FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Optimization of surrogate model settings using landscape analysis
Student:	Bc. Mikuláš Dvořák
Supervisor:	Ing. Zbyněk Pitra
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2020/21

Instructions

Approximation of the expensive black-box functions using regression models can effectively speed up the convergence of evolutionary algorithms. To achieve the most precise model predictions, it is necessary to make sure that the model parameters are set properly. Landscape analysis is considered to be one of the most convenient methods for such a task. However, the research in the area of surrogate model parameter settings for optimization of empirical functions is only starting.

1) Student will thoroughly study surrogate models and fitness landscape analysis (FLA) methodology.
2) He will measure FLA features on benchmark functions from the black-box optimization framework COCO in Matlab environment.
3) Student will train several types of classifiers for FLA feature classification considering the most convenient surrogate model settings.
4) Classifiers will be compared from the point of view of accuracy, classification correctness, and sensitivity.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 7, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Optimization of surrogate model settings using landscape analysis

Bc. Mikuláš Dvořák

Department of Applied Mathematics
Supervisor: Ing. Zbyněk Pitra

May 28, 2020

Acknowledgements

I would like to thank my supervisor Ing. Zbyněk Pitra for introducing me to this topic, for his patience, time, and valuable advice.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 28, 2020 . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Dvořák, Mikuláš. *Optimization of surrogate model settings using landscape analysis*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Tato práce se zaměřuje na výběr vhodného nastavení náhradních modelů v algoritmu DTS-CMA-ES pomocí *fitness landscape analysis*. Porovnává několik přístupů pro doporučování náhradních modelů pomocí různých statistických modelů. Mezi porovnané techniky patří klasifikace, klasifikace do více tříd a regrese. Pro každou techniku je použito několik modelů a jejich správnost klasifikace je porovnána pomocí přesnosti, senzitivity, specificity a F_1 skóre.

Klíčová slova CMA-ES, black-box optimalizace, náhradní model, selekce algoritmu, fitness landscape

Abstract

This thesis focuses on selecting the most suitable settings of surrogate models in DTS-CMA-ES algorithm using a fitness landscape analysis. Thesis compares different approaches for recommending the most suitable surrogate models with a variety of statistical models. Compared approaches are derived from classification, multi-label classification, and regression methods. For each method, few different statistical models are trained and their classification correctness is assessed with accuracy, sensitivity, specificity, and F_1 score.

Keywords CMA-ES, black-box optimization, surrogate model, algorithm selection, fitness landscape

Contents

List of Figures

List of Tables

Introduction

Optimization is a field of mathematics that has been studied for centuries. Many problems can be reduced into a problem of finding global optima of a function. Gradient descent methods or analytical solutions are often used to solve these problems.

Expensive black-box optimization is addressing optimization problems in settings where gradients are not available and evaluation of the optimized process costs valuable resources such as money or time. Stochastic methods are commonly used for such tasks.

The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) is a stochastic method suitable for optimization of a black-box function. *Surrogate model* is a regression model that can be used to accelerate the CMA-ES algorithm. Instead of evaluating the black-box function in every search point, the surrogate model is used to ease the number of expensive evaluations by approximating the underlying black-box function utilizing a regression model.

However, the combination of the CMA-ES with a surrogate model presents new challenges in tunning surrogate models to make the optimization more effective.

Fitness landscape analysis is a technique that is trying to characterize the structure of a fitness landscape. The fitness landscape structure can provide an important information about which surrogate model is most suitable for finding the global optima.

This thesis is addressing the problem of how to select the most convenient surrogate model in every generation of the surrogate assisted CMA-ES algorithm. To recommend the most convenient model, the fitness landscape analysis is used as a description of the space, where surrogate model is fitted.

To recommend a surrogate model, various classification strategies can be used, and by assessing their performance the most suitable classification model can be later utilized to make surrogate-assisted versions of the CMA-ES more effective.

The structure of this thesis is organized as follows. In Chapter 1, the theoretical background for surrogate assisted CMA-ES and the fitness landscape analysis is presented. Chapter 2 presents the design for algorithm selection and the design of classification models used for recommending the most convenient surrogate model. Finally, in Chapter 3, experiments with measured accuracies, a discussion about the experiments, and ideas for future work are presented.

Chapter **1**

Theoretical background

1.1 Black-box optimization

A *black-box function* is a function for which the analytical form is unknown. For example a program without known source code, a numerical method for partial differential equations, or a laboratory experiment [10].

Optimization is a field of mathematics studying the problem of finding global optima (the minimal or the maximal function value) of a function $f\colon \mathbb{X} \to \mathbb{R}$. The function f is often called the *objective function* and the set $\mathbb{X}$ is called the *input space.* In the case of finding the maximal function value, the optimization process is trying to find $\tilde{\boldsymbol{x}} \in \mathbb{X}$ such that $f(\tilde{\boldsymbol{x}}) \geq f(\boldsymbol{x})$ for every $\boldsymbol{x} \in \mathbb{X}$.

Black-box optimization (BBO) is defined as the study of design and analysis of algorithms that assume the objective and/or constraint functions are given by black-boxes [2]. The BBO problem can be even more challenging in the case when black-box functions are expensive to evaluate.

1.2 Surrogate modeling

Surrogate modeling is a technique based on building regression models of the original function using the already evaluated data points. This technique originated from response surface modeling where the regression models are usually simple polynomial models. Response surface modeling was introduced by George E. P. Box and K. B. Wilson in 1951 [6].

Surrogate modeling is often used for the optimization of a black-box function whose evaluations are expensive. It is used as a replacement for some evaluations and hence can lower the number of evaluations when searching for global optima.

The following definition is taken from the book [2] written by Charles Audet and Warren Hare.

The problem

$$\min_{\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^D} \left\{ \tilde{f}(\boldsymbol{x}) \mid \tilde{c}(\boldsymbol{x}) \leq 0 \right\}, \tag{1.1}$$

is said to be a surrogate for the problem

$$\min_{\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^D} \{ f(\boldsymbol{x}) \mid c(\boldsymbol{x}) \leq 0 \} \tag{1.2}$$

if $\tilde{f} \colon \mathbb{X} \to \mathbb{R} \cup \{\infty\}$ and $\tilde{c} \colon \mathbb{X} \to (\mathbb{R} \cup \{\infty\})^m$ share some similarities with f and c but are much faster to evaluate. The function $\tilde{f}$ and $\tilde{c}$ are said to be *surrogate functions* of the true functions f and c.

This thesis describes some of surrogate models used in the task of a single-objective continuous black-box optimization [41], namely, *polynomial* models, *Gaussian processes*, *artificial neural networks*, *support vector machines*, and *random forests*.

1.2.1 Low degree polynomials

This approach uses a polynomial function to model the true objective function. The smoothness of low degree polynomials might help with finding global optima when the underlying function is noisy. This section is based on [14].

The most common models for predictor $\tilde{f}$ of a given black-box function are *linear* and *quadratic*. The linear model is defined as

$$\tilde{f}(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{D} \beta_i x_i + \varepsilon, \tag{1.3}$$

and the quadratic model as

$$\tilde{f}(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{D} \beta_i x_i + \sum_{i=1}^{D} \beta_{ii} x_i^2 + \sum_{i=1}^{D} \sum_{j>i}^{D} \beta_{ij} x_i x_j + \varepsilon, \tag{1.4}$$

where ε is a random error which is assumed to be normally distributed with zero mean, β_0, β_i, β_{ii} and β_{ij} are unknown coefficients.

With enough sample points, the coefficients can be estimated with a standard method of the least squares. The estimation of $\boldsymbol{\beta}$ can be calculated with the equation

$$\boldsymbol{\beta} = \left(\boldsymbol{U}^T \boldsymbol{U} \right)^{-1} \boldsymbol{U}^T \boldsymbol{y}, \tag{1.5}$$

where

$$\boldsymbol{U} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_D^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_D^{(n)} \end{bmatrix}$$

for the linear model,

$$\boldsymbol{U} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_D^{(1)} & x_1^{(1)} x_2^{(1)} & \dots & x_{D-1}^{(1)} x_D^{(1)} & \left(x_1^{(1)}\right)^2 & \dots & \left(x_D^{(1)}\right)^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_D^{(n)} & x_1^{(n)} x_2^{(n)} & \dots & x_{D-1}^{(n)} x_D^{(n)} & \left(x_1^{(n)}\right)^2 & \dots & \left(x_D^{(n)}\right)^2 \end{bmatrix}$$

for the quadratic model, $x_i^{(j)}$ are values from domain space and $\boldsymbol{y}$ is a vector of measured responses.

1.2.2 Gaussian processes

Another family of models used for surrogate modeling are Gaussian processes.

In the book [43], written by Rasmussen et al., the Gaussian process is described as a generalization of the Gaussian probability distribution. However, instead of describing a random variable, it aims to describe a distribution of functions.

A Gaussian process is formally defined as a collection of random variables, any finite number of which have a joint Gaussian distribution.

Because a Gaussian process has a joint Gaussian distribution it is described by its mean and covariance function. The mean function $m(\boldsymbol{x})$ and the covariance function $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ of a real process $f(\boldsymbol{x})$ are defined as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}\left[f(\boldsymbol{x})\right], \\ \kappa(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}\left[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))^T\right] \end{aligned} \tag{1.6}$$

and these two functions describe the Gaussian process in the sense of the Gaussian distribution

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}')). \tag{1.7}$$

The posterior distribution can be inferred with rules for conditioning Gaussians as

$$\begin{aligned} p(\boldsymbol{f}^* | \boldsymbol{X}^*, \boldsymbol{X}, \boldsymbol{f}) = \mathcal{N}\left(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*\right), \\ \boldsymbol{\mu}^* &= \boldsymbol{\mu}(\boldsymbol{X}^*) + \boldsymbol{K}^{*^T} \boldsymbol{K}^{-1}(\boldsymbol{f} - \boldsymbol{\mu}(\boldsymbol{X})), \\ \boldsymbol{\Sigma}^* &= \boldsymbol{K}^{**} - \boldsymbol{K}^{*^T} \boldsymbol{K}^{-1} \boldsymbol{K}^*, \end{aligned} \tag{1.8}$$

where $\boldsymbol{f}$ is a vector of measured responses, $\boldsymbol{f}^*$ is a vector of estimations, $\boldsymbol{X}$ is a matrix with inputs of known responses, $\boldsymbol{X}^*$ is a matrix with inputs of unknown responses, $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{K}^*_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}^*_j)$ and $\boldsymbol{K}^{**}_{ij} = \kappa(\boldsymbol{x}^*_i, \boldsymbol{x}^*_j)$ for some covariance function κ.

The covariance function κ is often called a *kernel.* There is a large variety of available kernels.

There could be a very different outcome for different covariance functions, therefore, it is important to know which to choose as a prior. Figure 1.1 shows Gaussian processes realizations with different kernels. In sections below few popular kernels are described in more detail.

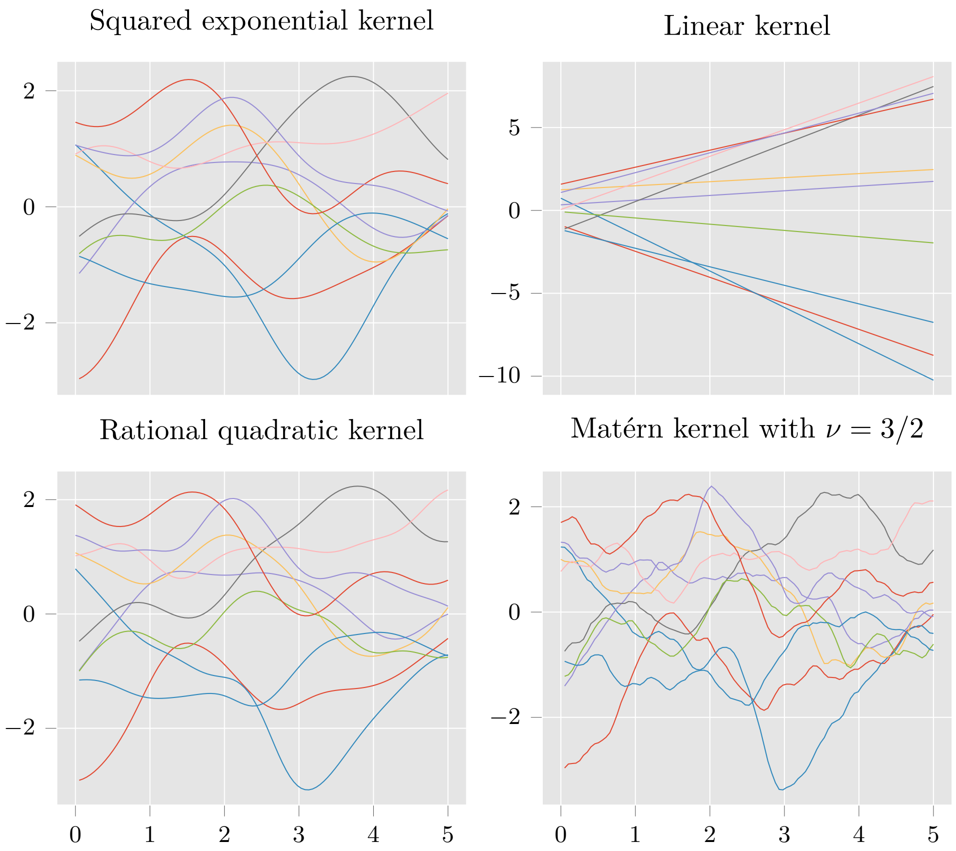

Figure 1.1: Examples of sampling functions from a prior of Gaussian processes with different kernels.

1.2.2.1 Polynomial kernel

Polynomial kernels are defined as follows:

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + \sigma_0^2)^p, \tag{1.9}$$

where $p \in \mathbb{N}$.

For $p = 1$ the kernel is called *linear* (LIN) and for $p = 2$ the kernel is *quadratic* (Q).

1.2.2.2 Squared exponential kernel

Squared exponential kernel (SE) is defined as follows:

$$\kappa_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\ell^2}\right), \tag{1.10}$$

where ℓ is the *characteristic length-scale*, a hyperparameter defining how far should vectors be in the input space to be uncorrelated in the output space. Figure 1.2 shows how changing the characteristic length-scale affects the prediction of GP with SE kernel.

1.2.2.3 Rational quadratic kernel

Rational quadratic kernel can be viewed as a generalization of SE kernel. The following equation defines the RQ kernel

$$\kappa_{\mathrm{RQ}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \left(1 + \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\alpha\ell^2}\right)^{-\alpha}. \tag{1.11}$$

The hyperparameter $\alpha > 0$ can be seen as a decomposition of the exponential function in SE kernel. This relation is described as

$$\lim_{\alpha \to +\infty} \sigma^2 \left(1 + \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\alpha\ell^2}\right)^{-\alpha} = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\ell^2}\right). \tag{1.12}$$

1.2.2.4 Matérn class kernels

This class of kernels was introduced by Matérn in 1960 [29] and is defined by the following equation.

$$\kappa_{\mathrm{Mat}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right), \tag{1.13}$$

where $\nu > 0, \ell > 0, K_\nu$ is a modified Bessel function.

There are two versions of this kernel which are often used in this field. The first is with $\nu = \frac{3}{2}$

$$\kappa_{\mathrm{Mat}}^{\frac{3}{2}}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \frac{\sqrt{3}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right) \exp\left(-\frac{\sqrt{3}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right) \tag{1.14}$$

and second with $\nu = \frac{5}{2}$

$$\kappa_{\mathrm{Mat}}^{\frac{5}{2}}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \frac{\sqrt{5}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell} + \frac{5\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\,\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\ell}\right). \tag{1.15}$$

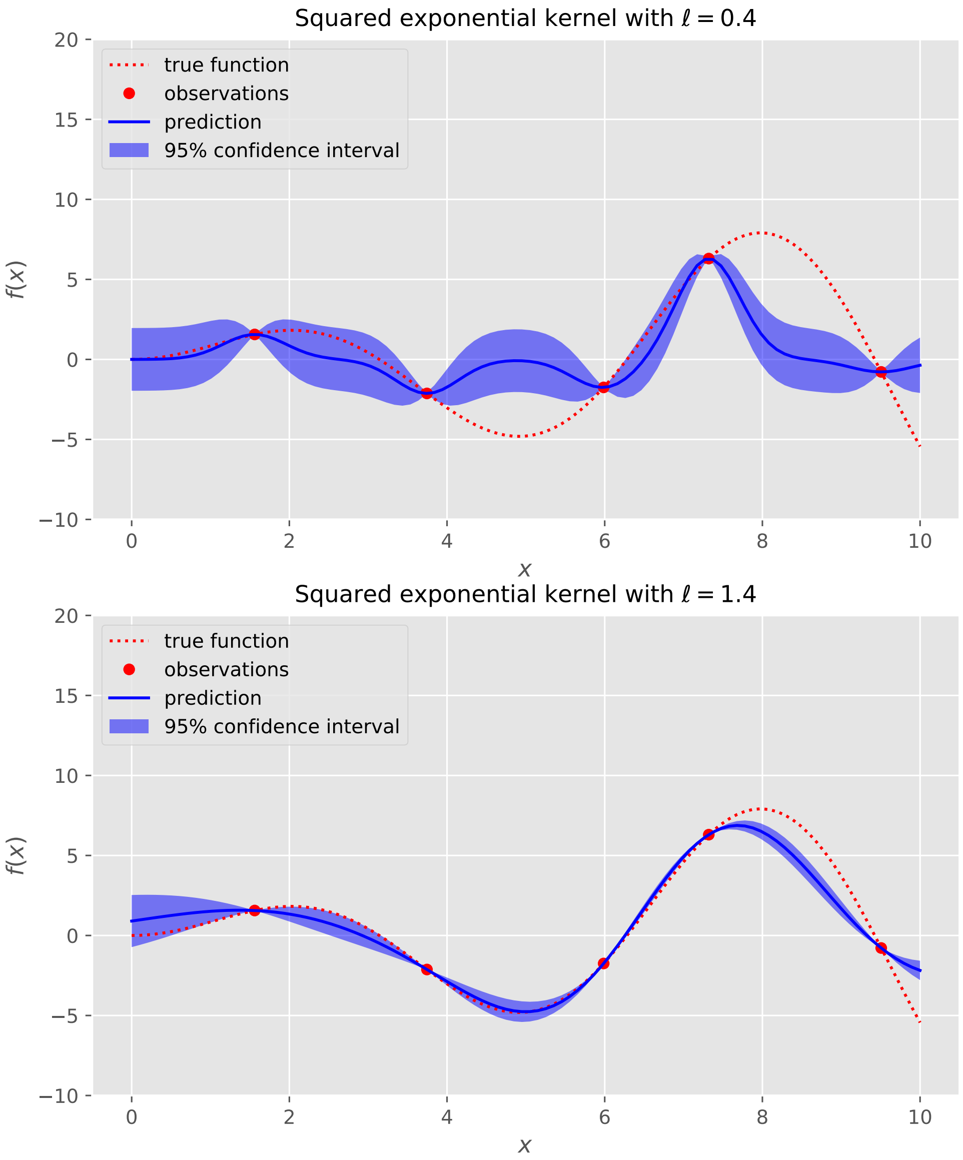

Figure 1.2: Gaussian process predictions using SE kernel for two different values of ℓ.

1.2.2.5 Gibbs kernel

Another kernel was introduced by Gibbs in his dissertation thesis in 1997 [13].

$$\kappa_{\text{Gibbs}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i=1}^{D} \left(\frac{2\ell_i(\boldsymbol{x})\ell_i(\boldsymbol{x}')}{\ell_i^2(\boldsymbol{x}) + \ell_i^2(\boldsymbol{x}')} \right)^{1/2} \exp\left(-\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{\ell_i^2(\boldsymbol{x}) + \ell_i^2(\boldsymbol{x}')} \right), \quad (1.16)$$

where ℓ_i is a positive function which can be different for each i and D is a number of dimensions of the vector $\boldsymbol{x}$. Making the hyperparameter ℓ configurable in every dimension makes this kernel more flexible.

