BATCH AND ONLINE IMPLICIT WEIGHTED GAUSSIAN PROCESSES FOR ROBUST NOVELTY DETECTION

by

RUBEN RAMIREZ PADRON

M.S. University of Central Florida, 2009
B.S. Universidad Central "Marta Abreu" de Las Villas, 1996

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Engineering
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term

2015

Major Professor: Avelino Gonzalez

ABSTRACT

This dissertation aims mainly at obtaining robust variants of Gaussian processes (GPs) that do not require using non-Gaussian likelihoods to compensate for outliers in the training data. Bayesian kernel methods, and in particular GPs, have been used to solve a variety of machine learning problems, equating or exceeding the performance of other successful techniques. That is the case of a recently proposed approach to GP-based novelty detection that uses standard GPs (i.e. GPs employing Gaussian likelihoods). However, standard GPs are sensitive to outliers in training data, and this limitation carries over to GP-based novelty detection. This limitation has been typically addressed by using robust non-Gaussian likelihoods. However, non-Gaussian likelihoods lead to analytically intractable inferences, which require using approximation techniques that are typically complex and computationally expensive. Inspired by the use of weights in quasi-robust statistics, this work introduces a particular type of weight functions, called here *data weighers*, in order to obtain robust GPs that do not require approximation techniques and retain the simplicity of standard GPs. This work proposes implicit weighted variants of batch GP, online GP, and sparse online GP (SOGP) that employ weighted Gaussian likelihoods. Mathematical expressions for calculating the posterior implicit weighted GPs are derived in this work. In our experiments, novelty detection based on our weighted batch GPs consistently and significantly outperformed standard batch GP-based novelty detection whenever data was contaminated with outliers. Additionally, our experiments show that novelty detection based on online GPs can perform similarly to batch GP-based novelty detection. Membership scores previously introduced by other authors are also compared in our experiments.

To my wife Roxana and my daughter Camila,

for their constant love

and for supporting me every day with a high probability.

To my family in Cuba, for inculcating in me

the values of Honesty and Perseverance

since I was as young as Camila is now.

ACKNOWLEDGMENTS

I met my advisor, Dr. Avelino Gonzalez, ten years ago in Guatemala City. I worked at a university in Guatemala City at that time, and I was asked to publicize and support Dr. Gonzalez's presentations about graduate studies at UCF. I applied to UCF motivated by his presentations, and I got accepted as a PhD student in 2007. I am sincerely grateful to Dr. Gonzalez for his continuous interest in fostering my personal and professional development during all these years at UCF. His many comments and suggestions on the multiple phases of the research presented here have been crucial for the successful completion of this dissertation.

When I started my graduate studies at UCF, I initially joined the Machine Learning Lab under the guidance of Dr. Michael Georgiopoulos and Dr. Georgios Anagnostopoulos. I am very thankful to both for the good times that we shared working together and for the many things about machine learning that I learned from them. I am particularly grateful to Dr. Georgiopoulos. The advice and support provided by him played a significant role during the time I was applying to UCF and during my first two years at UCF as my advisor. His support has continued as a member of my Dissertation Committee. I am very thankful to the other members of my current Dissertation Committee as well, for their support, helpful comments and suggestions: Dr. Boris Mederos, Dr. Kenneth Stanley, and Dr. Morgan C. Wang. This recognition extends to Dr. Jose Sepulveda, which was initially a member of my Committee but retired recently. Special thanks go out to Dr. Mederos for the multiple meetings on Skype to clarify my questions when I was learning the theory behind Gaussian processes and robust statistics. Additionally, I gratefully

acknowledge the support of UCF's I2Lab Fellowship and the College of Engineering and Computer Science during the first three years of my graduate studies.

I have worked in this dissertation in parallel with working full time as a software engineer during the last four years and being a new father for almost three years. Because of that, this dissertation would not be possible, despite the good will and plentiful support from the professors mentioned above, without the support of my wife, my sister in law, and my parents in law when they visited us. Thanks to you all for providing the essential time that allowed me to reach the point of writing these acknowledgments!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION AND BACKGROUND

This dissertation investigates the development of robust novelty detection algorithms that use weighted variants of Gaussian processes (GPs) as their theoretical foundation, which are also proposed in this work. Note that the term "novelty detection" covers the same types of problems and algorithms considered by the area of "outlier detection". Both terms differ mainly in the interpretation given to anomalous observations. Note also that the term *anomaly detection* is commonly used as a synonymous to outlier detection. Consequently, the three terms are used indistinctly in this work. However, the term "novelty detection" is favored in this dissertation for reasons exposed in the following subsection.

The objective of Chapter 1 is to introduce the various terms used in this dissertation. The research efforts made by others in the field of novelty detection are also discussed. We begin by providing a historical perspective on outlier detection (arguably the first term used to denote this research area). Afterwards, a review of modern outlier detection methods is offered. A particular emphasis is given to kernel methods, given that GPs rely on kernel functions to define covariance matrices. The advantages and limitations of modern techniques are also mentioned in this chapter.

1.1 Background

Prices for data storage have been falling at a rapid pace recently. This has enabled the recording and management of a variety of every-day activities, thereby resulting in storing increasingly

larger amounts of data for various purposes. Consequently, retrieving knowledge from data sets has become a very important practical problem, typically covered by statistics and –more recently– data mining. The primary goal of *data mining* is to find useful hidden patterns in large data sets. However, problems in data mining have become much more difficult recently, not only because of the larger size of data sets, but also because the increasing variety and complexity of the data.

From a very general point of view, the search for knowledge in large amounts of data can be done by employing two very different approaches. The first approach assumes that the initial data set contains all the information required to find the type of patterns we are interested in. Consequently, one or more learning algorithms are applied once to the available data, and any patterns obtained are considered valid for a relatively long time. This scenario is known as *batch learning*. The second scenario assumes that observations come on a serial fashion and possibly generated by a slowly changing distribution. These assumptions imply that knowledge needs to be constantly updated based on new input data. Algorithms designed to work under this setting are said to follow an *online learning* approach. The research described in this dissertation addresses both batch and online outlier detection.

Currently, many databases contain a mixture of data types: numerical and categorical variables, graphs, maps, images, video and sound, among others. Data mining researchers and practitioners employ several techniques, mainly from statistics and machine learning, in order to find interesting and actionable patterns within those large and potentially complex data sets (Tan, Steinbach, & Kumar, 2005). The wide applicability of data mining and the increasing complexity

of the problems it tackles have led to the creation of standards for describing data mining models. An example worthy of mention here is the Predictive Model Markup Language (PMML) (Guazzelli, Zeller, Lin, & Williams, 2009). PMML allows several mainstream data analysis software to exchange data mining models, e.g., IBM DB2 Data Warehouse, Microsoft SQL Server Analysis Services, Rattle/R, Statistica, SPSS, SAS Enterprise Miner, KNIME, and RapidMiner.

The problems traditionally considered in data mining are *clustering*, *classification*, *dimensionality reduction*, *association mining*, and *outlier detection*. Outlier detection is a growing field within data mining. It focuses on detecting unusual observations in data sets and processes. Hawkins defined an outlier as "*an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*" (Hawkins D. M., 1980). This general definition is broadly accepted nowadays. Detection of credit card fraud, computer network attacks, anomalous clinical results, suspicious activity in electronic commerce and faulty sensor readings are some of the most important applications of outlier detection.

The term *novelty detection* is another synonymous to outlier detection. The difference between the two terms is based on the interpretation given to the suspiciously abnormal data points. An *outlier* (or *anomaly*) is an observation that does not belong to the population or process being modeled, while novelty detection refers to "*the identification of new or unknown data or signals that a machine learning system is not aware of during training*" (Markou & Singh, 2003). Nevertheless, the terms *outlier detection*, *anomaly detection* and *novelty detection* denote the

same set of techniques and encompass the same theoretical and practical concerns. Because of that, they will be used indistinctly in this dissertation. However, novelty detection is the term used in the title because it better reflects the learning approach investigated in this research. While outliers and anomalies are observations that should not belong to the concept previously learned, novelty detection provides a more general interpretation because it opens the door to the realization that some of the abnormal observations actually belong to under-represented or emerging areas that could increasingly be considered normal. In those cases, the model should be adjusted accordingly over time, in order to include the novel observations as part of the normal class; even if initially they are not given much importance in the model. This idea is explored in this dissertation through the use of weights that embody the importance of observations. Weights would allow learning new regions of the normal class in a way that is robust to the presence of actual outliers in the training data.

1.2 Historical Origins of Outlier Detection

Outlier detection has been done for centuries. For instance, it was common practice among astronomers on the eighteenth century to reject observations with particularly large deviations from the sample mean. However, detection of "doubtful observations" was not based on any mathematical foundation at that time. Scientists dealing with series of observations used their experience and intuition to decide, arbitrarily, whether to keep or reject those observations that seemed to be erroneous measures, or possibly coming from another source. This practice, as might be expected, was highly controversial.

One of the first references to outlier detection was based on large residuals. It came from the head of the German School of Astronomers in 1838 (Anscombe & Guttman, 1960). This approach assumes that outliers are observations lying in low-probability regions of a stochastic model that describes a normal class. The underlying distribution of each normal class is estimated from the training data set. Outliers are determined by calculating the probabilities of obtaining a new observation from each normal class using the corresponding estimated class distributions.

In 1852, Benjamin Peirce, the father of Charles Sanders Peirce, published the first effort to establish a formal test for outlier detection (Peirce, 1852). Given a series of N observations, Peirce proposed to obtain a threshold T for the errors (which he called "*limit of error*") such that observations with residuals from the mean greater than T would be considered outliers. In Peirce's words, the principle behind his criterion is that "*the proposed observations should be rejected when the probability of the system of errors obtained by retaining them is less than that of the system of errors obtained by their rejection multiplied by the probability of making so many, and no more, abnormal observations*". When his work was published, one of the main difficulties practitioners were facing was to decide whether "doubtful observations" were actually outliers. The small amounts of data managed at that time allowed practitioners to determine those doubtful observations by visual inspection. Consequently, Peirce assumed that the number of observations proposed to be rejected, n, was known in advance. Peirce's paper is not straightforward to read because the notation is very different from modern notation. A paper published in 1855 by Benjamin Apthorp Gould, the founder of the American Astronomical Journal, provides a description of Peirce's criterion that is somewhat easier to understand despite

the old-fashioned notation (Gould, 1855). A brief description of Peirce's criterion is given below using modern notation.

Let $X = \{x_1, x_2, \dots, x_N\}$ be a series of *N* observations, and *n* the number of doubtful observations that we are intending to reject. Peirce denoted by *m* the number of unknown variables contained in the observations, which is a quantity that the practitioner must fix in advance. Let us denote by σ_1 the standard deviation of the original sample. The standard deviation of the remaining observations after removing the *n* doubtful observations is denoted by σ_2. Let us define $\lambda \equiv \frac{\sigma_1}{\sigma_2}$, and assume that the threshold $T = c\sigma_1$. The main goal of Peirce's criterion is to obtain a value for *c* such that any observation x_i with $|x_i - \mu_X| > T$ has a high probability of being an outlier.

To decide on rejecting *n* doubtful observations following Peirce's criterion, the following inequality must be satisfied:

$$\lambda^{N-n} e^{\frac{1}{2}n(c^2-1)} \left(2\Phi(-c)\right)^n < Q^N \,, \qquad (1.1)$$

where $Q^N = \frac{n^n (N-n)^{N-n}}{N^N}$ and Φ denotes the cumulative distribution function of the standard normal distribution. Clearly, $2\Phi(-c)$ denotes the probability of having residuals greater than *c* in absolute value, provided the residuals follow a standard normal distribution.

Peirce assumed that "*the excess of the sum of squares of the residual errors above the corresponding sum in the series remaining after the n observations have been excluded is only equal to the sum of the squares of the rejected residuals*". Under that assumption, which seems general enough, the following equations are obtained:

$$\lambda^2 = \frac{N-m-nc^2}{N-m-n}\,, \qquad (1.2)$$

$$c^2 = 1 + \frac{N-m-n}{n}(1-\lambda^2)\,. \qquad (1.3)$$

Given Peirce's assumption on the sum of squared residuals, inequality (1.1) becomes

$$\lambda^{N-n}R^n = Q^N\,, \qquad (1.4)$$

where $R = e^{\frac{1}{2}(c^2-1)}2\Phi(-c)$.

The application of the criterion consists of the following steps using the last three equations: 1) an approximate value for *R* is assumed; 2) the corresponding value for λ is estimated using *R* and *Q*; 3) an estimate for *c* is obtained using the estimate for λ. The process could be repeated iteratively to increase precision. After one or more iterations to estimate *c*, the threshold *T* is calculated as $T = c\sigma_1$. To apply Peirce's criterion, the threshold must first be determined for the hypothesis of $n = 1$. If the test supported rejecting one observation, the hypothesis of $n = 2$ is tested, and so on.

Peirce's criterion was highly controversial. Several scientists had harsh criticism, particularly Sir George Biddell Airy, the director of the Royal Greenwich Observatory. Airy wrote "*the whole theory is defective in its foundations, and illusory in its results*" (Airy, 1856). Despite the critics, Peirce's criterion was in use very soon after its publication. Among the first applications were analysis of astronomical data and the rejection of doubtful observations from the United States Coast Survey. The astronomer Joseph Winlock strongly criticized Airy's statements (Winlock, 1856). He stated that some of Airy's arguments, like the inapplicability of probability laws to observations that were already recorded, were not sound from a statistical point of view. Airy had asserted that it was as probable for the retention of doubtful observation to be beneficial as to

be harmful for data analysis. Regarding this assertion, Winlock's argument was that rejecting a valid observation does not necessarily introduce an error, while keeping an abnormal observation as valid definitely affects any statistics obtained from the data. In Winlock's words: "*we must reject whenever an observation is so doubtful, that retaining it makes our conclusions less reliable than they would be if its evidence had not been used*" (Winlock, 1856). Interestingly enough, this sentence reflects Peirce's own approach to the outlier rejection problem.

The second important work on outlier detection from the nineteen century was the approximation method proposed by Chauvenet (Chauvenet, 1868). That work was the appendix to Chauvenet's book "Manual of spherical and practical astronomy". The last section of the treatise, starting on page 558, offers a description of Peirce's criterion. After that review, Chauvenet proposed an approximate criterion for rejecting a single doubtful observation, based on the foundation of least squares. He mentioned that such approximations were also possible for the general case, but they were more cumbersome than Peirce's criterion, so he preferred not to develop it further.

Assuming that the residuals of the observations distributes $\mathrm{N}(0,\sigma^2)$, the actual number of residuals *n* to be expected greater than a threshold *T* in absolute value, where $T = c\sigma$, is given by $n = 2N\Phi(-\mathrm{c})$. Again, Φ denotes the cumulative distribution function of the standard normal distribution. The main idea behind Chauvenet's criterion is to find *T* such that, regardless of the number of observations *N*, on average, half an observation of valid data is rejected. Accordingly, if *c* satisfied $2N\Phi(-\mathrm{c}) = 0.5$ then any residual greater than $T = \mathrm{c}\sigma$ in absolute value should be rejected.

An important aspect to note from Chauvenet's criterion is that the threshold *c* decreases when the number of observations decreases, causing a variable proportion of observations to be rejected. The criterion is devised to test for a single outlier because *N* changes once an outlier is rejected, and consequently, the threshold must be updated. Chauvenet suggested a successive application of his criterion for the rejection of two or more outliers. It is also important to note that Chauvenet was supportive of Peirce's criterion, and he recommended it for those situations where his approximation was not applicable.

The third important work from the nineteenth century, in chronological order, is the outlier rejection method from Stone (Stone, 1868). His article offered an alternative to Peirce's criterion and Chauvenet's outlier rejection method. The main idea was to determine a threshold *T* for the residuals based on the proportion of outliers within a data set, which is assumed to be known in advance. Stone defined the term "*modulus of carelessness*" as the average number of observations containing exactly one outlier for the sample at hand. He denoted by *n* the modulus of carelessness. If the value *c* satisfying the following equation is found:

$$2\Phi(-c) = \frac{1}{n}, \qquad (1.5)$$

we would have a threshold $T = c\sigma$ that fits the expected proportion of outliers in the sample. Contrary to Chauvenet's method, by using this rule the value of *c* does not depend on the number of observations. Therefore, the number of outliers increases for larger number of observations. This approach seems to be the seed for the current rule of thumb that rules out normal observations which are farther than three standard deviations from the mean. It is trivial to realize that such rule of thumb actually corresponds to a modulus of carelessness approximately

equal to 370; i.e. a probability of encountering an outlier is assumed to be around 0.0027. Apparently, this was not the first rule of thumb to be used. The work of (Wright, 1884) proposed to reject observations with residuals above 3.37 times the standard deviation in absolute value (using old notation: five times the probable error).

The methods from the nineteenth century were always concerned with the probabilities of observations lying far enough from the sample mean assuming a normal distribution. A method described in (Irwin, 1925a) introduced the idea of taking into account the difference between neighboring observations. If observations taken at random from a normal population were arranged in descending order of magnitude, the frequency distribution of the differences between the p^{th} and $(p + 1)^{th}$ observations can be obtained (Irwin, 1925b). In particular, Irwin noted that for $p = 1$ and $p = 2$, those frequency distributions could be approximated by functions from the following family (using Irwin's notation):

$$y = y_0 e^{-\frac{1}{2}\left\{\frac{(x+h)^2 - h^2}{\Sigma^2}\right\}}, \qquad (1.6)$$

where $x \in [0, \infty)$, and the parameters y_0, h and Σ depend on the size of the sample at hand. Let us denote by σ the standard deviation of the sample population. If the probability of any of these two differences being greater than $c\sigma$ is small enough, then the corresponding observations (the first, or the first and second observations) should be rejected. The constant c, denoted by λ in Irwin's paper, is to be determined by the person using the method, based on which value for P(difference > $c\sigma$) is sufficiently small. Irwin commented on how to use his method for establishing the "outlierness" of differences when $p > 2$, but he noted that for typical data sets (at that time) "*it does not often happen that there are more than three or four outlying*

observations". As a curiosity, Irwin had strong words in his paper against Peirce's criterion, which was one of the outlier detection methods to which he compared his own method.

Another interesting point of view appeared in (Jeffreys, 1932). According to that paper, "*the probability that a given observation has been affected by an abnormal cause of error is a continuous function of the deviation*". Consequently, it might be better to take those probabilities into account than to completely reject some of the observations. Apparently, Jeffreys was the first one to propose a distribution for the data that is actually a sum of a normal distribution and a "contaminating" distribution:

$$f(\mathrm{x}) = (1-p)\,\frac{e^{-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} + p\,\frac{e^{-\frac{(\mathrm{x}-\mu-\mathrm{y})^2}{2\sigma_c^2}}}{\sqrt{2\pi\sigma_c^2}}, \qquad (1.7)$$

where μ is the population mean, x denotes an observation, σ denotes the standard deviation for normal observations, σ_c denotes the standard deviation of the contaminating distribution, and y denotes a systematic error. It is important to note that this formula was written here using modern notation, but Jeffreys wrote it slightly differently by using the concept "*modulus of precision*" instead of standard deviation. The problem that Jeffreys tackled in his paper was to estimate μ and y, given a series of *N* observations $\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N\}$. He proposed a solution to this problem under two different scenarios:

- The parameters *p*, σ and σ_c were known in advance.
- The parameters *p*, σ and σ_c are unknown, so they have to be estimated from the sample.

In the first case, by carrying out the maximum likelihood method, Jeffreys arrived at expressions to calculate μ and (μ + y) as weighted averages of the observations. Clearly, if p was assumed equal to zero, then we have that μ is equal to the arithmetic mean of the observations and y is indeterminate. Interestingly, under this derivation, large deviations have smaller weights than small deviations in the estimation of μ. On the contrary, large deviations have greater weights in estimating (μ + y). The solution requires successive approximations to the values of μ and y. The second case was also solved using successive approximations on a set of equations which included p, σ and σ_c as unknown variables.

Another work worth mentioning from the early twentieth century is that of William Thompson (Thompson, 1935). It seems to be the first publication where the Student's *t*-distribution was used in an outlier detection method to deal with the fact that most of the time practitioners do not have access to the mean and standard deviation of the population. In that case, those values are approximated by the sample mean $\bar{x}$ and the sample standard deviation s. Unless there are a large number of observations, those approximations do not justify using the normal distribution. To account for the possible error incurred in estimating the population parameters, Thompson defined a new random variable $\tau \equiv \frac{X-\bar{x}}{s}$. He showed that $\tau = t\sqrt{\frac{n+1}{n+t^2}}$, where t follows a Student *t*-distribution with $n = N - 2$ degrees of freedom. Thompson followed the same approach as Stone, fixing a priori the expected number of observations to be rejected, denoted by ϕ, for a sample of size N. Given ϕ and N, a probability p is calculated as $p = \frac{\phi}{N}$. A threshold value τ_0 is obtained such that $P(|\tau| > \tau_0) = p$. Consequently, any observation x_i such that $|x_i - \bar{x}| > \tau_0 s$

is considered an outlier. On average, this method will reject one valid observation (i.e. a non-outlier) in every $\frac{1}{\phi}$ observations.

The work of (Pearson & Sekar, 1936) argued that Thompson's method was essentially a test for the hypothesis H_0 that "*a sample of N observations has been drawn from a single normal population*". They praised Thompson's method because it provides control over the error of type I when rejecting the null hypothesis H_0. However, Thompson did not establish what the alternative hypotheses were. Pearson and Sekar stated as the alternative hypothesis that k observations, $k > 0$, come from normal populations having different means or standard deviations from the population from which the valid $N - k$ observations were drawn. By imposing outer limits in the extreme values for the studentized residuals τ, Pearson and Sekar showed that for samples with two or more outliers which are close together, any attempt to remove them one at a time using Thompson's method is worthless. This fact was subsequently named in literature as the "*masking effect*".

Another statistic from the beginnings of the twentieth century was based on the range of the sample. Having the observations sorted as $x_1 \leq x_2 \leq \cdots \leq x_N$ the statistic $w = \frac{x_N - x_1}{\sigma}$ can be used to establish abnormal observations. However, this statistic was limited to very small samples, since for the more than 12 values the probability law of w "*becomes very sensitive to relatively slight departures from normality in the tails of the population distribution*" (Pearson & Hartely, 1942). Consequently, "*the use of range for control purposes in larger samples is of doubtful value*". The w statistic was subsequently used in the Bliss-Cochran-Tukey rule (Bliss,

Cochran, & Tukey, 1956) in order to determine the presence of outliers from several small samples, each one with *N* observations.

Another interesting outlier rejection method was proposed in (Grubbs F. E., 1950). Again, the observations are assumed to be sorted as $x_1 \leq x_2 \leq \cdots \leq x_N$. This method tests the significance of the largest observation in the sample from a normal population using the following statistic:

$$\frac{S_{1,N-1}^2}{S^2} = \frac{\sum_{i=1}^{N-1}(x_i - \bar{x}_{1,N-1})}{\sum_{i=1}^{N}(x_i - \bar{x})}, \qquad (1.8)$$

where $\bar{x}_{j,k} = \frac{\sum_{i=j}^{k} x_i}{k-j+1}$; $k > j$, and $\bar{x}$ is the mean of the whole sample. If the significance of the smallest observation in the sample was the one to be tested, then a similar statistic $\frac{S_{2,N}^2}{S^2} = \frac{\sum_{i=2}^{N}(x_i - \bar{x}_{2,N})}{\sum_{i=1}^{N}(x_i - \bar{x})}$ can be employed. Grubbs found out that $\frac{S_{1,N-1}^2}{S^2} = 1 - \frac{1}{N-1}\left(\frac{x_N - \bar{x}}{s}\right)^2 = 1 - \frac{1}{N-1}T_N^2$. Here, T_N is the sudentized extreme deviation that Pearson and Sekar used for expanding Thompson's work. Grubbs obtained the exact distribution of $\frac{S_{1,N-1}^2}{S^2}$ (and similarly of $\frac{S_{2,N}^2}{S^2}$) in order to test the corresponding significance. He defined similar statistics to test the significance of either the two largest or the two smallest values in a small sample. It is important to note that Grubbs derived a general recursive expression for the cumulative probability function of $\frac{x_N - \bar{x}}{\sigma}$. That same result was previously published in (McKay, 1935). Grubbs pointed out that his derivation was obtained independently from McKay's work and it was much simpler. Grubbs' statistics are limited to a very small number of observations, given that the distribution on which they are based depends on *N*. Grubbs published tables with four significance values for values of *N* ranging from 2 to 25.

1.3 Main Aspects of a Novelty Detection Problem

Continuous advances in disk storage, memory speed and capacity, computational power, added to the decreasing cost of hardware and the surge of distributed systems running on "commodity hardware", have allowed data mining techniques to expand out of the realm of powerful companies and large institutions to become common place in a variety of application domains. The same computational advances have allowed the implementation of more complex and accurate algorithms. The multiple domains to which data mining techniques are currently applied have characteristics that determine the specific formulation of the problems to be solved. This section describes the main aspects of a novelty detection problem, particularly those that are determined by the corresponding application domain.

According to (Chandola, Banerjee, & Kumar, 2009), the most important factors determining the formulation of an anomaly detection problem are the following: the nature of data, the type of output, the type of anomaly, and the availability or unavailability of data labels. This section extends the categorization given in (Chandola, Banerjee, & Kumar, 2009) by adding two other factors that are also important when defining a novelty detection problem: computational requirements and the learning framework. These aspects are shown in Figure 1.1 below, inspired on a similar diagram from (Chandola, Banerjee, & Kumar, 2009). They are described in the subsections that follow. If some characteristics of these aspects were particularly relevant for the research described in this dissertation then they will be noted in the corresponding subsections.

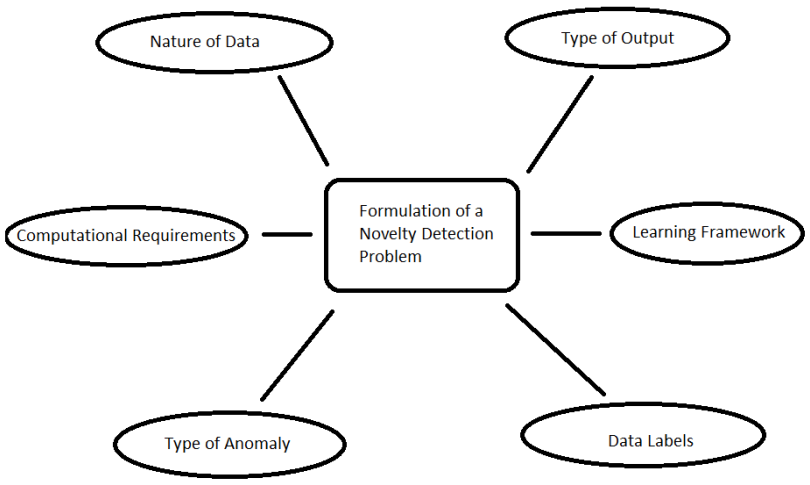

Figure 1.1: Main aspects of a novelty detection problem.

1.3.1 Nature of Data

Input data can be defined as a collection of data instances that represent fundamental characteristics of objects from the application domain. In this dissertation, data instances are also called *observations*. Observations are typically stored as univariate or multivariate vectors. Each component of an observation is denoted in this dissertation by the term *attribute* (*feature* and *characteristic* are also used here sporadically). Attributes might be numerical, categorical, unstructured text, images, videos, or sound, among other types. In the case of multivariate observations, whether attributes are of the same type or not makes an important difference regarding the algorithms that can be applied. Most of the current approaches have been focused on detecting outliers exclusively on a particular type of data. However, data sets with a mixture of data types (also called *mixed-attribute data sets*) appear in many real-world applications. A

very common case is the mixture of numerical and categorical data (Otey, Ghoting, & Parthasarathy, 2006), (Koufakou & Georgiopoulos, 2010).

This dissertation focuses on a particular type of kernel method for novelty detection. Kernel methods effectively decouple the underlying data types of the observations from the particular algorithm employed. Consequently, novelty detection techniques proposed here are not sensitive to changes in data types, as far as an appropriate kernel can be found.

Another useful classification for the nature of data is based on relationships among the observations (Tan, Steinbach, & Kumar, 2005). Most novelty detection algorithms assume no relationship among data instances. In that case, observations are also denoted by the terms *data points* and *data records*. Some possible relationships in related data are: sequential, spatial, and spatio-temporal. *Sequence data* contains linearly ordered data; for instance, time-series (also called *temporal data*), and genome sequences. *Spatial data* contain one or more attributes describing the spatial location of observations. Main examples are geological and ecological data. Spatial data with a temporal attribute is referred to as *spatio-temporal data*. Finally, the term g*raph data* typically denotes data that contain more general relationships (social data is a prime example of this category). A good review of current outlier detection methods for temporal data, including spatio-temporal data and sequences of graphs, is given in (Gupta, Gao, Aggarwal, & Han, 2014). Although data relationships are a fundamental component of some novelty detection techniques, this dissertation is not concerned with modeling data relationships. However, it is important to note that the methods employed in this work could benefit from such relationships through the use of kernel functions that are designed to leverage them.

1.3.2 Type of Output

Typically, the output from novelty detection methods are of two types: *labels* and *scores* (Chandola, Banerjee, & Kumar, 2009). Techniques that use labels classify each observation as either *outlier* or *normal*. Scoring techniques assign to each observation a score value (typically a real number) that states either its degree of "outlierness" or its degree of membership to the normal class. Score values are denoted by the term *novelty scores* or *membership scores*, respectively. Novelty scores allow analysts and researchers to maintain a ranked list of observations that were classified as outliers. Consequently, they can focus on the most relevant anomalies. Furthermore, novelty scores might be an important feature in some application domains to determine whether an outlier or a group of outliers should be added to the normal model or not. Finally, if the need appeared, then scores can be converted into labels by defining a threshold value. Because of the advantages of novelty scores over labels, this dissertation focuses on scoring algorithms.

The work of (Breunig, Kriegel, Ng, & Sander, 2000) was the first one to use scores to describe outliers. It was an important step towards establishing novelty scores through rankings. The work in (Hawkins, He, Williams, & Baxter, 2002) used the reconstruction error of replicator neural networks as the anomaly score for each observation. A rule-based approach was employed in (Fan, Miller, Stolfo, Lee, & Chan, 2001), defining novelty scores as the inverse of confidence factors. In (He, Xu, Huang, & Deng, 2004), novelty scores of categorical observations were defined based on the number of frequent itemsets in which they appeared. The work of (Byers & Raftery, 1998) calculated the novelty score of a data point as the distance to its k^{th} nearest neighbor.

1.3.3 Type of Anomaly

Anomalies can be classified into three different categories: *point anomalies*, *contextual anomalies,* and *collective anomalies*. Point anomaly is the simplest type of 'outlierness'. An observation is a point anomaly when it is considered an outlier with respect to data that is considered normal. For instance, a particularly high or low credit card transaction compared to the typical expenditure pattern of the card holder could be considered an anomaly. Another example is a very unusual sensor reading, far beyond the range of previous observations. Most novelty detection techniques focus on this type of anomaly.

Contextual anomaly (also called *conditional anomaly*) establishes that an observation can be considered an outlier only within a particular context (Song, Wu, Jermaine, & Ranka, 2007). The notion of a context is induced by the structure in the data set. Typically, attributes are classified as either *contextual attributes* (those defining the particular context on which the observation lies) or *behavioral attributes* (those containing non-contextual characteristics of the observations). Contextual information might be very useful when available (for instance, to deal with segmented data). However, deciding on which contextual attributes are appropriate is not always a straight-forward process. Contextual novelty detection has been explored mainly in the presence of related data, e.g. (Salvador & Chan, 2005), (Kou, Lu, & Chen, 2006).

A set of observations is called a *collective anomaly* if the occurrence of all the observations together is suspiciously abnormal, but the individual observations might not be anomalies by themselves. The following are examples of recent work on collective anomalies: (Shekhar, Lu, & Zhang, 2002), (Noble & Cook, 2003), (Sun, Chawla, & Arunasalam, 2006), and (Kou, Lu, &

Dos Santos, 2007). Collective anomalies cannot occur in data containing unrelated observations. Consequently, this dissertation does not consider algorithms that look for collective anomalies. Furthermore, this work is specifically interested in the problem of detecting point anomalies.

1.3.4 Data Labels

A training data set is labeled when each training observation has an attribute denoting the correct response that should be given by a machine learning algorithm that learnt a model from that data set. In the case of novelty detection, that label is the correct output for reporting an anomaly, i.e. a binary label or a score value. The existence or absence of labels in a data set defines the type of learning task to be undertaken. Essentially, there are three different types of learning tasks based on data labeling: *supervised, semi-supervised* and *unsupervised novelty detection*.

Techniques following a supervised learning approach assume that both normal and abnormal observations are correctly labeled. In those cases, a classification algorithm might be employed. However, most classification techniques need to be adapted because a major difficulty typically not present in traditional classification theory: imbalanced class distribution. It is common to have in a data set many more observations coming from normal data than from the class of outliers. Several authors have addressed this imbalance issue in different ways; e.g. (Joshi, Agarwal, & Kumar, 2001), (Joshi, Agarwal, & Kumar, 2002), (Phua, Alahakoon, & Lee, 2004) and (Vilalta & Ma, 2002).

For some domains it is very difficult to obtain a representative set of labeled outliers. There are two main reasons for this difficulty. First, outliers are typically rare observations in comparison to what is considered normal, and sometimes their rare occurrences are attached to high cost

effects (such as airplane engine failures or ecological catastrophes). Second, it is almost impossible to predict in advance every possibly type of anomaly that might appear in most systems. Consequently, supervised novelty detection has very limited applicability in practice.

Semi-supervised novelty detection methods are designed to be trained on a data set containing labels only for normal observations. The basic learning approach in this case is to find a model for the normal class. New observations not fitting well into that model are labeled as anomalies. Although semi-supervised techniques are more broadly applicable than supervised novelty detection techniques, sometimes it is difficult or even impossible to gather a representative training data set encompassing the normal class. Among the most important reasons behind that limitation we have: (1) it might be very difficult to define a region encompassing every possible normal observation; (2) the boundaries between normal and anomalous observations are not always well-defined; (3) the concept of normality might be changing with time, potentially turning previous labels as incorrect. The final section of this chapter explains how these limitations can be overcome with the use of online learning methods and robust techniques.

The third type of learning task, *unsupervised novelty detection*, deals with data sets without label information at all. Methods following an unsupervised approach are the most widely applicable. However, the lack of labels forces these techniques to implicitly assume that normal observations are much more frequent than outliers; which is not necessarily the case because outliers can be members of an undefined but large class. Additionally, outliers are assumed to be qualitatively different from normal observations. These assumptions become requirements on the application domains to which unsupervised novelty detection is applied. This dissertation focuses on

learning models of the normal class employing semi-supervised learning algorithms, which are not restricted by the assumptions of the unsupervised approach.

1.3.5 Computational Requirements

There are some application domains with very specific requirements or limitations. In those cases, traditional novelty detection techniques might not be directly applicable, and techniques specifically tailored to those constraints need to be devised. *Sensor networks* are a good example of an application domain with very particular characteristics which requires specialized novelty detection techniques; e.g. (Sheng, Li, Mao, & Jin, 2007) (Zhang, Meratnia, & Havinga, 2010). There are other scenarios where data are distributed across several nodes and novelty detection needs to be performed on the data as a whole without revealing sensitive information between nodes. That scenario is commonly called *privacy-preserving outlier detection*. Recent works on this specific domain are: (Vaidya & Clifton, 2004), (Aggarwal & Yu, 2008) and (Dai, Huang, Zhu, & Yang, 2010). This dissertation is not aimed at specifically constrained application domains.

1.3.6 Learning Framework

Outlier detection algorithms are typically trained off-line using a fixed training data set. The model obtained after the training phase is subsequently evaluated on new observations, which are taken from either a previously stored testing data set or data not available when the algorithm was trained. This approach is called *batch learning* or *off-line learning*. On the other hand, algorithms that can update their model incrementally while learning from a sequence of observations are called *online learning algorithms*.

Designing and implementing efficient batch learning algorithms becomes particularly hard when the amount of training data is very large. Even with an efficient implementation, expensive hardware might be required. Alternatively, online learning algorithms are well suited for large data sets. Furthermore, online learning can be particularly useful to keep models up-to-date when training data become available as a stream of data. This dissertation focuses on both batch and online learning algorithms.

1.4 Modern Approaches to Novelty Detection

Two recent surveys of the different approaches to outlier/novelty detection can be found in (Chandola, Banerjee, & Kumar, 2009) and (Pimentel, Clifton, Clifton, & Tarassenko, 2014). According to (Chandola, Banerjee, & Kumar, 2009), the different approaches to outlier detection methods can be classified into six broad groups: (1) *statistical methods*, (2) *classification-based methods,* (3) *clustering-based methods,* (4) *nearest neighbor-based methods,* (5) *information theoretic methods,* and (6) *the spectral approach*. Statistical methods typically estimate the probability distribution of the data and use statistical tests to determine whether new observations are potential outliers. Methods relying on other statistical techniques, such as linear and nonlinear regression and Gaussian processes, are also members of this category. The classification-based category refers to methods that were originally developed to solve binary or multi-class classification problems, but were subsequently modified to work as one-class classifiers. The clustering approach includes mostly methods that rely on unsupervised learning to determine one or more clusters of observations that belong to the normal class. Nearest neighbor-based methods take into account the distances to neighboring observations when

determining whether an observation is an outlier. Information theoretic methods assume that outliers have the highest impact on the information content of the data set, as estimated by an information theoretic measure. Finally, the spectral approach refers to methods that project the input data into a subspace.

The categorization given in (Pimentel, Clifton, Clifton, & Tarassenko, 2014) differs in various ways from the one described above. It lists the following categories: (1) *probabilistic methods*, (2) *distance-based methods,* (3) *reconstruction-based methods*, (4) *domain-based methods*, and (5) *information-theoretic methods*. Similar to (Chandola, Banerjee, & Kumar, 2009), the category of probabilistic methods include methods that estimate the generative density functions of the normal data and use hypothesis testing. Methods in this category can be classified into parametric and non-parametric. In the first subcategory there are methods leveraging parametric techniques, like Gaussian mixture models (GMMs), time-series techniques like ARIMA and ARMA, and state-space models like hidden Markov models (HMMs), Kalman filters and dynamic Bayesian networks. The non-parametric subcategory includes methods leveraging non-parametric techniques, like histograms and kernel density estimators (such as the Parzen windows estimator and Gaussian processes). Contrary to (Chandola, Banerjee, & Kumar, 2009), in (Pimentel, Clifton, Clifton, & Tarassenko, 2014) regression models are not fully included in the category of probabilistic methods: some methods using auto-regressive models are listed as probabilistic methods, while other methods using regression models are mentioned also in the "reconstruction-based" category. Distance-based methods consider mainly the subcategories of nearest neighbor-based methods and clustering-based approaches. The reconstruction-based approach to novelty detection consider methods that model the underlying data and determine

whether a new observation is an outlier based on its distance to the model's output (the reconstruction error). The main subcategories of the reconstruction-based approach are the neural network-based approach and the subspace-based approach (called the spectral approach in (Chandola, Banerjee, & Kumar, 2009)). Domain-based novelty detection refers to methods that construct boundaries around the normal class, without considering the actual class density or any approximation to it. Finally, the information-theoretic approach denotes exactly the same type of methods listed under that name in (Chandola, Banerjee, & Kumar, 2009).

The introduction to current methods in novelty detection given in this section follows mainly the classification proposed in (Chandola, Banerjee, & Kumar, 2009), with three modifications, two of them inspired by the review of Pimentel et al.: First, the nearest neighbor-based approach is considered a subcategory of the more general *distance-based* approach. This is based on the fact that all nearest-neighbor techniques require a distance, but distance-based techniques are not restricted to dealing exclusively with local information (as explained in a subsection below, there are distance-based methods that aim at finding global outliers). However, contrary to the categorization of Pimentel et al., we maintain the clustering-based approach as a separate category. A reason for that is that some clustering techniques do not exclusively rely on distances, but they also employ subspaces and density estimation, among other techniques. Furthermore, although most clustering algorithms explicitly rely on a distance, it is possible to find clusters by employing a distribution-based approach; for instance, using the expectation-maximization (EM) algorithm to estimate the parameters of Gaussian mixture models that better fit the data. Consequently, it is considered more important here to set aside the clustering approach (which does not have novelty detection as its original goal but as a sub-product of it),

than to limit it to a subcategory of distance-based novelty detection or to split it as various subcategories within the other categories. Second, the category of spectral methods is renamed here as *subspace-based methods*, which better describes the intention of methods included in that category. Finally, we have added a new category that was not considered in neither of the two reviews mentioned above: *angle-based methods*. Those methods benefit from the fact that angles are more stable than distances when working with high-dimensional data. In summary, this dissertation proposes the following categorization of modern novelty detection approaches: (1) *statistical,* (2) *classification-based,* (3) *clustering-based,* (4) *distance-based,* (5) *information theoretic*, (6) *subspace-based*, and (7) *angle-based*.

The following subsections briefly introduce modern approaches to novelty detection. It is important to note a few things before delving into these subsections. First, the area of novelty/outlier detection is so broad and dynamic that there could exist one or more methods not included in our literature review for which none of the categories described here is a good fit. Furthermore, the following subsections do not attempt to describe all possible techniques that fit into these categories, but to offer a representative set of examples of modern methods in each category. Second, there are methods that leverage a combination of approaches, and sometimes they can be considered members of multiple categories. Just to name a few examples: (Filev & Tseng, 2006) leverages fuzzy k-nearest neighbors clustering and the statistical technique Gaussian mixture models (GMMs) to model machine health status and predict anomalous conditions; the work in (Galeano, Peña, & Tsay, 2006) uses projection pursuit (a subspace technique) and an autoregressive moving average (ARMA) model (a statistical technique) in order to find outliers in multivariate time series; and the novelty detection method proposed in

(Kit, Sullivan, & Ballard, 2011) uses a growing neural gas (Fritzke, 1995) to detect changes in videos taken by a robot. Growing neural gas is a type of neural network, and in our review neural networks belong to the classification-based category. However, that particular type of neural network is essentially an incremental clustering algorithm, so that method could be categorized as clustering-based as well.

1.4.1 Statistical Novelty Detection

As noted in section 1.2, the statistical approach is the oldest. In general terms, statistical methods can be classified as *parametric* versus *nonparametric*, and *numeric* versus *categorical*. Parametric methods assume that the distribution of the normal class, denoted by $F(\mathbf{x}, \Theta)$, is known or can be effectively estimated from training data. The argument **x** denotes an observation and Θ denotes a vector of parameters. Typically, there are two ways of dealing with possible outliers. A hypothesis test can be applied with the null hypothesis that an observation **x** was generated from the distribution underlying normal data (Barnett & Lewis, 1994), (Eskin, 2000). This type of test is typically known as *outlier discordancy test*. The observation **x** can be considered an outlier if the null hypothesis was rejected. In that case, the corresponding test statistic can be used to provide a novelty score value for **x.** Alternatively, a novelty score for an observation **x** can be defined based on a previously defined criteria. For instance, the novelty score of an observation **x** can be equal to $\frac{1}{f(\mathbf{x},\Theta)}$, where *f* is the probability density function of the normal data. This example is an instance of *density-based novelty detection*, which is closely related to the local distance-based approach described in one of the following subsections.

Another example of a statistical novelty score is the distance of the observation to the estimated mean of the data from the normal class assuming a Gaussian distribution.

Whenever a novelty score is used without applying an outlier discordancy test, thresholds are needed to discriminate between normal and novel observations. The most widely-known statistical novelty threshold, employed for Gaussian models, is to declare as outliers those observations lying outside the $\mu \pm 3\sigma$ region, where μ denotes the distribution mean and σ denotes the standard deviation of the distribution. The *box plot rule* is another commonly employed technique. Any observation outside of the interval $[Q_1 - 1.5IQR, Q_3 + 1.5IQR]$ is declared an outlier, where Q_1 is the lower quartile, Q_3 is the upper quartile, and IQR denotes the inter quartile range (Horn, Feng, Li, & Pesce, 2001). The *Grubb's test*, which also assumes a Gaussian distribution but uses mean and standard deviation of a data sample, is another parametric technique worth of mentioning. Grubb's test was originally proposed for univariate data (Grubbs F. , 1969). However, it has been expanded to multivariate data (Aggarwal & Yu, 2001), (Aggarwal & Yu, 2008) and graph structured data (Shekhar, Lu, & Zhang, 2002).

Sometimes data from the normal class cannot be properly fitted by a Gaussian distribution but it can be modeled as a mixture of parametric distributions. The most common mixture is the combination of two or more Gaussian distributions, which is called a *Gaussian mixture model* (GMM). A brief introduction to GMMs is given in (Reynolds, 2009). Examples of novelty detection algorithms using GMMs can be found in (Song, Wu, Jermaine, & Ranka, 2007), (Agarwal, 2007), (Ilonen, Paalanen, & Kamarainen, 2006) and (Roberts, 2000). Mixtures of non-Gaussian distributions have been used for novelty detection as well. For instance, a mixture of

Poisson distributions was used in (Byers & Raftery, 1998). Note that the work of (Roberts, 2000) is also an example of another statistical technique applied to novelty detection: extreme value theory (EVT). A description of recent methods that apply EVT to novelty detection using multivariate and multimodal distributions is given in (Clifton, Hugueny, & Tarassenko, 2011).

Time series analysis is another area where parametric outlier detection has been widely used. The most common approach is to fit an autoregressive model to the training data and to use the magnitude of the residual corresponding to a new observation as its novelty score. Robust regression (Rousseeuw & Leroy, 1987) is typically used to minimize the effect of outliers that might be present in the training data. The work of (Hoares, Asbridge, & Beatty, 2002) proposed a method called the Automatic Dynamic Data Mapper (ADDaM), which outperformed other time-series methods when employed to detect artefacts in heart rate data. Several regression-based novelty detection techniques have been devised to handle multivariate time-series data; e.g., (Tsay, Peña, & Pankratz, 2000), (Chen, Chao, Hu, & Su, 2005), (Galeano, Peña, & Tsay, 2006).

Parametric methods using state-space models are typically used to detect outliers in time-series data, but they are listed separately here given that the approaches employed in those cases are not related to autoregressive modeling. State-space models typically contain a set of observed variables and a set of hidden states, both evolving through time. They assume that the distribution of the observed variables depend on the values of the hidden states at each particular point in time. These models include conditional probability distributions describing the likelihood of moving from state to state and also the likelihood of each possible observation

given each state. Hidden Markov models (HMMs) and Kalman filters are the most commonly used state-space models in novelty detection. Examples of recent works leveraging HMMs can be found in (Yeung & Ding, 2003), (Ariu, Giacinto, & Perdisci, 2007), and (Ntalampiras, Potamitis, & Fakotakis, 2011). Examples of recent methods using Kalman filters include (Quinn & Williams, 2007), (Lee & Roberts, 2008), and (Quinn, Williams, & McIntosh, 2009). Finally, more general probabilistic graphical models, such as dynamic Bayesian networks (DBNs), have also been employed for novelty detection; e.g. (Janakiram, Adi Mallikarjuna Reddy, & Phani Kumar, 2006) and (Pinto, Pronobis, & Reis, 2011).

In many real-world scenarios, it is not possible to define a priori the underlying distribution of the training data; which greatly limits the practical importance of parametric models. Nonparametric methods are more useful in those cases. They assume only some degree of smoothness from the underlying density in order to maintain a profile of the normal class. Among the most common profile-keeping techniques there are *histogram-based* and *density-based profiling* methods. The first approach typically involves constructing and maintaining an attribute-wise histogram for the data from the normal class. The novelty score assigned to a new observation is directly proportional to the heights of the bins of the histogram containing each attribute. Histogram-based novelty detection has been particularly useful for intrusion detection (Eskin, 2000), (Mahoney & Chan, 2002); structural damage detection (Manson, 2002); fraud detection (Yamanishi, Takeuchi, Williams, & Milne, 2004); and Web attacks detection (Kruegel & Vigna, 2003); among other domains.

The density-based approach involves using kernel functions to estimate the probability density function of the normal class. Parzen windows estimation (Parzen, 1962) is a commonly used density estimation technique for novelty detection; e.g. (Desforges, Jacob, & Cooper, 1998), (Yeung & Chow, 2002), (Vincent & Bengio, 2002), (Bengio, Larochelle, & Vincent, 2005), and (Fairley, Georgoulas, Stylios, & Rye, 2010). Recently, Gaussian processes (GPs) originally intended for regression have been leveraged to accomplish outlier detection, showing very good results on various data sets when compared to other state of the art kernel methods (Kemmler, Rodner, & Denzler, 2010), (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Four GP-based score functions were proposed (and compared to each other) to translate the output of GP-based regression to membership scores. The core proposition of this dissertation lies in further improving this particular method by making different variants of GPs more robust to outliers present in the training data. Consequently, GP-based novelty detection will be reviewed in detail in a subsequent chapter, among other state-of-the-art kernel methods.

Note that the statistical techniques described above are most suitable for numerical data. However, some of them may be applied to categorical data as well. For instance, histograms have been used to estimate the probability mass functions of categorical data (Yamanishi, Takeuchi, Williams, & Milne, 2004). As another example, informal box plots have been employed for novelty detection on ordinal and categorical data (Laurikkala, Juhola, & Kentala, 2000). Finally, the GP-based method proposed in (Kemmler, Rodner, & Denzler, 2010) can be applied to any type of data, as far as there is a kernel function defined for it.

1.4.2 Classification-based Novelty Detection

The classification-based approach involves training a classifier to discriminate between normal and anomalous data. Traditionally, classifiers are trained on a data set containing labeled examples for all the classes involved in the domain to be learned. A testing phase involves assigning labels to new observations. In the case of novelty detection, a classifier must learn a model from positive instances that are considered normal. The testing phase proceeds by recognizing whether new observations correspond to one of the normal classes or they should be declared outliers. In a broad sense, classification-based novelty detection can be split in two groups: *one-class* and *multi-class* anomaly detection. One-class algorithms assume that normal training examples belong to a single class. One-class Support Vector Machines (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001) and the Support Vector Data Description (SVDD) method (Tax & Duin, 2004) constitute two of the most representative examples of state of the art one-class kernel methods. For that reason, they will be described in more detail in a future chapter. In the case of multi-class novelty detection, a classifier must learn from a training data set containing normal examples from two or more classes. Among the most commonly used multi-class classifiers are *neural networks* (NNs), *Bayesian networks*, *rule-based classifiers*, and some *kernel-based classifiers*.

Neural networks have been applied both in one-class and multi-class scenarios; e.g. (Odin & Addison, 2000), (Stefano, Sansone, & Vento, 2000), (Augusteijn & Folkert, 2002), (Hawkins, He, Williams, & Baxter, 2002). A good review of applications of different types of NNs to novelty detection until around 2003 can be found in (Markou & Singh, 2003). Examples of more recent applications are given below.

In (Marsland, Nehmzow, & Shapiro, 2005), a "grows when required" neural network (GWR network) (Marsland, Shapiro, & Nehmzow, 2002) was employed to make a mobile robot ignore observations that were very similar to previous input, while highlighting novel parts of its environment. Similarly focused on robotic sensing, (Kit, Sullivan, & Ballard, 2011) proposes the use of a growing neural gas (Fritzke, 1995) to detect environmental changes using a camera. The work of (Haggett, Chu, & Marshall, 2008) employed a dynamic predictive coding neural network as a novelty detector. It compared three evolutionary algorithms to optimize the network structures: a simple genetic algorithm, NEAT (Stanley, 2004), and FS-NEAT (Whiteson, Stone, Stanley, Miikkulainen, & Kohl, 2005); with NEAT-optimized networks outperforming other networks. In (Wu, Wang, & Lee, 2010), an online fault detection method based on a self-organized map (SOM) was used as part of a maintenance system. The use of SOMs to detect anomalies in time series is explored in (Barreto & Aguayo, 2009). As a final example, the work of (García-Rodríguez, Angelopoulou, García-Chamiz, Orts-Escolano, & Morell-Giménez, 2012) uses a modified learning algorithm for a growing neural gas network to satisfy certain real-time constraints.

Bayesian networks are probabilistic classifiers, thus novelty detection methods that leverage them can be also considered examples of the statistical approach. As classifiers, they are typically used in multi-class scenarios where there are some examples from the abnormal class as well (i.e. in supervised learning). Given a new observation, they estimate the posterior probabilities of the class labels based on the prior probability of that observation conditioned on each label and the prior probabilities of the normal and abnormal classes. Novelty detection techniques using Bayesian networks can be classified in two broad disjoint groups: those

assuming independence between the data attributes, e.g. (Barbara, Couto, Jajodia, & Wu, 2001), (Sebyala, Olukemi, & Sacks, 2002), (Bronstein, et al., 2001), (Diehl & Hampshire, 2002) and (Wong, Moore, Cooper, & Wagner, 2003); and those assuming conditional dependencies between some attributes, e.g. (Janakiram, Adi Mallikarjuna Reddy, & Phani Kumar, 2006) and (Das & Schneider, 2007).

Rule-based classifiers are based on a set of rules that together model the response of the system to each observation. They have been applied in single-class and multi-class discrimination problems. In the case of novelty detection, rule-based classifiers label a new observation as an outlier if no rule labeling it as part of the normal class was found. Typically, rule-based techniques consist of two phases: a training step, in which a rule-learning algorithm learn 'normality rules' from the training data set; and a testing step, in which the algorithm must identify whether or not there are rules covering new observations as normal. In rare occasions in which a representative sample of the outliers' class is available, there could be rules covering it as well.

