# NOVEL MULTISTAGE PROBABILISTIC KERNEL MODELING IN HANDWRITING RECOGNITION

MAHDI BIPARVA

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

August 2013

# Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **MAHDI BIPARVA**

Entitled: **NOVEL MULTISTAGE PROBABILISTIC KERNEL MODELING IN HANDWRITING RECOGNITION**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Computer Science**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

Dr. Yuhong Yan

_____ Examiner

Dr. Adam Krzyzak

_____ Examiner

Dr. Louisa Lam

_____ Supervisor

Dr. Ching Y. Suen

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Christopher Trueman, Ph.D., Interim Dean

Faculty of Engineering and Computer Science

# Abstract

NOVEL MULTISTAGE PROBABILISTIC KERNEL
MODELING IN HANDWRITING RECOGNITION

MAHDI BIPARVA

The design of handwriting recognition systems has been widely investigated in pattern recognition and machine learning literature. It was first attempted to enhance the system's performance by improving the recognition rate to reach 100% which has not achieved yet. Despite the low misclassification error rate, there are still some misclassified test samples. This imposes a very high cost on the whole recognition system. The cost has to be reduced as much as possible which consequently leads to the consideration of reject option to prevent the recognition system from classifying test samples with high prediction uncertainty.

The main contribution of this thesis is to propose a novel multistage recognition system that is capable of producing true prediction probability outputs and then reject test samples accordingly. An argument is supported that principally formulated probabilistic classifiers are the best reliable candidates to be utilized in the consideration of reject option. The implementation of reject option based on either non-probabilistic classifier's output score or conversion to probability measures is prone to mistake when compared to an accurate prediction probability output.

The Convolutional Neural Network (CNN) is utilized as the automatic feature extractor that can properly harness the spatial correlation of the input raw handwritten images and extract a feature vector with strong discriminative properties. The SVM is used as a powerful classifier to accurately deal with the issue of big data sets. The authentic intuition of extracting the most informative training samples by using the

distinguished support vector set from the SVM is also proposed.

The Gaussian process classifier (GPC) in the Bayesian nonparametric modeling framework is introduced as the core element of the whole recognition system that can reliably provide an accurate estimate of the posterior probability of the class membership. Experiments under various inference methods, likelihood functions, covariance functions, and learning approaches are conducted in the hope of finding the best model configuration and parameterization. The models are evaluated on two popular handwritten numeral data sets known as MNIST and CENPARMI. The best GPC model in this multistage framework on MNIST can reach 100% reliability rate with the lowest rejection rate of 1.48%, the best result achieved in the field.

Another inherently probabilistic classifier, known as relevance vector machine (RVM), is also investigated. The RVM is formulated through the sparse Bayesian linear modeling to classification problems and it produces reliable prediction probability outputs. However, In comparison of the GPC with RVM, this argument is experimentally supported that the sparsity is not capable of improving the rejection performance on the data sets.

# Acknowledgments

First of all, I am very grateful of the blessings that God has bestowed upon me in this wonderful life, the opportunity to live long and healthy to see, think, and learn what he has created in this amazing world.

I would like to thank my supervisor, Professor Ching Y. Suen for all his support, and valuable feedback during my research.

I would also like to thank my friends, the research manager, and the secretary at the Center for Pattern Recognition and Machine Intelligence (CENPARMI).

Most of all, I would like to express my eternally sincere gratitude to my father, my mother, my sister, and my brothers who supported me throughout my studies and research very far away from the home. I'd be forever indebted for their unconditional love and encouragement.

Last but not least, my kind thanks go to all my close friends and my previous teachers and instructors.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, the motivation, challenges, previous work, and proposed methods will be presented along with an outline of the thesis. In section 1.2, the whole handwriting recognition field of research will be introduced briefly while the previous models and approaches will be presented in section 1.3. Meanwhile, the challenges, problems, and the rationale behind a better model will be highlighted in section 1.4. Section 1.5, will introduce the model and its characteristics while the outline and the thesis' structure will be given in section 1.6.

## 1.1 Research Topic

The field of pattern recognition and machine learning has emerged as a trending line of research in recent decades. The problems that have been addressed in the literature have been categorized based on the type of data available under supervised and unsupervised learning. In supervised learning, the data consist of the input and the output variables as the main goal is to seek a projection from the input space to the output space. To achieve a proper projection, the question is whether the raw input space does properly represent the correlation and adequate information in the data so that the projection to the output space would be sufficiently successful. Consequently, the idea of feature extraction emerges through two perspectives that

appeared frequently in the literature. One approach focuses on the most accurate method to conduct the feature extraction and use a simple projection to complete the whole learning process. The other approach is to use a complicated projection in which the feature extraction is implicitly embedded in the whole process. Researchers chose either one according to their domain and applications. Based on the type of the output values, the supervised learning problems are further divided into regression or classification cases. The former has real values while the latter is represented using discrete label values.

Unsupervised learning problems do not carry output values. It is only the input data that has to be processed in a way to provide appropriate interpretation or representation of the data. Unsupervised learning can be seen as a preliminary stage to other type of learning such as supervised learning. The kind of problems can be categorized as feature extraction, high-dimensional data visualization, and density estimation. Each of these approaches to the unlabeled data seeks particular goals at the end. For instance one goal in density estimation is to model the density of the input data in order to detect outliers consequently. Transformation of the input data nonlinearly into a possibly higher and complicated feature space to have specific characteristics is the goal sought in feature extraction treatment in unsupervised learning. Visualization of high-dimensional not-interpretable data in a low dimensional data space is the other treatment of unsupervised learning that provides the means and insights for human experts to better understand the trends and rules in new domains of science. For instance DNA sequence extraction can be named as such problems.

Many application domains have been studied by researchers in pattern recognition and machine learning in order to empirically prove the applicability of the proposed methods and algorithms. Such applied domains can be counted as handwriting recognition, speech recognition, object detection and recognition, data mining and natural language processing. Handwriting recognition has been extensively studied by many

researchers over the last few decades, and many approaches and models have been introduced so far. By using the digitized data of humans' handwriting, computers can apply different approaches and mechanisms to understand and interpret them. The ultimate goal sought in this domain is that machines can read human handwriting input with 100% reliability and accuracy. Indeed, this is very close to succeed as seen in some recent handwriting recognition systems [13], [15].

Based on how humans' handwriting data is acquired, the problems can be separated into on-line and off-line recognition systems. The on-line recognition systems use additional information during the input data acquiring process, such as pen pressure, time, orientation, movement and curvature. This is the reason such systems could attain high precision and reliable results in the literature. The challenge still remains in the off-line recognition systems since they lack the useful information during the learning and classification processes. This shortcoming has made them the focus of recent research in this area considering that off-line systems' performance and accuracy cannot be compared to the on-line ones.

## 1.2 Motivation

The performance of the designed system can be measured through the recognition rate on the test set. The higher the rate is, the more generalized the system is to the unseen test set. The need to avoid over and under fitting in order to reach a sufficient generalization level has always been pursued in the classification modeling design. To prevent any confusion and misinterpretation throughout this research, three measures of a classifier's performance will be defined according to the following rates:

$C =$ Number of correctly-classified test samples,

$R =$ Number of rejected test samples,

$N =$ Number of not-rejected correctly-classified test samples,

$T =$ Number of all test samples,

$$\text{Recognition rate} = \frac{C}{T}, \tag{1.1}$$

$$\text{Rejection rate} = \frac{R}{T}, \tag{1.2}$$

$$\text{Reliability rate} = \frac{N}{T-R}. \tag{1.3}$$

It should be noted that it is not sufficient to focus solely on the improvement of recognition rate during the system design. Since a model that would achieve 100% recognition rate is not yet achieved, there has always been some part of the training set that is misclassified. All this will strongly harm the reliability of the system while consequently increasing the decision cost.

For instance, in the case of automatic bank check recognition system, no matter how high recognition rate the system has, the reliability rate along with rejection rate play an important role throughout the whole decision-making process. If the system can reach 100% reliability rate, then it will assure that there will not be any misclassified test samples and the overall cost of the system will be low. However, another factor, known as the rejection rate, plays an important role as well. By introducing some rejection mechanism based on the classifier's prediction uncertainty measure, some samples can be rejected to decrease the decision making cost to the minimum. In other words, the designed recognition system rejects samples with low prediction certainty leading into reaching higher reliability rate which means reducing the misclassification error costs. It is too optimistic to say that 100% reliability rate can be achieved easily under no rejected samples condition as it is not practically reached yet on famous handwritten data sets.

Assuming the 100% reliability is achieved, reject option imposes new costs to the whole system. Under high rejection rate, the situation is as if the costs are shifted from dealing with misclassified samples to rejected ones. It is obvious that in this case, all the rejected samples have to be classified using human intervention in the automatic recognition process and this will be very undesirable. Now, it is clear that the research's focus is implicitly shifted to using classifiers that can provide the decision making mechanism a true proper probabilistic output values that could estimate principally the prediction uncertainty ultimately. This reliable uncertainty measure will implicitly lead to a system with almost the same recognition but lower rejection rates. As a result, the system will have much lower cost at the end. This will be investigated further and will form the basis of this experiment.

## 1.3 Previous Work

Pattern recognition and machine learning literature has developed many classification modeling approaches to provide accurate algorithms and methods to achieve high performance systems on various application such as speech, face, handwriting, fingerprint and gesture recognition. Moving from classical classifiers of K-nearest neighbor(KNN), Multi-layer perceptron(MLP), Hidden Markov model(HMM), Radial basis function networks(RBF) to more advanced classifier of Support vector machine(SVM), Deep belief network(DBN), and Deep restricted Boltzmann machine (DRBM), they have been extensively used and surveyed in the literature [18], [2], and [23].

The approaches to handwriting recognition have been quite broad. Some research has been developed on using single classifier such as SVM and then evaluate the performance on well-known isolated numeral data sets such as MNIST, USPS, CENPARMI [23]. Since SVM is a kernel machine, the feature extraction can be addressed in a principled way too. The other classification methods have been experimented

too. For instance [9] has used a 6-layer neural network to achieve a high recognition rate of 99.65% on MNIST data set.

Some researchers have attempted to use manually extracted features to improve recognition rate [12], [3]. The idea is that some handcrafted features could better represent human experts knowledge embedded in the raw data. Therefore, such extracted features will be more discriminative and informative to be processed in the task of classification. For instance, [24] extracts 8 direction gradient features after some preprocessing is applied and eventually feed them into an SVM classifier for the discrimination task. It is reported that the recognition rate reaches 91.15% on CENPARMI numeral data set.

The work completed in [22] compares several classification methods such as KNN and SVM along with Convolutional Neural Network (CNN) which reveals the superiority of CNN over the others. Following those results and the idea of automatic feature extraction, [20] proposed a trainable automatic feature extractor using the CNN performance and extract hidden features as new input variables for the SVM classifier. The high SVM discriminative power could make it a very reliable recognition system which could outperform both SVM and CNN when treated individually. These concepts are further investigated in the research of [33] where two models are proposed using CNN and SVM. In the first model called the Hybrid CNN-SVM, the idea of automatic feature extraction is completed using a trainable CNN on MNIST data set. Then, an SVM classifier is trained using the extracted hidden features. It is stated that the high recognition rate of 99.81% on the test set can be achieved, and further under 100% reliability rate condition, the recognition rate reaches 94.4 % resulting to the rejection rate of 5.60%. In other model, the combination of CNN trained on the raw data and SVM on the handcrafted features is examined. The results reveal that this combination model is not as successful as the hybrid model.

Numerous attempts are made to increase the recognition performance of a system using the idea of multiple classifier system (MCS) paradigm. Researches such as

[19] and [44], employ the ensemble approach to combine several classifiers' predicted labels to form the final decision under some schemes such as majority voting which outperform individual classifier approach. [8] proposed a system consisting of 35 committees of CNN classifiers which resulted into the recognition rate of 99.77% on MNIST data set. Beside the ensemble approach for the MCS, the other paradigm is the multistage cascade architecture. Using 5 CNN classifiers trained on different data sets using dissimilar spatial pooling of sub-sampling and max-pooling, the recognition rate of 99.77% could be achieved.

The importance of designing reliable systems for practical applications has attracted many researchers to incorporate the reject option into recognition systems. This has lead to the investigation and proposition of new rejection criteria. Linear Discriminant Analysis Measure (LDAM) has been proposed in [17]. It is claimed that this rejection criterion can surpass the performance of the other two classic criteria: the First Rank Measurement (FRM) in [10] and The First Two Rank Measurement (FTRM) in [45]. Two novel learning-based rejection criteria have been recently proposed as well by using the single classifier paradigm [50] which are known as SVM-based measurement (SVMM) and Area Under the Curve measurement (AUCM). Following the majority voting mechanism in MCS, that research incorporated the same idea for reject option in the hope of increasing reliability of recognition systems. It is claimed that by following MCS paradigm and using two novel rejection criteria, the rejection rate reaches 4.09% under 100% reliability rate on the MNIST data set.

Having said that what the literature review is and how different approaches been developed and applied, the following arguments will be supported in this research. It is not necessary to follow the MCS or other committee-based paradigms to achieve a reliable recognition system or decrease the cost of misclassification and rejection. The underlying classification methods and the means on which the rejection applies is the missing point in the recent researches. It will be shown that based on the separation of inference and decision processes given in the decision theory, the probability

theory gives the grounds on which the rejection option can fundamentally be placed. It can be obviously concluded that probabilistic classifiers would then be the best candidates of the underlying classifier to reach high reliability and low rejection state in a recognition system.

## 1.4 Challenges

There have been many achievements in the context of designing handwriting recognition systems considering numerous successful systems have been proposed. It has always been the ultimate goal of the community to achieve 100% reliable systems with the lowest rejection rate. This has been very difficult to achieve due to the fact that either the classifiers are not capable enough to provide the grounds for such demand or they have not been used properly at the right place. For instance, the utilization of not-inherently-probabilistic classifiers, such as SVM, in the context of rejection option can be seen as one of the issues.

Regarding the SVM, it is obvious that the predictions are not probabilistic in nature as they are hard binary decisions in classification and point estimates of the test samples in regression. SVM is basically categorized as decision machines in the context of classifier design which are not intrinsically probabilistic formulated. Thus, it should not be considered for decision making tasks such as rejection option. Posterior probabilities of class membership are the key concept that not many classification modeling approaches try to address. In the case of SVM, many attempts have been made to coerce posterior probability estimates from the classifier output values [36].

In the context of classification, assuming a test sample $\mathbf{x}_*$, classification model is strongly preferred to be capable of providing class membership posterior probability of $p(y_* \in \mathcal{C}|\mathbf{x}_*)$. As a result, the prediction uncertainty can be reliably expressed. This approach will lead to the distinction of the inference stage from the decision stage [11]. It is necessarily important to address the issues of applying asymmetric

misclassification costs, varying class proportions and most interestingly, the reject option by using the posterior probability estimates in practical applications.

In probabilistic binary classifiers, the logistic sigmoid function can be interpreted as a way to provide the estimate of the posterior probability of class membership. It is well explained in [5] that provided the underlying function $f(\mathbf{x})$ is sufficiently flexible, the estimate of $\sigma(f(\mathbf{x}))$ will become exact in the limit of infinite data set. On the contrary, hard decision machines such as SVM outputs the class label $\mathbf{y}_*$ of the test sample $\mathbf{x}_*$ as a result of hard binary decision. It is the well-known SVM deficiency that suffers from generating posterior probability for the test samples. However, there have been some attempts to mitigate this issue as [36] has proposed the a posteriori fitting of a sigmoid function to the fixed SVM outputs. Ultimately, although it is true that the output values will be presented in the range of $[0, 1]$, it will be illustrated later that the output value of the fitted sigmoid function will not provide a proper approximation to the desired posterior probability.

The argument given in [46] perfectly demonstrates this problem by developing an example given for SVM and the probabilistic classifier of Relevance Vector Machine (RVM). It is shown that considering the definition of log-odds as $\log(p(y \in \mathcal{C}_{+1}|\mathbf{x})/p(y \in \mathcal{C}_{-1}|\mathbf{x}))$, $f(\mathbf{x})$ cannot be a reliable function of the log-odds due to the nature of the objective function in SVM formalism. In the example, 1000 data samples are sampled uniformly from two overlapping probability distribution of $[0, 1]$ for one class and $[0.5, 1.5]$ for the other class. The SVM, RVM and correct predicted log-odds are depicted in Figure 1. The superiority of RVM providing a better posterior probability estimate to the SVM is clearly illustrated in the figure. SVM outputs are rescaled for the sake of a better visualization and it is clear that between $[0.5, 1]$, the correct log-odds is zero and outside is infinite. Generally it can be expressed that if the underlying function $f(\mathbf{x})$ can mimic the behavior of the correct log-odds, the sigmoid of the function can better estimate the true posterior probability. Therefore, the rejection decisions will be more reliable and the sigmoid outputs better represent

the uncertainty of the predictions. As illustrated in Figure 1, RVM offers a reasonable approximation of the correct log-odds whereas the SVM output is not following a reasonably similar behavior. It is perfectly clear that if this method of coercing posterior probability outputs out of the SVM is being used in a real application with reject option or asymmetric misclassification costs required, then the decisions will be very costly and imprecise due to the absence or lack of accurate posterior estimates.

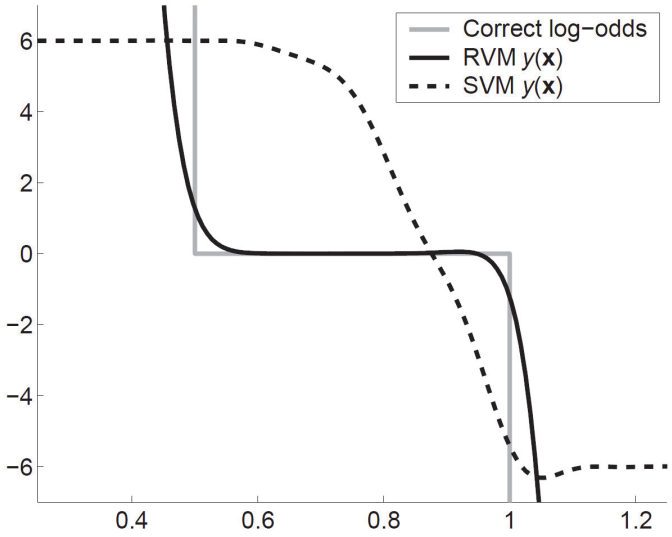


Figure 1: Comparison of correct log-odds with RVM and SVM estimates [46]

SVM is extensively used in handwriting recognition systems with the consideration of reject option. If the recent given justification is followed, the reason that SVM cannot reach a small number of rejected samples along with high reliability rate will be definitely apparent. One of the motivations behind this thesis is then to introduce true probabilistic classifiers such as GPC and RVM which are capable of providing a better estimate of the posterior probability and representation of the prediction uncertainty. However, it should be noted that the power and accuracy of SVM as a fast and reliable decision machine will be still harnessed in this research and it will be utilized at a proper stage in the classification framework.