1.2.2.6 Neural network kernel

Also a neural network can be used as a kernel for GP. How to derive the following neural network kernel is discussed in [43].

$$\kappa_{\text{NN}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2}{\pi} \arcsin \left(\frac{2\tilde{\boldsymbol{x}}^T \boldsymbol{\Sigma} \tilde{\boldsymbol{x}}'}{\sqrt{(1 + 2\tilde{\boldsymbol{x}}^T \boldsymbol{\Sigma} \tilde{\boldsymbol{x}}')(1 + 2\tilde{\boldsymbol{x}}'^T \boldsymbol{\Sigma} \tilde{\boldsymbol{x}}')}} \right), \quad (1.17)$$

where $\tilde{\boldsymbol{x}}$ is an augmented $\boldsymbol{x}$ with a bias component such that $\tilde{\boldsymbol{x}} = (1, x_1, \ldots, x_D)^T$ and $\boldsymbol{\Sigma}$ denotes a covariance function used for sampling network's weights from a multivariate normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$.

1.2.2.7 Mixing of kernels

A valid kernel has to be positive semi-definite. This property holds for addition and multiplication and hence these operations can be used to mix two kernels [43].

For instance addition of a SE kernel to a Q kernel results in a new kernel defined as follows:

$$\kappa_{\text{SE+Q}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\ell^2} \right) + (\boldsymbol{x}^T \boldsymbol{x}' + \sigma_0^2)^2. \quad (1.18)$$

1.2.3 Artificial neural network

Artificial neural network (ANN) is similar to a biological neural network. ANN models neurons and their activations using mathematical functions. In 1989, Hornik showed that ANNs are capable to approximate any continuous function [21].

Neural networks have been used in many cases for surrogate modeling [9, 22, 12].

Terminology and explanation of ANN basics are taken from [19].

1.2.3.1 Perceptron

Perceptron is the simplest neural network with just one artificial neuron. Perceptron, introduced by Frank Rosenblatt in 1958 [46], is defined by a sum of weighted inputs and its *activation function*.

$$y = f\left(\sum_{i=1}^{D} (w_i x_i) + b\right), \tag{1.19}$$

where $f(\xi)$ is a *step function* defined as

$$f(\xi) = \text{step}(\xi) = \begin{cases} 1 & \text{if } \xi \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{1.20}$$

$D \in \mathbb{N}$ is the number of dimensions of an input vector $\boldsymbol{x} \in \mathbb{R}^D$, $w_i \in \mathbb{R}$ is a weight of x_i and $b \in \mathbb{R}$ is a bias. Weights w_i and the bias b are fitted in the training phase.

The function $f(\xi)$ is called an activation function. The step activation function defined above is useful for binary classification since it has only two possible outputs. Many activation functions have been defined and some of them are described in the following section.

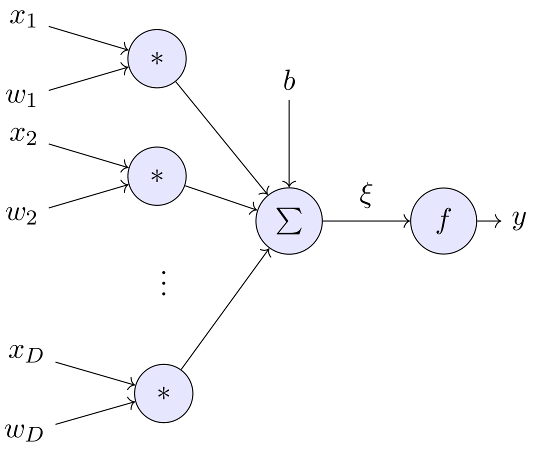

Figure 1.3: A computational graph of a perceptron.

1.2.3.2 Activation function

An activation function transforms the weighted sum of a neuron's inputs and often works as a nonlinear element of the network. The type of activation function differs based on ANN usage.

Step activation function has been already presented in Equation 1.20.

Another activation function is a *linear function*

$$\operatorname{lin}(x) = x. \tag{1.21}$$

A combination of a linear and a step function is called *rectified linear unit* (ReLU) and is widely used in deep learning:

$$\operatorname{relu}(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{1.22}$$

A *logistic function* (see graph in Figure 1.4) is a monotone nonlinear activation function defined as

$$\operatorname{logistic}(x) = \frac{e^x}{1 + e^x}. \tag{1.23}$$

A *hyperbolic tangent function* (see graph in Figure 1.4) is similar to the logistic function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1.24}$$

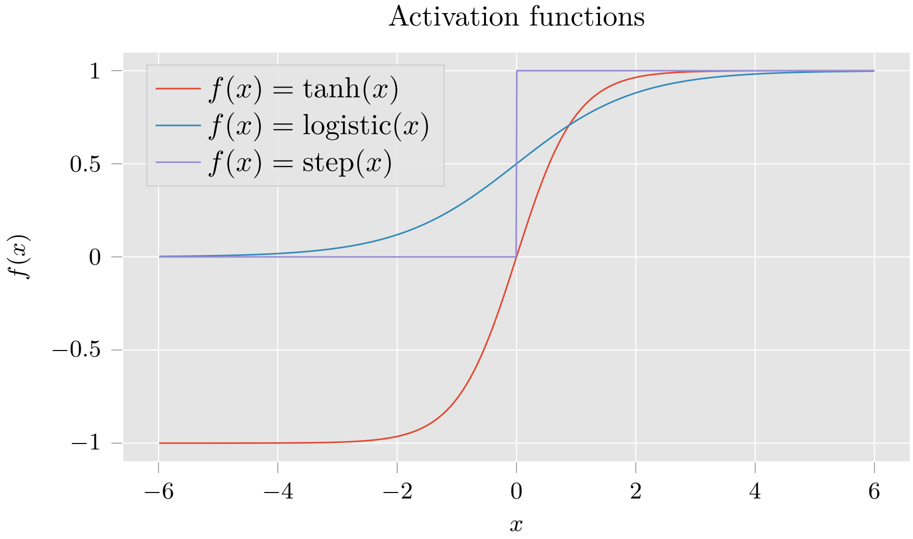

Figure 1.4: Graph of hyperbolic tangent, logistic and step activation functions.

1.2.3.3 Multilayer perceptron

One perceptron was found insufficient for solving problems such as separating data from the XOR function. This was discussed in 1969 by Minsky and Papert [33]. Later in 1985, a solution containing multiple neurons was found by Rumelhart et al. [47].

Connecting more neurons creates a graph structure. A *feedforward* neural network is a special case where the created graph does not contain any cycle. Feedforward neural network separable into layers, where neurons inside individual layers are not connected, is called a *multilayer perceptron*.

There are three types of layers – the input layer, the output layer, and the hidden layer.

The input layer is formed from non-computational nodes representing the input of the model. The hidden and output layers contain computational neurons. The output layer computes the output of the ANN model. The graphical representation of multilayer perceptron is depicted in Figure 1.5.

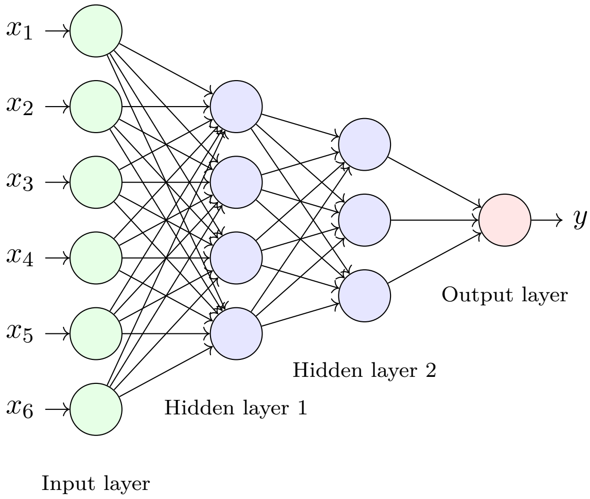

Figure 1.5: A multilayer perceptron with input layer composed of six neurons, two hidden layers with four and three neurons and output layer with one neuron.

1.2.4 Support vector machine

Support vector machine (SVM) was described by Vapnik in 1995 [50] as a generalization of finding an optimal separating hyperplane. The optimal separating hyperplane is a method that can separate linearly separable data. SVM uses a method called the *kernel trick* for problems that are not linearly separable.

SVM was used as a surrogate model in many papers, for instance in [26, 27, 48]. For example in [48], authors used support vector regression as a surrogate model of a fitness function. In [27] authors used support vector ranking for predicting the ranking of individuals in an evolution strategy.

1.2.4.1 Separating hyperplane

The key element of SVM is a separating hyperplane (Figure 1.6). Suppose a binary classification problem and the data

$$\{(x_i, y_i) \mid x_i \in \mathbb{R}^D, y \in \{1, -1\}, i = 1, \dots, n\} \tag{1.25}$$

which can be separated by a hyperplane

$$\boldsymbol{w}^T \boldsymbol{x} - b = 0. \tag{1.26}$$

The optimal hyperplane separates data without misclassification and the distance from the hyperplane and the nearest data point is maximal.

It can be shown that by minimizing $\|\boldsymbol{w}\|$ and satisfying inequalities

$$y_i((\boldsymbol{w}^T \boldsymbol{x}_i) - b) \geq 1, i = 1, \dots, n \tag{1.27}$$

yields the optimal $\boldsymbol{w}$ for the optimal separating hyperplane.

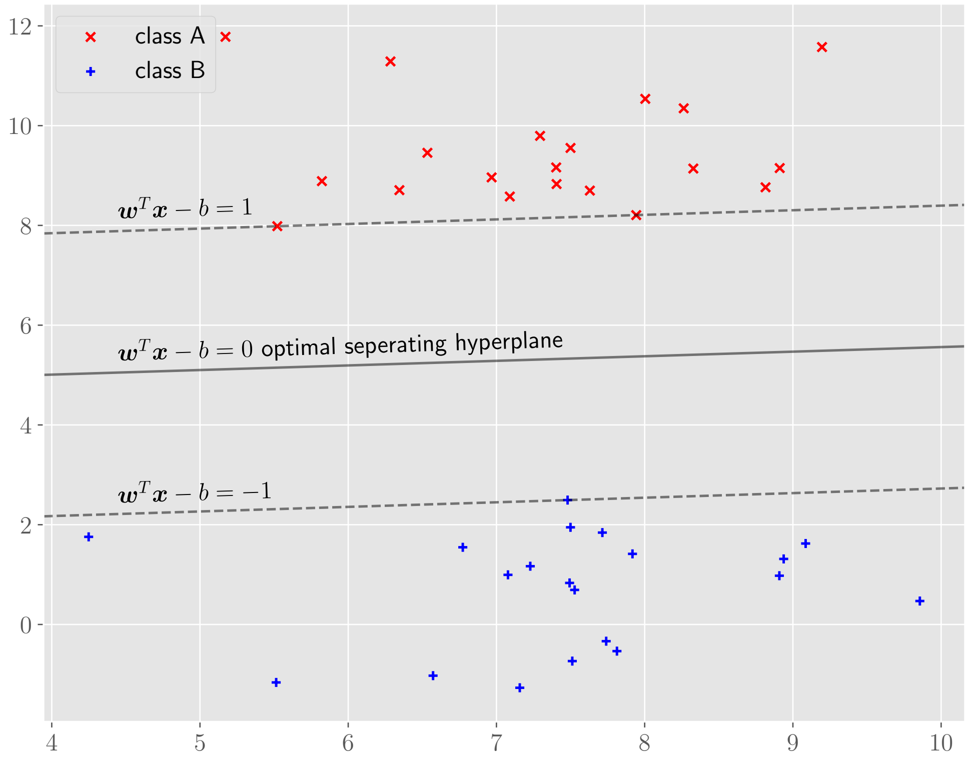

Figure 1.6: Optimal separating hyperplane for two classes.

The optimal hyperplane can be found using the Lagrange multipliers. The Lagrange function for this particular case can be written as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 - \sum_{i=1}^{n}(\alpha_i((\boldsymbol{x}_i^T\boldsymbol{w} - b)y_i - 1)). \tag{1.28}$$

Differentiation of the Lagrange function with respect to vector $\boldsymbol{w}$ and scalar b produces two following equations:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} =& \boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0,\\ \frac{\partial\mathcal{L}}{\partial b} =& -\sum_{i=1}^{n}\alpha_i y_i = 0.\end{aligned} \tag{1.29}$$

The Lagrange function with the combination of its derivatives yields expression

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^T\boldsymbol{x}_j). \tag{1.30}$$

The global maximum of the function W with subject to the constraints

$$\sum_{i=1}^{n}\alpha_i y_i = 0, \alpha_i \geq 0, i = 1, \ldots, n \tag{1.31}$$

and signum function produces the decision function

$$f(\boldsymbol{u}) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i\alpha_i\boldsymbol{u}^T\boldsymbol{x}_i - b\right) \tag{1.32}$$

which outputs the class of a new data point $\boldsymbol{u}$.

1.2.4.2 Kernel trick

Frequently, the problem is not linearly separable. To solve this obstacle kernels are used to transform data into a different space. To exploit this transformation the decision function changes to

$$f(\boldsymbol{u}) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i\alpha_i\kappa(\boldsymbol{u}, \boldsymbol{x}_i) - b\right) \tag{1.33}$$

and similarly the function W is

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}^{n}\alpha_i\alpha_j y_i y_j(\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{1.34}$$

Figure 1.7 shows a problem which is not linearly separable in the input space and is linearly separable in the space defined by RBF kernel function.

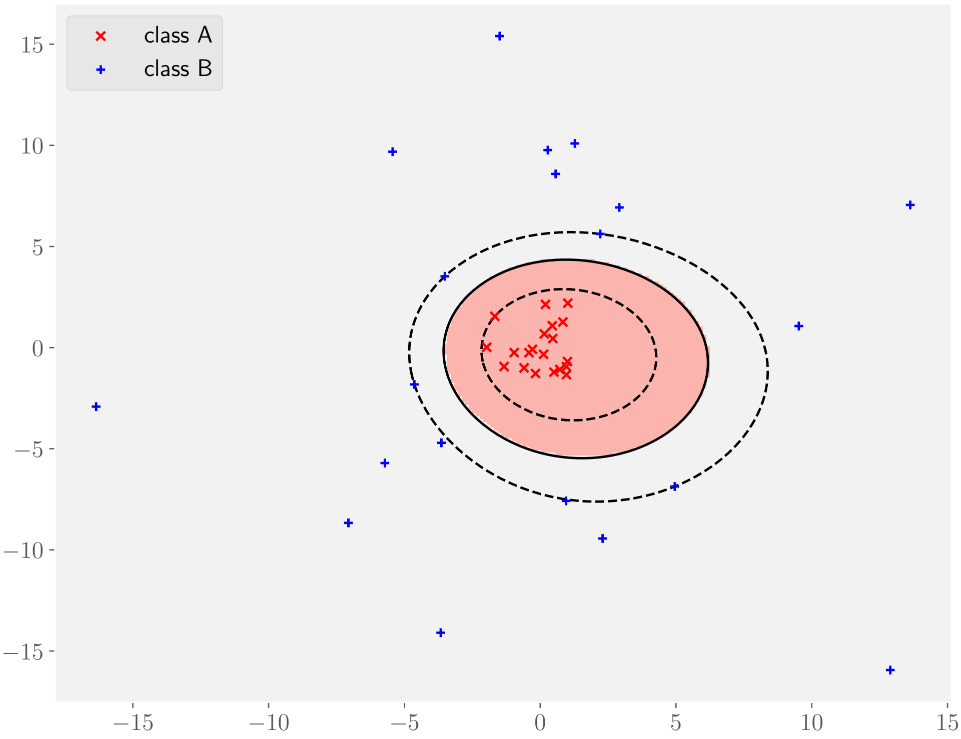

Figure 1.7: RBF kernel on a problem which is not linearly separable in input space.

1.2.4.3 Regression

Surrogate modeling is a regression task but SVM is defined for classification problems. This section explains how SVM is transformed to solve regression problems.

Support vector regression (SVR) estimates the true function with a linear function $f(\boldsymbol{x}, \boldsymbol{\alpha}) = (\boldsymbol{w}^T \boldsymbol{x}) - b$ and solves the problem by minimizing $\|\boldsymbol{w}\|$ with respect to the ε-insensitive loss function. The form of the ε-insensitive loss function is defined as

$$|y - f(\boldsymbol{x}, \boldsymbol{\alpha})|_\varepsilon = \begin{cases} 0 & \text{if } |y - f(\boldsymbol{x}, \boldsymbol{\alpha})| \leq \varepsilon, \\ |y - f(\boldsymbol{x}, \boldsymbol{\alpha})| - \varepsilon & \text{otherwise.} \end{cases} \tag{1.35}$$

It can be shown [11] that the solution of this problem is to maximize the quadratic form

$$W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\varepsilon \sum_{i=1}^{n} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{n} y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \boldsymbol{x}_i^T \boldsymbol{x}_j$$

subject to the constraints

$$\begin{gathered}\sum_{i=1}^{n} \alpha_i^* = \sum_{i=1}^{n} \alpha_i, \\ 0 \leq \alpha_i^* \leq C, i = 1, \ldots, n, \\ 0 \leq \alpha_i \leq C, i = 1, \ldots, n,\end{gathered}$$

where $C \in \mathbb{R}^+$ is a hyperparameter for regularization and (α_i, α_i^*) are Lagrange multiplier pairs that need to be found.

A prediction for a new data point $\boldsymbol{u}$ is then calculated with an equation:

$$f(\boldsymbol{u}) = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i)(\boldsymbol{u}^T \boldsymbol{x}_i) - b. \tag{1.36}$$

1.2.5 Random forest

Random forest is an ensemble of decision trees. This method can be used either for classification or regression tasks. The most popular regression random forest training method is *bagging* [7].

First regression and classification trees were introduced by Leo Breiman et al. in 1984 [8]. Since then they have been frequently used for their simplicity and interpretability. The following explanation of a regression tree is taken from [18].

1.2.5.1 Regression tree

A regression tree predicts the response value y_i based on input values $\boldsymbol{X}$. The regression tree differs from others presented methods in its structure. Instead of creating a mathematical function, it builds a set of decision rules which when applied to input features yield the response.

The rules are constructed recursively such that the rule splits the data into smaller groups with splitting criteria.

The algorithm for building the regression tree searches for the best variable of the input space in which the data can be split into two most convenient half-planes R_1 and R_2. In a regression task the splitting criteria is usually a sum of squares (see Equation 1.37).

Considering a regression problem the algorithm minimizes the following expression by searching a variable j and a split point s

$$\min_{j,s} \left(\min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2)^2 \right), \tag{1.37}$$

where c_1, c_2 are average values in corresponding half-planes and R_1, R_2 are half-planes described by the variable j and split point s such that $R_1(j,s) = \{\boldsymbol{X} \mid \boldsymbol{X}_j \leq s\}$ and $R_2(j,s) = \{\boldsymbol{X} \mid \boldsymbol{X}_j > s\}$.

This method divides recursively input space into two regions where each region has a constant response (Figure 1.8).

When building a tree a user needs to consider the depth of the tree because a large tree can overfit the training data.