A few examples of rule-based novelty detectors are the following: The well-known C4.5 algorithm to generate decision trees (Quinlan, 1993) has been used to detect outliers in categorical data (John, 1995). A learning rule algorithm known as RIPPER has been employed to describe temporal states constituting the normal operation of devices, which in turn can be used to detect anomalies (Salvador & Chan, 2005). Association rule mining (Agrawal & Srikant, 1995) has been employed for unsupervised one-class novelty detection on categorical data sets; e.g. (Mahoney & Chan, 2003), (He, Xu, Huang, & Deng, 2004), (Tandon & Chan, 2007).

Kernel-based classifiers have been particularly successful in recent years. Among them, support vector machines (SVMs) are likely the most widely used. They were originally defined as binary classifiers that find the maximum-margin separating hyperplane between instances of two classes. SVMs manage to obtain non-linear separating surfaces in the input space by applying linear techniques on a higher-dimensional feature space to where observations are mapped (Vapnik, 1995), (Abe, 2010). SVMs rely on kernel functions to accomplish the feature mapping (Shawe-Taylor & Cristianini, 2004). They have been employed in novelty detection through the *one-class SVM* approach, in which a SVM learns a boundary of a region containing the normal observations (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001). New observations are declared outliers if they resided outside of the region encompassing data from the normal class. The defining boundary is found by separating the training data from the origin in the feature space using the maximum-margin hyperplane approach.

An interesting application of the one-class SVM approach is detecting seizures in human EEG (Gardner, Krieger, Vachtsevanos, & Litt, 2006). In that work, intracranial normal EEG time series were partitioned into one-second segments that were used for the SVM to learn a model for normal EEG. Another application of one-class SVM to temporal data can be found in (Ma & Perkins, 2003). Other examples of recent works using one-class SVM are (Hardoon & Manevitz, 2005), (Zhuang & Dai, 2006), (Rabaoui, Kadri, & Ellouze, 2008), (Clifton, Clifton, Watkinson, & Tarassenko, 2011), (Zhu, Ye, Yu, Xu, & Li, 2014), (Metzler & Kalinina, 2014). Additionally, robust SVMs have been employed to better adjust to the likely presence of outliers within the training data (Song, Hu, & Xie, 2002), (Hu, Liao, & Vemuri, 2003).

One-class SVM relies on a parameter that denotes the expected percentage of outliers in the training data, which are allowed to remain outside of the region defining the normal class. It has been noted that the effectiveness of this method is highly affected by setting the value of this parameter (Manevitz & Yousef, 2002). The one-class Kernel Fisher Discriminant (KFD) classifier (Roth, 2006) was proposed to overcome this limitation. It relates one-class kernel-based classification to Gaussian density estimation in the feature space. A cross-validated likelihood criterion is used to estimate all the parameters of the model. Abnormal observations are those considered highly unlikely according to the Gaussian model.

The Support Vector Domain Description (SVDD) method (Tax & Duin, 1999) finds the hypersphere with minimum volume that contains all or most of the training data in the feature space. Although this method was designed to be a one-class classifier from its inception, it is included here because it is inspired by SVM. Furthermore, its mathematical derivation follows the same approach as SVM: it uses Lagrange multipliers to optimize a regularized expression (consisting of the squared radius of the hypersphere and the sum of slack variables denoting how much each point can outspread beyond the hypersphere). As in SVM, a subset of the training data is obtained as support vectors, and the rest of the training data can be safely discarded. After training the model, a new observation is considered an outlier if its distance to the center of the hypersphere is greater than the optimized radius. This method was subsequently expanded in (Tax & Duin, 2004) to learn also from negative examples if they were present in the training data. The expanded method was named Support Vector Data Description, although the acronym remained as SVDD. From here on, the term SVDD refers to the expanded method unless it is clearly stated otherwise. Note also that the expanded SVDD behaves exactly like the original

SVDD if no negative examples were labeled as such in the training data. Some extensions to SVDD have been proposed recently. Some of them have focused on improving the efficacy of the boundaries of the hypershere, e.g. (Wu & Ye, 2009) and (Le, Tran, Ma, & Sharma, 2010). Other extensions have focused in solving an optimization problem that includes various hyperspheres with different centers and radii (Le, Tran, Ma, & Sharma, 2011). Finally, some extensions have been proposed to improve the time complexity of SVDD, e.g. (Liu, Liu, & Chen, 2010) and (Peng & Xu, 2012).

Single-class Minimax Probability Machine (MPM) (Lanckriet, Ghaoui, Bhattacharyya, & Jordan, 2002) is another example of a kernel-based classifier that has been used for novelty detection. The reader can refer to (Lanckriet, El Ghaoui, & Jordan, 2003) and (Kwok, Tsang, & Zurada, 2007) for details on the MPM classifier.

1.4.3 Clustering-based Novelty Detection

Clustering is the action of grouping similar observations into classes. Each class must contain very similar observations while, at the same time, observations from different classes should be as dissimilar as possible (Tan, Steinbach, & Kumar, 2005). Clustering techniques have been traditionally linked to unsupervised learning. However, clustering has been applied in a semi-supervised scenario as well (Basu, Bilenko, & Mooney, 2004).

Novelty detection techniques based on clustering can be classified in two broad categories: clustering techniques that force every observation to belong to one of the clusters found in the data, and those that do not enforce cluster membership for all observations. In both cases, the basic approach consists of two steps: First, a clustering algorithm is applied to the training data

set. Second, some criteria are applied in order to determine which observations should be classified as outliers. When the clustering algorithm does not force all observations to belong to a cluster, the simplest criterion is to label as outliers those observations without a cluster membership; e.g. DBSCAN (Ester, Kriegel, Sander, & Xu, 1996), ROCK (Guha, Rastogi, & Shim, 2000), The *FindOut* algorithm (Yu, Sheikholeslami, & Zhang, 2002), and SNN (Ertöz, Steinbach, & Kumar, 2003). If the clustering technique assigned cluster memberships to all training observations, like the widely used *k*-means clustering, choosing what observations should be outliers is not straightforward. The most common criterion is to classify as outliers those observations lying far away from their closest cluster centroid (Smith, Bivens, Embrechts, Palagiri, & Szymanski, 2002), (Clifton, Bannister, & Tarassenko, 2007).

Note that the *k*-means clustering algorithm, although widely used, is very sensitive to outliers in the training data. To alleviate this limitation, a recent work has proposed to combine clustering and outlier detection within the same unified approach, extensible to all distance measures that can be expressed as a Bregman divergence (Chawla & Gionis, 2013). Another limitation of the *k*-means algorithm is that it requires the number of clusters *k* as an input parameter. The work in (Lei, Zhu, Chen, Lin, & Yang, 2012) has attempted to automate *k*-means clustering by estimating the number of clusters in the data through a method called *subtractive clustering*. Finally, it is well known that *k*-means clustering is very sensitive to the initial assignment of cluster centers (Peña, Lozano, & Larrañaga, 1999), which can lead the algorithm to local minima. Recent works have addressed this limitation as well; e.g. (Khan & Ahmad, 2004) and (Ahmed & Ashour, 2011). Some research has been devoted to novelty detection using fuzzy variants of *k*-means clustering. Within that area, it is relevant to note the works in (Wang, 2009) and (Filippone,

Masulli, & Rovetta, 2010), that applied a kernel-based approach in combination with fuzzy clustering.

After clusters are obtained by using a technique from either of the two broad categories mentioned above, a new test observation is typically labeled as an outlier based on how distant it is from the nearest cluster (or the centroid of the nearest cluster). Regardless of the cluster membership policy employed, sometimes several outliers were close enough to each other as to constitute a cluster by themselves. For that reason, some clustering methods also label as outliers the members of clusters whose size and/or density lies below certain threshold (Eskin, Arnold, Prerau, Portnoy, & Stolfo, 2002), (Pires & Santos-Pereira, 2005), (He, Xu, & Deng, 2003). Some clustering-based techniques are particularly designed to deal with very large data sets; for instance, see (Zhang, Ramakrishnan, & Livny, 1997), (Chiu, Fang, Chen, Wang, & Jeris, 2001) and (Yu, Sheikholeslami, & Zhang, 2002). Other clustering-based methods attempt to detect the appearance of new normal classes within an online learning framework (Spinosa, de Leon F. de Carvalho, & Gama, 2009). As a final example, (Zhou, Fu, Sun, & Fang, 2011) proposes a distributed novelty detection method, for scenarios where data are distributed across multiple computers and cannot be merged.

Most clustering-based methods require a distance defined on the input space. Consequently they appear to be very similar to distance-based novelty detectors. However, clustering-based novelty detectors cannot be classified into global or local techniques, because distance calculations are determined by cluster memberships. Another important difference is that the clustering-based

approach detects outliers as a by-product of the underlying clustering methods, which do not have finding outliers as a primary goal.

1.4.4 Distance-based Novelty Detection

Distance-based methods require the formulation of a distance (or, equivalently, a similarity function) defined on pairs of observations from the input space. Outliers can be determined based on the distance from an observation to other observations in the data set. In contrast to statistical methods, distance-based methods do not require an underlying distribution. Additionally, they are particularly well-suited to unsupervised learning scenarios. Some definitions of outliers use a global approach, where the distance of an observation to all other observations in the training data set is considered. Alternatively, a local approach can be employed, focusing on a neighborhood around each data point. The local approach is typically called *nearest neighbor-based novelty detection*.

The work of (Knorr & Ng, 1997) is a good example of a global approach to outlier detection. It defines an object O to be an outlier if "at least a fraction *p* of the objects in the data set lies greater than distance D from O". Alternatively, nearest neighbor-based methods assume that outliers occur in very low density neighborhoods. They can be divided in two broad subcategories: *k*-NN methods, based on the distances of observations to their k^{th} nearest neighbors, where *k* is a fixed integer; and methods focusing on the relative density around an observation by employing neighborhoods of a fixed measure. The later subcategory is known as the *density-based* approach.

One of the first papers proposing the nearest neighbor approach for outlier detection was (Hellman, 1970). Hellman's method looks at the *k*-nearest neighbors of the testing observation. A rejection rule based on the quantity of neighbors belonging to the same class was employed. Other novelty detection algorithms using the *k*-nearest neighbors approach show variations in the way novelty scores are calculated. For instance, the novelty score of a testing observation can be calculated as the distance to its k^{th} nearest neighbor (Byers & Raftery, 1998). Alternatively, the novelty score can be calculated as the sum of distances to the *k* nearest neighbors; e.g. (Eskin, Arnold, Prerau, Portnoy, & Stolfo, 2002), (Zhang & Wang, 2006). The work of Zhang and Wang tackles a problem that goes beyond deciding whether observations are outliers or not: to find the subspaces (subsets of features) in which observations are outliers. Their outlying subspace detection method is called High-Dimension Outlying Subspace Detection (HighDOD). It is important to note that despite *k*-NN being a simple and relatively old approach to novelty detection, it is still widely used. In a recent study (Ding, Li, Belatreche, & Maguire, 2014), *k*-NN outperformed three novelty detection techniques, including the state of the art SVDD, on various real-life data sets.

When a density-based approach is used, novelty scores are proportional to the inverse of the relative densities. Typically, novelty scores have been calculated as the number of nearest neighbors within a neighborhood of the testing observations (Knorr & Ng, 1997), (Knorr, Ng, & Tucakov, 2000). Another density-based procedure to calculate the novelty score using kernels was introduced recently: the *summation kernel similarity score* (SKSS) (Ramirez-Padron, Foregger, Manuel, Georgiopoulos, & Mederos, 2010). Essentially, the SKSS value of a testing observation **x** is the sum of the similarities between **x** and all its neighbors within a ball of fixed

radius *p*. SKSS was proposed within the geometric framework for kernel novelty detection introduced in (Eskin, Arnold, Prerau, Portnoy, & Stolfo, 2002). That geometric framework involves the use of kernel functions to map input data to very high-dimensional feature spaces where current distance-based novelty detectors could be applied. The main assumption behind the introduction of kernel methods in this case is that outliers might be better detected in the high-dimensional feature space associated with the kernel function.

We wrap up this section by listing other representative nearest neighbor-based methods: Local Outlier Factor (LOF) (Breunig, Kriegel, Ng, & Sander, 2000); Connectivity-based Outlier Factor (COF) (Tang, Chen, Fu, & Cheung, 2002); LOCI (Papadimitriou, Kitagawa, Gibbons, & Faloutsos, 2002), which can find anomalous micro-clusters besides individual outliers; and the Local Distance-based Outlier Factor (LDOF) method (Zhang, Hutter, & Jin, 2009), which was devised to work on scattered data sets.

1.4.5 Information Theoretic Novelty Detection

Information theoretic methods define as outliers those observations having the highest impact on the information content of the data set. The main assumption behind these methods is that outliers have a much higher impact on the information content of a data set than observations from the normal class. The basic technique of theoretic novelty detection is to find Pareto-optimal solutions (Deb, 2005) to a dual-objective optimization problem. Given a data set *D*, the problem consists in finding the minimal subset of instances *I*, such that $C(D) - C(D - I)$ is maximum, where C denotes an information theoretic measure. The observations in the subset *I* are considered outliers.

Among the most commonly used measures are *Kolmogorov complexity*, *entropy*, *relative entropy*, *conditional entropy*, and *relative conditional entropy*; e.g. (Arning, Agrawal, & Raghavan, 1996), (Lee & Xiang, 2001), (Keogh, Lonardi, & Ratanamahatana, 2004), (Lakhina, Crovella, & Diot, 2005), (Gu, Fogla, Dagon, Lee, & Škorić, 2006), (Ando, 2007), (Afgani, Sinanovic, & Haas, 2010). The information theoretic approach has been applied to data sets with related observations; for instance, sequential data (Arning, Agrawal, & Raghavan, 1996), (Lin, Keogh, Fu, & Van Herle, 2005); spatial data (Lin & Brown, 2006); and graph data (Noble & Cook, 2003).

1.4.6 Subspace-based Novelty Detection

Subspace-based novelty detection methods look for outliers in low-dimensional projections of the observations; under the assumption that outliers are easier to detect on low dimensional projections that encompass most of the variability in the data. This approach can be valuable when using high-dimensional data. In that case, all observations are typically distant from each other; to a point that differences between distances become irrelevant and the concept of neighborhood might not be useful anymore.

Principal Component Analysis (PCA) (Joliffe, 2002) has been commonly employed by outlier detection methods to obtain lower dimensional projections of the input data. Outliers can be found by looking for projections with high values along low-variance principal components (Parra, Deco, & Miesbach, 1996), (Dutta, Giannella, Borne, & Kargupta, 2007). Robust PCA (Huber & Ronchetti, 2009) has been employed as well; e.g. (Shyu, Chen, Sarinnapakorn, & Chang, 2003).

Kernel methods have been mentioned in this chapter as part of various approaches to novelty detection. Similarly, kernel methods have been devised within the subspace-based approach. The most notable example is the application of kernel PCA (KPCA) to novelty detection (Hoffmann, 2007). Essentially, training data are mapped through a kernel function into a very high-dimensional feature space, in which KPCA extracts the principal components. The novelty scores of data instances are calculated as the squared distances to the principal subspace (also called *squared reconstruction errors*). As an interesting follow-up, the work in (Li, Georgiopoulos, & Anagnostopoulos, 2011) computes the reconstruction error by projecting any test observation onto the orthogonal complement of the KPCA-generated principal subspace and subsequently calculating the Mahalanobis distance of that projection from the mean of all transformed training observations. This variant, called MD-based KPCA, showed about the same or better performance than one-class SVM and KPCA novelty detection on various real-life data sets.

It has been claimed that novelty detection techniques that use PCA display a degrading performance on high-dimensional input spaces containing low-relevance or noisy attributes. An alternative to PCA when prior class information is available has been offered in (Sofman, Bagnell, & Stentz, 2010). It uses Multiple Discriminant Analysis (MDA) to obtain the low-dimensional subspace. Interestingly, that work employs an online detection algorithm that can be seen as a particular case of a kernel online learning technique called NORMA (Kivinen, Smola, & Williamson, 2004). The NORMA algorithm will be described in a following chapter, among other kernel-based online learning techniques.

1.4.7 Angle-based Novelty Detection

As mentioned above, the concepts of distance and neighborhood become irrelevant when working with data in high dimensions. Assuming outliers are located at the borders of the distribution generating the normal class, the angle-based approach relies on a property that remains consistent even for high-dimensional data: if an observation is an outlier then most other objects in the data set will be located in similar directions from it (that is clearly not the case for most normal observations). A prime example of this approach is the work of (Kriegel, Schubert, & Zimek, 2008), where, for each observation, the spectrum of the angles to all other observations is obtained. Subsequently, an "outlierness" score is obtained for each data point based on how broad its spectrum is.

1.5 Advantages and Limitations of Modern Approaches

Statistical techniques for novelty detection have a variable computational complexity, depending on the type of statistical model employed. Typically, they are either linear or quadratic with respect to the number of observations. An advantage of these techniques is that novelty scores are usually related to confidence intervals, offering a statistical support to decision making based on the scores. Additionally, the use of robust statistics allows the application of this approach to data sets containing incorrect labels, as far as the training data is not extremely contaminated with outliers. Except in some well-known domains, the non-parametric approach should be preferred over the parametric approach. There are two main reasons behind this statement: First, for some data sets it might be very difficult to find a suitable known distribution.

Second: implementing hypothesis tests for complex distributions might be very difficult, computationally expensive, or both.

In general, statistical methods have some limitations that need to be considered as well. For instance, many methods consider attributes independently, preventing the detection of outliers that have common individual values for their attributes but a rare combination of values for two or more attributes. Additionally, virtually all statistical techniques become computationally expensive and inefficient when used on high-dimensional data sets.

Classification-based techniques benefit from powerful and thoroughly studied algorithms, sometimes with guaranteed convergence properties. Additionally, they typically have a fast response when evaluating new observations. Multi-class techniques are limited to supervised learning scenarios because labels are needed for the different normal classes. However, one-class classification algorithms can be employed in most practical situations when labels are available for a single normal class or even not present at all, e.g. (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001), (Roth, 2006). Classification-based techniques tend to be most effective when labels are available. However, it is important to keep in mind that the distribution of normal and abnormal labels are typically imbalanced, making the learning task more difficult compared to standard classification problems.

Although several classifiers only provide a binary novelty score, there are classification-based methods that provide novelty scores in a wide range of values, e.g. (Platt J. , 2000). Regarding computational complexity, the training algorithms of classifiers involving quadratic optimization, like SVM, can be slow on large data sets. However, efforts have been made in

order to improve their computational complexity. For instance, SVM training algorithms have achieved significant improvements on their time complexity, like Sequential Minimal Optimization (Platt J. C., 1999), (Keerthi, Shevade, Bhattacharyya, & Murthy, 2001), and linear time SVM (Joachims, 2006). These relatively advantageous characteristics of the classification-based approach to novelty detection make them particularly attractive for many real-life applications.

Clustering-based algorithms can have a quadratic or subquadratic training time complexity. However, their test phase is typically fast because new observations are compared only to representatives of a small quantity of clusters. Similarly to distance-based methods, clustering-based methods can work in unsupervised learning scenarios, and adapting them to different data types is relatively straightforward: by employing a clustering algorithm that can handle the new data type.

The main limitations of clustering-based methods are the following: Some clustering methods are not necessarily robust to the presence of outliers in the training data. Consequently, a relatively high quantity of outliers might not be detected. Furthermore, this limitation can be stated as follows: clustering-based methods typically detect outliers as a byproduct of the clustering process, and hence some might be unfit for novelty detection. A second limitation of clustering techniques is that many of them are not effective when the data contain small clusters of anomalies. As mentioned in section 1.4.3, this drawback can be alleviated by setting a size/density threshold constraint on the clusters obtained in the training phase. Finally, similarly

to distance-based novelty detection methods, clustering-based techniques are very sensitive to the curse of dimensionality.

Straightforward implementations of **distance-based methods** have at least a quadratic complexity, because of the computation of pairwise distances. However, some approaches have been proposed to obtain subquadratic time complexities. One of those approaches obtains a representative subset of the data set, called *outlier detection solving set*, which can be used to detect outliers faster while maintaining a prediction quality comparable to quadratic techniques (Angiulli, Basta, & Pizzuti, 2006). In the case of nearest neighbor-based methods, efficient data structures like *k*-d trees (Bentley J. , 1980) and R-trees (Sellis, Roussopoulos, & Faloutsos, 1987) have been used to efficiently locate nearest neighbors; e.g., (Roussopoulos, Kelley, & Vincent, 1995), (Yershova & LaValle, 2007), (Yen, Shih, Chang, & Li, 2010). Alternatively, observations can be grouped into regular (congruent and non-overlapping) regions of the attribute space to make nearest neighbor searching more efficient for large data sets, e.g. Elias methods (Rivest, 1974), (Cleary, 1979). Although both approaches are computationally efficient in the number of observations, unfortunately they do not scale well when the number of attributes are relatively high. In order to deal with the curse of dimensionality, further refinements have been proposed (Katayama & Satoh, 1997), (Hinneburg, Aggarwal, & Keim, 2000), (Tao, Yi, Sheng, & Kalnis, 2009). In general however, the ability of distance-based methods to differentiate between normal and anomalous data is strongly affected by high dimensionality.

Despite the limitations of distance-based algorithms, they have some advantages that make them attractive for novelty detection: These methods are unsupervised in nature, thereby fitting a wide

range of problem domains. Adapting a distance-based method to a particular data type is as straightforward as to define a distance function for that data type, whenever that is feasible. In contrast to many statistical methods, they do not require modeling the underlying distributions of the data. Finally, nearest neighbor methods can be more effective on semi-supervised learning scenarios (when labels for the normal data are available) than other more complex approaches.

Information theoretic-based methods have an exponential computational complexity when implemented to solve exactly the combinatorial dual-optimization problem from its definition. However, some approaches have offered scalable approximate solutions to that optimization problem (He, Deng, Xu, & Huang, 2006), (Ando, 2007). Similarly to clustering and distance-based methods, this approach is unsupervised in nature and no assumptions about underlying statistical distributions are needed. One of the major limitations of this approach is that defining novelty scores is not an easy task in the majority of cases. Additionally, information measures should be sensitive enough to detect a small percentage of outliers for the corresponding problem domain.

The computational complexity of the **subspace-based** approach to novelty detection varies with the type of projection technique employed. The most commonly used technique, PCA, is typically linear in the number of data instances and quadratic in the number of dimensions. However, some efforts have been made to improve on that complexity; e.g. a fast implementation of kernel PCA (Günter, Schraudolph, & Vishwanathan, 2007). Subspace-based methods can be applied in unsupervised learning scenarios, do not require prior knowledge of statistical distributions, and they are particularly devised to tackle the curse of dimensionality.

However, they assume that outliers can be distinguished from normal observations when data is projected into lower dimensional spaces. In many domains it is not easy or even possible to guarantee the veracity of that assumption.

The **angle-based** approach will work only if outliers are located at the borders of the data distribution and members of the normal class are grouped around some center area. Still, it can be a valuable alternative (or a complement) to subspace-based methods when working with high-dimensional data.

CHAPTER 2: STATE OF THE ART IN KERNEL NOVELTY DETECTION

This chapter reviews methods that are currently considered the state of the art in kernel-based novelty detection. Both batch and online approaches are considered. Kernel methods for pattern analysis have several advantages over other methods described in the previous chapter. These advantageous properties were the reason to select kernel methods as the theoretical framework for this dissertation. As an introduction to this chapter, some relevant advantages of kernel methods are summarized below.

The majority of novelty detection algorithms are limited to numerical data. They leverage multiple well-established techniques and theories. However, there is an increasing interest in working on data that have non-numerical attributes. As a result of that interest, several novelty detection algorithms have been proposed to build models from non-numerical data (Pimentel, Clifton, Clifton, & Tarassenko, 2014), (Chandola, Banerjee, & Kumar, 2009). Kernels methods have given researchers the capability to deal with multiple data types within a single framework, including complex data such as images, videos, DNA sequences and graphs (Shawe-Taylor & Cristianini, 2004). Consequently, kernel methods have blurred the classic distinction between statistical and syntactical pattern analysis.

Classic linear techniques for pattern analysis are computationally efficient and rely on well-studied mathematical properties. However, they do not generalize as well as non-linear techniques, like neural networks. On the other hand, most non-linear techniques don't rely on theoretical foundations as strong as those from linear models. Thanks to the "kernel trick", kernel methods offer the best from both worlds: the generalization capacity of non-linear techniques

and the theoretical advantages of well-established linear techniques (which are applied in the feature space). Several works have proposed the "kernelization" of different classic outlier detection algorithms (Eskin, Arnold, Prerau, Portnoy, & Stolfo, 2002), (Roth, 2006), (Latecki, Lazarevic, & Pokrajac, 2007), (Shen, 2007), (Oh & Gao, 2009).

Most kernel methods for novelty detection follow a classification-based approach. Consequently, classification-based kernel methods for novelty detection constitute a baseline to which new novelty detection algorithms should compare to; as done, for example, by the authors of GP-based novelty detection (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Despite their successful applications, most kernel methods for novelty detection have some limitations. Two of the most relevant limitations are: (1) there is no straightforward way of introducing prior information into the models; and (2) classification is provided only as point estimates of the unknown variables, without estimating the corresponding uncertainty. It is well-known that Bayesian learning techniques provide a practical approach to introducing prior information into models, through the use of prior distributions. Additionally, Bayesian modeling offers not only point estimates of unknown variables and parameters, but they also estimate the uncertainty associated to predictions. Recently, kernel methods have been used into the Bayesian nonparametric framework with great success (Rasmussen & Williams, 2006). The main idea behind this approach is to build a data-driven model employing Gaussian processes (GPs), using kernels as prior covariance functions. GPs have been applied almost exclusively for regression and classification problems. However, a few recent papers describe the use of GPs for novelty detection as well. Experimental comparisons described in (Kemmler M. , Rodner, Wacker, &

Denzler, 2013) have shown the performance advantages of batch GPs over state-of-the-art classification-based kernel methods, including the successful SVDD method.

This rest of this chapter is structured as follows: Section 2.1 offers an introduction to pattern analysis and kernel functions, including mathematical properties underlying the different kernel methods considered in subsequent sections. Section 2.2 describes some representative classification-based kernel methods for novelty detection. Both batch learning and online learning state-of-the-art methods are considered. Section 2.3 introduces GPs and describes their typical usage in machine learning, with emphasis in novelty detection as presented in (Kemmler M. , Rodner, Wacker, & Denzler, 2013).

2.1 Statistical Patterns and Kernel Methods

This section introduces several fundamental concepts underlying kernel methods. Kernel-based learning methods were introduced in machine learning to obtain non-linear patterns while relying essentially on well-established linear techniques. This section summarizes several advantages associated to this approach. To set the stage, let us assume that we have a training data set $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \epsilon \mathcal{X}, i = 1, \dots, N\}$, where $\mathcal{X}$ denotes a finite-dimensional domain. Additionally, we might have a corresponding set of labels $\mathbf{Y} = \{\mathrm{y}_i | \mathrm{y}_i \epsilon \mathbb{R}, i = 1, .., N\}$. When labels are available for all observations in $\mathbf{X}$, it is said that the learning problem is supervised, and we learn from a set of data points $(\mathbf{x}_i, \mathrm{y}_i)$. If labels were not present then we have an unsupervised learning problem. Typically for supervised learning, if the y_i labels took values in a finite subset of $\mathbb{R}$ then they denote the class of the corresponding observation $\mathbf{x}_i$. The case of labels taking values in an infinite set corresponds to a regression problem. In this dissertation, we denote the training

data by the symbol D. Generally, we specify whether the training data contain labels or not, writing $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) : \mathbf{x}_i \in \mathcal{X},\ \mathrm{y}_i \in \mathbb{R}\}$ and $D = \mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathcal{X}\}$, respectively. When a statement that involves the training data is applicable to both cases, as done in the following subsection, the training data is specified here as $D = \{\mathbf{z}_i :\ \mathbf{z}_i \in \mathcal{Z},\ i = 1, \dots, N\}$, where $\mathcal{Z}$ could be either $\mathcal{X}$ or $\mathcal{X} \cup \mathbb{R}$.

2.1.1 Statistical Patterns

The training data are assumed here to be independently and identically distributed (i.i.d.) according to some unknown probability measure. Under this assumption, the main goal of any pattern analysis method is to extract *general statistical patterns* from the training data, which can be defined as follows (Shawe-Taylor & Cristianini, 2004):

Definition: A general statistical pattern for a data set $D = \{\mathbf{z}_i :\ \mathbf{z}_i \in \mathcal{Z},\ i = 1, \dots, N\}$ with independently and identically distributed (i.i.d.) observations that are generated according to a probability distribution *P*, is a non-trivial non-negative function l that satisfies:

$$E_{\mathcal{Z}}[l(\mathbf{z})] \approx 0, \qquad (2.1)$$

where $E_{\mathcal{Z}}[l]$ denotes the expectation of the function *l* under the distribution of the training data.

As an example of a statistical pattern, consider the classic regression problem on a training data set $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) : \mathbf{x}_i \in \mathcal{X},\ \mathrm{y}_i \in \mathbb{R}\}$. In this case, a statistical pattern is defined as a loss function $l(\mathbf{x}, \mathrm{y}) = \mathcal{L}(g(\mathbf{x}), \mathrm{y})$, where g denotes the linear prediction function. The loss function $l(\mathbf{x}, \mathrm{y})$ measures the discrepancy between $g(\mathbf{x})$ and the correct label y; and consequently $l(\mathbf{x}, \mathrm{y})$ will be close to zero when evaluated in training observations that fit the pattern. As a second example consider the novelty detection problem. The training data in this case consist of observations from a space $\mathcal{X}$ that are labeled as members of the normal class, i.e. $D = \{\boldsymbol{X}, \mathbf{y}\} =$

$\{(\mathbf{x}_i, \mathrm{y}_i) \mid \mathbf{x}_i \epsilon \mathcal{X},\ \mathrm{y}_i = 1,\ i = 1, \ldots, N\}$. An appropriate pattern in this case would be a non-negative function $l: \mathcal{X} \to \mathbb{R}$ such that $l(\mathbf{x}) \approx 0$ for observations generated from the (generally unknown) distribution of the normal class. On the other hand, $l(\mathbf{x})$ should noticeably deviate from zero when evaluated on observations that are very different to the majority of observations in the training data. This statistical pattern $l(\mathbf{x})$ can be leveraged to define membership scores (or novelty scores) that help us to estimate whether a new observation belongs to the normal class or can be considered an outlier.

Based on the above definition of a statistical pattern, a *pattern analysis algorithm* is defined in (Shawe-Taylor & Cristianini, 2004) as an algorithm that, given a finite training data set D, its output is either a statistical pattern or an indication that no patterns were detected in D. One of the most important properties of a pattern analysis algorithm is to be *statistically stable*. Informally, this property denotes the fact that any pattern found actually resembles a characteristic of the data source instead of being obtained by chance. When a pattern is statistically stable, it should be obtained from different samples of the same data source. Of course, no algorithm can absolutely guarantee the stability of a pattern. For that reason, some probabilistic results have been derived that allow researchers to state their confidence in the output of a pattern analysis algorithm. The Rademacher complexity theory plays an important role in that sense (Bartlett & Mendelson, 2002), (Koltchinskii & Panchenko, 2000).

2.1.2 Kernel Functions for Pattern Analysis

Kernel functions were introduced in machine learning as a way of finding complex non-linear patterns in data sets of arbitrary data types through the application of linear methods to a

representation of the data in a high-dimensional inner product space, which is commonly called the *feature space* and it is denoted here by $\mathcal{F}$. Kernels allow mapping input data points to a feature space $\mathcal{F}$ without explicitly using the mapped feature vectors. The feature space must be a vector space, and it usually has a much higher dimension than the input space $\mathcal{X}$ (including infinite dimensions). This approach allows building complex non-linear discrimination surfaces in the original input space. The following definition establishes what a kernel function is (Shawe-Taylor & Cristianini, 2004):

Definition: Given a Hilbert space $\mathcal{F}$, and an arbitrary space $\mathcal{X}$, a function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function if $\forall\, \mathbf{x}, \mathbf{x}' \in \mathcal{X},\ k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, for some mapping function $\phi: \mathcal{X} \to \mathcal{F}$.

The kernel approach to pattern analysis entails using a kernel function k to map training observations $\mathbf{x}_i$ to $\mathcal{F}$ using the mapping function $\phi: \mathcal{X} \to \mathcal{F}$. Subsequently, a linear model is learned on the transformed data in $\mathcal{F}$. The linear model thus obtained is re-interpreted as a likely non-linear pattern in the original input space. This is a common process in several areas of mathematics, where data from a given space where a problem is difficult to solve are transformed into another space where a feasible well-known technique can be applied. The corresponding solution is then interpreted back into the original space. What makes kernel methods particularly special is that this process can be applied efficiently because of two very important characteristics. First, the algorithms to be "kernelized" rely exclusively on inner products in $\mathcal{F}$. Second, those inner products can be calculated directly from the observations in the input space by using the corresponding kernel function. Consequently, typically there is no need to obtain an explicit expression for the mapping function ϕ or the coordinates of the mapped observations in $\mathcal{F}$.

It is useful to think of kernel functions as similarity measures between any two observations. This interpretation is justified by considering the well-known geometric interpretation of an inner product. Another important aspect of kernel methods is that the particular algorithm to be applied becomes independent of the data type of the input space. Training data can be mapped to a feature space as far as an appropriate kernel function is defined for the corresponding data type. Consequently, another benefit of kernel methods is that non-linear patterns can be found for training data containing non-numerical attributes. Table 2.1 shows commonly used kernels that apply to numerical data. More specialized kernels are described in (Shawe-Taylor & Cristianini, 2004).

Table 2.1: Commonly used kernel functions. In this case data points $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, where d is a positive integer.

Kernel	**Kernel name (Equivalent model)**
$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$	Linear kernel (Linear classifier)
$k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^q$	Polynomial kernel (Polynomial of degree q)
$k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$	Gaussian kernel (Gaussian radial basis function network)
$k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{1}{2}\sum_{i=1}^{d} a_i(\mathbf{x_i} - \mathbf{x}_i')^2\right)$	Simple exponential kernel
$k(\mathbf{x}, \mathbf{x}') = tanh(\langle \mathbf{x}, \mathbf{x}' \rangle - \theta)$	Sigmoid kernel (Multi-Layer Perceptron with one hidden layer)

Considering that training data are always finite, the kernel function has a matrix expression associated to the data. The following definition relates the concepts of kernel function and kernel matrix:

Definition: Given a kernel function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ and a set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N | \, \mathbf{x}_i \in \mathcal{X}\}$, the corresponding kernel matrix $\boldsymbol{K}$ is defined as the $N \times N$ real matrix such that $\boldsymbol{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

A natural question at this point is what functions k are feasible kernels for using in a kernel method. It turns out that a kernel function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ should fulfill the characterization that appears below. Note that it implies that $k(\mathbf{x}, \mathbf{x}) \geq 0$, $\forall \mathbf{x} \in \mathcal{X}$.

Characterization: A symmetric function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a positive semi-definite kernel function if and only if for any positive integer N, any choice of objects $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathcal{X}$ and any choice of real numbers $c_1, c_2, \ldots, c_N$, the resulting N x N kernel matrix $\boldsymbol{K}$ is symmetric and satisfies that $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \boldsymbol{K}_{i,j} \geq 0$ (i.e. $\boldsymbol{K}$ is positive semi-definite[1]).

2.1.3 Kernel transformations

Choosing a kernel for a particular problem reflects most of our prior knowledge about the data source and the problem (actually, the only knowledge about the data that is not included in the kernel is the set of labels in supervised and semi-supervised problems). Consequently, operations on kernel matrices might represent important changes regarding our understanding of the data. The following results allow the creation and combination of kernel functions, providing the means to integrate prior knowledge into the kernel matrices (Shawe-Taylor & Cristianini, 2004):

[1] The expression "positive semi-definite" is used here as defined in Matrix Theory, i.e. $\mathbf{x}^T K \mathbf{x} \geq 0$ for all $\mathbf{x} \in R^m$. Please, note that in (Scholkpof and Smola, 2002) the term "positive definite" is used instead with the same meaning.

Let k_1 and k_2 be kernel functions defined over $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^n$. Additionally, $f: \mathcal{X} \to \mathbb{R}$, $\phi: \mathcal{X} \to \mathbb{R}^m$, and k_3 is a kernel function defined over $\mathbb{R}^m \times \mathbb{R}^m$. The following functions are kernels:

(i) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$

(ii) $k(\mathbf{x}, \mathbf{z}) = ak_1(\mathbf{x}, \mathbf{z})$; where $a \in \mathbb{R}^+$

(iii) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$

(iv) $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$

(v) $k(\mathbf{x}, \mathbf{z}) = k_3\big(\phi(\mathbf{x}), \phi(\mathbf{z})\big)$

(vi) $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T B\mathbf{z}$; where *B* is a symmetric positive semi-definite matrix.

It is important to consider how feature spaces are transformed by some of these operations. For instance, the new feature vectors $\phi(\mathbf{x})$ obtained through construct (i) satisfies $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x})]$, where $\phi_i(\mathbf{x})$ denotes a feature vector from kernel k_i; construct (ii) re-scales the vectors in the feature space by $\sqrt{a}$; and construct (iv) defines a one-dimensional feature space through function *f*.

Besides creating new kernels using the previous constructs, we can also benefit from some operations on current kernel matrices to achieve certain transformations on the feature space. Some simple transformations are listed below, assuming that k_1 is a kernel function:

- The function $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + a$, with $a \in \mathbb{R}^+$, is a kernel function having a feature space equal to the feature space of *k* with a constant-valued dimension added to it.
- The function

$$k(\mathbf{x},\mathbf{z}) = \begin{cases} k_1(\mathbf{x},\mathbf{z}) + a & if\ \mathbf{x} = \mathbf{z} \\ k_1(\mathbf{x},\mathbf{z}) & otherwise \end{cases}, \qquad (2.2)$$

where $a \in \mathbb{R}^+$, corresponds to adding a new feature with different values to the feature space associated to the kernel function k_1.

- The kernel

$$k(\mathbf{x},\mathbf{z}) = \frac{k_1(\mathbf{x},\mathbf{z})}{\sqrt{k_1(\mathbf{x},\mathbf{x})k_1(\mathbf{z},\mathbf{z})}} \qquad (2.3)$$

corresponds to a normalization of all vectors in the feature space of k_1, effectively mapping all observations to a hyper-sphere.

As a final note, it is possible to assess how different two kernel matrices are by defining a similarity measure. A simple similarity measure is the alignment between two kernels:

Definition: Let $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ be two kernel matrices of dimension $N \times N$. The alignment $A(\boldsymbol{K}_1, \boldsymbol{K}_2)$ is defined as

$$A(\boldsymbol{K}_1, \boldsymbol{K}_2) = \frac{\langle \boldsymbol{K}_1, \boldsymbol{K}_2 \rangle}{\sqrt{\langle \boldsymbol{K}_1, \boldsymbol{K}_1 \rangle \langle \boldsymbol{K}_2, \boldsymbol{K}_2 \rangle}}, \qquad (2.4)$$

where $\langle \boldsymbol{K}_i, \boldsymbol{K}_j \rangle = trace\left(\boldsymbol{K}_i^T \boldsymbol{K}_j\right)$ is the Frobenius inner product between the two matrices.

From this definition, it follows that the alignment between two kernel matrices corresponds to the cosine of the angle between the two matrices taken as N^2-dimensional vectors.

2.1.4 Classification of Kernels

Kernels functions have been categorized based on various characteristics, such as whether they are local kernels or not, separable or non-separable, stationary or non-stationary, among other

characteristics (Genton, 2001). The following class of kernels is of particular importance to leveraging kernels in a nearest-neighbor method:

Definition: A kernel function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is isotropic stationary if there is a function $g_k: \mathbb{R} \to \mathbb{R}$ such that $k(\mathbf{x}, \mathbf{z}) = g_k(\|\mathbf{x} - \mathbf{z}\|)$.

Isotropic stationary kernels are invariant to rotations and translations. The well-known Gaussian RBF kernel is an example of an isotropic stationary kernel. On the other hand, the polynomial kernel of degree *d* and the linear kernel are examples of non-isotropic stationary kernels. The previous definition is limited to input spaces where a norm is defined. A generalization to the class of isotropic stationary kernels that considers input spaces with arbitrary data types, called *similarity kernel*, was introduced in (Ramirez-Padron, Foregger, Manuel, Georgiopoulos, & Mederos, 2010):

Definition: A positive semi-definite kernel function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a similarity kernel if and only if there exist $c \in \mathbb{R}^+$ such that $\forall \mathbf{x} \in \mathcal{X}, k(\mathbf{x}, \mathbf{x}) = c$.

As noted in (Ramirez-Padron, Foregger, Manuel, Georgiopoulos, & Mederos, 2010), a similarity kernel *k* can be interpreted as a similarity measure in $\mathcal{X}$ that fulfills the following properties:

- Symmetry (by definition of kernel function).
- $\forall \mathbf{x} \in \mathcal{X},\ k(\mathbf{x}, \mathbf{x}) = c$ (by definition of similarity kernel).
- $\forall \mathbf{x}, \mathbf{z} \in \mathcal{X},\ k(\mathbf{x}, \mathbf{z}) \leq c$.
- $\exists d \in \mathbb{R}$ such that $\forall \mathbf{x}, \mathbf{z} \in \mathcal{X},\ k(\mathbf{x}, \mathbf{z}) \geq d$.

The previous properties are particularly important for nearest neighbor-based outlier detection methods that use kernels.

2.1.5 Properties of Data in Feature Spaces

Despite the absence of an explicit representation for the projection $\phi(\mathbf{x})$, a kernel function grants us access to several properties of data projected into a kernel-defined feature space. The following well-known properties are described in this subsection: the norm of a feature vector, the distance between feature vectors, characteristics of the center of mass of a set of feature vectors, and the variance of the norm of projections in the feature space (Shawe-Taylor & Cristianini, 2004).

The norm of a feature vector is obtained directly from the properties of inner products:

$$\|\phi(\mathbf{x})\|_2 = \sqrt{\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle} = \sqrt{k(\mathbf{x}, \mathbf{x})} \,. \qquad (2.5)$$

Similarly, the distance between two feature vectors can be easily calculated as:

$$\|\phi(\mathbf{x}) - \phi(\mathbf{z})\| = \sqrt{\langle \phi(\mathbf{x}) - \phi(\mathbf{z}), \phi(\mathbf{x}) - \phi(\mathbf{z}) \rangle}$$

$$= \sqrt{k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{z}) + k(\mathbf{z}, \mathbf{z})} \,. \qquad (2.6)$$

As before, let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ be a random sample from a domain space $\mathcal{X}$. Let us denote by $\phi(\mathbf{X})$ the image of $\boldsymbol{X}$ under ϕ (i.e. $\phi(\mathbf{X}) = \{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)\}$). The center of mass of $\phi(\mathbf{X})$ is defined as the sample mean of the feature vectors in $\phi(\mathbf{X})$:

$$\bar{\phi}_s = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i) \,. \qquad (2.7)$$

Given that the feature vectors in $\phi(\mathbf{X})$ are typically infinite-dimensional, it is not possible in general to have an explicit representation for the vector $\bar{\phi}_s$. However, the norm of $\bar{\phi}_s$ and its distance from other feature vectors are measurable quantities:

$$\left\|\bar{\phi}_s\right\|^2 = \langle \bar{\phi}_s, \bar{\phi}_s \rangle = \frac{1}{N^2}\sum_{i,j=1}^{N} k\left(\mathbf{x}_i, \mathbf{x}_j\right), \qquad (2.8)$$

$$\left\|\phi(\mathbf{x}) - \bar{\phi}_s\right\|^2 = k(\mathbf{x},\mathbf{x}) + \frac{1}{N^2}\sum_{i,j=1}^{N} k\left(\mathbf{x}_i, \mathbf{x}_j\right) - \frac{2}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}). \qquad (2.9)$$

Translating the origin of the feature space to the center of mass $\bar{\phi}_s$ corresponds to minimizing the sum of the squared norms of the feature vectors, which in turn implies a minimization of the trace of the kernel matrix (Shawe-Taylor & Cristianini, 2004). Consequently, centering the feature data can be expressed through the following kernel transformation, where k_c denotes the kernel function corresponding to the centered data and k denotes the original kernel function:

$$k_c(\mathbf{x},\mathbf{z}) = k(\mathbf{x},\mathbf{z}) + \frac{1}{N^2}\sum_{i,j=1}^{N} k\left(\mathbf{x}_i, \mathbf{x}_j\right) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{z}). \qquad (2.10)$$

The last property considered here is the variance of the norm of projections within the feature space. For that matter, let us assume that the feature data has zero mean (this can be achieved through the centering procedure formulated above). Let us denote by $\mathbf{\Phi}$ the matrix containing the feature vectors:

$$\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]^T. \qquad (2.11)$$

From classic statistics, the covariance matrix $\boldsymbol{C}$ of feature data can be written as $\boldsymbol{C} = \frac{1}{N}\mathbf{\Phi}^T\mathbf{\Phi}$. Let us denote by $\mathbf{v}$ a unit vector in the feature space. The projection $P_v\left(\phi(\mathbf{x})\right)$ of a vector $\phi(\mathbf{x})$ onto $\mathbf{v}$ is expressed as:

$$P_{\mathbf{v}}\left(\phi(\mathbf{x})\right) = \frac{\langle \mathbf{v}, \phi(\mathbf{x}) \rangle}{\|\mathbf{v}\|^2}\mathbf{v} = \langle \mathbf{v}, \phi(\mathbf{x}) \rangle \mathbf{v}. \qquad (2.12)$$

Consequently, $\left\|P_{\mathbf{v}}\left(\phi(\mathbf{x})\right)\right\| = \langle \mathbf{v}, \phi(\mathbf{x}) \rangle$. Given that the feature data have zero mean, it is obtained that the expected value of the norm of the projections $\mu_{\mathbf{v}} = \hat{E}\left[\left\|P_{\mathbf{v}}\left(\phi(\mathbf{x})\right)\right\|\right] = 0$. The variance of the norms of the projections onto $\mathbf{v}$ is expressed as follows (Shawe-Taylor & Cristianini, 2004):

$$\sigma_{\mathbf{v}}^2 = var\left(\left\|P_{\mathbf{v}}\left(\phi(\mathbf{x})\right)\right\|\right) = \frac{1}{N}\mathbf{v}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{v} = \mathbf{v}^T\boldsymbol{C}\mathbf{v}\,. \qquad (2.13)$$

There is no explicit expression for σ_v^2 for a general vector $\mathbf{v}$. However, if we assume $\mathbf{v} = \boldsymbol{\Phi}^T\boldsymbol{\alpha}$, an expression for it can be obtained as $\sigma_{\mathbf{v}}^2 = \frac{1}{N}\boldsymbol{\alpha}^T\boldsymbol{K}^2\boldsymbol{\alpha}$.

2.2 Classification-based Kernel Methods for Novelty Detection

Classification-based kernel methods for novelty detection take advantage of good generalization properties from statistical learning theory and the possibility of dealing with infinite dimensional feature spaces. Recently, they have been applied successfully in a variety of domains and their performance compares favorably to other methods currently used for novelty detection (Gardner, Krieger, Vachtsevanos, & Litt, 2006) (Liu, Liu, & Chen, 2010) (Blanchard, Lee, & Scott, 2010), (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Consequently, they can be considered state-of-the-art methods for novelty detection. New approaches or algorithms aiming at improving the effectiveness of novelty detection methods should be compared to one or more of these classification-based kernel methods. This section describes some of the most successful kernel-based methods. One subsection is devoted to batch methods and a second subsection describes current online methods.

2.2.1 Batch methods

One-class SVM (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001)

The goal of one-class SVM is to rely on the training sample to estimate the support of the corresponding distribution (i.e. the set of data points for which the density function is not zero-valued). Consequently, the model from one-class SVM is essentially a binary function *f* that specifies regions containing most of the data from the normal class. The function *f* should return 1 in a relatively small region that contains most of the observations, and -1 elsewhere. One-class SVM is formulated in the feature space $\mathcal{F}$ associated to a kernel function *k*. It looks for the hyperplane that better separates the feature vectors from the origin with maximum margin. A test observation $\boldsymbol{z}\epsilon\mathcal{X}$ will be declared a member of the normal class if the projection of $\mathbf{z}$ lies on the side of the optimal hyperplane facing most of the mapped training data; i.e. if $f(\boldsymbol{z}) = 1$.

Given an unsupervised training data set $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \epsilon \mathcal{X}, i = 1, \dots, N\}$, where $\mathcal{X}$ denotes a finite-dimensional domain, finding the separating hyperplane with optimal margin is stated as a quadratic optimization problem:

$$\min_{\mathbf{w}\in\mathcal{F},\xi\in\mathbb{R}^N,\rho\in\mathbb{R}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\upsilon N}\sum_i \xi_i - \rho \,, \qquad (2.14)$$

$$\text{subject to:} \quad \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle \geq \rho - \xi_i, \qquad \xi_i \geq 0 \,,$$

where υ denotes the expected rate of outliers in the data. The decision function *f* has the following expression:

$$f(\mathbf{x}) = sgn(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle - \rho) \,. \qquad (2.15)$$

The function f will be positive for most of the observations in the training data set, since the slack variables ξ_i are penalized in the quadratic problem. Introducing Lagrange multipliers $\alpha_i, \beta_i \geq 0$ the following Lagrangian is obtained:

$$L(\mathbf{w}, \boldsymbol{\xi}, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\upsilon N}\sum_i \xi_i - \rho - \sum_i \alpha_i[\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle - \rho + \xi_i] - \sum_i \beta_i \xi_i \ . \quad (2.16)$$

Equating to zero the derivatives of L with respect to the primal variables $\mathbf{w}$, $\boldsymbol{\xi}$, and ρ, the following conditions are found:

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i) \ , \quad (2.17)$$

$$\alpha_i = \frac{1}{\upsilon N} - \beta_i \leq \frac{1}{\upsilon N} \ , \quad (2.18)$$

$$\sum_i \alpha_i = 1 \ . \quad (2.19)$$

Substituting these conditions into L, a dual optimization problem is obtained:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \ , \quad (2.20)$$

$$\text{subject to:} \quad \sum_i \alpha_i = 1; \ \ 0 \leq \alpha_i \leq \frac{1}{\upsilon N}$$

The values for the α_i Lagrange parameters are obtained by solving the above dual optimization problem. As in the standard SVM method, those observations for which $\alpha_i > 0$ are called *support vectors*. Given that $\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$, the decision function is re-written as:

$$f(\mathbf{x}) = sgn(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho) \ . \quad (2.21)$$

The only parameter pending for estimation is ρ, which is calculated using the following expression:

$$\rho = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_k) \ , \quad (2.22)$$

where $\mathbf{x}_k$ is any support vector such that $0 < \alpha_k < \frac{1}{\upsilon N}$. Having any test observation $\boldsymbol{z}\epsilon\mathcal{X}$, it can be declared an outlier if $f(\boldsymbol{z}) = -1$, i.e. if $\mathbf{z}$ lies outside of the region containing most of the training data.

One-class SVM has been successfully applied to detecting epochs containing seizure activity from one-second windows of intracranial EEG (Gardner, Krieger, Vachtsevanos, & Litt, 2006). In (Clifton, Yin, Clifton, & Zhang, 2007), one-class SVM is leveraged to predict combustion instability from time-series data. The work of (Rabaoui, Kadri, & Ellouze, 2008) applies one-class SVM to detect events in continuous audio streams, reporting substantial improvements in performance compared to other popular approaches. One-class SVM has been also employed successfully to recognize physiological deterioration in patients under continuous monitoring (Clifton, Clifton, Watkinson, & Tarassenko, 2011). Other examples of applications of one-class SVM are listed in section 1.4.2.

Recently, an extension to one-class SVM, called one class-SVM with minimum within-class scatter (OC-WCSSVM) has been proposed; aimed at finding a more effective hyperplane using information of the scatter within the training data (An, Liang, & Liu, 2014). The corresponding experimental results showed improvements when compared to other modern algorithms on multiple real-world data sets. Another very recent extension is presented in (Khan, Ksantini, Ahmad, & Guan, 2014), where the low-variance directions of the data are taken into account to detect outliers. This method employs the estimated covariance matrix of the training data to control the direction of the separating hyperplane.

Support Vector Domain Description (SVDD) (Tax & Duin, 1999)

The approach employed by SVDD is to find a hypersphere in the feature space $\mathcal{F}$ that contains most of the observations in the training data set. This hypersphere would serve as the *data domain description* for the normal class represented by the training data set. It translates into a region covering most of the training points when mapped back into the input space $\mathcal{X}$. Observations lying outside of that region are considered outliers.

