, 118].

An interesting link between PCA and neural networks is demonstrated in [122] and from a different point of view in [96].

A.6 Summary

In this chapter we have demonstrated how principal component analysis (PCA) works. The assumption is made that the directions exhibiting the largest variance will contain the most interesting characteristics of the data set. The data is represented in a new orthogonal basis such that every axis maximizes variance along that direction, these directions are the principal

components of the data set. This allows to reduce the dimensionality of the data set by removing features in this new basis with lowest direction of variance.

PCA is in itself not a variable importance technique, but is often used in conjunction with partial least squares regression and variable importance in projection for determining variable importances.

Appendix B

Neural Networks

B.1 Motivation

The human brain processes information using a collection of interconnected cells called neurons, arranged in a neural network (NN). A topic of interest in the field of artificial intelligence is recreating intelligent behavior by modeling the low-level structure of the brain. The estimated number of neurons in the human brain is 10^{11} and they communicate using electrochemical signals. It is apparent that simulation of the full brain is not yet within reach due to computational constraints. However, simulations on a smaller scale proved sufficient to yield an interesting modeling tool with broad applicability, called artificial neural network (NN, the 'artificial' is implied).

For extensive literature on NN we refer to [10, 25, 36, 86]. A comparative predictive analysis of NN, non-linear regression an CART is provided in [93], including numerous applications and background information. A survey of NN for classification is given in [123]. The link between NN and traditional statistical models is examined in [96].

B.2 Algorithm

The behavior of biological neural networks is modeled using perceptrons, which are a simplified artificial representation of a biological neuron. Perceptrons are simple computational units taking multiple inputs. They will compute the net input by making a linear combination of all inputs. The coefficients of this linear combination are called the weight of that connection or also connection strength. Remark that weights can be negative, this is known in biology as inhibitory neurons eg. a neuron inhibiting other neurons to become active. An activation function is applied to the net input yielding the output. An artificial neural network is a collection of artificial neurons with a specific structure.

B.2.1 Topology and Activation Function

The expressiveness of the NN model is determined by the structure of the NN and the choice of activation function. The structure can be layered, hierarchical or hybrid, we discuss the layered variant called multi layer perceptron (MLP) introduced in [88]. The standard topology consists of three layers, the input layer, hidden layer and output layer. The activation function can be any arbitrary function, but several conventional choices are known to exhibit specific

model properties. The most simple NN consists of the inputs connected to just one perceptron with a linear activation function, as shown in Figure B.1. Remark that optimizing the weights yield a linear regression model. The single perceptron can be expanded to proper layer(s) and could use a non-linear activation function, also shown in Figure B.1. A commonly used activation function is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-\beta x}}, \quad \text{where } \beta \text{ is a slope parameter} \tag{B.1}$$

When using a topology with just a single hidden layer, NN will produce nonlinear regression models. More complex topologies include multiple hidden layers. The choice of the number of hidden layers and the number of perceptrons per layer are important issues and discussed in [10, 36].

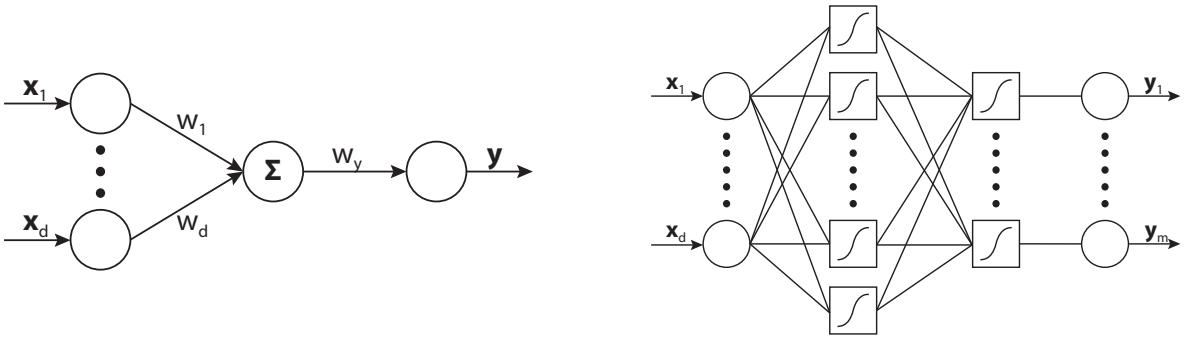


Figure B.1: Examples of possible topologies and activation function. On the left there is only one hidden layer and the output will be predicted using a linear combination of inputs, as determined by the weights w_i. On the right a more complex topology using a sigmoid activation function as described by Equation B.1

B.2.2 Model Training

Consistent with the biological counterpart a NN is said to be trained, implying gradual learning. In practice this often means that data points or observations are presented to the NN in sequence. The NN will then gradually optimize the connection strength between neurons to achieve a configuration yielding the most accurate prediction. Several training algorithms exist to adapt the connection weights when a training input is presented to the NN, but since this is not the focus of this thesis we refer to literature on back-propagation [86, 36, 25], Levenberg-Marquardt [33], conjugate gradient descent [79, 47, 19].