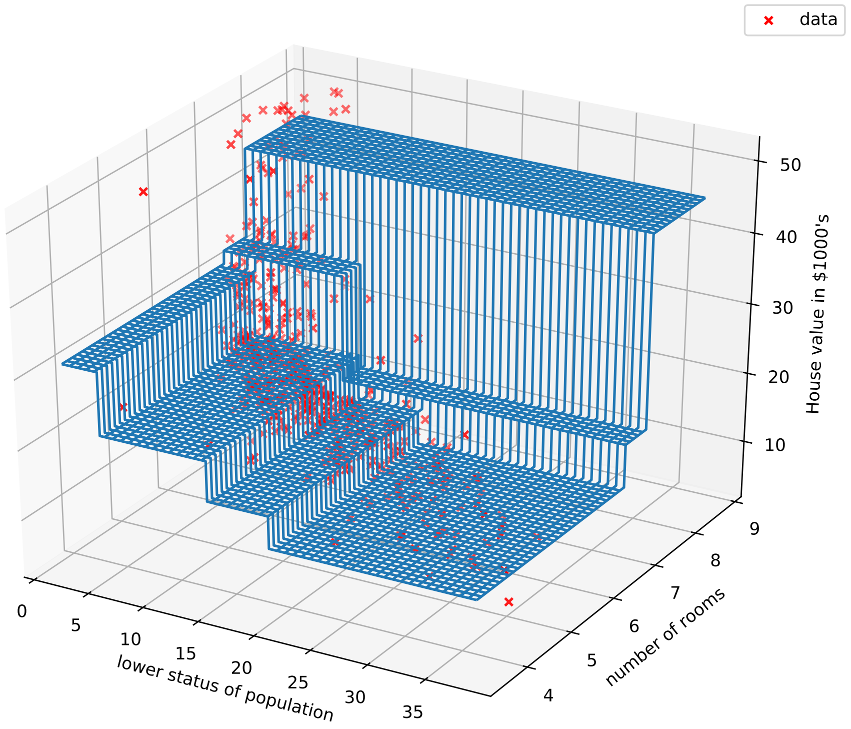

Figure 1.8: Regions of a regression tree fitted on a subset of the Boston dataset.

1.2.5.2 Bagging

Bagging is an abbreviation of bootstrap aggregation and it is a method for combining more predictive models. For the regression problem this method trains n models $\tilde{f}_i(\boldsymbol{x})$ with a randomly selected subset of data from the training set $\begin{bmatrix}\boldsymbol{Y} & \boldsymbol{X}\end{bmatrix}$. After the training phase, the bagging model yields the average output of $\tilde{f}_i(\boldsymbol{x})$, which can be described as

$$\tilde{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\tilde{f}_i(\boldsymbol{x}). \tag{1.38}$$

The idea behind this method is to reduce the overall variance by combining weak learners (experts on a subset of the training dataset).

1.3 Evolution strategy

Evolution strategy (ES) is an optimization technique inspired by the evolution theory. ES has been researched since the 1970s [44] starting with Ingo Rechenberg.

ES operates on *population* $\mathcal{P}$, a set containing search points $\boldsymbol{x} \in \mathbb{R}^D$ a.k.a. *individuals*. Individuals are compared by *fitness*, a function value of an individual. ES abstracts evolution into three steps: *recombination*, *mutation*, and *selection*. These steps are repeated in order to explore the input space and find global optima. Every repetition is considered as a *generation*.

The recombination step is simulating the process of reproduction of organisms. This process creates from *parents* (a subset of the population $\mathcal{P}$) new individuals a.k.a. *offspring*.

The mutation step simulates a genome mutation, this step usually slightly modifies the values of an individual.

The selection step simulates the survival of the fittest and selects a population for the new generation.

Evolution strategies are divided into categories [4] based on parameters of the algorithm.

- $(\mu + \lambda)$-ES is an evolution strategy that creates $\lambda \geq 1$ new offspring in a generation, mutates the offspring, adds the offspring into the population, and then discards λ individuals to keep the population size constant at μ individuals.

- (μ, λ)-ES creates λ new offspring from μ parents, mutates the offspring, and only from the offspring selects μ best individuals for the next generation.

- $(\mu/\rho \overset{+}{,} \lambda)$-ES introduces a new parameter ρ to adjust how many parents are involved in the recombination step. The selection of the new generation is determined based on the variant "+" or ",".

In evolution strategies a process called *self-adaptation* changes the mutation operator through the generation [4]. One of the algorithm that utilize self-adaptation process is *Covariance Matrix Adaptation Evolution Strategy* [15]. This algorithm and his surrogate versions are frequently used in black-box optimization.

1.3.1 CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES, see pseudocode in Algorithm 1) [15] falls into the $(\mu/\mu, \lambda)$-ES category. The algorithm can be simplified into a repetition of the following three steps:

(1) sample a new population of size λ by sampling from multivariate normal distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$,

(2) select the μ best offspring from the sampled population based on their fitness,

(3) update parameters of the multivariate distribution $\boldsymbol{m}$ and $\boldsymbol{\Sigma}$ with respect to the selected μ offspring.

1.3.1.1 Sampling

A multivariate normal distribution is sampled λ times to get a new population of points $\boldsymbol{x}_k^{(g)}$ for $k \in 1, \ldots, \lambda$. The parameter g denotes a generation number. The sampling is done with an equation:

$$\begin{aligned} \boldsymbol{x}_k^{(g+1)} &\sim \mathcal{N}\left(\boldsymbol{m}^{(g)}, \left(\sigma^{(g)}\right)^2 \boldsymbol{C}^{(g)}\right) \\ &\sim \boldsymbol{m}^{(g)} + \sigma^{(g)} \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{C}^{(g)}\right), \end{aligned} \tag{1.39}$$

where

$\boldsymbol{x}_k^{(g+1)}$ $\in \mathbb{R}^D$ is a k-th offspring from generation $g+1$,

$\boldsymbol{m}^{(g)}$ $\in \mathbb{R}^D$ is a mean vector of the search distribution in generation g,

$\sigma^{(g)}$ $\in \mathbb{R}^+$ is a step-size in generation g,

$\boldsymbol{C}^{(g)}$ $\in \mathbb{R}^{D \times D}$ is a covariance matrix in generation g.

The description of the $\boldsymbol{m}^{(g)}$, $\boldsymbol{C}^{(g)}$ and $\sigma^{(g)}$ parameter update derived from [15] is described in the following sections.

1.3.1.2 Update of the mean

The mean vector $\boldsymbol{m}$ is computed from weighted average of the μ best selected offspring.

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(g+1)}, \tag{1.40}$$

where

$\boldsymbol{x}_{i:\lambda}^{(g+1)}$ $\in \mathbb{R}^D$ is i-th best individual from generation $g+1$,

w_i $\in \mathbb{R}^+$ is weight coefficients for recombination.

Weights are restricted with constraints

$$\sum_{i=1}^{\mu} w_i = 1 \text{ and } w_1 \geq w_2 \geq \ldots \geq w_\mu > 0.$$

A discussion in [15] suggests to set weights $w_i \propto \mu - i + 1$.

1.3.1.3 Update of the covariance matrix

To update the covariance matrix $\boldsymbol{C}$, the CMA-ES combines two concepts *rank-μ-update* and *rank-one-update*.

Rank-μ-update
This update estimates covariance matrix from the best μ individuals. Their weighted sum and covariance matrix from previous generation are utilized to create a new covariance matrix. The *exponential smoothing* c_μ is used to adjust the influence of particular additions.

$$\boldsymbol{C}^{(g+1)} = (1 - c_\mu)\boldsymbol{C}^{(g)} + c_\mu \sum_{i=1}^{\mu} w_i \left(\frac{\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)\left(\frac{\boldsymbol{x}_{i:\lambda}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)^T, \tag{1.41}$$

where

c_μ $\in \mathbb{R}^+$ is exponential smoothing learning rate,

w_i $\in \mathbb{R}^+$ is weight coefficients for recombination.

Rank-one-update
Rank-one-update estimates the covariance matrix using the so-called *evolution path* $\boldsymbol{p}_c^{(g)}$. The reasoning behind this update is that the rank-μ-update does not have an information about the sign of individuals because $\boldsymbol{x}\boldsymbol{x}^T = (-\boldsymbol{x})(-\boldsymbol{x})^T$. The evolution path $\boldsymbol{p}_c^{(g)}$ is computed using the difference of two respective mean vectors $\boldsymbol{m}$ and thus have the information about the sign.

The evolution path $\boldsymbol{p}_c^{(g)}$ has an exponential smoothing adjustable using the parameter c_c

$$\boldsymbol{p}_c^{(g+1)} = (1 - c_c)\boldsymbol{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}, \tag{1.42}$$

where

$\boldsymbol{p}_c^{(g)}$ $\in \mathbb{R}^D$ is evolution path in generation g,

c_c ≤ 1 is a parameter for exponential smoothing,

μ_{eff} $\in \mathbb{R}$ is a normalization constant (see [15]).

Finally, using the evolution path the covariance matrix estimation is calculated with an equation

$$\boldsymbol{C}^{(g+1)} = (1 - c_{\text{cov}})\boldsymbol{C}^{(g)} + c_{\text{cov}}\boldsymbol{p}_c^{(g+1)}\boldsymbol{p}_c^{(g+1)^T}, \tag{1.43}$$

where $c_{\text{cov}} \in \mathbb{R}^+$ is a learning rate.

Combining updates

The covariance matrix $\boldsymbol{C}$ is computed using a combination of previously defined components.

$$\begin{aligned}\boldsymbol{C}^{(g+1)} =&(1-c_{\text{cov}})\boldsymbol{C}^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \underbrace{\boldsymbol{p}_c^{(g+1)}\boldsymbol{p}_c^{(g+1)^T}}_{\text{rank-one-update}} + \\ &+c_{\text{cov}}\left(1-\frac{1}{\mu_{\text{cov}}}\right)\underbrace{\sum_{i=1}^{\mu} w_i \left(\frac{\boldsymbol{x}_{i:\lambda}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)\left(\frac{\boldsymbol{x}_{i:\lambda}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)^T}_{\text{rank-}\mu\text{-update}},\end{aligned} \tag{1.44}$$

where μ_{cov} is a learning rate.

1.3.1.4 Update of the step-size

Similarly to the rank-one-update, the evolution path is utilized to update the step-size parameter $\sigma^{(g)}$.

The update of the evolution path $\boldsymbol{p}^{(g)}$ is defined with the following equation.

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\frac{\boldsymbol{m}^{(g+1)}-\boldsymbol{m}^{(g)}}{\sigma^{(g)}}, \tag{1.45}$$

where

$\boldsymbol{p}_\sigma^{(g)}$ $\in \mathbb{R}^D$ is the evolution path in generation g,

c_σ ≤ 1 is a parameter for exponential smoothing,

μ_{eff} $\in \mathbb{R}$ is a normalization constant (see [15]).

Finally, using the evolution path the step-size is calculated with the equation

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|}{\mathbb{E}\left\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\right\|}-1\right)\right), \tag{1.46}$$

where $d_\sigma \approx 1$, is a damping parameter.

1.3.2 S-CMA-ES

Surrogate Covariance Matrix Adaptation Evolution Strategy (S-CMA-ES) is a surrogate modification of the CMA-ES algorithm. S-CMA-ES utilizes surrogate modeling to decrease the number of evaluations of the objective function f.

S-CMA-ES algorithm trains a regression models from the data points $\mathcal{A} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i)) \mid i = 1,\ldots,n\}$, where n is a number of f-evaluated points. Trained models are sometimes used instead of the objective function f.

Algorithm 1: CMA-ES (simplified pseudocode) [3]

```
Input: original fitness function $f$, step-size $\sigma^{(0)} \in \mathbb{R}^+$, initial mean $\boldsymbol{m} \in \mathbb{R}^D$
Output: $\hat{\boldsymbol{x}}^{opt}$ point with the minimum achieved fitness
set the population size $\lambda, \mu, w_1, \dots, w_\mu$ and other parameters
  $(c_\sigma, d_\sigma, c_c, \mu_{\text{cov}}, c_{\text{cov}})$ to default values (for $\lambda$ and $\mu$ defaults are
  $\lambda = 4 + \lfloor 3\log(n) \rfloor, \mu = \lfloor \lambda/2 \rfloor$)
initialize the evolution path $\boldsymbol{p}_\sigma^{(0)} = 0, \boldsymbol{p}_c^{(0)} = 0,$ covariance matrix $\boldsymbol{C}^{(0)} = \boldsymbol{I}$
foreach generation $g = 0, 1, 2, \dots$ until stopping conditions met do
    $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{m}^{(g)}, (\sigma^{(g)})^2 \boldsymbol{C}^{(g)})$ for $k = 1, \dots, \lambda$        // sample population
    sorted $\boldsymbol{x}_{1:\lambda} \leftarrow f$-evaluate all $\boldsymbol{x}_1, \dots, \boldsymbol{x}_\lambda$        // fitness evaluation
    $\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$                    // selection and recombination
    $\boldsymbol{p}_\sigma^{(g+1)} \leftarrow$ aggregate the $(\sigma^{(g)})^2\boldsymbol{C}$-normalized difference of means
            $(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)})/\sigma^{(g)}\sqrt{\boldsymbol{C}^{(g)}}$ into evolution path $\boldsymbol{p}_\sigma^{(g)}$
    $\sigma^{(g+1)} \leftarrow$ update the step-size according to the length $\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|$
    $\boldsymbol{p}_c^{(g+1)} \leftarrow$ aggregate the $\sigma^{(g)}$-normalized difference of means
            $(\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g))}/\sigma^{(g)}$ into the evolution path $\boldsymbol{p}^{(g)}$
    $\boldsymbol{C}^{(g+1)} \leftarrow$ perform the rank-one update and rank-$\mu$-update
```

Algorithm 2 shows the pseudocode of the S-CMA-ES with the Gaussian process surrogate model. The main difference between CMA-ES (Algorithm 1) and S-CMA-ES (Algorithm 2) is the use of a surrogate model $f_{\mathcal{M}}$ trained by the trainModel function (Algorithm 3) on a subset of already evaluated points $\mathcal{T} \subset \mathcal{A}$. The process of switching between model-evaluated points and f-evaluated points is controlled by generation's attributes *original-fitness-evaluated*, *model-evaluated* and variable g_m. The variable g_m counts the amount of consecutive generations the algorithm should use the trained model instead of the function f. The generation's attributes are used for switching to the branch where the algorithm evaluates one generation with function f and trains a new model for the next g_m generations.

1.3.3 DTS-CMA-ES

An improved version of S-CMA-ES called Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy (DTS-CMA-ES) has been proposed in [39]. Authors have found that this approach usually reduces the number of necessary evaluations in expensive optimization more than the rest of compared methods.

DTS-CMA-ES utilizes an estimation of the uncertainty of the surrogate model prediction. Regression models such as Gaussian process or random forest are used for their capability of predicting a whole distribution instead of just function value prediction.

This algorithm differs from S-CMA-ES in evaluating points with the model

```
Algorithm 2: S-CMA-ES simplified version of [3]
Input: original fitness function $f$, step-size $\sigma^{(0)} \in \mathbb{R}^{+}$, initial mean
       $\boldsymbol{m} \in \mathbb{R}^{D}$, the number of consecutive model-evaluated generations
       $g_m$, maximum training set size $N_{max}$, kernel $\kappa$
Output: $\hat{\boldsymbol{x}}^{opt}$ point with the minimum achieved fitness from $\mathcal{A}$
$\mathcal{A} \leftarrow \emptyset; \lambda, \sigma^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{C} \leftarrow$ CMA-ES initialize
mark $g = 0$ as original-fitness-evaluated
foreach generation $g = 0, 1, 2, \ldots$ until stopping conditions met do
    $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{m}^{(g)}, (\sigma^{(g)})^2 \boldsymbol{C}^{(g)})$ for $k = 1, \ldots, \lambda$        // CMA-ES sampling
    if $g$ is original-fitness-evaluated then
        $y_k \leftarrow f(\boldsymbol{x}_k), k = 1, \ldots, \lambda$
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{\lambda}$
        $f_{\mathcal{M}} \leftarrow$ trainModel($\mathcal{A}, N_{max}, \kappa, \sigma^{(g)}, \boldsymbol{C}^{(g)}$)
        mark $(g + 1)$ as model-evaluated
    else
        $\hat{y}_k \leftarrow f_{\mathcal{M}}(\boldsymbol{x}_k), k = 1, \ldots, \lambda$
        if $g_m$ model generations have passed then
            mark $(g + 1)$ as original-fitness-evaluated
    $\sigma^{(g+1)}, \boldsymbol{m}^{(g+1)}, \boldsymbol{C}^{(g+1)} \leftarrow$ CMA-ES update based on $\boldsymbol{x}_{1:\lambda}$,
                                        sorted accordingly to $y_{1:\lambda}$
```

```
Algorithm 3: trainModel simplified version of [3]
Input: archive of original-evaluated points $\mathcal{A}$, maximum training set size
       $N_{max}$, kernel $\kappa$, CMA-ES state variables $\sigma^{(g)}, \boldsymbol{C}^{(g)}, \boldsymbol{m}^{(g)}$
Output: $f_{\mathcal{M}}$ trained GP model with hyperparameters $(m_{\mu}, \sigma_f^2, \ell, \sigma_n)$
$(\boldsymbol{X}_N, \boldsymbol{y}_N) \leftarrow$ selected at most $N_{max}$ points from archive $\mathcal{A}$ using selection
                     method (see [3])
$\boldsymbol{X}_N \leftarrow$ transform the selected points into the $(\sigma^{(g)})^2 \boldsymbol{C}^{(g)}$ basis
$\boldsymbol{y}_N \leftarrow$ standardize the $f$-values in $\boldsymbol{y}_N$ to zero mean and unit variance
$(m_{\mu}, \sigma_f^2, \ell, \sigma_n) \leftarrow$ fit the hyperparameters of $\mu(\boldsymbol{x})$ and $\kappa$ using ML estimation
```

M and the function f in every generation instead of switching between them through the generations. To save expensive evaluations DTS-CMA-ES uses parameter $\alpha^{(g)}$ for controlling how many points in a generation will be evaluated by the function f. The points for evaluation are selected based on a criterion C where the uncertainty of the prediction is utilized.

The DTS-CMA-ES algorithm is shown in Algorithm 4.

1.4 Surrogate model selection

In the step 4 of DTS-CMA-ES (Algorithm 4) a surrogate model is trained. The question is how to select the most convenient model to improve the optimization process.