Given the training data set $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \epsilon \mathcal{X}, i = 1, \dots, N\}$, the SVDD problem consists of finding the smallest hypersphere in the feature space that contains most of the feature vectors $\{\phi_1, \phi_2, \dots, \phi_N\}$, where $\phi_i = \phi(\mathbf{x}_i)$. Essentially, the hypersphere should not contain data points lying far away from its center. Remote observations should be considered outliers and should not be included in the model. To accomplish this goal, non-negative slack variables $\boldsymbol{\xi} = \{\xi_1, \xi_2, \dots, \xi_N\}$ are introduced, to allow for some training observations to be outside of the hypersphere. The corresponding optimization problem, called *soft minimal hyper-sphere*, is stated as follows:

$$min_{\boldsymbol{c},r,\boldsymbol{\xi}} \quad r^2 + C\|\boldsymbol{\xi}\|_1 , \tag{2.23}$$

subject to:

$$\|\phi_i - \boldsymbol{c}\|^2 = (\phi_i - \boldsymbol{c})^T(\phi_i - \boldsymbol{c}) \leq r^2 + \xi_i, \quad i = 1,2, \dots, N;$$

$$\xi_i \geq 0, \quad i = 1,2, \dots, N;$$

where C is a regularization parameter, and $\mathbf{c}$ and r denote the center and the radius of the hypersphere, respectively. Introducing Lagrange multipliers $\alpha_i \geq 0$ and $\beta_i \geq 0$ the following Lagrangian is obtained:

$$L(\boldsymbol{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi}) = r^2 + C\|\boldsymbol{\xi}\|_1 + \sum_{i=1}^{N} \alpha_i [\|\phi_i - \boldsymbol{c}\|^2 - r^2 - \xi_i] - \sum_{i=1}^{N} \beta_i \xi_i \,. \quad (2.24)$$

By differentiating L with respect to the primal variables $\boldsymbol{c}, r, \boldsymbol{\xi}$ the following equations are obtained:

$$\frac{\partial L}{\partial \boldsymbol{c}} = 2 \sum_{i=1}^{N} \alpha_i (\phi_i - \boldsymbol{c}) = 0 \,, \quad (2.25)$$

$$\frac{\partial L}{\partial r} = 2r(1 - \sum_{i=1}^{N} \alpha_i) = 0 \,, \quad (2.26)$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \,, \quad (2.27)$$

Finally, the following constraints are obtained:

$$\sum_{i=1}^{N} \alpha_i = 1 \,, \quad (2.28)$$

$$\boldsymbol{c} = \sum_{i=1}^{N} \alpha_i \phi_i \,, \quad (2.29)$$

$$0 \leq \alpha_i \leq C \,. \quad (2.30)$$

Substituting these constraints into the Lagrangian leads the following dual optimization problem:

$$max_{\boldsymbol{\alpha}} \ \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \,, \quad (2.31)$$

subject to:

$$\sum_{i=1}^{N} \alpha_i = 1 \,,$$

$$0 \leq \alpha_i \leq C, \ i = 1, 2, \ldots, N \,.$$

Feature vectors on the hypersphere's boundary have α_i coefficients such that $0 < \alpha_i < C$. The radius r of the hypersphere is calculated as the distance from its center to one of those vectors.

Feature vectors with $\alpha_i = C$ are located outside the hypersphere. Consequently, they are considered outliers. All feature vectors with positive α_i values influence the domain description, and they are called the *support vectors* (SVs) of the description. A test point $\boldsymbol{z}\epsilon\mathcal{X}$ is declared an outlier if its distance to the center of the hypersphere is greater than *r*; i.e. if:

$$d(\phi(\mathbf{z}), \boldsymbol{c}) = \sqrt{k(\mathbf{z}, \boldsymbol{z}) - 2\sum_{i=1}^{N} \alpha_i k(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)} > r \ . \qquad (2.32)$$

Given that $\sum_{i=1}^{N} \alpha_i = 1$, the value of *C* must be in the interval $\left[\frac{1}{N}, 1\right]$. Consequently, for $C < \frac{1}{N}$ no solution can be found. On the other hand, for $C > 1$ a solution covering all feature vectors can always be found.

The SVDD novelty detector has two important advantages. First, similar to one-class SVM, it relies on modeling the boundary of the density distribution of the normal data instead of the actual distribution, so that it is robust to variations of the distribution within the region defined as normal. Only variations of the distribution beyond that boundary will affect SVDD's performance. The second advantage is related to the estimation of the expected target error rate (error of type I). A target error occurs when an observation drawn from the target distribution is incorrectly classified as an outlier. Assuming that all observations in the training data set are actually drawn from the target distribution, support vectors with $\alpha_i = C$ are considered target errors. If a support vector $\mathbf{x}_i$ with $\alpha_i < C$ was removed from the training data before training, then the resulting boundary might shrink. In that case, evaluating the novelty detector on $\mathbf{x}_i$ will trigger a target error. On the other hand, training SVDD on the data set with one or more non-support vectors ($\alpha_i = 0$) left out renders the same solution that is obtained with the original training data. Because non-support vectors lies within the target boundary they won't be detected

as outliers. In summary, a leave-one-out estimation (Bishop, 1995) of the target error rate is given by the expression $\frac{\#SVs}{N}$, where #SVs denotes the number of support vectors.

Support Vector Data Description (SVDD) (Tax & Duin, 2004)**.** The Support Vector Domain Description method was subsequently expanded to accept negative examples (examples of outliers) as part of the learning process. The method was renamed as Support Vector Data Description. The optimization problem of the original SVDD was modified to consider negative examples as well. When there are no negative examples, the new SVDD remains the same as the Support Vector Domain Description. Following the notation in (Tax & Duin, 2004), the target objects (normal observations) are enumerated by indices *i*, *j*, and the negative examples are enumerated by indices *l*, *m*. The normal (target) objects are labeled as $y_i = 1$ and the negative examples are labeled as $y_l = -1$. Introducing slack variables ξ_i and ξ_l for both the target and the outlier examples, the modified primal optimization problem is as follows:

$$min_{c,r,\xi_i,\xi_l} \quad r^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \,, \qquad (2.33)$$

subject to:

$$\|\phi_i - \boldsymbol{c}\|^2 = (\phi_i - \boldsymbol{c})^T(\phi_i - \boldsymbol{c}) \leq r^2 + \xi_i; \quad \xi_i \geq 0; \quad \forall i$$

$$\|\phi_l - \boldsymbol{c}\|^2 = (\phi_l - \boldsymbol{c})^T(\phi_l - \boldsymbol{c}) \geq r^2 - \xi_l; \quad \xi_l \geq 0; \ \forall l$$

Introducing Lagrange multipliers $\alpha_i, \beta_i \geq 0$ and $\alpha_l, \beta_l \geq 0$ and applying a derivation process similar as the one used for Support Vector Domain Description, the following constraints are obtained:

$$\sum_i \alpha_i - \sum_l \alpha_l = 1 \,, \qquad (2.34)$$

$$\boldsymbol{c} = \sum_i \alpha_i \phi_i - \sum_l \alpha_l \phi_l \,, \quad (2.35)$$

$$0 \le \alpha_i \le C_1 \,, \quad 0 \le \alpha_l \le C_2; \quad \forall i, l \quad (2.36)$$

The dual optimization problem is expressed as:

$$max_{\{\alpha_i\},\{\alpha_l\}} \quad \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_l \alpha_l k(\mathbf{x}_l, \mathbf{x}_l) - \sum_{i,j} \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) - \sum_{l,m} \alpha_l \alpha_m k(\mathbf{x}_l, \mathbf{x}_m) + 2 \sum_{l,j} \alpha_l \alpha_j k\left(\mathbf{x}_l, \mathbf{x}_j\right) \,, \quad (2.37)$$

subject to:

$$\sum_i \alpha_i - \sum_l \alpha_l = 1 \,, \quad (2.38)$$

$$0 \le \alpha_i \le C_1; \quad 0 \le \alpha_l \le C_2; \quad \forall i, l \quad (2.39)$$

Let us assume there *N* observations in the training data (including both target and negative examples), and let us enumerate variables corresponding to all training observations by using index *i*. Defining new variables $\alpha_i' = \mathrm{y}_i \alpha_i$, constraints (2.38) and (2.39) change into

$$\sum_{i=1}^{N} \alpha_i' = 1 \text{ and } \boldsymbol{c} = \sum_{i=1}^{N} \alpha_i' \phi_i. \quad (2.40)$$

Consequently, by doing this transformation, the SVDD with negative examples is expressed mathematically in the same terms as the unsupervised version of SVDD. Similarly, the function to detect whether a test observation $\boldsymbol{z} \epsilon \mathcal{X}$ is an outlier has the same expression for both versions of SVDD.

The work in (Wu & Ye, 2009) aims at improving the margins of SVDD's hypersphere, so that its distance from outliers in the training data is maximized. That work is extended in (Le, Tran, Ma, & Sharma, 2010), where the margin between the hypersphere and normal observations is also maximized. Modeling multiple hyperspheres has also been proposed recently (Le, Tran, Ma, & Sharma, 2011), outperforming the original SVDD method in multiple data sets. Some efforts

have also been made to improve the speed of the SVDD method, such as *fast SVDD* (Liu, Liu, & Chen, 2010) and *efficient SVDD* (Peng & Xu, 2012).

2.2.2 Online Methods

Online SVDD (Tax & Laskov, 2003)

This method is based on a generalization of *incremental SVM* (Cauwenberghs & Poggio, 2001), which is an exact solution to supervised online SVM learning. Consequently, in order to understand online SVDD it is necessary to briefly introduce first the standard SVM binary classifier and incremental SVM.

The model of the standard soft-margin SVM binary classifier represents a separating hyperplane $f(\mathbf{x})$ in the feature space, which has maximum margin and allows for some mislabeled training examples (Cortes & Vapnik, 1995):

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \,, \qquad (2.41)$$

such that $y_i(\mathbf{w}, \phi(\mathbf{x_i}) + b) \geq 1 - \xi_i$ for i = 1,2,…,N; where $\mathbf{x}_i$ denote the training vectors, the labels y_i take value in {-1, 1}, and ξ_i are non-negative slack variables that are as small as possible (how small is defined by a regularization parameter C introduced below).

To obtain the trained SVM model, a primal optimization problem is written as:

$$min_{\mathbf{w},b,\xi_i} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \,, \qquad (2.42)$$

subject to:

$$y_i(\mathbf{w}, \phi(\mathbf{x_i}) + b) \geq 1 - \xi_i \ \text{ and } \ \xi_i \geq 0, \ \text{ for } i = 1, 2, \ldots, N.$$

Using the technique of Lagrange multipliers, this primal problem is converted into the following convex quadratic dual problem:

$$\max_b \min_{\alpha_i} W = \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i + b\sum_{i=1}^{N} y_i\alpha_i\,, \qquad (2.43)$$

subject to:

$$0 \le \alpha_i \le C, \text{ for } i = 1,2,\ldots,N.$$

Deriving W w.r.t. variables α_i and b, the following conditions, called Karush–Kuhn–Tucker (KKT) conditions, are obtained:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{N}\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + b y_i - 1 \quad \begin{cases} g_i \ge 0, & if\ \alpha_i = 0 \\ g_i = 0, & if\ 0 < \alpha_i < C \\ g_i \le 0, & if\ \alpha_i = C \end{cases}, \qquad (2.44)$$

$$\frac{\partial W}{\partial b} = \sum_{i=1}^{N} y_i\alpha_i = 0\,. \qquad (2.45)$$

Finally, the function for the optimal separating hyperplane can be written as a linear combination of values of the kernel function:

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) + b, \qquad (2.46)$$

for which coefficients α_i and b are calculated by solving the dual problem stated above.

The incremental SVM method allows the addition of training observations, one at a time, to an SVM model. It also allows removing a single observation from an SVM model. Training observations are explicitly classified into three categories: the set S of margin support vectors (i.e. those for which $0 < \alpha_i < C$), the set E of error support vectors (those exceeding the margin but not necessarily misclassified, for which $\alpha_i = C$), and the set R containing the rest of the vectors.

Let us assume that an optimal solution to the SVM optimization problem is already available, which could have been obtained through batch training. Let us assume also that a new vector $\mathbf{x}_c$, with label y_c, needs to be added to the corresponding SVM model. As a first step, $\mathbf{x}_c$ is added with coefficient $\alpha_c = 0$, which does not affect the model nor the KKT conditions. If $g_c > 0$ then the algorithm terminates; otherwise, the value α_c is incremented as much as possible, until either (1) $g_c = 0$, (2) $\alpha_c = C$, or (3) previously learned vectors migrate across sets *S*, *E*, and *R*. In the first two cases some model updates are executed and the algorithm terminates. In the third case, updates are applied to maintain the KKT conditions, and the previous step to increment α_c is repeated. A similar procedure is applied to remove an observation from the SVM model, in this case iteratively decrementing the corresponding α_c coefficient until it reaches zero, while keeping the KKT conditions fulfilled, which implies having an optimal model. Incremental SVM relies on equations that allow calculating, given a change in α_c denoted by $\Delta\alpha_c$, the corresponding changes in b, α_i, and the derivatives g_i (denoted by Δb, $\Delta\alpha_i$, and Δg_i, respectively). According to its authors, the algorithm converges to a solution identical to the SVM model obtained through standard approaches based on quadratic optimization of the dual problem.

Online SVDD is based on a generalization of the dual problem of SVM and the formulation of the incremental SVM algorithm. A general abstract form of the SVM optimization problem is considered in (Tax & Laskov, 2003), which is stated below (with some changes in notation to better match the notation used above):

$$\max_b \min_{\boldsymbol{\alpha}} W = \frac{1}{2}\boldsymbol{\alpha}^T M\boldsymbol{\alpha} - \boldsymbol{c}^T\boldsymbol{\alpha} + b(\boldsymbol{\alpha}^T\boldsymbol{\alpha} + d)\,, \qquad (2.47)$$

subject to

$$0 \leq \boldsymbol{\alpha} \leq C$$
$$\boldsymbol{a}^T \boldsymbol{\alpha} + d = 0$$

where **c** and $\boldsymbol{a}$ are vectors of size *N*, *M* is a $N \times N$ matrix, and *d* is a real value. Note that this problem becomes the standard SVM dual problem when using $\boldsymbol{c} = \mathbf{1}$, $\boldsymbol{a} = \mathbf{y}$, and $d = 0$. Alternatively, the above expression denotes the dual problem from the SVDD method if $\boldsymbol{c} = diag(M)$, $\boldsymbol{a} = \mathbf{y}$, and $d = -1$. An online version of SVDD is obtained by applying the mathematical derivation of incremental SVM to the general problem (2.47), and subsequently substituting the values of $\boldsymbol{c}$, $\boldsymbol{a}$ and *d* corresponding to the SVDD method. The only major difference between incremental SVM and online SVDD arises at the moment of defining an initial optimal model: when using incremental SVM for a classification problem, an initial solution can always be found for even a single observation. However, for online SVDD at least $\left\lceil \frac{1}{C} \right\rceil$ examples are needed to define an initial solution, where $C \in \left[\frac{1}{N}, 1\right]$. This is required in order to satisfy conditions $0 \leq \boldsymbol{\alpha} \leq C$ and $\mathbf{y}^T \boldsymbol{\alpha} = 1$ at the same time. The following procedure was proposed in (Tax & Laskov, 2003) to obtain the initial solution for online SVDD:

1. Take the first $\left\lfloor \frac{1}{C} \right\rfloor$ training observations into the set *E* and assign them weight *C*.
2. Take another observation $\mathbf{x}_k$, assign it a weight $\alpha_k = 1 - \left\lfloor \frac{1}{C} \right\rfloor C$, and put it into set *S*.
3. Compute gradients g_i of all objects in the solution.
4. Compute *b* such that for all observations in *E* the gradient is non-positive.
5. Enter the online learning phase of the algorithm.

A detailed analysis of the convergence properties and the algorithmic complexity of incremental SVM (and consequently online SVDD) is given in (Laskov, Gehl, Krüger, & Müller, 2006). The work of Laskov et al. also demonstrated the applicability of incremental SVM to real-life problems. Examples of more recent applications of online SVDD can be found in (Yin, Zhang, Li, Ren, & Fan, 2014) and (Kolev, Suvorov, Morozov, Markarian, & Angelov, 2015).

NORMA (Kivinen, Smola, & Williamson, 2004)

The term NORMA stands for Naive Online R_{reg} Minimization Algorithm. Actually, it denotes a collection of online algorithms that perform stochastic gradient descent with respect to a risk functional defined on the Hilbert (feature) space $\mathcal{F}$. NORMA is described here first in general terms (applicable to different types of learning problems). Subsequently, a variant of NORMA suited to online novelty detection is then briefly described.

The general approach used to develop NORMA considers a function estimation problem: to learn a mapping $f: \mathcal{X} \to \mathbb{R}$ from a training data set $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) | \mathbf{x}_i \epsilon \mathcal{X}, \mathrm{y}_i \epsilon \mathcal{Y},\ i = 1, \dots, N\}$. A loss function $l: \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$, given by $l\big(\hat{f}(\mathbf{x}), \mathrm{y}\big)$, penalizes the deviation of estimates $\hat{f}(\mathbf{x})$ from the observed label y. Although the authors of NORMA didn't state a particular relationship between $\mathcal{Y}$ and $\mathbb{R}$, it is apparent from their formulation that $\mathcal{Y} \subseteq \mathbb{R}$. Any estimate $\hat{f}$ obtained by the learning algorithm is called a *hypothesis*. It is assumed that the feature space $\mathcal{F}$ is a *reproducing kernel Hilbert space* (Aronszajn, 1950), (Schölkopf & Smola, 2001), which contains all possible hypotheses. The main implication of this assumption is that the associated kernel function *k* has the following reproducing property:

$$\langle f, k(\mathbf{x}, .) \rangle = f(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \qquad (2.48)$$

The concept of risk functional is very important in NORMA. Typically, all examples from the training data set D are assumed to be drawn independently from some distribution P defined over $\mathcal{X} \times \mathcal{Y}$. Given an estimate $\hat{f}$ of the function to be learned, the following expected risk offers a natural measure of the quality of the estimation $\hat{f}$:

$$R[\hat{f}, P] \equiv E_P \left[l(\hat{f}(\mathbf{x}), \mathrm{y}) \right]. \qquad (2.49)$$

Since P is unknown, the expected risk can be approximated by the empirical risk R_{emp}:

$$R_{emp}[\hat{f}, D] \equiv \frac{1}{N} \sum_{i=1}^{N} l(\hat{f}(\mathbf{x}_i), \mathrm{y}_i). \qquad (2.50)$$

However, to avoid overfitting, a regularized risk should be used instead of $\mathrm{R}_{\mathrm{emp}}$:

$$R_{reg}[\hat{f}, D] \equiv \frac{1}{N} \sum_{i=1}^{N} l(\hat{f}(\mathbf{x}_i), \mathrm{y}_i) + \frac{\lambda}{2} \|\hat{f}\|^2; \quad \lambda > 0. \qquad (2.51)$$

The previous risk functionals are the ones typically used in batch learning. A definition of a risk functional dealing with one example at a time is needed for online learning. For NORMA, the *instantaneous regularized risk* on a single example ($\mathbf{x}$, y) is defined as follows:

$$R_{inst}[\hat{f}, \mathbf{x}, \mathrm{y}] \equiv R_{reg}[\hat{f}, \{(\mathbf{x}, \mathrm{y})\}]. \qquad (2.52)$$

NORMA assumes the existence of an arbitrary initial hypothesis $\hat{f}_1$. After NORMA examines the t^{th} example $(\mathbf{x}_t, \mathrm{y}_t)$, it generates an updated hypothesis $\hat{f}_{t+1}$. Consequently, the goal for NORMA is to reduce the loss $l(\hat{f}_t(\mathbf{x}_t), \mathrm{y}_t)$ made by the learning algorithm when it predicts y_t based on $\mathbf{x}_t$ and previous examples $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1,2,\ldots,t-1}$. The main idea is to carry out the classic stochastic gradient descent with respect to the instantaneous risk R_{inst}. The general form of the update rule is the following:

$$\hat{f}_{t+1} = \hat{f}_t - \eta_t \frac{\partial}{\partial f} R_{inst}[f, \mathbf{x}_t, \mathrm{y}_t] \Big|_{f=\hat{f}_t}, \qquad (2.53)$$

where $\eta_t > 0$ is the *learning rate*, similar to the learning rate employed in multilayer neural networks. Employing properties of reproducing kernel Hilbert spaces, the update rule becomes:

$$\hat{f}_{t+1}(\mathbf{x}) \equiv (1 - \eta_t \lambda)\hat{f}_t(\mathbf{x}) - \eta_t l'\big(\hat{f}_t(\mathbf{x}_t), \mathrm{y}_t\big)\, k(\mathbf{x}_t, \mathbf{x}) . \qquad (2.54)$$

Consequently, the function $\hat{f}_t$, at any step *t*, can be written as a kernel expansion (Schölkopf, Herbrich, & Smola, 2001):

$$\hat{f}_t(\mathbf{x}) = \sum_{i=1}^{t-1} \alpha_i k(\mathbf{x}_i, \mathbf{x}) . \qquad (2.55)$$

Considering loss functions that are convex in the first argument, the general NORMA algorithm is summarized in the following two steps:

STEP 1. Choose as initial function estimate $\hat{f}_1 = 0$.

STEP 2. The coefficients $\alpha_1, \alpha_2, \ldots, \alpha_t$ are updated at step *t* using the following expressions:

$$\alpha_t \leftarrow -\eta_t l'\big(\hat{f}_t(\mathbf{x}_t), \mathrm{y}_t\big) , \qquad (2.56)$$

$$\alpha_i \leftarrow (1 - \eta_t \lambda)\alpha_i; \qquad i = 1, 2, \ldots, t-1 , \qquad (2.57)$$

where the symbol ← denotes a value assignment. To avoid a continuously increasing number of coefficients, the authors of NORMA suggested removing observations having very small coefficient values. This truncation procedure also allows NORMA to forget old instances that become irrelevant. This is particularly beneficial in the case of a changing distribution $P(x, y)$.

A variant of NORMA for novelty detection was derived in (Kivinen, Smola, & Williamson, 2004). It assumes an unsupervised learning scenario in which the following loss function is used:

$$l\left(\hat{f}(\mathbf{x}), \mathbf{x}\right) = max\left(0, \rho - \hat{f}(\mathbf{x})\right) - \upsilon\rho \ , \qquad (2.58)$$

where parameter ρ denotes the width of the margin, and $0 < \upsilon < 1$ allows to set an upper limit in the frequency of outlier alerts ($\hat{f}(\mathbf{x}) < \rho$). The update rule in this case, for $i = 1,2, \ldots, t-1$, is:

$$(\alpha_i,\ \alpha_t,\ \rho) = \begin{cases} \left((1-\eta_t)\alpha_i,\ \eta,\ \rho + \eta(1-\upsilon)\right), & if\ \hat{f}(\mathbf{x}) < \rho \\ \left((1-\eta_t)\alpha_i,\ 0,\ \rho - \eta\upsilon\right), & otherwise \end{cases} . \qquad (2.59)$$

Note that whenever $\hat{f}(\mathbf{x}) \geq \rho$ we have that $\alpha_t = 0$. This means that there is no need to keep the corresponding $\mathbf{x}_t$ vector in memory, which provides some sparseness to the underlying model. As an example of a recent and interesting application of the NORMA algorithm to novelty detection, note that it has been used with great success to learn the normal postures of elderly persons, using short video clips as training data (Yu, Yu, Rhuma, Naqvi, Wang, & Chambers, 2013). Using video monitoring, the algorithm identified abnormal postures that were likely corresponding to falls; achieving in some cases 100% fall detection rate and only 3% false detection rate.

2.3 Gaussian Processes for Novelty Detection

A Gaussian process (GP) is a stochastic process used to specify a probability distribution over a space of functions without constraining the corresponding functional model to a particular form. Essentially, GPs are a flexible nonparametric function estimation technique. GP models can be obtained from a training data set using a batch learning algorithm. Additionally, there are algorithms to learn GP models incrementally, such as Online GP and the Sparse Online GP (SOGP) (Csató & Opper, 2002). SOGP can be used in applications that impose relatively strong

memory constraints; for instance, in some embedded systems. Batch GP and Online GP might not be appropriate in those cases, given that their models do not include an approach to compensate for the potential absence of available memory. SOGP achieves this capability by adding a parameter *m* that specifies the capacity of the model, with the goal of building a sparse but efficient knowledge representation.

Gaussian processes have been used successfully in many areas as a powerful Bayesian regression tool, given its flexible modeling capabilities, and the fact that posterior GPs can be obtained analytically when using Gaussian likelihoods. Their use in machine learning has been mainly limited to solving regression and classification problems (Rasmussen & Williams, 2006), and to estimate the probability density functions underlying a set of observations (Csató, 2002) (Adams, Murray, & MacKay, 2009). In most cases, GPs have showed a great performance compared to other highly successful techniques (Rasmussen & Williams, 2006).

It was reported in (Kemmler, Rodner, & Denzler, 2010) and (Kemmler M. , Rodner, Wacker, & Denzler, 2013) that GPs can be effectively used for novelty detection as well. Their experimental results show that GP-based novelty detection can outperform on average the state-of-the-art SVDD algorithm. However, despite the general acceptance of GPs for the domains mentioned above, applications of GPs to novelty detection seem to have been limited to the approach originally published in (Kemmler, Rodner, & Denzler, 2010). Furthermore, the work of (Kemmler, Rodner, & Denzler, 2010) and (Kemmler M. , Rodner, Wacker, & Denzler, 2013) was constrained to the use of batch GP. To the best of our knowledge, the only work that has considered the application of online GPs to novelty detection is described in (Ramirez-Padron,

Mederos, & Gonzalez, 2013). Its preliminary experimental results show that the performance of novelty detection methods based on online GPs can be similar to the performance of batch GP-based novelty detection. Given the promising results presented in (Kemmler M. , Rodner, Wacker, & Denzler, 2013) and (Ramirez-Padron, Mederos, & Gonzalez, 2013), and the renowned flexibility of GPs as modeling tools, extending the application of GPs to particular types of novelty detection problems seems to be a promising research effort.

This section introduces Bayesian modeling, which lies at the core of the GP approach to novelty detection, and subsequently offers a brief introduction to GPs and its applications to regression and classification problems. This introduction is needed because applications of GPs to density estimation and novelty detection are based on ideas that were developed for regression and classification within the Bayesian framework. Finally, this section describes the approach proposed in (Kemmler, Rodner, & Denzler, 2010) and (Kemmler M. , Rodner, Wacker, & Denzler, 2013) for applying GPs to novelty detection.

2.3.1 Bayesian Modeling

The simplest approaches to learning a model from a training data set involve using an expression $f(\mathbf{x}, \mathbf{w})$ that is linear in the unknown parameters $\mathbf{w}$. Typically in the case of supervised learning, the model parameters are found by minimizing an error function that measures the misfit between the model and the training labels. A regularization approach is commonly employed to avoid over-fitting the training data. Regularization involves adding one or more penalty terms to the objective function of the optimization problem. Penalty terms are called *regularization terms*, because they measure how much the model deviates from some pre-defined desirable conditions.

The following regularized objective function is commonly used in various methods, including SVM:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}[f(\mathbf{x}_i, \mathbf{w}) - \mathrm{y}_i]^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 . \qquad (2.60)$$

The second term of $E(\mathbf{w})$ is the regularization term, which is inversely proportional to the smoothness of the model. The coefficient λ is a model parameter that allows fine-tuning the relative importance of the regularization term in comparison with the error term. Parameters modifying the complexity of the model, like λ, are called *complexity parameters*. Other complexity parameters are typically considered as part of the expression for $f(\mathbf{x}, \mathbf{w})$; for instance, if $f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{m} \mathbf{w}_i \, \phi_i(\mathbf{x})$, where $\{\phi_i\}$ is a set of *m* basis functions, then the number *m* is a complexity parameter. In the case of SVM, the parameters of the kernel function can be considered complexity parameters. In general, a model might have several complexity parameters that need to be estimated in addition to estimate the parameters $\mathbf{w}$. The act of estimating the complexity parameters constitutes an example of a task known in statistics as *model selection*. One of the classic approaches to model selection is to employ a *k*-fold cross-validation procedure over different combinations of values of the complexity parameters. As a result, complexity parameters are set to the combination of values for which the average of the function error $E(\mathbf{w})$ over the cross-validation runs is a minimum. This approach to model selection is clearly cumbersome and computationally expensive, and it might be infeasible for models with various complexity parameters that take values in large domains. One of the main advantages of the Bayesian approach to statistical inference is that it allows estimating the probability distributions of all model parameters in a unified way. Additionally, the introduction of prior distributions that favor model smoothness removes the need for explicit regularization

terms. In other words, prior distributions play the role of regularization terms, preventing the posterior model from deviating too much from prior conditions that are typically smooth.

The previous argument in favor of Bayesian learning methods is just one of their multiple advantages. When a system is modeled using a Bayesian framework, our knowledge is expressed by probability distributions defined on the parameters of the model. Initially, a model is built relying exclusively on our prior beliefs about the system. Subsequently, training data are used to adjust the probability distributions of the model, in a way that it provides a better explanation for the data. Another major advantage relies on the fact that predictions of Bayesian models are fully probabilistic; i.e. Bayesian models do not only produce point estimates of the dependent variables (as it is the case for regularized linear models), but they also provide posterior probability distributions for those variables. Consequently, they provide a measure of the uncertainty associated to predictions.

The core of Bayesian modeling is Bayes' theorem, published originally in 1763 for the specific case of updating the parameters of a Binomial distribution based on observational data (Bayes, 1763). The modern expression of Bayes' theorem, extended to arbitrary distributions, was presented in (Laplace, 1812). The prevailing interpretation of probabilities based on frequencies during the 19^{th} century caused Bayes theorem to be overlooked for about 100 years (Cox, 1946). Currently, there is a great interest in setting most statistical methods into a Bayesian framework (Bolstad, 2007). Similarly, many machine learning algorithms have been re-stated within a Bayesian framework (Bishop, 2006). Bayesian inference can be done on two levels. The first level is concerned with inferring the distribution of model parameters, while the second level

deals with selecting the most appropriate models. In the following subsections, the two levels of Bayesian inference are described in general terms, independently of any particular model.

First level of Bayesian Inference

Let us assume that a particular model *M* was chosen based on certain prior information, to fit a data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N | \, \mathbf{x}_i \in \mathcal{X}\}$, where $\mathcal{X}$ denotes a finite-dimensional space. The model *M* is defined by some fixed algebraic structure, and a set of parameters **w** for which we have a prior distribution $p(\mathbf{w}|M)$. The *first level of Bayesian inference* consists of inferring the posterior distribution of the parameters **w** given training data coming from the system that is being modeled. That inference is done through Bayes' theorem:

$$p(\mathbf{w}|\mathbf{X}, M) = \frac{p(\mathbf{X}|\mathbf{w},M)p(\mathbf{w}|M)}{p(\mathbf{X}|M)} = \frac{p(\mathbf{X}|\mathbf{w},M)p(\mathbf{w}|M)}{\int p(\mathbf{X}|\mathbf{w},M)p(\mathbf{w}|M)d\mathbf{w}} . \qquad (2.61)$$

The term $p(\mathbf{X}|\mathbf{w}, M)$ is called the *likelihood* of the data, and the term $p(\mathbf{X}|M)$ is called the *marginal likelihood* (or *evidence*). Point estimates of the model parameters **w** are typically obtained by calculating the mean or the mode (*maximum a posteriori*) of the posterior distribution $p(\mathbf{w}|\mathbf{X}, M)$. Employing the terms introduced above, Bayes' theorem can be stated as follows in general terms:

$$Posterior = \frac{(Likelihood)(Prior)}{Evidence} . \qquad (2.62)$$

Our beliefs about the parameters **w** are conditioned on the structure of the model *M*. However, if the model *M* was fixed a priori, then the conditioning on *M* is usually omitted to simplify the notation of Bayes' theorem:

$$p(\mathbf{w}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\mathbf{w})p(\mathbf{w})}{\int p(\mathbf{X}|\mathbf{w})p(\mathbf{w})d\mathbf{w}} . \qquad (2.63)$$

Researchers have defined families of likelihood functions. For each particular family of likelihoods there exists an associated family of prior distributions $p(\mathbf{w})$, called the *conjugate prior family*, such that if $p(\mathbf{w})$ is a conjugate prior then the posterior distribution $p(\mathbf{w}|\mathbf{X})$ remains a member of the same conjugate prior family. The main advantage of using conjugate priors is that typically the integral in the denominator of Bayes' theorem can be calculated analytically, which is very convenient (Bolstad, 2007).

One of the main motivations for obtaining the posterior distribution of the parameters **w** is to estimate the conditional *predictive distribution* of a new observation $\mathbf{x}_{N+1}$. The predictive distribution $p(\mathbf{x}_{N+1}|\mathbf{X})$ allows calculating point estimates of $\mathbf{x}_{N+1}$, typically expressed as the mean of $p(\mathbf{x}_{N+1}|\mathbf{X})$. Additionally, the predictive distribution can be used to obtain a *Bayesian credible interval* for $\mathbf{x}_{N+1}$. The predictive distribution $p(\mathbf{x}_{N+1}|\mathbf{X})$ can be obtained through a technique called *marginalization over a random variable*, which allows us to write $p(\mathbf{x}_{N+1}|\mathbf{X})$ employing known terms that involve another random variable. In this case, marginalizing $p(\mathbf{x}_{N+1}|\mathbf{X})$ over **w** consists of "injecting" the parameters **w** in the following manner:

$$p(\mathbf{x}_{N+1}|\mathbf{X}) = \int p(\mathbf{x}_{N+1}, \mathbf{w}|\mathbf{X}) d\mathbf{w} = \int p(\mathbf{x}_{N+1}|\mathbf{w}, \mathbf{X})\, p(\mathbf{w}|\mathbf{X}) d\mathbf{w}. \qquad (2.64)$$

Assuming that the values $\mathbf{x}_i$ are conditionally independent from each other given **w**, their distribution is fully determined by the likelihood term $p(\mathbf{x}_{N+1}|\mathbf{w})$ and the posterior term $p(\mathbf{w}|\mathbf{X})$, which leads to the following equation:

$$p(\mathbf{x}_{N+1}|\mathbf{X}) = \int p(\mathbf{x}_{N+1}|\mathbf{w})\, p(\mathbf{w}|\mathbf{X}) d\mathbf{w}. \qquad (2.65)$$

In the case of supervised learning (regression or classification problems), the training data are denoted by $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i)|\mathbf{x}_i \epsilon \mathcal{X}, \mathrm{y}_i \epsilon \mathbb{R},\ i = 1, \dots, N\}$, $\mathcal{X}$ being a finite-dimensional

space. The motivation behind obtaining the posterior distribution of the parameters **w** is to model the underlying mapping between the attributes $\mathbf{x}_i$ and the responses y_i. Only the response variables y_i are considered random in that case. Consequently, it is required to estimate the *predictive distribution* of the response y_{N+1} given a new observation $\mathbf{x}_{N+1}$ and D, which is typically obtained also through marginalization over **w**:

$$p(\mathrm{y}_{N+1}|D, \mathbf{x}_{N+1}) = \int p(\mathrm{y}_{N+1}, \mathbf{w}|D, \mathbf{x}_{N+1}) d\mathbf{w} = \int p(\mathrm{y}_{N+1}|\mathbf{w}, \mathbf{x}_{N+1})\, p(\mathbf{w}|D, \mathbf{x}_{N+1}) d\mathbf{w} \,. \quad (2.66)$$

Second Level of Bayesian Inference

Besides the parameters **w**, for which we are obtaining the conditional distributions, some terms in the Bayesian model, like the likelihood or the prior distribution, might in turn be conditioned on one or more parameters $\boldsymbol{\theta}$, typically called *hyperparameters*. By varying the values of those hyperparameters, it is possible to have prior distributions that better express our beliefs and previous domain knowledge. Let us assume that we can choose a model from a family of models $\mathcal{M} = \{M_i | i \in I\}$, where I is any index set. Each model M_i has a set of hyperparameters $\boldsymbol{\theta}_i$. A question that arises immediately is how to decide which model is a better fit to the training data D. The *second level of Bayesian inference* helps to provide an answer to this question. It consists of applying Bayes' theorem with a prior distribution over the set of models, and calculating the posterior distribution of the models given the data:

$$p(M_i|D) = \frac{p(D|M_i)p(M_i)}{p(D)} \,. \quad (2.67)$$

The posterior distribution of the models allows us to establish which models are more likely given the data.

A Note on Intractable Marginal Likelihoods

A difficulty commonly associated with Bayesian modeling is that the integral corresponding to the marginal likelihood term could be analytically intractable. Typically, that happens when prior distributions are not from the corresponding conjugate prior family. Sometimes we need to use priors that are inconvenient from an analytical point of view, in order to have priors that truly reflect our beliefs. In those cases, there are several ways of approximating the intractable integral. It is important to note however, that for some applications of the Bayes' theorem it is not necessary to calculate the marginal likelihood. For instance, in the second level of inference we are interested in determining which models provide the highest probability, but we are not interested in knowing the actual values of the corresponding posterior. Given that $p(D)$ is a constant term, the following *proportional form of Bayes' theorem* can be employed in that case to determine the most promising model:

$$p(M_i|D) \propto p(D|M_i)p(M_i)\ . \qquad (2.68)$$

2.3.2 Gaussian Processes

This subsection provides a comprehensive introduction to batch GP, online GP, and SOGP in the context of nonparametric Bayesian regression. Gaussian processes are nonparametric kernel-based function estimation techniques that model a probability distribution over a space $\mathfrak{F}$ of functions $f: \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ (d being a positive integer) is a continuous input space (Bishop, 2006), (MacKay, 1998), (Rasmussen & Williams, 2006), (Seeger M. , 2004). GPs

provide an assessment of the uncertainty associated to predicting $f(\mathbf{x})$ at any point $\mathbf{x} \in \mathcal{X}$. Note that $f_{\mathbf{x}}$ and $f(\mathbf{x})$ are used indistinctly in this work. Similarly, the random variable $\mathrm{y}(\mathbf{x})$ is sometimes denoted by $\mathrm{y}_{\mathbf{x}}$. The following definition of GPs is commonly used (Rasmussen & Williams, 2006), (Lifshits, 2012):

Definition: A Gaussian process is a collection of random variables $\{f_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{X}}$, such that any finite subcollection $\mathbf{f} = \{f_{\mathbf{x}_1}, f_{\mathbf{x}_2}, \dots, f_{\mathbf{x}_M}\}$, where M is any positive integer, has a joint Gaussian distribution.

This definition implies that a GP is completely determined by its mean function μ and covariance (kernel) function k:

$$\mu(f_{\mathbf{x}}) = E[f_{\mathbf{x}}]\,, \tag{2.69}$$

$$k\left(f_{\mathbf{x}_i}, f_{\mathbf{x}_j}\right) = E\left[\left(f_{\mathbf{x}_i} - \mu(f_{\mathbf{x}_i})\right)\left(f_{\mathbf{x}_j} - \mu\left(f_{\mathbf{x}_j}\right)\right)\right], \tag{2.70}$$

where $k\left(f_{\mathbf{x}_i}, f_{\mathbf{x}_j}\right) = cov(f_{\mathbf{x}_i}, f_{\mathbf{x}_j})$ is a positive definite kernel function. In practice, the function k directly depends on the points $\mathbf{x}_i$ and $\mathbf{x}_j$; hence, it is typically denoted by $k(\mathbf{x}_i, \mathbf{x}_j)$. Usually k also depends on some parameters $\boldsymbol{\theta}_k$, thus formally it should be denoted by $k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta}_k)$. We generally write $k(\mathbf{x}_i, \mathbf{x}_j)$ for the sake of simplifying notation. However, the existence of kernel parameters $\boldsymbol{\theta}_k$, which constitute hyperparameters for the GP, is implicitly assumed throughout this dissertation, unless stated otherwise.

Sometimes a GP is denoted as $f \sim \mathcal{GP}(\mu, k)$. This notation should be interpreted as follows: Given any arbitrary set of values $\{\mathbf{x}_i | \mathbf{x}_i \epsilon \mathcal{X};\ i = 1, \dots, N\}$, the corresponding set of f variables $\{f_{\mathbf{x}_1}, f_{\mathbf{x}_2}, \dots, f_{\mathbf{x}_N}\}$ follows a joint normal distribution $\mathcal{N}\left(\left(\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_N)\right)^T, K\right)$, where K

denotes the covariance matrix obtained by evaluating the kernel function k in all pairs $(\mathbf{x}_i, \mathbf{x}_j)$. A Bayesian approach to GP modeling focuses on establishing a prior distribution for the functions f, and subsequently estimating the posterior distribution $p(f|D)$. The mean function of a prior GP is denoted here by $\mu_0(\mathbf{x})$ and the prior covariance function is denoted by $k_0(\mathbf{x}, \mathbf{x}')$. Estimating the posterior GP implies obtaining expressions for its posterior mean function and its posterior covariance function, as described below.

2.3.2.1 Batch GP Regression

Let our training data $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) : \mathbf{x}_i \in \mathcal{X},\ \mathrm{y}_i \in \mathbb{R},\ i = 1,2 \ldots N\}$ be a set of input-output observations, where each y_i is a noisy observation of a latent variable f that depends on $\mathbf{x}_i$. A regression problem consists of constructing a model $f(\mathbf{x}; \mathbf{w})$ that provides the best possible fit to the training data. In a non-Bayesian approach, the quality of the models is assessed through certain optimization criterion, like the least-squares method or a regularized version of it. Consequently, the non-Bayesian solution provides only a point estimate of $f_{\mathbf{x}}$ given an observation $\mathbf{x}$. In contrast, the goal of the Bayesian approach to regression is to estimate a model defined as the posterior distribution $p(f_{\mathbf{x}}|D, \mathbf{x})$; which in turn is used to obtain the predictive distribution $p(\mathrm{y}_{\mathbf{x}}|\mathbf{x}, D)$. This involves (1) defining prior distributions $p(\mathbf{w})$ for the parameters of the underlying family of functions, (2) calculating the posterior distribution $p(\mathbf{w}|D)$ using the first level of Bayesian inference described above, and (3) calculating the distribution $p(f_{\mathbf{x}}|D, \mathbf{x})$.

This subsection describes how GPs are used to solve regression problems using a Bayesian non-parametric approach. Gaussian processes are plugged into the Bayesian regression framework to introduce a great deal of flexibility regarding the model: no need to enforce a fixed algebraic

structure on the function space; i.e. only basic properties like smoothness are required from *f*. This can be achieved because a GP model defines prior distributions $p(f)$ on a very flexible function space. We assume the classic approach that relates $\mathrm{y}(\mathbf{x})$ and $f(\mathbf{x})$ as follows:

$$\mathrm{y}(\mathbf{x}) = f(\mathbf{x}; \mathbf{w}) + \varepsilon, \qquad (2.71)$$

where $\mathbf{w}$ is a vector of function parameters, and the distribution of the observation error ε determines the conditional distribution of $\mathrm{y}_{\mathbf{x}}|f_{\mathbf{x}}$ (i.e. the likelihood model). The term ε denotes additive noise that follows an independent and identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, \sigma^2)$. We are ultimately concerned with estimating the probability density function $p(\mathrm{y}_{\mathbf{x}}|D, \mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$. However, note that sometimes GP models are used only to predict the most likely value of $f(\mathbf{x})$ given a new observation $\mathbf{x}$.

The variables $\{f_{\mathbf{x}}\}$ are used as latent random variables that are initially modeled using a prior GP (i.e. variables $\{f_{\mathbf{x}}\}$ play the role of the vector of parameters $\mathbf{w}$ in this case). Consequently, given an arbitrary finite set of indexes $\boldsymbol{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_M]^T$, the corresponding random vector $\mathbf{f} = [f(\mathbf{x}'_1),\ f(\mathbf{x}'_2), \ldots, f(\mathbf{x}'_M)]^T$ has the following joint prior distribution:

$$p_0(\mathbf{f}) = \frac{1}{\sqrt{(2\pi)^M |\boldsymbol{K}_0|}} e^{-\frac{1}{2}\left(\mathbf{f} - \boldsymbol{\mu}_0(\boldsymbol{X}')\right)^T \boldsymbol{K}_0^{-1} \left(\mathbf{f} - \boldsymbol{\mu}_0(\boldsymbol{X}')\right)}, \qquad (2.72)$$

where $\boldsymbol{\mu}_0(\boldsymbol{X}') = [\mu_0(\mathbf{x}'_1), \mu_0(\mathbf{x}'_2), \ldots, \mu_0(\mathbf{x}'_M)]^T$ and $\boldsymbol{K}_0 = \boldsymbol{K}_0(\boldsymbol{X}') = \left(k_0(\mathbf{x}'_i, \mathbf{x}'_j)\right)_{i,j}$ is an *M* x *M* matrix.

In the following, $\mathbf{f}_D = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^T$ denotes a Gaussian random vector as modeled by the GP on the indexes $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$. As mentioned above, the goal behind the GP derivation is to estimate the posterior distribution of $\mathbf{f}$ given D:

$$p_{post}(\mathbf{f}) = p(\mathbf{f}|\mathrm{D}) = \frac{p(D|\mathbf{f}) p_o(\mathbf{f})}{p(D)} = \frac{p(D|\mathbf{f}) \int p_o(\mathbf{f}, \mathbf{f}_D)\, d\mathbf{f}_D}{\int p(D, \mathbf{f}_D) d\mathbf{f}_D}$$

$$= \frac{\int p(D|\mathbf{f})\, p_o(\mathbf{f}, \mathbf{f}_D) d\mathbf{f}_D}{\int p(D|\mathbf{f}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D} = \frac{\int p(\mathbf{y}|\mathbf{f})\, p_o(\mathbf{f}, \mathbf{f}_D) d\mathbf{f}_D}{E_0[p(\mathbf{y} \mid \mathbf{f}_D)]}, \qquad (2.73)$$

where E_0 denotes expected value w.r.t. the prior GP. Consequently, the predictive distribution $p(\mathrm{y}_\mathbf{x}|D, \mathbf{x})$ given a single input $\mathbf{x}$ is written as:

$$p(\mathrm{y}_\mathbf{x}|\mathbf{x}, D) = \int p(\mathrm{y}_\mathbf{x}|f_\mathbf{x})\, p_{post}(f_\mathbf{x}) df_\mathbf{x}$$

$$= \frac{\int p(\mathrm{y}_\mathbf{x}|f_\mathbf{x})\, p(\mathbf{y}|f_\mathbf{x})\, p_o(f_\mathbf{x}, \mathbf{f}_D) d\mathbf{f}_D df_\mathbf{x}}{E_0[p(\mathbf{y}| \mathrm{f}_D)]}. \qquad (2.74)$$

The posterior $p_{post}(\mathbf{f})$ can be derived analytically only if the likelihood $p(\mathbf{y}|\, \mathbf{f}_D)$ is Gaussian, which in the case of single input $\mathbf{x}$ is written as $\mathrm{y}_\mathbf{x}|f_\mathbf{x} \sim \mathcal{N}(f_\mathbf{x}, \sigma^2)$. When the likelihood $p(\mathbf{y}|\mathbf{f}_D)$ is not Gaussian, calculating the posterior GP implies computing an *N*-dimensional integral which might be analytically intractable. In that case, there are techniques that can be used to approximate the posterior $p_{post}(\mathbf{f})$. In this work we are only employing Gaussian likelihoods. For that reason, approximation techniques are not considered here. Still, the fact that $\mathbf{f}$ appears within an integral in the equation corresponding to $p_{post}(\mathbf{f})$ requires evaluating that integral for doing predictions at arbitrary data points. To avoid computing high-dimensional integrals in that case, the parametrisation lemma (Csató & Opper, 2002) shows how predictions can rely only on linear and bilinear combinations of the kernel function evaluated in the training data D:

Parametrisation Lemma (Csató & Opper, 2002): Given a training data set $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i): \mathbf{x}_i \in \mathcal{X},\ \mathrm{y}_i \in \mathbb{R}, i = 1 \ldots N\}$, an arbitrary likelihood $\mathrm{p}(D|\mathbf{f}_D)$, and a prior GP with mean $\mu_0(\mathbf{x})$ and covariance function $k_0(\mathbf{x}, \mathbf{x}')$, the resulting posterior GP has mean and covariance functions given by:

$$\mu_{post}(\mathbf{x}) = \mu_0(\mathbf{x}) + \sum_{i=1}^{N} k_0(\mathbf{x}, \mathbf{x}_i) q_i = \mu_0(\mathbf{x}) + \boldsymbol{k}_\mathbf{x}^T \boldsymbol{q}, \qquad (2.75)$$

$$k_{post}(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') + \sum_{i,j=1}^{N} k_0(\mathbf{x}, \mathbf{x}_i) R_{ij} k_0(\mathbf{x}_j, \mathbf{x}')$$

$$= k_0(\mathbf{x}, \mathbf{x}') + \boldsymbol{k}_{\mathbf{x}}^T \boldsymbol{R}\, \boldsymbol{k}_{\mathrm{x}'} \,. \quad (2.76)$$

The parameters q_i and R_{ij} are given by:

$$q_i = \frac{\partial}{\partial E_0\left[f_{\mathbf{x}_i}\right]} ln \int p(D|\, \mathbf{f}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D, \quad (2.77)$$

$$R_{ij} = \frac{\partial^2}{\partial E_0\left[f_{\mathbf{x}_i}\right] \partial E_0\left[f_{\mathbf{x}_j}\right]} ln \int p(D|\, \mathbf{f}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D, \quad (2.78)$$

where $\mathbf{f}_D = \left[f_{\mathbf{x}_1}, f_{\mathbf{x}_2}, \ldots, f_{\mathbf{x}_N}\right]^T$, and the partial derivatives are calculated with respect to the prior mean of the GP at the $\mathbf{x}_i$ points. In the case of a Gaussian likelihood $p(D|\, \mathbf{f}_D)$, the predictive formulas from the posterior GP are written as follows:

Given the set *D* as defined above and an arbitrary point $\mathbf{x}_*$, we have that $f_{\mathbf{x}_*}|D, \mathbf{x}_* \sim \mathcal{N}(\mu_*, \sigma_*^2)$, with the posterior moments calculated as:

$$\mu_* = \mu_0(\mathbf{x}_*) + \boldsymbol{k}_*^T \boldsymbol{\alpha} \,, \quad (2.79)$$

$$\sigma_*^2 = k_0(\mathbf{x}_*, \mathbf{x}_*) + \boldsymbol{k}_*^T \boldsymbol{C}\, \boldsymbol{k}_* \,, \quad (2.80)$$

where

$$\boldsymbol{k}_* = \left(k_0(\mathbf{x}_*, \mathbf{x}_1), \ldots, k_0(\mathbf{x}_*, \mathbf{x}_N)\right)^T, \quad (2.81)$$

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \sigma^2 I)^{-1}\left(\mathbf{y} - \mu_0(\boldsymbol{X})\right) \,, \quad (2.82)$$

$$\boldsymbol{C} = -(\boldsymbol{K} + \sigma^2 I)^{-1} \,, \quad (2.83)$$

and $\boldsymbol{K} = \left(k_0(\mathbf{x}_i, \mathbf{x}_j)\right)_{i,j}$ is an *N* x *N* matrix. The posterior variance σ_*^2 is smaller than the prior variance of $f_{\mathbf{x}_*}$, because the matrix $\boldsymbol{C}$ is negative definite (Rasmussen & Williams, 2006). This reflects the reduction in uncertainty that is achieved by learning the training data. We can establish a similarity between the geometric approach of kernel methods (SVM-like algorithms) and GPs for regression as described here, by noting that the predictive mean μ_* becomes a linear combination of *N* kernel functions, each centered on a training data point, when $\mu_0(\mathbf{x}_*) = 0$ (i.e.

when a prior GP with zero mean is used). Additionally, the predictive distribution of the target value y_* can be obtained by just adding the noise variance σ^2 to the expression of σ_*^2; i.e. $y_*|D, \mathbf{x}_* \sim \mathcal{N}(\mu_*, \sigma_*^2 + \sigma^2)$.

Note that obtaining the posterior batch GP includes the inversion of a $N \times N$ covariance matrix in order to calculate the matrix $\boldsymbol{C}$. That operation has a computational complexity of $O(N^3)$ if we used a straightforward algorithm for matrix inversion. This complexity is a serious limitation when we have to work with large data sets. In that case, we can use matrix inversion algorithms that are slightly faster when used with large matrices. For instance, the popular Strassen algorithm has a computational complexity of approximately $O(N^{2.8074})$ (Strassen, 1969). However, all practical matrix inversion algorithms known to the author has computational complexities that are greater than $O(N^2)$ and tend to significantly deviate from $O(N^3)$ only for very large matrices, typically in the order of thousands. For the purposes of this dissertation, we consider that training a batch GP has a computational complexity of $O(N^3)$. It can be easily seen that evaluating the GP model in a new data point $\mathbf{x}_*$ has $O(N^2)$ computational complexity. Finally, note that batch GP has a space complexity of $O(N^2)$ as well, based on the need to keep the matrix $\boldsymbol{C}$ in memory.