## 1.5 The Proposal

It has always been demanded to reach an accurate and reliable recognition system for real practical applications. The focus in the literature has been mainly on improving the performance of classifiers. However, the recent researches are started to focus on the reject option and how to incorporate reliability in the classification process. It is imperative to consider both aspects of performance in the recognition system design process. The classification model has to be able to fundamentally provide the means to achieve high reliability performance along with low rejection rate on the unseen test set.

Probability theory is the most well-known approach that deals with the issue of reliability in a principled manner. Therefore, it can be said that classifiers that are intrinsically probabilistic can fundamentally measure the uncertainty of predications. This change of direction from non-probabilistic decision machines to probabilistic classification models is particularly followed in this thesis.

SVM is extensively used in pattern recognition and machine learning literature. The superiority of SVM in accurately formulating the classification tasks is experimentally proven. SVM is very fast and accurate in the process of training and prediction which is one of its attractive advantages. Moreover, the formulation of kernel trick in SVM and the idea of non-linear projection of input variable into a high dimensional infinite feature space have been added to the benefits of using SVM as the proper classifier. The formulation of objective functions so that the classification problem can be resolved using optimization is the other strength. However, the acknowledged deficiency of not being able to provide true uncertainty measure of the predictions has drastically reduced the chance of its practicality in demanding applications.

Gaussian process models are introduced in this research as a probabilistic counterpart of the SVM kernel machine. The Bayesian formulation in Gaussian processes has

provided the strong foundation in obtaining precise uncertainty measures of predictions. Gaussian process classifiers (GPC) are presented as the natural result of such approach to classification problems. The accurate estimate of posterior probability for class membership is principally provided under this classification modeling approach. Consequently, reliability of the recognition system can be substantially improved by using this measure of uncertainty. It will be experimentally shown that not only can GPC achieve a high rejection rate, but also recognition rate will be retained to some acceptable degree from the highest SVM rate achieved in the literature.

Another probabilistic approach is also investigated in the thesis. Following the idea of linear regression modeling approach and Logistic regression classifiers, a sparse kernel classifier can be achieved under Bayesian formulation using special priors on the weight parameters of the model. Relevance Vector Machine (RVM), as a parametric classifier, does systematically utilize the idea of kernel trick and can subsequently be regarded as the exact probabilistic counterpart of SVM. It is obvious that both classifiers are extremely sparse. RVM defines sparsity in the context of parametric methods, and Automatic Relevance Determination (ARD) treatment [30] of the prior on the weights of the model. Sparsity is defined as the natural outcome of the optimization of the objective function. It is apparent that sparse models do not utilize the entire training set in the prediction process while discarding some data during the training process. This is where the nonparametric Bayesian model of Gaussian process classifier will supersede the other two in the context of reject option. SVM lacks the probabilistic predictions on one hand, while on the other it suffers from the sparsity once reliability and rejection becomes an important factor along with recognition. Although the RVM is intrinsically probabilistic, the sparsity in the prediction process prevents the model from taking advantage of the entire training set to provide an accurate estimate of class membership posterior probability as accurate as GPC does.

Putting aside the importance of classifier design, feature extraction plays a significant influential role. There has to be ways to extract features such that the spatial correlation between adjacent pixels would be preserved and represented in the new projected feature space especially in the domain of two dimensional raw images. CNN has been chosen as the automatic feature extraction method due to its applicability and superiority in integration with powerful classifiers such as SVM. The efficiency of such hybrid treatment has been experimentally proven in the literature.

Being able to take advantage of fast and accurate recognition of SVM, and reliable probability predictions of GPC and RVM, a cascade classification paradigm will be approached in this thesis. The authentic intuition of extracting the most informative data samples from the training set using SVM is proposed. Once the SVM is trained, it is straightforward to extract support vector (SV) set. This SV set will then be regarded as the new training set and will be fed into the next stage classifier of either GPC, RVM or even SVM itself.

The proposed hybrid classification approach will be evaluated on two famous handwritten numeral data sets namely MNIST and CENPARMI. In the hope of achieving high generalization level, k-fold cross-validation is used throughout the experiment. The best kernel or covariance function with its hyperparameter values will be obtained by using 5-fold cross validation. The best model for GPC, RVM and SVM will be derived from the 5-fold cross-validation sets and then the performance of models will be compared based on three criteria. Recognition rate is the most obvious one. Area under the curve (AUC) of the receiver operating characteristic (ROC) is the other one to be calculated. This is considered as a way to measure the overall improvement of the classifier over the incorporation of reject option in the recognition system. Lastly, the rejection rate will be computed for a fixed reliability rate over all the three classifiers in order to show which one excels in measuring the prediction uncertainty. Obviously the cutoff point for the reliability and rejection rates will only be chosen for the sake of representing the superiority and capability of each model in

achieving the best estimate of prediction uncertainty.

## 1.6 Thesis Outline

In the following the content and organization of the thesis will be presented for each chapter briefly.

Chapter 2 will give a thorough theoretical background on the classification models and respectively various inference and learning methods. At the beginning of the chapter, the general idea of nonparametric models will be introduced. Gaussian processes are derived as the natural treatment of Bayesian learning approach to the classification problem. The formulation of regression problems out of which the Gaussian process regression (GPR) is derived will be then explained. GPR will be analyzed from two distinct points of view namely weight-space and function-space. Required inference and learning methods will be demonstrated respectively. Then the chapter continues with the extension of GPR to Gaussian process classification model (GPC). The two similar points of view leading to GPC will be described and the relative inference and learning methods will be studied. The analytic approximation of Laplace method for single latent GPC will be explained thoroughly as one of the obvious and quick approximate methods to GPC. The chapter ends with the treatment of the sparse Bayesian linear model which will lead to RVM classifier.

Chapter 3 will cover recognition system design, different stages and procedures to setup the classifiers along with any data preprocessing or partitioning. The first stage of the system is presented with the idea of automatic feature extraction using CNN. Then, the authentic intuition of melting together SVM and GPC or RVM will be proposed. The hybrid cascade modeling approach to take advantage of fast accurate SVM classifier in conjunction with the reliable probabilistic GPC or RVM classifiers will be introduced. Informative data extraction will be defined as a promising way to bypass computation complexity included in the nonparametric GPC classifier. Methods to

model assessment will be explained. AUC of ROC as a reliable approach to consider recognition and rejection performances together will be exploited respectively. Various multi-class classification strategies will be presented. The OVA scheme will be chosen as the accurate and efficient approach for single latent GPC and binary RVM. At the end, different rejection criteria proposed in the literature will be described and the one appropriate for the research will be chosen.

Chapter 4 will provide the experimental results while trying to give an analysis for each classifier. The whole multistage model will be evaluated on the famous handwritten numeral data sets of MNIST, and for the sake of generalization, CENPARMI. Single-latent and multi-latent Gaussian process classifiers will be separately examined. By using the 5-fold cross validation, the various configuration and parameterization of different covariance functions will be investigated and the best one for each covariance function will be chosen. Possible inference methods will be tried and compared in the hope of achieving better results. At the same time, the best RVM classifier will be found. In the same fashion, an SVM classifier using a similar kernel function to RVM will be trained and the best hyperparameter values cross-validated. At the end, the test results of all classifiers will be compared with each other.

Chapter 5 will draw the conclusions on the whole proposed hybrid cascade classification system. The contributions of the thesis will be highlighted by looking into the experimental achievements. A fair analysis on the whole experiment will be provided in hopes to pave the way for future research work and experiments.

# Chapter 2

# Theoretical Background

In this chapter, the theoretical background and the mathematical framework supporting this thesis will be explained. The chapter begins with an introduction to kernel models given in section 2.1. Then, the general idea of nonparametric Bayesian models will be expressed in section 2.2. It proceeds with the fundamental concepts of Gaussian Processes that is explained subsequently. Two perspectives toward understanding this mathematical model will be given by explaining how this model can be applied to regression and eventually extended to classification problems. Various inference methods, learning approaches, and covariance functions will be consequently presented. The idea of probabilistic classifiers will be extended to sparse Bayesian linear models in section 2.3 and further the formulation of Relevance Vector Machine (RVM) at the end.

## 2.1 Kernel Models

Pattern Recognition and machine learning problems can be categorized in parametric and nonparametric models for both regression and classification problems. In parametric problems, the ultimate goal is to learn the parameters of the model from the training set, and then discard the whole training data. Prediction and decision

making can be done properly by using the learned model's parameters. In other perspective, it can be illustrated such that a function maps the input vector to output value using adaptive parameters of the model. The mapping can be done in linear and nonlinear manners. For instance, neural networks are categorized as nonlinear parametric models.

On the other hand, nonparametric models are another class of models that relies on the whole or a subset of the training set for the task of prediction. They are also known as memory based models since they keep the training set during the whole process. Nonparametric models require a measure of similarity using the concept of kernel functions which are used to measure the similarity between two vectors in input space. Kernel models are seriously introduced in the context of large-margin classifiers and applied in models such as support vector machine. The technique of kernel substitution is applied in many well-known algorithms in machine learning literature such as principal component analysis [41], nearest-neighbor classifiers and the kernel Fisher discriminant [27], [40] , [4]. One evident property of nonparametric models is that they are generally fast in the training process but slow in the prediction of test samples.

Depending on whether kernel models rely on the whole or subset of the training set, learning algorithms can be categorized based on the concept of sparsity. One limitation of non-sparse kernel models is that they are extremely computationally demanding. The infeasiblity of evaluating all possible pairs of training samples with each other along with test samples during the training and testing processes is a huge practical burden. To tackle this problem, some learning algorithms have been introduced in order to provide sparse solutions. Therefore, the value of kernel functions for the new test samples would then need to be evaluated for a subset of training set during the prediction phase.

As seen in many researches, it is important to determine what learning algorithm suits the problem at hand. For cases the prediction speed is important and the nature

of the prediction task is purely recognition, the concept of sparsity might be helpful and promising. However, there are problems demanding for other types of tasks such as decision making and reject option. It is not certain if sparse solution can provide accurate outputs for making the right decision on issues such as rejection. Further investigation in this research will demonstrate that non-sparse kernel models such as Gaussian Processes are more accurate in reject option considered and provide better results when compared to the sparse learning algorithms such as non-probabilistic support vector machine or the probabilistic relevance vector machine.

In a bigger picture, it can be said that the superiority of performance in the consideration of reject option stems from the fundamental of probabilistic models and the advantages to non-probabilistic ones. According to the decision theory and probability theory, the best way to measure uncertainty is through the posterior probability. Therefore, the models that are intrinsically probabilistic and provide the measure of uncertainty in a principled framework are reliable for decision making and reject option. The SVM is basically a decision machine and any attempts to draw a probabilistic output measure out of it would not provide a reliable foundation to reject. As it will be presented later, Gaussian Process models and the RVM are two probabilistic models that attempt to provide the posterior probability output even approximately in the hope of achieving a better recognition and rejection rates.

## 2.2 Nonparametric Bayesian Models: Gaussian Processes

Gaussian processes are a class of probabilistic learning algorithms categorized as a kernel machine that make the inference, learning, prediction along with model selection in a practical, and principled approach. It means that on one hand they are benefited from the properties of being a kernel machine, and on the other hand they are purely probabilistic in nature. Consequently, they can provide a probabilistic interpretation of the model predictions regarding the uncertainty of the model for a

particular test sample. Moreover, the provided unified framework for model selection and learning procedures has been constructed very well under Bayesian formalism and hierarchical stages.

Furthermore, Gaussian processes are categorized as nonparametric methods. Unlike a parametric model for which the best weight parameters have to be determined, there is no risk of over-fitting which could threaten the performance of the whole model. Another interpretation that can be drawn from being nonparametric is that the number of training samples is infinite. This perspective will be encountered later when the concept behind prior over functions is explored.

Moreover, The Bayesian treatment in the framework is extremely beneficial. First, it provides a unified treatment during the whole model training, testing, and selection stages. Second, under Bayes' theorem and role of prior distributions, it is guaranteed that over-fitting will not occur. The reason is clear since the best model is chosen using a summation over all the hypothesis space instead of a single point estimate as in Maximum likelihood treatment. It should be noted that the use of regularization in ML approach is an attempt of having such behavior in the "frequentist" school of thought.

A typical probability distribution is a means to describe a finite-dimension random variable by assigning probability values to the possible outcomes of the variable in a principled manner. However, a stochastic process can be seen as a generalization of a probability distribution to functions. In Gaussian processes, benefiting from the ease of the computation of Gaussian distribution, all the inference and learning procedures will be developed in a systematic and straightforward fashion. Therefore, learning the distribution over function under Gaussian process framework will implicitly lead to map the input space to output space using the empirical data set in the supervised learning problems. In other words, the inductive problem is the transition from training data to a function that takes responsibility of making prediction over the entire input space.

There are two common but promising approaches to apply prior beliefs and assumptions to the characteristics expecting from the underlying function. The first is to put restriction on the class of functions that is going to be used [29]. For instance, the class of linear function may be chosen such that the problem is the difficulty to decide on the richness of the class of function. It is known that if the class is not rich enough to model the output, the predictions will not be sufficiently accurate. The attempt to increase the function's flexibility could potentially lead to over-fitting for which the model would exactly fit the data set very well, but the predictions on an unseen test set would be extremely inaccurate.

The second approach is to put prior probability on the function space and using the training data, increase the probability of the most likely functions for which the desired characteristics are achieved. For instance, the degree of smoothness of functions can determine the probability distribution over the function space. Despite avoiding over-fitting through prior distribution mechanism, the problem of dealing with uncountably infinite set of possible functions in finite time seems to be hugely problematic. However, under Gaussian process modeling approach this issue is principally addressed through the generalization of Gaussian probability distribution to Gaussian processes. While a probability distribution explains the random variables, a stochastic process governs the properties of functions. The outcome of the inference of Gaussian processes for finite training set by ignoring the remaining infinite data points in the input space will be the same as when all the possible infinite points are considered during the inference. Therefore, Gaussian process model is implemented for finite space of functions in practical applications and this would support the theoretical concept of uncountably infinite space of functions as it is further explained in [38], [6] through the concept of marginalization. It is worth noting that once the prior distribution is defined over the function space, its combination with the training dataset would give the posterior distribution over the functions.

In the Gaussian processes, different properties and characteristics representing the

class of the underlying functions could be expected through various covariance functions and their corresponding hyperparameters. The task of learning in Gaussian process is different from other learning algorithms, since learning for a nonparametric is no longer defined as finding the best weight configuration and values of the model. Therefore, the choice of a suitable covariance function along with the proper hyperparameter values is the key to the learning process.

In the following sections, the Gaussian process approach for the supervised learning problems for continuous outputs (regression) and discrete outputs (classification) will be explored.

### 2.2.1 Regression Modeling

It is intuitive to begin with Gaussian process framework in regression problems and extend the framework to classification problems. In regression problems, the mapping is from the input space to the continuous output space. There are two approaches to present the fundamental theory of Gaussian processes. One approach is through the weight-space viewpoint utilizing the straightforward treatment of standard linear regression while applying the idea of prior distribution. The posterior distribution will take form as the inference takes place. Interestingly enough, the concept of kernel will emerge as a result of mathematical manipulation at the end. The second approach is through function-space viewpoint which is completely coherent with the concept of stochastic processes. In this regard, by defining a prior distribution and put it on the whole function space, the inference and learning will be followed systematically and consequently the posterior process is resulted.

#### 2.2.1.1 Weight-space View

The model of linear combination of input values to achieve the output has been extensively studied in pattern recognition and machine learning literature [6]. The ease in implementation and interpretation is apparent although its limited capacity

to provide a flexible mapping from input space to output space is a serious burden. However, it will be shown that this limitation can be effectively removed through the introduction of kernel models.

#### 2.2.1.1.1 Bayesian Analysis of Linear Model

In this section, the linear model and the Bayesian treatment will be reviewed along with the derivation of the posterior and prediction distribution for the input space [38].

The training set is indicated by $D$ in which there are $n$ observations, $D = \{(x_i, y_i)|i = 1, \cdots, n\}$. $\mathbf{x}$ denotes the input vector of dimension D while $y$ denotes a scalar output in regression and target in classification cases. The design matrix $X$ is defined as a $D \times n$ matrix in which the column vector inputs $\mathbf{x}$ of the whole training set are aggregated. By collecting all the target values in the vector $\mathbf{y}$, the dataset can be written as $D = (X, \mathbf{y})$. The standard linear model is now defined as

$$f(x) = \mathbf{x}^T \mathbf{w}. \tag{2.1}$$

It is worth mentioning that the bias weight has been augmented in $\mathbf{x}$, and $\mathbf{w}$ is a vector of weights or parameters of the linear model. It is clear that $f$ is the function value. By considering Gaussian additive noise for the model, the observation target value $y$ is

$$y = f(x) + \varepsilon. \tag{2.2}$$

It is assumed that this noise is independently, identically distributed using the following Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{2.3}$$

The likelihood is the probability density of the observation given the parameters. Since the independence assumption has been made, the likelihood factors over the training dataset as follows

$$p(\mathbf{y}|X, \mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) = \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 I). \tag{2.4}$$

According to Bayesian formalism, a prior over parameters is needed. Prior plays the role of expressing one's beliefs about the parameters before any observation is made. In the following, a Gaussian prior over the parameters is considered

$$\mathbf{w} \sim \mathcal{N}(0, \Sigma_p). \tag{2.5}$$

As it is known, Bayes' theorem states that

$$posterior = \frac{likelihood \times prior}{marginal\ likelihood}.$$

Therefore, the parameter posterior probability distribution is

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}, \tag{2.6}$$

where the dominator is the marginal likelihood obtained through integration over parameter space

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w}). \tag{2.7}$$

Putting aside the marginal likelihood in Eq. (2.6), and following the approach of *maximum a posteriori* (MAP) for parameters estimation, the posterior distribution will have the form of a Gaussian of

$$p(\mathbf{w}|\mathbf{y}, X) \sim \mathcal{N}(\frac{1}{\sigma_n^2} A^{-1} X\mathbf{y}, A^{-1}), \tag{2.8}$$

$$A = \sigma_n^{-2} X X^T + \Sigma_p^{-1}. \tag{2.9}$$

It is clear that since the posterior has the form of a Gaussian distribution, the mean and mode are the same. In non-Bayesian "frequentist" treatment, the prediction observation value is the linear regression function $f$ using the test input $x_*$ and the parameter point estimate of maximum likelihood. However, in the Bayesian treatment, the predictive distribution is achieved through averaging the output of the linear model over the whole parameter space with respect to the obtained Gaussian posterior distribution as below

$$\begin{aligned} p(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|X, \mathbf{y}) d\mathbf{w} = \int \mathbf{x}_*^T \mathbf{w}\, p(\mathbf{w}|X, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma_n^2} \mathbf{x}_*^T A^{-1} X\mathbf{y},\ \mathbf{x}_*^T A^{-1} \mathbf{x}_*) \end{aligned} \tag{2.10}$$

#### 2.2.1.1.2 High-dimensional Feature Space Projection

Now that the concepts behind linear model, Bayesian analysis of putting a prior on the model's parameters along with the posterior and the prediction distribution have been introduced, the concept of high-dimensional feature space projection can be presented [38]. It is evident that the capacity of linear model is limited since not every mapping from input to output space can be recovered linearly. This can be alleviated through the projection of input space into a high-dimensional feature space, and the whole inference and learning is placed in this new space. It should be noted that since the nonlinearly projected functions are not dependent on the model parameters, the model is still regarded linear and therefore analytically tractable.