Algorithm 4: DTS-CMA-ES simplified version of [3]

```
Input: original fitness function $f$, step-size $\sigma^{(0)} \in \mathbb{R}^+$, initial mean
       $\boldsymbol{m} \in \mathbb{R}^D$, initial ratio of original-evaluated points $\alpha^{(0)}$, criterion for
       the selection of original-evaluated points $\boldsymbol{C}$, self-adaptation
       parameters $\beta, \epsilon^{(0)}, \epsilon_{min}, \epsilon_{max}, \alpha_{min}, \alpha_{max}$, maximum training set
       size $N_{max}$, kernel $\kappa$
Output: $\hat{\boldsymbol{x}}^{opt}$ point with the minimum achieved fitness
$\mathcal{A} \leftarrow \emptyset; \lambda, \sigma^{(0)}, \boldsymbol{m}^{(0)}, \boldsymbol{C} \leftarrow$ CMA-ES initialize    // initialization
foreach generation $g = 0, 1, 2, \ldots$ until stopping conditions met do
    $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{m}^{(g)}, (\sigma^{(g)})^2 \boldsymbol{C}^{(g)})$ for $k = 1, \ldots, \lambda$    // CMA-ES sampling
    $f_{\mathcal{M}1} \leftarrow$ trainModel$(\mathcal{A}, N_{max}, \kappa, \sigma^{(g)}, \boldsymbol{C}^{(g)})$    // $1^{st}$ model training
    $(\hat{\boldsymbol{y}}, \hat{s}^2) \leftarrow f_{\mathcal{M}1}([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda])$    // model-fitness evaluation
    $\boldsymbol{X}_{orig} \leftarrow$ select $\lceil \alpha^{(g)} \lambda \rceil$ best points according to the criterion $C$
    $\boldsymbol{y}_{orig} \leftarrow f(\boldsymbol{X}_{orig})$    // original-fitness evaluation
    $\mathcal{A} = \mathcal{A} \cup \{(\boldsymbol{X}_{orig}, \boldsymbol{y}_{orig})\}$
    $f_{\mathcal{M}2} \leftarrow$ trainModel$(\mathcal{A}, N_{max}, \kappa, \sigma^{(g)}, \boldsymbol{C}^{(g)})$    // model retrain
    $\boldsymbol{y} \leftarrow f_{\mathcal{M}2}([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda])$    // $2^{nd}$ model prediction
    $(\boldsymbol{y})_k \leftarrow (\boldsymbol{y}_{orig})_i$ for all original-evaluated $(\boldsymbol{y}_{orig})_i \in \boldsymbol{y}_{orig}$    // replace
    $(\alpha^{(g+1)}, \epsilon^{(g+1)}) \leftarrow$ selfAdaptation$(\epsilon^{(g)}, \hat{\boldsymbol{y}}, \boldsymbol{y}; \beta\epsilon_{min}, \epsilon_{max}, \alpha_{min}, \alpha_{max})$
    sorted $\boldsymbol{x}_{1:\lambda} \leftarrow$ sort $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on $(y_1, \ldots, y_\lambda)^T$    // population sort
    $\sigma^{(g+1)}, \boldsymbol{m}^{(g+1)}, \boldsymbol{C}^{(g+1)} \leftarrow$ CMA-ES update based on $\boldsymbol{x}_{1:\lambda}$
$\hat{\boldsymbol{x}}^{opt} \leftarrow x_k$ from $\mathcal{A}$ corresponding to the minimal $y_k$
```

Algorithm 5: selfAdaptation [3]

```
Input: smoothed error from the last generation $\epsilon^{(g)}$, vector of the first
       model prediction $\hat{\boldsymbol{y}}$, vector of the second model prediction with
       original evaluations $\boldsymbol{y}$, update rate $\beta$, minimum and maximum
       ranking error: $\epsilon_{min}$, $\epsilon_{max}$, min. and max. ratio of original-evaluated
       points: $\alpha_{min}$, $\alpha_{max}$
Output: $\alpha^{(g+1)}$-ratio of original-evaluated points for the next generation,
        $\epsilon^{(g+1)}$-new smoothed error
$\epsilon^{RDE} \leftarrow RDE_\mu(\hat{\boldsymbol{y}}, \boldsymbol{y})$    // estimation of the model's error
$\epsilon^{(g+1)} \leftarrow (1 - \beta)\epsilon^{(g)} + \beta\epsilon^{RDE}$    // exponential smoothing of the error
$\alpha^{(g+1)} \leftarrow \alpha_{min} + \max\{0, \min\{1, \frac{\epsilon^{(g+1)} - \epsilon_{min}}{\epsilon_{max} - \epsilon_{min}}\}\}(\alpha_{max} - \alpha_{min})$    // $\alpha$ update
```

One way to describe the surrogate model selection problem is to use a framework for algorithm selection proposed by Rice in [45]. This framework is designed with five main components:

Problem space is a space of possible problems. This can be a possibly infinite set of problems.

Algorithm space is a space of possible algorithms that can be used to solve a problem from problem space.

Feature space is a space of possible characterizations. The feature space has a finite dimension.

Performance space is a space describing the performance of a particular algorithm for a particular problem.

Selection mapping is a function that gives a model M for a particular features $f(p)$ of a problem p such that it minimizes models error ε.

The following diagram [35, 45] (Figure 1.9) illustrates the main parts of this framework and their relations.

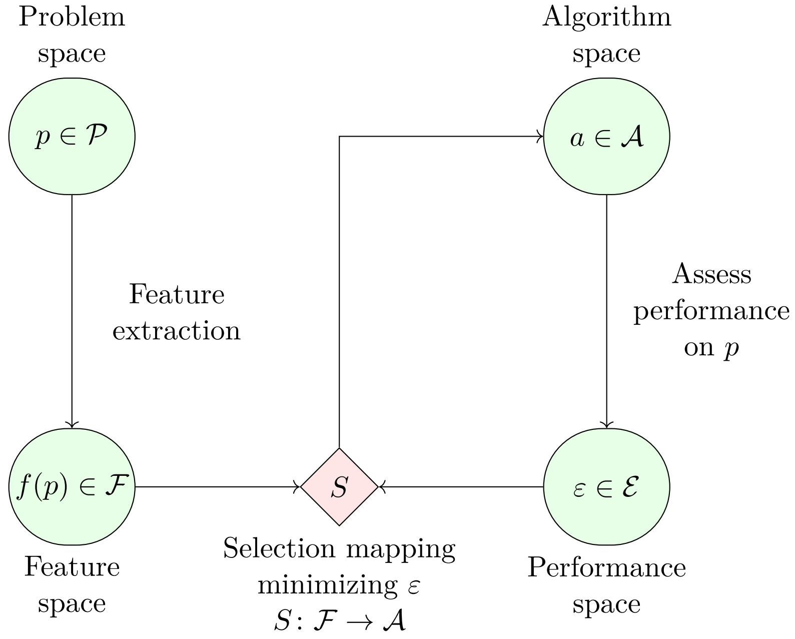

Figure 1.9: Rice's framework for Algorithm selection problem.

The problem of selecting surrogate model for DTS-CMA-ES is very similar. The main question is: How to automatically select the best surrogate model which can differ in every generation?

Model pooling which selects the best surrogate model based on the performance of different models from the limited history of previous generations, was addressing the same problem [20].

Cross validation could be used for every model to obtain the best model for known data. This could be done by cross validating multiple models on known data points, comparing their performance, and selecting the most promising model with the premise that the selected model will have good performance as well.

This thesis further develops research published in [40, 42] where authors used *fitness landscape analysis* to obtain a description of the fitness landscape (feature space) from which a selection mapping S could recommend the most suitable surrogate model.

1.4.1 Fitness landscape analysis

Fitness landscape analysis is trying to characterize the structure of a fitness function with measurable features. Features are calculated from already evaluated data points. As these features are describing the structure of a fitness function, they could give information from which the most suitable surrogate model could be obtained.

The important set of high-level fitness landscape properties was proposed in [32]. Authors call them high-level because an expert is needed to interpret them correctly for a given landscape.

Considered high-level properties from [32] are described in the following list and some of them are shown in Figure 1.10.

Multimodality is a property describing the number of local optima of a function. An unimodal function has only one optimum, unlike a multimodal function, that can have multiple optima points.

Global structure is a structure formed from all non-optima points. This refers to the overall structure of the optimized function.

Separability refers to a problem that can be separated into lower dimensional subproblems which might be easier to solve.

Variable scaling is a dimension scaling. This can mean that in one dimension a smaller step size might be needed than in other dimensions.

Search space homogeneity refers to a search space without phase transitions. In other words, the function behaves similarly in all areas.

Basin size homogeneity is a size relation between basins of attractions. This could be viewed as how hard is it to overcome local optima.

Global to local optima contrast measures a difference between local and global optima w.r.t. average fitness.

Plateaus refer to subspaces where the function is constant.

Later, in [31], the same authors discussed a new set of low-level features that are linked to the mentioned high-level properties and can be automatically measured with various techniques. The authors also proposed a relationship graph (Figure 1.11) between the high-level and low-level features.

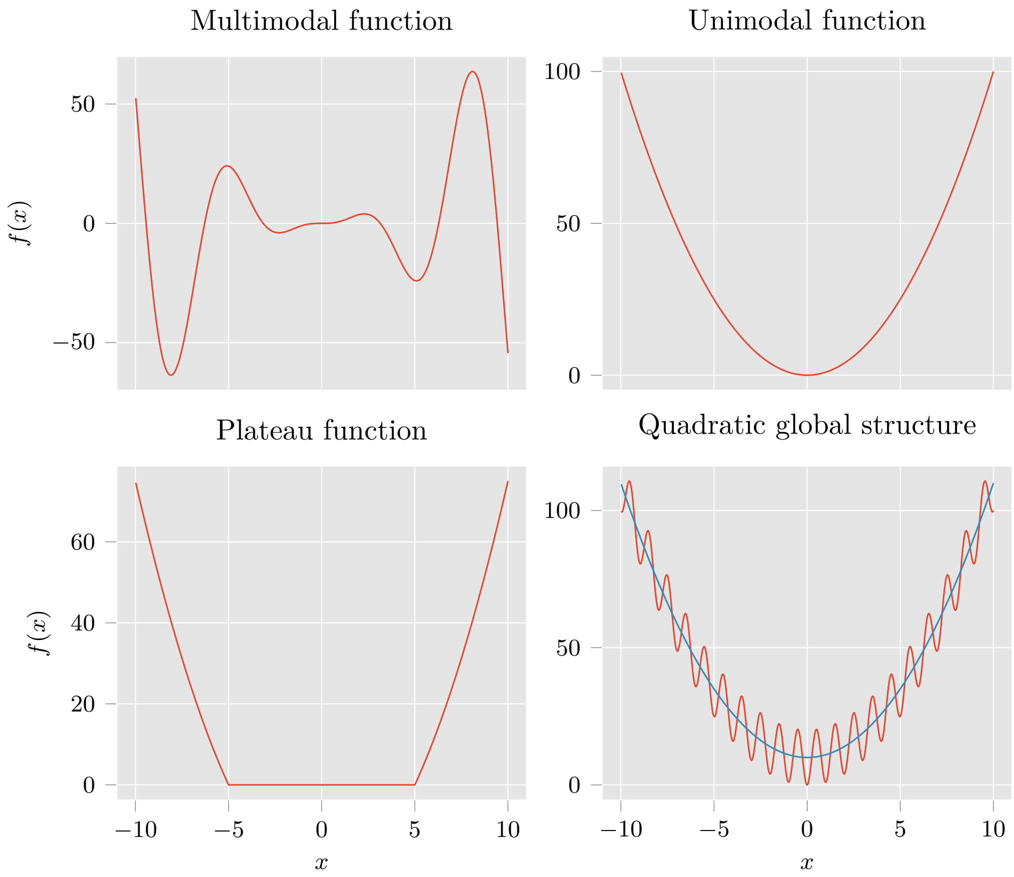

Figure 1.10: Visual description of high-level properties. The left top figure shows a function with multiple optima points, the top right figure shows a function with one optima point, the left bottom figure shows a function with a plateau area and the bottom right figure shows a rugged function with a quadratic global structure.

Additional low-level features can be useful for analyzing the structure of a fitness landscape such as *Nearest-Better Clustering*, *Information Content of Fitness Sequences*, or *Dispersion*. These low-level features were used in [42]. All of the mentioned low-level features are described in the following sections.

1.4.1.1 Convexity

Authors in [31] have introduced two features for quantifying the convexity of a fitness function.

Features are computed by sampling 1000 pairs of random points from the input space. For each pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ a linear combination with random weights

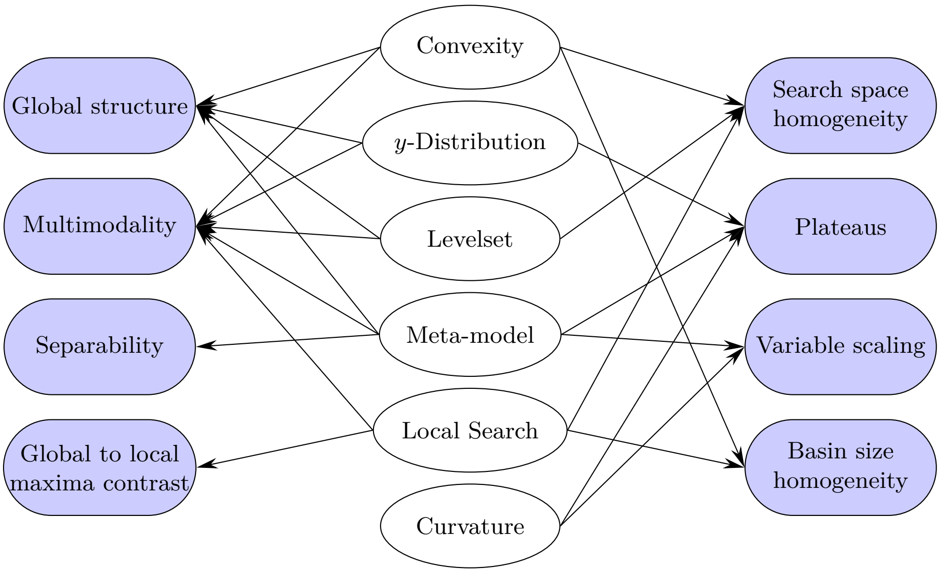

Figure 1.11: A relationship between high-level and low-level features [31].

is computed; then a difference d_i between the function value of the new point and a convex combination of the original function values $(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2))$ is computed. Finally, the number of times this difference d_i is less than predefined threshold -10^{-10} divided by the number of pairs gives the probability of convexity.

The second feature is the average of the differences.

1.4.1.2 y-Distribution

Features based on the distribution of the fitness function values. In [31] authors have presented three features attainable from y-Distribution: *skewness*, *kurtosis*, and *number of peaks*.

The skewness of a distribution tells us how asymmetric the distribution is. The skewness is computed from central moments with the following equation

$$s = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})}\right)^3}. \tag{1.47}$$

The kurtosis of a distribution measures how much the distribution differs from the normal distribution in the sense of tailedness.

$$\gamma = n\frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{\sum_{i=1}^{n}(x_i - \bar{x}^2)^2} - 3 \tag{1.48}$$

The last feature is an estimation of the number of peaks in the y-Distribution.

1.4.1.3 Levelset

Levelset features are calculated from a dataset split into two classes based on a threshold in function values. As a split value, the median value or quantile values have been studied in [31].

Linear, quadratic, and mixture discriminant analysis are used on the partitioned dataset to separate classes. The idea is that for the right choice of the threshold value a multimodal fitness landscape cannot be separated with linear or quadratic discriminant analysis. However, the mixture discriminant analysis should have better performance on a multimodal fitness landscape.

The features are defined as cross validated misclassification errors for each type of discriminant analysis.

1.4.1.4 Meta-model

Features from this class are acquired from fitting a linear and quadratic regression model.

The model performance, specifically the adjusted R^2 value of linear and quadratic models, has been used in [31] together with a minimum and a maximum value of the absolute values of the linear model coefficients. For the quadratic model author used maximum absolute value divided by the minimum absolute value of the fitted model's coefficients.

1.4.1.5 Local search

This class of features is built upon a local search algorithm Nelder-Mead.

Nelder-Mead is applied to randomly selected points in input space and for each point finds a local optimum. From the found local optima points a variety of features can be calculated. Such as a number of unique local optima points or features from hierarchically clustered optima points.

1.4.1.6 Curvature

Curvature is estimated from the numerically approximated gradients and from Hessian matrices calculated from randomly sampled points of the input space.

The approximated gradients and Hessian matrices are used for calculating the final features. In [31], authors formed various features based on statistics derived from the approximated partial derivatives.

1.4.1.7 Nearest-Better Clustering

The features based on Nearest-Better Clustering (NBC) have been proposed in [23]. The authors presented five features with an explanation of what they should represent. These features should help recognize funnel structures in the fitness landscape.

NBC features are computed from sets of nearest neighbor distances and nearest-better neighbor distances.

The distance to the nearest neighbor of a point $\boldsymbol{x}$ from a population $\mathcal{P}$ is defined as

$$d_{\text{nn}}(\boldsymbol{x}, \mathcal{P}) = \min\{\text{dist}(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{y} \in \mathcal{P} \setminus \{\boldsymbol{x}\}\} \tag{1.49}$$

and the distance to the nearest-better neighbor is defined similarly but only for the neighbors which have better fitness value

$$d_{\text{nb}}(\boldsymbol{x}, \mathcal{P}) = \min\{\text{dist}(\boldsymbol{x}, \boldsymbol{y}) \mid f(\boldsymbol{y}) < f(\boldsymbol{x}) \land \boldsymbol{y} \in \mathcal{P}\}. \tag{1.50}$$

The set of all nearest neighbor distances and all nearest-better distances within the population can be constructed from above equations as follows:

$$\begin{aligned} \mathcal{D}_{\text{nn}} =& \{d_{\text{nn}}(\boldsymbol{x}, \mathcal{P}) \mid \boldsymbol{x} \in \mathcal{P}\}, \\ \mathcal{D}_{\text{nb}} =& \{d_{\text{nb}}(\boldsymbol{x}, \mathcal{P}) \mid \boldsymbol{x} \in \mathcal{P}\}. \end{aligned} \tag{1.51}$$

From these sets, the authors constructed five features. The first one is a division of standard deviations. The idea is that the set $\mathcal{D}_{\text{nb}}$ should contain more extreme values if the problem is multimodal and therefore $\text{sd}(\mathcal{D}_{\text{nb}})$ should be a greater number.

$$\text{nbf}_1 = \frac{\text{sd}(\mathcal{D}_{\text{nn}})}{\text{sd}(\mathcal{D}_{\text{nb}})} \tag{1.52}$$

Another feature is a division of mean values of the two sets.

$$\text{nbf}_2 = \frac{\text{mean}(\mathcal{D}_{\text{nn}})}{\text{mean}(\mathcal{D}_{\text{nb}})} \tag{1.53}$$

The third feature measures the Pearson correlation between the two sets. Authors expected that for a funnel structure, the correlation should be high, whereas, in a random peaks setting, it should be much lower.

$$\text{nbf}_3 = \text{cor}(\mathcal{D}_{\text{nn}}, \mathcal{D}_{\text{nb}}) \tag{1.54}$$

The fourth feature is computed from the set Q formed from the division of distances for every individual in a population $\mathcal{P}$. This feature should be larger for random peak landscapes.

$$\begin{aligned} Q_{\text{nn/nb}} =& \left\{ \frac{d_{\text{nn}}(\boldsymbol{x}, \mathcal{P})}{d_{\text{nb}}(\boldsymbol{x}, \mathcal{P})} \mid \boldsymbol{x} \in \mathcal{P} \right\} \\ \text{nbf}_4 =& \frac{\text{sd}(\mathcal{Q}_{\text{nn/nb}})}{\text{mean}(\mathcal{Q}_{\text{nn/nb}})} \end{aligned} \tag{1.55}$$

The last feature is computed with indegree function $\text{deg}^-(\boldsymbol{x})$. This function yields the number of input vertices in the graph constructed from nearest-better neighbors. The best optima of a funnel landscape should have a larger indegree value than an optimum in a random peak landscape.

$$\text{nbf}_5 = -\text{cor}(\{(\text{deg}^-(\boldsymbol{x}), f(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathcal{P}\}) \tag{1.56}$$

1.4.1.8 Dispersion

Dispersion of a function measures how close together sampled points are [28]. The dispersion features are derived from this idea. They average differences between dispersion values below a certain moving threshold value.

A dispersion function is low when the sum differences of dispersions when moving the threshold is less than zero. A dispersion function is high when the sum differences is greater than zero.

In Figure 1.12, two functions are showing how different thresholds influence the function values from which the dispersion can be calculated.