The marginal likelihood $p(\mathbf{y}|\boldsymbol{X})$ is typically used for selecting appropriate values for kernel hyperparameters. The marginal likelihood is expressed as the integral of the likelihood times the prior distribution:

$$p(\mathbf{y}|\boldsymbol{X}) = \int p(\mathbf{y}|\mathbf{f}_D, \boldsymbol{X})\, p(\mathbf{f}_D|\boldsymbol{X}) d\mathbf{f}_D \,. \qquad (2.84)$$

Given that the marginal likelihood is typically used as an argument for optimization problems, any monotonic function of it is equally useful for hyperparameter selection. For operational

convenience, $\log p(\mathbf{y}|\boldsymbol{X})$ is commonly used instead. The likelihood $\mathbf{y}|\mathbf{f}_D \sim \mathcal{N}(\mathbf{f}_D, \sigma^2 I)$, and $\mathbf{f}_D|\boldsymbol{X}$ has a $\mathcal{N}(\mu_0(\boldsymbol{X}), \boldsymbol{K})$ distribution, where $\boldsymbol{K} = k_0(\boldsymbol{X}, \boldsymbol{X})$. By applying a classic result from multivariate statistics stating that the multiplication of two multivariate Gaussian distributions is also a Gaussian distribution, integrating with respect to $\mathbf{f}_D$, and applying logarithm, the following expression is obtained for the log marginal likelihood (Rasmussen & Williams, 2006):

$$\begin{aligned}\text{Log}\, p(\mathbf{y}|\boldsymbol{X}) = &-\frac{1}{2}\big(\mathbf{y} - \boldsymbol{\mu}_0(\boldsymbol{X})\big)^T [\boldsymbol{K} + \sigma^2 I]^{-1}\big(\mathbf{y} - \boldsymbol{\mu}_0(\boldsymbol{X})\big) \\ &-\frac{1}{2} log|\boldsymbol{K} + \sigma^2 I| - \frac{N}{2} log(2\pi)\,. \end{aligned} \qquad (2.85)$$

2.3.2.2 Online GP

The parametrisation lemma allows us to avoid integration for doing predictions. However, to calculate the coefficients q_i and R_{ij} using a batch learning approach we still have to deal with *N*-dimensional integrals. Additionally, if not all training data were known in advance then incremental learning becomes the ideal approach. A solution is to employ an online learning approach to obtain a sequence of approximated posterior processes, learning from one observation $(\mathbf{x}_t, \mathrm{y}_t)$ at a time (Opper M. , 1998). Online GP (Csató & Opper, 2002) uses this approach. It assumes that the data are conditionally independent, and thus the likelihood can be expressed as:

$$p(D|\,\mathbf{f}_D) = \prod_{i=1}^{N} p\big(\mathrm{y}_i|f_{\mathbf{x}_i}\big)\,. \qquad (2.86)$$

At any step *t*, the Gaussian approximation obtained after learning observation $(\mathbf{x}_t, \mathrm{y}_t)$ is denoted by $\hat{p}_t(\mathbf{f})$; where the hat denotes approximation of the posterior to the closest GP when the likelihood is not Gaussian (no approximation is needed when the likelihood is Gaussian). The

posterior GP at step $t+1$, denoted by $\hat{p}_{t+1}(\mathbf{f})$, is estimated as the Gaussian distribution closest to the following Bayesian update:

$$p_{t+1}(\mathbf{f}) = \frac{p(\mathrm{y}_{t+1}|\mathbf{f})\,\hat{p}_t(\mathbf{f})}{\int p(\mathrm{y}_{t+1}|\mathbf{f})\,\hat{p}_t(\mathbf{f})d\mathbf{f}} = \frac{p(\mathrm{y}_{t+1}|\mathbf{f})\,\hat{p}_t(\mathbf{f})}{E_t[p(\mathrm{y}_{t+1}|\mathbf{f})]}\,. \qquad (2.87)$$

It can be assumed without loss of generality that $\mathbf{f}$ contains f_{t+1}, so that the likelihood $p(\mathrm{y}_{t+1}|\mathbf{f}) = p(\mathrm{y}_{t+1}|f_{t+1})$. Applying the parametrisation lemma sequentially, using at each step $t+1$ the prior GP $\hat{p}_t(\mathbf{f})$ and the likelihood $p(\mathrm{y}_{t+1}|f_{t+1})$, the following recursive expressions are obtained for the moments of the posterior GP at step $t+1$:

$$\mu_{t+1}(\mathbf{x}) = \mu_t(\mathbf{x}) + k_t(\mathbf{x}, \mathbf{x}_{t+1})q_{t+1}\,, \qquad (2.88)$$

$$k_{t+1}(\mathbf{x}, \mathbf{x}') = k_t(\mathbf{x}, \mathbf{x}') + k_t(\mathbf{x}, \mathbf{x}_{t+1})r_{t+1}k_t(\mathbf{x}_{t+1}, \mathbf{x}')\,. \qquad (2.89)$$

The parameters q_{t+1} and r_{t+1}, which depend on the likelihood, are given by:

$$q_{t+1} = \frac{\partial}{\partial E_t[f_{t+1}]}\, ln\, E_t[p(\mathrm{y}_{t+1}|f_{t+1})]\,, \qquad (2.90)$$

$$r_{t+1} = \frac{\partial^2}{\partial E_t[f_{t+1}]}\, ln\, E_t[p(\mathrm{y}_{t+1}|f_{t+1})]\,. \qquad (2.91)$$

By unfolding the previous recursive equations, the iterative formulations for estimating the moments of the online GP at step t are written as follows:

$$\mu_t(\mathbf{x}) = \mu_0(\mathbf{x}) + \boldsymbol{k}_{\mathbf{x}}^T\boldsymbol{\alpha}_t\,, \qquad (2.92)$$

$$k_t(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') + \boldsymbol{k}_{\mathbf{x}}^T\boldsymbol{C}_t\boldsymbol{k}_{\mathbf{x}'}\,, \qquad (2.93)$$

where $\boldsymbol{k}_{\mathbf{x}} = \big(k_0(\mathbf{x}, \mathbf{x}_1), \ldots, k_0(\mathbf{x}, \mathbf{x}_t)\big)^T$, the vector $\boldsymbol{\alpha}_t$ is an approximation to the first t coefficients in $\boldsymbol{q}$, and $\boldsymbol{C}_t$ is an approximation to the first $t \times t$ coefficients in $\boldsymbol{R}$. The expressions to calculate $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ at each step $t+1$ are listed in Figure 2.1, which contains Matlab-like pseudocode for the function that learns a new data point $(\mathbf{x}_{t+1}, \mathrm{y}_{t+1})$. The online GP algorithm

starts with $\boldsymbol{\alpha} = \boldsymbol{C} = 0$. At each step $t+1$, it recursively updates the vector $\boldsymbol{\alpha}$ and the matrix $\boldsymbol{C}$. The Online GP adds each new observation to a set of learned vectors. The algorithm is fully described in (Csató & Opper, 2002).

For the particular case of a GP with a Gaussian likelihood, the parameters q_{t+1} and r_{t+1} are expressed as:

$$q_{t+1} = \frac{\mathrm{y}_{t+1} - m_{t+1}}{\sigma_{t+1}^2 + \sigma^2}, \quad (2.94)$$

$$r_{t+1} = -\frac{1}{\sigma_{t+1}^2 + \sigma^2}, \quad (2.95)$$

where

$$m_{t+1} = E_t[f_{t+1}] = \mu_0(\mathbf{x}_{t+1}) + \boldsymbol{k}_{\mathbf{x}_{t+1}}^T \boldsymbol{\alpha}_t, \quad (2.96)$$

$$\sigma_{t+1}^2 = k_0(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) + \boldsymbol{k}_{\mathbf{x}_{t+1}}^T \boldsymbol{C}_t \boldsymbol{k}_{\mathbf{x}_{t+1}}. \quad (2.97)$$

Similar to batch GP, online GP has a space complexity of $O(N^2)$. Training an online GP model on *N* data points also has a computational complexity of $O(N^3)$. This can be seen in Figure 2.1, where calculating the matrix $\boldsymbol{C}$ at each particular step $t+1$ has a computational complexity of $O((t+1)^2)$. Summing these quadratic complexities for the *N* learning steps leads to $O(N^3)$. Finally, given that the equations to calculate the posterior moments for a new observation $\mathbf{x}_*$ remain the same as in batch GP, the predictive operation is $O(N^2)$ as well.

```
% The obj parameter stands for an instance of our OnlineGP class.
% Symbol 0_t stands for a zero column vector of length t.
function trainOnline(obj, x_{t+1}, y_{t+1})
        t ← obj.Size;     % t = Number of points learned so far.
        obj.addObservationToLearnedVectors(x_{t+1}, y_{t+1});
        Calculate q_{t+1} and r_{t+1} according to likelihood model.

       if (t == 0)
         % First data point to learn.
         α ← q_{t+1}
         C ← r_{t+1}
        else
          k_{x_{t+1}} ← (k_0(x_{t+1}, x_1), ..., k_0(x_{t+1}, x_t))^T
          s_{t+1} ← [C k_{x_{t+1}}; 1]
          α ← [α; 0] + q_{t+1} s_{t+1}
          C ← [C  0_t; 0_t^T  0] + r_{t+1} s_{t+1} s_{t+1}^T
        end
end
```

Figure 2.1: Matlab-like pseudocode for the Online GP training algorithm.

2.3.2.3 Sparse Online GP

As noted in previous sections, training batch GP and online GP involve a computational complexity of $O(N^3)$. This complexity is a serious limitation when we have to work with large data sets. Furthermore, given the quadratic space complexity of batch GP and online GP, we might find GPs infeasible for some relatively large data sets. Consequently, several approximation techniques for GP modeling have been devised in recent years. These approximation techniques are typically based on finding a small subset of the training data that is representative of the process to be learned, which leads to a sparse knowledge representation (Seeger, Williams, & Lawrence, 2003), (Tresp, 2001), (Williams & Seeger, 2001), (Smola & Bartlett, 2001) (Csató & Opper, 2002), (Gibbs, 1997). This subsection describes the sparse

online GP (SOGP) method (Csató & Opper, 2002), which is employed in the theoretical and experimental sections of our work.

SOGP overcomes memory limitations by having a capacity parameter m that determines the maximum number of most relevant observations to keep in memory. The set of most relevant observations is called the *BV* set. SOGP has a much better space complexity than GP: $O(m^2)$ instead of $O(N^2)$. Additionally, SOGP modeling achieves a time complexity that is linear with respect to the data size: $O(Nm^2)$ (Csató, 2002).

The SOGP algorithm takes into account the representation of the input vectors $\mathbf{x}$ through a mapping ϕ into a feature space $\mathcal{F}$, which is typically of much higher dimension than the original space $\mathcal{X}$. The mapping $\phi: \mathcal{X} \to \mathcal{F}$ is given implicitly through the kernel function, such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$. Let us assume that after learning the first t observations $\{(\mathbf{x}_i, \mathrm{y}_i), i = 1,2, \dots, t\}$, the *BV* set kept the vectors $\{(\mathbf{x}_{i_1}, \mathrm{y}_{i_1}), (\mathbf{x}_{i_2}, \mathrm{y}_{i_2}), \dots, (\mathbf{x}_{i_r}, \mathrm{y}_{i_r})\}$, where $\{i_1, i_2, \dots, i_r\} \subseteq \{1,2, \dots, t\}$. Given a new training observation $(\mathbf{x}_{t+1}, \mathrm{y}_{t+1})$, the feature vector $\phi_{t+1} = \phi(\mathbf{x}_{t+1})$ is decomposed as:

$$\phi_{t+1} = \hat{\phi}_{t+1} + v_{res} = \boldsymbol{\Phi}_t \hat{\boldsymbol{e}}_{t+1} + \sqrt{\gamma_{t+1}} \phi_{res} , \qquad (2.98)$$

where $\hat{\phi}_{t+1}$ denotes the orthogonal projection of ϕ_{t+1} onto the span of the *BV* set; $\boldsymbol{\Phi}_t = [\phi_{i_1}, \phi_{i_2}, \dots, \phi_{i_r}]$; ϕ_{res} denotes the corresponding unit vector orthogonal to the space spanned by $\boldsymbol{\Phi}_t$; and $\gamma_{t+1} = \left\| \phi_{t+1} - \hat{\phi}_{t+1} \right\|^2$. The coordinates $\hat{\boldsymbol{e}}_{t+1} = \boldsymbol{K}_t^{-1} \boldsymbol{k}_{\mathbf{x}_{t+1}}$, where $\boldsymbol{K}_t$ denotes the $r \times r$ covariance matrix of the vectors in $\boldsymbol{\Phi}_t$, and $\boldsymbol{k}_{\mathbf{x}_{t+1}} = \left(k(\mathbf{x}_{t+1}, \mathbf{x}_{i_1}), \dots, k(\mathbf{x}_{t+1}, \mathbf{x}_{i_r}) \right)^T$. Given that the inversion of $\boldsymbol{K}_t$ at each learning step is an expensive operation, the SOGP

algorithm maintains a variable $\boldsymbol{Q}_t = \boldsymbol{K}_t^{-1}$, which is updated recursively when a new vector is added to the *BV* set:

$$\boldsymbol{Q}_{t+1} = \begin{bmatrix} \boldsymbol{Q}_t & \mathbf{0}_t \\ \mathbf{0}_t^T & 0 \end{bmatrix} + \gamma_{t+1}^{-1} \begin{bmatrix} \hat{\boldsymbol{e}}_{t+1} \\ -1 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{e}}_{t+1} \\ -1 \end{bmatrix}^T , \qquad (2.99)$$

where $\mathbf{0}_t$ stands for a zero column vector of length *t*. Finally, $\gamma_{t+1} = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \boldsymbol{k}_{\mathbf{x}_{t+1}}^T \boldsymbol{Q}_t \boldsymbol{k}_{\mathbf{x}_{t+1}}$. The remainder of this section briefly describes the SOGP learning algorithm.

SOGP learns the first observation $(\mathbf{x}_1, \mathrm{y}_1)$ by adding it to the *BV* set and initializing variables as follows: $\boldsymbol{\alpha} = q_1,\ \boldsymbol{C} = r_1,\ \boldsymbol{Q} = k(\mathbf{x}_1,\ \mathbf{x}_1)^{-1}$. At each subsequent step $t+1$, the procedure to learn observation $(\mathbf{x}_{t+1}, \mathrm{y}_{t+1})$ depends on γ_{t+1}. If $\gamma_{t+1} \geq \epsilon$, where ϵ is some small tolerance, then $(\mathbf{x}_{t+1}, \mathrm{y}_{t+1})$ is added to the *BV* set and a full online update is executed, as done in online GP. The matrix $\boldsymbol{Q}$ is also updated using equation (2.99). On the other hand, if $\gamma_{t+1} < \epsilon$ then SOGP learns from the projection $\hat{\phi}_{t+1}$, disregarding the residual vector v_{res}. The *BV* set is not altered in that case and variables $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ are updated without increasing their sizes:

$$\eta_{t+1} \leftarrow (1 + \gamma_{t+1} r_{t+1})^{-1} , \qquad (2.100)$$

$$\mathbf{s}_{t+1} \leftarrow \boldsymbol{C}\boldsymbol{k}_{\mathbf{x}_{t+1}} + \hat{\boldsymbol{e}}_{t+1} , \qquad (2.101)$$

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + q_{t+1} \eta_{t+1} \mathbf{s}_{t+1} , \qquad (2.102)$$

$$\boldsymbol{C} \leftarrow \boldsymbol{C} + r_{t+1} \eta_{t+1} \mathbf{s}_{t+1} \mathbf{s}_{t+1}^T . \qquad (2.103)$$

Let us assume that a full online update occurred at certain step $t+1$ and the size of the *BV* set went over its capacity *m*. In that case, the GP needs to be "pruned". Let us denote the GP at that step by GP_{t+1}. Pruning is done by removing from the *BV* set the basis vector that contributes the least to the GP representation, carrying out a recomputation (and the corresponding reduction in size) of $\boldsymbol{\alpha}$, $\boldsymbol{C}$ and $\boldsymbol{Q}$. Following (Csató, 2002), the pruning formulas are written here such that the

basis vector occupying the $m+1$ position is the one to be removed. However, a vector at any position *i* can be removed from the *BV* set using the same formulas with index *i* instead. The removal of a basis vector is done in a way that the model retains as much information from it as possible.

The problem of removing the $m+1$-th basis vector from the *BV* set is solved by approximating GP_{t+1} by the Gaussian process $\widehat{GP}_{t+1}$ that has the minimum Kullback-Leibler distance $KL(GP||\, GP_{t+1})$ to GP_{t+1}, among all SOGPs containing the same *BV* set as GP_{t+1} and having the coefficients corresponding to the $m+1$-th basis vector equal to zero. This optimization problem leads to the following equations to obtain $\widehat{GP}_{t+1}$:

$$\widehat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}^{(m)} - \frac{\alpha^*}{c^*+q^*}(\boldsymbol{Q}^* + \boldsymbol{C}^*)\,, \tag{2.104}$$

$$\widehat{\boldsymbol{C}}_{t+1} = \boldsymbol{C}^{(m)} + \frac{\boldsymbol{Q}^*\boldsymbol{Q}^{*T}}{q^*} - \frac{(\boldsymbol{Q}^*+\boldsymbol{C}^*)(\boldsymbol{Q}^*+\boldsymbol{C}^*)^T}{q^*+c^*}\,, \tag{2.105}$$

$$\widehat{\boldsymbol{Q}}_{t+1} = \boldsymbol{Q}^{(m)} - \frac{\boldsymbol{Q}^*\boldsymbol{Q}^{*T}}{q^*}\,, \tag{2.106}$$

where the different terms are obtained from partitioning $\boldsymbol{\alpha}_{t+1}$, $\boldsymbol{C}_{t+1}$, and $\boldsymbol{Q}_{t+1}$ as follows:

$$\boldsymbol{\alpha}_{t+1} = \begin{bmatrix} \boldsymbol{\alpha}^{(m)} \\ \alpha^* \end{bmatrix}, \tag{2.107}$$

$$\boldsymbol{C}_{t+1} = \begin{bmatrix} \boldsymbol{C}^{(m)} & \boldsymbol{C}^* \\ \boldsymbol{C}^{*T} & c^* \end{bmatrix}\,, \tag{2.108}$$

$$\boldsymbol{Q}_{t+1} = \begin{bmatrix} \boldsymbol{Q}^{(m)} & \boldsymbol{Q}^* \\ \boldsymbol{Q}^{*T} & q^* \end{bmatrix}. \tag{2.109}$$

In order to decide which basis vector to remove, an "importance" score is computed for each *i*-th vector, equal to the error corresponding to using the approximation $\widehat{GP}_{t+1}$:

$$\varepsilon_{t+1}(i) = \frac{\left(\boldsymbol{\alpha}_{t+1}(i)\right)^2}{q(i)+c(i)} - \frac{s(i)}{q(i)} + ln\left(1 + \frac{c(i)}{q(i)}\right), \qquad (2.110)$$

where $q(i)$, $\mathrm{c}(i)$ and $s(i)$ are the i-th diagonal elements of the matrices $\boldsymbol{Q}$, $\boldsymbol{C}$ and $\boldsymbol{S}_{t+1} = (\boldsymbol{C}_{t+1}^{-1} + \boldsymbol{K}_{t+1})^{-1}$, respectively. It was shown in (Csató, 2002) that the error $\varepsilon_{t+1}(i)$ can be effectively approximated by:

$$\hat{\varepsilon}_{t+1}(i) = \frac{\left(\boldsymbol{\alpha}_{t+1}(i)\right)^2}{q(i)+c(i)}, \qquad (2.111)$$

which is the expression employed in the experimental section of our work. The SOGP algorithm is summarized in the pseudo-code shown in Figure 2.2 below.

2.3.3 Gaussian Processes for Binary Classification

This section describes the use of GPs for solving binary classification problems. The goal in this case is to find a discriminative model for the posterior probability $p(\mathrm{y}_{N+1} = +1|D, \mathbf{x}_{N+1})$ given a data set $D = \{\boldsymbol{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i)|\mathbf{x}_i \epsilon \mathcal{X},\ \mathrm{y}_i \epsilon \{+1, -1\}, i = 1, \dots, N\}$, where $\mathcal{X}$ denotes a finite-dimensional space. Note that only the conditional probability of y = +1 needs to be modeled because $p(\mathrm{y}_{N+1} = -1|D, \mathbf{x}_{N+1}) = 1 - p(\mathrm{y}_{N+1} = +1|D, \mathbf{x}_{N+1})$.

The main limitation to use a GP in this case is that the predictions of a GP are defined over the set of real values, whereas the response value for classification is limited to the interval [0, 1]. The easiest way to overcome this limitation is to apply a sigmoid function $\sigma(a)$ to the outcome of a GP in order to get values in [0, 1]. First, a GP prior is defined over a space of latent functions $f: \mathcal{X} \to \mathbb{R}$. In this case, a noise-free model is typically assumed, i.e. $\mathrm{y}_i = f(\mathbf{x}_i)$. Consequently, the GP prior can be expressed as follows for the training data set:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, C), \qquad (2.112)$$

where $C(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$ is the noise-free covariance matrix. However, it is common to define the covariance matrix C as $C(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \nu\delta_{ij}$, were ν is a noise-like parameter that ensures that C is a positive definite matrix (i.e. $\boldsymbol{z}^T C \boldsymbol{z} > 0$ for every non-zero vector $\boldsymbol{z}$).

```
% The obj parameter stands for an instance of SOGP class.
% Variables α and C are properties of the OnlineGP, which are
initially empty.
function trainOnline(obj, x_{t+1}, y_{t+1})
  t ← obj.Size;
  Calculate q_{t+1} and r_{t+1} according to likelihood model.

  if (t == 0)
    % First data point to learn.
    sparseUpdateAllowed = false;
    s_{t+1} ← 1
    ê_{t+1} ← [ ]
    γ_{t+1} ← [ ]
  else
    k_{x_{t+1}} ← (k(x_{t+1}, x_{i_1}), ..., k(x_{t+1}, x_{i_r}))^T
    ê_{t+1} ← Q k_{x_{t+1}}
    γ_{t+1} ← k(x_{t+1}, x_{t+1}) − k_{x_{t+1}}^T ê_{t+1}

    if (γ_{t+1} < ϵ)
      sparseUpdateAllowed = true;
      s_{t+1} ← C k_{x_{t+1}} + ê_{t+1}
    else
      sparseUpdateAllowed = false;
      s_{t+1} ← [C k_{x_{t+1}}; 1]
    end
  end

  if (sparseUpdateAllowed == true)
      obj.runSparseUpdate(s_{t+1}, q_{t+1}, r_{t+1}, γ_{t+1});
  else
      obj.runFullUpdate(x_{t+1}, y_{t+1}, s_{t+1}, q_{t+1}, r_{t+1}, γ_{t+1}, ê_{t+1});
  end
end

function runSparseUpdate(obj, s_{t+1}, q_{t+1}, r_{t+1}, γ_{t+1});
  % Size didn't change. No need to modify Q and BV set.
  η_{t+1} ← (1 + γ_{t+1} r_{t+1})^{-1}
  α ← α + q_{t+1} η_{t+1} s_{t+1}
  C ← C + r_{t+1} η_{t+1} s_{t+1} s_{t+1}^T
end
```

```
function runFullUpdate(obj, x_{t+1}, y_{t+1}, s_{t+1}, q_{t+1}, r_{t+1}, γ_{t+1}, ê_{t+1})
  t ← obj.Size;
  if (t == 0)
      % First data point to learn.
      α ← q_{t+1}
      C ← r_{t+1}
      Q ← k(x_{t+1}, x_{t+1})^{-1}
  else
      α ← [α; 0] + q_{t+1} s_{t+1}
      C ← [C  0_t; 0_t^T  0] + r_{t+1} s_{t+1} s_{t+1}^T
      Q ← [Q  0_t; 0_t^T  0] + γ_{t+1}^{-1} [ê_{t+1}; −1] [ê_{t+1}; −1]^T
  end

  obj.addObservationToBVSet(x_{t+1}, y_{t+1});

  if ((t + 1) > obj.Capacity)
     obj.prune();
  end
end


function prune(obj)
  Calculate scores ε_{t+1}(i) for all basis vectors.
  i ← index of basis vector with the smallest score.
  Swap (i-th basis vector, m+1-th basis vector)

  α ← α^(m) − (α*/(c*+q*)) (Q* + C*)
  C ← C^(m) + (Q*Q*^T)/q* − ((Q*+C*)(Q*+C*)^T)/(q*+c*)
  Q ← Q^(m) − (Q*Q*^T)/q*
end
```

Figure 2.2: Matlab-like pseudocode for the SOGP training algorithm.

Let us consider any function *f* sampled from the GP. The solution to the classification problem is based on a deterministic function of *f*, which is built through a sigmoid function $\sigma(a)$:

$$\pi_f(\mathbf{x}) \equiv p(\mathrm{y} = +1|\mathbf{x}) = \sigma\big(f(\mathbf{x})\big)\,. \qquad (2.113)$$

Note that π_f is a non-Gaussian stochastic process over the space of functions $\{g|g\colon \mathbb{R} \to [0,1]\}$. Although the function *f* is shown alone here, it is important to keep in mind that its distribution is conditioned on any observed data $\boldsymbol{X}$, $\mathbf{y}$. Inference for classification is done in two steps:

STEP 1: The posterior distribution of the latent GP $p(f_{N+1}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})$ is determined as follows:

$$\begin{aligned} p(f_{N+1}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1}) &= \int p(f_{N+1}, \mathbf{f}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})d\mathbf{f} \\ &= \tfrac{1}{p(\mathbf{y})}\int p(\mathbf{y}|\mathbf{f}, f_{N+1}, \boldsymbol{X}, \mathbf{x}_{N+1})p(f_{N+1}, \mathbf{f}|\boldsymbol{X}, \mathbf{x}_{N+1})d\mathbf{f} \\ &= \tfrac{1}{p(\mathbf{y})}\int p(\mathbf{y}|\mathbf{f}, \boldsymbol{X})p(\mathbf{f}|\boldsymbol{X})p(f_{N+1}|\mathbf{f}, \boldsymbol{X}, \mathbf{x}_{N+1})d\mathbf{f} \\ &= \int p(\mathbf{f}|\mathbf{y}, \boldsymbol{X})p(f_{N+1}|\mathbf{f}, \boldsymbol{X}, \mathbf{x}_{N+1})d\mathbf{f}\,. \qquad (2.114) \end{aligned}$$

In the previous derivation $\mathbf{f}$ denotes a vector taking values in $\mathbb{R}^N$. The conditional distribution $p(f_{N+1}|\mathbf{f}, \boldsymbol{X}, \mathbf{x}_{N+1})$ is obtained as the posterior GP, using the same equations as for the regression case:

$$f_{N+1}|\mathbf{f}, \boldsymbol{X}, \mathbf{x}_{N+1} \sim \mathcal{N}(C_{N+1}^T C^{-1}\mathbf{f}, C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) - C_{N+1}^T C^{-1} C_{N+1})\,, \qquad (2.115)$$

where C_{N+1} denotes the column vector $k(\boldsymbol{X}, \mathbf{x}_{N+1})$. However, the posterior distribution $p(\mathbf{f}|\boldsymbol{X}, \mathbf{y})$ is not Gaussian. This makes the posterior distribution $p(f_{N+1}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})$ a non-Gaussian stochastic process.

Having a Gaussian posterior for f_{N+1} facilitates further analytical treatment. Three approaches to obtain a Gaussian approximation to this posterior have been proposed. One technique makes use of local variational bounds on logistic sigmoid functions (Gibbs & MacKay, 2000). A second approach employs an approximation technique called *expectation propagation* (Opper & Winther, 2000) (Minka, 2001) (Seeger M. , 2003). The third approach consists of obtaining a Gaussian Laplace approximation to the posterior $p(\mathbf{f}|\mathbf{y}, \boldsymbol{X})$, and then approximating the posterior distribution of f_{N+1} as the integral of two Gaussian distributions (Rasmussen & Williams, 2006).

STEP 2: The expected value of $\pi_f(\mathbf{x})$ is calculated according to the following expression:

$$\bar{\pi}_f(\mathbf{x}_{N+1}) = \mathbb{E}\left[\pi_f(\mathbf{x}_{N+1})\right] = \mathbb{E}_{f_{N+1}}[p(\mathrm{y}_{N+1} = +1 \mid \boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})]$$

$$= \int p(\mathrm{y}_{N+1} = +1 \mid f_{N+1}, \boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1}) p(f_{N+1}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})\, df_{N+1}$$

$$= \int \sigma(f_{N+1}) p(f_{N+1}|\boldsymbol{X}, \mathbf{y}, \mathbf{x}_{N+1})\ df_{N+1}\,. \qquad (2.116)$$

If we employ the approximated posterior distribution from STEP 1 then $\bar{\pi}_f(\mathbf{x})$ can be obtained by using well-established results that allow approximating the convolution of a sigmoid function (e.g., the cumulative Gaussian or the logistic sigmoid function) and a Gaussian function (Spiegelhalter & Lauritzen, 1990). Alternatively, $\bar{\pi}_f(\mathbf{x})$ can be obtained using Monte Carlo sampling methods (Neal, 1997).

2.3.4 Gaussian Processes for Novelty Detection

The use of GP regression and GP binary classification as novelty detection techniques was originally proposed in (Kemmler, Rodner, & Denzler, 2010). That work was subsequently

expanded in (Kemmler M. , Rodner, Wacker, & Denzler, 2013), where the authors provided links between their approach and other algorithms, and offered multiple experimental results. The training data in this case are denoted by $D = \{\boldsymbol{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) | \mathbf{x}_i \epsilon \mathcal{X},\ \mathrm{y}_i = 1,\ i = 1, \dots, N\}$. As noted in (Kemmler M. , Rodner, Wacker, & Denzler, 2013), two problems appear when attempting to use posterior GPs for novelty detection: (1) GPs are not designed to estimate the probability density of the input data (as done by other statistical methods employed in novelty detection) and (2) applying a regression technique to a data set in which the dependent variable y is constant should lead to a constant regression function, which is the simplest model that fits the data. These problems are circumvented by using a prior GP with a zero-mean prior GP. If a smooth kernel function *k* is used, then functions *f* sampled from the posterior GP will be smooth and will evaluate to zero or near-zero at data points that are distant from training observations, while evaluating close to 1 at points near those in D. In other words, after training the GP on *D*, if a test observation $\mathbf{x}_*$ is very near to points in *D* then the corresponding posterior mean μ_* will be close to 1, but μ_* will be close to zero for data points that are distant from training observations. In a similar fashion, the posterior variance of the GP (σ_*^2) will be greater for observations that are increasingly distant from points in D. Consequently, class membership scores based on the posterior mean μ_*, the posterior variance σ_*^2, or a combination of both, can be used to detect novel observations. The lower the membership score of a given input $\mathbf{x}_*$, the higher the likelihood of $\mathbf{x}_*$ being an outlier.

Table 2.2 lists the four measures proposed in (Kemmler, Rodner, & Denzler, 2010) and (Kemmler M. , Rodner, Wacker, & Denzler, 2013): Probability (*P*), Mean (*M*), Negative Variance (*V*) and Heuristic (*H*). The score *V* was previously proposed as part of a clustering

technique in (Kim & Lee, 2006), and score *H* was successfully applied to object categorization in (Kapoor, Grauman, Urtasun, & Darrell, 2010). Note that the probability score *P*, despite its name, is actually the value at y = 1 of the posterior probability density function of y_*.

Table 2.2: Membership scores for novelty detection using Gaussian processes. Table taken from (Kemmler, Rodner, & Denzler, 2010).

Membership score	**Expression**
Probability (*P*)	$p(y_* = 1 \mid \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)$
Mean (*M*)	$\mu_* = E[y_* \mid \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*]$
Negative Variance (*V*)	$-\sigma_*^2 = -Var(y_* \mid \boldsymbol{X}, \mathbf{y}, \mathbf{x}_*)$
Heuristic (*H*)	$\mu_* \, \sigma_*^{-1}$

The experiments in (Kemmler, Rodner, & Denzler, 2010) compared these membership scores using GP regression (GP-Reg) and approximated binary GP classification using both Laplace approximation (LA) and expectation propagation (EP). Additionally, the work in (Kemmler, Rodner, & Denzler, 2010) compared GP-based novelty detection using these membership scores to Support Vector Data Descriptor (Tax & Duin, 2004). The corresponding experiments were run on all object categories (classes) of the Caltech 101 image database (Fei-Fei, Fergus, & Perona, 2004), where one class at a time was used as the target class. Each experiment assessed the performances of SVDD and each GP-based scoring method on a different target class. Subsequently, an average performance value was obtained for each detection method by averaging the corresponding performance values across all classes. Note that no detailed

analyses of performance on individual image categories were offered in (Kemmler, Rodner, & Denzler, 2010). Two image-based kernel functions were employed: the pyramid of oriented gradients (PHoG) (Bosch, Zisserman, & Munoz, 2007) and the spatial pyramid matching (SPM) kernel (Lazebnik, Schmid, & Ponce, 2006).

According to the analysis done in (Kemmler, Rodner, & Denzler, 2010), scores using GP regression (GP-Reg-*P*, GP-Reg-*M*, GP-Reg-*V*, GP-Reg-*H*) performed consistently similar or better than the corresponding scores based on approximate GP classification with LA and EP. Additionally, performance values obtained through the SPM kernel were consistently higher across all methods than the corresponding performance values obtained through the PHoG kernel. That motivated Kemmler et al. to focus their conclusions on results obtained when using the SPM kernel. Average performance values from GP regression were better than those obtained from SVDD for all membership scores except when using GP-Reg-*M* (the GP-Reg-*M* score showed a great variation in performance across image categories). In particular, novelty detection based on GP-Reg-*V* consistently outperformed all other methods on the Caltech 101 data set.

As mentioned above, the experimental work described in (Ramirez-Padron, Mederos, & Gonzalez, 2013) shows that the performance of novelty detection methods based on online GPs can be similar to the performance of batch GP-based novelty detection. Interestingly, it was also reported in that work that the probability score (P) and the heuristic score (*H*) consistently outperformed the other two scores. This result suggests that scores that combine the posterior

mean and the posterior variance of the GP might be better fitted to GP-based novelty detection that scores employing each of these statistics alone.

The high performance of GP-based novelty detection has also been shown in other domains aside of visual object recognition. For instance, GP-based novelty detection outperformed widely popular methods like Gaussian mixture models, Parzen density estimation and SVDD in doing defect detection in wire ropes, novel bacteria identification based on Raman spectroscopy, attribute prediction, and background subtraction (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Additionally, GP-based novelty detection has been reported as a very accurate technique for doing video segmentation through event detection, using a frame-by-frame processing approach (Krishna, Bodesheim, & Denzler, 2013), (Krishna, Bodesheim, Körner, & Denzler, 2014). It was established in (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001) that SVDD is equivalent to one-class SVM when the kernel used in both methods has a constant value for $k(\mathbf{x},\mathbf{x})$, for all $\mathbf{x} \in \mathcal{X}$. The experiments described in (Kemmler M. , Rodner, Wacker, & Denzler, 2013) employed kernels having that property. Consequently, the work of Kemmler et al. also showed (in an indirect way) that GP-based novelty detection can outperform one-class SVM in multiple application domains.

A difficulty of GP-based novelty detection is that the technique of maximum likelihood estimation (MLE), commonly used to automatically estimate hyperparameters in GP regression and GP classification, cannot be employed in this case. Given that all labels are equal to 1, MLE leads to an ill-posed optimization problem, which makes MLE solvers crash due to numerical instabilities (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Hyperparameter estimation in

GP-based novelty detection is currently considered an open problem. A very recent publication (Xiao, Wang, & Xu, 2014) has proposed a possible solution based on the expected differences in the prediction of mean and variance between edge samples and interior samples. However, it is important to note that this dissertation does not attempt to solve the problem of hyperparameter estimation in the case of GP-based novelty detection, which is considered here an important topic for further research. The main purpose of this dissertation is to propose robust variants of batch GP and online GPs for doing regression, and to assess their performance compared to standard batch GP and online GPs when used for GP-based novelty detection. The specific scope and goals of this dissertation are described in detail in the following chapter.

CHAPTER 3: PROBLEM STATEMENT

Gaussian processes have been used to solve many regression and classification problems with a performance typically exceeding that of other state-of-the-art machine learning techniques (Rasmussen & Williams, 2006). Interestingly, there are very few cases of GPs applied to novelty detection. However, the experimental work reported in (Kemmler M. , Rodner, Wacker, & Denzler, 2013) demonstrate that GP-based novelty detection can outperform state-of-the-art methods. Those encouraging results and the advantages of Bayesian methods mentioned in the previous chapter support the choice for the general topic considered in this dissertation: **to advance the state of the art of GP-based novelty detection**. Given that GPs are kernel methods, this choice allows proposing new solutions that can be easily adapted to different data types and can work effectively with high-dimensional data, in order to fit the needs of many modern application domains.

The following section states the specific problems addressed in this work, aiming at making contributions to the general topic stated above. This chapter concludes with two brief sections: one section states the hypothesis considered in this dissertation, and the last section lists the contributions of this research effort.

3.1 The Specific Problems

There are two specific problems considered in this dissertation: (1) **develop robust variants of Gaussian processes in order to further improve the performance of GP-based novelty detection**, and (2) **explore the applicability of online GP and SOGP, and the corresponding**

robust variants proposed here, to novelty detection. Each of these topics is described in more detail in the following subsections.

3.1.1 The Need for Robust GP-based Novelty Detection

As described in previous chapters, novelty detection techniques build a model of the normal class based on the training data, and they subsequently use that model to assign labels or outlier scores to new observations. However, if the method employed to build the model is sensitive to outliers (i.e. non-robust) then the resulting model might lead to inaccurate outlier detection when the training data contain mislabeled examples or observations with erroneous data. This might be reflected in operation by incorrectly labeling outliers as members of the normal class (the so called *masking effect*), or labeling normal observations as outliers, which is usually called the *swamping effect* (Rousseeuw & Hubert, 2011). Having incorrectly labeled data is a common issue in real-life data sets. Consequently, modern methods should be robust. However, many of the methods previously described in our introductory chapters assume that training data are correct. Arguably, methods from statistical novelty detection are more amenable to the introduction of a robust approach, by directly leveraging techniques from robust statistics. A good review of some robust techniques is given in (Rousseeuw & Hubert, 2011). Some novelty detection methods can be made robust by replacing their estimators of location and scale by their robust counterparts. For instance, the mean can be replaced by the median or an *M*-estimator of location; and the standard deviation can be replaced by a robust measure of scale, like the MAD (the median of absolute deviations from the median) and the interquartile range (IQR). In the case of multivariate data, robust estimators of location and dispersion can be obtained by using the minimum covariance determinant (MCD) method (Rousseeuw P. J., 1985), (Hubert &

Debruyne, 2010). Note that robust estimators that are even more efficient can be obtained when the MCD robust estimates mentioned above are used to assign weights to the observations based on their robust Mahalanobis distance to the MCD-based mean, and subsequently those weights are used to obtain new robust estimators of location and dispersion (Rousseeuw & Hubert, 2011). Subspace-based novelty detection methods can also benefit from robust statistics. A simple example is the use of a robust PCA method, like ROBPCA (Hubert, Rousseeuw, & Vanden Branden, 2005), instead of the classical PCA.

In the particular case of GP regression, one of the main difficulties is the need for properly optimizing hyperparameters, which are the noise variance and the parameters of the covariance function. Maximum likelihood estimation (MLE) is a method widely used to estimate hyperparameters. It is easy to understand, asymptotically efficient in many cases (Daniels, 1961), and relatively simple to implement. However, it is well-known that MLE is highly sensitive to the presence of outliers in the data, which could radically affect posterior distributions in a Bayesian framework (Agostinelli & Greco, 2013). As a simple example, consider the data shown in Figure 3.1, taken from sampling the function $y = 5\sin(x)$ from -10 to 10 at regular increments of 0.5, with noise variance $\sigma^2 = 0.5$. A few sample points were randomly converted to outliers. Figure 3.2 shows the useless posterior GP obtained by using the simple exponential kernel (see equation 4.39) and hyperparameters that were estimated by MLE. Furthermore, obtaining an effective GP in the presence of outliers can be challenging regardless of whether or not MLE is employed to estimate hyperparameters (Jylänki, Vanhatalo, & Vehtari, 2011). As an example of this, Figure 3.3 shows the posterior GP trained on the same artificial data set using again the simple exponential kernel, but this time employing the suitable hyperparameter values $a_1 = 1$

and $\sigma^2 = 0.5$. Although the results are much better than relying on the MLE method, it is clear that the prediction of the posterior GP becomes affected by the outliers in the training data.

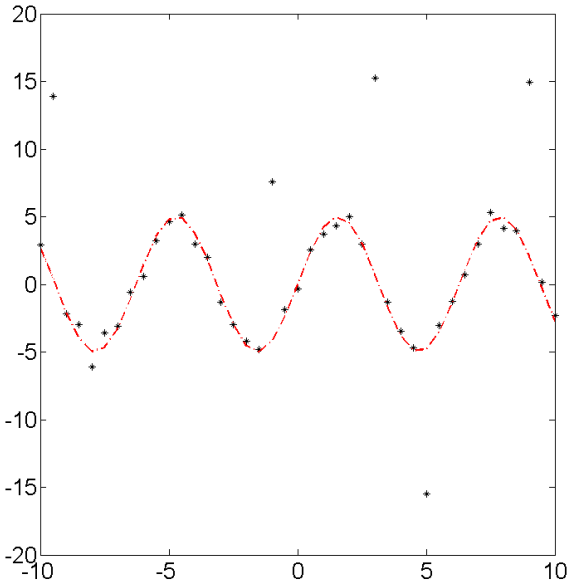

Figure 3.1: Data from sampling $y = 5\sin(x)$ at regular increments of 1 from -10 to 10, with noise variance 0.5 and added outliers. The underlying true function is shown as a discontinuous red line.

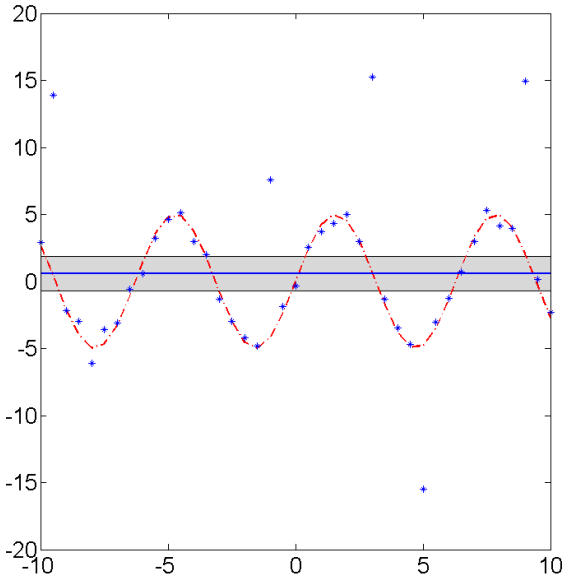

Figure 3.2: Posterior GP obtained by using hyperparameter values obtained from MLE. The continuous blue line denotes the posterior mean, and the shaded area denotes the corresponding 95% confidence interval. The MLE method was called with suitable initial values $\sigma^2 = 0.5$ and $a_1 = 1$, but numerical instability led MLE to incorrect estimates $\sigma^2 = 40.2909$ and $a_1 = 1.4756e - 06$.

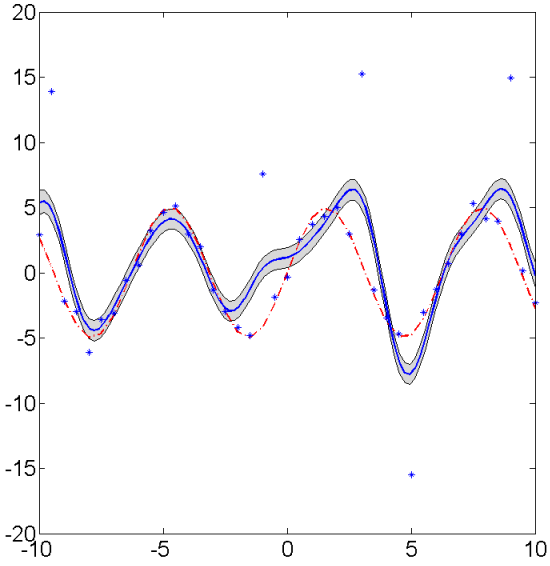

Figure 3.3: Posterior GP obtained by using suitable hyperparameter values: $a_1 = 1$ and $\sigma^2 = 0.5$. The continuous blue line denotes the posterior mean, and the shaded area denotes the corresponding 95% confidence interval.

A typical approach to tackle this problem is to employ likelihood functions that are robust to outliers. For instance, robust pseudo-likelihoods have been employed to obtain robust posterior distributions (Greco, Racugno, & Ventura, 2008). The presence of outliers has also been handled by using likelihoods corresponding to robust distributions. For instance, a Student-t observation model is employed in (Jylänki, Vanhatalo, & Vehtari, 2011) to obtain a robust GP. However, employing non-Gaussian likelihoods leads to analytically intractable inference, which requires the use of approximation techniques that may be complex, computationally expensive and/or inefficient. Consequently, **the most important specific problem of this dissertation is to obtain robust variants of Gaussian processes, both for batch and online learning, which could be implemented without using approximation techniques. These robust GPs can be leveraged to improve the effectiveness of the GP-based novelty detection approach described in** (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Inspired by the use of weights in robust and quasi-robust statistics in order to obtain robust estimates, this dissertation explores weights as the mechanism to address this specific problem.

3.1.2 The Need for Online GP-based Novelty Detection

Most novelty detection algorithms follow a batch approach. Additionally, in the case of supervised and semi-supervised learning, it is commonly assumed that all observations labeled as members of the normal class are labeled correctly. In general, training data are assumed to be fully and truly representative of the normal class. However, in practice training data are typically not only affected by a small percentage of outliers but might not be representative of the whole input space, which in turn affects the resulting models. Additionally, even in the case of a complete and high-quality training data set, in various domains the statistical properties of

variables and processes are highly dynamic; so that even a well-trained system will eventually become obsolete. Consequently, processing data as they arrive has become a necessity of several modern applications, such as fraud detection, automatic surveillance, network intrusion detection, and interactive training systems. Updating the corresponding normality model in an incremental fashion has been an important concern in recent applications (Pokrajac, Lazarevic, & Latecki, 2007).

The intrinsic variability of processes is denoted in machine learning by the term *concept drift* (Zliobaite, 2009). As a simple example of the nature of concept drift in novelty detection, consider a probability density function for normal data consisting of two local maxima. It may be case that an algorithm was initially trained using data coming mostly from a neighborhood of the first maximum. While in operation, however, data coming from a neighborhood of the second local maximum will be incorrectly classified as outliers. If the algorithm did not provide a way of updating its domain knowledge, those normal data points would always be considered outliers. The outputs of many commercial systems for novelty detection are analyzed by human experts to determine whether abnormal observations are actually outliers. However, few systems address the problem of integrating the experts' decisions back into the system in an effective and speedy way.

A common solution to the problems imposed by incomplete training data and concept drift is to re-train the algorithm using updated training data. In general, that approach is not efficient. For instance, re-training might be a resource-consuming process, because of the memory and the time required to store and process increasingly bigger, typically high-dimensional, data sets.

Some online outlier detection algorithms address these limitations by regularly applying a batch learning algorithm to a small subset of the original training data added to a set containing the latest misclassified observations. One example of that approach is the online training algorithm for support vector machine (SVM) proposed in (Zhang & Shen, 2005), which employs a one-class SVM. It iteratively applies an SVM training algorithm to a training data set composed of the new observations and the support vectors of previous iterations, producing an updated decision function at each step. Although this algorithm resembles an incremental approach (the model is updated for new training observations), it actually applies a batch training algorithm at each step, which can be a resource-consuming process. Additionally, previously well-classified observations might be miss-classified in the future. Consequently, the algorithm needs to adjust for the loss of previous knowledge. It is desirable to have a mechanism that allows a faster and more efficient update of the model than the periodic application of a batch training technique.

Learning algorithms capable of updating their knowledge in a truly online fashion (i.e. learning one observation at a time) seem to offer a more promising solution to the problems of incomplete training data and concept drift (Giraud-Carrier, 2000). Online learning is typically more computationally efficient than batch re-training. Furthermore, it appears to be a more natural approach to problems involving online data processing (e.g., video surveillance, network traffic monitoring, monitoring sensor data in real time, and auditing credit card transactions). However, the online learning approach has been applied to novelty detection in very few cases. To mention one example, an incremental version of the Local Outlier Factor (LOF) algorithm (Breunig, Kriegel, Ng, & Sander, 2000) was introduced in (Pokrajac, Lazarevic, & Latecki, 2007) with good results. Another important result is the Incremental Connectivity-Based Outlier Factor

(COF) algorithm (Pokrajac, Reljin, Pejcic, & Lazarevic, 2008). These two algorithms use a distance-based approach (specifically nearest neighbor-based techniques). There are some advantages in using this approach: it is well fitted to unsupervised novelty detection, the concept of a distance between observations is applicable to a great number of data types, and the online learning operations are relatively simple. However, the distance-based approach also has disadvantages. For instance, it is highly sensitive to the presence of normal observations in low-density areas of the training data. Additionally, outliers can be misclassified as normal observations when they appear within small clusters of outliers. Finally, novelty detection algorithms using the distance-based approach typically face a trade-off between the computational complexity of classifying new observations and the amount of memory needed to operate.

A good online novelty detection method should combine the strengths of the distance-based approach and of methods building a knowledge model, like one-class SVM and GPs. It is desirable to have algorithms that use simple and efficient incremental operations as well. One novelty detection algorithm that follows such an approach is the one described in (Kivinen, Smola, & Williamson, 2004). In this algorithm, the training examples are available one at a time from the sequence of pairs $\{(\mathbf{x}_1, \mathrm{y}_1), (\mathbf{x}_2, \mathrm{y}_2), \dots, (\mathbf{x}_t, \mathrm{y}_t), \dots.\}$. The learning algorithm produces a sequence of models $\{f_1, f_2, \dots, f_t, \dots\}$ that serve as decision functions. At iteration *t,* the algorithm computes its decision function as follows:

$$f_t(x) = \sum_{i=1}^{t} \alpha_i k(x_i, x) \; . \qquad (3.1)$$

The coefficients α_i are updated at each iteration. To deal with an increasingly larger number of α coefficients, the authors proposed ways to store and update only a subset of them at each iteration *t*. Although that work follows a truly incremental approach, it has at least three drawbacks. First, the method does not take into account the relevance of examples already learned or the relative importance of the current example used to update the decision function. Second, there is no guarantee that a decision function f_t improves on the previous decision function, from iteration to iteration. Third, the update rules for the α coefficients depend on parameters that are difficult to estimate.

Other works, like those of (Tax & Laskov, 2003) and (Laskov, Gehl, Krüger, & Müller, 2006) have also focused on online learning algorithms that can be directly used for novelty detection or leveraged for that purpose. The first one proposed an online variant of the SVDD method, which was called *Online SVDD*. The second paper studied the convergence properties of an exact incremental SVM method proposed in (Cauwenberghs & Poggio, 2001), for which, according to (Laskov, Gehl, Krüger, & Müller, 2006), no successful practical applications had been reported. Laskov et al. offered some improvements on that exact algorithm. A more recent variant of an online SVDD, called *Incremental SVDD*, was proposed in (Tavakkoli, Nicolescu, Bebis, & Nicolescu, 2008). It was reported in that paper that training Incremental SVDD is faster and requires less memory than training SVDD and Online SVDD, and it is capable of outperforming those methods. Online kernel methods such as Online SVDD and Incremental SVDD have addressed some of the limitations typically related to batch learning algorithms. However, there are still difficulties associated to online kernel-based methods. For instance, they usually depend

on kernel parameters, and effectively estimating those parameters in the online learning approach is, in general, more difficult than in the batch approach.

Online GP and SOGP have the advantages offered by the GP Bayesian formulation while providing an incremental learning approach that is appropriate to many modern problems. SOGP in particular is well suited to problems dealing with limited memory and/or very large or undetermined number of observations (e.g. sensor streams). Given the recent successful applications of GPs in machine learning, and novelty detection in particular, it is expected that online GP-based novelty detection will provide results similar to those obtained from batch GP-based novelty detection. That would mean that they could compare favorably to modern online classification-based kernel methods, while benefitting from the advantages provided by the Bayesian approach. However, to the best of our knowledge the work reported in (Ramirez-Padron, Mederos, & Gonzalez, 2013) offers the only application of online GPs to novelty detection. The experimental results in that work, although preliminary, show that novelty detection using online GPs can achieve performances similar to those from batch GP, even under strong sparseness constraints in the case of SOGP. For these reasons, **the second specific problem considered in this dissertation is to expand the experimental work presented by the author in** (Ramirez-Padron, Mederos, & Gonzalez, 2013)**, by comparing the performance of batch GP and online GPs when used for novelty detection on various data sets. We are particularly interested in comparing the capabilities of the robust variants of online GPs and batch GP introduced in this dissertation, when used on training data contaminated with outliers.**

3.2 Hypothesis

The main hypothesis in this dissertation is that **robust variants of GPs can be proposed in a way that the computational complexity of the robust GPs are similar to the computational complexity of the corresponding standard GPs, but the robust variants are more effective than standard GPs at solving regression problems with data contaminated with outliers. It is expected that the new robust GPs will perform better than standard GPs when used for GP-based novelty detection in the presence of outliers. This advantage should be confirmed experimentally for batch and online robust variants of GPs.**

3.3 Contributions

This research provides the following contributions to the field of machine learning:

1. New robust variants of batch GP, online GP, and SOGP within a regression framework.
2. An experimental comparison of robust GP regression and standard GP regression in various simulated problems, using training data with and without outliers.
3. An experimental design to compare the effectiveness of robust GP-based novelty detection to standard GP-based novelty detection. The experimental comparison includes batch GP, online GP, and SOGP with two different capacities. Experiments are run on data sets containing no outliers as well as data sets contaminated with outliers.
4. Experimental results obtained from implementing the experimental design. These results allow the following analyses:
    a. A comparison of GP-based novelty detection using standard GPs versus GP-based novelty detection using robust GPs.

b. A comparison of GP-based novelty detection using batch GPs versus GP-based novelty detection using online GPs.
c. A comparison of the four membership scores employed in (Kemmler M. , Rodner, Wacker, & Denzler, 2013).