One thing all the learning algorithms have in common is that they aim to find the global minimum of the error surface of the NN. The error of a NN is determined by an error function, commonly chosen to be the SSE as introduced in Equation 1.6. One could view the error of a NN as a function of the free parameters in the model, described by the connection weights. So by the relation between regression and optimization, this multi-dimensional error function can be optimized to find an optimal NN model. Unfortunately it might be difficult to find the global minimum for this error surface since local optima can be present as well. This is a typical problem in multi-dimensional optimization where the minimum is lower than its surrounding neighborhood but above the global minimum, and most algorithms are not able to find the global minimum if they find a local minimum first.

The initial connection weights are chosen randomly, determining the starting position of the optimization. The learning algorithm will then try to find a minimum, and will possibly depend on the order in which the observations or data points are presented. Remark that a single training run does not give much guarantee in terms of prediction accuracy. Therefore, multiple NN are trained starting from different configurations of initial weights, division of data into training and test set, and training data ordering. The best models obtained by repeating this process can be combined in an ensemble to provide robust results [34].

B.3 Variable Importance

Many FS techniques are developed by the NN community, an overview is given in [67, 114]. A multi layer perceptron based dimensionality reduction technique is proposed in [73], while a genetic algorithm is used in [66] to determine optimal subsets. We will consider model dependent techniques, and summarize only select methods.

Since NN do also model non-linear behavior, inter-dependent variables can be handled by the modeling technique. Any FS technique applied on NN must not make the assumption that a linear model is sufficient, for example using individual linear correlations as a dependence measure is not enough. In practice this means that variables will have to be selected in sequence, and the model rebuilt after each removal to allow the model to take into account some of the dependencies.

B.3.1 Using Model Structure

When NN are used for non-linear modeling, the weights are no longer easy to interpret. A heuristic proposed in [119] exploits both the weight value and the network topology, describing the 'saliency' of a variable i:

$$S_i = \sum_{j \in H} \left(\frac{|w_{ji}|}{\sum_{i' \in I} |w_{ji'}|} \sum_{k \in O} \frac{|w_{kj}| H}{\sum_{j' \in H} |w_{kj'}|} \right) \tag{B.2}$$

where I, H, O denote respectively the input, hidden and output layer. Without loss of generality, suppose that each hidden and output node incoming weight vector sums to one. Then the previous equation can be written as:

$$S_i = \sum_{j \in H} \sum_{o \in O} |w_{oj} w_{ji}| \tag{B.3}$$

It is now easier to see that S_i for output o is the sum of the absolute values of contributions over all paths from i to o. Each individual path receives a score of the product of the weights from input i to hidden unit j and from j to o. Remark that this measure will depend on the magnitude of the input, so all variables should be in a similar range.

B.3.2 Using Model Error

Here the relevance of a variable is determined by the change in prediction error caused by fixing its value to the experimental mean, and varying other parameters. In order for this approach to be applicable, variables should be independent and the training set should be

representative of the input space. Saliency based pruning is a backward method introduced in [82] using the variation of the learning error as evaluation criterion:

$$S_i = MSE - MSE(\overline{x}_i), \quad \text{where } MSE(\overline{x}_i) = \frac{1}{n}\sum_{l=1}^{n}(\hat{f}(x_1^l, \ldots, \overline{x}_i, \ldots, x_d^l) - y^l)^2 \qquad \text{(B.4)}$$

Where $\overline{x}_i = \sum_{l=1}^{n} x_i^l/n$. Variables are then eliminated in the increasing order of S_i, thus eliminating variables with low impact first. Remark that this method does not take into account correlated variables.

B.4 Properties

B.4.1 Model Evaluation

The NN training approach is sensible in case of real time observations, as is typical for the brain of living organisms. The model is slowly adapted in time to accommodate for new conditions, in contrast to the traditional statistical problems where all data is known in advance. Suppose we are in the traditional setting where all data is available prior to model building. In order to 'teach' the NN the relation between the input and output variables, the data set is split into a training and test set. The training set will be used to optimize the connection weights, while the test set will serve to validate the generated model on data the NN did not see before. In other words it will check the ability of the NN model to generalize to new cases. This is a crucial step in providing reliable models since the model might have a low prediction error but over-fits the training set, in which case the model did not capture the more general trend of the data. This often hints at an overly complex topology, and better results could be obtained with smaller networks.