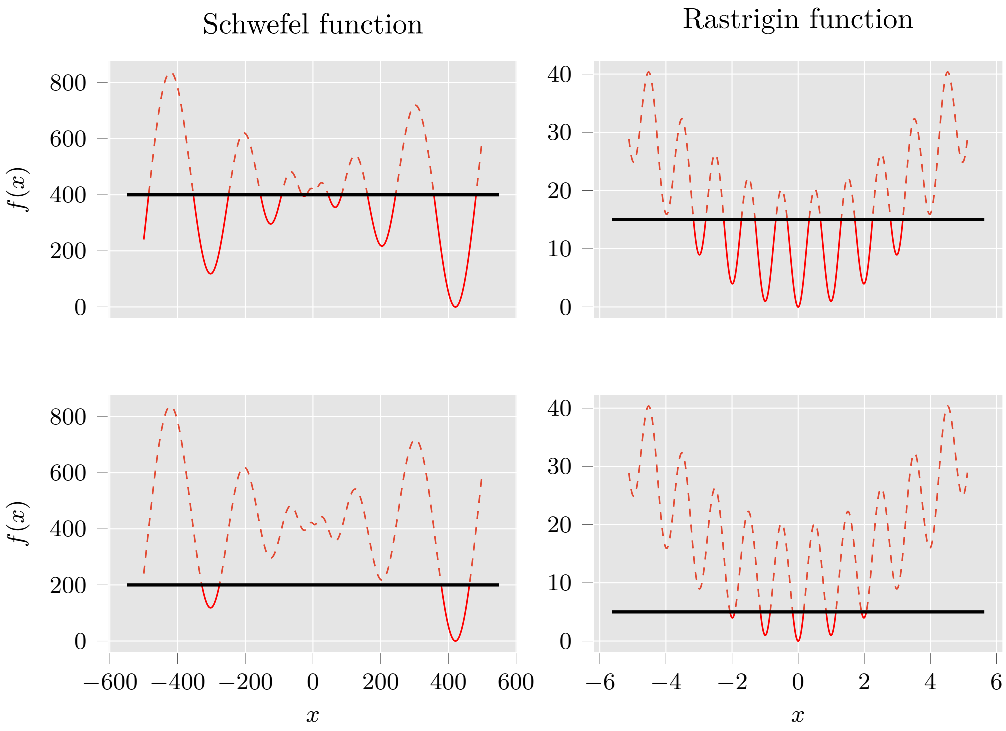

Figure 1.12: A decrease in fitness threshold will tend to decrease the dispersion on the Rastrigin function (figures on the right) and increase the dispersion of the Schwefel function (figures on the left) [28].

To estimate the dispersion authors in [28] sampled the space and took the best $n\%$ points from which they averaged distances between them. This step was repeated with different n values and the final dispersion was computed by subtracting those results. Specifically, they computed the value:

$$\text{dispersion} = \text{dispersion}(100, 409600) - \text{dispersion}(100, 100).$$

The dispersion function is presented with Pseudocode 6.

Algorithm 6: Dispersion [28]

```
Input: Integer $s_b$ - the fixed sample size, Integer $s_v$ - the variable sample
       size, $f(\boldsymbol{x})$ - the objective fitness function
Output: Float $d$ - an estimation of the dispersion
foreach $i = 1, \ldots, s_v$ do
    Create a random point $\boldsymbol{x}$
    Evaluate its fitness $f(\boldsymbol{x})$
    Add $\{\boldsymbol{x}, f(\boldsymbol{x})\}$ to array $allPoints$
$bestPoints \leftarrow$ best $s_b$ points of $allPoints$
$d \leftarrow$ average pairwise distance of $bestPoints$
```

1.4.1.9 Information Content of Fitness Sequences

Information Content of Fitness Sequences (ICoFS) introduced in [34], measures how difficult is it to describe a given fitness function. For instance, a low information function would be a constant fitness function as opposed to a high information function such as some multimodal complicated fitness function.

This method uses neighboring values and compares their fitness values. The comparisons are later transformed into discrete information from which the features are computed.

Authors in [34] used Latin Hypercube Design sampling with a sorting algorithm for obtaining the neighboring values. To create a sequence of discrete values from the measured data points, authors used equation

$$\Psi(i, \varepsilon) = \begin{cases} \bar{1} & \text{if } \frac{y_{i+1} - y_i}{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2} < -\varepsilon, \\ 0 & \text{if } \left| \frac{y_{i+1} - y_i}{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2} \right| \leq \varepsilon, \\ 1 & \text{if } \frac{y_{i+1} - y_i}{\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2} > \varepsilon. \end{cases} \tag{1.57}$$

where $\varepsilon \geq 0$ is the sensitivity parameter, $(\boldsymbol{x}_{i+1}, y_{i+1})$ and $(\boldsymbol{x}_i, y_i)$ are two neighboring data points.

From the above function a sequence $\boldsymbol{\varepsilon} = \{\bar{1}, 0, 1\}^{n-1}$ is formed. The *information content* is calculated from this sequence using

$$\boldsymbol{H}(\boldsymbol{\varepsilon}) = -\sum_{a \neq b} p_{\mathrm{ab}} \log_6 p_{\mathrm{ab}}. \tag{1.58}$$

Additionally, the *partial information content* $\boldsymbol{M}(\varepsilon)$ is calculated to approximate smoothness of the fitness function.

$$\boldsymbol{M}(\boldsymbol{\varepsilon}) = \frac{|\Phi(\boldsymbol{\varepsilon})|}{n-1}, \tag{1.59}$$

where $\Phi(\boldsymbol{\varepsilon})$ is a function which deletes 0 characters from the sequence $\boldsymbol{\varepsilon}$ and squeezes repeating characters 1 and $\bar{1}$. The length of this new sequence approximates the number of concavity changes.

From the information content and the partial information content four metrics are computed: *maximum information content*

$$\boldsymbol{H}_{\max} = \max_{\varepsilon}\{\boldsymbol{H}(\varepsilon)\}, \tag{1.60}$$

settling sensitivity

$$\varepsilon_s = \log_{10}\left(\min_{\varepsilon}\{\varepsilon \mid \boldsymbol{H}(\varepsilon) < 0.05\}\right), \tag{1.61}$$

initial partial information

$$M_0 = M(\varepsilon = 0), \tag{1.62}$$

and *half partial information sensitivity*

$$\varepsilon_{0.5} = \log_{10}\left(\max_{\varepsilon}\{\varepsilon \mid M(\varepsilon) > 0.5M_0\}\right). \tag{1.63}$$

1.4.1.10 CMA-ES features

Authors in [42] proposed features related to the DTS-CMA-ES algorithm. They are computed from the CMA-ES settings, from points $\boldsymbol{X} = \{\boldsymbol{x}_i \mid i = 1, \dots, n\}$ for which the function value is known and from a dimension D of optimized black-box function f.

Generation number ϕ_g is current number of generation g.

Step-size ϕ_σ is the $\sigma^{(g)}$ from the CMA-ES algorithm.

Number of restarts $\phi_{\textbf{restart}}$ of the CMA-ES measures how many times the DTS-CMA-ES was restarted, this can be signal how difficult the problem is.

Mahalanobis mean distance $\phi_{d(\boldsymbol{m})}$ of the CMA-ES mean $\boldsymbol{m}^{(g)}$ to the mean of the empirical distribution of all points $\boldsymbol{X}$.

C evolution path $\phi_{\boldsymbol{p}_c}$ is square of the evolution path length $\left\|\boldsymbol{p}_c^{(g)}\right\|^2$,

σ evolution path ratio is a ratio between σ evolution path length and the expected length of a random evolution path

$$\phi_{\boldsymbol{p}_\sigma} = \frac{\left\|\boldsymbol{p}_\sigma^{(g)}\right\|^2}{\mathbb{E}\left\|\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\right\|}.$$

CMA similarity likelihood is the log-likelihood of all points $\boldsymbol{X}$ with respect to the CMA-ES distribution

$$\phi_{\mathcal{L}} = -\frac{n}{2}\left(D \log 2\pi\sigma^{(g)^2} + \log\det \boldsymbol{C}^{(g)}\right) - \frac{1}{2}\sum_{\boldsymbol{x}\in\boldsymbol{X}}\left(\left(\frac{\boldsymbol{x} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)^T \boldsymbol{C}^{(g)^{-1}} \left(\frac{\boldsymbol{x} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}\right)\right).$$

Chapter 2

Design

This chapter presents the design of the surrogate model selection for DTS-CMA-ES based on the fitness landscape analysis. The design continues on the research done by my supervisor and his colleagues in [40, 42].

It is important to understand the type of the data used to train a surrogate model in the DTS-CMA-ES algorithm. Using the explanation from [40]: For each generation g of the DTS-CMA-ES algorithm set of surrogate models $\mathcal{M}$ are trained on a training set $\mathcal{T}$. The training set $\mathcal{T}$ is a subset of all evaluated data points archive $\mathcal{A}$. Afterwards, the surrogate model $M \in \mathcal{M}$ is utilized to select new population $\mathcal{P}$. The question is how to select the most convenient surrogate model from sets $\mathcal{A}, \mathcal{T}, \mathcal{P}$? Figure 2.1 shows how this problem can be implemented in the Rice's framework defined in Section 1.4.

The design of each component of the presented framework for surrogate model selection problem is described in the following sections. The main goal of this chapter is to design the selection mapping.

2.1 Data space

In DTS-CMA-ES, three sets of data points are emerging. The first one is an archive $\mathcal{A}$ containing all f-evaluated data points $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i) \mid i = 1, \ldots, n\}$, where n is a number of f-evaluated points. The second one is the training set $\mathcal{T}$ containing f-evaluated data points which are a subset of $\mathcal{A}$ and are utilized for fitting a surrogate model in DTS-CMA-ES. The training set is selected to contains data points that are close to the space where the surrogate model is fitted. The last set is sampled population $\mathcal{P}$, where the function values are unknown. These sets are changing each generation.

The problems used for retrieving the data $\{\mathcal{A}^{(i)}, \mathcal{T}^{(i)}, \mathcal{P}^{(i)} \mid i = 1, \ldots, g\}$ in this thesis were obtained from running the DTS-CMA-ES algorithm on Black-Box Optimization Benchmarks from the COmparing Continuous Optimisers

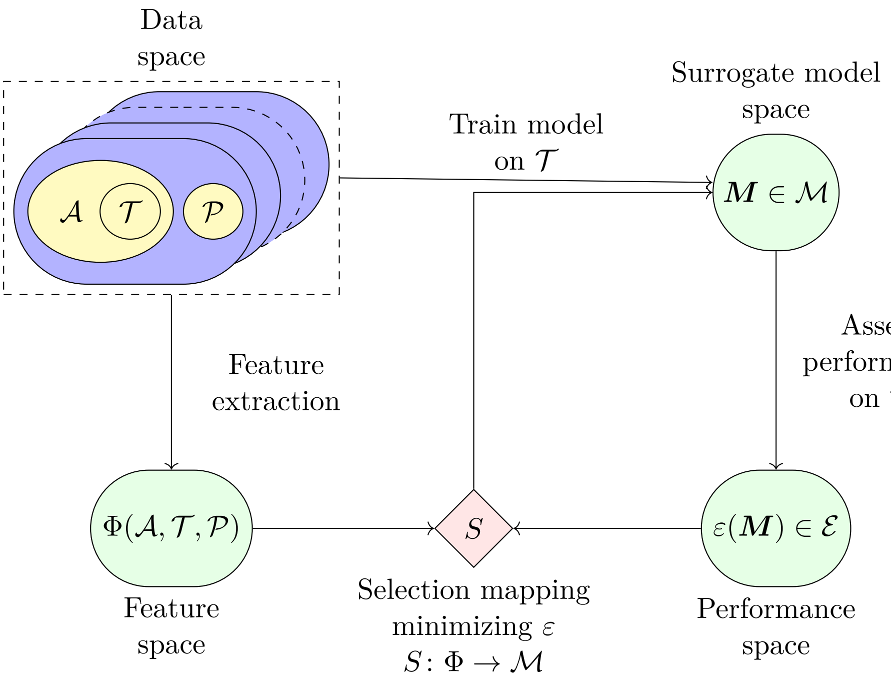
Data space
$\mathcal{A}$
$\mathcal{T}$
$\mathcal{P}$
Train model on $\mathcal{T}$
Surrogate model space
$\boldsymbol{M} \in \mathcal{M}$
Feature extraction
Assess performance on $\mathcal{P}$
$\Phi(\mathcal{A}, \mathcal{T}, \mathcal{P})$
S
$\varepsilon(\boldsymbol{M}) \in \mathcal{E}$
Feature space
Selection mapping minimizing ε
$S \colon \Phi \to \mathcal{M}$
Performance space

Figure 2.1: Modified Rice's framework for surrogate model selection in DTS-CMA-ES.

(COCO) platform, namely, problems in dimensions 2, 3, 5, 10, and 20 on instances 11-15 [16, 17].

2.2 Feature space

The features are computed on datasets $\mathcal{A}, \mathcal{T}$, and $\mathcal{T} \cup \mathcal{P}$. Some of fitness landscape features described in Section 1.4.1 need additional evaluations of the fitness function f (e.g. Local search features) and therefore, they are not suitable if the fitness function is expensive to evaluate. The following low-level feature sets were measured: y-Distribution, Levelset, Meta-Model, Nearest-Better Clustering, Dispersion, Information Content, and CMA-ES features.

The data generated for this thesis were already used in [42]. Unlike in [42], more metrics in the performance space are investigated (see Section 2.4). Features were calculated using the code from R-package flacco [24] reimplemented in MATLAB language.

The features were calculated from runs of the DTS-CMA-ES algorithm on the benchmark functions mentioned in Section 2.1. Sets $\{\mathcal{A}^{(i)}, \mathcal{T}^{(i)}, \mathcal{P}^{(i)} \mid i =$

$1, \ldots, g\}$ were extracted for 25 uniformly selected generations for 8 considered surrogate models (see Section 2.3).

The algorithm was terminated if one of the following two conditions is true:

(1) the target fitness value 10^{-8} is found, or

(2) the number of evaluations of the optimized fitness function f reached $250n$.

2.3 Model space

Few surrogate models can be connected with the DTS-CMA-ES because the used version of the algorithm needs an estimation of uncertainty for the corresponding prediction. Therefore, we have tested Gaussian processes with various covariance functions.

To measure performance in the performance space following covariance functions were used: $\kappa_{\text{LIN}}, \kappa_{\text{SE}}, \kappa_{\text{RQ}}, \kappa_{\text{SE}}, \kappa_{\text{Mat}}^{\frac{5}{2}}, \kappa_{\text{NN}}, \kappa_{\text{Gibbs}}$, and $\kappa_{\text{SE+Q}}$. Covariance functions for GPs were described in Section 1.2.2.

2.4 Performance space

Performance can be measured with a variety of evaluation metrics and the question is which metric would be the most convenient for surrogate model selection task. In [42], authors used Ranking Difference Error. However, error measures such as MSE, MAE, Kendall τ or R^2 could be more convenient for the investigation of the relationships between model performance and fitness landscape features.

RDE Ranking Difference Error is an useful metric because CMA-ES state variables depends on the best μ points from the population. Therefore, the ranking is important and might provide sufficient information. RDE is computed as follows:

$$\text{RDE}_{\mu}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\sum_{i:(\rho(\boldsymbol{y}))_i \leq \mu} |(\rho(\boldsymbol{y}))_i - (\rho(\hat{\boldsymbol{y}}))_i|}{\max_{\pi \in \text{Permutations of } (1,\ldots,\lambda)} \sum_{i:\pi(i) \leq \mu} |i - \pi(i)|},$$

where $(\rho(\boldsymbol{y}))_i$ is the rank of y_i in the $\boldsymbol{y}$ vector.

Kendall rank correlation coefficient is a correlation coefficient which takes into account the ranking of two measured series of observations. It is computed as

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)},$$

where n_c is a number of concordant pairs and n_d is a number of discordant pairs. Concordant pairs are defined by set

$$\{((y_i, \hat{y}_i), (y_j, \hat{y}_j)) \mid (y_i > y_j \wedge \hat{y}_i > \hat{y}_j) \vee (y_i < y_j \wedge \hat{y}_i < \hat{y}_j), i < j\}$$

and discordant by

$$\{((y_i, \hat{y}_i), (y_j, \hat{y}_j)) \mid (y_i > y_j \wedge \hat{y}_i < \hat{y}_j) \vee (y_i > y_j \wedge \hat{y}_i < \hat{y}_j), i < j\}.$$

$\boldsymbol{R^2}$ is a coefficient of determination calculated from

$$R^2 = \frac{\sum_{i=1}^{\lambda}(\hat{y}_i - m(\boldsymbol{y}))^2}{\sum_{i=1}^{\lambda}(y_i - m(\boldsymbol{y}))^2},$$

where $m(\boldsymbol{y})$ is the mean value of the vector $\boldsymbol{y}$.

MSE Mean Squared Error which is a typical error measure for many problems.

$$\text{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{\lambda}\sum_{i=1}^{\lambda}(y_i - \hat{y}_i)^2$$

MAE Mean Absolute Error which is similar to MSE but uses absolute differences.

$$\text{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{\lambda}\sum_{i=1}^{\lambda}|y_i - \hat{y}_i|$$

2.5 Selection mapping

Selection mapping $S \colon \Phi \to \mathcal{M}$ represents a component which can map landscape features $\phi \in \Phi$ to a model $M \in \mathcal{M}$ such that it minimizes the error ε this model would have. Intelligibly, the goal is to select the best model for a given set of landscape features.

For selection mapping we utilized the measured fitness landscape features (see Section 2.2) and errors for eight different GP kernels. To obtain labels for classification, the minimal error for every data point is found and its corresponding kernel used as a label. However, more surrogate models can yield the same error for a given problem or the difference between errors is insignificant. Therefore, we measured accuracy of each classifier in two ways, the first measures the exact match and the second measures matches for the group of equally performing models.

We have followed the research in [40], where authors used a classification tree for selection mapping. However, the results were not very convincing. Therefore, we tests more classification models to have better performance.

One way to approach this is using regression models instead of classification models and predict the error ε. This could provide greater flexibility with selecting the best model such as recommending more models with similar performance. This approach was used in [37] for the flow shop scheduling problem. Authors used multiple methods based on ranking, regression, and classification for recommending the most suitable model. Additionally, the classification approach could be exhaustively tested using more models and their parameters.

We implement three methods: classification, regression, and multi-label classification.

2.5.1 Classification approach

Classifier $S_c \colon \Phi \to \mathcal{M}$ can be trained on labels of the best performing models. To obtain a label for a data point, the minimal error value for each GP kernel can be found and its kernel set as a label. It is not always clear which model should be selected as a label because multiple models can have equal errors. To address this ambiguity, multi-label classifiers are considered in approach from Section 2.5.3.

The classifier S_c is then trained with the training dataset and later can be used as a recommender for a new data point in the run of DTS-CMA-ES algorithm.

Eight different kernels are considered and the classifier should be able to recommend one of them. Many of the classification models are only defined for two classes, to solve this obstacle known strategies for making multiclass classifier from binary classifiers can be used. The two used in this thesis are: *One-vs-One* and *One-vs-Rest* [5].

One-vs-One strategy trains $\frac{n(n-1)}{2}$ classifiers, where n is the number of classes, for every pair of classes. The dataset is split for every pair of classes and a classifier is trained on this new dataset. The resulting class is selected using votes of individual classifiers.

One-vs-Rest strategy trains n classifiers. For every class a binary classifier is trained to classify if a data point belongs or does not belong to a given class. Problem with this strategy is that constructed datasets are likely to be imbalanced.

2.5.2 Regression approach

A regression model $S_r \colon \Phi \to \mathcal{E}^{|\mathcal{M}|}$ can be trained to predict an error of a surrogate model for given landscape features. The S_r model yields errors from which a minimum is found and its corresponding surrogate model is recommended.

Because some regression models yield only one prediction, for every surrogate model one regression model can be trained to predict multiple errors.

2.5.3 Multi-label approach

We use multi-label classification because some models have very similar performance for some data points and more labels could be assigned to them.

From the original dataset the best performing models are found and used as labels for classification model S_c training. Trained model is then capable to predict multiple labels for given landscape features. However, for a fair comparison with the single-label classification and with the regression approach, only one label has to be predicted.

A regression model is utilized to predict best performing model to select single-label from predicted labels by the multi-label classifier. Regression considers only labels predicted by the multi-label classifier and from the regression model the best performing model is selected as the final recommended label.

Because multi-label classifier can yield an empty prediction, a single-label classifier is used together with the multi-label classifier. The single-label classifier is only used when the multi-label classifier yields empty prediction. The single-label classifier could be replaced with a naive method that recommends the most frequent label from the training set.