The mapping of the input space with D dimensions turning into a feature space with $N$ dimensions is denoted by $\phi(\mathbf{x})$. $\Phi(X)$ will indicate the column aggregation of $\phi(\mathbf{x})$ for all the samples in the training set. The model and all the analysis will remain the same as what it was mentioned in 2.2.1.1.1 but the difference is the substitution of $\mathbf{x}$ with $\phi(\mathbf{x})$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}. \tag{2.11}$$

It is clear that now the parameter vector $\mathbf{w}$ has length $N$. The prediction distribution will be reformulated as follows and based on some mathematical manipulations given in [38] the kernel function will show up naturally.

$$f_*|\mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi\mathbf{y},\ \phi_*^T A^{-1}\phi_*) \tag{2.12}$$

Where $\Phi = \Phi(X)$, $\phi(\mathbf{x}_*) = \phi_*$, and $A = \sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}$. Eq. 2.12 can be further manipulated based on the approach [38] such that $K = \Phi^T\Sigma_p\Phi$ and the prediction distribution become

$$\begin{aligned} f_*|\mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(&\phi_*^T\Sigma_p\Phi(K + \sigma_n^2 I)\mathbf{y}, \\ &\phi_*^T\Sigma_p\phi_* - \phi_*^T\Sigma_p\Phi(K + \sigma_n^2 I)^{-1}\Phi^T\Sigma_p\phi_*). \end{aligned} \tag{2.13}$$

The concept of *covariance function* or *kernel* of $k(.,.)$ can be introduced explicitly if $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}')$. It is obvious that $\phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}')$ is an inner product with respect to $\Sigma_p$. Since $\Sigma_p$ is positive definite, it can be written as $\Sigma_p = (\Sigma_p^{1/2})^2$, and consequently by defining $\psi(\mathbf{x}) = \Sigma_p^{1/2}\phi(\mathbf{x})$, the simple dot product representation $k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}).\psi(\mathbf{x}')$ will be derived.

The kernel substitution is utilized a lot in algorithms that extensively contain dot

products. The appearance of dot products in the input space can be replaced by the kernel value of $k(\mathbf{x}, \mathbf{x}')$. The main interest in algorithms such as Gaussian processes where the feature space is uncountably infinite or difficult to compute is in computing the kernel function rather than the feature value itself.

As shown previously, the weight-space view of the Gaussian process using Bayesian analysis of linear regression model with the feature space projection leads naturally to the application of the kernel trick. The intrinsic approach of defining prior distribution over function space will be investigated and consequently Gaussian process model will be presented more thoroughly in the next section.

#### 2.2.1.2 Function-space View

Following the standard linear regression model of the previous section of $f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}$ with prior $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$, a simple Gaussian process can be obtained having the mean and covariance of [38]

$$\mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}] = 0, \tag{2.14}$$

$$\mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] = \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}\mathbf{w}^T]\phi(\mathbf{x}') = \phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}'). \tag{2.15}$$

Therefore, $f(\mathbf{x})$ and $f(\mathbf{x}')$ jointly have zero mean and covariance $\phi(\mathbf{x})^T\Sigma_p\phi(\mathbf{x}')$ of a Gaussian distribution. Considering the desire to evaluate $f(\mathbf{x})$ at specific values of $\mathbf{x}$ of the training set (i.e. $\mathbf{x}_1, \cdots, \mathbf{x}_n$), the ultimate interest is in the joint distribution of the function values $f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)$, which is denoted by $\mathbf{f}$. Thus, the linear model for the entire dataset is

$$\mathbf{f} = \Phi^T\mathbf{w}. \tag{2.16}$$

It is demonstrated that the mean and covariance of the Gaussian process will be

$$\mathbb{E}[\mathbf{f}] = \Phi^T \mathbb{E}[\mathbf{w}] = 0, \tag{2.17}$$

$$\text{cov}[\mathbf{f}] = \Phi^T \mathbb{E}[\mathbf{w}\mathbf{w}^T]\Phi = \Phi^T \Sigma_p \Phi = K. \tag{2.18}$$

K is the Gram matrix with elements of the covariance or kernel function

$$K_{rs} = k(\mathbf{x}_r, \mathbf{x}_s) = \text{cov}(f(\mathbf{x}_r), f(\mathbf{x}_s)) = \phi(\mathbf{x}_r)^T \Sigma_p \phi(\mathbf{x}_s). \tag{2.19}$$

The previous derivations provide an interesting insight into the Gaussian process model. As noted previously, this definition implies the consistency requirement or marginalization property for the stochastic process $f(\mathbf{x})$ [6], [38]. In other words, the joint probability distribution for any finite set can be achieved in a consistent manner. Another point about Gaussian stochastic process is that the second-order statistics are enough to specify the joint distribution of a set of samples. Since there is no prior knowledge about the mean of the process, it is typically assumed to be zero. The only specification of a Gaussian process will be its covariance function as a result. For a pair of data points, this will be given by Eq. 2.19, and for the whole training set by the Gram matrix of K. The choice of covariance function depends on the notion of similarity in the context of the problem, and fortunately Gaussian process framework provides a mechanism to decide upon the best covariance function. One instance of covariance function is the squared exponential (SE) given below

$$\text{cov}(f(\mathbf{x}_r), f(\mathbf{x}_s)) = k(\mathbf{x}_r, \mathbf{x}_s) = \exp(-\frac{1}{2}|\mathbf{x}_r - \mathbf{x}_s|^2). \tag{2.20}$$

It is shown in [38] that there exists a possibly infinite expansion in terms of basis functions for every positive definite covariance function which results from Mercer's Theorem. For instance, SE covariance function can be derived from the linear combination of the infinite Gaussian-shaped basis functions. Covariance functions will be particularly explained further in section 2.2.3.

Now that the Gaussian process prior is defined using the concept of the covariance function, back to the linear regression model introduced in section 2.2.1.1, using the additive Gaussian noise, the observation model can be indicated as

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_n^2 I), \tag{2.21}$$

using the Gaussian process where the marginal distribution $p(\mathbf{f})$ is a Gaussian with zero mean and a covariance defined by a Gram matrix K

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|0, K). \tag{2.22}$$

Based on the general rule of the linear-Gaussian model and Bayes' theorem derived in [6], the marginal distribution of the noisy observation $\mathbf{y}$ becomes

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\mathbf{y}|0, C) \tag{2.23}$$

where C is the covariance given by

$$\mathrm{cov}(y_r, y_s) = k(\mathbf{x}_r, \mathbf{x}_s) + \sigma_n^2 \delta_{rs}, \tag{2.24}$$

$$\mathrm{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I, \tag{2.25}$$

where $\delta$ is a Kronecker delta. Until now, a model of joint distribution for a data set has been defined using Gaussian process perspective. However, the ultimate goal in regression problems is to predict the continuous target value of a new test sample $\mathbf{x}_*$. The predictive distribution of $p(y_*|\mathbf{y}_n)$ is required. Where $\mathbf{y}_{n+1}$ denotes vector $(y_1, \cdots, y_n, y_{n+1})^T$, the derivation begins first by defining the joint distribution of $p(\mathbf{y}_{n+1})$ over the whole training set including the new test sample as follows

$$p(\mathbf{y}_{n+1}) = \mathcal{N}(\mathbf{y}_{n+1}|0, C_{n+1}). \tag{2.26}$$

Considering the joint distribution is Gaussian, the rules to achieve a conditional distribution out of the joint distribution derived in [6] can be used. First, the joint covariance is demonstrated as below in a blocked-form matrix

$$C_{n+1} = \begin{bmatrix} C_n & \mathbf{k} \\ \mathbf{k} & c \end{bmatrix}. \tag{2.27}$$

$C_n$ indicates the $n \times n$ covariance matrix given by Eq. 2.25. Respectively, the vector $\mathbf{k}$ contains elements of $k(\mathbf{x}_m, \mathbf{x}_{n+1})$ for $m = 1, \cdots, n$, while $c = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1}) + \sigma_n^2$. Ultimately, the predictive distribution for the Gaussian process regression can be derived as follows

$$p(y_*|\mathbf{y}) = \mathcal{N}(y_*|m(\mathbf{x}_*), \mathrm{cov}(\mathbf{x}_*)), \tag{2.28}$$

$$m(\mathbf{x}_*) = \mathbf{k}^T C_n^{-1} \mathbf{y}, \tag{2.29}$$

$$\mathrm{cov}(\mathbf{x}_*) = c - \mathbf{k}^T C_n^{-1} \mathbf{k}. \tag{2.30}$$

These equations are the key results of Gaussian process regression. Since $\mathbf{k}$ is a function of the test sample $\mathbf{x}_*$, it is apparent that the mean and covariance of the predictive Gaussian distribution depend on the test sample. Furthermore, the marginal likelihood or evidence $p(\mathbf{y}|X)$ can be defined here. Previously, in Eq. 2.23 it is derived that

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X) d\mathbf{f} = \mathcal{N}(\mathbf{y}|0, K + \sigma_n^2 I). \tag{2.31}$$

Therefore, it can be stated explicitly as

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi. \tag{2.32}$$

Until now, the whole Gaussian process framework along with relative derivations and procedures is presented. The Gaussian process regression will be the base of the derivations given in the next section. It is evident that the Gaussian process

classification is a discriminative approach to supervised learning. It will be shown that a response function on the Gaussian process regression model to achieve the probability values for the discrimination task is required.

### 2.2.2 Classification Modeling

Learning problems with continuous output values are investigated and the Gaussian process regression model is introduced in section 2.2.1.1. The nonparametric Bayesian treatment of Gaussian process model for classification problems in supervised learning will be studied in this section. Obviously, the focus is on probabilistic approaches to obtain the class probabilities rather than using other methods to provide a "guess" for the class labels. However, through the integration of prediction probabilities with the zero-one loss function in decision theory, the result of class label decisions will be the same as class label guesses at the end. For complicate nonlinear loss functions or consideration of reject option, computation of prediction probabilities using a probabilistic model is necessary.

The bright side of Gaussian process regression is that the whole derivation and computation is analytically tractable. This stems from the fact that the likelihood function is a Gaussian distribution. Consequently, the combination of a Gaussian process prior with a Gaussian likelihood function results in a Gaussian posterior process. Due to discrete class labels, a Gaussian likelihood is definitely inappropriate for classification problems; therefore, methods of approximate inference and learning will be used considering the infeasiblity of exact inference.

The same approach of generalizing the linear regression model to Gaussian process regression will be followed for the classification problems. Similarly, using the Gaussian process prior, the well-known linear logistic regression will be generalized to Gaussian process classification (GPC). Laplace approximation method will be covered in detail due to the need of using approximate methods for inference and learning procedures.

#### 2.2.2.1 Discriminative or Generative Approach to Classification

The main goal in classification learning methods is to find the joint probability $p(y, \mathbf{x})$, where $y$ is the class label and $\mathbf{x}$ the input vector. According to Bayes' theorem and the product rule, this joint probability can be written either as $p(y)p(\mathbf{x}|y)$ or $p(\mathbf{x})p(y|\mathbf{x})$. The first approach is called generative approach. By using prior probabilities of each class $p(y)$ and class-conditional probability $p(\mathbf{x}|y)$, the posterior probability of each class can be computed accordingly as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{\sum_{m=1}^{C} p(C_m)p(\mathbf{x}|C_m)}. \tag{2.33}$$

The second approach is called discriminative, since the posterior is modeled directly without looking at modeling class-conditional probability. It is evident that modeling two main distributions either $p(\mathbf{x}|y)$ or $p(y|\mathbf{x})$ can be approached using either parametric or nonparametric models. In the parametric case, the class-conditional distribution can be modeled by using Gaussian density distribution and then in a Bayesian treatment, by putting hyperprior on the hyperparameters, the best values can be obtained. However, it should be noted that this approach is prone to mistake since Gaussian modeling of the class-conditional might be inappropriate.

In the nonparametric case, the simple idea of transforming a regression model output into a class probability values using a proper response function can be considered. The purpose of the response function is to squash the regression function domain into $[0, 1]$ in which a valid probabilistic interpretation is promised [38]. Two possible choices of such response functions exist. The first is logistic function which can be written in the case of binary discrimination where either $y = 1$ or $y = 0$ as follows

$$p(y = 1|\mathbf{x}) = \lambda(\mathbf{x}^T\mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}^T\mathbf{w})}. \tag{2.34}$$

The other choice is to use the cumulative density function of a standard normal

distribution known as inverse probit function as given below

$$p(y = 1|\mathbf{x}) = \int_{-\infty}^{\mathbf{x}^T\mathbf{w}} \mathcal{N}(z|0, 1)dz. \tag{2.35}$$

Deciding whether to apply discriminative or generative approach to a problem has been challenging in pattern recognition and machine learning literature. Although both ways of expressing the joint distribution p(y,$\mathbf{x}$) are correct, the true insight into the problem to choose the correct form is eminent. In cases that only classification is required, modeling $p(y|\mathbf{x})$) is superfluous. In cases where the problems of dealing with missing input values, outliers, and unlabeled data samples is essential, they can be addressed in a principled fashion through the generative approach by deriving $p(\mathbf{x}) = \sum_y p(y)p(\mathbf{x}|y)$. The Gaussian process classification model studied in this chapter is approached in the discriminative fashion [38].

#### 2.2.2.2 Weight-Space Perspective to Discriminative Linear Classifiers

The concepts behind Gaussian process classifiers will be explored in this section [38]. First, the binary classification approach is presented in the beginning. Later, the extension to multi-class classification approach will be explained. It is obvious the likelihood function is given by Bernoulli distribution in binary treatment as follows

$$p(y|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w})^t(1 - \sigma(\mathbf{x}^T\mathbf{w}))^{1-t}, \tag{2.36}$$

where $\mathbf{w}$ is the weight vector and $\sigma(z)$ can be any sigmoid function. In the case of logistic function $\sigma(z) = \lambda(z)$, the model is called logistic regression. Whereas, in the case of inverse probit function of $\sigma(z) = \Phi(z)$, it is called probit regression.

According to the rule of probability stating the sum of probabilities of two classes in binary classification must be 1, the equation can be written such that $p(y = -1|\mathbf{x}, \mathbf{w}) = 1 - p(y = +1|\mathbf{x}, \mathbf{w})$. If the common literature convention for binary classification to represent labels as $y = +1$ and $y = -1$ is followed, then the likelihood

for samples with class labels of $y_i = +1$ is given by $\sigma(\mathbf{x}_i^T\mathbf{w})$. On the other hand, for samples with $y_i = -1$, the likelihood is expressed by $1 - \sigma(\mathbf{x}_i^T\mathbf{w})$. Since the so far introduced sigmoid functions are both symmetric, where $\sigma(-z) = 1 - \sigma(z)$, the likelihood can be written as:

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \sigma(y_i f_i). \tag{2.37}$$

Having a data set of $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \cdots, n\}$, the definition of a Gaussian prior on the weight space is given by $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$. Then the un-normalized log posterior can be denoted by

$$\log p(\mathbf{w}|X, \mathbf{y}) \propto -\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w} + \sum_{i=1}^{n} \log \sigma(y_i f_i) + C, \tag{2.38}$$

where $C$ is constant which is not dependent on $\mathbf{w}$. As the posterior distribution's parameters in linear regression model derived in Eq. 2.8 denote, the probability distribution is modeled analytically by using a Gaussian distribution with respective mean and variance. However, since the likelihood of sigmoid function is non-Gaussian, the posterior distribution would not have a simple analytical form for classification.

For some sigmoid functions such as logistic and inverse probit functions, the log-likelihood is a concave function with respect to $\mathbf{w}$. The quadratic penalty is concave too as Eq. 2.38 denotes. Thus, the whole log posterior is a concave function, allowing the maximum to be found easily. Newton's method is the widespread optimization method to find the maximum in such situations. It is stated in [26] that the other name of this algorithm is iteratively reweighed least squares (IRLS). This is evident that the approach to find the best point estimate of parameters is not fully Bayesian. This is known as maximum a posteriori(MAP) in the Bayesian community and penalized maximum likelihood among the frequentist school of thought.

To make predictions for a test sample $\mathbf{x}_*$ while using the training set $\mathcal{D}$, the

predictive probability can be expressed as

$$p(y_* = +1|x_*, \mathcal{D}) = \int p(y_* = +1|\mathbf{w}, \mathbf{x}_*)p(\mathbf{w}|\mathcal{D})d\mathbf{w}. \tag{2.39}$$

It is clear that the likelihood function becomes $p(y_* = +1|\mathbf{w}, \mathbf{x}_*) = \sigma(\mathbf{x}_*^T\mathbf{w})$. This integration is the essence of Bayesian prediction. It should be noted that the likelihood function in the multi-class classification problem is represented by either the softmax function or multinomial probit function. The softmax function is denoted by

$$p(y = C_m|x, W) = \frac{\exp(\mathbf{x}^T\mathbf{w}_m)}{\sum_{m'} \exp(\mathbf{x}^T\mathbf{w}_{m'})}, \tag{2.40}$$

where $\mathbf{w}_m$ is the weight vector of class $m$, $\mathbf{w}_{m'}$ is the weight vector of all classes except $m$, and all weight vectors are aggregated in the big matrix $W$. Correspondingly, the concave log-likelihood for multi-class problem can be written as $\sum_{i=1}^{n}\sum_{m=1}^{C}\delta_{m,y_i}[\mathbf{x}_i\mathbf{w}_m - \log(\sum_{m'}\exp(\mathbf{x}_i^T\mathbf{w}_{m'}))]$. It is clear that for the labeling scheme, a vector of length $C$ containing all zeros but one corresponding to the class that data samples belongs to is employed.