B.4.2 Data Distribution

The number of training data required to achieve a good model depends on several factors such as the number of weights to optimize and the unknown complexity of the underlying function. In addition the data must be representative to ensure the model will not be biased. This is related to the problem of imbalanced data as described in Section 1.2.12.5. The effect is explained using a classification example: If a NN would be trained to recognize unhealthy or healthy patients and the actual population has a ratio of 10% unhealthy and 90% healthy patients, the training data should reflect this ratio known as the prior probability. Suppose the training data would contain 50% unhealthy and 50 % healthy patients. The NN model would predict more patients to be unhealthy than the actual 10-90 ratio, since this allows for a lower overall prediction error. Another aspect to keep in mind is that NN in general tend to extrapolate poorly. Since weights are only adapted for data known to the network, there is no way to incorporate knowledge about areas outside the domain of the training data.

B.4.3 Related Methods

Other approaches to NN exist as well. Most notably radial basis function networks (RBFN) [14, 81, 36], probabilistic neural networks (PNN) [103] and generalized regression neural networks (GRNN) [104], all resembling kernel-based approximation. The activation function is commonly chosen to be Gaussian. Conceptually a Gaussian curve is positioned around

a center in the input space for each neuron in the network. The centers are determined by cluster analysis, or coincide with a random subset of input points. A curve positioned that way can be thought of as expressing confidence in a local neighborhood: it is unlikely the response will differ much in close proximity of a known solution.

In unsupervised learning a self organizing feature map (SOFM) is a type of NN designed for unsupervised learning, where the network attempts to learn the structure of the data. It is not within the scope of this thesis to elaborate on this topic, but we refer to the literature [56, 57].

B.5 Applications

While NN are capable of prediction, they are also often used in context of pattern recognition and classification. For example in the medical sciences NN are employed to aid diagnoses [26]. Many applications in the financial sector NN are discussed in [112]. Matching images is also a popular application of NN. Here the NN is trained to recognize specific traits of an image, for example in context of facial recognition [65]. Other applications include the estimation of wind turbine power curves [68] and time series forecasting [29].

B.6 Summary

In this chapter we have shown that neural networks (NN) are artificial representations of biological neural networks. A model is represented by interconnected nodes, where each connection has an associated weight and each node performs a function combining all its inputs into one output. The modeling process consists of finding optimal connection weights as to minimize the prediction error. Models are said to be trained, typically by presenting the training data sequentially and gradually modifying the connection weights using a learning algorithm. The network topology determines the complexity of the functions that can be expressed. We note that Variable importance measures for NN can either use the model structure, or the model error.

Appendix C

Variable Types

Formally we can distinguish between four types of variables: nominal, ordinal, interval and ratio variable types. Each variable type has a set of applicable operations. The operations of interest are: distinctness ($=$ and $\neq$), order ($<$, $\leq$, $>$ and $\geq$), addition ($+$ and $-$), multiplication ($*$ and $/$). Remark that these operations are inclusive, that is to say if it is possible to use multiplication on a variable then surely all other operators must be defined as well.

Nominal variables can only be distinguished from another, while ordinal variables can be ordered as well. Such variables are referred to as categorical variables, and lack properties of numbers. Even though they might be represented by a number, they should not be handled as one. Numeric variables can be either interval variables or ratio variables. For an interval variable only the differences between values are meaningful, so plus and minus operators are allowed. As the name suggests, the ratio of values is also meaningful for variables of the ratio variable type. This extends the allowed operations by the division and multiplication.

In addition variables can be either discrete or continuous, describing the number of output values that variable can take. A discrete variable only has a finite (or countably infinite) number of output values. Remark that both categorical and numeric variables can be discrete. A continuous variable is one whose values are numbers in $\mathbb{R}$, implying that only numeric variables can be continuous.

Due to the limitation in the applicable operators on this different variable types, they also differ in permissible transformations. With permissible transformations we refer to transformations that do not change the meaning of the variable. For example the same volume could be measured in liters or gallons, and the conversion from liter to gallons or vice versa is a permissible transformation.

A nominal variable can only be transformed by a one-to-one mapping since such mapping only requires the distinctness property. A transformation on ordinal variables should preserve the order, so if the transformation is a function it must be monotonic. Transformations on interval variables must also preserve the meaning of addition and subtraction. For example converting from temperature in Celsius to Fahrenheit requires both a scaling and translation. Ratio variables can be scaled by a constant factor with preservation of their ratio.

Bibliography

[1] Gnuplot. URL http://www.gnuplot.info/.

[2] Russ Abbott, Behzad Parviz, and Chengyu Sun. Genetic programming reconsidered. In Hamid R. Arabnia and Youngsong Mun, editors, *Proceedings of the International Conference on Artificial Intelligence, Volume 2 & Proceedings of the International Conference on Machine Learning; Models, Technologies & Applications, MLMTA '04*, volume 2, pages 1113–1116, Las Vegas, Nevada, USA, June 2004. CSREA Press. ISBN 1-932415-32-7. URL http://abbott.calstatela.edu/PapersAndTalks/GeneticProgrammingReconsidered.pdf.

[3] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pages 420–434, London, UK, 2001. Springer-Verlag. ISBN 3-540-41456-8. URL http://portal.acm.org/citation.cfm?id=645504.656414.