CHAPTER 4: IMPLICIT WEIGHTED GAUSSIAN PROCESSES

As described in the previous chapter, the MLE method for parameter estimation is highly sensitive to outliers in the training data, which could strongly affect the posterior distributions, which are used for GP regression in our case. The most common approach to tackle this problem is to modify the likelihood function so that it becomes robust to outliers. The standard approaches rely on introducing robust pseudo-likelihoods (Greco, Racugno, & Ventura, 2008) or using likelihoods corresponding to robust distributions, e.g. (Jylänki, Vanhatalo, & Vehtari, 2011). These approaches typically lead to intractable inferences, which usually require the use of computationally expensive approximation techniques.

The usage of weighted likelihoods in Bayes formula was proposed in (Agostinelli & Greco, 2013). That work assigns a weight function as an exponent to each term of the likelihood, with the goal of diminishing the effect of anomalous observations. That approach is briefly described here. Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ be an i.i.d. sample drawn from a random variable X with probability density $p(\mathbf{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^{\boldsymbol{p}}$, with $p \geq 1$. Let $\hat{F}_N$ denote the empirical cumulative distribution function based on $\mathbf{X}$. In (Agostinelli & Greco, 2013), a weighted likelihood function is defined as:

$$L^w(\mathbf{X}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta})^{w(\mathbf{x}_i;\, \boldsymbol{\eta},\, \hat{F}_N)}, \qquad (4.1)$$

where the weight function w is bounded differentiable and non-negative, $\boldsymbol{\eta}$ denotes unknown parameters of w, and typically $\boldsymbol{\theta} \subseteq \boldsymbol{\eta}$. The intuition behind this approach is the following: given that weights very near to zero are assigned to outliers, the corresponding weighted likelihood

terms take values approximately equal to 1. Consequently, the expression for $L^w(\mathbf{X}|\boldsymbol{\theta})$ depends little on likelihood terms corresponding to outliers in the data. For weight functions satisfying a certain sufficient condition, $L^w(\mathbf{X}|\boldsymbol{\theta})$ has the asymptotic properties of the "genuine" likelihood function $L(\mathbf{X}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta})$ when no outliers are present. Additionally, the parameter values $\widehat{\boldsymbol{\theta}}$, estimated through MLE on the weighted likelihood, are high breakdown estimators in the presence of outliers. Aside from being restrictive regarding the location of the weight functions, the type of weight functions proposed in (Agostinelli & Greco, 2013) might be expensive to compute, given its dependency on parameter estimates and on the empirical cumulative distribution function.

As a separate concern, classic regression analysis assumes that errors are normally distributed with a constant variance. However, in many cases the assumption of constant noise variance does not hold, leading to another difficulty: learning from heteroscedastic data. Weighted least squares is a solution to this problem within the linear regression framework. It involves adding weights to the least squares formulation; i.e., the coefficients $\boldsymbol{\beta}$ and b of a linear regression model $\mathrm{y} = \boldsymbol{\beta}\mathbf{x} + b$ are obtained by minimizing the following loss expression:

$$W(\boldsymbol{\beta}, b) = \sum_{i=1}^{N} w_i(\mathrm{y}_i - \boldsymbol{\beta}\mathbf{x}_i - b)^2, \qquad (4.2)$$

where typically each weight w_i is the reciprocal of the error variance at point $\mathbf{x}_i$, estimated from the data (Ryan, 1996). Assigning a small weight to an observation $(\mathbf{x}_i, \mathrm{y}_i)$ allows the prediction of the model at $\mathbf{x}_i$ to depart from y_i without incurring in a great loss.

We are not aware of any work that uses weighted likelihoods in GPs to effectively learn from heteroscedastic data. Additionally, the MLE method is not only sensitive to outliers but also to

misspecifications of heteroscedasticity in data (Carroll & Ruppert, 1982). Standard GPs assume that noise is equally distributed in the training data (i.e. homoscedasticity), which makes standard GPs sensitive to the presence of varying noise levels and outliers in the data when hyperparameters are estimated through the MLE method. The problem of varying uncertainty has been effectively handled in GPs by modeling noise variance as a function of the input data, leading to heteroscedastic GP models (Goldberg, Williams, & Bishop, 1998), (Kersting, Plagemann, Pfaff, & Burgard, 2007). However, these approaches equally weight all data points, which greatly limit their capability to effectively model data affected by outliers. To effectively deal with both problems (outliers and heteroscedasticity) using weights, the weight of each observation should be individually assessed, even when taking into account other observations. To the best of our knowledge, the only application of individual uncorrelated weights in GPs to model heteroscedastic data is presented in (Rottmann & Burgard, 2010). In that work, it is noted that "*the weight of a sample and thus the importance on the predictive distribution can be regulated by adapting the observation noise correspondingly*". Their approach is to determine an individual weight for each sample and subsequently employ the weights to estimate individual noise levels for each training point. The individual noise levels are added to the set of hyperparameters of the GP model, which makes this approach very expensive computationally. There are at least two further limitations associated to weighted GPs as proposed in (Rottmann & Burgard, 2010): (1) Weighted cross-validation (Sugiyama, Krauledat, & Müller, 2007) is employed to estimate the GP hyperparameters. In general, cross-validation (CV) is a computationally demanding technique. (2) The CV criterion used in (Rottmann & Burgard, 2010) depends only on the predicted mean of the model (leaving out the quality of variance

prediction). To arrive at the final predictive model, a second GP is employed to model predictive variances, leveraging the posterior means of the first GP as its mean function. CV is applied again to estimate the parameters of the second GP. Although this approach seems to effectively deal with both heteroscedasticity and the presence of outliers, the addition of a noise parameter per observation and the dual application of CV make it impractical for most real-life applications.

To the best of our knowledge, no work has proposed the use of weight functions in the definition of the likelihood terms in order to obtain robust GPs or improve the effectiveness of the MLE method. Contrary to the work of (Rottmann & Burgard, 2010), here we propose GPs that are robust and can effectively model heteroscedastic data without the need to add a hyperparameter to the GP model for each observation in the training data. The work presented in this chapter is inspired by the weighted likelihood approach of (Agostinelli & Greco, 2013). However, our approach differs from that of (Agostinelli & Greco, 2013) in various aspects: we propose using weighted likelihoods that include weights as part of the definition of the likelihood terms, instead of requiring that weight functions must be exponents of the likelihood terms in a joint likelihood. Additionally, our weight functions (called here "data weighers") do not depend on empirical cumulative distribution functions. We only require from the weight functions to take values in the interval (0, 1] and that weights denote how consistent data points are with respect to the underlying model, based on their relative positions to other training observations. We do not specify a particular placeholder for the weight functions within the term of a weighted likelihood. However, we require that weighted likelihoods retain the properties of "genuine" likelihoods, to guarantee that proper posteriors are obtained, and consequently the validity of the

Bayesian inference. We claim that weighted likelihoods, as defined later in this chapter, can be useful to obtain robust GP models as well as for dealing with heteroscedastic data. Similar to the work of (Rottmann & Burgard, 2010), our work proposes to take advantage of individual weights. However, it does that through the use of a weighted likelihood instead of adding multiple hyperparameters to the GP models. Consequently, the computational complexity of obtaining the posterior GP would be affected only by the complexity of calculating the weight functions. When using a heteroscedastic data weigher, our GP models can effectively learn heteroscedastic data without explicitly modeling the different noise variances; a property that we call *implicit heteroscedasticity*.

The proposed approach is illustrated by deriving a weighted Gaussian likelihood formulation for batch GP, online GP, and SOGP. As a consequence of having the weight functions within the likelihood terms, the mathematical formulation of our weighted GPs using a weighted Gaussian likelihood remains very similar to the formulation of standard GPs. Essentially, we propose robust weighted GPs that do not require the use of approximation techniques because in this particular case our weighted likelihood is also a Gaussian distribution. Because of this and to differentiate our approach from other applications of weights in GPs, our weighted GPs are called *implicit weighted GPs*. Note however that this and subsequent chapters might refer to our implicit weighted GPs just as weighted GPs, whenever that becomes clear from the context.

Three data weighers are introduced in this chapter: one to make GPs robust to outliers, one to obtain implicitly heteroscedastic GPs, and a third one that combines the previous two in order to obtain robust and implicitly heteroscedastic GPs. The applicability of these data weighers is

demonstrated through experiments on simulated data. It is shown through the experiments that the optimization surfaces from the MLE method are highly distorted by the presence of outliers in the data in the case of standard (non-weighted) GPs. However, MLE's optimization surfaces were shaped as if the data contained no outliers when our weighted likelihoods were used.

This rest of this chapter is structured as follows: Section 4.1 provides some mathematical preliminaries from robust statistics, which will be used in the following sections. Section 4.2 introduces our approach to obtain weighted GPs by using weighted likelihoods. Implicit weighted variants of batch GP, online GP and SOGP are derived for the particular case of weighted Gaussian likelihoods, including the formulas needed to estimate GP hyperparameters using the MLE method. It is important to note that our weighted Gaussian likelihood is not a weighted likelihood function according to the definition given in (Agostinelli & Greco, 2013). Section 4.3 introduces our three data weighers. A comparison of the computational complexities of traditional GPs and the implicit weighted GPs proposed here is given in section 4.4. Section 4.5 provides experimental evidence of the benefits of learning weighted GPs models from data containing outliers and/or heteroscedastic regions. Finally, the positive effect of the weighted likelihood approach on the MLE optimization surface is shown in section 4.6.

4.1 Robust Potentials and Weights

Robust statistics (Huber & Ronchetti, 2009), (Maronna, Martin, & Yohai, 2006) yields estimation methods that are not greatly affected by outliers. Given a set of observations $\{y_i : i = 1 \dots N\}$, let us assume the following data model:

$$y_i = \theta + \eta_i \,, \qquad (4.3)$$

where the true value of θ is unknown and the additive errors $\{\eta_i : i = 1 \ldots N\}$ are independent and identically distributed (i.i.d.) random variables. This data model is known as a location model. It is commonly considered that the distribution of errors comes from a parametric family. However, typically there is a small percentage of errors that do not obey the assumed distribution. The idea of robust statistics is to give less influence to abnormal data in order to better estimate θ. To accomplish this goal, a special type of cost functions $\rho(\cdot)$, called robust potentials, play a key role in the following optimization problem:

$$\hat{\theta} = arg\ min \sum_{i=1}^{N} \rho(\mathrm{y}_i - \theta), \qquad (4.4)$$

where $\hat{\theta}$ is called the *M*-estimator of location. A potential function $\rho: \mathbb{R} \longrightarrow \mathbb{R}$ satisfies the following two properties:

- **Symmetry** $\rho(-z) = \rho(z)$, (4.5)
- **Robustness** $\lim_{z \to +\infty} \frac{\psi(z)}{z} = 0$, (4.6)

where $\psi(z) = \frac{\partial \rho(z)}{\partial z}$ is known as the *influence function* (Hampel, Ronchetti, Rousseeuw, & Stahel, 1986), (Maronna, Martin, & Yohai, 2006). The robustness property implies that errors η_i that are too large have a lower cost than that quantified by the quadratic potential z^2. Given a robust potential ρ, a possible way to solve problem (4.4) is to solve the following equation:

$$\sum_{i=1}^{N} \psi(\mathrm{y}_i - \theta) = 0\ . \qquad (4.7)$$

Assuming that ψ has derivative at 0, the robust weight functions are constructed as follows:

$$w(z) = \begin{cases} \frac{\psi(z)}{z}, & if\ z \neq 0 \\ \psi'(0), & if\ z = 0 \end{cases}\ . \qquad (4.8)$$

Using equation (4.8), the equation (4.7) can be rewritten as

$$\sum_{i=1}^{N} w(\mathrm{y}_i - \theta)(\mathrm{y}_i - \theta) = 0\,, \qquad (4.9)$$

which provides a method to compute $\hat{\theta}$ as a weighted mean through an iterative scheme:

$$\theta_{k+1} = \frac{\sum_{i=1}^{N} w(\mathrm{y}_i - \theta_k)\mathrm{y}_i}{\sum_{i=1}^{N} w(\mathrm{y}_i - \theta_k)}\,. \qquad (4.10)$$

If the recursive equation (4.10) converges then its limit is $\hat{\theta}$. The definitions of the weights $w(\cdot)$ and the robustness property of ρ ensure that outlying observations receive small weights; therefore the contribution of these observations to the model is small. Therefore, they are appropriate to be used as weights in approaches different to *M*-estimators but related to them, such as weighted regression in robust statistics (Agostinelli & Greco, 2013), (Maronna, Martin, & Yohai, 2006).

Three main classes of robust potentials can be found in the literature: 1) Monotone ψ (e.g. the Huber´s potential), 2) Soft redescending ψ (e.g. Cauchy´s potential), and 3) Hard redescender ψ. The later class includes the well-known and widely used Welsh´s potential, which has a scale factor *k*:

$$\rho(\mathrm{z}) = 1 - \frac{1}{2k}\mathrm{e}^{-k\mathrm{z}^2}, \qquad (4.11)$$

For some robust potentials we have that $\frac{\psi(z)}{z} \to 0$ very rapidly when $z \to \infty$. Consequently, observations which are not too distant from the corresponding *M*-estimate would have a very small influence in the estimation process. Additionally, the robustness property can yield numerical algorithms that are ill-posed (Rey, 1983). To overcome these limitations, the robustness property can be relaxed as follows:

$$\lim_{z \to +\infty} \frac{\psi(z)}{z} = \gamma, \;\; \gamma \in (0,1)\,. \qquad (4.12)$$

Typically γ is set to a small value. A potential that satisfies equation (4.12) is called a *quasi-robust potential* (Rey, 1983), usually denoted by $\rho_Q(\cdot)$. A common way of building a quasi-robust potential is to add a small quadratic perturbation to a robust potential:

$$\rho_Q(z) = (1-\gamma)\rho(z) + \gamma z^2. \qquad (4.13)$$

A computation of the corresponding weight function reveals that the weights are also a convex combination:

$$w_Q(z) = (1-\gamma)w(z) + \gamma \cdot 1\,. \qquad (4.14)$$

Equation (4.14) leads to computing the estimator $\hat{\theta}$ without severely penalizing potential outliers. The Welsh's potential will be employed in defining our data weighers later in this dissertation. Its corresponding weight function is written as:

$$w_Q(z) = (1-\gamma)e^{-kz^2} + \gamma\,. \qquad (4.15)$$

4.2 Implicit Weighted Gaussian Processes

Our approach assumes that the likelihood $p(D|\mathbf{f}_D)$ depends on a collection of weights $\mathbf{w}_D = [w_1, w_2, \ldots, w_N]$, where each $w_i \in (0,1]$. These weights express how consistent each data point $(\mathbf{x}_i, y_i)$ is with respect to the underlying model. We assume that the training data are conditionally independent, so that:

$$p(D|\mathbf{f}_D; \mathbf{w}_D) = p(\mathbf{y}|\mathbf{f}_D; \mathbf{w}_D) = \prod_{i=1}^{N} p(y_i|\, f_i; w_i)\,. \qquad (4.16)$$

A particular structure is not demanded here from the expression of $p(y_i|f_i; w_i)$. However, $p(y_i|f_i; w_i)$ must be a "genuine" likelihood (i.e. a posterior obtained by using it in a Bayesian

expression must be a proper probability distribution). This guarantees that posterior GPs are valid for doing inferences. We call any likelihood $p(D|\mathbf{f}_D; \mathbf{w}_D)$ having this property an *implicit weighted likelihood* (this and subsequent sections might use the term *weighted likelihood* if it is clear from the context that we refer to the implicit type introduced in this work).

Weights might be given as part of the training data, but most likely they would have to be calculated using certain weight functions, which we call *data weighers*. The input to a data weigher is a finite collection $D = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) : \mathbf{x}_i \in \mathcal{X},\ \mathrm{y}_i \in \mathbb{R}\}$, and their output consists of a weight per data point in D (note however that for the case of online GPs we need to calculate at each learning step $t + 1$ only the weight for the observation $(\mathbf{x}_{t+1}, \mathrm{y}_{t+1})$). To calculate w_i for an observation $(\mathbf{x}_i, \mathrm{y}_i)$ our data weighers use a neighborhood N_i of $\mathbf{x}_i$. The data weighers are devised based on whether we are learning from heteroscedastic data or from data containing outliers or both. In the case of heteroscedasticity, each weight w_i must be inversely proportional to the variance of the set $\{\mathrm{y}_j : \mathbf{x}_j \in N_i\}$. In that case, weights are normalized so that they take values in (0, 1]. Our experiments show that heteroscedastic data weighers can allow GPs to effectively model heteroscedastic data without the need for modeling noise variance as a function of the input data or adding hyperparameters to the GPs. This property is called here *implicit heteroscedasticity*. In the case of data with outliers, each w_i is estimated based on the relationship of the corresponding observation y_i to robust estimations of mean and variance of the set $\{\mathrm{y}_j : \mathbf{x}_j \in N_i\}$. This approach makes the GP robust by assigning small weights to observations that significantly deviates from the robust estimate of location. In order to obtain a

posterior GP that is both robust and implicitly heteroscedastic, we propose a data weigher that is a combination of a robust data weigher and a heteroscedastic data weigher.

For some observations, the corresponding neighborhoods might contain so few data points that weights could not be estimated reliably. A *default weight* is assigned to those observations. Using 1 as the default weight could become problematic if the recipient observations were actually unworthy of the highest possible weight; i.e. the posterior model will be over-confident when making predictions on the corresponding input regions, with predictive means inaccurately biased towards the training data. To avoid this risk, we chose 0.5 as the default weight; which denotes the uncertainty associated to the lack of data in N_i. The risk in this case is obtaining posterior means slightly biased towards the prior means, and posterior variances greater than it should be for some input regions. Researchers might prefer to take this risk however, given that most real-life data contain noise and outliers. Details on how to define N_i for GP regression and when to assign the default weight are given in a section below.

This approach allows using any implicit weighted likelihood. However, this work focuses in introducing the following implicit weighted Gaussian likelihood:

$$p(D|\mathbf{f}_D;\mathbf{w}_D) = \frac{1}{\sqrt{(2\pi)^N|\mathbf{W}|}} e^{-\frac{1}{2}(\mathbf{y}_D-\mathbf{f}_D)^T\mathbf{W}^{-1}(\mathbf{y}_D-\mathbf{f}_D)}, \qquad (4.17)$$

where $\mathbf{W} = \sigma^2 \text{diag}\left(\frac{1}{\mathrm{w}_1}, \frac{1}{\mathrm{w}_2}, \ldots, \frac{1}{\mathrm{w}_N}\right)$, and $|\mathbf{W}|$ denotes the determinant of $\mathbf{W}$. The main reason for this choice is that posterior GPs are obtained analytically. Note that weights were introduced into the original Gaussian likelihood in a way that smaller weights effectively increase the noise variance for the corresponding observations. Consequently, the smaller a weight w_i the more

irrelevant becomes that y_i and f_i greatly differ from each other. In other words, the corresponding likelihood is near to 1 for a broad range of f_i values when the weight is very small. This allows us to obtain models with predictions that greatly deviate from "dubious" training observations without incurring in high losses.

4.2.1 Implicit Weighted Batch GP

Recalling that the joint distribution for the prior GP is $p_o(\mathbf{f}_D) = \frac{1}{\sqrt{(2\pi)^N |K_D|}} e^{-\frac{1}{2}(\mathbf{f}_D - E_0[\mathbf{f}_D])^T K_D^{-1} (\mathbf{f}_D - E_0[\mathbf{f}_D])}$, we use the formula for joint Gaussian distributions of two random vectors to find an analytic expression for the log marginal likelihood $ln \int p(\mathbf{y}|\, \mathbf{f}_D; \mathbf{w}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D$:

$$p(\mathbf{y}|\, \mathbf{f}_D; \mathbf{w}_D) p_o(\mathbf{f}_D) = p\left(\begin{bmatrix}\mathbf{f}_D \\ \mathbf{y}\end{bmatrix};\ \mathbf{w}_D\right) = \mathcal{N}\left(\begin{bmatrix}E_0[\mathbf{f}_D] \\ E_0[\mathbf{f}_D]\end{bmatrix}, \begin{bmatrix}K_D^{-1} + \mathbf{W}^{-1} & -\mathbf{W}^{-1} \\ -\mathbf{W}^{-1} & \mathbf{W}^{-1}\end{bmatrix}^{-1}\right). \quad (4.18)$$

Applying properties of multivariate Gaussian distributions:

$$ln \int p(\mathbf{y}|\, \mathbf{f}_D; \mathbf{w}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D = ln\left(N(E_0[\mathbf{f}_D], K_D + \mathbf{W})\right)$$

$$= ln\left(\frac{1}{\sqrt{(2\pi)^N |K_D + \mathbf{W}|}} e^{-\frac{1}{2}(\mathbf{y} - E_0[\mathbf{f}_D])^T [K_D + \mathbf{W}]^{-1} (\mathbf{y} - E_0[\mathbf{f}_D])}\right)$$

$$= -\frac{1}{2}(\mathbf{y} - E_0[\mathbf{f}_D])^T [K_D + \mathbf{W}]^{-1} (\mathbf{y} - E_0[\mathbf{f}_D]) - \frac{1}{2} ln|K_D + \mathbf{W}| - \frac{N}{2} ln(2\pi)\,. \quad (4.19)$$

Consequently, the parameters of the normalization lemma $\boldsymbol{q}$ and $\boldsymbol{R}$ are expressed as:

$$\boldsymbol{q} = [K_D + \mathbf{W}]^{-1} (\mathbf{y} - E_0[\mathbf{f}_D])\,, \quad (4.20)$$

$$\boldsymbol{R} = -[K_D + \mathbf{W}]^{-1}\,. \quad (4.21)$$

These results lead to the following expression for the prediction of a posterior weighted batch GP when using our weighted Gaussian likelihood:

$$\mu_* = \mu_0(\mathbf{x}_*) + \boldsymbol{k}_*^T \boldsymbol{\alpha}, \quad (4.22)$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) + \boldsymbol{k}_*^T \boldsymbol{C}\, \boldsymbol{k}_*, \quad (4.23)$$

where

$$\boldsymbol{\alpha} = [K_D + \mathbf{W}]^{-1}\big(\mathbf{y} - \boldsymbol{\mu}_0(\boldsymbol{X})\big), \quad (4.24)$$

$$\boldsymbol{C} = -[K_D + \mathbf{W}]^{-1}. \quad (4.25)$$

4.2.1.1 Estimation of Weighted GP Hyperparameters

Let us denote the vector of kernel parameters by $\boldsymbol{\theta}_k = \big(\theta_{k_1}, \theta_{k_2}, \dots, \theta_{k_l}\big)$, where $l \geq 0$ (i.e. $\boldsymbol{\theta}_k$ might be empty). We denote the GP hyperparameters by $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_{l+1}) = (\sigma^2, \boldsymbol{\theta}_k)$. The MLE method is commonly used to estimate the values of hyperparameters. This section derives the MLE formulation to estimate hyperparameters for weighted batch GP using the weighted Gaussian likelihood from equation (4.17). MLE consists of finding the parameter values that maximize the following log marginal likelihood with respect to $\boldsymbol{\theta}$:

$$\mathcal{L}_1 = \ln\big(p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta})\big) = ln \int p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{f}_D) p_o(\mathbf{f}_D) d\mathbf{f}_D \,. \quad (4.26)$$

The expression for $\mathcal{L}_1$ in our case is given by equation (4.19). In general, there is no analytical solution to this optimization problem. Numerical optimization methods that benefit from the derivatives of the objective function are typically employed. Denoting $K_D + \boldsymbol{W}$ by K_p, the derivative of $\mathcal{L}_1$ w.r.t. to each θ_i is written as:

$$\frac{\partial \mathcal{L}_1}{\partial \theta_i} = \frac{1}{2}(\mathbf{y} - E_0[\mathbf{f}_D])^T K_p^{-1} \frac{\partial K_p}{\partial \theta_i} K_p^{-1}(\mathbf{y} - E_0[\mathbf{f}_D]) \; - \frac{1}{2} tr\left(K_p^{-1} \frac{\partial K_p}{\partial \theta_i}\right), \quad (4.27)$$

where the different derivatives of K_p are as follows:

$$\frac{\partial K_p}{\partial \sigma^2} = \frac{\partial (K_D + \mathbf{W})}{\partial \sigma^2} = \frac{\partial \mathbf{W}}{\partial \sigma^2} = \text{diag}\left(\frac{1}{w_1}, \frac{1}{w_2}, \ldots, \frac{1}{w_N}\right), \quad (4.28)$$

$$\frac{\partial K_p}{\partial \theta_{k_i}} = \frac{\partial (K_D + \mathbf{W})}{\partial \theta_{k_i}} = \frac{\partial K_D}{\partial \theta_{k_i}}. \quad (4.29)$$

4.2.1.2 Optimizing hyperparameters with priors

In the case of having a prior $p(\boldsymbol{\theta})$ for the hyperparameters, that prior is included into the MLE formulation by maximizing the following log posterior instead of the log marginal likelihood:

$$\mathcal{L}_2 = ln\big(p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})\big). \quad (4.30)$$

In our particular case, all hyperparameters θ_i are non-negative and are assumed independent from other hyperparameters, so that $p(\boldsymbol{\theta}) = \prod_{i=1}^{|\boldsymbol{\theta}|} p(\theta_i)$, and the objective function can be written as:

$$\mathcal{L}_2 = \mathcal{L}_1 + \sum_{i=1}^{|\boldsymbol{\theta}|} ln\big(p(\theta_i)\big). \quad (4.31)$$

In our experiments, each hyperparameter θ_i is assumed to be distributed $ln\,\mathcal{N}\big(\mu_{\theta_i}, \sigma_{\theta_i}^2\big)$, where μ_{θ_i} and $\sigma_{\theta_i}^2$ are the mean and variance, respectively, of the transformed variable $ln(\theta_i)$. Consequently:

$$\mathcal{L}_2 = \mathcal{L}_1 - \sum_{i=1}^{|\boldsymbol{\theta}|} \left[ln(\theta_i) + ln\big(\sigma_{\theta_i}\sqrt{2\pi}\big) + \frac{\big(ln\theta_i - \mu_{\theta_i}\big)^2}{2\sigma_{\theta_i}^2} \right]. \quad (4.32)$$

The objective function is simplified by removing constant terms:

$$\mathcal{L}_2^* = -\frac{1}{2}(\mathbf{y} - E_0[\mathbf{f}_D])^T K_p^{-1}(\mathbf{y} - E_0[\mathbf{f}_D]) - \frac{1}{2} ln\big|K_p\big|$$

$$- \sum_{i=1}^{|\boldsymbol{\theta}|} \left[ln(\theta_i) + \frac{\big(ln\theta_i - \mu_{\theta_i}\big)^2}{2\sigma_{\theta_i}^2} \right]. \quad (4.33)$$

The corresponding derivative is obtained as follows:

$$\frac{\partial \mathcal{L}_2^*}{\partial \theta_i} = \frac{\partial \mathcal{L}_1}{\partial \theta_i} - \frac{1}{\theta_i}\left(1 + \frac{ln\theta_i - \mu_{\theta_i}}{\sigma_{\theta_i}^2}\right). \quad (4.34)$$

4.2.2 Implicit Weighted Online GP

The expressions for the implicit weighted online GP are obtained here by finding the expressions for the terms q_{t+1} and r_{t+1} when the weighted Gaussian likelihood is used. The derivation proceeds as follows:

$$E_t[p(y_{t+1}|f_{t+1})] = \int p(\mathrm{y}_{t+1}|f_{t+1}) p_t(f_{t+1})\, df_{t+1}$$

$$= \int \mathcal{N}\left(f_{t+1}, \frac{\sigma^2}{\mathrm{w}_{t+1}}\right) \mathcal{N}\left(E_t[f_{t+1}], K_t(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})\right) df_{t+1}$$

$$= \int \mathcal{N}\left(\begin{bmatrix} E_t[f_{t+1}] \\ E_t[f_{t+1}] \end{bmatrix}, \begin{bmatrix} \frac{1}{\sigma_{t+1}^2} + \frac{\mathrm{w}_{t+1}}{\sigma^2} & -\frac{\mathrm{w}_{t+1}}{\sigma^2} \\ -\frac{\mathrm{w}_{t+1}}{\sigma^2} & \frac{\mathrm{w}_{t+1}}{\sigma^2} \end{bmatrix}^{-1}\right) df_{t+1}$$

$$= \mathcal{N}\left(E_t[f_{t+1}],\ \sigma_{t+1}^2 + \frac{\sigma^2}{\mathrm{w}_{t+1}}\right). \quad (4.35)$$

Consequently, the parameters q_{t+1} and r_{t+1} in our case are written as follows:

$$q_{t+1} = \frac{\partial}{\partial E_t[f_{t+1}]} ln\, E_t[p(\mathrm{y}_{t+1}|f_{t+1})] = \frac{(\mathrm{y}_{t+1} - m_{t+1})}{\sigma_{t+1}^2 + \frac{\sigma^2}{\mathrm{w}_{t+1}}}, \quad (4.36)$$

$$r_{t+1} = -\frac{1}{\sigma_{t+1}^2 + \frac{\sigma^2}{\mathrm{w}_{t+1}}}, \quad (4.37)$$

where $m_{t+1} = E_t[f_{t+1}]$ and w_{t+1} is a function of $\{(\mathbf{x}_i, \mathrm{y}_i): i = 1, 2 \ldots t+1\}$.

The model parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$ are updated at each step of the training algorithm shown in Figure 2.1, using the values q_{t+1} and r_{t+1} just obtained. However, weights for previously learned observations would likely change at each learning step if they were recalculated. Consequently, for the case of weighted online GPs the prediction of the model is likely affected by this

discrepancy between previous and current weights. An ideal solution to this issue would be to update parameters $\boldsymbol{\alpha}$ and $\mathbf{C}$ in a way that weight changes are properly reflected in the model. However, this is a very difficult approach given the recursive nature of the learning algorithm, and it will not be explored in this work. Weights are expected to change significantly at the beginning of the learning process. However, they should tend to stabilize after learning a considerable number of observations (assuming there is not a significant concept drift in our data). Based on this rationale, a second approach consists of not attempting to correct for changes in the weights of previously learned data; in other words, it is expected that weight stability will be achieved eventually. This is the approach taken in this dissertation.

4.2.3 Implicit Weighted Sparse Online GP

The formulation of the implicit weighted SOGP is essentially the same as that of SOGP, with the exception that the terms q_{t+1} and r_{t+1} are calculated using equations (4.36) and (4.37). Note that the existence of a *BV* set with a fixed capacity *m* increases the risk of learning observations with inadequate weights. Although in the case of weighted online GP it is reasonable to expect that weights should be increasingly more accurate as more training data arrive, that expectation is justified in the case of weighted SOGP only if the capacity *m* is large enough to accommodate a representative sample of the input space.

Given that SOGP regularly removes from the *BV* set the least informative observations, it is intuitive to expect that removed observations were typically learned with low weights. If that was the case, the discrepancies between previous and current weights will be alleviated by the

removal process. The validity of this expectation is considered in the experimental section of this chapter.

4.3 Data Weighers

This section introduces three data weighers to be used in regression problems: *HeteroscedasticReg*, *RobustReg* and *HeteroscedasticRobustReg*. These data weighers take values in the interval $(\gamma,1]$, where $0 < \gamma \ll 0.5$. They have three parameters: a neighborhood radius *r*, a neighborhood size *s*, and a weight floor γ. As mentioned above, these data weighers work by focusing on a data point $(\mathbf{x}_i, \mathrm{y}_i)$ at a time. The corresponding neighborhood N_i is defined here as the closed ball $B(\mathbf{x}_i; r) = \{\, \mathbf{x}_j \in \mathbf{X} : \|\mathbf{x}_i - \mathbf{x}_j\| \leq r\}$. We assign a non-default weight to $(\mathbf{x}_i, \mathrm{y}_i)$ if and only if N_i contains at least *s* observations. The following subsections describe how each data weigher calculates non-default weights.

4.3.1 HeteroscedasticReg DataWeigher

For each observation $(\mathbf{x}_i, \mathrm{y}_i)$ such that $|N_i| \geq s$, where s is the neighborhood size, this data weigher first executes two steps: (1) calculates a robust variance v_i for the set $\{\mathrm{y_j}: \mathbf{x_j} \in N_i\}$, and (2) calculates a preliminary weight $w_i' = \frac{1}{v_i}$. After doing that for all observations, each w_i' is normalized by dividing it by the maximum of the w_i' values. The normalized weights are denoted here by w_i''. Finally, we leverage the quasi-robust approach to guarantee that no weights are lower than γ, by computing non-default weights as $w_\mathrm{i} = \gamma + (1 - \gamma)\, w_i''$.

4.3.2 RobustReg DataWeigher

For each observation $(\mathbf{x}_i, \mathrm{y}_i)$ to receive a non-default weight, this data weigher calculates a robust mean μ_i and robust variance v_i of the set $\{\mathrm{y_j}: \mathbf{x_j} \in N_i\}$. Subsequently, each weight is calculated using the Welsh's quasi-robust weight function from equation (4.15):

$$w_\mathrm{i} = (1-\gamma)e^{-\frac{(y_i-\mu_i)^2}{v_i}} + \gamma\,. \qquad (4.38)$$

Note that the scale factor *k* was set to 1 here because the term $z = \frac{y_i-\mu_i}{\sqrt{v_i}}$ is already normalized to scale.

4.3.3 HeteroscedasticRobustReg DataWeigher

For each observation $(\mathbf{x}_i, \mathrm{y}_i)$ to receive a non-default weight, this data weigher calculates the heteroscedastic and robust weights as described above. Subsequently, w_i is calculated as the minimum of the two weights.

4.4 Notes on Computational Complexity

This section analyzes how the computational complexities of learning GP regression models are affected by the use of an implicit weighted likelihood that employs any of the three data weighers introduced above. The reliance on searching for observations within neighborhoods of data points naturally leads to implementations that employ space-partitioning data structures such as *k*-d trees (Bentley J. , 1980). Constructing a *k*-d tree can be achieved in $O(N \cdot logN)$ computational complexity (Wald & Havran, 2006). Searching for the neighborhood of each particular data point has $O(logN)$ computational complexity. Once the neighborhood of an observation has been found, what methods are employed to calculate the robust statistics

determine the computational complexity of calculating the corresponding weight. As noted below, in our experiments we employed the *mcdcov* function from the MATLAB library LIBRA (Verboven & Hubert, 2010), which relies on the minimum covariance determinant (MCD) estimator (Rousseeuw P. J., 1984). It was shown recently that the *MCD* estimator has $O\left(N^{\frac{d(d+3)}{2}}\right)$ computational complexity (Bernholt & Fischer, 2004), where d denotes the dimensionality of the input space $\mathcal{X}$. Consequently, the use of the *mcdcov* in our data weighers implies a $O(N^2)$ complexity for calculating the robust mean and variance for each neighborhood.

In the case of implicit weighted batch GP, our data weighers have $O(N \cdot logN)$ computational complexity for constructing the *k*-d tree for the training data set, $O(N \cdot logN)$ computational complexity to search for the neighborhoods of all points in the data set, and $O(N^3)$ complexity to calculate the robust statistics (and weights) for *N* neighborhoods. This leads to an aggregated $O(N^3)$ computational complexity for processing the training data set to calculate weights. Consequently, our implicit weighted batch GP has the same computational complexity than batch GP.

In the case of implicit weighted online GP, a *k*-d tree should be constructed iteratively by adding observations at each learning step. Adding a single observation to a *k*-d tree that contains *t* observations has $O\big(\log(t+1)\big)$ complexity. Consequently, building a *k*-d tree from an empty tree for *N* observations requires $O\big(log(N!)\big)$, which is better than $O(N \cdot logN)$. The same complexity is required for neighborhood searching for the first *N* observations. The use of the *MCD* estimator again requires $O(N^3)$ time complexity. Consequently, the computational

complexity of our implicit weighted online GP is $O(N^3)$, that is the computational complexity of online GP.

The discussion for online GP holds in the case of implicit weighted SOGP, with the exception that processing *N* observations online would eventually require removing observations from the *BV* set, and consequently removing them from the *k*-d tree as well. Removing a data point from a *k*-d tree has the same logarithmic complexity than adding a data point, i.e. $O(\log(n))$, where *n* denotes the number of data points in the tree. Let us consider the worst-case scenario in which we have a *k*-d tree containing *m* observations already, and learning the next *N* observations will trigger *N* additions and *N* removals. That scenario leads to $O(2N \cdot log(m)) = O(N \cdot log(m^2))$ as the complexity for iteratively constructing the tree, and $O(N \cdot log(m))$ for neighborhood searching. The time complexity of using the *MCD* estimator in this case for the *N* neighborhoods is $O(m^3) \leq O(Nm^2)$. Consequently, implicit weighted SOGP and SOGP share the same computational complexity: $O(Nm^2)$.

4.5 Experiments

The different variants of GP regression were implemented in MATLAB. Our implementation of batch GP was validated against the NETLAB toolbox (Nabney, 2004). The online GP and SOGP were validated against Grollman's Online Gaussian Process C++ Library, available at the time of writing at http://brown-rlab.googlecode.com/svn/trunk/SOGP/. Robust means and robust variances were calculated using the function *mcdcov* from the MATLAB library LIBRA (Verboven & Hubert, 2010).

The hyperparameters for the batch GPs and weighted batch GPs were estimated by applying the MLE method to the marginal likelihood. The online GP and SOGP variants used the values of hyperparameters estimated for the corresponding batch variant whenever possible. This was done for two reasons: First, estimating parameters in the case of online GPs is a very difficult problem. Furthermore, it is impossible to obtain reliable parameter values before some critical mass of training data has been learned. Second, given that the main goal of our experiments is to compare the predictive quality of the implicit weighted GPs versus standard GPs, using the best possible parameters in each case supports a more fair comparison (parameters should be better estimated by leveraging all the training data). If the values estimated for batch GP on a particular data set were useless, then all the models in that case employed the values estimated for weighted batch GP. Those cases will be noted throughout this section. The well-known simple exponential kernel was used in our experiments:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{1}{2}\sum_{i=1}^{d} a_i (\mathbf{x_i} - \mathbf{x}_i')^2}, \qquad (4.39)$$

where d denotes the dimensionality of the space $\mathcal{X}$. In order to simplify the hyperparameter space so that the implicit weighted GPs could be easily contrasted to their standard counterparts, the experiments consisted of simulated data with $d = 1$. Consequently, GPs in our experiments have only two hyperparameters: the noise variance σ^2 and the scale parameter a_1 from the kernel. Each training data set was randomly "shuffled", so that the data were not presented to the online GPs as time series. Shuffling was done only once for each experiment, so that training observations were given in the same order to all GP variants.

4.5.1 Heteroscedastic Data without Outliers

This is a simple experiment that shows the potential advantages of weighted GPs when modeling heteroscedastic data. We fabricated a training data set containing two regions, each with a different noise variance. The underlying ground-truth function was $y(x) = 10 \cdot sinc(x)$. A Gaussian noise was added to $y(x)$. The noise variance was set to 0.2 for $x \leq 0$, and to 1.2 for $x > 0$. The training data is shown in Figure 4.1(a).

The parameters estimated for the batch GP were σ^2 = 1.095 and a_1 = 3.5051. Notice that σ^2 is similar to the greater of the two actual variances. Prediction from batch GP is shown in Figure 4.1(b), including the 95% confidence interval. The dotted red line shows the underlying true function. Notice how batch GP over-estimated the variance on the left side of the data. Figure 4.1(c) shows the better fit achieved by a weighted GP that employed a data weigher that assigns a weight = 1 to observations having negative x values and a weight = 0.1667 otherwise. These weights were found by assigning a preliminary weight to each observation equal to the inverse of the corresponding noise variance, and subsequently dividing those preliminary weights by the maximum preliminary weight (i.e., 5). The estimated hyperparameters for the weighted batch GP were σ^2 =0.16481 and a_1 = 3.6243; i.e. this time σ^2 was estimated near the lowest of the actual noise variances. Greater variances for the right side of the graph were achieved through the lower weights in that region.

The predictions from the online models are shown in Figure 4.2 and Figure 4.3. Both weighted GPs were capable of implicitly modeling the different variances. In the case of the sparse GPs, black circles denote the data points kept in the *BV* sets. SOGP retained mostly points from the

right side. However, the weighted SOGP kept mainly observations from the region having the smallest noise variance, which received the greater weights. This supports our expectation: that weighted SOGP would be prone to avoid having data with relatively small weights in the *BV* set.

These results show the potential capabilities of weighted GP for implicitly modeling heteroscedasticity. However, actual noise variances are rarely known. In practical terms, we should compare standard GP models to weighted GP models that rely on more generic data weighers. Consequently, we repeated the experiment employing HeteroscedasticReg, with s = 3, r = 2.0 and γ = 0.05. The values estimated for the weighted GP hyperparameters were σ^2 = 0.13797 and a_1 = 2.8615. The graphs in Figure 4.4 show the predictions that were obtained in this case. The weighted variants of batch and online GP were still able to implicitly model the heteroscedasticity in the data. However, this was not achieved by the weighted SOGP. Despite that, it still retained more data from the left side than SOGP. An increase in the capacity m was required to compensate for the use of a generic data weigher. The weighted SOGP started to model the heteroscedasticity for $m = 28$ (around 35% of the data); as illustrated in Figure 4.4(d).

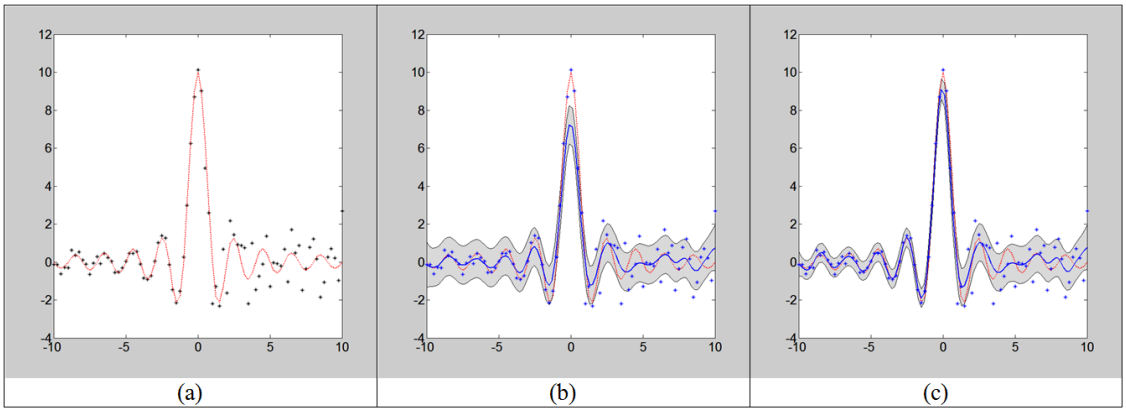

Figure 4.1: (a) The simulated training data set with two regions having different variances. (b) Prediction from batch GP. (c) Prediction from weighted batch GP.

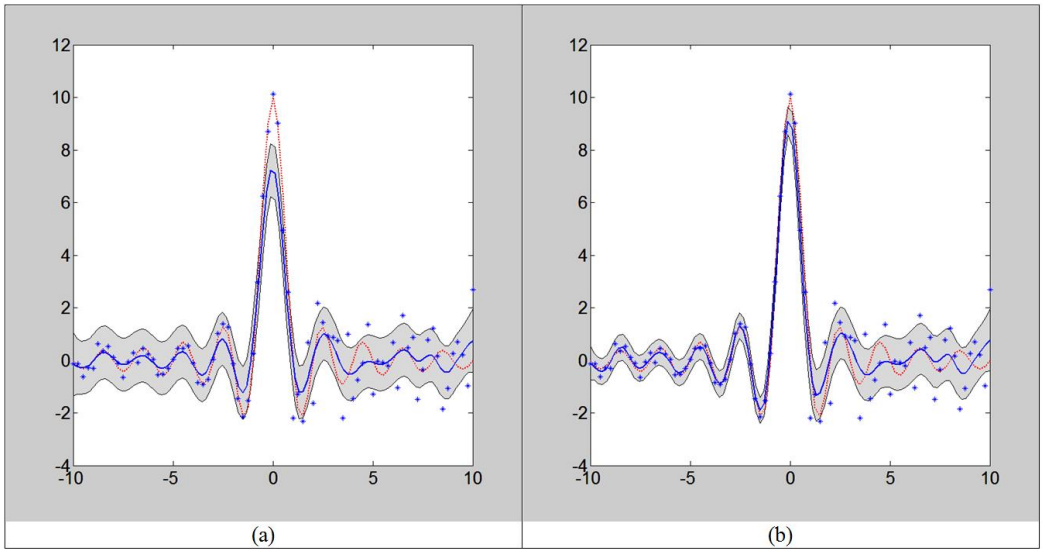

Figure 4.2: Prediction of online GPs on the heteroscedastic data. (a) Online GP (hyperparameters as used in batch GP). (b) Weighted Online GP (hyperparameters as used in weighted batch GP).

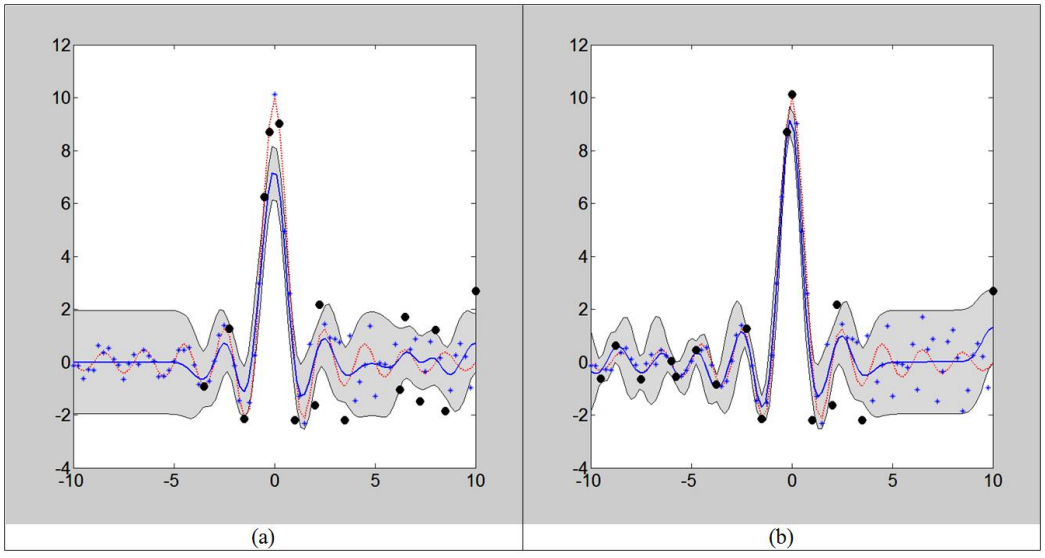

Figure 4.3: Prediction of SOGPs on the heteroscedastic data, where capacity m was set to 16 (~ 20% of data). (a) SOGP (hyperparameters as used in Batch GP). (b) Weighted SOGP (hyperparameters as used in weighted batch GP).

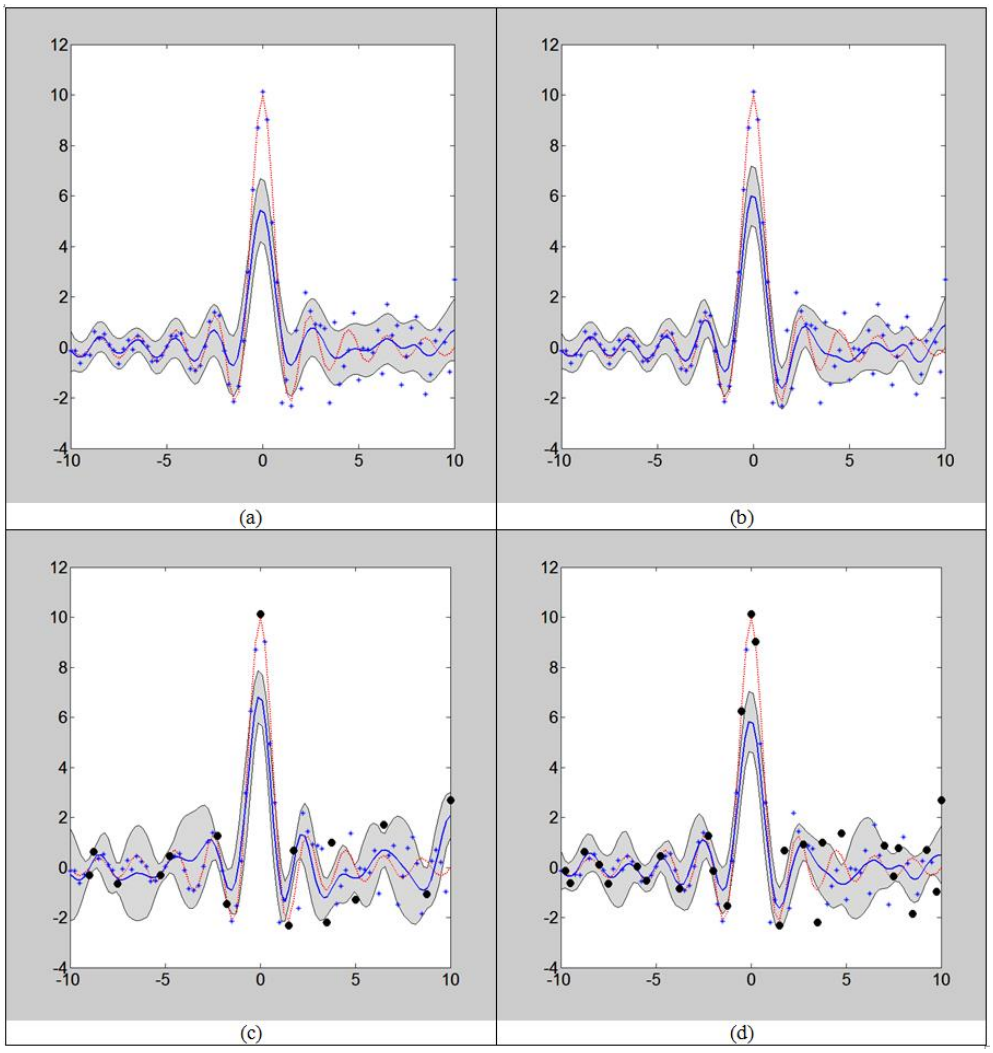

Figure 4.4: Prediction of weighted GPs on the heteroscedastic data. Models were trained using HeteroscedasticReg. (a) Weighted GP. (b) Weighted online GP (hyperparameters as used in weighted batch GP). (c) Weighted SOGP with capacity m = 16, which is ~ 20% of data (hyperparameters as used in weighted batch GP). (d) Weighted SOGP with capacity m = 28, which is ~ 35% of data (hyperparameters as used in weighted batch GP).

4.5.2 Homoscedastic Data with Outliers

Our second experiment focused on learning homoscedastic data containing outliers. The ground-truth function in this case was $y(x) = sin\left(\frac{1}{2}x\right)\left(\frac{10log(x+2)}{x}\right) + \frac{x^2}{200}$. It was sampled at 60 equidistant points from x = 0.25 to 30. A Gaussian noise with variance equal to 0.5 was added to the sample. Subsequently, each observation was randomly considered, with probability 0.1, to be converted into an outlier by adding a value randomly taken from the set $[-10, -8] \cup [8, 10]$. The resulting training data, shown in Figure 4.5(a), contained approximately 10% of outliers.

The hyperparameters of batch GP were estimated as σ^2 = 13.6013 and a_1 = 5.3929e-08. The corresponding prediction rendered a flat line, as shown in Figure 4.5(b). Using weighted batch GP with RobustReg (s = 3, r = 2.0, γ = 0.005), the MLE method gave us σ^2 = 0.20699 and a_1 = 0.10553. The trained weighted batch GP lead to the highly accurate prediction shown in Figure 4.5(c). Finally, prediction from batch GP was greatly improved when we used the same parameters as the weighted batch GP. However, as can be seen in Figure 4.5(d), its results were not as good as those obtained from the weighted batch GP.

The online GP and SOGP models were run employing the hyperparameter values estimated for the weighted batch GP. The corresponding predictions are shown in Figure 4.6 and Figure 4.7. The advantage of using our weighted online GP over the online GP is clear from the graphs. However, the two SOGP variants performed similarly. In the case of our weighted online GP, 14 out of 60 points (approx. 23.33% of the data) were given the default weight. On the other hand, the small capacity forced our weighted SOGP to learn 37 data points (approx. 61.67% of the data) using the default weight. This high percentage of default weights explains why the

weighted SOGP behaved similarly to SOGP; i.e. most training observations were similarly relevant to the weighted model, which is about the same as using SOGP. Figure 4.7(c) shows a histogram of the weights of the basis vectors kept by the weighted SOGP.

When the capacity *m* was increased to 20 (one third of the data), only 14 observations (approx. 23.33% of the data) received the default weight. Results were greatly improved, as can be seen in Figure 4.8, which shows the weighted SOGP prediction and the histogram of weights of the basis vectors after training. As in the first experiment, SOGP tended to keep in the *BV* set observations that were not in agreement with the underlying model, while the weighted SOGP retained fewer observations that deviated from the underlying model, which also received smaller weights.