Pseudocode 7 shows how predictions from classifiers are mixed together in order to get predictions for the unknown data points.

Algorithm 7: predict

```
Input: test data $\boldsymbol{X}$, trained multi-label classifier $S_c^m$, trained single-label
       classifier $S_c^s$, trained regression model $S_r$
Output: $\hat{\boldsymbol{y}}$ vector of predicted kernels
$\hat{\boldsymbol{Y}}^m \leftarrow S_c^m(\boldsymbol{X})$      // predict multiple kernels for each data point
$\hat{\boldsymbol{y}}^s \leftarrow S_c^s(\boldsymbol{X})$     // predict exactly one kernel for each data point
$\hat{\boldsymbol{Y}}^{\text{error}} \leftarrow S_r(\boldsymbol{X})$                  // predict error of each kernel
$\hat{\boldsymbol{Y}}^{\text{combined}} \leftarrow \text{merge}(\hat{\boldsymbol{Y}}^m, \hat{\boldsymbol{y}}^s)$   // merge predictions such that for
                                  empty multi-label prediction insert
                                  the single-label prediction
$\hat{\boldsymbol{Y}}^{\text{error}} \leftarrow \text{setInfinity}(\hat{\boldsymbol{Y}}^{\text{combined}}, \hat{\boldsymbol{Y}}^r)$
  // from regressed errors set non
     relevant predictions to infinity
$\hat{\boldsymbol{y}} \leftarrow \arg\min(\hat{\boldsymbol{Y}}^{\text{error}})$         // select kernels where predicted
                                  error was minimal
```

Chapter 3

Experiments

Experiments are implemented with open source Python [49] libraries: Jupyter Notebook [25], scikit-learn [38], Numpy [36], Pandas [30], and TensorFlow [1]. Source code is available on a USB flash drive attached to this thesis.

The design of measured experiments is described in Section 2.5 and the experiments are compared w.r.t. following measures:

Accuracy is defined as a fraction $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}}$, where TP is a number true positive classifications, TN is a number true negative classification, and $\mathrm{P}+\mathrm{N}$ is a number of all data points.

Sensitivity measures how often data points for a binary classification are correctly classified as the first class. It is defined by a fraction $\frac{\mathrm{TP}}{\mathrm{P}}$, where P is a number of values from a measured class.

Specificity measures how often data points for a binary classification are correctly classified as the second class. It is defined by a fraction $\frac{\mathrm{TN}}{\mathrm{N}}$, where N is a number of values that are not the measured class.

F_1 **score** is a measurement often used for imbalanced classification and it is defined as a harmonic mean of precision and recall. It is defined by a fraction $\frac{2\,\mathrm{TP}}{2\,\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$, where FN is a number of false negative classifications.

Considering that multiple surrogate models can perform equally in some cases, we have to define when the classified point can be marked as TP, TN, FP, or FN. Therefore, we measure experiments with two versions for each score measure. The first version of a measurement is calculated from exact matches of classified class and the true class. The second measurement is calculated from loose matches of similarly performing of surrogate models. The loose measures are marked with a subscript 2 as: accuracy_2, sensitivity_2, specificity_2, and $\mathrm{F}_1\,\text{score}_2$.

3.1 Data preprocessing

Data of measured fitness landscape features and errors of different GPs kernels contains some numbers that are not in the set of real numbers $\mathbb{R}$. How to deal with infinity, negative infinity and NaN values had been iteratively discovered.

For landscape features NaN values were replaced with the mean value, infinity values were replaced with a maximum and negative infinity values were replaced with a minimum of the given feature. Feature values are later scaled to have zero mean and unit variance.

The measured errors contain some NaN values. In this case, NaN values are emerging when the model could not be fitted. Therefore, NaN values are replaced with a maximum of the given error. Infinity values are replaced also with a maximum value.

3.2 Selection mapping

As stated in Section 2.5 we are using different classifiers $S\colon \Phi \to \mathcal{M}$ to predict most convenient model M for a set of features Φ. The following hyperparameters of machine learning models we used as default.

Decision tree and random forest classifiers used as criterion for splitting Gini index.

Support vector classifier used a linear kernel for faster training and maximum iteration was set to 1000.

Artificial neural network was trained with two hidden layers with 50 and 25 neurons respectively, dropout chance was set to 0.2, and training epochs was set to 20.

Regression tree and random forest of regression trees used as criterion for splitting MSE.

Support vector regression used a linear kernel, ε-insensitive loss function, and maximum iteration was set to 1000.

Hyperparameters, that differs from stated defaults are described in respective sections.

3.2.1 Classification

Classification models were trained on 80 % of the dataset and hyperparameters settings were searched with a 5-fold cross validation method. The depth of trees are set low to find if the problem is easy to separate and not too high because of overfitting.

The considered models and their hyperparameters are:

Decision tree classifier with a maximum depth parameter $2, 3, \ldots, 19$.

Random forest classifier with a maximum depth parameter $2, 3, \ldots, 9$ and a number of estimators parameter set to $100, 200, \ldots, 1000$.

Support vector classifier for one-vs-rest strategy and parameter C set to $0.01, 0.1, 1, 10, 100$ and for one-vs-one strategy is only one parameter considered as $C = 0.1$.

Artificial neural network used ReLU activation function in hidden layers and softmax activation function in the output layer.

3.2.2 Regression

Regression model is trained to predict GPs kernel error for given landscape features. Model with minimal predicted error is selected as recommended and compared with the true minimal error model.

Regression models were trained on the same 80 % part of the dataset as the classification models. Hyperparameters were found by searching the hyperparameter space with a 5-fold cross validation method.

The considered models and their hyperparameters are slightly different. More hyperparameters could be explored, however, the regression approach does not seem to outperform baseline method. Following models for regression were measured:

Regression tree with a maximum depth parameter $10, 11, \ldots, 18$.

Random forest of regression trees with a maximum depth parameter $6, 7, 8, 9$ and a number of estimators parameter set to 100.

Support vector regression with one-vs-rest strategy and parameter C set to $0.01, 0.1$ and parameter ε set to $0, 0.1$.

Artificial neural network used ReLU activation function in hidden layers and linear activation function in the output layer.

3.2.3 Multi-label classification

Multi-label classification model is trained to give a prediction about viable GPs kernels. Predictions can contain none or multiple kernels which is a problem for comparing with other methods. To obtain a single recommended model, both previous methods are utilized to select single GP kernel for a given data point. Exact explanation of this method is in Section 2.5.3.

Multi-label classification models were trained on the same 80 % part of the dataset as the classification models. Hyperparameters were found by searching the hyperparameter space with a 5-fold cross validation method.

Following models were trained:

Decision tree with a maximum depth parameter $6, 7, \ldots, 19$.

Random forest of classification trees with a maximum depth parameter $6, 7, \ldots, 10$ and a number of estimators parameter set to 100.

Support vector classifier with one-vs-rest strategy and parameter C set to $0.01, 0.1, 1, 10, 100$.

Artificial neural network used ReLU activation function in hidden layers and sigmoid activation function in the output layer.

3.3 Accuracy comparison

The final performance was measured on the test dataset containing 20 % of the original dataset.

We used a baseline model that recommends the most frequent class in the training dataset, solely for comparing if the naive solution is not the best one.

Table 3.1 and Table 3.2 compares models in terms of accuracy and accuracy$_2$ respectively. For comparisons in terms of sensitivity, specificity and F$_1$ score see Appendix B.

Table 3.1: Exact accuracy for each considered model trained with landscape features and predicting the best performing model w.r.t. different error measures. The table is divided into three categories based on used strategies for classification.

Model	Error measure				
	RDE	MSE	R^2	MAE	Kendall
	Single-label classification				
Baseline	0.220	0.259	0.207	0.266	0.390
Decision tree	0.239	0.294	0.285	0.295	0.396
Random forest	**0.250**	**0.308**	**0.300**	**0.319**	**0.401**
SVC one-vs-rest	0.239	0.288	0.282	0.289	0.392
SVC one-vs-one	0.239	0.286	0.287	0.291	0.394
Neural network	0.247	0.295	0.299	0.300	0.399
	Regression				
Regression tree	0.192	0.199	0.156	0.203	0.358
Random forest	0.198	0.214	0.165	0.214	0.372
SVR	0.193	0.175	0.161	0.182	0.362
Neural network	0.187	0.258	0.172	0.102	0.370
	Multi-label classification				
Decision tree	0.221	0.243	0.242	0.269	0.351
Random forest	0.246	0.280	**0.300**	0.317	**0.401**
SVC	0.194	0.246	0.266	0.278	0.398
Neural network	0.248	0.281	**0.300**	0.315	**0.401**

Table 3.2: Loose accuracy$_2$ for each considered model trained with landscape features and predicting the best performing model w.r.t. different error measures. The table is divided into three categories based on used strategies for classification.

Model	Error measure				
	RDE	MSE	R^2	MAE	Kendall
Single-label classification					
Baseline	0.309	0.350	0.209	0.293	0.422
Decision tree	0.319	0.383	0.293	0.323	0.428
Random forest	0.333	**0.399**	**0.308**	**0.346**	**0.434**
SVC one-vs-rest	0.323	0.380	0.290	0.317	0.423
SVC one-vs-one	0.324	0.380	0.295	0.320	0.426
Neural network	0.329	0.383	0.306	0.327	0.432
Regression					
Regression tree	0.318	0.315	0.163	0.247	0.396
Random forest	0.321	0.331	0.170	0.257	0.412
SVR	0.315	0.268	0.166	0.219	0.400
Neural network	0.308	0.350	0.175	0.134	0.410
Multi-label classification					
Decision tree	0.320	0.347	0.249	0.304	0.387
Random forest	**0.334**	0.389	**0.308**	**0.346**	**0.434**
SVC	0.309	0.353	0.271	0.313	0.429
Neural network	**0.334**	0.388	**0.308**	0.344	**0.434**

3.4 Discussion and future work

The baseline model was outperformed with almost every presented classification method. However, the differences between highest accuracy scores and the baseline scores are very small. From the accuracy tables in the previous section it is clear that the best model is random forest for almost all considered approaches. For the purpose of DTS-CMA-ES random forest does not have to be the most suitable option because time complexity of prediction might be too large. The neural network model might be better in terms of time but its performance is slightly worse. Single-label classification approach have the highest accuracy, but its accuracy is very similar to the multi-label classification method. The advantage of the multi-label classification is that it provides more flexibility for tuning the settings and hence provides more room for improvement.

The poor performance of considered models does not suggest if trained classifiers should be employed for improvement of the DTS-CMA-ES. To provide more clarity to this question, DTS-CMA-ES should be run on the benchmark functions and its performance should be assessed.

From my perspective, future research should be focused on the regression approach because in many applications this approach have satisfying results, for instance, in [37] or in SATzilla [51], but in this case this approach was significantly worse compared to other approaches. Additionally, with improvement of the regression approach, the multi-label approach could be also improved since it depends on it.

More investigation could be focused on imbalance of the data and also researching if more measured benchmark functions and the data from them would help classifiers to better separate the landscape feature space.

Another improvement of presented methods could be derived solely from improvement of FLA research by obtaining more descriptive landscape features. The FLA features could also be improved with feature selection that could reduce the feature space and help machine learning models to have better performance.

Additional improvement could be from utilizing more surrogate models than GP and its kernels, because GP kernels could behave too similarly for given data and hence they could be hard to distinguish in FLA space.

Conclusion

Expensive black-box optimization is an important research topic because many engineering tasks are expansive to evaluate. An engineering experiment might cost significant amount of money or might take a lot of time to evaluate. Therefore, having a fast and reliable algorithm for optimizing such task is very valuable.

This thesis investigated how to utilize fitness landscape analysis for predicting the most suitable surrogate model the DTS-CMA-ES algorithm. In the first chapter, an introduction of regression models suitable for surrogate assisted CMA-ES was presented. The DTS-CMA-ES algorithm was explained and closely studied to describe how the surrogate models are used. Further, framework for algorithm selection problem was presented and features derived from fitness landscape analysis were described. The second chapter explains how the framework for algorithm selection problem was used in DTS-CMA-ES for surrogate model recommendation through fitness landscape analysis.

A design of various methods for classifying the data from FLA to predict the most convenient surrogate model were presented. In the last chapter, settings of experiments with the classification methods were introduced in detail and they were compared with a variety of classification metrics. Experiments were also compared with a naive method of recommending the most frequent model as a baseline.

Used classification methods had better performance than the baseline method, however, the accuracy is better only by a small margin. From the results of sensitivity it seems that the classifiers often learn only how to distinguish between two classes. Possible problems might be with an imbalance of classes in the training dataset and/or with similar performance of some surrogate models for some data points. The later one was addressed with a multi-label classification methods but this approach did not bring significant improvement. Although the accuracy scores are not very large, trained classifiers might improve performance of the DTS-CMA-ES algorithm because even a small improvement in accuracy might useful.

Further research, that is beyond the scope of this thesis, might be needed. Number FLA features could be reduced with a feature selection method and in reduced space classifiers might have better performance. Another improvement may be found by exploring the regression approach which had satisfying results in [37, 51].

Bibliography

1. ABADI, Martín et al. Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.
2. AUDET, Charles; HARE, Warren. *Derivative-Free and Blackbox Optimization*. Springer International Publishing, 2017. ISBN 978-3-319-68913-5.
3. BAJER, Lukáš; PITRA, Zbyněk; REPICKÝ, Jakub; HOLEŇA, Martin. Gaussian process surrogate models for the CMA evolution strategy. *Evolutionary computation*. 2019, vol. 27, no. 4, pp. 665–697.
4. BEYER, Hans-Georg; SCHWEFEL, Hans-Paul. Evolution strategies–A comprehensive introduction. *Natural computing*. 2002, vol. 1, no. 1, pp. 3–52.
5. BISHOP, Christopher M. *Pattern recognition and machine learning*. springer, 2006. ISBN 978-0387310732.
6. BOX, G. E. P.; WILSON, K. B. On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*. 1951, vol. 13, no. 1, pp. 1–38. Available from DOI: `10.1111/j.2517-6161.1951.tb00067.x`.
7. BREIMAN, Leo. Bagging predictors. *Machine learning*. 1996, vol. 24, no. 2, pp. 123–140.
8. BREIMAN, Leo; FRIEDMAN, Jerome; STONE, Charles J; OLSHEN, Richard A. *Classification and regression trees*. CRC press, 1984. ISBN 978-1138469525.
9. CARPENTER, William C; BARTHELEMY, J-FM. A comparison of polynomial approximations and artificial neural nets as response surfaces. *Structural Optimization*. 1993, vol. 5, no. 3, pp. 166–174.

10. CASSIOLI, Andrea. *A Tutorial on Black–Box Optimization.* Available also from: `https://www.lix.polytechnique.fr/~dambrosio/blackbox_material/Cassioli_1.pdf`. [Online; visited 4. 2. 2020].

11. DRUCKER, Harris; BURGES, Christopher JC; KAUFMAN, Linda; SMOLA, Alex J; VAPNIK, Vladimir. Support vector regression machines. In: *Advances in neural information processing systems.* 1997, pp. 155–161.

12. GASPAR-CUNHA, Antonio; VIEIRA, Armando. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *Int. J. Comput. Syst. Signal.* 2005, vol. 6, no. 1, pp. 18–36.

13. GIBBS, Mark N. *Bayesian Gaussian processes for regression and classification.* 1998. PhD thesis. Citeseer.

14. HAN, Zhong-Hua; ZHANG, Ke-Shi, et al. Surrogate-based optimization. *Real-world applications of genetic algorithms.* 2012, pp. 343–362.

15. HANSEN, Nikolaus. The CMA evolution strategy: a comparing review. In: *Towards a new evolutionary computation.* Springer, 2006, pp. 75–102.

16. HANSEN, Nikolaus; AUGER, Anne; FINCK, Steffen; ROS, Raymond. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* 2010. Technical report. Citeseer.

17. HANSEN, Nikolaus; AUGER, Anne; FINCK, Steffen; ROS, Raymond. *Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup.* 2012. Technical report. Citeseer.

18. HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009. ISBN 978-0-387-84858-7.

19. HAYKIN, Simon. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994. ISBN 978-0-02-352761-6.

20. HEJL, Vojtěch. *Selection of surrogate models for evolutionary black-box optimization in noisy environment.* Thákurova 9, 160 00 Praha 6, Czech Republic, 2018. Master's thesis. Faculty of Information Technology, Czech Technical University in Prague.

21. HORNIK, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural Networks.* 1991, vol. 4, no. 2, pp. 251–257. Available from DOI: `10.1016/0893-6080(91)90009-t`.

22. JIN, Yaochu; OLHOFER, Markus; SENDHOFF, Bernhard. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on evolutionary computation.* 2002, vol. 6, no. 5, pp. 481–494.

23. KERSCHKE, Pascal; PREUSS, Mike; WESSING, Simon; TRAUTMANN, Heike. Detecting funnel structures by means of exploratory landscape analysis. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 2015, pp. 265–272.

24. KERSCHKE, Pascal; TRAUTMANN, Heike. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-package flacco. In: BAUER, Nadja; ICKSTADT, Katja; LÜBKE, Karsten; SZEPANNEK, Gero; TRAUTMANN, Heike; VICHI, Maurizio (eds.). *Applications in Statistical Computing – From Music Data Analysis to Industrial Quality Improvement*. Springer, 2019, pp. 93–123. Studies in Classification, Data Analysis, and Knowledge Organization. Available from DOI: 10.1007/978-3-030-25147-5_7.

25. KLUYVER, Thomas et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (eds.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 2016, pp. 87–90.

26. LOSHCHILOV, Ilya; SCHOENAUER, Marc; SEBAG, Michele. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. In: *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 2012, pp. 321–328.

27. LOSHCHILOV, Ilya; SCHOENAUER, Marc; SEBAG, Michele. Intensive surrogate model exploitation in self-adaptive surrogate-assisted cma-es (saacm-es). In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 2013, pp. 439–446.

28. LUNACEK, Monte; WHITLEY, Darrell. The dispersion metric and the CMA evolution strategy. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*. ACM Press, 2006. Available from DOI: 10.1145/1143997.1144085.

29. MATÉRN, Bertil. *Spatial variation*. 1960. Technical report.

30. MCKINNEY, Wes et al. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*. 2010, vol. 445, pp. 51–56.

31. MERSMANN, Olaf; BISCHL, Bernd; TRAUTMANN, Heike; PREUSS, Mike; WEIHS, Claus; RUDOLPH, Günter. Exploratory landscape analysis. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 2011, pp. 829–836.

32. MERSMANN, Olaf; PREUSS, Mike; TRAUTMANN, Heike. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In: *International Conference on Parallel Problem Solving from Nature*. 2010, pp. 73–82.

33. MINSKY, Marvin; PAPERT, Seymour. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969. ISBN 9780262534772.

34. MUÑOZ, Mario A; KIRLEY, Michael; HALGAMUGE, Saman K. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE transactions on evolutionary computation.* 2014, vol. 19, no. 1, pp. 74–87.

35. MUÑOZ, Mario A; SUN, Yuan; KIRLEY, Michael; HALGAMUGE, Saman K. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences.* 2015, vol. 317, pp. 224–245.

36. OLIPHANT, Travis E. *A guide to NumPy.* Trelgol Publishing USA, 2006. ISBN 978-1517300074.

37. PAVELSKI, Lucas Marcondes; DELGADO, Myriam Regattieri; KESSACI, Marie-Éléonore. Meta-learning on flowshop using fitness landscape analysis. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* 2019, pp. 925–933.

38. PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research.* 2011, vol. 12, no. Oct, pp. 2825–2830.

39. PITRA, Zbyněk; BAJER, Lukáš; HOLEŇA, Martin. Doubly trained evolution control for the surrogate CMA-ES. In: *International Conference on Parallel Problem Solving from Nature.* 2016, pp. 59–68.