[4] Evolved Analytics. Datamodeler package. URL http://www.evolved-analytics.com/.

[5] Kellie J. Archer and Ryan V. Kimes. Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52:2249–2260, January 2008. ISSN 0167-9473. doi: 10.1016/j.csda.2007.08.015. URL http://portal.acm.org/citation.cfm?id=1316079.1316183.

[6] A. F. Ashour, L. F. Alvarez, and V. V. Toropov. Empirical modelling of shear strength of rc deep beams by genetic programming. *Computers & Structures*, 81(5):331 – 338, 2003. ISSN 0045-7949. doi: DOI:10.1016/S0045-7949(02)00437-6. URL http://www.sciencedirect.com/science/article/pii/S0045794902004376.

[7] V. Belle, T. Deselaers, and S. Schiffer. Randomized trees for real-time one-step face detection and recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, 2008. doi: 10.1109/ICPR.2008.4761365.

[8] Richard A. Berk. *Statistical Learning from a Regression Perspective*. Springer Series in Statistics. Springer, New York, 2008. ISBN 978-0-387-77500-5. doi: 10.1007/978-0-387-77501-2.

[9] G. Biau. Analysis of a random forests model. *ArXiv e-prints*, 2010. URL http://adsabs.harvard.edu/abs/2010arXiv1005.0208B.

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

[11] Leo Breiman. Original fortran implementation of random forests. URL http://www.stat.berkeley.edu/~breiman/RandomForests/reg_software.htm.

[12] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, August 1996. ISSN 0885-6125. doi: 10.1023/A:1018054314350. URL http://portal.acm.org/citation.cfm?id=231986.231989.

[13] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1010933404324. 10.1023/A:1010933404324.

[14] D. S. Broomhead and D. Lowe. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[15] Alexandre Bureau, Josee Dupuis, Kathleen Falls, Kathryn L Lunetta, Brooke Hayward, Tim P Keith, and Paul Van Eerdewegh. Identifying snps predictive of phenotype using random forests. *Genetic epidemiology*, 28:171–82, 2005. doi: 10.1002/gepi.20041.

[16] J. Burez and D. Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3, Part 1):4626–4636, 2009. ISSN 0957-4174. doi: DOI:10.1016/j.eswa.2008.05.027. URL http://www.sciencedirect.com/science/article/B6V03-4SHMCF0-1/2/3a0c963f431e508849f011697faba9d0.

[17] Jin-Ding Cai and Ren-Wu Yan. Fault diagnosis of power electronic circuit based on random forests algorithm. In *Natural Computation, 2009. ICNC '09. Fifth International Conference on*, pages 214–217, 2009. doi: 10.1109/ICNC.2009.390.

[18] Duygu Çalisir and Esin Dogantekin. A new intelligent hepatitis diagnosis system: Pca-lssvm. *Expert Systems with Applications*, 38(8):10705 – 10708, 2011. ISSN 0957-4174. doi: DOI:10.1016/j.eswa.2011.01.014. URL http://www.sciencedirect.com/science/article/B6V03-523V3B5-1/2/80607f03fddc0557dbe64e762934f435.

[19] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *Circuits, Devices and Systems, IEEE Proceedings G*, 139(3):301–310, 1992.

[20] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6:1–6, June 2004. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1007730.1007733. URL http://doi.acm.org/10.1145/1007730.1007733.

[21] G. Chen and S. Qian. Denoising of hyperspectral imagery using principal component analysis and wavelet shrinkage. *Geoscience and Remote Sensing, IEEE Transactions on*, 49(3):973–980, march 2011. ISSN 0196-2892. doi: 10.1109/TGRS.2010.2075937.

[22] Akim Demaille, Joel E. Denny, and Paul Eggert. Bison - gnu parser generator. URL http://www.gnu.org/software/bison/.

[23] Ramon Diaz-Uriarte and Sara Alvarez de Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006. ISSN 1471-2105. doi: 10.1186/1471-2105-7-3. URL http://www.biomedcentral.com/1471-2105/7/3.

[24] Evolved Analytics LLC. *DataModeler Release 1.0*. Evolved Analytics LLC, 2010. URL www.evolved-analytics.com.

[25] L. Fausett. *Fundamentals of Neural Networks*. Prentice Hall, New York, 1994.

[26] Hiroshi Fujita, Tetsuro Katafuchi, Toshiisa Uehara, and Tsunehiko Nishimura. Application of artificial neural network to computer-aided diagnosis of coronary artery disease in myocardial spect bull's-eye images. *Journal of Nuclear Medicine*, 33(2):272–276, 1992. URL http://jnm.snmjournals.org/cgi/content/abstract/33/2/272.