#### 2.2.2.3 Gaussian Process Classification

The regular Bayesian treatment of the linear model using a prior on the weight-space is given in the previous section. The transition from weight-space to function-space in linear regression to define Gaussian process regression (GPR) is explained in sec. 2.2.1. Similarly, the Gaussian process classification (GPC) will be approached based on the foundation given in section 2.2.2.2 for the classification problems using linear logistic and probit regression models

To begin, the binary classification will be considered by looking at the Gaussian process to define a prior over the latent function $f(\mathbf{x})$. This would lead to the use of a sigmoid function such as the logistic function or the inverse probit function to squash the latent function to reach a prior on $p(y = +1|\mathbf{x}) = \sigma(f(\mathbf{x}))$. Since $f$ is

stochastic and $p(y = +1|\mathbf{x})$ is a deterministic function of $f$, it can be concluded that $p(y = +1|\mathbf{x})$ is stochastic too.

It should be noted that the latent function $f$, used as a nuisance function, serves to provide a convenient formulation of the model. It will be integrated out later during the derivation and computation. It is clear that a nuisance variable is not observed and particularly interesting to obtain. It has a mean to carry on the entire derivation using the observed inputs $X$ and class labels $\mathbf{y}$ to obtain $p(y = +1|\mathbf{x})$.

The inference in binary classification using a single-latent Gaussian process model can be explained in two different stages [38]. In the back stage, the goal is the derivation of the prediction probability as follows

$$p(y_* = +1|X, \mathbf{y}, \mathbf{x}_*) = \int p(y_* = +1|f_*)p(f_*|X, \mathbf{y}, \mathbf{x}_*)df_*.$$

It is obvious that $p(y_* = -1|X, \mathbf{y}, \mathbf{x}_*) = 1 - p(y_* = +1|X, \mathbf{y}, \mathbf{x}_*)$ based on the Bernoulli distribution Eq. 2.36, $p(y_* = +1|f_*) = \sigma(f_*)$. Therefore, the prediction probability becomes

$$p(y_* = +1|X, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*)p(f_*|X, \mathbf{y}, \mathbf{x}_*)df_*. \tag{2.41}$$

It is evident that for certain sigmoid activation functions, the integral is analytically intractable, although in the binary case, simple numerical techniques can be sufficiently used due to the one dimensional space integration. For instance, the use of logistic function makes the integration intractable. On the contrary, inverse probit function can be analytically computed. In the front stage, the second term in the integration 2.41 is required to be investigated, namely the distribution of the latent variable for a test sample $\mathbf{x}_*$. Through the use of Bayes' theorem, it can be manipulated as below

$$
\begin{aligned}
p(f_*|X,\mathbf{y},\mathbf{x}_*) &= \int p(f_*,\mathbf{f}|X,\mathbf{y},\mathbf{x}_*)d\mathbf{f} \\
&= \frac{1}{p(\mathbf{y}|X)}\int p(f_*,\mathbf{f}|X,\mathbf{x}_*)p(\mathbf{y}|f_*,\mathbf{f},X,\mathbf{x}_*)d\mathbf{f} \\
&= \frac{1}{p(\mathbf{y}|X)}\int p(f_*|\mathbf{f},X,\mathbf{x}_*)p(\mathbf{f}|X)p(\mathbf{y}|\mathbf{f},X)d\mathbf{f} \\
&= \int p(f_*|\mathbf{f},X,\mathbf{x}_*)p(\mathbf{f}|X,\mathbf{y})d\mathbf{f}. \qquad (2.42)
\end{aligned}
$$

The non-Gaussian likelihood has made the whole integration of Eq. 2.42 analytically intractable in the front stage analysis too. Consequently, approximations are needed for practical application of GPC.

As the derivation for the front stage of inference denotes, the conditional distribution of the predicted latent value given all the training latent values $p(f_*|\mathbf{f},X,\mathbf{x}_*)$ can be obtained using the GPR results of Eq. 2.29 and 2.30 as follows

$$
p(f_*|\mathbf{f}) = \mathcal{N}(f_*|\mathbf{k}^T C_n^{-1}\mathbf{y}, c - \mathbf{k}^T C_n^{-1}\mathbf{k}) \qquad (2.43)
$$

By searching for the Gaussian approximation of the posterior distribution of $p(\mathbf{f}|X,\mathbf{y})$, the standard result for the convolution of two Gaussian distributions can be exploited.

Two approaches can be used to resolve this intractability. One of them is based on Monte Carlo sampling methods [31] while the other is based on the analytical approximation methods. The latter methods try to approximate the non-Gaussian posterior with a Gaussian distribution so that the integration could be resolved through the technique of convolution of Gaussian distributions. *Laplace approximation* method found in [51] is the most straightforward and promising one. This method will be explained in detail in the next section. Another technique is the *expectation propagation* (EP) [34], [28]. Technical details and implementation algorithm can be found in [38]. Once the distribution of the predicted latent value has been found, the integration of the sigmoid function with the approximate Gaussian will be explicitly derived.

#### 2.2.2.3.1 Laplace Approximation for Single-Latent GPC

Laplace Approximation is a method that tries to approximate the posterior distribution $p(\mathbf{f}|X,\mathbf{y})$ with a Gaussian distribution $q(\mathbf{f}|X,\mathbf{y})$ [38]. Using a second order Taylor expansion of $\log p(\mathbf{f}|X,\mathbf{y})$ around its maximum, the approximate Gaussian distribution will be found as

$$q(\mathbf{f}|X,\mathbf{y}) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1}) \propto \exp(-\frac{1}{2}(\mathbf{f}-\hat{\mathbf{f}})^T A(\mathbf{f}-\hat{\mathbf{f}}), \tag{2.44}$$

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} p(\mathbf{f}|X,\mathbf{y}), \tag{2.45}$$

$$A = -\bigtriangledown\bigtriangledown \log p(\mathbf{f}|X,\mathbf{y})|_{\mathbf{f}=\hat{\mathbf{f}}}, \tag{2.46}$$

where the covariance is approximated through the second derivative or Hessian of the negative log posterior as the maximum point. It is evident that the mode and mean in Gaussian distribution fall on each other.

First, the Gaussian approximation to the posterior distribution will be given and then, the result of the convolution will lead to the analytical Gaussian approximate to the integral of Eq. 2.42. Subsequently, the integral of Eq. 2.41 will be treated using some approximation for logistic sigmoid function and analytic solution for inverse probit function. Lastly, the marginal likelihood will also be approximated using Laplace method since it is important for the model hyperparameter learning.

##### 2.2.2.3.1.1 Posterior Distribution

From Bayes' theorem [38], the posterior distribution can be written as below

$$p(\mathbf{f}|X,\mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X). \tag{2.47}$$

Since the normalization factor is independent of $\mathbf{f}$ and the computation is with respect to $\mathbf{f}$, the un-normalized posterior can be considered for further computation. From Eq. 2.47, it is evident that the first term $p(\mathbf{y}|\mathbf{f})$ is the likelihood function and represented

by one of the aforementioned sigmoid functions. Meanwhile, the second term $p(\mathbf{f}|X)$ is the famous zero-mean Gaussian process with the covariance matrix $C_n$ . Therefore, the log-posterior can be given by the quantity

$$\begin{aligned}\Psi(\mathbf{f}) &= \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|X) \\ &= \log p(\mathbf{y}|\mathbf{f}) + -\frac{1}{2}\mathbf{f}^T C_n^{-1}\mathbf{f} - \frac{1}{2}\log|C_n| - \frac{n}{2}\log 2\pi. \end{aligned} \tag{2.48}$$

The first derivative of $\Psi(\mathbf{f})$ with respect to $\mathbf{f}$ is needed to find the mode of the posterior. Since the likelihood function depends nonlinearly on $\mathbf{f}$, it is not possible to simply find the mode by setting the first derivative to zero. Thus, some iterative optimization approach based on Newton-Raphson method has to be taken into consideration. Iterative reweighed least square(IRLS) is one common method [6]. Therefore, the second derivative of $\Psi(\mathbf{f})$ will be needed. It is worth mentioning that the Laplace method itself needs to address the variance of the approximate Gaussian using Hessian matrix, which is the negative of the second derivative

$$\bigtriangledown\Psi(\mathbf{f}) = \bigtriangledown \log p(\mathbf{y}|\mathbf{f}) - C_n^{-1}\mathbf{f}, \tag{2.49}$$

$$\bigtriangledown \bigtriangledown\ \Psi(\mathbf{f}) = \bigtriangledown \bigtriangledown \log p(\mathbf{y}|\mathbf{f}) - C_n^{-1} = -W - C_n^{-1}, \tag{2.50}$$

where $W = \bigtriangledown \bigtriangledown \log p(\mathbf{y}|\mathbf{f})$ is a diagonal matrix. The big matrix $W$ could be created due to the fact that the likelihood function factorizes over training samples. The first and second derivatives of the log-likelihood $\log p(y_i|f_i)$ can be derived for the log of the logistic function $-\log(1+\exp(-y_i f_i))$ according to [38] as follows

$$\frac{\partial}{\partial f_i}\log p(y_i|f_i) = t_i - \pi_i, \tag{2.51}$$

$$\frac{\partial^2}{\partial f_i^2}\log p(y_i|f_i) = -\pi_i(1-\pi_i), \tag{2.52}$$

where $\pi_i = p(y_i = +1|f_i)$ and $\mathbf{t} = (\mathbf{y}+1)/2$. Additionally, The first and second derivatives of the log-likelihood $\log p(y_i|f_i)$ can be derived for the log of the inverse

probit function $\log \Phi(y_i f_i)$) according to [38] as follows

$$\frac{\partial}{\partial f_i} \log p(y_i|f_i) = \frac{y_i \mathcal{N}(f_i)}{\Phi(y_i f_i))}, \tag{2.53}$$

$$\frac{\partial^2}{\partial f_i^2} \log p(y_i|f_i) = -\frac{\mathcal{N}(f_i)^2}{\Phi(y_i f_i))^2} - \frac{y_i f_i \mathcal{N}(f_i)}{\Phi(y_i f_i))}. \tag{2.54}$$

Now that the first and second derivatives of the log-likelihood are derived, the maximum value of the posterior will be achieved through initiating Newton's method and iterate until convergence to the mode [38]

$$\begin{aligned} \mathbf{f}^{new} = \mathbf{f} - (\bigtriangledown \bigtriangledown \Psi)^{-1} \bigtriangledown \Psi &= \mathbf{f} + (C_n^{-1} + W)^{-1}(\bigtriangledown \log p(\mathbf{y}|\mathbf{f}) - C_n^{-1}\mathbf{f}) \\ &= (C_n^{-1} + W)^{-1}(W\mathbf{f} + \bigtriangledown \log p(\mathbf{y}|\mathbf{f})). \end{aligned} \tag{2.55}$$

Obviously upon convergence, the mode $\hat{\mathbf{f}}$ of the posterior will be found while demonstrating that the gradient will be zero. Thus, it can be written simply in the self-consistent equation that

$$\hat{\mathbf{f}} = C_n(\bigtriangledown \log p(\mathbf{y}|\hat{\mathbf{f}})). \tag{2.56}$$

Having found the mode, the Hessian matrix can be evaluated simply at the mode $\hat{\mathbf{f}}$.

$$A = - \bigtriangledown \bigtriangledown \Psi(\mathbf{f}) = W + C_n^{-1}. \tag{2.57}$$

Finally, the Gaussian approximation to the posterior distribution $p(\mathbf{f}|\mathbf{y})$ can be defined as

$$q(\mathbf{f}|X, \mathbf{y}) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1}). \tag{2.58}$$

Now that the Gaussian approximate of the posterior is found, using the posterior Gaussian process denoted in Eq. 2.43, the Gaussian approximate to the integral of

Eq. 2.42 for the front stage inference will be given by the mean and variance of [6]

$$\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*] = \mathbf{k}^T C_n^{-1} \hat{\mathbf{f}} = \mathbf{k}^T \bigtriangledown \log p(\mathbf{y}|\hat{\mathbf{f}}) \tag{2.59}$$

$$\text{var}_q[f_*|X, \mathbf{y}, \mathbf{x}_*] = c - \mathbf{k}^T (C_n + W^{-1})^{-1} \mathbf{k}. \tag{2.60}$$

Until now, the posterior distribution of latent function for the test sample has been successfully approximated. The next step is the inference at the back end stage according to Eq. 2.41 which all will be followed in the next section.

#### 2.2.2.3.1.2 Predictive Distribution

The back end stage of inference for GPC is the approximation to the integration of

$$\bar{\pi}_* \sim \mathbb{E}_q(\pi_*|X, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) q(f_*|X, \mathbf{y}, \mathbf{x}_*) df_*. \tag{2.61}$$

Two approaches to approximate this integration need to be distinguished from one another [38]. The first approach is the MAP prediction where it tries to give the solution for Eq. 2.61 through the sigmoid of the expectation of $f_*$, namely $\bar{\pi}_* = \sigma(\mathbb{E}_q(\pi_*|\mathbf{y}))$. It should be noted that due to the nonlinearity of the sigmoid function, the true reliable solution is given through the averaged predictions. Therefore, this integral has to be approximated in some fashion. However, according to [5] the predicted class labels for the test samples are the same for both averaged prediction method and MAP for the special case of binary classification.

Based on the nature of the problem, it is unnecessary to carry the burden of computing the averaged predictions when the class label prediction is the ultimate goal. However, if the confidence of the predictions is needed for the case reject option considered, then the averaged prediction is essential. Previously mentioned, either sampling or analytical approximation methods can be employed.

One approximation method is given in [51]. This method uses the idea that the logistic function $\lambda(x)$ is the cumulative density function of the probability density

function $p(x) = \text{sech}^2(x/2)/4$, known as logistic or sech-squared distribution. If $p(x)$ is approximated by mixtures of Gaussian distributions, then the logistic function can be approximated by a linear combination of error functions.

The other method is based on the close similarity between logistic function and inverse probit function [25]. This method proceeds by proposing the idea of horizontally rescaling the logistic function in order to have the same slope as inverse probit function does at the origin. This is given by defining the scale factor $\nu^2 = \pi/8$.Thus $\lambda(x)$ is approximated by $\Phi(\nu x)$. It is clear that the benefit of following this approximation is that the integration Eq. 2.61 can be analytically obtained through the convolution of two Gaussian distributions and ultimately expressed by another inverse probit function.

$$\int \Phi(\nu x)\mathcal{N}(x|\mu, \sigma^2)dx = \Phi\left(\frac{\mu}{(\nu^{-2} + \sigma^2)^{1/2}}\right). \tag{2.62}$$

Therefore, the application of the approximation $\lambda(x) \simeq \Phi(\nu x)$ to the inverse probit function on both sides of Eq. 2.62 leads to the approximation to the integration Eq. 2.61 as follows

$$\int \lambda(x)\mathcal{N}(x|\mu, \sigma^2)dx \simeq \lambda\left(\kappa(\sigma^2)\mu\right), \tag{2.63}$$

$$\kappa(\sigma^2) = \left(1 + \pi\sigma^2/8\right)^{-1/2}. \tag{2.64}$$

Now that the approximation is achieved, by applying results of predicted posterior mean 2.59 and predicted posterior variance 2.60, the back end approximation is given as

$$\bar{\pi}_* = \lambda\left(\kappa(f_*|\mathbf{y})\mathbb{E}_q[f_*|X, \mathbf{y}, \mathbf{x}_*]\right), \tag{2.65}$$

$$\kappa(\sigma^2) = (1 + \pi \text{ var}_q[f_*|X, \mathbf{y}, \mathbf{x}_*]/8)^{-1/2} \tag{2.66}$$

At this point, Laplace approximations for both the posterior and predictive distributions for binary classification have been expressed. The inference stage in GPC can be wrapped up at this point. The learning stage will be given in the next section. To learn the hyperparameters of the model, the marginal likelihood in needed. It will be described that in the similar manner, the Laplace approximation to the marginal likelihood can be achieved.

#### 2.2.2.3.1.3 Marginal Likelihood

It is important to compute the marginal likelihood [38] to conduct the hyperparameter learning in GPC

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)d\mathbf{f} = \int \exp(\Psi(\mathbf{f}))d\mathbf{f}. \tag{2.67}$$

Following the same approach as previous sections, the Laplace method proceeds by defining the second order Taylor expansion of $\Psi(\mathbf{f})$ locally around the mode $\hat{\mathbf{f}}$. By defining $\Psi(\mathbf{f}) \simeq \Psi(\hat{\mathbf{f}}) - \frac{1}{2}(\mathbf{f}-\hat{\mathbf{f}})^T A(\mathbf{f}-\hat{\mathbf{f}}))$, the approximation $q(\mathbf{y}|X)$ will be expressed as [6]

$$\begin{aligned} p(\mathbf{y}|X) \simeq q(\mathbf{y}|X) &= \exp(\Psi(\hat{\mathbf{f}})) \int \exp(-\frac{1}{2}(\mathbf{f}-\hat{\mathbf{f}})^T A(\mathbf{f}-\hat{\mathbf{f}}))d\mathbf{f} \\ &= \exp(\Psi(\hat{\mathbf{f}}))\frac{(2\pi)^{n/2}}{|A|^{1/2}} \end{aligned} \tag{2.68}$$

Therefore the log-marginal likelihood will take the following form

$$\begin{aligned} \log q(\mathbf{y}|X,\theta) &= \Psi(\hat{\mathbf{f}}) - \frac{1}{2}\log|W + C_n^{-1}| + \frac{n}{2}\log(2\pi), \\ &= -\frac{1}{2}\hat{\mathbf{f}}^T C_n^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log|W + C_n^{-1}|. \end{aligned} \tag{2.69}$$

It is obvious that $\theta$ contains the entire hyperparameter set. Using this approximation to the marginal likelihood, various learning procedures such as MAP estimate can be utilized to optimize the hyperparameters and find the maximum values.

#### 2.2.2.3.2 Other Approximation and Sampling Methods

Expected propagation(EP) is another analytical approximation, similar to the Laplace method, follows the idea of Gaussian approximation to the posterior probability in the GPC model. Since the likelihood function is non-Gaussian, for either probit or logistic functions, the integration 2.42 will be intractable. Approximating the likelihood $p(y_i|f_i)$ by a local likelihood function, predictive probability and marginal likelihood can be respectively computed [38]. In the case of EP for GPC, the local likelihood approximation is an unnormalized Gaussian function of the latent variable $f_I$.

The other approach to tackle this intractability problem is to use sampling methods. Methods, such as the Markov Chain Monte Carlo (MCMC), attempt to bypass the integration intractability in posterior distribution. Unlike the two other methods, they are not categorized as an analytical approximation considering they theoretically guarantee accurate answer to the intractable integration. However, the long-run specification of sampling methods, make them practically unfavorable.