40. PITRA, Zbyněk; BAJER, Lukáš; HOLEŇA, Martin. Knowledge-based Selection of Gaussian Process Surrogates. In: *Workshop & Tutorial on Interactive Adaptive Learning.* 2019, p. 48.

41. PITRA, Zbyněk; BAJER, Lukáš; REPICKÝ, Jakub; HOLEŇA, Martin. Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17.* ACM Press, 2017. Available from DOI: 10.1145/3067695.3082539.

42. PITRA, Zbyněk; REPICKỲ, Jakub; HOLEŇA, Martin. Landscape analysis of gaussian process surrogates for the covariance matrix adaptation evolution strategy. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* 2019, pp. 691–699.

43. RASMUSSEN, Carl. *Gaussian processes for machine learning.* Cambridge, Mass: MIT Press, 2006. ISBN 978-0262182539.

44. RECHENBERG, Ingo. Evolutionsstrategien. In: *Simulationsmethoden in der Medizin und Biologie.* Springer, 1978, pp. 83–114.

45. RICE, John R. et al. The algorithm selection problem. *Advances in computers.* 1976, vol. 15, no. 65-118, pp. 5.

46. ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*. 1958, vol. 65, no. 6, pp. 386.

47. RUMELHART, David E; HINTON, Geoffrey E; WILLIAMS, Ronald J. *Learning internal representations by error propagation*. 1985. Technical report. California Univ San Diego La Jolla Inst for Cognitive Science.

48. ULMER, Holger; STREICHERT, Felix; ZELL, Andreas. Evolution strategies with controlled model assistance. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*. 2004, vol. 2, pp. 1569–1576.

49. VAN ROSSUM, Guido; DRAKE, Fred L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 978-1441412690.

50. VAPNIK, Vladimir N. The nature of statistical learning theory. 1995. ISBN 978-0-387-94559-0.

51. XU, L.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research*. 2008, vol. 32, pp. 565–606. ISSN 1076-9757. Available from DOI: `10.1613/jair.2490`.

Appendix A

Acronyms

ANN Artificial Neural Network

BBO Black-Box Optimization

CMA-ES Covariance Matrix Adaptation Evolution Strategy

DTS-CMA-ES Doubly Trained Surrogate Covariance Matrix Adaptation Evolution Strategy

ES Evolution Strategy

FLA Fitness Landscape Analysis

FN False Negative

FP False Positive

GP Gaussian Process

MAE Mean Absolute Error

MSE Mean Squared Error

MZOE Mean Zero One Error

NaN Not a Number

NN Neural Network

RDE Ranking Difference Error

SVC Support Vector Classifier

SVM Support Vector Machine

SVR Support Vector Regression

TN True Negative

TP True Positive

Appendix B

Measurements

B.1 Classification

The following tables show measured metrics for defined single-label classifiers. For each classifier and each class sensitivity, specificity and F_1 score were computed.

Table B.1: Measured F$_1$ score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Decision tree						
kernel	F$_1$ score	sensitivity	specificity	F$_1$ score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.000	0.000	1.000	0.000	0.000	1.000
SE	0.000	0.000	1.000	0.000	0.000	1.000
LIN	0.018	0.014	0.999	0.018	0.009	0.999
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.472	0.707	0.462	0.472	0.675	0.470
RQ	0.030	0.012	0.992	0.030	0.015	0.994
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.349	0.573	0.611	0.349	0.559	0.619
MSE						
NN	0.156	0.134	0.933	0.156	0.114	0.934
SE	0.115	0.074	0.979	0.115	0.064	0.985
LIN	0.090	0.132	0.996	0.090	0.049	0.995
Q	0.192	0.193	0.945	0.192	0.154	0.944
Mat	0.542	0.583	0.624	0.542	0.603	0.665
RQ	0.179	0.121	0.948	0.179	0.109	0.956
SE+Q	0.045	0.029	0.992	0.045	0.024	0.993
Gibbs	0.333	0.519	0.713	0.333	0.478	0.710
R^2						
NN	0.250	0.240	0.875	0.250	0.242	0.876
SE	0.093	0.057	0.984	0.093	0.055	0.984
LIN	0.453	0.555	0.851	0.453	0.546	0.854
Q	0.218	0.226	0.891	0.218	0.224	0.892
Mat	0.382	0.553	0.651	0.382	0.549	0.650
RQ	0.039	0.021	0.996	0.039	0.020	0.996
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.248	0.254	0.902	0.248	0.253	0.902
MAE						
NN	0.174	0.122	0.945	0.174	0.132	0.947
SE	0.071	0.041	0.986	0.071	0.039	0.987
LIN	0.137	0.120	0.996	0.137	0.081	0.996
Q	0.037	0.022	0.997	0.037	0.019	0.997
Mat	0.465	0.662	0.510	0.465	0.658	0.515
RQ	0.159	0.121	0.944	0.159	0.104	0.944
SE+Q	0.184	0.133	0.949	0.184	0.138	0.951
Gibbs	0.313	0.394	0.780	0.313	0.390	0.780
Kendall						
NN	0.060	0.033	0.989	0.060	0.032	0.990
SE	0.000	0.000	1.000	0.000	0.000	1.000
LIN	0.603	0.932	0.176	0.603	0.921	0.174
Q	0.126	0.086	0.950	0.126	0.082	0.951
Mat	0.000	0.000	1.000	0.000	0.000	1.000
RQ	0.001	0.001	1.000	0.001	0.001	1.000
SE+Q	0.055	0.022	0.995	0.055	0.029	0.996
Gibbs	0.175	0.133	0.949	0.175	0.130	0.951

Table B.2: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Random forest						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.071	0.045	0.980	0.071	0.042	0.980
SE	0.011	0.007	0.997	0.011	0.006	0.997
LIN	0.061	0.027	0.997	0.061	0.032	0.998
Q	0.044	0.020	0.995	0.044	0.023	0.996
Mat	0.475	0.693	0.494	0.475	0.659	0.502
RQ	0.069	0.049	0.978	0.069	0.038	0.979
SE+Q	0.004	0.001	0.999	0.004	0.002	0.999
Gibbs	0.365	0.573	0.643	0.365	0.558	0.652
MSE						
NN	0.098	0.082	0.975	0.098	0.059	0.974
SE	0.052	0.037	0.994	0.052	0.027	0.996
LIN	0.119	0.180	0.995	0.119	0.067	0.995
Q	0.197	0.182	0.956	0.197	0.149	0.956
Mat	0.573	0.760	0.463	0.573	0.773	0.502
RQ	0.138	0.100	0.966	0.138	0.080	0.970
SE+Q	0.104	0.060	0.974	0.104	0.062	0.976
Gibbs	0.293	0.376	0.801	0.293	0.339	0.798
R^2						
NN	0.241	0.206	0.911	0.241	0.205	0.912
SE	0.076	0.044	0.992	0.076	0.042	0.992
LIN	0.469	0.643	0.815	0.469	0.636	0.818
Q	0.139	0.093	0.966	0.139	0.094	0.966
Mat	0.406	0.699	0.545	0.406	0.693	0.544
RQ	0.055	0.029	0.992	0.055	0.030	0.993
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.221	0.189	0.935	0.221	0.189	0.935
MAE						
NN	0.164	0.096	0.970	0.164	0.108	0.972
SE	0.037	0.022	0.997	0.037	0.019	0.998
LIN	0.185	0.163	0.997	0.185	0.110	0.997
Q	0.041	0.024	0.997	0.041	0.021	0.997
Mat	0.499	0.787	0.437	0.499	0.780	0.443
RQ	0.118	0.081	0.975	0.118	0.069	0.976
SE+Q	0.178	0.119	0.961	0.178	0.124	0.964
Gibbs	0.324	0.406	0.787	0.324	0.400	0.786
Kendall						
NN	0.061	0.037	0.990	0.061	0.033	0.991
SE	0.002	0.001	1.000	0.002	0.001	1.000
LIN	0.603	0.977	0.095	0.603	0.969	0.093
Q	0.029	0.016	0.990	0.029	0.015	0.991
Mat	0.001	0.001	1.000	0.001	0.001	1.000
RQ	0.008	0.002	0.999	0.008	0.004	0.999
SE+Q	0.067	0.023	0.995	0.067	0.035	0.997
Gibbs	0.165	0.114	0.973	0.165	0.107	0.974

Table B.3: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

SVC one-vs-rest						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.092	0.060	0.967	0.092	0.058	0.968
SE	0.004	0.002	0.997	0.004	0.002	0.997
LIN	0.015	0.006	0.999	0.015	0.007	1.000
Q	0.059	0.030	0.985	0.059	0.033	0.986
Mat	0.478	0.732	0.426	0.478	0.710	0.437
RQ	0.059	0.038	0.970	0.059	0.033	0.970
SE+Q	0.025	0.008	0.995	0.025	0.013	0.996
Gibbs	0.324	0.433	0.723	0.324	0.419	0.728
MSE						
NN	0.092	0.062	0.970	0.092	0.056	0.970
SE	0.011	0.005	0.997	0.011	0.005	0.998
LIN	0.070	0.104	0.996	0.070	0.038	0.996
Q	0.175	0.157	0.958	0.175	0.128	0.959
Mat	0.569	0.771	0.420	0.569	0.795	0.460
RQ	0.133	0.100	0.950	0.133	0.080	0.951
SE+Q	0.056	0.039	0.979	0.056	0.032	0.978
Gibbs	0.257	0.305	0.825	0.257	0.274	0.821
R^2						
NN	0.248	0.227	0.900	0.248	0.221	0.900
SE	0.062	0.037	0.986	0.062	0.036	0.986
LIN	0.442	0.646	0.786	0.442	0.638	0.789
Q	0.118	0.072	0.963	0.118	0.080	0.965
Mat	0.391	0.632	0.582	0.391	0.627	0.582
RQ	0.019	0.009	0.995	0.019	0.010	0.995
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.191	0.168	0.926	0.191	0.168	0.926
MAE						
NN	0.110	0.055	0.968	0.110	0.070	0.971
SE	0.005	0.002	0.999	0.005	0.003	0.999
LIN	0.107	0.089	0.998	0.107	0.060	0.998
Q	0.024	0.014	0.996	0.024	0.013	0.996
Mat	0.485	0.790	0.386	0.485	0.790	0.393
RQ	0.106	0.067	0.961	0.106	0.064	0.962
SE+Q	0.111	0.078	0.962	0.111	0.075	0.962
Gibbs	0.274	0.314	0.811	0.274	0.310	0.810
Kendall						
NN	0.020	0.008	0.993	0.020	0.010	0.994
SE	0.025	0.015	0.997	0.025	0.013	0.997
LIN	0.603	0.947	0.140	0.603	0.940	0.140
Q	0.115	0.079	0.946	0.115	0.075	0.947
Mat	0.052	0.032	0.991	0.052	0.029	0.992
RQ	0.000	0.000	1.000	0.000	0.000	1.000
SE+Q	0.005	0.002	0.999	0.005	0.002	1.000
Gibbs	0.093	0.065	0.977	0.093	0.057	0.977

Table B.4: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

SVC one-vs-one						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.102	0.073	0.958	0.102	0.068	0.959
SE	0.014	0.008	0.994	0.014	0.007	0.994
LIN	0.040	0.017	0.997	0.040	0.021	0.998
Q	0.097	0.060	0.971	0.097	0.061	0.973
Mat	0.472	0.705	0.453	0.472	0.681	0.463
RQ	0.080	0.049	0.966	0.080	0.045	0.967
SE+Q	0.035	0.012	0.991	0.035	0.018	0.993
Gibbs	0.324	0.422	0.736	0.324	0.408	0.742
			MSE			
NN	0.113	0.077	0.958	0.113	0.073	0.959
SE	0.031	0.013	0.994	0.031	0.016	0.996
LIN	0.066	0.096	0.996	0.066	0.036	0.996
Q	0.177	0.166	0.953	0.177	0.134	0.953
Mat	0.565	0.731	0.462	0.565	0.757	0.504
RQ	0.141	0.104	0.950	0.141	0.085	0.953
SE+Q	0.076	0.057	0.967	0.076	0.047	0.966
Gibbs	0.269	0.323	0.819	0.269	0.293	0.816
			R^2			
NN	0.257	0.248	0.886	0.257	0.241	0.886
SE	0.086	0.055	0.979	0.086	0.052	0.979
LIN	0.458	0.602	0.828	0.458	0.595	0.831
Q	0.162	0.115	0.951	0.162	0.121	0.952
Mat	0.387	0.599	0.610	0.387	0.595	0.610
RQ	0.053	0.029	0.989	0.053	0.029	0.989
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.224	0.222	0.905	0.224	0.222	0.905
			MAE			
NN	0.139	0.073	0.957	0.139	0.096	0.961
SE	0.009	0.005	0.998	0.009	0.005	0.998
LIN	0.117	0.099	0.997	0.117	0.067	0.997
Q	0.061	0.037	0.988	0.061	0.036	0.988
Mat	0.483	0.755	0.425	0.483	0.755	0.432
RQ	0.119	0.076	0.957	0.119	0.073	0.959
SE+Q	0.142	0.106	0.956	0.142	0.100	0.956
Gibbs	0.286	0.329	0.812	0.286	0.324	0.811
			Kendall			
NN	0.013	0.006	0.995	0.013	0.007	0.995
SE	0.013	0.009	0.998	0.013	0.007	0.998
LIN	0.604	0.958	0.118	0.604	0.955	0.119
Q	0.092	0.061	0.962	0.092	0.056	0.962
Mat	0.055	0.033	0.992	0.055	0.030	0.993
RQ	0.004	0.003	0.999	0.004	0.002	0.999
SE+Q	0.002	0.001	0.999	0.002	0.001	0.999
Gibbs	0.105	0.074	0.977	0.105	0.065	0.977

Table B.5: Measured F$_1$ score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Neural network						
kernel	F$_1$ score	sensitivity	specificity	F$_1$ score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.093	0.053	0.973	0.093	0.057	0.974
SE	0.000	0.000	1.000	0.000	0.000	1.000
LIN	0.128	0.046	0.992	0.128	0.073	0.995
Q	0.155	0.092	0.971	0.155	0.099	0.974
Mat	0.470	0.659	0.519	0.470	0.632	0.530
RQ	0.023	0.016	0.991	0.023	0.012	0.990
SE+Q	0.001	0.000	1.000	0.001	0.000	1.000
Gibbs	0.360	0.589	0.637	0.360	0.558	0.642
MSE						
NN	0.142	0.127	0.947	0.142	0.098	0.946
SE	0.070	0.035	0.993	0.070	0.036	0.998
LIN	0.084	0.127	0.995	0.084	0.047	0.995
Q	0.214	0.211	0.944	0.214	0.174	0.945
Mat	0.560	0.667	0.526	0.560	0.698	0.571
RQ	0.091	0.064	0.973	0.091	0.051	0.974
SE+Q	0.017	0.013	0.996	0.017	0.009	0.996
Gibbs	0.327	0.480	0.743	0.327	0.439	0.740
R^2						
NN	0.275	0.270	0.880	0.275	0.267	0.880
SE	0.011	0.005	0.999	0.011	0.005	0.999
LIN	0.485	0.587	0.863	0.485	0.575	0.866
Q	0.213	0.185	0.927	0.213	0.185	0.928
Mat	0.394	0.672	0.547	0.394	0.667	0.546
RQ	0.039	0.021	0.995	0.039	0.021	0.995
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.210	0.170	0.945	0.210	0.170	0.945
MAE						
NN	0.202	0.152	0.921	0.202	0.174	0.925
SE	0.007	0.004	1.000	0.007	0.004	1.000
LIN	0.094	0.078	0.998	0.094	0.052	0.998
Q	0.093	0.064	0.983	0.093	0.058	0.984
Mat	0.482	0.699	0.498	0.482	0.697	0.504
RQ	0.142	0.104	0.957	0.142	0.089	0.957
SE+Q	0.083	0.051	0.977	0.083	0.051	0.977
Gibbs	0.335	0.431	0.779	0.335	0.424	0.778
Kendall						
NN	0.072	0.040	0.991	0.072	0.039	0.992
SE	0.014	0.007	0.999	0.014	0.007	0.999
LIN	0.603	0.952	0.137	0.603	0.943	0.135
Q	0.092	0.059	0.962	0.092	0.056	0.962
Mat	0.000	0.000	1.000	0.000	0.000	1.000
RQ	0.001	0.001	1.000	0.001	0.001	1.000
SE+Q	0.054	0.015	0.995	0.054	0.029	0.997
Gibbs	0.169	0.123	0.968	0.169	0.113	0.969

B.2 Regression

The following tables show measured metrics for defined regressor models later utilized to select a class with the minimal predicted value. For each classifier and each class sensitivity, specificity and F_1 score were computed.

Table B.6: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

			Regression tree			
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.045	0.023	0.986	0.045	0.025	0.987
SE	0.151	0.089	0.912	0.151	0.099	0.916
LIN	0.280	0.083	0.983	0.280	0.179	0.992
Q	0.062	0.030	0.990	0.062	0.034	0.992
Mat	0.325	0.342	0.686	0.325	0.331	0.685
RQ	0.374	0.507	0.557	0.374	0.468	0.554
SE+Q	0.158	0.109	0.933	0.158	0.110	0.938
Gibbs	0.058	0.037	0.980	0.058	0.032	0.980
			MSE			
NN	0.040	0.021	0.983	0.040	0.022	0.984
SE	0.183	0.096	0.920	0.183	0.121	0.931
LIN	0.002	0.002	0.999	0.002	0.001	0.999
Q	0.003	0.002	0.999	0.003	0.002	0.999
Mat	0.355	0.320	0.704	0.355	0.331	0.713
RQ	0.378	0.521	0.580	0.378	0.476	0.581
SE+Q	0.181	0.215	0.852	0.181	0.184	0.851
Gibbs	0.013	0.007	0.994	0.013	0.007	0.993
			R^2			
NN	0.123	0.085	0.946	0.123	0.086	0.946
SE	0.139	0.128	0.902	0.139	0.130	0.902
LIN	0.072	0.040	0.975	0.072	0.043	0.975
Q	0.085	0.055	0.962	0.085	0.057	0.963
Mat	0.262	0.276	0.782	0.262	0.275	0.782
RQ	0.182	0.398	0.604	0.182	0.393	0.604
SE+Q	0.087	0.082	0.916	0.087	0.088	0.917
Gibbs	0.122	0.095	0.941	0.122	0.096	0.941
			MAE			
NN	0.061	0.023	0.986	0.061	0.034	0.987
SE	0.197	0.129	0.903	0.197	0.157	0.911
LIN	0.020	0.004	0.999	0.020	0.010	0.999
Q	0.011	0.007	0.995	0.011	0.006	0.995
Mat	0.318	0.307	0.747	0.318	0.304	0.748
RQ	0.296	0.490	0.592	0.296	0.447	0.586
SE+Q	0.213	0.259	0.828	0.213	0.267	0.831
Gibbs	0.019	0.010	0.992	0.019	0.010	0.992
			Kendall			
NN	0.158	0.095	0.958	0.158	0.105	0.961
SE	0.187	0.187	0.908	0.187	0.177	0.912
LIN	0.602	0.701	0.563	0.602	0.685	0.567
Q	0.238	0.241	0.844	0.238	0.227	0.845
Mat	0.066	0.043	0.976	0.066	0.043	0.977
RQ	0.093	0.071	0.972	0.093	0.063	0.973
SE+Q	0.065	0.053	0.962	0.065	0.044	0.961
Gibbs	0.182	0.147	0.960	0.182	0.129	0.960