[27] Mike Gashler, Christophe Giraud-Carrier, and Tony Martinez. Decision Tree Ensemble: Small Heterogeneous Is Better Than Large Homogeneous. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 900–905. IEEE, December 2008. ISBN 978-0-7695-3495-4. doi: 10.1109/ICMLA.2008.154. URL http://dx.doi.org/10.1109/ICMLA.2008.154.

[28] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. *Pattern Recogn. Lett.*, 31:2225–2236, October 2010. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2010.03.014. URL `http://dx.doi.org/10.1016/j.patrec.2010.03.014`.

[29] Iffat A. Gheyas and Leslie S. Smith. A neural network approach to time series forecasting. In *WCE 2009*, volume 2, July 2009.

[30] Malcolm Gladwell. The ketchup conundrum. *The New Yorker*, (6):128–135, September 2004.

[31] Ulrike Grömping. Variable importance assessment in regression: Linear regression versus random forest. *The American Statistician*, 64:308–319, November 2009. URL `http://pubs.amstat.org/doi/abs/10.1198/tast.2009.08199?journalCode=tas`.

[32] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435. URL `http://portal.acm.org/citation.cfm?id=944919.944968`.

[33] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993, 1994.

[34] L.K. Hansen and P. Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):993–1001, 1990. doi: 10.1109/34.58871.

[35] R. Hariharan, I. Kiss, and I. Viikki. Noise robust speech parameterization using multiresolution feature extraction. *Speech and Audio Processing, IEEE Transactions on*, 9(8):856 –865, nov 2001. ISSN 1063-6676.

[36] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[37] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21:1263–1284, 2009. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2008.239.

[38] Y.-C. Ho, C. G. Cassandras, C.-H. Chen, and L. Dai. Ordinal optimisation and simulation. *The Journal of the Operational Research Society*, 51(4):490–500, Apr. 2000. URL `http://www.jstor.org/stable/254177`.

[39] Yu-Chi Ho, Qian-Chuan Zhao, and Qing-Shan Jia. *Ordinal Optimisation - Soft Optimization for Hard Problems*. Springer, 2007. ISBN 978-1-4419-4243-2.

[40] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933. ISSN 0022-0663. doi: DOI:10.1037/h0071325.

[41] Ping-Cheng Hsieh and Pi-Cheng Tung. A novel hybrid approach based on sub-pattern technique and whitened pca for face recognition. *Pattern Recognition*, 42(5):978–984, 2009. ISSN 0031-3203. doi: DOI:10.1016/j.patcog.2008.09.024. URL `http://www.sciencedirect.com/science/article/B6V14-4TNWGS6-2/2/2f4e66bbab8ffae715217839918cbc98`.

[42] Jeih-Weih Hung and Lin-Shan Lee. Optimization of temporal filters for constructing robust features in speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(3):808–832, may 2006. ISSN 1558-7916. doi: 10.1109/TSA.2005.857801.

[43] Intel Corp. Thread building blocks. URL `http://threadingbuildingblocks.org/`.

[44] Hemant Ishwaran. Variable importance in binary regression trees and forests. *Electronic Journal of Statistics*, 1:519–537, 2007. URL `http://arxiv.org/abs/0711.2434v1`.

[45] J. E. Jackson. *A user's guide to principal components*. John Wiley & Sons, Inc., 1991.

[46] Abhishek Jaiantilal. Implementation of random forests with matlab interface. URL `http://code.google.com/p/randomforest-matlab/`.

[47] E. M. Johansson, F. U. Dowla, and D. M. Goodman. Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, 2(4):291–301, 1991. doi: 10.1142/S0129065791000261.

[48] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2nd edition, 2002. ISBN 978-0-387-95442-4. doi: 10.1007/b98835.

[49] Elsie M. Jordaan. *Development of Robust Inferential Sensors : industrial application of support vector machines for regression*. PhD thesis, Technische Universiteit Eindhoven, December 2002.

[50] A. Kalos, A. Kordon, G. Smits, and S. Werkmeister. Hybrid model development methodology for industrial soft sensors. In *American Control Conference, 2003. Proceedings of the 2003*, volume 6, pages 5417 – 5422, june 2003. doi: 10.1109/ACC.2003.1242590.

[51] Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *EuroGP'2003*, volume 2610 of *Lecture Notes in Computer Science*, pages 71–83. Springer-Verlag, 2003.

[52] Maarten Keijzer. Scaled symbolic regression. *Genetic Programming and Evolvable Machines*, 5: 259–269, September 2004. ISSN 1389-2576. doi: 10.1023/B:GENP.0000030195.77571.f9. URL `http://portal.acm.org/citation.cfm?id=996967.996974`.

[53] Jr. Kenneth E. Kinnear. Generality and difficulty in genetic programming: Evolving a sort. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 287–294, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-299-2. URL `http://portal.acm.org/citation.cfm?id=645513.657759`.

[54] Kenneth.E. Jr. Kinnear. Evolving a sort: lessons in genetic programming. In *Neural Networks, IEEE International Conference on*, volume 2, pages 881–888, 1993. doi: 10.1109/ICNN.1993.298674.