As a final step, we assessed the robustness of the models. Our comparison took into account only the batch GP variants, given that other variants are approximations to the corresponding batch cases. As before, estimation of hyperparameters for the batch GP led to flat line predictions. Consequently, we compared the robustness of batch GP and weighted batch GP using the values estimated for the weighted batch GP. The predictions of batch GP and weighted batch GP were plotted for data sets having the same ground-truth function used in this second experiment, but containing different percentages of outliers: 5%, 10%, 15%, 20%, 30%, 40% and 50%. The weighted GP employed RobustReg with the same parameter values used before. At 5%, 10% and 15% of outliers the weighted batch GP models were highly accurate and clearly better than the corresponding batch GP models. Both GP variants behaved similarly at around 20% of outliers in the data, still rendering accurate predictions. The performance of both models started to degrade at a similar rate at 30% of outliers. For some of the datasets we also estimated

hyperparameters for the batch GP using as starting point for the MLE method the values from the corresponding weighted GP model. In those cases, batch GP still performed worse than the weighted GP. More details about the relationship between weights and MLE in the case of outliers are given later in this chapter.

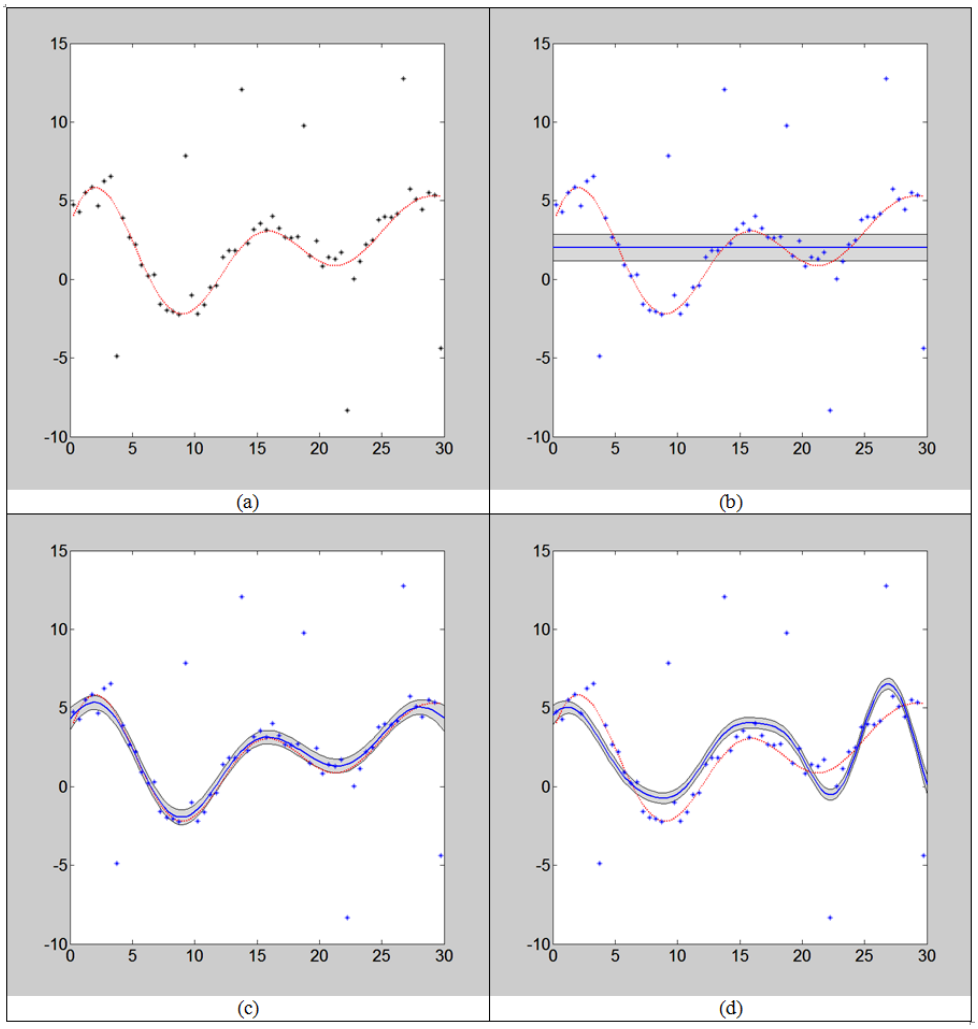

Figure 4.5: (a) Simulated training data set containing outliers. (b) Prediction using batch GP, with GP hyperparameters obtained through MLE. (c) Prediction using weighted batch GP, with GP hyperparameters obtained through MLE. (d) Prediction using batch GP, with GP hyperparameters as were estimated for weighted batch GP.

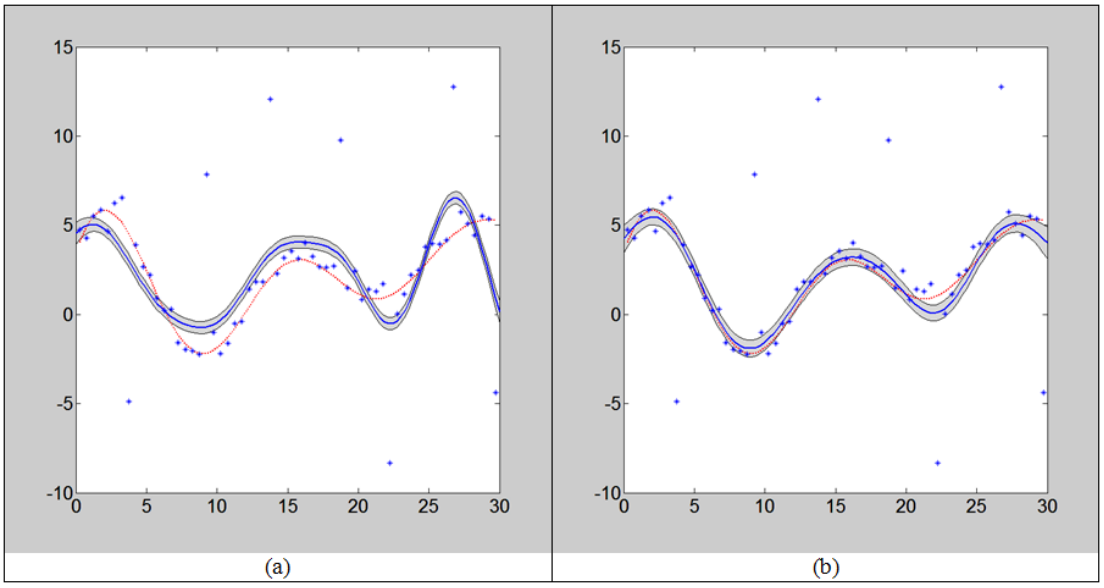

Figure 4.6: Prediction of online GPs on the homoscedastic data with outliers. (a) Online GP (hyperparameters as used in weighted batch GP). (b) Weighted Online GP (hyperparameters as used in weighted batch GP).

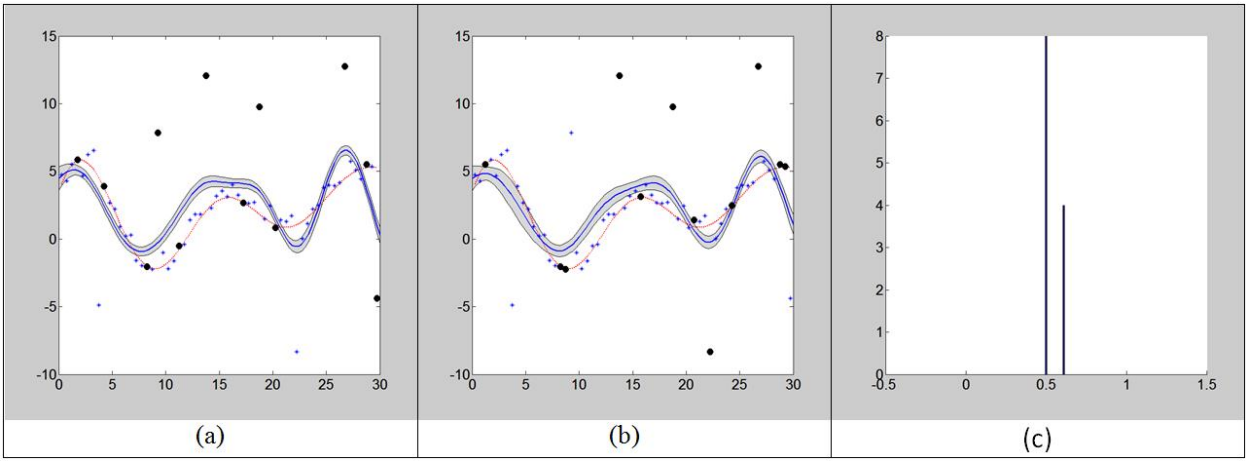

Figure 4.7: Prediction of SOGPs on the homoscedastic data with outliers; capacity m = 12 (~ 20% of data). (a) SOGP (hyperparameters as used in weighted batch GP). (b) Weighted SOGP (hyperparameters as used in weighted batch GP). (c) Histogram of the weights of the final basis vectors of the weighted SOGP.

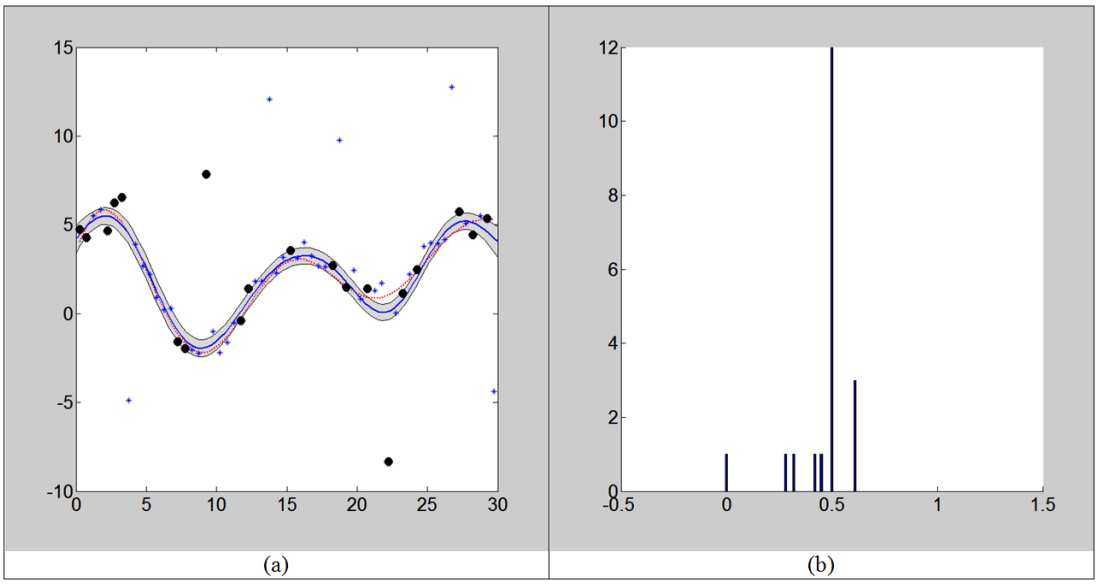

Figure 4.8: Results from weighted SOGP with capacity m = 20. (a) Prediction. (b) Histogram of weights of the final basis vectors.

4.5.3 Heteroscedastic Data with Outliers

The third experiment focused on learning heteroscedastic data containing outliers. The training data were generated as done in the second experiment, except that samples for which x < 15 were affected by a Gaussian noise with variance equal to 0.5 and other samples were affected by a Gaussian noise with variance equal to 1.5. Figure 4.9(a) shows the training data. The hyperparameters estimated for batch GP were σ^2 = 24.1159 and a_1 = 1.2238e-07. Prediction using those values rendered a useless flat line, as shown in Figure 4.9(b). Using weighted batch GP and employing HeteroscedasticRobustReg (s = 3, r = 2.0 and γ = 0.005), the estimation of GP hyperparameters gave us σ^2 = 0.22363 and a_1 =0.12524. The corresponding plot is shown in Figure 4.9(c). Similar to the first experiment, σ^2 was estimated near the smaller of the two actual noise variances. A batch GP using those values for its hyperparameters produced the prediction

shown in Figure 4.9(d), which is clearly inferior to the prediction depicted in Figure 4.9(c). The online GP and SOGP models employed the hyperparameter values of the weighted batch GP. Their predictions are shown in Figure 4.10 and Figure 4.11.

Histograms of the weights assigned by the three weighted GP variants to the observations that remained in the *BV* set after training are shown in Figure 4.12. In the case of weighted batch GP, no observation received the default weight; which might explain its good performance compared to the other models. Weighted online GP learned 15 out of 60 points using the default weight. Weighted SOGP learned 29 data points out of 60 using the default weight (approx. 48.33% of the data). Furthermore, the weights of the majority of vectors in its *BV* set were concentrated in [0.5, 0.6]. These facts help understand why the predictions of weighted SOGP and SOGP were very similar. The prediction from our weighted online GP was similar to the prediction obtained from the standard online GP. In this case most weights were concentrated around 0 and 0.5. Consequently, its predictions should be similar to those obtained from a standard online GP that was trained on a smaller data set containing only the data points with weights significantly greater than zero. This remark gives an insight into why the implicit weighted GPs and the standard online GPs behaved similarly.

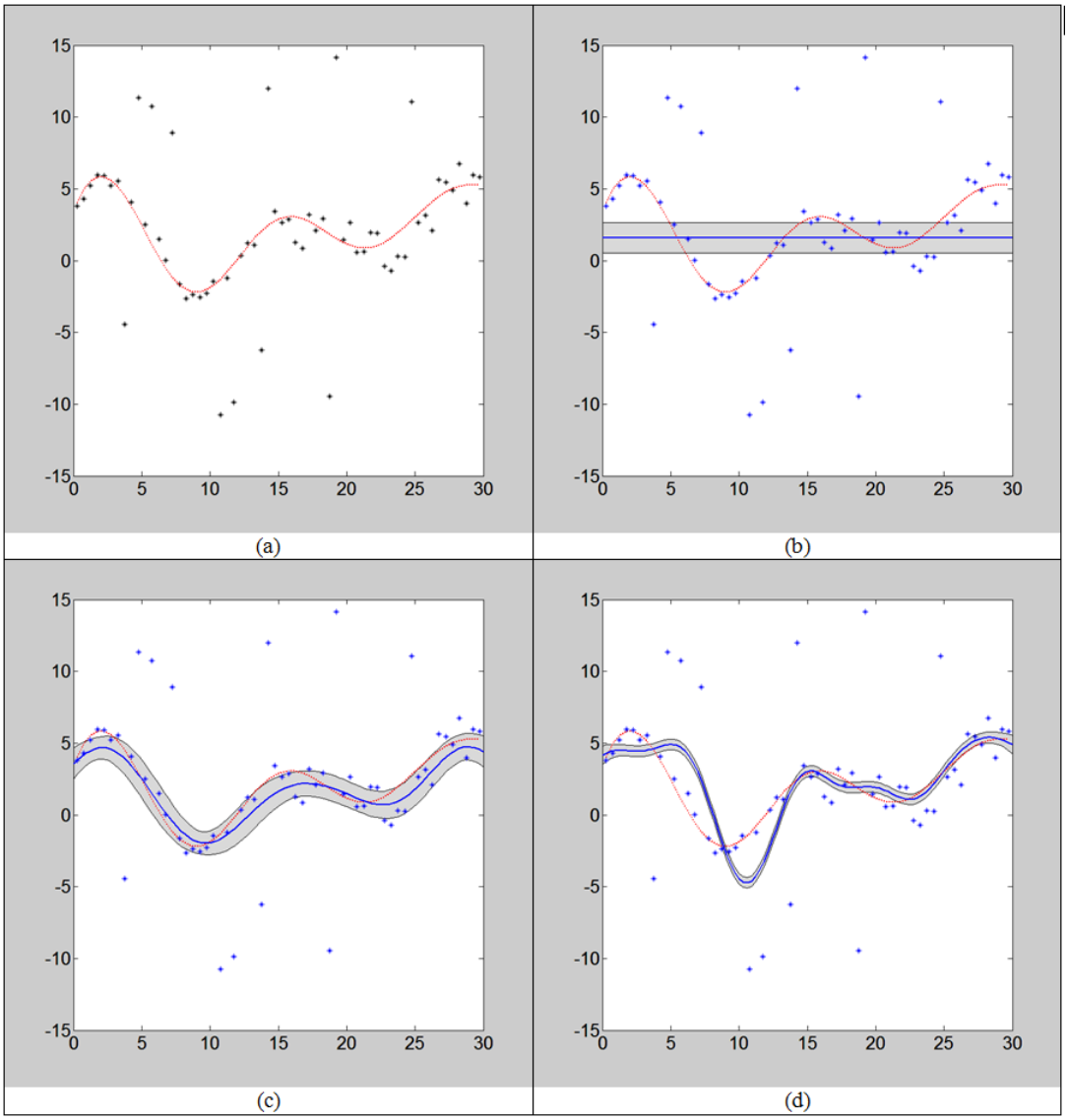

Figure 4.9: (a) The simulated heteroscedastic data set containing outliers. (b) Prediction from the batch GP, with GP hyperparameters obtained through MLE. (c) Prediction from the weighted batch GP, with GP hyperparameters obtained through MLE. (d) Prediction from the batch GP, using values of hyperparameters obtained for weighted batch GP.

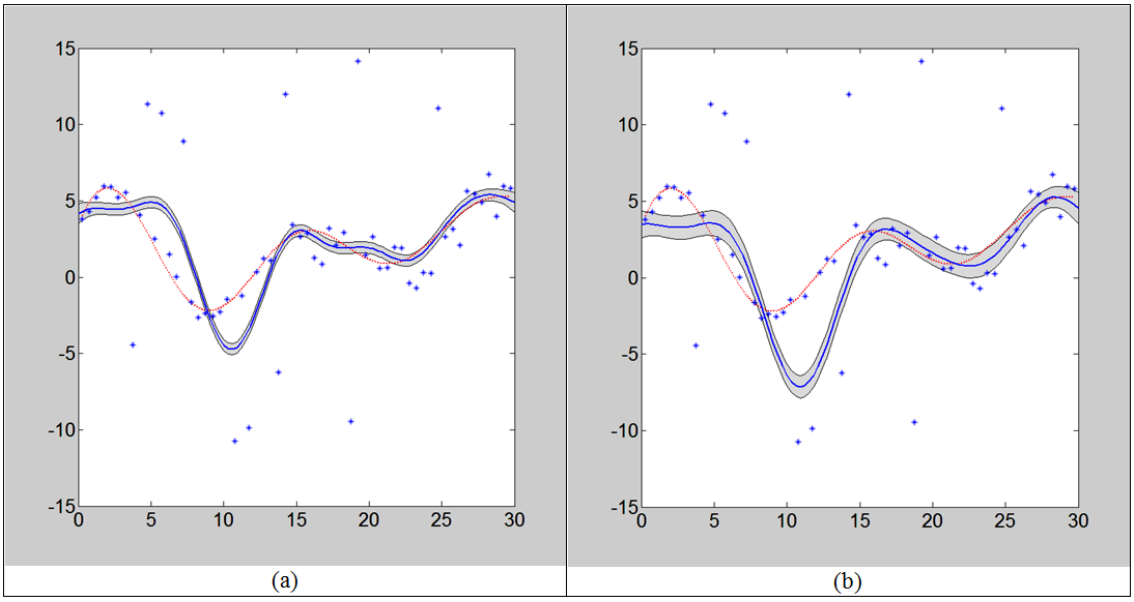

Figure 4.10: Prediction of online GPs trained on the heteroscedastic data with outliers. (a) Online GP (hyperparameters as used in weighted batch GP). (b) Weighted Online GP (hyperparameters as used in weighted batch GP).

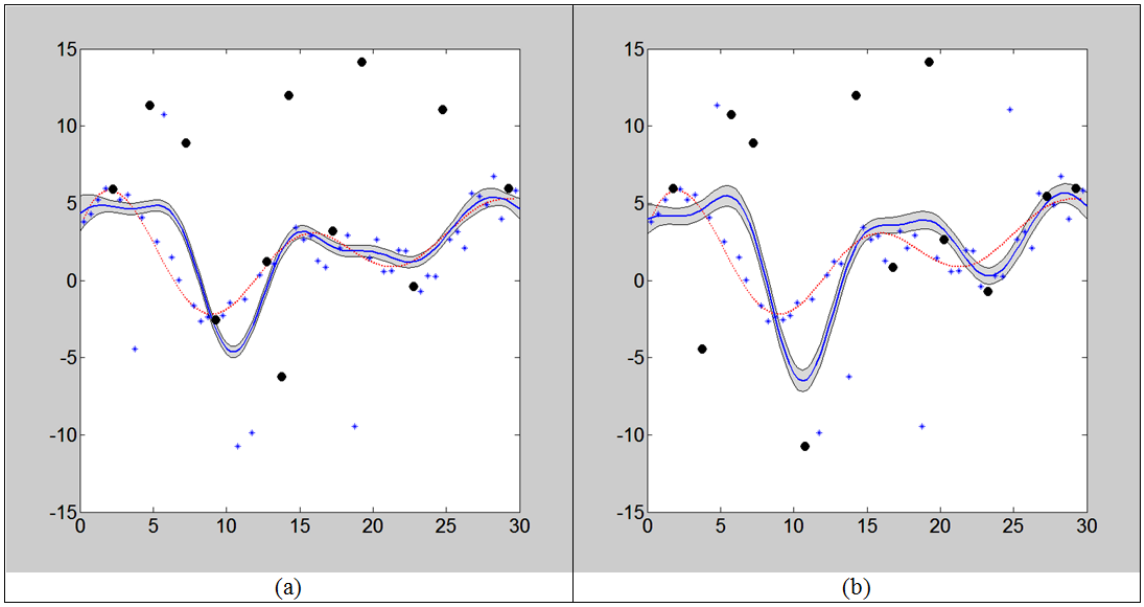

Figure 4.11: Prediction of SOGPs trained on the heteroscedastic data with outliers, where capacity m was set to 12 (~ 20% of data). (a) SOGP (hyperparameters as used in weighted batch GP). (b) Weighted SOGP (hyperparameters as used in weighted batch GP).

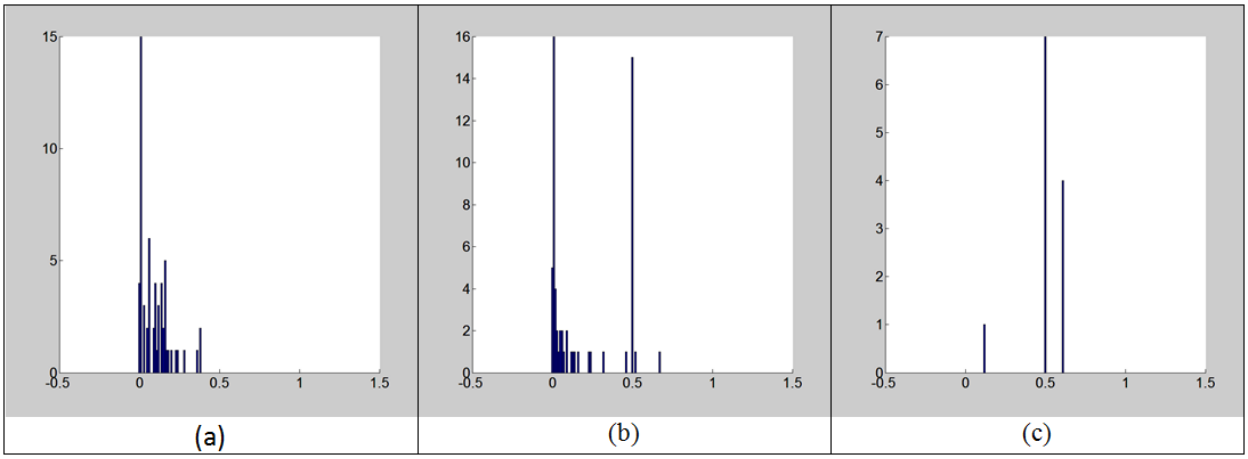

Figure 4.12: Histograms of weights from the three weighted GP models after training. (a) Weights assigned to all data points by weighted batch GP. (b) Weights assigned to final basis vectors by the weighted online GP. (c) Weights assigned to final basis vectors by the weighted SOGP.

4.6 Effect of Weights on the MLE Method

In this section we focus on the difficulty to estimate hyperparameters using the MLE method for the batch GP when the data contain outliers, and why the same estimation method was effective when employed for the weighted batch GPs. Figure 4.13(a) shows the MLE optimization surface corresponding to batch GP from the second experiment. Figure 4.13(b) shows the MLE optimization surface for the weighted batch GP from the same experiment. Note that the optimization surface from batch GP is mostly flat and has no global minimum. The use of weights reshaped the optimization surface so that it had a global minimum and convergence could be easily achieved, as seen in Figure 4.13(b).

We generated a new data set using the sampling procedure from the second experiment, except that it contained no outliers. On this "clean" data, batch GP rendered an accurate prediction, shown in Figure 4.13(d). The estimated hyperparameters: σ^2 = 0.37709 and a_1 = 0.1086. Figure 4.13(c) shows the corresponding optimization surface. This surface is almost identical to that in Figure 4.13(b). Consequently, outliers can make the MLE method ineffective, while using a robust data weigher may allow the MLE method to regain its effectiveness for estimating GP hyperparameters.

To explore this issue further, we ran MLE on the data from the second experiment using prior distributions for the hyperparameters. Both prior distributions were defined as $ln\,\mathcal{N}(0,1)$, to match the starting point (1, 1) previously given to the optimization procedure. The optimization surfaces for batch GP and weighted batch GP were similar to the corresponding surfaces when no priors were used. In the case of weighted batch GP, the estimated values for the hyperparameters were almost the same as before. The values estimated for batch GP (σ^2 = 11.4616, a_1 = 0.16112) were more effective this time, as shown in Figure 4.14. However, even the use of reasonable priors did not allow the standard batch GP to achieve the effectiveness of our weighted batch GP. We repeated the study described in this section for the data set of our third experiment. The same results were obtained.

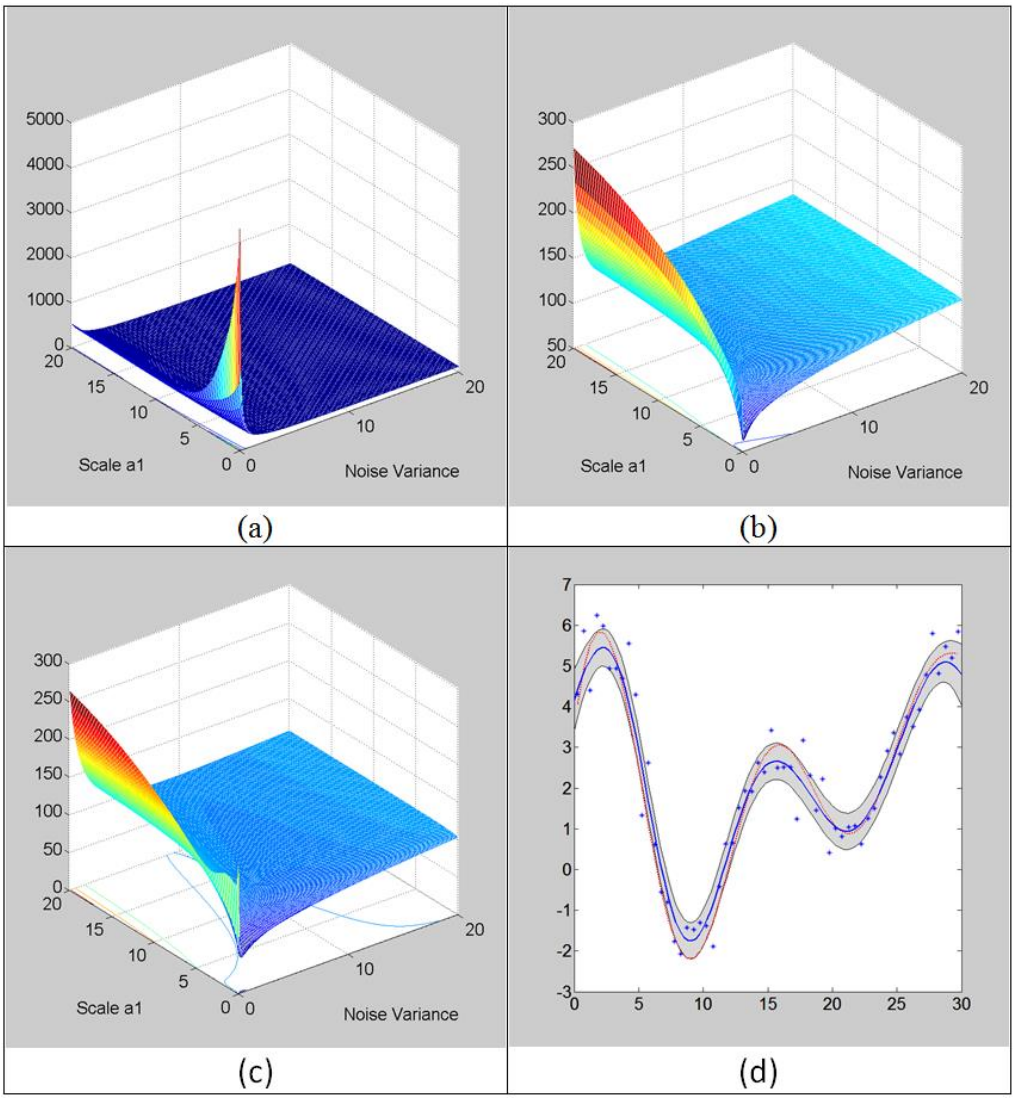

Figure 4.13: (a) MLE optimization surface from batch GP trained on data set with outliers from the second experiment. (b) MLE optimization surface from weighted batch GP trained on data set with outliers from the second experiment. (c) MLE optimization surface from batch GP trained on similar data but without outliers ("clean" data set). (d) Prediction of the batch GP model trained on the "clean" data set.

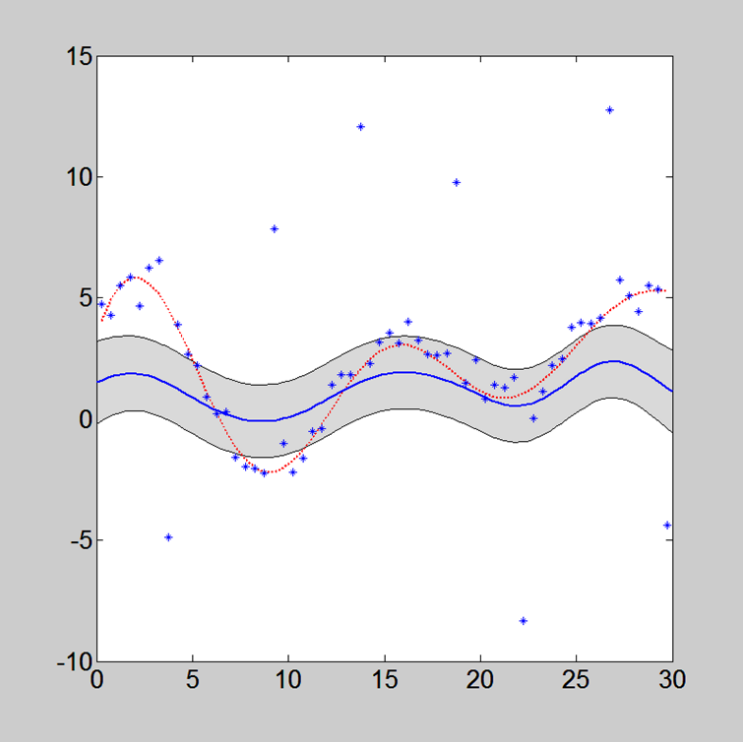

Figure 4.14: Predictions from the batch GP model when trained on the second data set, this time using log normal priors for its hyperparameters.

CHAPTER 5: IMPLICIT WEIGHTED GAUSSIAN PROCESSES FOR NOVELTY DETECTION

The implicit weighted GPs proposed in the previous chapter can be used for novelty detection in the same way as standard GPs were employed in (Kemmler M. , Rodner, Wacker, & Denzler, 2013) and (Ramirez-Padron, Mederos, & Gonzalez, 2013); which was described in section 2.3.4. There is only one modification required: we need a data weigher that can provide a preliminary assessment of the importance of an observation based on its distance to observations that are highly representative of the target class. There are clearly various ways in which a data weigher could be defined. In this chapter, we present a data weigher for novelty detection that relies on the assumption (commonly employed in data mining algorithms) that the importance of an observation is inversely proportional to its distance to a robust mean of the training observations. This data weigher, called here *Robust Data Weigher*, is introduced in the following subsection. Subsequently, this chapter describes the experimental setup used to compare the performances of standard GPs and weighted GPs for novelty detection. A subsequent subsection describes the data sets employed in our experiments, as well as the kernel used in each case. The final two subsections provide the results of our experiments and offer some closing remarks, respectively.

5.1 Robust Data Weigher

Similar to the data weighers described in chapter 4, our robust data weigher leverages the quasi-robust approach to guarantee that the resulting weights take values in the interval $(\gamma,1]$, where $0 < \gamma \ll 0.5$. Given the training data $D = \{\boldsymbol{X}, \mathbf{y}\} = \{(\mathbf{x}_i, \mathrm{y}_i) \mid \mathbf{x}_i \epsilon \mathcal{X},\ \mathrm{y}_i = 1,\ i = 1, \dots, N\}$, let us denote by $\boldsymbol{\mu_X}$ a robust average of all observations in $\boldsymbol{X}$. In our experiments, $\boldsymbol{\mu_X}$ was calculated by

the function *mcdcov* from the MATLAB library LIBRA (Verboven & Hubert, 2010). The *mcdcov* function was also employed to estimate the corresponding robust covariance matrix $\Sigma_{\boldsymbol{X}}$. This matrix is used in the calculation of distances to the robust mean, as described below. For each observation $\mathbf{x}_i$, the robust data weigher for novelty detection is calculated as follows:

$$w_{\mathrm{i}} = (1-\gamma)e^{-d_M(\mathbf{x}_i, \boldsymbol{\mu}_{\boldsymbol{X}})^2} + \gamma\,, \qquad (5.1)$$

where $d_M(\mathbf{x}_i, \boldsymbol{\mu}_{\boldsymbol{X}})$ denotes a robust version of the Mahalanobis distance (Mahalanobis, 1936):

$$d_M(\mathbf{x}_i, \boldsymbol{\mu}_{\boldsymbol{X}}) = (\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{X}})^T {\Sigma_{\boldsymbol{X}}}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{X}})\,. \qquad (5.2)$$

It is important to note that w_{i} receives the default value 0.5 if the number of observations in $\boldsymbol{X}$ is less than a certain threshold *s* (similar to data weighers introduced in the previous chapter). The experiments described in this chapter use different values of *s*, which will be noted in each case. Note also that the use of the Mahalanobis distance in our data weigher implies that the data in $\boldsymbol{X}$ lie on a hyper-ellipsoid. Furthermore, it is assumed that the hyper-ellipsoid that contains the target class has no regions that constitute a potential source of outliers.

Contrary to the case of our implicit weighted GP regression, the data weigher proposed in this chapter applies the *mcdcov* function to observations in $\boldsymbol{X}$ instead of labels. This has a strong negative impact in the computational complexity of weight calculation, which turns out to be $\left(N^{\frac{d(d+3)}{2}}\right)$, where *d* denotes the dimensionality of the input space $\mathcal{X}$. Because of this polynomial complexity, our data weigher can only be applied to input spaces of small dimensionality. In particular, the *mcdcov* function works with data sets of up to 50 dimensions. Consequently, for problems of higher dimensionality the data have to be projected into low-dimensional subspaces for our data weigher to be able to calculate their weights.

5.2 Experimental Setup

The main purposes of our experiments are (1) to determine whether each weighted GP variant is capable of outperforming the corresponding standard GP, (2) to determine how the performance of the different online GPs compare to the performance of batch GP-based novelty detection, and (3) to compare the four membership scores employed in (Kemmler M. , Rodner, Wacker, & Denzler, 2013). Our experiments considered four data sets, which are described in the following section. The first three data sets consist of a single target class each, while the fourth data set consists of eight separate target classes. In that latter case, our experimental setup (which is described below) was executed independently on each target class. In other words, for each target class, its observations became the normal observations while observations from the other classes were used as contamination sources and for testing purposes. Consequently, the fourth data set is actually the source of eight different training data sets for our experimental purposes.

At each experiment, the labels in the training data for members and not members of the target class are all known. We employ 10-fold stratified cross-validation (CV) (Kohavi, 1995) to validate the GP-based novelty detectors, each detector using one membership score at a time. Stratified CV delivers training data folds that contain roughly the same class proportions as in the training data. Consequently, given the use of 10-fold CV, all GPs were trained on approximately 90% of the target class at each CV step. SOGPs were cross-validated at two different capacities: $m = 10$ and $m = 30$.

To assess the performance of each GP for each membership score, we leveraged *receiver operating characteristic* (ROC) curves (Fawcett T. , 2006). An ROC curve is a graph commonly

used in machine learning to visualize the performance of a binary classifier, by plotting the *true positive rate* (proportion of positive observations correctly classified as such) against the *false positive rate* (proportion of negative observations that were incorrectly classified) of the classifier. In our case, positive observations correspond to true members of the normal class. An ROC curve is constructed by plotting these two rates at numerous discrimination thresholds that range from the minimum to the maximum of the scores provided by the classifier on a particular data set. In our case, the membership scores obtained on a training data from each stratified CV were used to obtain each ROC curve. The overall quality of each novelty detector was assessed by estimating the area under its ROC curve, called *AUC value* (Fawcett T. , 2006). AUC values tend to be in the interval [0.5, 1], where AUC values near 0.5 correspond to a random binary classifier. A widely-used rule of thumb is to categorize the quality of classifiers according to the traditional academic grading system: excellent classifiers have AUC values between 0.9 and 1, classifiers with AUC values from 0.8 to 0.9 are typically considered good, and those having AUC values from 0.7 to 0.8 are considered fair. Classifiers with AUC values that are less than 0.6 are considered failed models and are typically discarded.

In our experiments, the 10-fold stratified CVs were repeated 30 times for each combination of GP variant and membership score, so that 30 ROC curves (and consequently 30 AUC values) were obtained in each case. In total, an application of our experimental setup on a single training data set generated 960 AUC values (from the combinations of eight GP instances and four membership scores). The cross-validation procedures were implemented by the author in Matlab, leveraging the functions *cvpartition* and *perfcurve* from the Matlab Statistics toolbox.

Given that the data sets did not contain outliers originally, the experimental setup just described was also run on various "contaminated" versions of the data sets, in order to compare the performance of the GP-based novelty detectors when the training data contained outliers. Let us denote by l, where $l \in [0,1]$, a level of contamination; i.e. the percentage of observations labeled as members of the target class that are outliers. For positive values of l, we calculated how many observations from the non-target classes had to be added to the target class to achieve the contamination level l:

$$observations\_to\_add = \left\lceil \frac{l\,N_1}{1-l} \right\rceil, \qquad (5.3)$$

where N_1 denotes the number of observations originally in the data set that are members of the target class. If $observations\_to\_add \geq 1$, then the original data set was contaminated before executing each particular CV run by randomly choosing that number of observations from the non-target class and temporarily labeling them as members of the target class. This allowed us to obtain training data sets with the required level of outliers. Employing this contamination procedure, our experimental setup was repeated for the following contamination levels: 0.05, 0.10, 0.15, and 0.20.

5.2.1 Comparison of Standard GPs and Weighted GPs

In order to compare each variant of standard GP (i.e. batch, online, SOGP with m = 10, and SOGP with m = 30) against the corresponding weighted GP variant, the set of AUC values from each combination of data set, contamination level and membership score were analyzed by a one-way ANOVA. The significance level α for the ANOVAs was set to 0.01 instead of the most commonly used 0.05, to compensate for the multiple inferences in this case. From those AUC

differences that were significant at the 0.01 significance level, we counted those for which the absolute value of average AUC difference represented a relative change in AUC of at least 2%. This was done to gather statistics not only on significant differences but also on differences that were both significant and represented a noticeable effect on the quality of outlier detection.

5.2.2 Comparison of Batch GPs and Online GPs

In order to simplify the comparison of batch GPs and the different online GPs (i.e. Online GP, SOGP with m = 10, and SOGP with m = 30), we decided to employ a single score per data set. For each data set, if there was a single score that significantly outperformed all other scores at various levels of outlier contamination, then that score is chosen as the suitable score for the data set. Otherwise, our comparison would be performed by employing a score that fulfills the following two conditions: (1) it was not the worst score for the data set and (2) it exhibited good performance across most data sets. Because of this dependency on selecting suitable scores, the comparison between batch and online GPs was actually performed after the comparison of the scores. Consequently, it is described in the last subsection of the experimental results, despite being the second goal of this dissertation.

The comparison was implemented separately on results corresponding to standard GPs and on results corresponding to weighted GPs. This was done to assess whether batch GPs and online GPs compared similarly in both cases. In each case, multivariate one-way ANOVAs were run on the AUC values obtained from each experiment at each contamination level, to determine whether there were significant differences between the performances of the four GPs under comparison. If a multivariate ANOVA indicated that AUC differences between the GPs were

significant at the 0.01 significance threshold α, then the AUC values were subsequently compared pairwise by employing Tukey HSD test for multiple comparisons (Lane, 2010). The significance level for the pairwise comparisons was also set to 0.01 to compensate for the multiple tests.

Once the results from the ANOVAs and Tukey HSD were acquired, we obtained a ranking of the different types of GPs for each data set and each contamination level. The rankings were established through the following procedure: If the multivariate ANOVA indicated that differences in AUC were not significant, then all GPs under comparison are allocated to the same rank (Rank 1). Otherwise, the pairwise comparisons from Tukey HSD test are employed to attempt to allocate the different GPs into as many ranks as possible. This was done in a way that, given any rank $Rank_i$, all GPs allocated to $Rank_i$ showed significantly better performance than the GPs in all subsequent ranks according to the pairwise comparisons. As a final step, the appearances of each GP type in each particular rank were counted, aggregating over all contamination levels.

5.2.3 Comparison of Scores

The membership scores are compared using only the experimental results from standard batch GP and weighted batch GP, given that the different types of online GPs are essentially approximations to the corresponding batch GP. The differences in performance for the different scores were analyzed separately for the cases of batch GP and weighted batch GP. A multivariate one-way ANOVA was run to decide in each case whether there were significant differences between the AUC values from the four scores. If a multivariate ANOVA indicated significant

differences between the four scores at the 0.01 significance threshold α, then the scores were compared pairwise by employing Tukey HSD test, using 0.01 as the significance level. We based our analysis of the pairwise comparisons on counting the instances (for each data set and contamination level) of two different cases: (1) one of the scores was significantly better than the others, and (2) one of the scores was significantly worse than the others. This was done in order to further reduce the likelihood of erroneously accepting differences between scores as significant when they were actually due to chance.

5.3 Data Sets and Kernels

This section describes the data sets employed in our experiments. Note that the first data set was generated as a proof of concept. It is one of the simplest data sets that can be used to compare the weighted and standard GP variants under different levels of contamination. The rest of the data sets belong to real-life problems of varying levels of difficulty.

5.3.1 Points within Circles

This data set consists of a set of two-dimensional points. Observations from the target class were generated as random points in the circle centered at (0, 0) with a radius equal to 20. The target class consisted of 100 points. Four small groups of 10 points each were generated as random points within circles of radius = 3, centered at different locations: (-30, -30), (30, -30), (-30, 30), and (30, 30). Figure 5.1 (below) shows the particular data set employed in our experiments. Weighted GP-based novelty detection should perform significantly better than standard GP-based novelty detection on this data set.

The simple squared exponential kernel given in equation 4.39 was employed in this case. As mentioned above, automatic estimation of hyperparameters for GP-based novelty detection is an open problem not addressed in this work. We relied on the intuitive interpretation of the parameters of the simple squared exponential kernel, assigning to each scale parameter a_i the inverse of the robust variance of the corresponding attribute, as calculated by the mcdcov function of the LIBRA library (Verboven & Hubert, 2010). This estimation approach was used on other data sets whenever the simple squared exponential kernel was employed. The GP hyperparameter noise variance was set to 0.0001 (any small value should be fine here, to denote the lack of noise in the labels). Finally, the robust data weigher (5.1) was configured so that $\gamma = 0.0001$ and at least 5 observations were required to calculate a non-default weight.

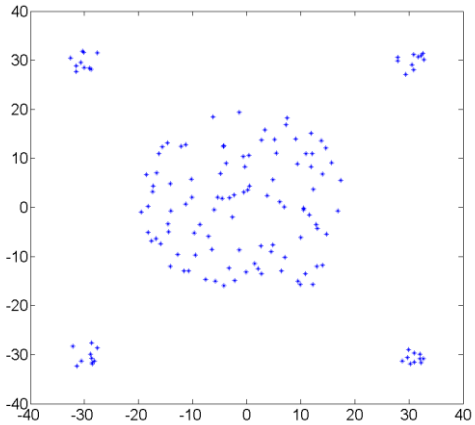

Figure 5.1: The simple "Points within Circles" data set. Random observations on the center correspond to the target class. The small clusters on the corners are used as outliers, both as a source of contamination and for testing purposes.

5.3.2 Vertebral Column

The Vertebral Column data set was retrieved from the UCI Machine Learning Repository (Lichman, 2013). It was originally compiled by Dr. Henrique da Mota during a medical residence in the Group of Applied Research in Orthopaedics (GARO) of the Centre Médico-Chirurgical et de Réadaptation des Massues, in France. It consists of 100 observations corresponding to patients having a normal vertebral column and 210 observations taken from the same number of abnormal patients (60 patients had disk hernias and 150 patients had a displacement of vertebras called Spondylolisthesis). There are six numeric attributes in the data set, which denote properties derived from the shape and orientation of the pelvis and lumbar spine: (1) pelvic incidence, (2) pelvic tilt, (3) lumbar lordosis angle, (4) sacral slope, (5) pelvic radius and (6) grade of spondylolisthesis. The observations corresponding to the normal patients were considered in our experiments as examples of the target class.

The simple squared exponential kernel was employed in this case. As mentioned above, estimation of hyperparameters was done by assigning to each scale parameter a_i the inverse of the robust variance of the corresponding attribute, as calculated by the mcdcov function of the LIBRA library (Verboven & Hubert, 2010). The GP hyperparameter noise variance was set to 0.0001. Finally, the robust data weigher (5.1) was configured so that $\gamma = 0.0001$ and at least 12 observations were required to calculate a non-default weight.

5.3.3 Pima Indians Diabetes

This data set consists of 768 observations, each having eight numeric attributes (not counting the class attribute). It contains features of women at least 21 years old of Pima Indian heritage, and

was obtained also from the UCI Machine Learning Repository (Lichman, 2013). The class labels indicate whether the corresponding patient tested positive for diabetes. In our experiments the absence of diabetes was considered the target class. There are 500 observations corresponding to non-diabetic patients and 268 observations from women that were diagnosed as diabetic. The eight attributes are listed below:

1. Number of times pregnant
2. Plasma glucose concentration from an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

As noted in the UCI repository, this data set contains zeroes in places where zero cannot be a valid value, so they most likely denote missing values. Given that the main goal of our experiments is to assess the effectiveness of weighted GPs as robust novelty detectors, no rows were omitted in our experiments and no efforts were made to compensate for missing values. The simple squared exponential kernel given in equation 4.39 was also employed in this case. Similar to the experiments with the Vertebral Column data set, each scale parameter a_i was assigned the inverse of the robust variance of the corresponding attribute. The GP

hyperparameter noise variance was set to 0.0001. The robust data weigher (5.1) was configured so that $\gamma = 0.0001$ and at least 16 observations were required to calculate a non-default weight.

5.3.4 Caltech 101

We decided to use the Caltech 101 image database (Fei-Fei, Fergus, & Perona, 2004), given the good performance of GP-based novelty detection on it, reported in (Kemmler M. , Rodner, Wacker, & Denzler, 2013). This was particularly true when using the spatial pyramid matching (SPM) kernel (Lazebnik, Schmid, & Ponce, 2006). Additionally, using Caltech 101 allows us to contrast our results to those of Kemmler, Rodner, Wacker & Denzler (2013). Caltech 101 contains pictures of objects taken from 101 categories. The size of each image is approximately 300 x 200 pixels. Contrary to the work of Kemmler et al., which focused on average performance of novelty detectors across all image categories, our work focused on eight individual object categories from Caltech 101. Those categories were chosen here based on experimental results reported by (Lazebnik, Schmid, & Ponce, 2006). The SPM kernel achieved high classification performance on four of them (minaret, Windsor chair, Joshua tree, and okapi), and poor classification performance on the remaining four categories (cougar body, beaver, crocodile, and ant). These eight categories were chosen for our experiments because they were identified as key examples in the work of Lazebnik et al. Our experiments were run independently for each of the eight categories serving as target classes; while the remaining categories remained in the training data for doing the cross-validations and also as a source of contamination when adding the different levels of outliers to each target class. Our experiments employed the SPM kernel as well. For that reason, SPM is briefly introduced next.

The SPM kernel works on descriptor vectors calculated on the images that are input to the kernel. It takes into account coarse spatial information about local features from those images. The SPM kernel makes use of the pyramid match kernel (Grauman & Darrell, 2005). Consequently, it is important to understand pyramid matching in order to fully understand SPM. Let us denote two sets of local features obtained from two images as X_1 and X_2. Primary examples of such local features are SIFT descriptors (Lowe, 1999), which were used in our experiments. Both feature sets X_1 and X_2 must take values in the same *d*-dimensional feature space. The original pyramid matching kernel is applied as follows: A sequence of increasingly finer grids with resolutions 0, 1, … , L is placed over the space of local features so that at each resolution *l*, the corresponding grid has a total of $D_l = 2^{dl}$ cells. Any two points, one from X_1 and the other from X_2, are a match at resolution *l* if they fall into the same cell at that resolution. At each resolution *l*, a histogram is built for each feature set, with each bin corresponding to a different grid cell. The histogram intersection function $I(.,.)$ (Swain & Ballard, 1991) is used to calculate the total number of feature matches at a given resolution *l*:

$$I_l = I\left(H_{X_1}^l, H_{X_2}^l\right) = \sum_{i=1}^{D_l} \min\left(H_{X_1}^l(i), H_{X_2}^l(i)\right) \qquad (5.4)$$

where $H_{X_1}^l$ and $H_{X_2}^l$ denotes the histograms at resolution *l* for X_1 and X_2 respectively. Finally, the pyramid match kernel κ_L is calculated as follows:

$$\kappa_L(\mathrm{X}_1, \mathrm{X}_2) = \frac{1}{2^L} I_0 + \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} I_l \qquad (5.5)$$

The SPM kernel uses spatial information by applying the pyramid match kernel in the two-dimensional image space instead of the space of local features. Before doing that, each feature set is quantized into *M* feature types, where *M* is a fixed number. SPM then applies the pyramid

match kernel in the image space *M* times, each time constrained to the coordinates associated to the corresponding feature type. The SPM kernel is defined as the sum of pyramid match kernels on the image space over the *M* feature types:

$$k_L(\mathrm{X}_1, \mathrm{X}_2) = \sum_{m=1}^{M} \kappa_L\left(C_{X_1}^m, C_{X_2}^m\right) \qquad (5.6)$$

where $C_{X_i}^m$ denotes the image coordinates associated to features from the feature set X_i that are of type *m*. A normalization of histograms by the total weight of all features in the image permits the evaluation of the kernel on images of different sizes.

A pre-processing step is needed to create a dictionary of size *M*, which is used to quantize the feature set from each image. The elements of the dictionary were selected in (Lazebnik, Schmid, & Ponce, 2006) as the centroids of the *M* clusters obtained by applying the *k*-means algorithm to features taken from all classes from multi-class classification problems. Quantization of each feature vector was performed by choosing its nearest element from the dictionary. It was shown by Lazebnik et al. that support vector machines using the SPM kernel outperform other modern classifiers on three image datasets, including the Caltech 101 database (Fei-Fei, Fergus, & Perona, 2004).

In our experiments, we used the Matlab implementation of the SPM kernel from (Lazebnik, Schmid, & Ponce, 2006), keeping their recommended values for the parameters: *M* = 200 and *L* = 2. Similarly, we used their default SIFT descriptors of 16 x 16 pixel patches computed over a dense grid spaced at eight pixels. However, we had to generate a dictionary for each target class, given that we are modeling a target class at a time instead of dealing with a multi-classification

problem. The pyramid histograms for all images from the eight categories were built against the dictionary of the particular target class to be learned in each case.

Contrary to the previous data sets, the observations in this case were high-dimensional, corresponding to pyramid histograms of length 4,200 each. In this case, we employed ROBPCA (Hubert, Rousseeuw, & Vanden Branden, 2005) to project the data into a low-dimensional subspace before calculating the weights. The projected data consisted of the robust principal components that made for 95% of the variance in the data, or the first 50 components if more than 50 components were required to cover up to 95% of the variance (the author is not aware of any instance in which such cutoff was needed). In our experiments, projection by ROBPCA was employed only for the purpose of weight calculation. In other words, the training data was handed entirely to the corresponding GP, given that the SPM kernel required the data in its original format in order to work properly.

5.4 Experiment Results and Analyses

This section contains three subsections. The first one summarizes the results from the one-way ANOVAs that compared the performance of standard and weighted GPs. The second subsection contains the results from the comparison of batch GPs and online GPs. The third subsection contains the analysis of the performance of the four scores. As a final note, all standard and weighted GP variants performed very poorly on the target class "ant" from Caltech 101, with most AUC values falling in the interval (0.5, 0.6). Consequently, the "ant" class was excluded from our experiment results and the corresponding analysis. The "ant" target class was already identified in (Lazebnik, Schmid, & Ponce, 2006) as a class in which multi-classification kernel

methods using the SPM kernel showed a particularly poor performance. As noted in (Ramirez-Padron, Mederos, & Gonzalez, 2013), the SPM kernel is not invariant to translations and/or rotations. Consequently, images containing exactly the same object at different locations and/or rotated are considered different objects by the kernel function. Not only was this the case for the images in the "ant" class, but images of ants ranged from hand-drawn sketches of ants to photos of ants in different positions with very different backgrounds.