Table B.7: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

			Random forest			
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.007	0.004	0.998	0.007	0.004	0.998
SE	0.074	0.037	0.983	0.074	0.040	0.988
LIN	0.307	0.084	0.984	0.307	0.195	0.994
Q	0.004	0.001	1.000	0.004	0.002	1.000
Mat	0.378	0.439	0.601	0.378	0.438	0.606
RQ	0.367	0.545	0.493	0.367	0.496	0.480
SE+Q	0.074	0.045	0.976	0.074	0.042	0.977
Gibbs	0.026	0.018	0.991	0.026	0.014	0.990
			MSE			
NN	0.045	0.023	0.979	0.045	0.025	0.979
SE	0.223	0.083	0.946	0.223	0.137	0.969
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.399	0.412	0.638	0.399	0.414	0.646
RQ	0.345	0.510	0.576	0.345	0.436	0.563
SE+Q	0.179	0.165	0.898	0.179	0.155	0.900
Gibbs	0.000	0.000	1.000	0.000	0.000	1.000
			R^2			
NN	0.067	0.039	0.976	0.067	0.039	0.976
SE	0.118	0.085	0.953	0.118	0.085	0.954
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.319	0.450	0.643	0.319	0.446	0.642
RQ	0.182	0.497	0.471	0.182	0.492	0.470
SE+Q	0.042	0.020	0.984	0.042	0.025	0.985
Gibbs	0.005	0.003	0.995	0.005	0.003	0.995
			MAE			
NN	0.034	0.017	0.985	0.034	0.019	0.985
SE	0.179	0.088	0.943	0.179	0.121	0.952
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.350	0.383	0.684	0.350	0.374	0.683
RQ	0.289	0.528	0.542	0.289	0.470	0.532
SE+Q	0.218	0.186	0.886	0.218	0.218	0.892
Gibbs	0.000	0.000	1.000	0.000	0.000	1.000
			Kendall			
NN	0.127	0.061	0.976	0.127	0.076	0.980
SE	0.219	0.233	0.901	0.219	0.218	0.906
LIN	0.615	0.749	0.521	0.615	0.733	0.524
Q	0.240	0.229	0.868	0.240	0.214	0.869
Mat	0.071	0.035	0.981	0.071	0.044	0.983
RQ	0.070	0.060	0.977	0.070	0.045	0.977
SE+Q	0.070	0.052	0.957	0.070	0.049	0.957
Gibbs	0.181	0.142	0.966	0.181	0.124	0.966

Table B.8: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Support vector regression						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.077	0.046	0.966	0.077	0.049	0.967
SE	0.187	0.113	0.913	0.187	0.123	0.922
LIN	0.002	0.002	0.999	0.002	0.001	0.999
Q	0.030	0.014	0.992	0.030	0.016	0.992
Mat	0.347	0.338	0.719	0.347	0.338	0.726
RQ	0.348	0.432	0.642	0.348	0.388	0.638
SE+Q	0.240	0.224	0.846	0.240	0.227	0.854
Gibbs	0.083	0.054	0.961	0.083	0.050	0.961
			MSE			
NN	0.151	0.115	0.910	0.151	0.120	0.912
SE	0.206	0.089	0.941	0.206	0.129	0.959
LIN	0.128	0.037	0.941	0.128	0.113	0.946
Q	0.106	0.080	0.944	0.106	0.081	0.945
Mat	0.254	0.152	0.893	0.254	0.170	0.910
RQ	0.197	0.188	0.793	0.197	0.169	0.782
SE+Q	0.127	0.066	0.945	0.127	0.087	0.950
Gibbs	0.301	0.457	0.670	0.301	0.457	0.673
			R^2			
NN	0.115	0.081	0.938	0.115	0.082	0.939
SE	0.163	0.172	0.884	0.163	0.167	0.884
LIN	0.134	0.105	0.923	0.134	0.105	0.923
Q	0.114	0.090	0.941	0.114	0.088	0.941
Mat	0.162	0.129	0.878	0.162	0.128	0.878
RQ	0.202	0.315	0.767	0.202	0.307	0.767
SE+Q	0.026	0.016	0.981	0.026	0.016	0.981
Gibbs	0.222	0.398	0.727	0.222	0.398	0.728
			MAE			
NN	0.070	0.052	0.954	0.070	0.048	0.954
SE	0.152	0.086	0.932	0.152	0.108	0.938
LIN	0.062	0.122	0.930	0.062	0.094	0.930
Q	0.132	0.104	0.941	0.132	0.116	0.943
Mat	0.224	0.189	0.815	0.224	0.183	0.813
RQ	0.263	0.255	0.760	0.263	0.285	0.769
SE+Q	0.081	0.048	0.967	0.081	0.052	0.968
Gibbs	0.278	0.378	0.730	0.278	0.380	0.731
			Kendall			
NN	0.045	0.024	0.984	0.045	0.025	0.985
SE	0.147	0.125	0.937	0.147	0.119	0.939
LIN	0.607	0.796	0.415	0.607	0.783	0.417
Q	0.171	0.141	0.902	0.171	0.133	0.902
Mat	0.090	0.064	0.973	0.090	0.060	0.974
RQ	0.163	0.150	0.941	0.163	0.141	0.944
SE+Q	0.056	0.036	0.978	0.056	0.034	0.979
Gibbs	0.098	0.069	0.974	0.098	0.061	0.974

Table B.9: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Neural network						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.000	0.000	1.000	0.000	0.000	1.000
SE	0.039	0.017	0.990	0.039	0.020	0.993
LIN	0.004	0.002	1.000	0.004	0.002	1.000
Q	0.034	0.003	0.997	0.034	0.017	0.999
Mat	0.412	0.418	0.703	0.412	0.422	0.720
RQ	0.346	0.583	0.338	0.346	0.547	0.306
SE+Q	0.049	0.020	0.988	0.049	0.026	0.990
Gibbs	0.000	0.000	1.000	0.000	0.000	1.000
MSE						
NN	0.000	0.000	1.000	0.000	0.000	1.000
SE	0.000	0.000	1.000	0.000	0.000	1.000
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.519	0.999	0.000	0.519	1.000	0.000
RQ	0.000	0.000	1.000	0.000	0.000	1.000
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.000	0.000	1.000	0.000	0.000	1.000
R^2						
NN	0.287	0.575	0.590	0.287	0.556	0.588
SE	0.179	0.248	0.788	0.179	0.255	0.790
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.220	0.294	0.639	0.220	0.293	0.638
RQ	0.000	0.000	1.000	0.000	0.000	1.000
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.009	0.004	0.997	0.009	0.004	0.997
MAE						
NN	0.177	0.439	0.561	0.177	0.413	0.557
SE	0.231	0.474	0.428	0.231	0.498	0.427
LIN	0.000	0.000	1.000	0.000	0.000	1.000
Q	0.000	0.000	1.000	0.000	0.000	1.000
Mat	0.000	0.000	1.000	0.000	0.000	1.000
RQ	0.000	0.000	1.000	0.000	0.000	1.000
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.000	0.000	1.000	0.000	0.000	1.000
Kendall						
NN	0.000	0.000	1.000	0.000	0.000	1.000
SE	0.140	0.099	0.940	0.140	0.110	0.943
LIN	0.611	0.861	0.310	0.611	0.853	0.314
Q	0.107	0.072	0.956	0.107	0.067	0.956
Mat	0.083	0.063	0.987	0.083	0.049	0.987
RQ	0.182	0.174	0.924	0.182	0.175	0.929
SE+Q	0.045	0.032	0.979	0.045	0.027	0.979
Gibbs	0.057	0.039	0.987	0.057	0.032	0.987

B.3 Multi-label classification

The following tables show measured metrics for defined multi-label classifiers. How multi-label classifiers were utilized see description Section 2.5.3. For each classifier and each class sensitivity, specificity and F_1 score were computed.

Table B.10: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Decision tree						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.102	0.074	0.960	0.102	0.068	0.960
SE	0.129	0.075	0.946	0.129	0.077	0.950
LIN	0.246	0.086	0.979	0.246	0.163	0.987
Q	0.069	0.045	0.975	0.069	0.042	0.975
Mat	0.371	0.412	0.683	0.371	0.388	0.685
RQ	0.277	0.206	0.855	0.277	0.210	0.869
SE+Q	0.068	0.038	0.972	0.068	0.040	0.973
Gibbs	0.315	0.479	0.698	0.315	0.431	0.697
			MSE			
NN	0.243	0.143	0.908	0.243	0.197	0.921
SE	0.388	0.238	0.855	0.388	0.316	0.896
LIN	0.070	0.107	0.992	0.070	0.040	0.991
Q	0.190	0.191	0.941	0.190	0.156	0.941
Mat	0.370	0.386	0.704	0.370	0.353	0.699
RQ	0.129	0.097	0.939	0.129	0.080	0.939
SE+Q	0.123	0.094	0.950	0.123	0.084	0.951
Gibbs	0.292	0.370	0.801	0.292	0.338	0.798
			R^2			
NN	0.225	0.213	0.879	0.225	0.212	0.879
SE	0.156	0.137	0.925	0.156	0.133	0.925
LIN	0.388	0.421	0.881	0.388	0.409	0.882
Q	0.179	0.158	0.921	0.179	0.158	0.922
Mat	0.332	0.417	0.716	0.332	0.414	0.715
RQ	0.127	0.101	0.927	0.127	0.104	0.928
SE+Q	0.071	0.055	0.958	0.071	0.055	0.958
Gibbs	0.192	0.186	0.906	0.192	0.186	0.906
			MAE			
NN	0.168	0.135	0.937	0.168	0.134	0.938
SE	0.177	0.119	0.933	0.177	0.127	0.937
LIN	0.138	0.124	0.993	0.138	0.089	0.993
Q	0.091	0.066	0.975	0.091	0.061	0.975
Mat	0.405	0.477	0.653	0.405	0.467	0.653
RQ	0.246	0.191	0.890	0.246	0.195	0.897
SE+Q	0.183	0.141	0.934	0.183	0.147	0.936
Gibbs	0.312	0.385	0.788	0.312	0.380	0.788
			Kendall			
NN	0.148	0.092	0.954	0.148	0.100	0.957
SE	0.139	0.094	0.959	0.139	0.098	0.962
LIN	0.574	0.756	0.383	0.574	0.744	0.382
Q	0.129	0.097	0.930	0.129	0.090	0.929
Mat	0.088	0.069	0.968	0.088	0.062	0.968
RQ	0.072	0.049	0.978	0.072	0.045	0.979
SE+Q	0.102	0.076	0.960	0.102	0.071	0.960
Gibbs	0.157	0.129	0.950	0.157	0.115	0.951

Table B.11: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

Random forest						
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
RDE						
NN	0.070	0.045	0.980	0.070	0.041	0.980
SE	0.062	0.029	0.987	0.062	0.032	0.992
LIN	0.288	0.078	0.986	0.288	0.179	0.995
Q	0.034	0.018	0.995	0.034	0.018	0.996
Mat	0.444	0.635	0.520	0.444	0.592	0.520
RQ	0.132	0.076	0.960	0.132	0.076	0.967
SE+Q	0.024	0.007	0.997	0.024	0.012	0.998
Gibbs	0.341	0.560	0.656	0.341	0.510	0.657
MSE						
NN	0.169	0.086	0.964	0.169	0.107	0.971
SE	0.445	0.218	0.896	0.445	0.328	0.952
LIN	0.091	0.140	0.995	0.091	0.051	0.995
Q	0.195	0.182	0.956	0.195	0.147	0.956
Mat	0.448	0.588	0.547	0.448	0.536	0.537
RQ	0.089	0.068	0.974	0.089	0.050	0.974
SE+Q	0.110	0.062	0.973	0.110	0.066	0.976
Gibbs	0.293	0.376	0.801	0.293	0.339	0.798
R^2						
NN	0.241	0.206	0.911	0.241	0.205	0.912
SE	0.076	0.044	0.992	0.076	0.042	0.992
LIN	0.469	0.643	0.815	0.469	0.636	0.818
Q	0.139	0.093	0.966	0.139	0.094	0.966
Mat	0.406	0.699	0.545	0.406	0.693	0.544
RQ	0.055	0.029	0.992	0.055	0.030	0.993
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.221	0.189	0.935	0.221	0.189	0.935
MAE						
NN	0.166	0.095	0.969	0.166	0.109	0.972
SE	0.069	0.030	0.993	0.069	0.037	0.996
LIN	0.185	0.163	0.997	0.185	0.110	0.997
Q	0.041	0.024	0.997	0.041	0.021	0.997
Mat	0.486	0.768	0.449	0.486	0.748	0.448
RQ	0.164	0.099	0.969	0.164	0.098	0.974
SE+Q	0.178	0.118	0.961	0.178	0.124	0.964
Gibbs	0.324	0.406	0.787	0.324	0.400	0.786
Kendall						
NN	0.097	0.043	0.986	0.097	0.054	0.988
SE	0.005	0.002	1.000	0.005	0.002	1.000
LIN	0.602	0.977	0.098	0.602	0.966	0.094
Q	0.030	0.016	0.990	0.030	0.016	0.991
Mat	0.002	0.001	1.000	0.002	0.001	1.000
RQ	0.006	0.003	1.000	0.006	0.003	1.000
SE+Q	0.054	0.020	0.996	0.054	0.029	0.997
Gibbs	0.149	0.110	0.975	0.149	0.095	0.975

Table B.12: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

			SVC			
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.015	0.008	0.994	0.015	0.008	0.994
SE	0.196	0.149	0.838	0.196	0.153	0.837
LIN	0.022	0.006	0.998	0.022	0.011	0.998
Q	0.034	0.017	0.988	0.034	0.019	0.988
Mat	0.347	0.368	0.660	0.347	0.368	0.664
RQ	0.332	0.297	0.781	0.332	0.294	0.795
SE+Q	0.104	0.042	0.961	0.104	0.064	0.966
Gibbs	0.241	0.271	0.809	0.241	0.250	0.809
			MSE			
NN	0.159	0.117	0.929	0.159	0.118	0.932
SE	0.375	0.256	0.848	0.375	0.311	0.884
LIN	0.064	0.076	0.993	0.064	0.036	0.993
Q	0.140	0.116	0.966	0.140	0.096	0.966
Mat	0.412	0.497	0.583	0.412	0.463	0.576
RQ	0.193	0.125	0.928	0.193	0.124	0.937
SE+Q	0.088	0.065	0.964	0.088	0.055	0.964
Gibbs	0.233	0.243	0.863	0.233	0.220	0.861
			R^2			
NN	0.212	0.172	0.926	0.212	0.168	0.926
SE	0.152	0.165	0.871	0.152	0.163	0.871
LIN	0.437	0.518	0.864	0.437	0.502	0.866
Q	0.109	0.072	0.973	0.109	0.069	0.973
Mat	0.367	0.574	0.595	0.367	0.570	0.595
RQ	0.043	0.023	0.992	0.043	0.023	0.992
SE+Q	0.009	0.005	0.997	0.009	0.004	0.997
Gibbs	0.219	0.215	0.907	0.219	0.215	0.907
			MAE			
NN	0.085	0.054	0.981	0.085	0.051	0.981
SE	0.154	0.082	0.937	0.154	0.107	0.943
LIN	0.138	0.117	0.998	0.138	0.079	0.998
Q	0.026	0.014	0.997	0.026	0.014	0.997
Mat	0.447	0.648	0.497	0.447	0.637	0.498
RQ	0.210	0.140	0.911	0.210	0.153	0.918
SE+Q	0.158	0.127	0.944	0.158	0.121	0.944
Gibbs	0.283	0.317	0.821	0.283	0.312	0.821
			Kendall			
NN	0.040	0.022	0.990	0.040	0.022	0.991
SE	0.005	0.003	1.000	0.005	0.002	1.000
LIN	0.602	0.977	0.081	0.602	0.973	0.081
Q	0.030	0.016	0.990	0.030	0.016	0.990
Mat	0.002	0.002	1.000	0.002	0.001	1.000
RQ	0.002	0.002	1.000	0.002	0.001	1.000
SE+Q	0.008	0.004	0.999	0.008	0.004	1.000
Gibbs	0.149	0.106	0.974	0.149	0.095	0.975

Table B.13: Measured F_1 score, sensitivity and specificity with two defined variants. The first variant measures exact matches with the dataset and the second measures when the classified kernel was in the group of similarly performing kernels. Table is separated into subtables where each subtable shows measured metrics w.r.t. used error measure.

			Neural network			
kernel	F_1 score	sensitivity	specificity	F_1 score$_2$	sensitivity$_2$	specificity$_2$
			RDE			
NN	0.070	0.045	0.980	0.070	0.041	0.980
SE	0.066	0.035	0.987	0.066	0.035	0.991
LIN	0.269	0.076	0.986	0.269	0.166	0.995
Q	0.045	0.020	0.995	0.045	0.024	0.996
Mat	0.445	0.640	0.522	0.445	0.592	0.522
RQ	0.133	0.078	0.960	0.133	0.077	0.968
SE+Q	0.015	0.006	0.997	0.015	0.007	0.998
Gibbs	0.343	0.559	0.656	0.343	0.513	0.657
			MSE			
NN	0.187	0.086	0.960	0.187	0.120	0.969
SE	0.402	0.179	0.909	0.402	0.283	0.959
LIN	0.120	0.182	0.995	0.120	0.068	0.995
Q	0.197	0.182	0.956	0.197	0.149	0.956
Mat	0.457	0.599	0.540	0.457	0.552	0.534
RQ	0.102	0.075	0.970	0.102	0.057	0.972
SE+Q	0.104	0.060	0.974	0.104	0.062	0.976
Gibbs	0.293	0.376	0.801	0.293	0.339	0.798
			R^2			
NN	0.241	0.206	0.911	0.241	0.205	0.912
SE	0.076	0.045	0.992	0.076	0.042	0.992
LIN	0.469	0.643	0.815	0.469	0.636	0.818
Q	0.139	0.093	0.966	0.139	0.094	0.966
Mat	0.406	0.699	0.545	0.406	0.693	0.544
RQ	0.055	0.029	0.992	0.055	0.030	0.993
SE+Q	0.000	0.000	1.000	0.000	0.000	1.000
Gibbs	0.220	0.189	0.935	0.220	0.189	0.935
			MAE			
NN	0.170	0.098	0.969	0.170	0.112	0.972
SE	0.087	0.032	0.988	0.087	0.047	0.993
LIN	0.185	0.163	0.997	0.185	0.110	0.997
Q	0.041	0.024	0.997	0.041	0.021	0.997
Mat	0.482	0.758	0.452	0.482	0.738	0.451
RQ	0.158	0.097	0.968	0.158	0.095	0.972
SE+Q	0.177	0.118	0.962	0.177	0.123	0.964
Gibbs	0.324	0.406	0.787	0.324	0.400	0.786
			Kendall			
NN	0.078	0.039	0.988	0.078	0.043	0.989
SE	0.009	0.002	0.999	0.009	0.005	1.000
LIN	0.602	0.975	0.098	0.602	0.965	0.094
Q	0.030	0.016	0.990	0.030	0.016	0.991
Mat	0.003	0.001	1.000	0.003	0.002	1.000
RQ	0.007	0.002	0.999	0.007	0.004	0.999
SE+Q	0.072	0.023	0.995	0.072	0.038	0.997
Gibbs	0.154	0.111	0.974	0.154	0.099	0.975

Appendix C

Contents of Flash Drive

- `readme.txt` .... the file with Flash drive contents description
- `src` .... the directory of source codes
  - `experiments` .... the directory with experiments
    - `*_classifier.ipynb` .... the Jupyter Notebook source code files for training classifiers
    - `Accuracy_report.ipynb` ... the Jupyter Notebook source code file for assessing models performance
    - `*.py` . the Python source code files with a commonly used functions
    - `models` .... the directory of trained models in Pickle format
  - `tex` .... the directory of LaTeX source codes
    - `figures` .... directory of Jupyter Notebook for generating figures
    - `*.tex` .... the LaTeX source code files of the thesis
- `text` .... the thesis text directory
  - `thesis.pdf` .... the Diploma thesis in PDF format