This section explained the foundation of Gaussian process classification method. The distinction between discriminative and generative approaches is clearly made. Afterwards, the linear logistic regression model is introduced and the Bayesian analysis to define prior distribution on the weight parameters is given. Based on this model, the extension to the Gaussian process classification is explained. The idea of approximation and sampling methods and the reason they are necessary to use is developed, and the Laplace approximation method for binary classification is described in details. The way the posterior, predictive, and marginal likelihood are analytically approximated is covered in details. Other approximate and sampling methods are briefly introduced in the previous section.

### 2.2.3 Covariance Function

Insights from previous sections reveal that the role of the covariance function in GPC is of high importance while defining the prior on the class of functions which are sought. It is intuitive to say the measure of similarity between the two points are crucial and that when the training samples similar to the test samples has high similarity, they will be more informative to help predict the test samples' class label.

A valid covariance function must be positive semidefinite. Given the set of training samples, the Gram matrix K is defined as $K_ij = k(\mathbf{x}_i, \mathbf{y}_j)$. The matrix K is called covariance matrix since $k$ defines the covariance function. Covariance functions have some particular properties. For instance they are symmetric $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. A stationary covariance function defines as a function that is translation invariant in the input space. They are a function of $\mathbf{x} - \mathbf{x}'$. Radial basis functions (RBF) or generally homogeneous kernels are the ones that are only dependent on the magnitude of the distance of the two input arguments. Thus they are a function of $||\mathbf{x} - \mathbf{x}'||$ [6]. In the following, some examples of covariance functions along with their properties will be provided as the ways to construct new kernels from old ones will be explained at the end.

#### 2.2.3.1 Stationary Covariance Functions

A stationary covariance function is defined as a function of $\tau = \mathbf{x} - \mathbf{x}'$. So, $k$ can be written as a single argument of $k(\tau)$. The isotropic stationary covariance function can be defined when it is a function of $r$ where $r = |\tau|$. In the following, some examples of common isotropic covariance function will be given, and their properties will be counted respectively [38]. Clearly, the covariance function will be introduced in a normalized formed, such that $k(0) = 1$. Any desired process variance can also be obtained through multiplication of the function value with a positive constant factor $\sigma_f^2$.

#### 2.2.3.1.1 Squared Exponential Covariance Function

The Squared exponential (SE) covariance function has the following form

$$k_{SE}(r) = \exp(-\frac{r^2}{2l^2}), \tag{2.70}$$

where $l$ denotes the characteristic length-scale. It is clear that there can be a specific length-scale for each input dimension. This is infinitely differentiable, meaning the GP using this covariance function has a mean square derivatives of all orders and therefore it is very smooth. It is shown in [38] that if the input $\mathbf{x}$ is projected into a feature space defined by Gaussian-shaped basis functions centered in $\mathbf{x}$-space, then the SE covariance function can be naturally derived.

#### 2.2.3.1.2 The Matern Class of Covariance Function

The Matern class of covariance function has the form of

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}(\frac{\sqrt{2\nu}r}{l})^{\nu} K_{\nu}(\frac{\sqrt{2\nu}r}{l}), \tag{2.71}$$

where the parameter $\nu$ governs the smoothness of the process and $K_{\nu}$ is a modified Bessel function [1]., The covariance function can be represented simply by using half integer values for $\nu$ . For instance, the Matern covariance function with $\nu = 3/2$ and $\nu = 5/2$ can be represented as below

$$k_{\nu=3/2}(r) = (1 + \sqrt{3}r)\exp(-\sqrt{3}r), \tag{2.72}$$

$$k_{\nu=5/2}(r) = (1 + \sqrt{5}r + \frac{5r^2}{3})\exp(-\sqrt{5}r). \tag{2.73}$$

The process $f(\mathbf{x})$ is $k$-times mean square differentiable if $\nu > k$.

#### 2.2.3.1.3 Exponential Covariance Function

By setting $\nu = 1/2$ in the Matern class of covariance function, the exponential covariance function can be achieved as follows

$$k(r) = \exp(-r/l). \tag{2.74}$$

The Gaussian process defined using this covariance function is correspondingly mean square continuous, but it is not mean square differentiable. The process has a very rough characteristic which might limit its usage. It is less flexible than Matern covariance function.

#### 2.2.3.1.4 Rational Quadratic Covariance Function

The rational quadratic (RQ) covariance function can be defined as

$$k(r) = (1 + \frac{r^2}{2\alpha l^2})^{-\alpha}. \tag{2.75}$$

It can be seen as a scale mixture of SE covariance functions with different length-scales. The length-scales of the mixing components will be more diffusive as the hyperparameter $\alpha$ is getting smaller.

#### 2.2.3.1.5 Common Properties of Stationary Covariance Functions

The covariance functions explained previously decay monotonically with $r$ and are always positive. Beside the definition of isotropic covariance function, they can be expressed anisotropically as $r^2 = (\mathbf{x}-\mathbf{x}')^T M(\mathbf{x}-\mathbf{x}')$, where $M$ is positive semidefinite. if $M$ is defined diagonal, the idea of putting different length-scale on each dimension will be followed. This idea is well-known as Automatic Relevance Determination(ARD) [30].

#### 2.2.3.2 Dot Product Covariance Function

The other covariance function, that is widely used in the literature, is the dot product or linear covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x} \cdot \mathbf{x}'. \tag{2.76}$$

If $\sigma_0^2 = 0$, it is called homogeneous linear kernel. Otherwise, it is called inhomogeneous. As it will be explained later, the form above can be obtained through the construction of a new kernel from the addition of the constant kernel with the linear kernel. Its generalized form can be obviously given as before as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x}\Sigma_p\mathbf{x}'. \tag{2.77}$$

The same idea behind ARD can be applied using the general covariance matrix $\Sigma_p$. Based on the fact that the positive integer power of a covariance function is also valid, the linear covariance function can be extended to the polynomial covariance function of

$$k(\mathbf{x}, \mathbf{x}') = (\sigma_0^2 + \mathbf{x}\Sigma_p\mathbf{x}')^p. \tag{2.78}$$

#### 2.2.3.3 Non-Stationary Covariance Function

One example of a non-stationary covariance function is called the neural network covariance function. When the transfer function of a one-layer neural network is set to the error function of $h(z) = \text{ert}(z)$, it takes the following form

$$k_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}}'}{\sqrt{(1 + 2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^T\Sigma\tilde{\mathbf{x}}')}}\right), \tag{2.79}$$

where $\tilde{\mathbf{x}} = (1, x_1, \cdots, x_d)^T$. Derivations and details are well described in [38], [52]. Another example of a non-stationary neural network covariance function is given in

[38] for other hidden unit transfer function of $h(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2/2\sigma^2)$.

To introduce new non-stationary covariance functions, it is intuitive to employ an arbitrary non-linear mapping or warping $\mathbf{u}(\mathbf{x})$ of the input $\mathbf{x}$. Then, a stationary covariance function in $\mathbf{u}$ space can be used. It is worth mentioning that $\mathbf{x}$ and $\mathbf{u}$ do not have the same dimensionality. For instance, the covariance below is implementing the same idea

$$k(x, x') = \exp\left(-\frac{2\sin^2(2\pi\frac{x-x'}{\gamma})}{l^2}\right), \tag{2.80}$$

Other approach to non-stationary is the covariance functions that have variable length-scale according to the input space. Further details and derivations can be reached in [38].

#### 2.2.3.4 Constructing New Kernels

Various covariance functions are introduced and explained, so this part will focus on the ways they can be modified and combined with each other to form new ones [38]. The most straightforward approach is summation where the sum of two kernels is a kernel. This method can be used to combine kernels with various length-scale characteristics. Next, the product of two kernels is a kernel. It is effectively used in the explanation of the polynomial covariance function. Vertical rescaling can be used such that

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \frac{k(\mathbf{x}, \mathbf{x}')}{\sqrt{k(\mathbf{x}, \mathbf{x}')}\sqrt{k(\mathbf{x}, \mathbf{x}')}}. \tag{2.81}$$

This is to ensure that $k(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x}$.

If an arbitrary fixed kernel $h(x, z)$ is defined and the convolution of $g(\mathbf{x}) = \int h(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\mathbf{z}$, knowing that $k(\mathbf{z}, \mathbf{z}') = f(\mathbf{z}) \cdot f(\mathbf{z}')$, the covariance function of the

convolution can be obtained [38] as

$$\mathrm{cov}\left(g(\mathbf{x}), g(\mathbf{x}')\right) = \int h(\mathbf{x}, \mathbf{z}) k(\mathbf{z}, \mathbf{z}') h(\mathbf{x}', \mathbf{z}') d\mathbf{z} d\mathbf{z}'. \tag{2.82}$$

## 2.3 Sparse Bayesian Linear Modeling: Relevance Vector Machine

So far, there has been an emphasis on the nonparametric kernel models with the explanation on the Bayesian analysis toward modeling. Gaussian process model serves as a general framework to address both the regression and classification problems in the supervised learning context. GP, working as an intrinsically probabilistic model, provides prediction uncertainty for test samples and offers a principled mechanism towards addressing tasks related to decision making process such as reject option very well.

Another probabilistic model will be introduced in this section and consequently the model will be developed for both regression and classification problems. Provided that a linear regression model can be followed, its Bayesian treatment will naturally lead to a sparse Bayesian model which is known as Relevance Vector Machine (RVM) [46]. It should be noted that the RVM is a purely parametric probabilistic model that uses Bayesian treatment to provide a sparse solution to supervised learning classification problems.

RVM has been proposed in [46] in correspondence to support vector machine (SVM) [49] to offer a model carrying forward the advantages and discarding the disadvantage of SVM. On one hand, Sparsity and kernel property are considered as two main advantages of the SVM. On the other hand, non-probabilistic prediction is the most noticeable disadvantage of SVM. Therefore, a Bayesian treatment acts as a promising approach to address this disadvantage in a principled fashion. In the following, first the model will be introduced for regression and afterwards, the

classification will be explained based on the regression formalism.

### 2.3.1 Regression Modeling

Regression modeling under RVM formalism begins by introducing the linear model of [6], [46]

$$f(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \tag{2.83}$$

where the training set is defined as $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$ containing the pair of training input and target values. The fixed nonlinear basis functions is denoted by $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_M(\mathbf{x}))^T$. The weight vector is also expressed as $\mathbf{w} = (w_1, w_2, \cdots, w_M)^T$. Following the standard probabilistic formalism of additive noise of

$$y_n = f(\mathbf{x}_n;\mathbf{w}) + \epsilon_n, \tag{2.84}$$

where $\epsilon_n$ are samples from a zero-mean Gaussian distribution with variance $\sigma^2$. Thus, the conditional distribution can be given as

$$p(y_n|\mathbf{x}) = \mathcal{N}(y_n|f(\mathbf{x}_n), \sigma^2). \tag{2.85}$$

Following the successful approach of kernel machines for substitution of the basis functions with a kernel, the linear model of Eq. 2.83 can be expressed by using kernel function as

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i k(\mathbf{x}, \mathbf{x}_i) + b, \tag{2.86}$$

where $k$ is a kernel function such that $\phi_i(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}_i)$ and $b$ represents the bias parameter. Assuming the independence of the target value, the likelihood of the

complete data set can be written as

$$p(\mathbf{y}|X, \mathbf{w}, \sigma^2) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)$$
$$= (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{y} - \Phi\mathbf{w}||^2\right), \tag{2.87}$$

where $\Phi$ is the design matrix defined as $\Phi = [\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \cdots, \boldsymbol{\phi}(\mathbf{x}_N)]^T$ such that each row is redefined as $\boldsymbol{\phi}(\mathbf{x}_i) = [1, k(\mathbf{x}_i, \mathbf{x}_1), k(\mathbf{x}_i, \mathbf{x}_2), \cdots, k(\mathbf{x}_i, \mathbf{x}_N)]^T$. If the maximum likelihood approach for estimating the best values of model parameters $\mathbf{w}$ and $\sigma^2$ is followed, the undesired phenomena of over-fitting will be highly expected. However, this could be avoided by explicit definition of a prior distribution over the weight space. Following [46], the preference for the smoother functions is promoted by using a zero-mean Gaussian prior distribution over $w$ of

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1}). \tag{2.88}$$

It is evident that the prior has to be defined such that there is only one hyperparameter for each weight parameters instead of a single shared hyperparameter [6]. This is where the RVM model distinguishes itself from other Bayesian approaches. This hierarchical prior formalism can be further followed into one more lower level and define hyperpriors over $\boldsymbol{\alpha}$ and the noise variance $\sigma^2$. Further details and explanation can be followed in [46] using Gamma distribution.

It is important to define an individual hyperparameter for each weight as it will lead to the sparse solution at the end. Maximizing the evidence with respect to these hyperparameters will cause a significant of them go to infinity and consequently the posterior distributions of the weights will have the peak concentrated at zero. Therefore, it is evident that those basis functions associated with such weights will not affect the predictions of the model, meaning they are effectively pruned out and a sparse model is achieved. The inference in the model proceeds in such a way that

according to [46], the posterior distribution for the weights is given as

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\alpha}, \sigma^2) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.89}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\Phi^T\mathbf{y} \tag{2.90}$$

$$\boldsymbol{\Sigma} = (\sigma^{-2}\Phi^T\Phi + A)^{-1}, \tag{2.91}$$

where $A = diag(\alpha_0, \alpha_1, \cdots, \alpha_N)$. One should take note that in the case of Eq. 2.86, $\Phi = K$ the Gram matrix.

The next step is the hyperparameter learning procedure. Following the hierarchical model of [46] while assuming the uniform hyperpriors, the optimal values for the hyperparameters can be obtained through maximizing the marginal likelihood or the evidence. This framework is called evidence approximation or type-2 maximum likelihood procedure. The marginal likelihood or evidence can be obtained by integrating out the weight parameters as

$$p(\mathbf{y}|X, \boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{y}|X, \mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})d\mathbf{w}. \tag{2.92}$$

Since this is the integration of two Gaussian distributions, it can be evaluated according to [6] such that the log-marginal likelihood is as

$$\begin{aligned}\log p(\mathbf{y}|X, \boldsymbol{\alpha}, \sigma^2) &= \log \mathcal{N}(\mathbf{y}|0, C), \\ &= -\frac{1}{2}\{N\log(2\pi) + \log|C| + \mathbf{y}^T C^{-1}\mathbf{y}\},\end{aligned} \tag{2.93}$$

where $C$ is defined as

$$C = \sigma^2 I + \Phi A^{-1}\Phi^T. \tag{2.94}$$

Therefore, the ultimate goal in the learning stage is to maximize the Eq. 2.93 with respect to $\boldsymbol{\alpha}$ and $\sigma^2$ which can be approached by taking the derivative of the marginal likelihood and setting it to zero. Then, the terms below have to be repeatedly

evaluated until a convergence criterion is met

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2}, \tag{2.95}$$

where $\mu_i$ is the i-th posterior mean weight in Eq. 2.90 and the quantity $\gamma_i$ is the measure of accuracy of weight parameter $w_i$ being determined from the training data.

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}. \tag{2.96}$$

$\Sigma_{ii}$ denotes the i-th diagonal element of the posterior weight covariance given in Eq. 2.91. Following the interpretation in [46], when $\alpha_i$ is small and $w_i$ fits the data, $\gamma_i \approx 1$. On the other hand, $\gamma_i \approx 0$, when $\alpha_i$ is large and $w_i$ is constrained by the prior. In the similar manner, the noise variance is given as

$$(\sigma^2)^{new} = \frac{||\mathbf{y} - \Phi\boldsymbol{\mu}||^2}{N - \sum_i \gamma_i}. \tag{2.97}$$

The learning procedure can be summarized in the following steps. First the hyperparameters $\boldsymbol{\alpha}$ and $\sigma$ have to be initialized. Then, both the posterior mean and variance of Eq. 2.90 and 2.91 have to be evaluated. Hyperparameters have to be re-estimated using Eq. 2.95 and 2.97 afterwards. By using these new values, the posterior mean and variance will have to be re-evaluated. This procedure is repeated until a convergence criterion is satisfied. The other approach that is similar to this evidence approximation method is the EM algorithm that can be found in [6].

It can be observed that some of the hyperparameters have some large values which can be regarded as basically infinite values once the convergence is achieved. Following the same idea that the respective weight parameter will then have posterior distribution of zero mean and variance, it can be concluded that the basis function related to the weight parameter plays no role in the prediction process. Thus, the sparse model can be derived from this intuition and the remaining non-zero weight parameters are collected as relevance vectors. Having found $\boldsymbol{\alpha}_{MP}$ and $\sigma_{MP}$ after the

convergence of the evidence approximation procedure, the predictive distribution on the target value for a test sample $x_*$ is

$$p(y|X, \mathbf{x}_*, \mathbf{y}, \boldsymbol{\alpha}_{MP}, \sigma_{MP}) = \int p(y|\mathbf{x}, \mathbf{w}, \sigma_{MP}) p(\mathbf{w}|X, \mathbf{y}, \boldsymbol{\alpha}_{MP}, \sigma_{MP}) d\mathbf{w},$$
$$= \mathcal{N}(y|f_*, \sigma_*^2), \tag{2.98}$$

where

$$f_* = \boldsymbol{\mu}^T \boldsymbol{\phi}(\mathbf{x}_*), \tag{2.99}$$
$$\sigma_*^2 = \sigma_{MP}^2 + \boldsymbol{\phi}(\mathbf{x}_*)^T \Sigma \boldsymbol{\phi}(\mathbf{x}_*). \tag{2.100}$$

In this part, the Bayesian treatment of linear regression model is given such that the prior over the weight parameters can lead to a sparse solution. The inference and learning procedures were explained while the predictive posterior was derived at the end. In the next part, the same procedures will be followed for classification problems and the RVM model will be developed in a principled framework.

### 2.3.2 Classification Modeling

The same direction as Gaussian processes will be followed to adapt the regression model to classification problems in RVM. The likelihood function of additive noise in regression approach will be changed with the Bernoulli distribution for binary classification, and by applying the conventional approach of logistic sigmoid link function $\sigma(f) = 1/(1 + \exp(-f))$ to the linear regression model of $f$, the likelihood function can be written as

$$p(\mathbf{y}|\mathbf{w}) = \prod_{i=1}^{N} [\sigma(f(\mathbf{x}_i; \mathbf{w}))]^{y_i} [1 - \sigma(f(\mathbf{x}_i; \mathbf{w}))]^{1-y_i}, \tag{2.101}$$

where the target labels follow the labeling convection of $t_i \in \{0, 1\}$. Unlike the regression case, it is not obviously possible to analytically integrate out the weights and

obtain both the weight posterior $p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha})$ and marginal likelihood $p(\mathbf{y}|\boldsymbol{\alpha})$. Therefore, an approximation method is required. Following [46], the Laplace method is the most common approximation that can be used [25].