[55] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997. ISSN 0004-3702. Special issue on relevance.

[56] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982. ISSN 0340-1200. doi: 10.1007/BF00337288.

[57] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325.

[58] A.K. Kordon, A.N. Kalos, F.A. Castillo, E.M. Jordaan, G.F. Smits, and M.E. Kotanchek. Competitive advantages of evolutionary computation for industrial applications. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 166–173, sept. 2005. doi: 10.1109/CEC.2005.1554681.

[59] Arthur Kordon, Elsa Jordaan, Lawrence Chew, Guido Smits, Torben Bruck, Keith Haney, and Annika Jenings. Biomass inferential sensor based on ensemble of models generated by genetic programming. In *Genetic and Evolutionary Computation – GECCO 2004*, volume 3103 of *Lecture Notes in Computer Science*, pages 1078–1089. Springer Berlin / Heidelberg, 2004. URL `http://dx.doi.org/10.1007/978-3-540-24855-2_118`. 10.1007/978-3-540-24855-2_118.

[60] John R. Koza. *Genetic Programming - On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[61] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.

[62] John R. Koza, F. H. Bennet, D. Andre, and M. A. Keane. Genetic programming iii: Darwinian invention and problem solving. *Evolutionary Computation*, 7(4):451–453, 1999.

[63] John R. Koza, M. A. Keane, M. J. Streeter, W. Mydiowec, J. Yu, and G. Lanza. *Genetic Programming IV - Routine Human-Competitive Machine Intelligence*, volume 5 of *Genetic Programming*. Springer, 2003. ISBN 978-0-387-25067-0.

[64] Fatma Latifoglu, Kemal Polat, SadIk Kara, and Salih Günes. Medical diagnosis of atherosclerosis from carotid artery doppler signals using principal component analysis (pca), k-nn based weighting pre-processing and artificial immune recognition system (airs). *Journal of Biomedical Informatics*, 41(1):15–23, 2008. ISSN 1532-0464. doi: DOI:10.1016/j.jbi.2007.04.001. URL `http://www.sciencedirect.com/science/article/B6WHD-4NG3TC1-1/2/fddbbf4f8bae4cfa3208bcec7629146e`.

[65] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, jan 1997. ISSN 1045-9227. doi: 10.1109/72.554195.

[66] Sergio Ledesma, Gustavo Cerda, Gabriel Aviña, Donato Hernández, and Miguel Torres. Feature selection using artificial neural networks. In Alexander Gelbukh and Eduardo Morales, editors, *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pages 351–359. Springer Berlin / Heidelberg, 2008. doi: 10.1007/978-3-540-88636-5_34. URL `http://dx.doi.org/10.1007/978-3-540-88636-5\\_34`.

[67] Philippe Leray and Patrick Gallinari. Feature selection with neural networks. *Behaviormetrika*, 26: 16–6, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.4570`.

[68] Shuhui Li, Donald C. Wunsch, Edgar O'Hair, and Michael G. Giesselmann. Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. *Journal of Solar Energy Engineering*, 123(4):327–332, 2001. doi: 10.1115/1.1413216. URL `http://link.aip.org/link/?SLE/123/327/1`.

[69] Andy Liaw. Implementation of random forests in r. URL `http://cran.r-project.org/web/packages/randomForest/index.html`.

[70] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101:578–590, June 2006. URL `http://pubs.amstat.org/doi/abs/10.1198/016214505000001230`.

[71] Roderick J. A. Little. Regression with missing x's: A review. *Journal of the American Statistical Association*, 87(420):1227–1237, 1992. ISSN 01621459. URL `http://www.jstor.org/stable/2290664`.

[72] Boyang Liu, Raphael T. Haftka, Mehmet A. Akgün, and Akira Todoroki. Permutation genetic algorithm for stacking sequence design of composite laminates. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):357–372, 2000. ISSN 0045-7825. doi: 10.1016/S0045-7825(99)90391-2. URL `http://www.sciencedirect.com/science/article/pii/S0045782599903912`.

[73] R. Lotlikar and R. Kothari. *Multilayer perceptron based dimensionality reduction*, volume 3, pages 1691–1695. IEEE, 2002.

[74] Guan-Chun Luh and Chun-Yi Lin. Pca based immune networks for human face recognition. *Applied Soft Computing*, 11(2):1743–1752, 2011. ISSN 1568-4946. doi: DOI:10.1016/j.asoc.2010.05.017. URL `http://www.sciencedirect.com/science/article/B6W86-508607C-2/2/94182e6d9cbf2e9222117c33a26775e5`. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[75] Lunetta, L Brooke Hayward, Jonathan Segal, and Paul Van Eerdewegh. Screening large-scale association study data: exploiting interactions using random forests. *BMC Genetics*, 5(1):32, 2004. ISSN 1471-2156. doi: 10.1186/1471-2156-5-32. URL `http://www.biomedcentral.com/1471-2156/5/32`.