5.4.1 Comparison of Standard GPs and Weighted GPs

The tables in this subsection list how many times the performance of weighted GP-based novelty detection showed significant differences at the 0.01 significance level when compared to the corresponding standard GP, aggregating over the four membership scores. The significant differences in AUC values were categorized as either positive or negative relative changes. Relative changes were calculated as follows:

$$Relative\_change = 100\frac{(meanAUC_{Weighted\ GP} - meanAUC_{Standard\ GP})}{|meanAUC_{Standard\ GP}|}. \qquad (5.7)$$

Consequently, a positive relative change indicates the percentage by which weighted GP-based novelty detection outperformed novelty detection based on the corresponding standard GP. Similarly, a negative relative change indicates the percentage by which weighted GP-based novelty detection underperformed compared to novelty detection based on the corresponding standard GP. Additionally, each table lists the number of positive and negative significant differences that corresponded to a relative change in AUC greater or equal than 2% (in absolute values).

5.4.1.1 Points within Circles

The table below shows that the proposed weighted GPs significantly outperformed standard GPs in the vast majority of cases for this data set. Note that standard GPs were never able to outperform our weighted GPs.

Table 5.1: Number of positive and negative relative changes in AUC that were significant at $\alpha = 0.01$, and how many of them had its absolute value greater or equal than 2%. Points within Circles data set.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch				
0	Online				
0	SOGP_m10	1		1	
0	SOGP_m30				
5	Batch	4		1	
5	Online	4		1	
5	SOGP_m10	4		4	
5	SOGP_m30	4		4	
10	Batch	4		4	
10	Online	4		4	
10	SOGP_m10	4		4	
10	SOGP_m30	4		4	
15	Batch	4		4	
15	Online	4		4	
15	SOGP_m10	4		4	
15	SOGP_m30	4		4	
20	Batch	4		4	
20	Online	4		4	
20	SOGP_m10	4		4	
20	SOGP_m30	4		4	
All Combined	Batch	16		13	
All Combined	Online	16		13	
All Combined	SOGP_m10	17		17	
All Combined	SOGP_m30	16		16	

5.4.1.2 Vertebral Column

The table below shows that our weighted bath GPs clearly outperformed the standard batch GPs, while the opposite was never the case. However, there is not a clear winner between the two GP variants if we considered all types of GPs (i.e. batch and online GPs combined).

Table 5.2: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Vertebral Column data set.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	1			
0	Online				
0	SOGP_m10				
0	SOGP_m30	3			
5	Batch	3			
5	Online				
5	SOGP_m10				
5	SOGP_m30	2			
10	Batch	4			
10	Online		1		
10	SOGP_m10				
10	SOGP_m30		3		
15	Batch	4			
15	Online		3		
15	SOGP_m10				
15	SOGP_m30		1		
20	Batch	4			
20	Online		2		
20	SOGP_m10				
20	SOGP_m30	1		1	
All Combined	Batch	16			
All Combined	Online		6		
All Combined	SOGP_m10				
All Combined	SOGP_m30	6	4	1	

5.4.1.3 Pima Indians Diabetes

The values in Table 5.3 shows that our weighted GPs clearly outperformed the corresponding standard GPs in all cases except for SOGPm_10, which clearly was too limited in capacity to benefit from our data weigher. Interestingly, from the significant positive differences only those corresponding to SOGP_m30 represented an increase in AUC value that exceeded a 2% relative change.

Table 5.3: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Pima Indians Diabetes data set.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	4			
0	Online	4			
0	SOGP_m10				
0	SOGP_m30	4		3	
5	Batch	4			
5	Online	4			
5	SOGP_m10				
5	SOGP_m30	4		4	
10	Batch	4			
10	Online	4			
10	SOGP_m10				
10	SOGP_m30	4		3	
15	Batch	4			
15	Online	4			
15	SOGP_m10				
15	SOGP_m30	4		1	
20	Batch	4			
20	Online	4			
20	SOGP_m10				
20	SOGP_m30	4		3	
All Combined	Batch	20			
All Combined	Online	20			
All Combined	SOGP_m10				
All Combined	SOGP_m30	20		14	

5.4.1.4 Caltech 101

Individual Target Classes

The values shown in tables ranging from Table 5.4 to Table 5.10 indicate mixed results for the Caltech 101 data set. Our weighted GPs outperformed standard GPs for the target classes Beaver, Cougar Body, Crocodile and Joshua Tree. On the other hand, results for the target classes Minaret, Okapi and Windsor Chair favor the standard GPs. Note however that for all target classes our weighted batch GPs outperformed standard batch GPs. More insight about this regularity and a possible explanation for the mixed results in the case of online GPs are provided in section 5.4.1.5, which contains an analysis of the results from all data sets.

Table 5.4: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Beaver target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	2	1		
0	Online	3			
0	SOGP_m10		2		
0	SOGP_m30	3			
5	Batch	3		2	
5	Online	4		3	
5	SOGP_m10	1		1	
5	SOGP_m30	4		4	
10	Batch	4		2	
10	Online	4		4	
10	SOGP_m10				
10	SOGP_m30	4		4	
15	Batch	4		4	
15	Online	4		4	
15	SOGP_m10	2		2	
15	SOGP_m30	4		4	
20	Batch	4		4	
20	Online	4		4	
20	SOGP_m10	3		3	
20	SOGP_m30	4		4	
All Combined	Batch	17	1	12	
All Combined	Online	19		15	
All Combined	SOGP_m10	6	2	6	
All Combined	SOGP_m30	19		16	

Table 5.5: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Cougar Body target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	3		1	
0	Online		4		
0	SOGP_m10	1	1	1	1
0	SOGP_m30		4		
5	Batch	4		4	
5	Online				
5	SOGP_m10	1	1	1	1
5	SOGP_m30				
10	Batch	4		4	
10	Online	4		4	
10	SOGP_m10		1		1
10	SOGP_m30	4		4	
15	Batch	4		4	
15	Online	4		4	
15	SOGP_m10	1		1	
15	SOGP_m30	4		4	
20	Batch	4		4	
20	Online	4		4	
20	SOGP_m10	1		1	
20	SOGP_m30	4		4	
All Combined	Batch	19		17	
All Combined	Online	12	4	12	
All Combined	SOGP_m10	4	3	4	3
All Combined	SOGP_m30	12	4	12	

Table 5.6: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Crocodile target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	4		4	
0	Online	4			
0	SOGP_m10	3		2	
0	SOGP_m30	4			
5	Batch	4		4	
5	Online	4		4	
5	SOGP_m10	3		3	
5	SOGP_m30	3		3	
10	Batch	4		4	
10	Online	4		4	
10	SOGP_m10	2		2	
10	SOGP_m30	3		3	
15	Batch	4		4	
15	Online	4		4	
15	SOGP_m10	2		2	
15	SOGP_m30	4		4	
20	Batch	4		4	
20	Online	4		4	
20	SOGP_m10	3		3	
20	SOGP_m30	4		4	
All Combined	Batch	20		20	
All Combined	Online	20		16	
All Combined	SOGP_m10	13		12	
All Combined	SOGP_m30	18		14	

Table 5.7: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Joshua Tree target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch		4		2
0	Online		4		2
0	SOGP_m10	2	2		
0	SOGP_m30		4		3
5	Batch	1			
5	Online				
5	SOGP_m10	1		1	
5	SOGP_m30		1		1
10	Batch	4		4	
10	Online				
10	SOGP_m10	2		2	
10	SOGP_m30		1		
15	Batch	4		4	
15	Online	3		3	
15	SOGP_m10	3		3	
15	SOGP_m30	2		2	
20	Batch	4		4	
20	Online	3		3	
20	SOGP_m10	3		3	
20	SOGP_m30	2		2	
All Combined	Batch	13	4	12	2
All Combined	Online	6	4	6	2
All Combined	SOGP_m10	11	2	9	
All Combined	SOGP_m30	4	6	4	4

Table 5.8: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Minaret target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch		3		
0	Online		4		
0	SOGP_m10		4		
0	SOGP_m30		4		
5	Batch	4			
5	Online		3		
5	SOGP_m10		3		3
5	SOGP_m30		3		1
10	Batch	4		3	
10	Online		3		2
10	SOGP_m10		4		4
10	SOGP_m30		1		
15	Batch	4		4	
15	Online		3		3
15	SOGP_m10		4		4
15	SOGP_m30		1		1
20	Batch	4		4	
20	Online		4		4
20	SOGP_m10		3		3
20	SOGP_m30		3		3
All Combined	Batch	16	3	11	
All Combined	Online		17		9
All Combined	SOGP_m10		18		14
All Combined	SOGP_m30		12		5

Table 5.9: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Okapi target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch		4		
0	Online		4		
0	SOGP_m10		4		2
0	SOGP_m30		4		
5	Batch				
5	Online		2		
5	SOGP_m10		3		3
5	SOGP_m30		2		
10	Batch	2		1	
10	Online		1		
10	SOGP_m10		2		2
10	SOGP_m30		2		
15	Batch	3		2	
15	Online		2		2
15	SOGP_m10		3		3
15	SOGP_m30		1		
20	Batch	4		3	
20	Online				
20	SOGP_m10		2		2
20	SOGP_m30				
All Combined	Batch	9	4	6	
All Combined	Online		9		2
All Combined	SOGP_m10		14		12
All Combined	SOGP_m30		9		

Table 5.10: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. Windsor Chair target class.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	4			
0	Online		3		
0	SOGP_m10		1		
0	SOGP_m30	1	2		
5	Batch	4		4	
5	Online		4		4
5	SOGP_m10		3		3
5	SOGP_m30		3		3
10	Batch	4		4	
10	Online		4		4
10	SOGP_m10		4		4
10	SOGP_m30		4		4
15	Batch	4		4	
15	Online		4		4
15	SOGP_m10		4		4
15	SOGP_m30		4		4
20	Batch	4		4	
20	Online		4		4
20	SOGP_m10		1		1
20	SOGP_m30		4		4
All Combined	Batch	20		16	
All Combined	Online		19		16
All Combined	SOGP_m10		13		12
All Combined	SOGP_m30	1	17		15

All Target Classes Combined

Table 5.11 contains the totals obtained by aggregating the results from Table 5.4 to Table 5.10. It is clear by looking at this table that results are highly mixed for the Caltech 101 data set. However, it is also evident that for all target classes our weighted batch GPs outperformed standard batch GPs in the vast majority of cases. As mentioned above, more insight about this regularity is offered in section 5.4.1.5.

Table 5.11: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Caltech 101 data set. All classes combined.

		Significant differences alpha = 0.01		Significant differences alpha = 0.01 and Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	13	12	5	2
0	Online	7	19		2
0	SOGP_m10	6	14	3	3
0	SOGP_m30	8	18		3
5	Batch	20		14	
5	Online	8	9	7	4
5	SOGP_m10	6	10	6	10
5	SOGP_m30	7	9	7	5
10	Batch	26		22	
10	Online	12	8	12	6
10	SOGP_m10	4	11	4	11
10	SOGP_m30	11	8	11	4
15	Batch	27		26	
15	Online	15	9	15	9
15	SOGP_m10	8	11	8	11
15	SOGP_m30	14	6	14	5
20	Batch	28		27	
20	Online	15	8	15	8
20	SOGP_m10	10	6	10	6
20	SOGP_m30	14	7	14	7
All Combined	Batch	114	12	94	2
All Combined	Online	57	53	49	29
All Combined	SOGP_m10	34	52	31	41
All Combined	SOGP_m30	54	48	46	24

5.4.1.5 Analysis of Results

The weighted GP variants greatly outperformed the corresponding standard GPs for all positive contamination levels when data from the Points within Circles data set became contaminated with outliers, as shown in Table 5.1. This outstanding performance was obtained because of the simplicity of the corresponding detection problem. In the case of Pima Indians Diabetes, it is remarkable that there was not a single occasion in which standard GPs significantly outperformed weighted GPs, as can be seen in Table 5.3. On the other hand, weighted GPs significantly outperformed standard GPs for all GP types except for SOGP with m = 10. This result suggests that m = 10 is a very low capacity value for this particular problem. Interestingly,

significant differences related to relative AUC changes of at least 2% corresponded only to weighted SOGP with m = 30. It is not clear to the author why other weighted GPs did not achieve the same level of positive relative change. A possible explanation is that the capacity m = 30 permitted the weighted SOGP to properly learn from the data but at the same it enforced the removal of superfluous or potentially misleading observations. This suggestion is based on extrapolating the observation from the previous chapter on how our weighted SOGP tended to retain less outlying data points than standard SOGP.

In the case of classes from Caltech 101, if we looked at the results from all classes combined, shown in Table 5.11, there were more cases in which weighted GPs outperformed the standard GPs than the opposite, except –again– for SOGP with m = 10. However, there were several occasions in which standard GPs outperformed weighted GPs as well. At first glance, it seems that there was not a clear winner between weighted and standard GPs for the Caltech 101 data set. However, if we considered only the case of weighted batch GPs vs. standard batch GPs, then a clear regularity emerges: novelty detection based on weighted batch GP consistently and significantly outperformed novelty detection based on standard batch GP for all target classes and all positive contamination levels. Given that online GP and SOGP are approximations to batch GP, the regularity mentioned above substantiates the superiority of weighted GP-based novelty detection also in the case of the Caltech 101 data set.

Finally, results from the Vertebral Column data set showed the least number of significant differences, with various cases in which standard GPs significantly outperformed weighted GPs. However, the regularity mentioned above appears here as well: weighted batch GPs consistently

and significantly outperformed standard batch GPs for all positive contamination levels, as shown in Table 5.2. Furthermore, there was not a single case in which standard batch GP significantly outperformed weighted batch GP when data was contaminated with outliers. If we restricted our analysis only to the case of significant differences corresponding to relative AUC changes of at least 2%, note that only one significant difference remains: a case in which weighted SOGP with $m = 30$ outperformed the corresponding standard SOGP at 20% contamination level. The author believes that the same possible reason given above for a similar case from the Pima Indians Diabetes data set may apply in this case.

The aggregated results presented in Table 5.12 offer the big picture of differences in performance across all data sets. These results clearly highlight the regularity across all individual data sets noted above (i.e. that novelty detection based on weighted batch GP consistently and significantly outperformed novelty detection based on standard batch GP whenever data was contaminated with outliers). This regularity validates our hypothesis that our implicit weighted GPs perform better than standard GPs when training data is contaminated with outliers.

Table 5.12: Number of positive and negative relative changes in AUC that were significant at α = 0.01, and how many of them had absolute value greater or equal than 2%. Results aggregated from all data sets.

		Significant differences alpha = 0.01		Abs(Relative_Change_Perc) >= 2%	
ContaminationLevel	GPType	Positive_AUC_Changes	Negative_AUC_Changes	Positive_AUC_Changes	Negative_AUC_Changes
0	Batch	18	12	5	2
0	Online	11	19	0	2
0	SOGP_m10	7	14	4	3
0	SOGP_m30	15	18	3	3
5	Batch	31		15	
5	Online	16	9	8	4
5	SOGP_m10	10	10	10	10
5	SOGP_m30	17	9	15	5
10	Batch	38		26	
10	Online	20	9	16	6
10	SOGP_m10	8	11	8	11
10	SOGP_m30	19	11	18	4
15	Batch	39		30	
15	Online	23	12	19	9
15	SOGP_m10	12	11	12	11
15	SOGP_m30	22	7	19	5
20	Batch	40		31	
20	Online	23	10	19	8
20	SOGP_m10	14	6	14	6
20	SOGP_m30	23	7	22	7
All Combined	Batch	166	12	107	2
All Combined	Online	93	59	62	29
All Combined	SOGP_m10	51	52	48	41
All Combined	SOGP_m30	96	52	77	24

5.4.2 Comparison of Scores

This section analyzes the tables that resulted from the analysis of the multivariate one-way ANOVAs employed to compare the performance of the four scores. This analysis was done in two steps. The first step consisted in obtaining one table per data set that registered whether there was a best and worst novelty score for each combination of outlier contamination level and GP type (based on the existence of pair-wise significant differences from Tukey HSD test, as described in section 5.2). Note that this first table contains 10 rows, given that two types of GPs (weighted batch GP and standard batch GP) were run at each of the five contamination levels (0%, 5%, 10%, 15% and 20%). For the cases in which no score was significantly better or worse

than the other scores, the corresponding table cells were labeled as "NA". The second step consisted in aggregating into a second table the information from the corresponding first table over the different contamination levels. Each second table shows how many times each score was significantly better or worse than the others for each GP type. Given that our conclusions, which are presented in a following section, were based on the aggregated tables from the second step, only those tables are presented in this subsection. The reader can refer to Appendix A to review the first set of tables, which served as the source for the tables presented here.

5.4.2.1 Points within Circles

The following table indicates that the Mean membership score was the worst score for the Points within Circles data set in four out of ten instances. No score was significantly better than the rest for this data set.

Table 5.13: Counting best and worst novelty detection scores, aggregated over all contamination levels. Points within Circles data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		4						
WeightedBatch								
TOTALS		4						

5.4.2.2 Vertebral Column

The following table shows that the Mean score performed significantly worse than the other scores for the Vertebral Column data set. As with the previous data set, no score was significantly better than the others in this case.

Table 5.14: Counting best and worst novelty detection scores, aggregated over all contamination levels. Vertebral Column data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		5						
WeightedBatch		2						
TOTALS		7						

5.4.2.3 Pima Indians Diabetes

In the case of the Pima Indians Diabetes data set, we obtained that the Negative Variance score was the best score in various occasions. This can be seen in Table 5.15 below. Note that the Mean score was significantly outperformed by the other scores for all GP types and all contamination levels.

Table 5.15: Counting best and worst novelty detection scores, aggregated over all contamination levels. Pima Indians Diabetes data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		5	2					
WeightedBatch		5	4					
TOTALS		10	6					

5.4.2.4 Caltech 101

Individual Target Classes

The values shown in tables ranging from Table 5.16 to Table 5.22 depicts a mixed set of results for the Caltech 101. This is not surprising given the different types of objects with different backgrounds contained in the images of each target class. However, a few regularities are worth of mentioning here. The Mean score behaved unreliably, ranging from the best score in very few cases to the worst score in multiple cases. The Negative Variance score was the worst score for multiple target classes, and it never was the best score. Finally, the Probability and Heuristic

scores appeared at least once as the best score for two and three target classes, respectively. There was no target class for which these two scores could be labeled as the worst score.

Table 5.16: Counting best and worst novelty detection scores, aggregated over all contamination levels. Beaver data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		2						
WeightedBatch		5			3			
TOTALS		7			3			

Table 5.17: Counting best and worst novelty detection scores, aggregated over all contamination levels. Cougar Body data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		1		1				
WeightedBatch		2		1				
TOTALS		3		2				

Table 5.18: Counting best and worst novelty detection scores, aggregated over all contamination levels. Crocodile data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch	1			1				
WeightedBatch	1			4				
TOTALS	2			5				

Table 5.19: Counting best and worst novelty detection scores, aggregated over all contamination levels. Joshua Tree data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch				1				
WeightedBatch	1			5				
TOTALS	1			6				

Table 5.20: Counting best and worst novelty detection scores, aggregated over all contamination levels. Minaret data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch		3		1				
WeightedBatch				3				
TOTALS		3		4				

Table 5.21: Counting best and worst novelty detection scores, aggregated over all contamination levels. Okapi data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch				4			1	
WeightedBatch				3				
TOTALS				7			1	

Table 5.22: Counting best and worst novelty detection scores, aggregated over all contamination levels. Windsor Chair data set.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch				2			4	
WeightedBatch				5				
TOTALS				7			4	

All Target Classes Combined

The table that appears below aggregates the results from the individual target classes of the Caltech 101 data set. The regularities noted above for the case of the individual target classes are easier to spot when these aggregated results are considered.

Table 5.23: Counting of best and worst novelty detection scores. All target classes combined.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch	1	6		10			5	
WeightedBatch	2	7		21	3			
TOTALS	3	13		31	3		5	

5.4.2.5 Analysis of Results

Various conclusions can be derived from the tables that compare the membership scores. Arguably the most important conclusion is that no score can be considered better or worse than the other three in absolute terms. However, there are some interesting observations to be made. The Mean score showed a fluctuating performance across all data sets, leaning in the majority of cases towards bad performance, a characteristic that was also reported by (Kemmler M. , Rodner, Wacker, & Denzler, 2013). The Negative Variance score was the score of choice in (Kemmler M. , Rodner, Wacker, & Denzler, 2013). However, this score performed worse than the Probability and Heuristic scores on all the data sets except Pima Indian Diabetes. It is difficult to determine the reasons behind this difference in results, given that the work by Kemmler et al. averaged the AUC values across all 101 classes contained in Caltech 101. Furthermore, although their work also employed the SPM kernel, they used a different clustering technique to build their dictionaries. In any case, our results show that Negative Variance (a.k.a. GP-Reg-*V*) is not necessarily the score of choice for visual object recognition and other tasks. For the data sets considered here, the Heuristic and Probability scores performed similarly or better than the other two scores in most cases. Additionally, these two scores were the only scores that never appeared as the worst score in our experiments. This result in conjunction with the disappointing performance of the Mean and Negative Variance scores suggest that membership scores that use a combination of the posterior mean and the posterior variance are more appropriate for GP-based novelty detection in most cases. Interestingly, the Heuristic and Probability scores were the only scores in the work by Kemmler et al. that significantly outperformed SVDD for the two kernels used in that research. That fact was not highlighted in their paper given the good average

performance of the Negative Variance score when they used their preferred SPM kernel. Table 5.24 aggregates the results discussed in this section across all data sets. The conclusions offered here can be seen clearly from the aggregated values.

Table 5.24: Counting of best and worst novelty detection scores. All target classes combined.

/ Score	Mean		NegVariance		Probability		Heuristic	
GPType	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Batch	1	20	2	10			5	
WeightedBatch	2	14	4	21	3			
TOTALS	3	34	6	31	3		5	

5.4.3 Comparison of Batch GPs and Online GPs

As mentioned above, the first step in order to compare batch GPs and the different types of online GPs was to select a suitable membership score for each data set, employing the approach described in section 5.2.2. The membership score selected for each data set is listed in Table 5.25.

The following subsections contain tables that show the number of times each type of GP was allocated to each particular rank, aggregating over all contamination levels. Note that standard GPs and weighted GPs were aggregated separately. The reader may refer to Appendix B to look at the same information listed for individual contamination levels.

Table 5.25: Suitable membership score for each data set.

Data set	Suitable Membership Score
Points within Circles	Heuristic
Vertebral Column	Heuristic
Pima Indians Diabetes	Negative Variance
Caltech 101	Heuristic

5.4.3.1 Points within Circles

The table shown below contains the rank allocation for the different GP types, aggregated over all contamination levels. It is clear that batch GP and online GP were the best performers, both in the standard and weighted cases. SOGP_m30 showed as good performance as online GP and batch GP only when using weighted GPs. The fact that SOGP_m10 occupied the last rank in most cases indicates that its capacity was not appropriate for this problem.

Table 5.26: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Points within Circles data set.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10	1		4	
	SOGP_m30	1	4		
Weighted	BatchGP	5			
	OnlineGP	5			
	SOGP_m10	1	4		
	SOGP_m30	5			

5.4.3.2 Vertebral Column

Similarly to the previous data set, online GP and batch GP consistently shared Rank 1 as the best performers. SOGP_m30 was consistently assigned to Rank 2. SOGP_m10 was consistently allocated to the last rank. Again, this indicates that capacity equal to 10 was not appropriate for this problem. These results can be seen below, in Table 5.27.

Table 5.27: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Vertebral Column data set.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10			5	
	SOGP_m30		5		
Weighted	BatchGP	5			
	OnlineGP	5			
	SOGP_m10			5	
	SOGP_m30		5		

5.4.3.3 Pima Indians Diabetes

The results for the Pima Indians Diabetes were very similar to those obtained for the previous two data sets. They are shown in the table below.

Table 5.28: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Pima Indians Diabetes data set.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10		4	1	
	SOGP_m30		5		
Weighted	BatchGP	5			
	OnlineGP	5			
	SOGP_m10			5	
	SOGP_m30		5		

5.4.3.4 Caltech 101

Individual Target Classes

The results for the different target classes of the Caltech 101 data set were different depending on whether the standard GPs or our weighted GPs were employed. In the case of standard GPs, online GP and batch GP shared Rank 1 as the best performers in the vast majority of cases, and SOGPm_30 was capable of achieving similar performances in many cases. SOGP_m10 was

typically relegated to Rank 2, although in a few cases it was also able to match the performance of batch GP and online GP. In the case of weighted GPs, there were more differences between the different GP types. In various cases batch GP outperformed online GP, and SOGP_m10 tended to occupy Ranks 3 and 4 for multiple target classes. The following tables show the results corresponding to each target class. Section 5.4.3.5 offers some insight regarding the different results obtained when employing standard GPs and weighted GPs.

Table 5.29: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Beaver target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10	3	2		
	SOGP_m30	5			
Weighted	BatchGP	2	3		
	OnlineGP	4	1		
	SOGP_m10		2	3	
	SOGP_m30	5			

Table 5.30: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Cougar Body target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10	3	2		
	SOGP_m30	5			
Weighted	BatchGP	5			
	OnlineGP	3	2		
	SOGP_m10		3	2	
	SOGP_m30	3	2		

Table 5.31: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Crocodile target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10	4	1		
	SOGP_m30	5			
Weighted	BatchGP	5			
	OnlineGP	3	2		
	SOGP_m10	1	2	2	
	SOGP_m30	2	3		

Table 5.32: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Joshua Tree target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10		4	1	
	SOGP_m30	4	1		
Weighted	BatchGP	5			
	OnlineGP	2	3		
	SOGP_m10			2	3
	SOGP_m30		2	3	

Table 5.33: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Minaret target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	4	1		
	OnlineGP	4	1		
	SOGP_m10		1	4	
	SOGP_m30	2	3		
Weighted	BatchGP	5			
	OnlineGP	1	4		
	SOGP_m10		1	1	3
	SOGP_m30	1	1	3	

Table 5.34: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Okapi target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10		5		
	SOGP_m30	5			
Weighted	BatchGP	5			
	OnlineGP	3	2		
	SOGP_m10		3	2	
	SOGP_m30	3	2		

Table 5.35: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. Windsor Chair target class.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	5			
	OnlineGP	5			
	SOGP_m10		4	1	
	SOGP_m30	4	1		
Weighted	BatchGP	5			
	OnlineGP		5		
	SOGP_m10			4	1
	SOGP_m30		4	1	

All Target Classes Combined

The following table aggregates the results from the individual target classes. The results mentioned above for the individual target classes still can be seen here.

Table 5.36: Counting of rank allocation for each particular GP type, aggregated over contamination levels. Caltech 101 data set. All target classes combined.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	34	1		
	OnlineGP	34	1		
	SOGP_m10	10	19	6	
	SOGP_m30	30	5		
Weighted	BatchGP	32	3		
	OnlineGP	16	19		
	SOGP_m10	1	11	16	7
	SOGP_m30	14	14	7	

5.4.3.5 Analysis of Results

Table 5.37, shown below, aggregates the previous results over all data sets. It provides a big picture of how the different GP types compared to each other, both for standard GPs and weighted GPs.

Table 5.37: Counting of rank allocation for each particular GP type, aggregated over contamination levels and all data sets.

Standard/Weighted	GP Type	Rank 1	Rank 2	Rank 3	Rank 4
Standard	BatchGP	49	1		
	OnlineGP	49	1		
	SOGP_m10	11	23	16	
	SOGP_m30	31	19		
Weighted	BatchGP	47	3		
	OnlineGP	31	19		
	SOGP_m10	2	15	26	7
	SOGP_m30	19	24	7	

Three conclusions are apparent from the results shown in Table 5.37, as well as from the notes from the previous subsections:

(1) Online GP can provide as good performances as batch GP when using standard GP variants, based on how batch GP and (non-sparse) online GP were both allocated to Rank 1 in 49 cases (out of 50 cases in total). However, if we were using our weighted GP variants, then batch GP took greater advantage of weights than online GPs: our weighted online GP was assigned to Rank 1 in 31 cases, while in the other 19 cases it was downgraded to Rank 2. Comparatively, our weighted batch GP occupied Rank 1 in 47 cases. Taking into account that weighted GPs consistently and significantly outperformed standard GPs for all data sets only in the batch case (as concluded in section 5.4.1.5), it is reasonable to hypothesize that such significant difference is the main reason for having significant differences between weighted batch GP and weighted online GP. However, the author also noticed that in a few cases, such effect was intensified by an

actual decrease in performance of weighted online GP compared to standard online GP (this can be seen in the tables of Appendix B, which contain the average performance of the different GP types for each contamination level). In those cases, apparently the set of observations used to train the weighted online GPs were not large enough to offset the possibly misleading influence of imprecise weights calculated at the beginning of the training process (when few observations are available to the online GP). Finally, note that all cases in which online GP and batch GP were not assigned to Rank 1 correspond to the more complex Caltech 101 data set. This is can be easily verified by looking at the tables corresponding to other data sets (i.e. Table 5.26, Table 5.27 and Table 5.28).

(2) The performances of SOGP_m10 were typically allocated to the last ranks, indicating that capacity $m = 10$ was insufficient to grasp the difficulties of the problems at hand.

(3) In many cases SOGP_m30 shared the rank of the best performer batch GP. This makes SOGP an attractive alternative to consider when implementing GP-based novelty detection on systems with strong memory constraints, as far as a suitable capacity limit can be employed. This was particularly evident when using standard GPs, where SOGP_m30 was allocated to rank 1 in 62% of the experiments. When our weighted GP variants were used, this was the case in only 38% of the experiments. Apparently SOGP_m30 was affected by misleading weights that were calculated at the beginning of the training process, as it should have been the case with online GP as well. Still, note that the effect of initial imprecise weights should fade with time, provided online GPs are trained on a long enough sequence of observations.

CHAPTER 6: CONCLUSIONS

This chapter concludes this dissertation. It provides a brief summary of the work and results described in the previous chapters. Additionally, it offers conclusions that are derived from our results, and suggests further research that can departure from the theoretical and experimental work described here.

6.1 Summary

The increasing amount of data present in real-life problems, its variety, and the ever growing need to process data at faster speeds, make the problem of automated novelty detection particularly important. Multiple methods and approaches have been proposed to address this problem. Most of these are reviewed in chapter 1 of this dissertation. Kernel methods taken from the classification approach have been particularly successful, such as the Support Vector Data Description (SVDD) method (Tax & Duin, 2004), one-class SVM (Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson, 2001) and Online SVDD (Tax & Laskov, 2003). It has been proposed recently to use Gaussian processes (GPs) as part of an approach to novelty detection that builds membership scores based on the predictive distribution of GPs (Kemmler M. , Rodner, Wacker, & Denzler, 2013). This GP-based novelty detection approach has been used with great success on multiple real-life problems, and it has been proven to outperform state-of-the-art methods such as SVDD and one-class SVM (Kemmler M. , Rodner, Wacker, & Denzler, 2013), (Krishna, Bodesheim, & Denzler, 2013). Additionally, GP-based novelty detection also

benefits from various advantages associated to Bayesian learning methods, which were briefly reviewed in chapter 2 of this dissertation.

Despite the recent success of GP-based novelty detection, that approach has a potential limitation that was demonstrated with a simple example in chapter 3: standard GPs employing Gaussian likelihoods are highly sensitive to outliers in the training data (Jylänki, Vanhatalo, & Vehtari, 2011). This limitation is particularly evident when maximum likelihood estimation (MLE) is employed to estimate the hyperparameters of the GP model. MLE is commonly used to estimate hyperparameters, but it has been shown that it is highly sensitive to outliers in the data (Agostinelli & Greco, 2013). Current efforts to address this problem include the use of pseudo-likelihoods (Greco, Racugno, & Ventura, 2008) and likelihoods corresponding to robust distributions (Jylänki, Vanhatalo, & Vehtari, 2011). However, these approaches lead to analytically intractable inferences, which involve the use of approximation techniques that are typically complex, computationally expensive and/or inefficient. The work in (Agostinelli & Greco, 2013) proposes the use of weighted likelihoods in Bayes formula to obtain robust Bayesian inferences. Weighted likelihoods are defined in (Agostinelli & Greco, 2013) as joint likelihoods in which weight functions serve as exponents of each likelihood term. Aside from being restrictive regarding the location of the weight functions and not being specifically applied to GPs, the type of weight functions proposed in (Agostinelli & Greco, 2013) might be expensive to compute, given its dependency on parameter estimates and the empirical cumulative distribution function. The work in (Rottmann & Burgard, 2010) employs weights in GPs in order to model heteroscedastic data. Their approach employs weights to estimate a noise level parameter for each training observation, using cross-validation twice and introducing a

complicated dual-GP model. That approach leads to greatly increasing the computational complexity of calculating the posterior GP. In summary, we are not aware of any work that employs weighted likelihoods in GPs to model either data that contain outliers or heteroscedastic data.

The main motivation for the work described in this dissertation is to address the lack of robustness of standard GPs described above by using weight functions within the likelihood terms. In the case of GP regression, this is done in a way that the computational complexity of our proposed implicit weighted GPs is the same as the computational complexity of standard GPs. These goals included obtaining implicit weighted variants of batch GP, online GP, and sparse online GP (SOGP). Although our main focus is robustness, this dissertation also proposes a weight function that allows an implicit weighted GP to effectively model heteroscedastic data. Additionally, this work focuses on a comprehensive experimental study of the advantages that our robust weighted GPs would convey to the GP-based novelty detection approach described in (Kemmler M. , Rodner, Wacker, & Denzler, 2013). We were particularly interested in studying the performance of online GPs given the preliminary experimental work described in (Ramirez-Padron, Mederos, & Gonzalez, 2013), which shows that the performance of GP-based novelty detection using online GPs can be similar to the performance of batch GP-based novelty detection in many cases.

The main contributions of this study are listed below:

1) The first chapter of this dissertation expands a relatively recent survey on novelty (anomaly) detection offered in (Chandola, Banerjee, & Kumar, 2009), by adding two factors to their

categorization of important factors that define a novelty detection problem: computational requirements and learning framework. These two factors make explicit current technological trends into the formulation of a novelty detection problem, such as distributed computing in big data projects and the need for online learning techniques.

2) This dissertation offers its own classification of modern approaches to novelty detection, which is based on a revision of two previous categorizations: (Chandola, Banerjee, & Kumar, 2009) and (Pimentel, Clifton, Clifton, & Tarassenko, 2014). The classification of modern methods for novelty detection proposed here is as follows: (1) *statistical*, (2) *classification-based*, (3) *clustering-based*, (4) *distance-based*, (5) *information theoretic*, (6) *subspace-based*, and (7) *angle-based*.

3) Implicit weighted variants of batch GP, online GP, and SOGP for the case of a Gaussian likelihood within a regression framework. Weight functions are employed as part of the likelihood terms, without enforcing strong constraints on the weight functions or adding a variable number of hyperparameters to the GP models. The mathematical derivation of our weighted GPs included expressions for hyperparameter estimation using MLE on marginal likelihoods and posterior marginal likelihoods.

4) Three data weighers that can be used for implicit weighted GP regression are proposed in chapter 4: *HeteroscedasticReg*, which allows learning from heteroscedastic data without the need for modeling noise variances (a property that is coined here as '*implicit heteroscedasticity*'); *RobustReg*, which allows obtaining robust GPs; and *HeteroscedasticRobustReg* for obtaining both robust and implicitly heteroscedastic GPs. As shown in section 4.4, the computational

complexities of our weighted GPs match the computational complexities of the corresponding standard GPs.

5) A preliminary experimental comparison of implicit weighted GP regression and standard GP regression in various simple simulated problems confirmed the effectiveness of our approach. Data with and without outliers were used, as well as heteroscedastic data.

6) A robust data weigher to be used in robust GP-based novelty detection is proposed in chapter 5. It was noted that this particular data weigher cannot be used on high-dimensional data because of its polynomial computational complexity. Subspace projection techniques can be leverage in order to calculate weights in the case of high-dimensional data.

7) A detailed experimental comparison of GP-based novelty detection using standard and weighted variants of batch GP and online GPs. The experiments were run on one simulated data set and three real-life multivariate data sets, showing that:

- Our weighted batch GP consistently and significantly outperformed standard batch GP when used for novelty detection.
- From the four membership scores that were used in (Kemmler M. , Rodner, Wacker, & Denzler, 2013), the Heuristic and Probability scores reported better performance across all data sets than the Mean and Negative Variance scores.
- Novelty detection using online GP and SOGP performed similar to batch GP-based novelty detection in many cases. This is particularly true for the case of online GP.

6.2 Conclusions

This dissertation has introduced implicit weighted GPs, which are defined as GPs that make use of weighted likelihoods that include weight functions called here *data weighers*. Data weighers have to take values in (0, 1] and the weights they assign to observations are proportional to how consistent those observations are with respect to the underlying model. We require that data weighers be used in a likelihood expression in a way that the corresponding weighted likelihood is a "genuine likelihood". The data weighers proposed in this dissertation are based on a quasi-robust potential to avoid numerical issues that might appear when weight functions are derived from robust potentials. We developed the mathematical expressions for implicit weighted GPs that employ an implicit weighted Gaussian likelihood.

The preliminary experiments from chapter 4 suggest that our approach allows the effective application of the MLE method to estimate GP hyperparameters for regression problems when the data contain outliers, removing a well-known limitation of the MLE method. Additionally, our weighted GPs outperformed standard GPs in most cases when data was contaminated with outliers, regardless of whether GP hyperparameters were estimated using MLE or appropriate values were used instead. Our experiments indicate that, in the case of heteroscedastic data and employing a heteroscedastic data weigher, the MLE method estimates the GP variance σ^2 near the value of the smallest noise variance in the data, as far as weights are inversely proportional to locally-estimated noise variances. Additionally, it was shown that the use of an implicit weighted Gaussian likelihood with a robust data weigher favors robust GP models while avoiding analytically intractable inferences and the associated approximation techniques. Interestingly,

our weighted SOGP tended to retain fewer data points that were not in accordance with the underlying model than standard SOGP, which indicates an increase in the quality of SOGP's set of basis vectors in that case.

Novelty detection based on our weighted batch GPs consistently and significantly outperformed novelty detection based on standard batch GPs whenever data was contaminated with outliers, for all the data sets used in our study in chapter 5. The same strong assessment cannot be made in the case of the various online GPs under comparison. However, there were many more cases of weighted online GPs outperforming the corresponding standard online GPs on contaminated data than the opposite. Chapter 5 also expanded the experimental work presented by the author in (Ramirez-Padron, Mederos, & Gonzalez, 2013), by comparing membership scores and the performance of batch GP and online GPs when used for novelty detection. The Heuristic and Probability scores performed similarly or better than the other two scores in most cases, and they never appeared as the worst score in any of the experiments. This leads to the conclusion that these scores should be preferred over Mean and Negative Variance in most cases. Regarding our comparison of online GPs and batch GPs, it was noted that online GP provided as good performance as batch GP in all cases when using standard GPs. However, in the case of our weighted GP variants, batch GP tended to perform better than online GP, which indicates that weighted batch GP was able to better leverage weights than online GP. Finally, the performance of SOGPs in general was inferior to the performance of batch GP, particularly when weighted GP variants were used. However, it is worth noting that SOGP with capacity $m = 30$ was able to perform as well as batch GP in 50% of the cases. This makes SOGP a compelling option for GP-based novelty detection on systems imposing strong memory constraints.

6.3 Future Research

In the experiments described in chapter 4, the parameters of the data weighers were easy to determine because of the low dimensionality of the data. The neighborhood size s was set to a value that greatly limited the need to rely on the default weight. The values of γ were small enough to accommodate the expected levels of model disagreement of outliers in the data. How to effectively estimate these parameters in a more general scenario is an open question worthy of further research. A related question is how sensitive our weighted GPs are to variations in the values of parameters of the data weighers. Additionally, we are interested in identifying real-life regression problems for which our approach would be particularly well-fitted. These questions apply to the case of GP-based novelty detection as well.

Chapter 4 offered preliminary experimental evidence of the benefits of using the weighted GPs proposed in this dissertation for solving regression problems where data is potentially contaminated with outliers. However, a more comprehensive comparison of weighted GPs and standard GPs for doing regression is needed, based on multi-dimensional data sets taken from real-life problems. Such study should implement a statistical comparison similar in nature to the experimental setup offered in chapter 5.

Our experiments in chapter 4 suggest that using the proposed weighted GPs for solving regression problems allows the use of MLE to estimate GP hyperparameters in the case of data containing outliers. A theoretical treatment of this experimental result and a more comprehensive experimental setup is needed in order to assess its validity in general terms. As a related topic, it is known that MLE cannot be used for estimating GP hyperparameters for GP-based novelty

detection problems. How to estimate hyperparameters in that case is still an open problem. It seems to the author that exploring and potentially improving the estimation procedure proposed in (Xiao, Wang, & Xu, 2014) is another research path worth taking.

The gap in performance between our weighted variants of batch GP and online GP shown in chapter 5 might be greatly reduced or even closed in cases where online GP can learn from a relatively large data set, to compensate for any incorrect weights calculated at the beginning of the learning process, when few observations are available. Further experimental research would be needed to determine whether this is actually the case. Additionally, it would be interesting to design data weighers that, contrary to the robust data weigher proposed in chapter 5, allow working with regions containing the target class that can have non-ellipsoidal shapes.

As noted in chapter 5, the SPM kernel employed for the target classes of the Caltech 101 data set is not invariant to translations and rotations. Improving this kernel by making it invariant to these transformations, as well as less sensitive to image backgrounds, should allow SOGP-based novelty detection to perform better under low capacity constraints.

As a final note, the author believes that the theoretical framework provided by Rademacher complexity (Bartlett & Mendelson, 2002), (Koltchinskii & Panchenko, 2000) would make possible a theoretical study of the learning capabilities and complexities of GP-based novelty detection when employing both standard and implicit weighted GPs.

APPENDIX A: COMPARISON OF STANDARD AND WEIGHTED GPs

Best and worst novelty detection scores. Points within Circles data set.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	NA
0	WeightedBatch	NA	NA
5	Batch	NA	Mean
5	WeightedBatch	NA	NA
10	Batch	NA	Mean
10	WeightedBatch	NA	NA
15	Batch	NA	Mean
15	WeightedBatch	NA	NA
20	Batch	NA	Mean
20	WeightedBatch	NA	NA

Best and worst novelty detection scores. Vertebral Column data set.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	Mean
0	WeightedBatch	NA	Mean
5	Batch	NA	Mean
5	WeightedBatch	NA	Mean
10	Batch	NA	Mean
10	WeightedBatch	NA	NA
15	Batch	NA	Mean
15	WeightedBatch	NA	NA
20	Batch	NA	Mean
20	WeightedBatch	NA	NA

Best and worst novelty detection scores, if any. Pima Indians Diabetes data set.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	Mean
0	WeightedBatch	NegVariance	Mean
5	Batch	NA	Mean
5	WeightedBatch	NA	Mean
10	Batch	NA	Mean
10	WeightedBatch	NegVariance	Mean
15	Batch	NegVariance	Mean
15	WeightedBatch	NegVariance	Mean
20	Batch	NegVariance	Mean
20	WeightedBatch	NegVariance	Mean

Best and worst novelty detection scores. Caltech 101 data set. Beaver target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	Mean
0	WeightedBatch	Probability	Mean
5	Batch	NA	Mean
5	WeightedBatch	Probability	Mean
10	Batch	NA	NA
10	WeightedBatch	Probability	Mean
15	Batch	NA	NA
15	WeightedBatch	NA	Mean
20	Batch	NA	NA
20	WeightedBatch	NA	Mean

Best and worst novelty detection scores. Caltech 101 data set. Cougar Body target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	Mean
0	WeightedBatch	NA	Mean
5	Batch	NA	NA
5	WeightedBatch	NA	Mean
10	Batch	NA	NA
10	WeightedBatch	NA	NA
15	Batch	NA	NegVariance
15	WeightedBatch	NA	NA
20	Batch	NA	NA
20	WeightedBatch	NA	NegVariance

Best and worst novelty detection scores. Caltech 101 data set. Crocodile target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	NegVariance
0	WeightedBatch	NA	NA
5	Batch	NA	NA
5	WeightedBatch	NA	NegVariance
10	Batch	NA	NA
10	WeightedBatch	NA	NegVariance
15	Batch	NA	NA
15	WeightedBatch	NA	NegVariance
20	Batch	Mean	NA
20	WeightedBatch	Mean	NegVariance

Best and worst novelty detection scores. Caltech 101 data set. Joshua Tree target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	NA
0	WeightedBatch	Mean	NegVariance
5	Batch	NA	NA
5	WeightedBatch	NA	NegVariance
10	Batch	NA	NA
10	WeightedBatch	NA	NegVariance
15	Batch	NA	NegVariance
15	WeightedBatch	NA	NegVariance
20	Batch	NA	NA
20	WeightedBatch	NA	NegVariance

Best and worst novelty detection scores. Caltech 101 data set. Minaret target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	NA	NegVariance
0	WeightedBatch	NA	NegVariance
5	Batch	NA	NA
5	WeightedBatch	NA	NegVariance
10	Batch	NA	Mean
10	WeightedBatch	NA	NegVariance
15	Batch	NA	Mean
15	WeightedBatch	NA	NA
20	Batch	NA	Mean
20	WeightedBatch	NA	NA

Best and worst novelty detection scores. Caltech 101 data set. Okapi target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	Heuristic	NegVariance
0	WeightedBatch	NA	NegVariance
5	Batch	NA	NegVariance
5	WeightedBatch	NA	NegVariance
10	Batch	NA	NegVariance
10	WeightedBatch	NA	NegVariance
15	Batch	NA	NegVariance
15	WeightedBatch	NA	NA
20	Batch	NA	NA
20	WeightedBatch	NA	NA

Best and worst novelty detection scores. Caltech 101 data set. Windsor Chair target class.

ContaminationLevel	GPType	Best Score	Worst Score
0	Batch	Heuristic	NegVariance
0	WeightedBatch	NA	NegVariance
5	Batch	Heuristic	NegVariance
5	WeightedBatch	NA	NegVariance
10	Batch	Heuristic	NA
10	WeightedBatch	NA	NegVariance
15	Batch	Heuristic	NA
15	WeightedBatch	NA	NegVariance
20	Batch	NA	NA
20	WeightedBatch	NA	NegVariance

APPENDIX B: COMPARISON OF BATCH AND ONLINE GPs

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Points within Circles data set.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	All (0.9995)			
0	Weighted	All (0.9995)			
5	Standard	BatchGP, OnlineGP (0.9930)	SOGP_m30 (0.9270)	SOGP_m10 (0.4157)	
5	Weighted	BatchGP, OnlineGP, SOGP_m30 (1.0000)	SOGP_m10 (0.9494)		
10	Standard	BatchGP, OnlineGP (0.9672)	SOGP_m30 (0.8091)	SOGP_m10 (0.2785)	
10	Weighted	BatchGP, OnlineGP, SOGP_m30 (1.0000)	SOGP_m10 (0.9318)		
15	Standard	BatchGP, OnlineGP (0.9252)	SOGP_m30 (0.7071)	SOGP_m10 (0.2283)	
15	Weighted	BatchGP, OnlineGP, SOGP_m30 (1.0000)	SOGP_m10 (0.9191)		
20	Standard	BatchGP, OnlineGP (0.8638)	SOGP_m30 (0.6060)	SOGP_m10 (0.2270)	
20	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.9999)	SOGP_m10 (0.9138)		

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Vertebral Column data set.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP (0.8780)	SOGP_m30 (0.8615)	SOGP_m10 (0.8339)	
0	Weighted	BatchGP, OnlineGP (0.8791)	SOGP_m30 (0.8675)	SOGP_m10 (0.8366)	
5	Standard	BatchGP, OnlineGP (0.8702)	SOGP_m30 (0.8454)	SOGP_m10 (0.8200)	
5	Weighted	BatchGP, OnlineGP (0.8720)	SOGP_m30 (0.8484)	SOGP_m10 (0.8153)	
10	Standard	BatchGP, OnlineGP (0.8624)	SOGP_m30 (0.8304)	SOGP_m10 (0.7482)	
10	Weighted	BatchGP, OnlineGP (0.8640)	SOGP_m30 (0.8218)	SOGP_m10 (0.7456)	
15	Standard	BatchGP, OnlineGP (0.8554)	SOGP_m30 (0.8129)	SOGP_m10 (0.6781)	
15	Weighted	BatchGP, OnlineGP (0.8568)	SOGP_m30 (0.8124)	SOGP_m10 (0.6779)	
20	Standard	BatchGP, OnlineGP (0.8482)	SOGP_m30 (0.8014)	SOGP_m10 (0.6408)	
20	Weighted	BatchGP, OnlineGP (0.8519)	SOGP_m30 (0.8019)	SOGP_m10 (0.6503)	

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Pima Indians Diabetes data set.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP (0.7379)	SOGP_m30 (0.6996)	SOGP_m10 (0.6838)	
0	Weighted	BatchGP, OnlineGP (0.7459)	SOGP_m30 (0.7229)	SOGP_m10 (0.6836)	
5	Standard	BatchGP, OnlineGP (0.7316)	SOGPm_10, SOGP_m30		
5	Weighted	BatchGP, OnlineGP (0.7396)	SOGP_m30 (0.7191)	SOGP_m10 (0.6918)	
10	Standard	BatchGP, OnlineGP (0.7268)	SOGPm_10, SOGP_m30		
10	Weighted	BatchGP, OnlineGP (0.7346)	SOGP_m30 (0.7129)	SOGP_m10 (0.6894)	
15	Standard	BatchGP, OnlineGP (0.7220)	SOGPm_10, SOGP_m30		
15	Weighted	BatchGP, OnlineGP (0.7293)	SOGP_m30 (0.7041)	SOGP_m10 (0.6866)	
20	Standard	BatchGP, OnlineGP (0.7178)	SOGPm_10, SOGP_m30		
20	Weighted	BatchGP, OnlineGP (0.7228)	SOGP_m30 (0.7015)	SOGP_m10 (0.6823)	

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Beaver target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7802)	SOGP_m10 (0.7514)		
0	Weighted	OnlineGP, SOGP_m30 (0.7848)	BatchGP (0.7773)	SOGP_m10 (0.7518)	
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7603)	SOGP_m10 (0.7316)		
5	Weighted	OnlineGP, SOGP_m30 (0.7771)	BatchGP (0.7682)	SOGP_m10 (0.7436)	
10	Standard	All (0.7351)			
10	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7656)	SOGP_m10 (0.7287)		
15	Standard	All (0.7136)			
15	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7560)	SOGP_m10 (0.7152)		
20	Standard	All (0.6906)			
20	Weighted	SOGP_m30 (0.7618)	BatchGP, OnlineGP (0.7464)	SOGP_m10 (0.7116)	

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Cougar Body target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7633)	SOGP_m10 (0.7174)		
0	Weighted	BatchGP (0.7658)	OnlineGP, SOGP_m30 (0.7515)	SOGP_m10 (0.7105)	
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7347)	SOGP_m10 (0.6971)		
5	Weighted	BatchGP (0.7562)	OnlineGP, SOGP_m30 (0.7404)	SOGP_m10 (0.6930)	
10	Standard	All (0.6956)			
10	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7309)	SOGP_m10 (0.6829)		
15	Standard	All (0.6771)			
15	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7217)	SOGP_m10 (0.6650)		
20	Standard	All (0.6590)			
20	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7118)	SOGP_m10 (0.6530)		

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Crocodile target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7339)	SOGP_m10 (0.7147)		
0	Weighted	BatchGP (0.7497)	OnlineGP, SOGP_m30 (0.7405)	SOGP_m10 (0.7198)	
5	Standard	All (0.6617)			
5	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.7068)	SOGP_m10 (0.6672)		
10	Standard	All (0.6046)			
10	Weighted	BatchGP, OnlineGP (0.6802)	SOGP_m30, SOGP_m10 (0.6342)		
15	Standard	All (0.5446)			
15	Weighted	BatchGP (0.6694)	OnlineGP, SOGP_m30 (0.6222)	SOGP_m10 (0.5771)	
20	Standard	All (0.4876)			
20	Weighted	All (0.5784)			

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Joshua Tree target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP (0.8966)	SOGP_m30 (0.8932)	SOGP_m10 (0.8534)	
0	Weighted	BatchGP, OnlineGP (0.8796)	SOGP_m30 (0.8666)	SOGP_m10 (0.8490)	
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.8532)	SOGP_m10 (0.8050)		
5	Weighted	BatchGP, OnlineGP (0.8554)	SOGP_m30 (0.8330)	SOGP_m10 (0.8044)	
10	Standard	BatchGP, OnlineGP, SOGP_m30 (0.8137)	SOGP_m10 (0.7551)		
10	Weighted	BatchGP (0.8390)	OnlineGP (0.8200)	SOGP_m30 (0.7972)	SOGP_m10 (0.7661)
15	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7742)	SOGP_m10 (0.7099)		
15	Weighted	BatchGP (0.8193)	OnlineGP (0.7885)	SOGP_m30 (0.7667)	SOGP_m10 (0.7439)
20	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7319)	SOGP_m10 (0.6640)		
20	Weighted	BatchGP (0.7954)	OnlineGP (0.7680)	SOGP_m30 (0.7438)	SOGP_m10 (0.7223)

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Minaret target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	SOGP_m30 (0.9982)	BatchGP, OnlineGP (0.9978)	SOGP_m10 (0.9971)	
0	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.9968)	SOGP_m10 (0.9944)		
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.9692)	SOGP_m10 (0.9348)		
5	Weighted	BatchGP (0.9827)	OnlineGP, SOGP_m30 (0.9562)	SOGP_m10 (0.8922)	
10	Standard	BatchGP, OnlineGP (0.9394)	SOGP_m30 (0.9109)	SOGP_m10 (0.8825)	
10	Weighted	BatchGP (0.9613)	OnlineGP (0.9260)	SOGP_m30 (0.8936)	SOGP_m10 (0.8246)
15	Standard	BatchGP, OnlineGP (0.8962)	SOGP_m30 (0.8484)	SOGP_m10 (0.8151)	
15	Weighted	BatchGP (0.9309)	OnlineGP (0.8639)	SOGP_m30 (0.8229)	SOGP_m10 (0.7411)
20	Standard	BatchGP, OnlineGP (0.8624)	SOGP_m30 (0.8017)	SOGP_m10 (0.7492)	
20	Weighted	BatchGP (0.8995)	OnlineGP (0.8052)	SOGP_m30 (0.7554)	SOGP_m10 (0.6934)

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Okapi target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP, SOGP_m30 (0.9416)	SOGP_m10 (0.9344)		
0	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.9350)	SOGP_m10 (0.9160)		
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.9234)	SOGP_m10 (0.8965)		
5	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.9154)	SOGP_m10 (0.8598)		
10	Standard	BatchGP, OnlineGP, SOGP_m30 (0.9039)	SOGP_m10 (0.8688)		
10	Weighted	BatchGP, OnlineGP, SOGP_m30 (0.9017)	SOGP_m10 (0.8232)		
15	Standard	BatchGP, OnlineGP, SOGP_m30 (0.8887)	SOGP_m10 (0.8387)		
15	Weighted	BatchGP (0.9072)	OnlineGP, SOGP_m30 (0.8718)	SOGP_m10 (0.7917)	
20	Standard	BatchGP, OnlineGP, SOGP_m30 (0.8638)	SOGP_m10 (0.8084)		
20	Weighted	BatchGP (0.8945)	OnlineGP, SOGP_m30 (0.8594)	SOGP_m10 (0.7594)	

Counting of rank allocations for each particular GP type, for different contamination levels, including average performance of each rank. Caltech 101 data set. Windsor Chair target class.