The inference stage demonstrates that the Laplace approximation for the posterior distribution has to be obtained using the current value of $\boldsymbol{\alpha}$. Then, the RVM model proceeds into the learning stage by using the approximate posterior mean and variance. The hyperparameter values of $\boldsymbol{\alpha}$ need to be re-estimated. This approach is repeated until the convergence criterion is met.

Since the normalization terms can be omitted during the maximization of the posterior $p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}) \propto p(\mathbf{y}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})$, the maximization of the log of the posterior is equivalent to the maximization of the following

$$\log\{p(\mathbf{y}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{i=1}^{N}[y_i \log f_i + (1 - y_i)\log(1 - f_i)] - \frac{1}{2}\mathbf{w}^T A\mathbf{w} + Const, \tag{2.102}$$

where $f_i = \sigma(f(\mathbf{x}_i; \mathbf{w}))$, and $A = \text{diag}(\alpha_i)$. The maximization can be done through the iterative reweighed least squares (IRLS) to find $\mathbf{w}_{MP}$. The gradient vector and Hessian matrix of the log posterior is given as

$$\bigtriangledown \log p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}) = \Phi^T(\mathbf{y} - \mathbf{f}) - A\mathbf{w}, \tag{2.103}$$

$$\bigtriangledown \bigtriangledown \log p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}) = -(\Phi^T B\Phi + A), \tag{2.104}$$

where B is an $N \times N$ diagonal matrix having the elements of $b_i = f_i(1-f_i)$, and $\Phi$ is the design matrix. Having found the converged $\mathbf{w}_{MP}$ out of IRLS algorithm, the negative inverse Hessian represents the covariance matrix of the Gaussian approximation to the posterior weight distribution. By setting the first derivative of the Eq. 2.103 to

zero, the mean and covariance of the Laplace approximation will be expressed as

$$\mathbf{w}_{MP} = A^{-1}\Phi^T(\mathbf{y} - \mathbf{f}) \tag{2.105}$$

$$\Sigma = (\Phi^T B\Phi + A)^{-1}. \tag{2.106}$$

Having found the statistics $\Sigma$ and $\mathbf{w}_{MP}$ of the Gaussian approximation, the hyperparameters $\boldsymbol{\alpha}$ are re-estimated through similar procedures described in the regression problems using Eq. 2.95. If in the same manner, the marginal likelihood is maximized and the re-estimated value for $\boldsymbol{\alpha}$ derived, it will be seen that it is the same as Eq. 2.95 [6]. The generalization to multi-class classification is straightforward [6]. Defining $K$ linear model for each class of the form

$$f_k = \mathbf{w}_k^T\mathbf{x}, \tag{2.107}$$

using the softmax function, the output can be processed as below

$$a_k(\mathbf{x}) = \frac{\exp(f_k)}{\sum_j \exp(f_j)}. \tag{2.108}$$

Then, the log-likelihood can be expressed as

$$\log p(T|\mathbf{w}_1, \cdots, \mathbf{w}_K) = \prod_{i=1}^{N}\prod_{k=1}^{K} a_{ik}^{y_{ik}}. \tag{2.109}$$

It is clear that $T$ is a matrix in which the row $n$ follows the 1-of-K coding scheme for the data point $n$. The IRLS algorithm can be consequently followed to provide a principled approach to find the hyperparameters and the approximate model [6].

Having explained the basic RVM model for the classification problems, a fast sequential marginal likelihood maximization algorithm in order to mitigate the high computational complexity of $O(M^3)$ of RVM, where $M$ denotes the number of basis functions, is proposed in [47]. The original marginal likelihood maximization method explained previously proceeds with all the $M$ basis functions initially in the model,

and then iteratively update hyperparameters all at once. Due to the sparsity mechanism and after some iteration, some basis functions will be pruned out and the whole algorithm will accelerate. However, some first iterations will still have the computational complexity of $O(M^3)$. In the faster algorithm of [47], an empty model with no basis functions will be initialized and sequentially add basis functions to increase the marginal likelihood while modifying their weights[47].

## 2.4 Summary

The kernel models and the idea of projecting input space into a high dimensional feature space are first introduced in this chapter. Meanwhile, SVM as a well-known common kernel machine, which is formalized in a sparse manner, is mentioned. The concept of parametric and nonparametric models along with their properties, advantages, and disadvantages are explained.

Then Gaussian processes as a probabilistic, Bayesian nonparametric model is introduced. The regression modeling according to this framework is derived, and developed. Two distinct perspectives for regression leading to the development of GP model are expressed known as the regular weight space view and the unfamiliar strange function space view. The Bayesian treatment of both perspectives is to great extent formalized and the predictive distribution is obtained at the end.

Following the derived regression model, GP for classification begins with distinguishing between discriminative and generative approaches in classification. It is shown that GP classifier is following the discriminative framework. Upon the derivation and justification of the model, the necessity of using approximate methods becomes apparent. Laplace approximation method as the common and practical way to provide an accurate result is chosen, and the posterior, predictive and marginal likelihood are approximated properly. The idea of covariance function is expanded afterwards. Different types of them based on the properties and forms are categorized

and properly explained.

Relevance Vector Machine (RVM) as the result of the sparse Bayesian modeling to linear regression model is then presented, and the derivation and development for regression problems is explained under this framework. Then the extension to classification problems is given and the Laplace method is applied as the required approximation method to the intractable integration.

It is evident that these two models are expressed as probabilistic models that can provide prediction uncertainty in a principled framework. Furthermore, it is guaranteed that for decision making tasks such as reject option, probability theory is the appropriate promising framework that can be applied in a reliable manner.

# Chapter 3

# Multistage Probabilistic Classification Modeling

In this chapter, the whole proposed system will be explained in detail and the reasons behind the cascade architecture and the approach to the integration of classifiers will be described. This multistage architecture will be justified in the following sections. The idea of automatic feature extraction using the CNN will be explained in 3.1 as the entry stage to the whole proposed system. Next, the issue of high computational complexity ahead of using the GPC will be investigated in 3.2 and the intuitive utilization of SVM to tackle this issue as the second stage of the model will be demonstrated. The approaches to model selection chosen in the experiment such as cross-validation will be explained in 3.3. Different Schemes to handle the issue of multi-class classification will be studied in 3.4. The justification to the appropriate scheme will be given respectively. Lastly, the considerations and requirements of reject option in the classification framework will be given in 3.5.

## 3.1 Preliminary Stage: Feature Extraction

In a high dimensional input space, it is of great importance to use some methods of feature extraction. In the case of two dimensional input spaces of handwritten

images, the extraction method has to be able to be applied to a spatial input space domain, and obviously harness the high correlation of adjacent pixels. Therefore it is clear that not all feature extraction methods have such capability.

The Convolutional Neural Network (CNN) proposed in [21] is originally considered a deep classification model to supervised learning problems. However, by looking carefully into the structure of the model, the two-stage training procedure introduced in the paper reveals the CNNs capability to be used both as an automatic feature extractor or a trainable classifier. The structure of CNN consists of the input layer which is fed with the raw two-dimension input data and the consecutive pairs of convolutional and sub-sampling layers. The feature extraction is clearly done through applying convolutional filtering and then sub-sampling operation alternately. This pair of operations can be followed in the consecutive manner until the desired level of feature extraction is reached. Once the discriminative features are extracted, the end part of the model will act as a trainable classifier which is composed of one fully connected layer and one output layer. As a typical multi-layer neural network, back-propagation is employed to take care of the training procedure.

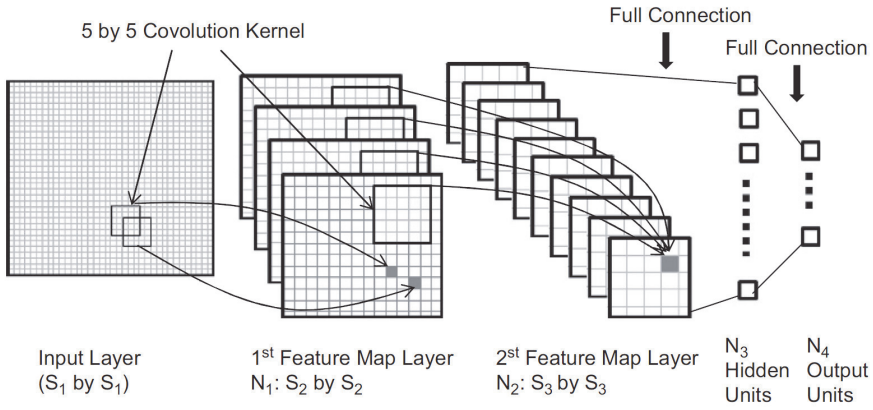


Figure 2: Structure of the simplified CNN [33]

The simplified CNN architecture is proposed in [35]. This architecture is specifically redesigned in the hope of improving the accuracy and speed of feature extraction

for handwriting data sets. The research demonstrates that the simplification does not harm the accuracy while still capable of precisely extracting spatial features usable for discrimination purposes. Following the simplified CNN version and the layer configuration of [33] illustrated in Figure 2, the normalized and centered raw image will be fed into the CNN model. The model is trained until it meets the convergence criterion after several hundred epochs. Then, the output of the third layer represents the extracted feature vector. The vectors particularly are not interpretable for human experts. However, the experiments in [33] reveal that the vector contains discriminative features which can be perfectly used by different trainable classifiers.

## 3.2 Compensating for GPC Complexity in Big Data

Despite the benefits of the complex modeling framework of the Gaussian processes expressed in chapter 2, the high computational complexity of $O(n^3)$ and the burden of matrix inversion encounters a lot during the modeling process threaten the applicability of such powerful modeling framework. There are some researches and proposals on approximation methods and sparse matrix manipulation for the case of big data given in [38]. For instance, the reduced-rank approximation methods of the gram matrix can be mentioned as such attempts. However, a simple, authentic and practical intuition is proposed in this research and form the keystone of the whole classification framework.

It is stated in [42] that the support vector (SV) set obtained from training an SVM classifier on a data set consist of the most informative samples based on which the classification task can be placed. In other words, the idea is further explained such that if the support vectors which are extracted from a specific data set using an SVM classifier are used to train a second SVM classifier with a totally different kernel and hyperparameter values, the classification result will no longer be worse that when the second SVM classifier is individually trained on the whole data set. This idea

has been used in [42] to introduce the idea of virtual support vectors in the hope of incorporating invariances into the classification machine and consequently improve the classifier's generalization to unseen data sets. The authentic intuition proposed in this research is inspired by this idea.

The proposed framework can be divided into the following steps. Frist, an SVM with the best configuration and parameterization on the data set is trained. Second, the support vector set is extracted from the trained SVM and a new training set is formed. Third, the probabilistic classifier of choice such as GPC or RVM is fed with the new training set and the model selection is followed. The best of both worlds will be ultimately benefited from following this hybrid cascade modeling paradigm. To start off, it should be noted that SVM model provides a fast, reliable and accurate recognition process to handle big data sets. Furthermore, the best probabilistic model is configured on the new training set consisting of the extracted support vectors. Consequently, the rejection process will be succeeded using the principally reliable measure of uncertainty outputted from the probabilistic classifier. On one hand, the big data set has been dealt with by means of the powerful model of SVM. On the other hand to propose the best measure of uncertainty for the reject option, probabilistic model such as GPC or RVM can be utilized. In this manner, the issue of high complexity of the probabilistic model will also be addressed properly.

## 3.3 Model Assessment

Model assessment is one of the critical parts in a classification task to assess different models while selecting the one with the best performance result. Different covariance functions or kernels would lead to distinct models with various characteristics and there has to be obviously a way to assess them and distinguish one from the others. The common approach to select the model based on the highest recognition and accuracy is k-fold cross-validation [6]. By using this approach, the training set is

partitioned into k different but equal folds. One fold will be held as the validation set while the remaining $k-1$ folds will be merged together and from a new training set. Successive tries of this method will lead to k distinct pairs of training and validation sets. The model will be altered depending on the parameters to be validated and then the one with the lowest averaged validation error or the highest averaged validation recognition rate will be chosen.

Up to this point, k-fold cross-validation is introduced as the method to select the model with the highest recognition rate. Furthermore, Receiver Operating Characteristics (ROC) curves, which provide the means to visualize and select the best classifier, are very well used in pattern recognition and machine learning literature, so their application is very widespread. By defining all the true positive, true negative, false positive along with false negative, the confusion matrix can be defined as a two-by-two matrix measuring all of these four values [14]. The true positive (TP) rate and false positive (FP) rate are defined as below

$$TP\ rate \approx \frac{\text{Positives correctly classified}}{\text{Total positives}}, \tag{3.1}$$

$$FP\ rate \approx \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}}. \tag{3.2}$$

The ROC curve can be drawn such that the vertical axis represents the TP rate and the horizontal axis the FP rate. Depending on the model configuration and parameterization, varying the threshold based on which the TP and FP rates measured generate one ROC curve in the whole ROC space. It is evident that this threshold mechanism is done based on the output score or probability that the particular classifier generates.

This ROC curve is not very applicable to classifier comparison since it is two dimensional. In order to derive scalar single value representing the expected performance, the area under the ROC curve (AUC) is defined in [14].The AUC is computed using

the trapezoidal approach explained in detail in [16]. It is evident that the higher the AUC value is, the better the classifier performs. The AUC of a few classifiers will be compared with each other in this research in order to prove experimentally the superiority of probabilistic classifier in achieving a lower rejection rate.

It is important to note that the definition of true positive and true negative followed in this research is oriented such that the AUC of ROC curve provides the means of achieving lower rejection rate. This is applied through the following definition: true positive represents the test or validation samples that are correctly rejected. Those that are correctly kept and recognized are categorized as true negative. In other words, if a misclassified sample is rejected, it is marked as true positive. Meanwhile, it is marked as true negative if a correctly classified sample is kept for recognition. Through comparisons, it can be said that the higher AUC of a classifier is, the better performance in terms of the rejection rate it can achieve.

## 3.4 Multi-class Classification Strategies

It is obvious that the numeral handwritten classification is a 10 class problem. The multi-class issue has to be dealt with properly during the design of the classification framework. There are different schemes and strategies in dealing with the multi-class situation as reviewed in [39]. One-vs-all (OVA) and all-vs-all (AVA) are two common schemes that use a binary classifier as the underlying classification model to address the multi-class issue. Other than these two simple and straightforward approaches, there are two other popular approaches. One is the single-machine approach in which a single multi-class classifier will be constructed to take care of the whole training procedure under one consistent model formulation. Gaussian process classifier using multi-latent likelihood function can be categorized as such. The other approach is known as error correcting approach which uses the idea of error correcting coding theory. First, a collection of binary classifiers must be chosen and then a method to

combine their output is needed.

Assuming that there are $k$ classes, the data set is partitioned into $k$ sets in OVA scheme such that the data samples belong to the first class is labeled to represent the first class and the whole samples of the remaining $k-1$ classes comprise the second class in the set. The similar procedure will be repeated for the other sets too. Once the partitioning is completed, one binary classifier is trained on each set. Then the final class label will be specified by the whole classification system according to the binary classifier with the highest output score or predicted probability values. AVA scheme is quite similar, but the difference is that $k(k-1)/2$ classifiers are required to be trained. Obviously it is slower and more computationally demanding than the OVA scheme since the number of the underlying classifiers is much higher and the need for cross-validation is eminent.

The argument is supported experimentally in [39] that despite the claims of single-machine scheme and error correcting approaches to reach the superior performance, the simple straightforward scheme of either OVA or AVA is observed as accurate and reliable as them as long as the underlying binary classifier is strong, well-trained and tuned, and regularized such as SVM classifier. This observation has been considered as the basis of multi-class treatment of handwritten recognition data using Gaussian process classifier and RVM in this research.

Since the main concern in conducting the multi-class classification experiments in [39] is to choose the scheme with the highest recognition performance, it might be relevant to see if the observation can be generalized to the cases where the rejection performance is as invaluable as recognition performance. In the case of a probabilistic classifier, the underlying binary classifier's predicted probability output is not normalized over the whole class labels. It will be interesting to see whether the rejection performance of the single-latent GPC would be better than the multi-latent GPC or not. In the former approach, the OVA multi-class scheme will be employed since the

core classifier is binary. In the latter approach, the single-machine multi-class approach will be followed which due to inaccurate approximations in the classification framework despite to normalized probability values might not be competitive enough. This will be further investigated in chapter 4.

## 3.5 Reject Option

Once the model has been tuned and trained properly, the next step is to measure the test error or recognition rate. Considering the reject option, it is also required to measure the AUC of ROC curve and rejection rate. These rates and measures can be used to compare between models and see which one could actually achieve the best overall performance. Reject option can be implemented in the experiment based on the best cutoff point of the measured ROC curve. The generalization of reject option can be followed in pursuing the idea of the AUC of ROC curve. In other words, if the AUC of a classifier is higher than another one, it can be experimentally concluded that the rejection rate based on various cutoff points for the classifier with the higher AUC value will be generally lower than the rejection rate for the classifier with lower AUC value.

There are some different measurements proposed in [17] that can be considered as the rejection criterion. The two famous and obvious ones are first rank measurement (FRM) [10] and the first two rank measurement (FTRM) [45]. The first one measures the highest classifier output values while the second approach is the difference between the top two output values. The output value can be a scoring value or predicted probability value depending on the choice of the classifier. The ROC curve is illustrated, the AUC is measured and a cutoff point for rejection rate is chosen based on this measurement. It should be noted that the ROC curve is obtained using the FTRM method.

Considering the simple rejection criterion of FTRM, the difference of the predicted

output values of the top two ranks will have the following interpretation. Having measured the difference of the top two ranks for the all test samples, two possible cases can happen. For the correctly-classified test samples, the first rank has a high prediction probability value, reflecting the model certainty of the decision, while the second rank has consequently a far lower value. As a result, the FTRM value will be large. On the other side, for misclassified test samples, the top rank prediction probability output will be very close to the second rank, and therefore, the FTRM value will be small. This is where the superiority of using an intrinsically probabilistic classifier such as GPC or RVM will shine against a non-probabilistic classifier such as SVM.

## 3.6 Summary

The proposed classification framework is summarized in the block-diagram illustrated in Figure 3. The whole framework is split into 5 distinct stages as follows. The top row of the figure depicts the conceptual modules sequentially and the bottom row illustrates the corresponding implementation of each stage. The feature extraction using CNN is the first stage of the proposed framework. In the second stage, the informative sample extraction is implemented using an SVM based on the computed features from the CNN. In this stage, the new training set will be created for the main part of the system. GPC or RVM will act as the probabilistic model which is trained using the same features from CNN of the indexed support vectors from SVM.