[76] M. Maragoudakis, E. Loukis, and P.-P. Pantelides. Random forests identification of gas turbine faults. In *Systems Engineering, 2008. ICSENG '08. 19th International Conference on*, pages 127–132, 2008. doi: 10.1109/ICSEng.2008.81.

[77] Trent McConaghy. *Latent Variable Symbolic Regression for High-Dimensional Inputs*, pages 103–118. Springer, 2010. doi: 10.1007/978-1-4419-1626-6_7.

[78] Md. Khademul Islam Molla and Keikichi Hirose. Single-mixture audio source separation by subspace decomposition of hilbert spectrum. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):893–900, march 2007. ISSN 1558-7916. doi: 10.1109/TASL.2006.885254.

[79] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993. doi: DOI:10.1016/S0893-6080(05)80056-5.

[80] A. Montillo and Haibin Ling. Age regression from faces using random forests. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2465–2468, 2009. doi: 10.1109/ICIP.2009.5414103.

[81] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.

[82] John E. Moody and Joachim Utans. Principled architecture selection for neural networks: Application to corporate bond rating prediction. In *NIPS'91*, pages 683–690, 1991.

[83] United Nations. Human development reports, . URL `http://www.hdr.undp.org/`.

[84] United Nations. Human development research papers, . URL `http://hdr.undp.org/en/reports/global/hdr2010/papers/`.

[85] Y. Panagakis, C. Kotropoulos, and G.R. Arce. Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):576 –588, march 2010. ISSN 1558-7916. doi: 10.1109/TASL.2009.2036813.

[86] D. Patterson. *Artificial Neural Networks*. Prentice Hall, Singapore, 1996.

[87] Vern Paxson. Flex - the fast lexical analyzer. URL `http://flex.sourceforge.net/`.

[88] CORPORATE PDP Research Group. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. MIT Press, Cambridge, MA, USA, 1986.

[89] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901. URL `http://stat.smmu.edu.cn/history/pearson1901.pdf`.

[90] M. Pechenizkiy, A. Tsymbal, and S. Puuronen. Pca-based feature transformation for classification: issues in medical diagnostics. In *Computer-Based Medical Systems, 2004. CBMS 2004. Proceedings. 17th IEEE Symposium on*, pages 535 – 540, june 2004. doi: 10.1109/CBMS.2004.1311770.

[91] Cassio L. Pennachin, Moshe Looks, and João A. de Vasconcelos. Robust symbolic regression with affine arithmetic. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 917–924, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0072-8. doi: http://doi.acm.org/10.1145/1830483.1830648. URL `http://doi.acm.org/10.1145/1830483.1830648`.

[92] Kemal Polat and Salih Günes. Automatic determination of diseases related to lymph system from lymphography data using principles component analysis (pca), fuzzy weighting preprocessing and anfis. *Expert Systems with Applications*, 33(3):636 – 641, 2007. ISSN 0957-4174. doi: DOI:10.1016/j.eswa.2006.06.004. URL `http://www.sciencedirect.com/science/article/B6V03-4KBVRSB-1/2/a7e2f50bb67db4993002d309402d67d1`.

[93] Muhammad A. Razi and Kuriakose Athappilly. A comparative predictive analysis of neural networks (nns), nonlinear regression and classification and regression tree (cart) models. *Expert Syst. Appl.*, 29:65–74, July 2005. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2005.01.006. URL `http://dx.doi.org/10.1016/j.eswa.2005.01.006`.

[94] R. P. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997. doi: doi:10.1162/evco.1997.5.2.123. URL `ftp://ftp.idsia.ch/pub/rafal/PIPE.ps.gz`.

[95] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. Springer, 2003. ISBN 978-0-387-95420-2.

[96] Warren S. Sarle. Neural networks and statistical models. In *SUGI'94*, 1994.

[97] G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley, New York, 1989.

[98] Jonathon Shlens. A tutorial on principal component analysis, April 2009. URL `http://www.snl.salk.edu/~shlens/pca.pdf`.

[99] David S. Siroky. Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163, 2009. URL `http://www.i-journals.org/ss/viewarticle.php?id=33`.

[100] G. Smits and E. Vladislavleva. Ordinal pareto genetic programming. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3114 –3120, 2006. doi: 10.1109/CEC.2006.1688703.

[101] Guido Smits and Mark Kotanchek. Pareto-front exploitation in symbolic regression. In Una-May O'Reilly, Tina Yu, Rick L. Riolo, and Bill Worzel, editors, *Genetic Programming Theory and Practice II*, chapter 17, pages 283–299. Springer, Ann Arbor, 13-15 May 2004. ISBN 0-387-23253-2.