ContaminationLevel	Standard/Weighted	Rank 1 (Rank 1 Avg AUC)	Rank 2 (Rank 2 Avg AUC)	Rank 3 (Rank 3 Avg AUC)	Rank 4 (Rank 4 Avg AUC)
0	Standard	BatchGP, OnlineGP, SOGP_m30 (0.9553)	SOGP_m10 (0.9124)		
0	Weighted	BatchGP (0.9602)	OnlineGP, SOGP_m30 (0.9484)	SOGP_m10 (0.9095)	
5	Standard	BatchGP, OnlineGP, SOGP_m30 (0.8929)	SOGP_m10 (0.8021)		
5	Weighted	BatchGP (0.9179)	OnlineGP, SOGP_m30 (0.8632)	SOGP_m10 (0.7396)	
10	Standard	BatchGP, OnlineGP (0.8145)	SOGP_m30 (0.7817)	SOGP_m10 (0.6629)	
10	Weighted	BatchGP (0.8538)	OnlineGP, SOGP_m30 (0.7401)	SOGP_m10 (0.6198)	
15	Standard	BatchGP, OnlineGP, SOGP_m30 (0.7503)	SOGP_m10 (0.6105)		
15	Weighted	BatchGP (0.8025)	OnlineGP (0.7017)	SOGP_m30 (0.6567)	SOGP_m10 (0.5140)
20	Standard	BatchGP, OnlineGP, SOGP_m30 (0.6906)	SOGP_m10 (0.5402)		
20	Weighted	BatchGP (0.7473)	OnlineGP, SOGP_m30 (0.6006)	SOGP_m10 (0.5014)	

REFERENCES

Abe, S. (2010). *Support Vector Machines for Pattern Classification.* Springer-Verlag.

Adams, R., Murray, I., & MacKay, D. (2009). The Gaussian Process Density Sampler. *Advances in Neural Information Processing Systems (NIPS 2009), 21,* pp. 9-16.

Afgani, M., Sinanovic, S., & Haas, H. (2010). The Information Theoretic Approach to Signal Anomaly Detection for Cognitive Radio. *International Journal of Digital Multimedia Broadcasting, 2010,* 18.

Agarwal, D. (2007). Detecting anomalies in cross-classified streams: a Bayesian approach. *Knowledge and Information Systems, 11*(1), 29-44.

Aggarwal, C., & Yu, P. (2001). Outlier detection for high dimensional data. *Proceedings of the ACM SIGMOD International Conference on Data Mining* (pp. 37-46). ACM Press.

Aggarwal, C., & Yu, P. (2008). Outlier detection with uncertain data. *Proceedings of the SIAM International Conference on Data Mining,* (pp. 483-493).

Aggarwal, C., & Yu, P. (2008). *Privacy-preserving data mining: models and algorithms.* Springer-Verlag.

Agostinelli, C., & Greco, L. (2013). A Weighted Strategy to Handle Likelihood Uncertainty in Bayesian Inference. *Computational Statistics, 28,* 319-339.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Proceedings of the Eleventh International Conference on Data Engineering* (pp. 3-14). IEEE Computer Society.

Ahmed, A. H., & Ashour, W. (2011). An Initialization Method for the K-means Algorithm using RNN and Coupling Degree. *International Journal of Computer Applications, 25*(1), 1-6.

Airy, G. B. (1856). Letter from Professor Airy, Astronomer Royal, to the Editor. *Astronomical Journal, 4*, 137-138.

An, W., Liang, M., & Liu, H. (2014). An improved one-class support vector machine classifier for outlier detection. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 1-9.

Ando, S. (2007). Clustering needles in a haystack: An information theoretic analysis of minority and outlier detection. *Seventh IEEE International Conference on Data Mining. ICDM 2007* (pp. 13-22). IEEE.

Angiulli, F., Basta, S., & Pizzuti, C. (2006). Distance-Based Detection and Prediction of Outliers. *IEEE Transactions on Knowledge and Data Engineering, 18*, 145-160.

Anscombe, F. J., & Guttman, I. (1960). Rejection of outliers. *Technometrics, 2*(2), 123-147.

Ariu, D., Giacinto, G., & Perdisci, R. (2007). Sensing Attacks in Computers Networks with Hidden Markov Models. *Machine Learning and Data Mining in Pattern Recognition. Lecture Notes in Computer Science. 4571*, pp. 449-463. Springer.

Arning, A., Agrawal, R., & Raghavan, P. (1996). A linear method for deviation detection in large databases. In E. Simoudis, J. Han, & U. Fayyad (Ed.), *Proceedings KDD 1996* (pp. 164-169). Portland, OR: AAAI Press.

Aronszajn, N. (1950). Theory of Reproducing Kernels. *Transactions of the American Mathematical Society, 68*(3), 337-404.

Ash, R. B. (1990). *Information Theory.* Dover Publications.

Augusteijn, M. F., & Folkert, B. A. (2002). Neural network classification and novelty detection. *International Journal of Remote Sensing, 23*(14), 2891-2902.

Barbara, D., Couto, J., Jajodia, S., & Wu, N. (2001). Detecting novel network intrusions using bayes estimators. *Proceedings of the First SIAM International Conference on Data Mining.* SIAM.

Barnett, V., & Lewis, T. (1994). *Outliers in Statistical Data* (3 ed.). John Wiley.

Barreto, G., & Aguayo, L. (2009). Time series clustering for anomaly detection using competitive neural networks. *Advances in Self-Organizing Maps,Lecture Notes in Computer Science. 5629*, pp. 28-36. Springer.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research, 3*, 463–482.

Basu, S., Bilenko, M., & Mooney, R. (2004). A probabilistic framework for semi-supervised clustering. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 59-68). ACM.

Bay, S., & Schwabacher, M. (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 29-38). ACM Press.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS. Communicated by Mr. Price, in a letter to John Canton, AMFRS. *Philosophical Transactions, 53*, 370 - 418.

Bengio, Y., Larochelle, H., & Vincent, P. (2005). Non-local Manifold Parzen Windows. *Advances in Neural Information Processing Systems*, (pp. 115-122).

Bentley , J. L. (1980). Multidimensional Divide and Conquer. *Communications of the ACM, 23*(4), 214-229.

Bentley, J. (1980). Multidimensional Divide-and-Conquer. *Communications of the ACM, 23*(4), 214-229.

Bernholt, T., & Fischer, P. (2004). The complexity of computing the MCD-estimator. *Theoretical Computer Science, 326*, 383-398.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

Bishop, C. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Blanchard, G., Lee, G., & Scott, C. (2010). Semi-Supervised Novelty Detection. *Journal of Machine Learning Research, 11*, 2973 - 3009.

Blei, D., Jordan, M., & Ng, A. (2003). Hierarchical Bayesian Models for Applications in Information Retrieval. In J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, et al. (Eds.), *Bayesian Statistics* (Vol. 7, pp. 25-43). Oxford University Press.

Bliss, C. I., Cochran, W. G., & Tukey, J. W. (1956). A rejection criterion based upon the range. *Biometrika, 43*, 418-422.

Bolstad, W. (2007). *Introduction to Bayesian Statistics* (2 ed.). John Wiley & Sons, Inc.

Bosch, A., Zisserman, A., & Munoz, X. (2007). Representing shape with a spatial pyramid kernel. *Proceedings of the 6th ACM International Conference on Image and Video Retrieval* (pp. 401-408). ACM.

Breunig, M., Kriegel, H., Ng, R., & Sander, J. (2000). LOF: Identifying density-based local outliers. *International Conference on Management of Data*, (pp. 1-12).

Bronstein, A., Das, J., Duro, M., Friedrich, R., Kleyner, G., Mueller, M., et al. (2001). Self-aware services: using Bayesian networks for detecting anomalies in Internet-based services. *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings* (pp. 623-638). IEEE.

Byers, S. D., & Raftery, A. E. (1998). Nearest neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association, 93*(442), 577-584.

Carroll, R. J., & Ruppert, D. (1982). A Comparison Between Maximum Likelihood and Generalized Least Squares in a Heteroscedastic Linear Model. *Journal of the American Statistical Association, 77*(380), 878-882.

Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In T. K. Leen, T. G. Dietterich, & V. Tresp, *Advances in Neural Information Processing Systems* (Vol. 13, pp. 409–415). MIT Press.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys, 41*, 15:1–15:58.

Chauvenet, W. (1868). *A treatise on the method of least squares.*

Chawla, S., & Gionis, A. (2013). k-means–: A unified approach to clustering and outlier detection. *2013 SIAM International Conference on Data Mining* (pp. 189-197). SIAM.

Chen, D., Chao, X., Hu, B., & Su, Q. (2005). Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra. *Analytical Sciences, 21*(2), 161-166.

Chen, S., Gunn, S., & Harris, C. (2001). The Relevance Vector Machine Technique for Channel Equalization Application. *IEEE Transactions on Neural Networks, 12*(6), 1529-1532.

Chiu, T., Fang, D., Chen, J., Wang, Y., & Jeris, C. (2001). A robust and scalable clustering algorithm for mixed type attributes in large database environment. *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 263-268). ACM.

Cleary, J. (1979). Analysis of an algorithm for finding nearest neighbors in euclidean space. *ACM Transactions on Mathematical Software, 5*(2), 183-192.

Clifton, D., Bannister, P., & Tarassenko, L. (2007). A framework for novelty detection in jet engine vibration data. *Key Engineering Materials, 347*, 305-310.

Clifton, D., Hugueny, S., & Tarassenko, L. (2011). Novelty Detection with Multivariate Extreme Value Statistics. *J Sign Process Syst, 65*(3), 371-389.

Clifton, L., Clifton, D., Watkinson, P., & Tarassenko, L. (2011). Identification of patient deterioration in vital-sign data using one-class support vector machines. *Federated Conference on Computer Science and Information Systems (FedCSIS)* (pp. 125-131). IEEE.

Clifton, L., Yin, H., Clifton, Y., & Zhang, Y. (2007). Combined Support Vector Novelty Detection for Multi-channel Combustion Data. *IEEE International Conference on Networking, Sensing and Control* (pp. 495-500). IEEE.

Cortes, C., & Vapnik, V. (1995). Support Vector Networks. *Machine Learning*(20), 273-297.

Cox, R. (1946). Probability, Frequency and Reasonable Expectation. *American Journal of Physics, 14*(1), 1 - 13.

Csató, L. (2002). *Gaussian Processes – Iterative Sparse Approximations. PhD thesis.* Birmingham, UK: Aston University.

Csató, L., & Opper, M. (2002). Sparse On-line Gaussian Processes. *Neural Computation, 14*(3), 641 – 668.

Dai, Z., Huang, L., Zhu, Y., & Yang, W. (2010). Privacy Preserving Density-Based Outlier Detection. *2010 International Conference on Communications and Mobile Computing (CMC)* (pp. 80-85). IEEE.

Daniels, H. E. (1961). The Asymptotic Efficiency of a Maximum Likelihood Estimator. *Fourth Berkeley Symposium on Mathematical Statistics and Probability, vol 1: Contributions to the Theory of Statistics* (pp. 151-163). Berkeley, CA: University of California Press.

Das, K., & Schneider, J. (2007). Detecting anomalous records in categorical datasets. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 220-229). ACM.

Deb, K. (2005). Multi-Objective Optimization. In E. Burke, & G. Kendall (Eds.), *Search Methodologies* (pp. 273-316). Springer.

Desforges, M., Jacob, P., & Cooper, J. (1998). Applications of probability density estimation to the detection of abnormal conditions in engineering. *Proceedings of the Institute of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 212,* pp. 687-703.

Diehl, C., & Hampshire, J. (2002). Real-time object classification and novelty detection for collaborative video surveillance. *Proceedings of the 2002 International Joint Conference on Neural Networks, 2002. IJCNN '02*, (pp. 2620 - 2625).

Ding, X., Li, Y., Belatreche, A., & Maguire, L. P. (2014). An experimental evaluation of novelty detection methods. *Neurocomputing, 135*, 313-327.

Dutta, H., Giannella, C., Borne, K., & Kargupta, H. (2007). Distributed top-k outlier detection from astronomy catalogs using the DEMAC system. *Proceedings of 7th SIAM International Conference on Data Mining.* SIAM.

Ertöz, L., Steinbach, M., & Kumar, V. (2003). Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval*, 83-104.

Eskin, E. (2000). Anomaly Detection over Noisy Data using Learned Probability Distributions. *Proceedings of the International Conference on Machine Learning* (pp. 255-262). Morgan Kaufmann.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. *Proceedings of the Conference on Applications of Data Mining in Computer Security* (pp. 78-100). Kluwer Academics.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (pp. 226-231). AAAI Press.

Fairley, J., Georgoulas, G., Stylios, C., & Rye, D. (2010). A Hybrid Approach for Artifact Detection in EEG Data. *Artificial Neural Networks – ICANN 2010. Lecture Notes in Computer Science. 6352/2010*, pp. 436-441. Springer.

Fan, W., Miller, M., Stolfo, S. J., Lee, W., & Chan, P. K. (2001). Using artificial anomalies to detect unknown and known network intrusions. *Proceedings of the IEEE International Conference on Data Mining* (pp. 123–130). IEEE Computer Society.

Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Fawcett, T., & Provost, F. (1999). Activity Monitoring: Noticing Interesting Changes in Behavior. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 53-62).

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. *IEEE CVPR Workshop on Generative-Model Based Vision.* IEEE.

Filev, D., & Tseng, F. (2006). Real time novelty detection modeling for machine health prognostics. *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)* (pp. 529-534). IEEE.

Filippone, M., Masulli, F., & Rovetta, S. (2010). Applying the Possibilistic c-Means Algorithm in Kernel-Induced Spaces. *IEEE Transactions on Fuzzy Systems, 18*(3), 572-584.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, (pp. 625-632).

Galeano, P., Peña, D., & Tsay, R. (2006). Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association, 101*(474), 654--669.

Gamerman, D., & Lopes, H. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference* (2 ed.). Chapman & Hall/CRC.

García-Rodríguez, J., Angelopoulou, A., García-Chamiz, J., Orts-Escolano, S., & Morell-Giménez, V. (2012). Autonomous growing neural gas for applications with time constraint: optimal parameter estimation. *Neural Networks, 32*, 196-208.

Gardner, A., Krieger, A., Vachtsevanos, G., & Litt, B. (2006). One-Class Novelty Detection for Seizure Analysis from Intracranial EEG. *The Journal of Machine Learning Research, 7*, 1025-1044.

Genton, M. G. (2001). Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research, 2*, 299–312.

Gibbs, M. (1997). *Bayesian Gaussian Processes for Regression and Classification. PhD thesis.* University of Cambridge.

Gibbs, M., & MacKay, D. (2000). Variational Gaussian Process Classifiers. *IEEE Transactions on Neural Networks, 11*, 1458 - 1464.

Gilks, W., Richardson, S., & Spiegelhalter, D. (Eds.). (1995). *Markov Chain Monte Carlo in Practice.* Chapman & Hall/CRC.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications, 13*(4), 215–223.

Goldberg, P. W., Williams, C. I., & Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. *Advances in Neural Information Processing Systems* (pp. 493-499). MIT Press.

Gould, B. A. (1855). On Peirce's Criterion for the Rejection of Doubtful Observations, with tables for facilitating its application. *Astronomical Journal, 4*, 81.

Grauman, K., & Darrell, T. (2005). Pyramid match kernels: Discriminative Classification with Sets of Image Features. *Proceedings of the International Conference on Computer Vision.*

Greco, L., Racugno, W., & Ventura, L. (2008). Robust Likelihood Functions in Bayesian Analysis. *Journal of Statistical Planning and Inference, 138*(5), 1258-1270.

Grubbs, F. (1969). Procedures for detecting outlying observations in samples. *Technometrics, 11*(1), 1-21.

Grubbs, F. E. (1950). Sample Criteria for Testing Outlying Observations. *The Annals of Mathematical Statistics, 21*(1), 27-58.

Gu, G., Fogla, P., Dagon, D., Lee, W., & Škorić, B. (2006). Measuring intrusion detection capability: An information-theoretic approach. *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security* (pp. 90-101). ACM.

Guazzelli, A., Zeller, M., Lin, W. C., & Williams, G. (2009). PMML: An Open Standard for Sharing Models. *The R Journal, 1*(1).

Guha, S., Rastogi, R., & Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information Systems, 25*(5), 345-366.

Günter, S., Schraudolph, N., & Vishwanathan, S. (2007). Fast iterative kernel principal component analysis. *Journal of Machine Learning Research, 8*, 1893--1918.

Guo, S. M., Shen, L. C., & Tsai, J. S. (2009). A boundary method for outlier detection based on support vector domain description. *Pattern Recognition, 42*, 77-83.

Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering, 26*(9).

Hadi, A. S., Imon, A. M., & Werner, M. (2009). Detection of outliers. *Wiley Interdisciplinary Reviews: Computational Statistics, 1*(1), 57—70.

Haggett, S. J., Chu, D. F., & Marshall, I. W. (2008). Evolving a dynamic predictive coding mechanism for novelty detection. *Knowledge-Based Systems, 21*, 217-224.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., & Stahel, W. A. (1986). *Robust Statistics. The Approach Based on Influence Functions.* New York: John Wiley and Sons.

Hardoon, D. R., & Manevitz, L. M. (2005). fMRI Analysis via One-class Machine Learning Techniques. *International Joint Conference on Artificial Intelligence, IJCAI'05* (pp. 1604-1605). SanFrancisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hawkins, D. M. (1980). *Identification of Outliers.* Chapman and Hall.

Hawkins, S., He, H., Williams, G. J., & Baxter, R. A. (2002). Outlier detection using replicator neural networks. *Data Warehousing and Knowledge Discovery. Lecture Notes in Computer Science. 2454/2002*, pp. 113-123. Springer-Verlag.

He, Z., Deng, S., Xu, X., & Huang, J. (2006). A Fast Greedy Algorithm for Outlier Mining. *Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science. 3918*, pp. 567-576. Springer.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters, 24*, 1641-1650.

He, Z., Xu, X., Huang, J., & Deng, S. (2004). A Frequent Pattern Discovery Method for Outlier Detection. *Advances in Web-Age Information Management. Lecture Notes in Computer Science. 3129*, pp. 726-732. Springer.

Hellman, M. (1970). The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics, 6*(3), 179-185.

Hinneburg, A., Aggarwal, C., & Keim, D. (2000). What Is the Nearest Neighbor in High Dimensional Spaces? *Proceedings of the 26th International Conference on Very Large Data Bases* (p. 515). Morgan Kaufmann Publishers Inc.

Hoares, S., Asbridge, D., & Beatty, P. (2002). On-line novelty detection for artefact identification in automatic anaesthesia record keeping. *Med. Eng. Phys., 24*(10), 673-681.

Hoffmann, H. (2007). Kernel PCA for novelty detection. *Pattern Recognition, 40*(3), 863-874.

Horn, P., Feng, L., Li, Y., & Pesce, A. (2001). Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry, 47*(12), 2137-2145.

Hu, W., Liao, Y., & Vemuri, V. (2003). Robust Anomaly Detection using Support Vector Machines. *International Conference on Machine Learning*, (pp. 282-289).

Huber, P., & Ronchetti, E. M. (2009). *Robust Statistics* (2 ed.). Wiley.

Hubert, M., & Debruyne, M. (2010). Minimum covariance determinant. *Wiley Interdisciplinary Reviews: Computational Statistics, 2*(1), 36-43.

Hubert, M., Rousseeuw, P. J., & Vanden Branden, K. (2005). ROBPCA: A New Approach to Robust Principal Component Analysis. *Technometrics, 47*, 64-79.

Ilonen, J., Paalanen, P., & Kamarainen, J. (2006). Gaussian mixture pdf in one-class classification: computing and utilizing confidence values. *Proceedings of the 18th International Conference on Pattern Recognition (ICPR). 2*, pp. 577-580. IEEE.

Irwin, J. O. (1925a). On a Criterion for the Rejection of Outlying Observations. *Biometrika, 17*(3), 238-250.

Irwin, J. O. (1925b). The Further Theory of Francis Galton's Individual Difference Problem. *Biometrika, 17*, 100-128.

Janakiram, D., Adi Mallikarjuna Reddy, V., & Phani Kumar, A. (2006). Outlier Detection in Wireless Sensor Networks using Bayesian Belief Networks. *First International Conference on Communication System Software and Middleware, 2006. Comsware 2006* (pp. 1-6). IEEE.

Jeffreys, H. (1932). An alternative to the rejection of observations. *Proceedings of the Royal Society of London, 137*(831), 78-87.

Joachims, T. (2006). Training linear SVMs in linear time. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 217-226). ACM.

John, G. (1995). Robust Decision Trees: Removing Outliers from Databases. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (pp. 174-179). Menlo Park, CA: AAAI Press.

Joliffe, I. (2002). *Principal Component Analysis* (2 ed.). Springer.

Joshi, M., Agarwal, R., & Kumar, V. (2001). Mining needle in a haystack: classifying rare classes via two-phase rule induction. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 91-102). ACM Press.

Joshi, M., Agarwal, R., & Kumar, V. (2002). Predicting Rare Classes: Comparing Two-Phase Rule Induction to Cost-Sensitive Boosting. In T. Elomaa, H. Mannila, & H. Toivonen (Ed.), *Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science. 2431*, pp. 145-167. Springer Berlin/Heidelberg.

Jylänki, P., Vanhatalo, J., & Vehtari, A. (2011). Robust Gaussian Process Regression with a Student-t Likelihood. *The Journal of Machine Learning Research, 12*, 3227-3257.

Kapoor, A., Grauman, K., Urtasun, R., & Darrell, T. (2010). Gaussian Processes for Object Categorization. *International Journal of Computer Vision, 88*(2), 169-188.

Katayama, N., & Satoh, S. (1997). The SR-tree: An index structure for high-dimensional nearest neighbor queries. *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (pp. 369-380). ACM.

Keerthi, S., Shevade, S., Bhattacharyya, C., & Murthy, K. (2001). Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation, 13*, 637-649.

Kemmler, M., Denzler, J., Rösch, P., & Popp, J. (2010). Classification of Microorganisms via Raman Spectroscopy Using Gaussian Processes. *Pattern Recognition. Lecture Notes in Computer Science* (pp. 81-90). Springer Berlin / Heidelberg.

Kemmler, M., Rodner, E., & Denzler, J. (2010). One-class Classification with Gaussian Processes. *Proceedings of the Asian Conference on Computer Vision. Lecture Notes in Computer Science. 6493*, pp. 489 - 500. Springer.

Kemmler, M., Rodner, E., Wacker, E. S., & Denzler, J. (2013). One-class classification with Gaussian processes. *Pattern Recognition, 46*, 3507–3518.

Keogh, E., Lonardi, S., & Ratanamahatana, C. (2004). Towards parameter-free data mining. *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 206-215). ACM.

Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most Likely Heteroscedastic Gaussian Process Regression. *24th International Conference on Machine Learning*, (pp. 393-400).

Khan, N. M., Ksantini, R., Ahmad, I. S., & Guan, L. (2014). Covariance-guided One-Class Support Vector Machine. *Pattern Recognition, 47*, 2165-2177.

Khan, S. S., & Ahmad, A. (2004). Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters, 25*(11), 1293-1302.

Kim, H. C., & Lee, J. (2006). Pseudo-density Estimation for Clustering with Gaussian Processes. *Advances in Neural Networks* (pp. 1238-1243). Springer.

Kit, D., Sullivan, B., & Ballard, D. (2011). Novelty detection using growing neural gas for visuo-spatial memory. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1194-1200). IEEE.

Kivinen, J., Smola, A. J., & Williamson, R. C. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing, 52*(8), 2165-2176.

Knorr, E. M., & Ng, R. T. (1997). A unified approach for mining outliers. *CASCON '97, Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative Research* (pp. 236-248). IBM Press.

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB Journal, 8*(3), 237-253.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Fourteenth International Joint Conference on Artificial Intelligence*, (pp. 1137–1143).

Kolev, D., Suvorov, M., Morozov, E., Markarian, G., & Angelov, P. (2015). Incremental Anomaly Identification in Flight Data Analysis by Adapted One-Class SVM Method. In P. Koprinkova-Hristova, V. Mladenov, & N. K. Kasabov (Eds.), *Artificial Neural Networks. Methods and Applications in Bio-/Neuroinformatics* (Vols. Springer Series in Bio-/Neuroinformatics. Vol 4, pp. 373-391). Springer International Publishing.

Koltchinskii, V., & Panchenko, D. (2000). Rademacher processes and bounding the risk of function learning. *High Dimensional Probability, 2*, 443–459.

Kou, Y., Lu, C., & Chen, D. (2006). Spatial weighted outlier detection. *Proceedings of SIAM Conference on Data Mining.*

Kou, Y., Lu, C., & Dos Santos, R. (2007). Spatial outlier detection: a graph-based approach. *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007* (pp. 281-288). IEEE.

Koufakou, A., & Georgiopoulos, M. (2010). A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes. *Data Mining and Knowledge Discovery, 20*(2), 259-289.

Kriegel, H. P., Schubert, M., & Zimek, A. (2008). Angle-based Outlier Detection. *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD).* Las Vegas, NV.: ACM.

Krishna, M. V., Bodesheim, P., & Denzler, J. (2013). Video Segmentation by Event Detection: A Novel One-Class Classification Approach. *4th International Workshop on Image Mining. Theory and Applications (IMTA-4).*

Krishna, M. V., Bodesheim, P., Körner, M., & Denzler, J. (2014). Temporal Video Segmentation by Event Detection: A Novelty Detection Approach. *Pattern Recognition and Image Analysis, 24*(2), 243-255.

Kruegel, C., & Vigna, G. (2003). Anomaly detection of web-based attacks. *Proceedings of the 10th ACM Conference on Computer and Communications Security* (pp. 251-261). ACM.

Kwok, J., Tsang, I., & Zurada, J. (2007). A class of single-class minimax probability machines for novelty detection. *IEEE Transactions on Neural Networks, 18*(3), 778-785.

Lakhina, A., Crovella, M., & Diot, C. (2005). Mining anomalies using traffic feature distributions. *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (pp. 217-228). ACM.

Lanckriet, G., El Ghaoui, L., & Jordan, M. (2003). Robust novelty detection with single-class MPM. *Advances in Neural Information Processing Systems. NIPS 2003* (pp. 905-912). MIT Press.

Lanckriet, G., Ghaoui, L., Bhattacharyya, C., & Jordan, M. (2002). A robust minimax approach to classification. *The Journal of Machine Learning Research, 3*, 555-582.

Lane, D. M. (2010). Tukey's Honestly Significant Difference (HSD). In N. J. Elkind (Ed.), *Encyclopedia of Research Methods.* Sage Publications.

Laplace, P. (1812). *Théorie Analytique des Probabilities.* Paris: Courcier.

Laskov, P., Gehl, C., Krüger, S., & Müller, K.-R. (2006). Incremental Support Vector Learning: Analysis, Implementation and Applications. *Journal of Machine Learning, 7*, 1909-1936.

Latecki, L. J., Lazarevic, A., & Pokrajac, D. (2007). Outlier Detection with Kernel Density Functions. *Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition. Lecture Notes in Artificial Intelligence. 4571*, pp. 61-75. Springer Verlag.

Laurikkala, J., Juhola, M., & Kentala, E. (2000). Informal identification of outliers in medical data. *The Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, (pp. 20-24).

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2169-2178). IEEE.

Le, T., Tran, D., Ma, W., & Sharma, D. (2010). An optimal sphere and two large margins approach for novelty detection. *International Joint Conference on Neural Networks (IJCNN)* (pp. 1-6). IEEE.

Le, T., Tran, D., Ma, W., & Sharma, D. (2011). Multiple distribution data description learning algorithm for novelty detection. *15th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2011. Lecture Notes in Computer Science. 6635*, pp. 246-257. Shenzhen; China: Springer.

Lee, H., & Roberts, S. (2008). On-line novelty detection using the Kalman filter and extreme value theory. *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, (pp. 1-4).

Lee, W., & Xiang, D. (2001). Information-theoretic measures for anomaly detection. *Proceedings. 2001 IEEE Symposium on Security and Privacy* (pp. 130-143). IEEE.

Lei, D., Zhu, Q., Chen, J., Lin, H., & Yang, P. (2012). Automatic K-Means Clustering Algorithm for Outlier Detection. *International Conference on Information Engineering and Applications. Lecture Notes in Electrical Engineering. 154*, pp. 363-372. Springer.

Li, C., Georgiopoulos, M., & Anagnostopoulos, G. C. (2011). Kernel Principal Subspace Mahalanobis Distances for Outlier Detection. *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp. 2528-2535). IEEE.

Lichman, M. (2013). *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science.

Lifshits, M. (2012). *Lectures on Gaussian Processes.* Springer.

Lin, J., Keogh, E., Fu, A., & Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. *Proceedings of the 18th IEEE Symposium on Computer-based Medical Systems* (pp. 329-334). IEEE Computer Society.

Lin, S., & Brown, D. (2006). An outlier-based data association method for linking criminal incidents. *Decision Support Systems, 41*(3), 604-615.

Liu, Y., Liu, Y.-C., & Chen, Y. (2010). Fast Support Vector Data Descriptions for Novelty Detection. *IEEE Transactions on Neural Networks, 21*(8), 1296 - 1313.

Lowe, D. G. (1999). Object Recognition from Local Scale-invariant Features. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1150–1157). IEEE.

Ma, J., & Perkins, S. (2003). Time-series novelty detection using one-class support vector machines. *Proceedings of the International Joint Conference on Neural Networks*, (pp. 1741-1745).

MacKay, D. (1994). Bayesian Methods for Backpropagation Networks. In E. Domany, J. van Hemmen, & K. Schulten (Eds.), *Mdels of Neural Networks III* (pp. 211-254). Springer.

MacKay, D. (1998). Introduction to Gaussian Processes. In C. Bishop (Ed.), *Neural Networks and Machine Learning* (Vols. NATO ASI Series, vol 168, pp. 133-165). Berlin: Springer.

Mahalanobis, P. C. (1936). On the Generalised Distance in Statistics. *Proceedings of the National Institute of Sciences of India, 2*(1), 49–55.

Mahoney, M., & Chan, P. (2002). Learning nonstationary models of normal network traffic for detecting novel attacks. *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 376-385). ACM.

Mahoney, M., & Chan, P. (2003). Learning rules for anomaly detection of hostile network traffic. *Third IEEE International Conference on Data Mining. ICDM 2003* (pp. 601-604). IEEE Computer Society.

Manevitz, L., & Yousef, M. (2002). One-class SVMs for Document Classification. *Journal of Machine Learning Research, 2*, 139-154.

Manson, G. (2002). Identifying damage sensitive, environment insensitive features for damage detection. *Proceedings of the IES Conference.*

Markou, M., & Singh, S. (2003). Novelty detection: A review, part 1: Statistical approaches. *Signal Processing, 83*, 2481-2497.

Markou, M., & Singh, S. (2003). Novelty detection: a review—part 2: neural network based approaches. *Signal Processing, 83*(12), 2499-2521.

Markou, M., & Singh, S. (2006). A Neural Network-Based Novelty Detector for Image Sequence Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(10), 1664-1677.

Maronna, R. A., Martin, D. R., & Yohai, V. J. (2006). *Robust Statistics: Theory and Methods.* Wiley.

Marsland, S., Nehmzow, U., & Shapiro, J. (2005). On-line novelty detection for autonomous mobile robots. *Robotics and Autonomous Systems, 51*(2), 191-206.

Marsland, S., Shapiro, J., & Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Networks, 15*(8-9), 1041-1058.

Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. (2009). Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams. *Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science. 5782*, pp. 79-94. Springer.

McDiarmid, C. (1989). On the Method of Bounded Differences. *Surveys in Combinatorics, 141*, 148-188.

McKay, A. T. (1935). The distribution of the difference between the extreme observation and the sample mean in samples of n from a normal universe. *Biometrika, 27*, 466-471.

Metzler, S., & Kalinina, O. V. (2014). Detection of atypical genes in virus families using a one-class SVM. *BMC Genomics, 15*, 913.

Minka, T. (2001). *A Family of Approximate Algorithms for Bayesian Inference. PhD thesis.* MIT.

Nabney, I. T. (2004). *NETLAB: Algorithms for Pattern Recognition.* Springer.

Neal, R. (1997). *Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702.* Department of Computer Statistics, University of Toronto.

Noble, C., & Cook, D. (2003). Graph-based anomaly detection. *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 631-636). ACM Press.

Ntalampiras, S., Potamitis, I., & Fakotakis, N. (2011). Probabilistic Novelty Detection for Accoustic Surveillance Under Real-World Conditions. *IEEE Transactions on Multimedia, 13*(4), 713-719.

Odin, T., & Addison, D. (2000). Novelty detection using neural network technology . *COMADEM 2000: 13th International Congress on Condition Monitoring and Diagnostic Engineering Management*, (pp. 731-743).

Oh, J. H., & Gao, J. (2009). A kernel-based approach for detecting outliers of high-dimensional biological data. *BMC Bioinformatics, 10*((Suppl 4):S7).

Opper, M. (1998). A Bayesian Approach to On-line Learning. In D. Saad (Ed.), *On-line Learning in Neural Networks.* Cambridge University Press.

Opper, M., & Winther, O. (2000). Gaussian Processes for Classification. *Neural Computation, 12*(11), 2655 - 2684.

Otey, M., Ghoting, A., & Parthasarathy, S. (2006). Fast Distributed Outlier Detection in Mixed-Attribute Data Sets. *Data Mining and Knowledge Discovery*, *12*(2), 203-228.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., & Faloutsos, C. (2002). *Loci: Fast outlier detection using the local correlation integral.* Intel Research Laboratory.

Parra, L., Deco, G., & Miesbach, S. (1996). Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Computation, 8*(2), 260-269.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics, 33*(3), 1065-1076.

Pearson, E. S., & Hartely, H. O. (1942). The Probability Integral of the Range in Samples of n Observations From a Normal Population. *Biometrika, 32*, 301-310.

Pearson, E. S., & Sekar, C. (1936). The Efficiency of Statistical Tools and a Criterion for the Rejection of Outlying Observations. *Biometrika, 28*(3/4), 308-320.

Peirce, B. (1852). Criterion for the rejection of doubtful observations. *Astronomical Journal, 2*, 161.

Peña, J. M., Lozano, J. A., & Larrañaga, P. (1999). An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters, 20*(10), 1027-1040.

Peng, X., & Xu, D. (2012). Efficient support vector data descriptions for novelty detection. *Neural Computing and Applications, 21*(8), 2023-2032.

Phua, C., Alahakoon, D., & Lee, V. (2004). Minority report in fraud detection: classification of skewed data. *SIGKDD Explorer Newsletter, 6*(1), 50-59.

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing, 99,* 215-249.

Pinto, A. S., Pronobis, A., & Reis, L. P. (2011). Novelty Detection Using Graphical Models for Semantic Room Classification. *15th Portuguese Conference on Artificial Intelligence, EPIA 2011. Progress in Artificial Intelligence. Lecture Notes in Computer Science. 7026*, pp. 326-339. Springer.

Pires, A., & Santos-Pereira, C. (2005). Using clustering and robust estimators to detect outliers in multivariate data. *Proceedings of the International Conference on Robust Statistics.*

Platt, J. (2000). Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods. In P. Bartlett, B. Schölkopf, D. Schuurmans, & A. Smola (Eds.), *Advances in Large Margin Classifiers* (pp. 61-74). MIT Press.

Platt, J. C. (1999). Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning* (pp. 185-208). MIT Press.

Pokrajac, D., Lazarevic, A., & Latecki, L. J. (2007). Incremental local outlier detection for data streams. *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining.*

Pokrajac, D., Reljin, N., Pejcic, N., & Lazarevic, A. (2008). Incremental Connectivity-Based Outlier Factor Algorithm. *Proceedings of the Conference Visions of Computer Science*, *8.* London.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers.

Quinn, J. A., & Williams, C. K. (2007). Known Unknowns: Novelty Detection in Condition Monitoring. *Third Iberian Conference, IbPRIA 2007. Pattern Recognition and Image Analysis. Lecture Notes in Computer Science. 4477*, pp. 1-6. Springer.

Quinn, J. A., Williams, C. K., & McIntosh, N. (2009). Factorial Switching Linear Dynamical Systems Applied to Physiological Condition Monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(9), 1537-1551.

Rabaoui, A., Kadri, H., & Ellouze, N. (2008). New approaches based on one-class SVMs for impulsive sounds recognition tasks. *IEEE Workshop on Machine Learning for Signal Processing* (pp. 285-290). IEEE.

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (pp. 427-438). ACM.

Ramirez-Padron, R., Foregger, D., Manuel, J., Georgiopoulos, M., & Mederos, B. (2010). Similarity Kernels for Nearest Neighbor-based Outlier Detection. *The Ninth International Symposium on Intelligent Data Analysis, Lectures Notes in Computer Science. 6065*, pp. 157-170. Tucson, Arizona: Springer-Verlag.

Ramirez-Padron, R., Mederos, B., & Gonzalez, A. J. (2013). Novelty Detection Using Sparse Online Gaussian Processes for Visual Object Recognition. *Twenty-Sixth International Florida Artificial Intelligence Research Society Conference.*

Rasmussen, C., & Williams, C. (2006). *Gaussian Processes for Machine Learning.* The MIT Press.

Rey, W. J. (1983). *Introduction to Robust and Quasi-robust Statistical Methods.* Springer.

Reynolds, D. (2009). Gaussian Mixture Models. In *Encyclopedia of Biometrics* (pp. 659-663).

Rivest, R. (1974). On the optimality of Elias's algorithm for performing best-match searches. *Information Processing, 74*, 678-681.

Roberts, S. J. (2000). Extreme value statistics for novelty detection in biomedical signal processing. *IEE Proceedings on Science, Technology and Measurement. 147*, pp. 363-367. IET.

Rodner, E., Wacker, E., Kemmler, M., & Denzler, J. (2011). One-Class Classification for Anomaly Detection in Wire Ropes with Gaussian Processes in a Few Lines of Code. *Proceedings of the 12th IAPR Conference on Machine Vision Applications (MVA 2011).*

Roth, V. (2006). Kernel fisher discriminants for outlier detection. *Neural Computation, 18*(4), 942–960.

Rottmann, A., & Burgard, W. (2010). Learning Non-stationary System Dynamics Online using Gaussian Processes. *32nd DAGM Symposium*, (pp. 192-201). Darmstadt, Germany.

Rousseeuw, P. J. (1984). Least Median of Squares Regression. *Journal of American Statistical Association, 79*, 871-880.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, & W. Wertz (Eds.), *Mathematical Statistics and Applications* (Vol. B, pp. 283-297). Dordrecht, The Netherlands.

Rousseeuw, P. J., & Hubert, M. (2011). Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1*(1), 73-79.

Rousseeuw, P., & Leroy, A. (1987). *Robust regression and outlier detection.* John Wiley & Sons, Inc.

Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (pp. 71-79). ACM.

Ryan, T. P. (1996). *Modern Regression Methods.* Wiley-Interscience.

Salvador, S., & Chan, P. (2005). Learning States and Rules for Detecting Anomalies in Time Series. *Applied Intelligence, 23*(3), 241-255.

Schölkopf, B., & Smola, A. (2001). *Learning with Kernels.* Cambridge, MA: MIT Press.

Schölkopf, B., Herbrich, R., & Smola, A. (2001). A generalized representer theorem. *Proceedings of the Conference on Computational Learning Theory* (pp. 416-426). Springer.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation, 13*(7), 1443-1471.

Sebyala, A., Olukemi, T., & Sacks, L. (2002). Active platform security through intrusion detection using naive bayesian network for anomaly detection. *Proceedings of the London Communications Symposium.*

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalization Error Bounds and Sparse Approximations. PhD thesis.* University of Edinburg.

Seeger, M. (2004). Gaussian Processes for Machine Learning. *International Journal of Neural Systems, 14*(2), 69-106.

Seeger, M., Williams, C., & Lawrence, M. (2003). Fast Forward Selection to Speed up Sparse Gaussian Processes. In C. Bishop, & B. Frey (Ed.), *Proceedings Ninth International Workshop on Artificial Intelligence and Statistics.* Key West, Florida.

Sellis, T., Roussopoulos, N., & Faloutsos, C. (1987). The R-tree: A dynamic index for multi-dimensional objects. *The VLDB Journal*, 507-518.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis.* Cambridge University Press.

Shekhar, S., Lu, C.-T., & Zhang, P. (2002). Detecting graph-based spatial outliers. *Intelligent Data Analysis, 6*(5/2002), 451-468.

Shen, Y. (2007). Outlier Detection Using the Smallest Kernel Principal Components. *PhD dissertation*. Department of Statistics, Temple University.

Sheng, B., Li, Q., Mao, W., & Jin, W. (2007). Outlier detection in sensor networks. *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing* (pp. 219-228). ACM.

Shyu, M., Chen, S., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. *Proceedings of the 3rd IEEE International Conference on Data Mining* (pp. 353-365). IEEE.

Smith, R., Bivens, A., Embrechts, M., Palagiri, C., & Szymanski, B. (2002). Clustering approaches for anomaly based intrusion detection. *Proceedings of Intelligent Engineering Systems through Artificial Neural Networks* (pp. 579-584). ASME Press.

Smola, A., & Bartlett, P. (2001). Sparse Greedy Gaussian Process Regression. In T. Leen, T. Dietterich, & V. Tresp (Ed.), *Advances in Neural Information Processing Systems. 13*, pp. 619 - 625. The MIT Press.

Sofman, B., Bagnell, J., & Stentz, A. (2010). Anytime online novelty detection for vehicle safeguarding. *IEEE International Conference on Robotics and Automation. ICRA 2010* (pp. 1247-1254). IEEE.

Song, Q., Hu, W., & Xie, W. (2002). Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 32*(4), 440-448.

Song, X., Wu, M., Jermaine, C., & Ranka, S. (2007). Conditional anomaly detection. *IEEE Trans. Knowl. Data Eng., 19*(5), 631-645.

Spiegelhalter, D., & Lauritzen, S. (1990). Sequential Updating of Conditional Probabilities on Directed Graphical Structures. *Networks, 20*, 579 - 605.

Spinosa, E. J., de Leon F. de Carvalho, A. P., & Gama, J. (2009). Novelty detection with application to data streams. *Intelligent Data Analysis, 13*(3), 405-422.

Stanley, K. O. (2004). *Efficient evolution of neural networks through complexification, Ph.D. thesis.* University of Texas at Austin.

Stefano, C., Sansone, C., & Vento, M. (2000). To reject or not to reject: that is the question: An answer in the case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 30*(1), 84-94.

Stone, E. J. (1868). On the Rejection of Discordant Observations. *Monthly Notices of the Royal Astronomical Society*.

Strassen, V. (1969). Gaussian Elimination is not Optimal. *Numerical Mathematics, 13*, 354-356.

Sugiyama, M., Krauledat, M., & Müller, K. R. (2007). Covariate Shift Adaptation by Importance Weighted Cross Validation. *Journal of Machine Learning Research, 8*, 985–1005.

Sun, P., Chawla, S., & Arunasalam, B. (2006). Mining for Outliers in Sequential Databases. *Proceedings of the Sixth SIAM International Conference on Data Mining.* Society for Industrial Mathematics.

Swain, M., & Ballard, D. (1991). Color Indexing. *International Journal of Computer Vision, 7*(1), 11-32.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining.* Addison Wesley.

Tandon, G., & Chan, P. (2007). Weighting versus pruning in rule validation for detecting network and host anomalies. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 697-706). ACM.

Tang, J., Chen, Z., Fu, A., & Cheung, D. (2002). Enhancing effectiveness of outlier detections for low density patterns. *Advances in Knowledge Discovery and Data Mining*, 535-548.

Tao, Y., Yi, K., Sheng, C., & Kalnis, P. (2009). Quality and efficiency in high dimensional nearest neighbor search. *Proceedings of the 35th SIGMOD international conference on Management of Data* (pp. 563-576). ACM.

Tavakkoli, A., Nicolescu, M., Bebis, G., & Nicolescu, M. (2008). Efficient Background Modeling through Incremental Support Vector Data Description. *Proceedings of the 19th International Conference on Pattern Recognition* (pp. 1-4). Tampa, FL: IEEE.

Tax, D. (2001). One-class classification; Concept-learning in the absence of counter-examples. *Ph.D. thesis*. Delft University of Technology.

Tax, D. M., & Duin, R. P. (2004). Support Vector Data Description. *Machine Learning, 54*(1), 45-66.

Tax, D. M., & Laskov, P. (2003). Online SVM Learning: From classification to data description and back. *IEEE 13th Workshop on Neural Networks for Signal Processing. NNSP'03.*, (pp. 499-508).

Tax, D., & Duin, R. (1999). Support vector domain description. *Pattern Recognition Letters, 20*(11), 1191-1199.

Thompson, W. R. (1935). On a criterion for the rejection of observations and the distribution of the ratio of deviation to sample standard deviation. *The Annals of Mathematical Statistics, 6*(4), 214-219.

Tikhonov, A. N., & Arsenin, V. Y. (1977). *Solutions of Ill-posed Problems.* W.H. Winston.

Tipping, M. (2001). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research, 1*, 211-244.

Tresp, V. (2001). Scaling Kernel-based Systems to Large Data Sets. *Data Mining and Knowledge Discovery, 5*(3), 197 - 211.

Tsay, R., Peña, D., & Pankratz, A. (2000). Outliers in multivariate time series. *Biometrika, 87*(4), 789-804.

Vaidya, J., & Clifton, C. (2004). Privacy-preserving outlier detection. *Fourth IEEE International Conference on Data Mining* (pp. 233-240). IEEE.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag.

Verboven, S., & Hubert, M. (2010). Matlab Library LIBRA. *Wiley Interdisciplinary Reviews: Computational Statistics, 2*, 509-515.

Vilalta, R., & Ma, S. (2002). Predicting rare events in temporal domains. *Proceedings of the IEEE International Conference on Data Mining* (pp. 474-481). IEEE Computer Society.

Vincent, P., & Bengio, Y. (2002). Manifold Parzen Windows. *Advances in Neural Information Processing Systems*, (pp. 825-832).

Von Mises, R. (1964). *Mathematical Theory of Probability and Statistics.* New York: Academic Press.

Wald, I., & Havran, V. (2006). On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N). *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, (pp. 61-69).

Wang, C. H. (2009). Outlier identification and market segmentation using kernel-based clustering techniques. *Expert Systems with Applications, 36*(2), 3744-3750.

Wei, L., Yang, Y., Nishikawa, R., Wernick, M., & Edwards, A. (2005). Relevance Vector Machine for Automatic Detection of Clustered Microcalcifications. *IEEE Transactions on Medical Imaging, 24*(10), 1278-1285.

Whiteson, S., Stone, P., Stanley, K. O., Miikkulainen, R., & Kohl, N. (2005). Automatic feature selection in neuroevolution. *Proceedings of the Genetic and Evolutionary Computation.* ACM Press.

Williams, C., & Seeger, M. (2001). Using the Nystrom Method to Speed up Kernel Machines. In T. Leen, T. Dietterich, & V. Tresp (Ed.), *Advances in Neural Information Processing Systems. 13*, pp. 682–688. The MIT Press.

Winlock, J. (1856). On Professor Airy's objections to Peirce's criterion. *The Astronomical Journal, 4*, 145-147.

Wong, W., Moore, A., Cooper, G., & Wagner, M. (2003). Bayesian network anomaly pattern detection for disease outbreaks. *Proceedings of the 20th International Conference on Machine Learning* (pp. 808-815). AAAI Press.

Wright, T. W. (1884). *A Treatise on the Adjustment of Observations by the Method of Least Squares.* New York.

Wu, F., Wang, T., & Lee, J. (2010). An online adaptive condition-based maintenance method formechanical systems. *Mechanical Systems and Signal Processing, 24*(8), 2985-2995.

Wu, M., & Jermaine, C. (2006). Outlier detection by sampling with accuracy guarantees. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 767–772). ACM Press.

Wu, M., & Ye, J. (2009). A small sphere and large margin approach for novelty detection using training data with outliers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(11), 2088-2092.

Xiao, Y., Wang, H., & Xu, W. (2014). Hyperparameter Selection for Gaussian Process One-Class Classification. *IEEE Transactions on Neural Networks and Learning Systems*.

Yamanishi, K., Takeuchi, J., Williams, G., & Milne, P. (2004). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery, 8*(3), 275-300.

Yen, S., Shih, C., Chang, H., & Li, T. (2010). Nearest neighbor searching in high dimensions using multiple KD-trees. *Proceedings of the 10th WSEAS International Conference on Signal processing, Computational Geometry and Artificial Vision*, (pp. 40-45).

Yershova, A., & LaValle, S. (2007). Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. *IEEE Transactions on Robotics, 23*(1), 151-157.

Yeung, D. Y., & Ding, Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition, 36*(1), 229-243.

Yeung, D., & Chow, C. (2002). Parzen-Window Network Intrusion Detectors. *International Conference on Pattern Recognition. 4.* IEEE Computer Society.

Yin, G., Zhang, Y. T., Li, Z. N., Ren, G. Q., & Fan, H. B. (2014). Online fault diagnosis method based on Incremental Support Vector Data Description and Extreme Learning Machine with incremental output structure. *Neurocomputing, 128*, 224-231.

Yu, D., Sheikholeslami, G., & Zhang, A. (2002). Findout: Finding outliers in very large datasets. *Knowledge and Information Systems, 4*(4), 387-412.

Yu, M., Yu, Y., Rhuma, A., Naqvi, S. M.-R., Wang, L., & Chambers, J. A. (2013). An Online One Class Support Vector Machine-Based Person-Specific Fall Detection System for

Monitoring an Elderly Individual in a Room Environment. *IEEE Journal on Biomedical and Health Informatics, 17*(6), 1002-1014.

Zhang, J., & Wang, H. (2006). Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and Information Systems, 10*(3), 333-355.

Zhang, K., Hutter, M., & Jin, H. (2009). A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data. *Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science. 5476*, pp. 813-822. Springer.

Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery, 1*(2), 141-182.

Zhang, Y., Meratnia, N., & Havinga, P. (2010). Outlier Detection Techniques for Wireless Sensor Networks: A Survey. *IEEE Communications Surveys and Tutorials, 12*(2), 159-170.

Zhang, Z., & Shen, H. (2005). Application of online-training SVMs for real-time intrusion detection with different considerations. *Computer Communications, 28*(12), 1428-1442.

Zhou, J., Fu, Y., Sun, C., & Fang, Y. (2011). Unsupervised distributed novelty detection on scientific simulation data. *Journal of Computational Information Systems, 7*(5), 1533-1540.

Zhu, F., Ye, N., Yu, W., Xu, S., & Li, G. (2014). Boundary detection and sample reduction for one-class Support Vector Machines. *Neurocomputing, 123*, 166-173.

Zhuang, L., & Dai, H. (2006). Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers, 1*(7), 32-40.

Zliobaite, I. (2009). *Learning under Concept Drift: an Overview*. Vilnius University, Lithuania, Faculty of Mathematics and Informatics.