The fourth stage implements the reject option using the computation of FTRM based on which both AUC of ROC curve and threshold mechanism will be instructed. The class label prediction is followed in the last stage. The multistage architecture of the proposed classification framework helps instruct the integration of various powerful models namely CNN, SVM and GPC under a unified consistent classification system. It is obvious that this architecture is specifically chosen considering the reject

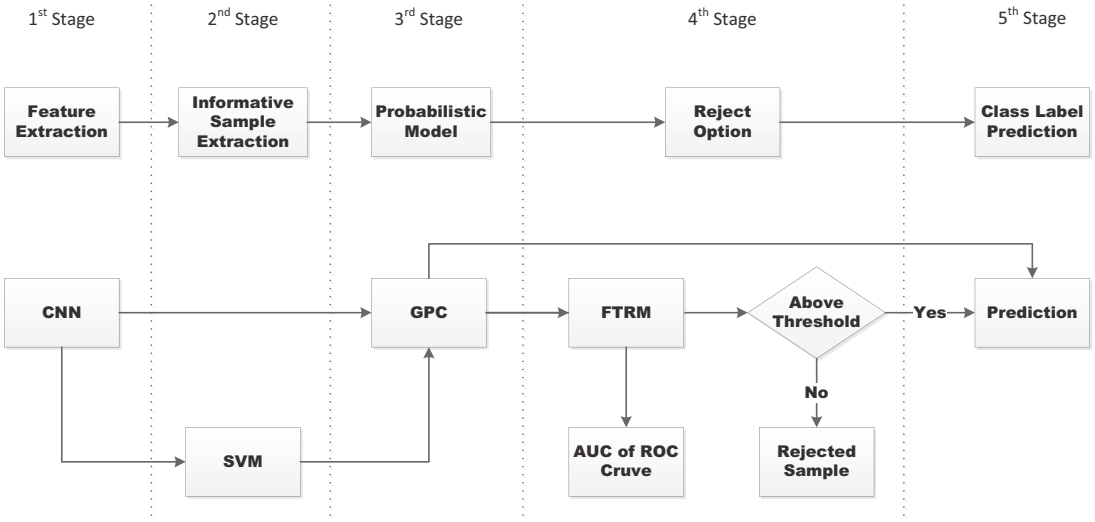


Figure 3: Block-Diagram of the Proposed classification framework

option as the main requirement.

# Chapter 4

# Experiments and Results

This chapter will discuss the whole experiment conducted under different model configurations, hyperparameter values, inference methods, learning approaches, and covariance or kernel functions. Three classifiers of GPC, RVM and SVM will be evaluated on two popular numeral data sets. An extensive set of experiments will be conducted on MNIST in section 4.1. GPC model will be examined under various parameter and configuration scenarios in the hope of finding the best model using cross-validation. Different multi-class schemes using single-latent and multi-latent GPC models will be investigated as well. The test results of GPC will be compared consequently with the best model of RVM and SVM and later to the best results to our knowledge in the literature. The same experimental setup will be extended to the CENPARMI numeral data set in section 4.2. The goal is to see if the superiority of the nonparametric Bayesian modeling approach to reject option can be generalized properly. At the end, the comparison to RVM, SVM, and the best models in the literature will be made.

## 4.1 MNIST Numeral Data Set

In this section, the proposed model will be experimented and verified on the public numeral handwritten data set known as MNIST [21]. It is frequently used in the

literature as the benchmark to measure the performance of different classification systems. It consists of 70000 numeral samples of which 60000 samples are randomly separated as the training set while the remaining 10000 samples are labeled as the test set. This data set originally is a subset of the big NIST data set. The numeral handwritten samples are represented as grayscale bitmap images that are already centered and normalized to the size of $28 \times 28$. Some samples of this data set are illustrated in Figure 4.

Figure 4: Samples from the MNIST numeral data set [21]

It is discovered in [37] and [20] that a better generalization level can be achieved using distortion methods applied to the original handwritten datasets. Following the results obtained in [33], the Simard's elastic distortion [43] is utilized to expand the entire training data set in the hope of achieving better performance results and more accurate models. The scaling and rotation transformations have been considered during the CNN feature extraction training stage. The simplified CNN model with the following architecture is employed. The number of feature maps in the layer one and two are 25 and 50 respectively. The hidden layer is the last layer consisting of

100 single units which can be extracted as the feature vector.

Having extracted the features properly, the 100 dimension feature vectors are then fed into an SVM classifier as the next stage of the proposed framework. According to the high recognition rate of the hybrid CNN-SVM model reported in [33], the parameters of $C = 128$ and $\sigma = 2^{-11}$ are chosen respectively for the $C$ as the model parameter to control the trade-off between slack variable penalty and the margin along with $\sigma$ as the kernel width of the RBF kernel.

Once the SVM training is done, the stage of SV set extraction begins. 1034 support vectors are extracted for all the 10 numeral classes. Obviously, they are represented in the 100 dimension feature space. From this stage of the experiment, it can be assumed that the problem is training a model on a 1034 sample training set of 100 dimension input space. Therefore, the whole modeling configuration and parameterization from this point onwards is totally detached form the one before. This approach can be seen as cascade architecture in the classification system design. The cascade approach is a multistage system of classifiers as it is apparent in the proposed system.

The arrangement for k-fold cross-validation also needs to be taken care of. In order to determine the best configuration and parameterization in the modeling stage, cross-validation has to be considered. It will guarantee the sufficient generalization to the test set. Using the new training set, 5-fold cross-validation is chosen for the whole experiment. The training set is randomly permuted and then partitioned into 5 folds. Then, 5 sets of pair of training and validation sets are created. Each set will be consequently treated with respect to the chosen multi-class scheme. It is obvious that for each set the training set is used to train the model based on a particular configuration and parameterization. Then, the validation misclassification error rate is measured on the corresponding validation set. Once this procedure is done for all the 5 sets, the average misclassification error rate is calculated for this specific model.

In the following, each one of GPC, RVM, and SVM models will be trained respectively while the best configuration and parameterization will be chosen using the

5-fold cross-validation method. The best models will be then evaluated on the MNIST test set and the model's performance in terms of misclassification error rate, AUC of ROC curve along with rejection rate will be calculated for later comparison.

### 4.1.1 GPC

In the following, the extracted features will be treated for the classification purpose under two different multi-class strategies while the experiment will be correspondingly conducted for each one to find the best configuration and parameterization. Different scenarios will be experimented to see how various covariance functions with varying parameters will affect the results, what is the effect of different inference and learning approaches for the Gaussian process classifier. Additionally, various types of likelihood functions for both single-latent and multi-latent cases will be examined towards achieving better performance results. GPstuff [48] is an extensive implementation of Gaussian process model that has been used throughout the entire experiments of the research.

#### 4.1.1.1 Binary Classification Approach

Having reduced the input space to 100-dimension feature space and extracted the SV set from the training set, the new training set is prepared to be fed into the second stage classification models. The main classifier is the GPC model and will be extensively experimented under different likelihood, inference, and covariance function scenarios. However, RVM will be regarded as another probabilistic classification model. Since it is a sparse kernel model like SVM, it will be shown empirically its test performance is not as high as GPC in the search of low rejection rate.

Up to this point, the 5-fold cross-validation (CV) is approached in the experiment and the training set is properly partitioned to 5 training-validation sets. Next, the OVA multi-class scheme to each one of the 5 CV sets needs to be applied. Each CV set is further split to create 10 new binary CV sets such that each one of the OVA sets

has one numeral class as the positive class label while the samples of the 9 remaining numeral classes are merged to form the negative class label. The Binary classifier such as GPC will be trained on the OVA sets once all sets are properly created.

The experiment reached to the point that the underlying binary classifier is the focus of interest. Using cross-validation, the best model has to be decided. Covariance functions play the premier role of producing models with various characteristics. At the beginning, selecting the proper covariance function is one important issue that needs to be addressed. Next, the model's behavior is varied by the hyperparameter values of that particular covariance function.

The next degree of freedom in the training procedure of the GPC model is the choice of inference method. There are a few possible methods of making the inference in the Gaussian process modeling framework. The most common, obvious, and fastest method is the Laplace approximation. However, it is concluded in [32] that "Expectation Propagation" (EP) algorithm is, in terms of accuracy, always the method of choice". Nonetheless, due to the large number of classifiers for a particular configuration and parameterization, the Laplace approximation method is primarily approached and then the extension to EP is followed.

In addition to the two previous degrees of freedom in the model selection process, the choice of likelihood function is highly influential. It is mentioned in chapter 2 that probit and logistic functions are two common likelihood functions that can be considered for single-latent GPC. Though logistic function is more widespread in the applications of GPC, probit likelihood function proves experimentally to be more accurate and faster to compute. The accuracy stems from the fact that the predictive probability distribution does not need to be approximated at all in the case of probit function and it can be analytically computed.

In the first part of the experiment, the training will proceed with variable inference methods and various covariance functions with their own hyperparameters. There is not going to be any specific procedure initiated for this part to learn the

hyperparameters. The best model is found using the manual search strategy on the hyperparameter space. Next, the experiment is extended to other inference methods. Later, it continues into the second part where learning will be considered as well. The learning procedure proceeds with defining proper hyperpriors on the hyperparameters and applying the MAP estimate procedure.

#### 4.1.1.1.1 Training Using Inference Procedure

The experiment is followed by a manual search strategy on the hyperparameter space of a specific covariance function. The best hyperparameter values will identified through cross-validation approach for each covariance function. The manual search is implemented only for the Laplace approximation method since it is fast enough to handle the time-consuming 50 classifier training process properly. Once they are experimentally explored and found, the two other inference methods of EP and MCMC will be initiated for the best hyperparameter values in order to see whether they can provide a better performance for the GPC model. It should be noted that since the prediction probability for probit likelihood function is analytically tractable, the results are accurately reliable whereas for logistic likelihood function, the prediction probability needs to be approximated and the results will not be as accurate. Thus, the grid search will be conducted using probit likelihood function and Laplace approximation inference method for the beginning and the extension will be given later.

##### 4.1.1.1.1.1 Manual Search on Hyperparameter Space

In this part of the experiment, different hyperparameter values of each covariance function will be investigated while the best value for which the lowest misclassification error rate is achieved will be selected. A 2 dimensional grid search is used for the covariance functions with two hyperparameters such as squared exponential, exponential, Matern, neural network, rational quadratic, and periodic covariance functions. It is only linear covariance function that has one hyperparameter for which 1-dimensional search is conducted.

The misclassification error rate of the GPC model for each of the aforementioned covariance functions is measured on each one of the five training-validation sets. Next, the averaged 5-fold cross-validation misclassification error rate is calculated. At the end, the hyperparameter values giving the lowest misclassification error rate will be chosen. The results for each covariance function is given as follows.

- **Squared Exponential Covariance Function**

  Squared exponential covariance function has been widely used in the literature. Following Eq. 2.70, it has two hyperparameters namely length-scale and magnitude that need to be cross-validated. Table 1 provides the averaged 5-fold cross-validation misclassification error rate in percentage for various combinations of length-scale and magnitude hyperparameters. It is clear that the lowest averaged misclassification error rate over all the pairs can be obtained for length-scale $l = 50$ and magnitude $\sigma_f^2 = 200$ as it is highlighted in the table.

Table 1: Averaged misclassification error rates using Squared Exponential covariance function

| Hyperparameter | | Magnitude | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 0.01 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 0.1 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 1 | 48.80 | 48.21 | 48.22 | 48.32 | 48.22 | 48.22 | 48.22 | 48.22 |
| | 10 | 48.65 | 39.37 | 28.89 | 25.83 | 29.38 | 29.88 | 30.64 | 31.42 |
| | 50 | 85.87 | 61.23 | 40.26 | 29.74 | 25.10 | 23.55 | **23.46** | 23.63 |
| | 100 | 85.87 | 85.87 | 48.48 | 36.18 | 28.36 | 26.27 | 24.82 | 23.75 |

- **Exponential Covariance Function**

  Exponential covariance function is the next one to be examined. It has the same types of hyperparameter as SE covariance function as indicated in Eq. 2.74. The 2 dimensional grid search is applied in the same manner and the results are shown in Table 2. It is emphasized that the lowest averaged misclassification error rate over different pairs of hyperparameter values for this covariance function can be achieved for length-scale $l = 100$ and magnitude $\sigma_f^2 = 50$.

- **Matern Covariance Function**

Table 2: Averaged misclassification error rates using Exponential covariance function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 |
| Length-scale | 0.01 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 0.1 | 92.36 | 91.57 | 91.37 | 91.18 | 91.37 | 91.37 |
| | 1 | 40.15 | 40.15 | 40.25 | 40.34 | 40.24 | 40.24 |
| | 10 | 50.75 | 40.93 | 31.82 | 28.52 | 28.98 | 29.29 |
| | 50 | 82.72 | 45.21 | 35.21 | 27.75 | 26.88 | 27.94 |
| | 100 | 85.87 | 49.23 | 37.63 | 27.91 | **26.80** | 27.07 |

Matern covariance function is the next covariance function that is experimented in this part. The special case of Matern $\nu = 3/2$ has been chosen in the experiment based on Eq. 2.72. The same hyperparameters as two previous covariance functions are applicable to be cross-validate using the grid search. The averaged misclassification error rates is shown in Table 3 for different pairs of length-scale and magnitude values. It is illustrated in the table that the lowest averaged misclassification error rate can be obtained using length-scale $l = 100$ and magnitude $\sigma_f^2 = 200$.

Table 3: Averaged misclassification error rates using Matern 3.2 covariance function

| Hyperparameter | | Magnitude | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 0.01 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 0.1 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 1 | 42.78 | 42.78 | 42.78 | 42.78 | 42.78 | 42.78 | 42.78 | 42.78 |
| | 10 | 48.84 | 39.95 | 30.35 | 28.60 | 29.39 | 29.77 | 29.77 | 29.87 |
| | 50 | 85.87 | 49.36 | 37.84 | 26.89 | 24.10 | 24.00 | 25.32 | 26.46 |
| | 100 | 85.87 | 77.64 | 42.02 | 32.16 | 25.76 | 24.80 | **22.96** | 23.93 |
| | 200 | 85.87 | 85.87 | 52.31 | 38.43 | 30.40 | 27.38 | 25.88 | 24.72 |

- **Neural Network Covariance Function**

Neural network covariance function has the form given in Eq. 2.79. Bias and weight variables are two hyperparameters of this covariance function. They are

cross-validated in Table 4 in attempts to find the lowest averaged misclassification error rate. Despite trying different ranges for the grid search, the results are not as promising as expected to. The results in the table denote that for three pairs the lowest averaged misclassification error rate could be achieved. Even though they are not low enough, they are representing the best results among various grid search value ranges. One of the best results is achieved for $Bias = 0.1$ and $Weight = 0.1$. This pair is chosen for later parts of the experiment.

Table 4: Averaged misclassification error rates using Neural Network covariance function

| Hyperparameter | | Weight | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 |
| Bias | 0.01 | 30.80 | **29.30** | 29.39 | 29.49 | 29.49 | 29.49 |
| | 0.1 | 31.09 | **29.30** | 29.39 | 29.49 | 29.49 | 29.49 |
| | 1 | 31.95 | **29.30** | 29.39 | 29.49 | 29.49 | 29.49 |
| | 10 | 35.31 | 30.66 | 29.39 | 29.49 | 29.49 | 29.49 |
| | 50 | 39.19 | 32.61 | 29.68 | 29.39 | 29.49 | 29.49 |
| | 100 | 40.47 | 34.27 | 30.85 | 29.39 | 29.39 | 29.39 |

- **Linear(Dot Product) Covariance Function**

Linear covariance function is always considered as the simplest and most common covariance function applied in the literature. The search for the best hyperparameter value can be easily accomplished by using one-dimensional cross-validation search. Following Eq. 2.77, the homogeneous linear covariance function is utilized in the experiment. Thus, only coefficient hyperparameter $\Sigma_p$ needs to be dealt with.

Table 5: Averaged misclassification error rates using Linear covariance function

| Coefficient (Sigma2) | 0.01 | 0.1 | 1 | 10 | 50 | 100 |
|---|---|---|---|---|---|---|
| Error rate | 26.45 | **23.05** | 27.60 | 35.22 | 40.08 | 41.83 |

It can be apparently seen in Table 5 that the lowest misclassification error rate is achievable for linear coefficient of 0.1. It should be noted that since ARD

approach considered on all the feature dimensions does not provide any better result, one coefficient common to all the dimensions has been decided. This treatment exactly applies to other covariance functions too. The reason can be justified as such that since the features are properly extracted from the raw input data, the feature vector on one dimension has as invaluable and discriminative information as the other ones. Thus, method such as ARD cannot further lead to any improvement by putting emphasis on some dimensions and ignoring the others.

- **Rational Quadratic Covariance Function**

  The other covariance function that is examined in the experiment is rational quadratic covariance function given in Eq. 2.75. It has three hyperparameters where two of them are the same as other covariance functions namely length-scale and magnitude. These two have been cross-validated in the same manner similar to other covariance functions using a 2-dimensional grid search. The last one is the alpha. To avoid exhaustive 3-dimensional search, through a preliminary search stage, $\alpha = 0.1$ is chosen to remains fixed throughout the entire experiment. Table 6 illustrates various averaged misclassification error rates and the best result that can be obtained using length-scale $l = 50$ and magnitude $\sigma_f^2 = 100$ is highlighted.

Table 6: Averaged misclassification error rates using Rational Quadratic covariance function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 |
| Length-scale | 0.01 | 85.87 | 65.38 | 41.79 | 38.59 | 37.23 | 36.94 |
| | 0.1 | 85.87 | 53.66 | 39.87 | 35.47 | 34.59 | 34.50 |
| | 1 | 85.69 | 48.33 | 38.02 | 32.37 | 31.20 | 30.90 |
| | 10 | 84.94 | 45.79 | 35.59 | 28.43 | 29.28 | 29.16 |
| | 50 | 85.87 | 70.33 | 40.95 | 31.17 | 25.46 | **24.58** |
| | 100 | 85.87 | 85.87 | 48.67 | 36.77 | 28.46 | 26.54 |

- **Periodic Covariance Function**

Periodic covariance function is the last type that is considered in the experiment. It has the largest number of hyperparameters. Following Eq. 2.80, magnitude and length-scale hyperparameters are treated the same as previous covariance functions. The period and length-scale for the squared exponential component are specific to this covariance function. They both are fixed to the value 10 based on a preliminary experiment. Table 7 indicates the best result in terms of the lowest averaged misclassification error rate which can be obtained by using length-scale $l = 10$ and magnitude $\sigma_f^2 = 10$.