[102] Guido Smits, Arthur Kordon, Katherine Vladislavleva, Elsa Jordaan, and Mark Kotanchek. Variable selection in industrial datasets using pareto genetic programming. In John Koza, Tina Yu, Rick Riolo, and Bill Worzel, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, pages 79–92. Springer US, 2006. ISBN 978-0-387-28111-7. URL `http://dx.doi.org/10.1007/0-387-28111-8_6`. 10.1007/0-387-28111-8_6.

[103] Donald F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990. doi: 10.1016/0893-6080(90)90049-Q.

[104] Donald F. Specht. A general regression neural network. *Neural Networks, IEEE Transactions on*, 2:568–576, 1991. doi: 10.1109/72.97934.

[105] Lee Spector. *Towards Practical Autoconstructive Evolution: Self-Evolution of Problem-Solving Genetic Programming Systems*, volume 8, chapter 2, pages 17–33. Springer, 2011. ISBN 978-1-4419-7746-5. doi: 10.1007/978-1-4419-7747-2_2.

[106] Carolin Strobl, Anne L. Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1): 25+, January 2007. ISSN 1471-2105. doi: 10.1186/1471-2105-8-25. URL `http://dx.doi.org/10.1186/1471-2105-8-25`.

[107] Carolin Strobl, Anne L. Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):307+, July 2008. ISSN 1471-2105. doi: 10.1186/1471-2105-9-307. URL `http://dx.doi.org/10.1186/1471-2105-9-307`.

[108] K. V. Sujatha and S. Meenakshi Sundaram. A combined pca-mlp model for predicting stock index. In *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*, A2CWiC '10, pages 17:1–17:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0194-7. URL `http://doi.acm.org/10.1145/1858378.1858395`.

[109] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson Education, 2006. ISBN 0-321-42052-7.

[110] Astro Teller and David Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 321–328, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann. URL `http://www.cs.cmu.edu/afs/cs/usr/astro/public/papers/GR.ps`.

[111] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1): 267–288, 1996.

[112] Robert R. Trippi and Efraim Turban, editors. *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*. McGraw-Hill, Inc., New York, NY, USA, 1992. ISBN 1557384525.

[113] Edwin van Dam, Dick den Hertog, Bart Husslage, and Gijs Rennen. Space-filling designs, 2009. URL `www.spacefillingdesigns.nl`.

[114] A. Verikas and M Bacauskiene. Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335, 2002. URL `http://scholar.google.com/scholar?cluster=11273977636929002914`.

[115] Katya Vladislavleva. Symbolic regression. URL `http://www.symbolicregression.com/`.

[116] Katya Vladislavleva. *Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming*. PhD thesis, Tilburg University, 2008.

[117] Katya Vladislavleva, Kalyan Veeramachaneni, Matt Burland, Jason Parcon, and Una-May O'Reilly. Knowledge mining with genetic programming methods for variable selection in flavor design. In *GECCO*, pages 941–948, 2010.

[118] Huiyuan Wang, Yan Leng, Zengfeng Wang, and Xiaojuan Wu. Application of image correction and bit-plane fusion in generalized pca based face recognition. *Pattern Recognition Letters*, 28(16):2352–2358, 2007. ISSN 0167-8655. doi: DOI:10.1016/j.patrec.2007.07.015. URL `http://www.sciencedirect.com/science/article/B6V15-4PCGRPP-2/2/9823c85f150e1c2a2bd0979a943b50e4`.

[119] M Yacoub and Y Bennani. Hvs: A heuristic for variable selection in multilayer artificial neural network classifier. In *ANNIE'97*, pages 527–532, 1997.

[120] Weizhong Yan. Application of random forest to aircraft engine fault diagnosis. In *Computational Engineering in Systems Applications, IMACS Multiconference on*, volume 1, pages 468–475, 2006. doi: 10.1109/CESA.2006.4281698.

[121] Chen Yang, Laijun Lu, Heping Lin, Renchu Guan, Xiaohu Shi, and Yanchun Liang. A fuzzy-statistics-based principal component analysis (fs-pca) method for multispectral image enhancement and display. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(11):3937–3947, nov. 2008. ISSN 0196-2892. doi: 10.1109/TGRS.2008.2001386.

[122] Mao Ye, Zhang Yi, and JianCheng Lv. A globally convergent learning algorithm for pca neural networks. *Neural Computing & Applications*, 14:18–24, 2005. ISSN 0941-0643. URL `http://dx.doi.org/10.1007/s00521-004-0435-y`.

[123] Guoqiang Peter Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C, IEEE Transactions on*, pages 451–462, 2000.

[124] Lei Zhang, R. Lukac, Xiaolin Wu, and D. Zhang. Pca-based spatially adaptive denoising of cfa images for single-sensor digital cameras. *Image Processing, IEEE Transactions on*, 18(4):797–812, april 2009. ISSN 1057-7149. doi: 10.1109/TIP.2008.2011384.

[125] Hui Zou and Trevor Hastie. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005. URL `http://www.stat.purdue.edu/~tlzhang/mathstat/ElasticNet.pdf`.