Table 7: Averaged misclassification error rates using Periodic covariance function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 |
| Length-scale | 0.01 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 0.1 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 | 92.56 |
| | 1 | 45.38 | 45.38 | 45.39 | 45.19 | 45.29 | 45.29 |
| | 10 | 64.02 | 42.89 | 33.00 | **25.10** | 25.52 | 26.57 |
| | 50 | 85.87 | 85.87 | 47.42 | 36.16 | 27.67 | 26.32 |
| | 100 | 85.87 | 85.87 | 73.13 | 42.02 | 34.73 | 31.85 |

#### 4.1.1.1.1.2 Extension to Other Inference Methods and Likelihood Functions

The experiment on the search over hyperparameter space for each covariance function is finished in section 4.1.1.1.1.1 and the best hyperparameter values are obtained using the inference method of Laplace approximation and probit likelihood function. Using these best hyperparameter values, the experiment will be extended to two other inference methods, EP and MCMC, by using the same probit likelihood function. In another set of experiment, the GPC with the logistic likelihood function on the three inference methods will be examined. The reason is to imply experimentally that the logistic likelihood function can rarely lead to an improved predication over the probit likelihood function.

The averaged misclassification error rates for each covariance function under different inference methods and likelihood functions are given in Table 8. For each

covariance function, the best hyperparameter value is shown from the manual search experiment. The lowest averaged misclassification error rate in each row is highlighted to distinguish it from the others. It can be observed form the table that it is almost the probit-Laplace configuration that leads to the lowest rate. Squared Exponential covariance function is the only one that has the lowest error rate under probit-EP configuration. It should be noted that the MCMC inference method could not be applied to rational quadratic covariance function due to some numerical restriction.

Table 8: Averaged misclassification error rates using all inference methods and likelihood functions

| | | | Laplace | | EP | | MCMC | |
|---|---|---|---|---|---|---|---|---|
| | | | Probit | Logistic | Probit | Logistic | Probit | Logistic |
| Squared Exponential | $l = 50$ | $\sigma_f^2 = 200$ | 23.46 | 24.92 | **23.25** | 31.62 | 23.55 | 25.01 |
| Exponential | $l = 100$ | $\sigma_f^2 = 50$ | **26.80** | 27.76 | 27.56 | 28.03 | 27.28 | 27.92 |
| Matern 3.2 | $l = 100$ | $\sigma_f^2 = 200$ | **22.96** | 25.28 | 23.62 | 25.35 | 23.72 | 25.67 |
| Neural Network | $Bias = 0.1$ | $Weight = 0.1$ | **29.30** | 34.16 | 29.30 | 34.83 | 29.52 | 34.64 |
| Linear | $\sigma^2 = 0.1$ | | **23.05** | 24.32 | 23.54 | 24.23 | 23.55 | 24.32 |
| Rational Quadratic | $l = 50$ | $\sigma_f^2 = 100$ | **24.58** | 26.51 | 24.97 | 31.45 | NA | NA |
| Periodic | $l = 10$ | $\sigma_f^2 = 10$ | **25.10** | 27.26 | 25.48 | 27.36 | 25.67 | 27.08 |

It is clear that probit likelihood function has tractable analytical solution whereas logistic likelihood function has to be approximated in the integration Eq. 2.41 giving the prediction probability of the test sample. The superiority of probit to logistic is experimentally supported by the results obtained in this part of the experiment. It can be further observed that the MCMC sampling method could not provide any improvement in the results neither for probit nor logistic likelihood functions. This might stem from the fact that they need to be sampled for a very long time in order to be able to achieve a comparative result. However, it should be noted that each classifier has been sampled a few hundred times in this experiment. This means 10000 samples have been taken from the model to achieve the averaged cross-validation error rate for a specific arrangement of likelihood function and inference method.

#### 4.1.1.1.2 Training Using Learning and Inference Procedures

The experiment of single-latent GPC will further be extended by considering the option of learning the best hyperparameter values using the MAP estimate approach. The main focus of this part is on the capability of MAP and MCMC learning procedures to find better hyperparameter values than the manual search experimented in section 4.1.1.1.1.1. The MAP learning proceeds by maximizing the un-normalized a posteriori distribution of the hyperparameters given the latent functions. In a Bayesian treatment, defining hyperpriors on the hyperparameters will have the role of a regularization term in the maximum-likelihood estimation treatment.

In this part of the experiment, some common hyperpriors will be defined on the hyperparameters. The MAP estimate is followed by optimizing the product of likelihood and hyperprior terms with respect to each hyperparameter. The MCMC estimate is approached in the following steps. First the hyperparameters are sampled from the model to obtain the best values. Then, the latent functions are sampled using these hyperparameter values in the next stage. Finally, the predictions are generated using the averaged samples of latent functions.

The student-t distribution is defined as the hyperprior on the length-scale hyperparameter. The uniform distribution is chosen for the magnitude hyperparameter. Indeed, they apply to the covariance functions that have these two hyperparameters. For neural network covariance function, the uniform distribution will be set as the hyperprior for both bias and weight hyperparameters. The uniform distribution is the common hyperprior defined on the coefficient hyperparameter.

The covariance functions' hyperparameters are initialized using the best values achieved in section 4.1.1.1.1.1. This could boost the optimization procedure to better search the space in the hope of a lower minimum. Since the logistic likelihood function could not achieve any one of the best results in the previous part of the experiment, this observation is generalized and only probit likelihood function is experimented in this part.

Table 9 illustrates the averaged misclassification error rate for each covariance function under different inference methods. It should be recalled that the learning method of choice is MAP for the both inference methods of Laplace and EP while for the MCMC inference method, it is the MCMC learning approach.

Table 9: Averaged misclassification error rates using MAP Learning and Various Inference Methods for Single-Latent GPC

| | Laplace | EP | MCMC |
|---|---|---|---|
| Squared Exponential | 24.30 | 24.87 | **23.67** |
| Exponential | **27.19** | 27.87 | 27.66 |
| Matern 3.2 | 24.78 | **24.77** | 24.96 |
| Neural Network | 29.58 | 29.78 | **29.30** |
| Linear | **24.03** | 24.12 | 24.31 |
| Rational Quadratic | **45.51** | 45.51 | NA |
| Periodic | **24.50** | 25.19 | 25.17 |

The lowest averaged misclassification error rate in each row is highlighted. The lowest averaged misclassification error rate is achieved using Laplace inference method for all of th covariance functions except Matern 3.2, neural network, and squared exponential. EP is a little lower for Matern 3.2 than Laplace. MCMC could obtain the best result for squared exponential and neural network covariance functions.

#### 4.1.1.1.3 Discussion: Best Binary GPC hyperparameter Values

The experiment on single-latent GPC is completed at this point of the experiment. The best results are obtained for all of the covariance functions under two different scenarios. The first one is to use only inference procedure to find the latent functions and prediction probabilities and then use the manual search strategy to find the best hyperparameter values. The other scenario is to replace the manual search with a learning approach such as MAP. Then, the optimization procedure is initiated to automatically find the best hyperparameter values.

Table 10 illustrates the best results of both scenarios represented in tables 8 and 9. It can be observed that the manual search strategy for all of the covariance functions

except periodic covariance function could outperform the MAP or MCMC estimate approaches. At the end, the best results obtained from this table will be further compared with the best results of multi-latent GPC models. This comparison will result into finding the best GPC model with the lowest averaged misclassification error rate on the CV sets.

Table 10: Best results of Single-Latent GPC model

| | Inference | | Learning & Inference | |
|---|---|---|---|---|
| Squared Exponential | **EP** | **23.25** | MCMC | 23.67 |
| Exponential | **Laplace** | **26.80** | Laplace | 27.19 |
| Matern 3.2 | **Laplace** | **22.96** | EP | 24.77 |
| Neural Network | **Laplace** | **29.30** | MCMC | 29.30 |
| Linear | **Laplace** | **23.05** | Laplace | 24.03 |
| Rational Quadratic | **Laplace** | **24.58** | Laplace | 45.51 |
| Periodic | Laplace | 25.10 | **Laplace** | **24.50** |

#### 4.1.1.2 Multi-class Classification Approach

Multi-latent GPC model will be examined in this part of the experiment. It will be investigated how transferring from single-latent to multi-latent likelihood functions can affect the recognition and reliability performance of the system. At the first glance, this might be advantageous since there is no need to train explicitly 10 different binary classifiers for each cross-validation set leading to 50 distinct models. Using multi-latent likelihood function, this all comes under GPC modeling framework and will be addressed properly. However, the performance and accuracy of the final results need to be observed experimentally to figure out whether this theoretical model can lead to any better results.

Following the same procedure as the binary classification approach, the 100-dimension feature space will be extracted from CNN model. The informative training samples will be also extracted from the support vector set of the trained SVM classifier. The 5-fold cross validation will be setup again so that the best hyperparameter values of various covariance functions can be discovered. The same type of covariance functions

as the single-latent approach will be examined. It should be noted that due to numerical infeasiblity and implementation complexity, unlike the single-latent approach, one type of likelihood function is applicable for each inference methods, softmax likelihood function for the Laplace approximation method, and multinomial probit function for the EP method.

#### 4.1.1.2.1 Training Using Inference Procedure

In the similar manner to single-latent experiment, the manual search on various covariance functions will be conducted using different inference methods and the hyperparameter learning will be postponed to section 4.1.1.2.3. It will be analyzed at the end to see whether the learning procedure using MAP can improve the performance of the models in the multi-latent GPC or not.

##### 4.1.1.2.1.1 Manual Search on Hyperparameter Space

The manual search strategy using the Laplace inference method will be considered in the first part. The aim is to find the hyperparameter values for which the highest recognition performance can be achieved. It is obvious that the highest performance is measured based on the lowest averaged cross-validation misclassification error rate. Afterwards, the EP inference methods will be examined by utilizing the best hyperparameter values obtained in this section.

- **Squared Exponential Covariance Function**

  A $6 \times 6$ grid search is created for SE covariance function. As illustrated in Table 11, the lowest averaged misclassification error rate is achieved for length-scale $l = 50$ and magnitude $\sigma_f^2 = 400$.

- **Exponential Covariance Function**

  The same grid search size is used for exponential covariance function since the hyperparameters are similar to SE covariance function. The lowest averaged misclassification error rate of 27.10% is achieved for length-scale $l = 100$ and

Table 11: Averaged misclassification error rates using Squared Exponential Covariance Function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 1 | 89.72 | 89.02 | 89.25 | 89.85 | 89.27 | 89.65 |
| | 10 | 32.13 | 26.84 | 27.95 | 29.00 | 29.99 | 31.41 |
| | 50 | 46.28 | 34.51 | 26.99 | 25.97 | 24.90 | **24.32** |
| | 100 | 69.84 | 40.65 | 33.02 | 29.72 | 26.90 | 26.08 |
| | 200 | 85.87 | 49.93 | 39.77 | 36.66 | 32.83 | 29.82 |
| | 400 | 85.87 | 83.70 | 47.61 | 43.47 | 39.77 | 36.66 |

magnitude $\sigma_f^2 = 400$.:

Table 12: Averaged misclassification error rates using Exponential Covariance Function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 1 | 71.69 | 73.81 | 79.29 | 82.03 | 83.78 | 84.87 |
| | 10 | 35.11 | 30.16 | 28.82 | 29.02 | 29.03 | 29.01 |
| | 50 | 39.55 | 30.25 | 27.38 | 27.59 | 27.48 | 27.19 |
| | 100 | 41.72 | 31.45 | 28.25 | 27.38 | 27.20 | **27.10** |
| | 200 | 45.59 | 33.84 | 28.88 | 27.87 | 27.20 | 27.11 |
| | 400 | 52.04 | 37.06 | 30.74 | 28.70 | 27.67 | 27.20 |

- **Matern Covariance Function**

  By cross-validating the two hyperparameters using the same grid search approach, the best model performance is obtained for length-scale $l = 100$ and magnitude $\sigma_f^2 = 400$ for the averaged misclassification error rate of 24.51%.

Table 13: Averaged misclassification error rates using Matern Covariance Function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 1 | 86.90 | 86.98 | 87.68 | 87.88 | 88.18 | 88.96 |
| | 10 | 33.65 | 28.54 | 28.81 | 29.10 | 29.50 | 28.72 |
| | 50 | 41.24 | 30.81 | 26.43 | 25.37 | 24.98 | 26.61 |
| | 100 | 50.21 | 36.56 | 29.31 | 27.00 | 25.48 | **24.51** |
| | 200 | 82.42 | 42.71 | 34.89 | 31.87 | 28.92 | 27.01 |
| | 400 | 85.87 | 55.28 | 40.95 | 38.33 | 34.71 | 31.49 |

- **Neural Network Covariance Function**

  The averaged misclassification error rates for this covariance function is illustrated in Table 14. The lowest error rate is achieved for several pairs. Despite further attempts on the hyperparameters space out of the grids range, no better results could be achieved. $Bias = 400$ and $Weight = 400$ are chosen as one of the pairs that results into the lowest error rate.

Table 14: Averaged misclassification error rates using Neural Network Covariance Function

| Neural Network | | Weight | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Bias | 1 | **33.19** | **33.19** | **33.19** | **33.19** | **33.19** | **33.19** |
| | 10 | 33.37 | **33.19** | **33.19** | **33.19** | **33.19** | **33.19** |
| | 50 | 34.32 | 33.28 | **33.19** | **33.19** | **33.19** | **33.19** |
| | 100 | 34.80 | 33.37 | **33.19** | **33.19** | **33.19** | **33.19** |
| | 200 | 35.56 | 33.47 | 33.28 | **33.19** | **33.19** | **33.19** |
| | 400 | 36.26 | 33.94 | 33.37 | 33.28 | **33.19** | **33.19** |

- **Linear(Dot Product) Covariance Function**

  The linear covariance function has one hyperparameter. Using a one dimensional search, the best value can be achieved. Table 15 denotes that the lowest averaged misclassification error rate of 24.62% is obtained for $\sigma^2 = 0.1$.

Table 15: Averaged misclassification error rates using Linear Covariance Function

| Coefficient (Sigma2) | 0.001 | 0.01 | 0.1 | 1 | 10 | 50 | 100 | 200 | 400 |
|---|---|---|---|---|---|---|---|---|---|
| Error rate | 40.86 | 29.92 | **24.62** | 29.51 | 38.05 | 44.69 | 49.42 | 56.26 | 65.95 |

- **Rational Quadratic Covariance Function**

  Following the same grid search approach to find the best hyperparameter values for this covariance function, the averaged error rate of 25.37% is achieved for the hyperparameters of length-scale $l = 50$ and magnitude $\sigma_f^2 = 400$ as it is illustrated in Table 16.

Table 16: Averaged misclassification error rates using Rational Quadratic Covariance Function

| Hyperparameter | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 1 | 40.84 | 34.25 | 32.98 | 33.06 | 32.70 | 32.49 |
| | 10 | 40.26 | 30.92 | 28.92 | 29.57 | 29.18 | 29.02 |
| | 50 | 47.60 | 34.98 | 28.61 | 26.61 | 25.57 | **25.37** |
| | 100 | 72.55 | 40.95 | 33.50 | 30.40 | 27.38 | 26.26 |
| | 200 | 85.87 | 50.21 | 39.87 | 36.66 | 32.92 | 29.72 |
| | 400 | 85.87 | 83.70 | 47.70 | 43.37 | 39.77 | 36.66 |

- **Periodic Covariance Function**

  Periodic covariance function is the last to be examined. Similar to the single-latent GPC experiment for this particular covariance function, the period and length-scale for the squared exponential component are fixed to the value 10 based on the preliminary experiment results. The two remaining hyperparameters are cross-validated using the grid search strategy and the results are shown in Table 17. The lowest averaged misclassification error rate of 25.58% is obtained for length-scale $l = 50$ and magnitude $\sigma_f^2 = 400$:.

Table 17: Averaged misclassification error rates using Periodic Covariance Function

| Periodic | | Magnitude | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 1 | 78.17 | 79.12 | 82.46 | 83.89 | 84.37 | 86.13 |
| | 10 | 37.52 | 27.38 | 26.05 | 27.68 | 28.53 | 29.52 |
| | 50 | 64.48 | 41.14 | 31.97 | 29.72 | 27.19 | **25.58** |
| | 100 | 85.87 | 49.06 | 39.50 | 36.24 | 31.97 | 29.62 |
| | 200 | 85.87 | 80.86 | 47.04 | 42.21 | 39.40 | 36.24 |
| | 400 | 85.87 | 85.87 | 73.34 | 54.92 | 47.04 | 42.21 |

#### 4.1.1.2.1.2 Extension to Other Inference Methods Using Probit Function

The experiment on multi-latent GPC model will be extended to the EP inference method using multinomial probit likelihood function on the best hyperparameter values achieved in the section 4.1.1.2.1.1. MCMC has been omitted due to the high computational complexity and long training time for multi-latent GPC model. It

is observed in the single-latent GPC experiment that the MCMC method could not reach any better results when compared to the Laplace and EP methods.

Table 18: Averaged misclassification error rates using the Laplace and EP inference methods and the Multinomial Probit likelihood function

| | | | Laplace | EP |
|---|---|---|---|---|
| Squared Exponential | $l = 50$ | $\sigma_f^2 = 400$ | **24.32** | 24.82 |
| Exponential | $l = 100$ | $\sigma_f^2 = 400$ | **27.10** | 27.58 |
| Matern 3.2 | $l = 100$ | $\sigma_f^2 = 400$ | 24.51 | **24.40** |
| Neural Network | $Bias = 400$ | $Weight = 400$ | 33.19 | **30.99** |
| Linear | $\sigma^2 = 0.1$ | | **24.62** | 25.28 |
| Rational Quadratic | $l = 50$ | $\sigma_f^2 = 400$ | **25.37** | 26.05 |
| Periodic | $l = 50$ | $\sigma_f^2 = 400$ | 25.58 | **25.19** |

Table 18 indicates the comparison of the Laplace approximation and EP inference methods together for each covariance function using the best hyperparameter values. The lowest averaged misclassification error rate in each row is highlighted to distinguish it from the other. It is clear that the improvement is not very significant. The best results of this part will be compared with the results achieved using the MAP estimate learning procedure in the section 4.1.1.2.3.

#### 4.1.1.2.2 Training Using Learning and Inference Procedure

The same MAP estimate learning approach as the one followed in the single-latent GPC experiment will be considered for multi-latent GPC in the same manner. Since similar types of covariance function are used, the same hyperpriors on the hyperparameters will be defined. This will form the base of a fair comparison. Initializing the optimization procedure by using the best hyperparameter values obtained in section 4.1.1.2.1.1, the averaged misclassification error rate is achieved for the two inference methods of Laplace and EP for each covariance function as illustrated in Table 19.

It can be observed from the results in the table that the Laplace method provides superior performance results in most of the cases even though the improvements

Table 19: Averaged misclassification error rates using MAP Learning and The Laplace and EP Inference Methods For Multi-Latent GPC

| | Laplace | EP |
|---|---|---|
| Squared Exponential | **24.42** | 24.69 |
| Exponential | **27.20** | 27.39 |
| Matern 3.2 | **24.61** | 24.98 |
| Neural Network | 33.19 | **30.89** |
| Linear | **24.03** | 24.90 |
| Periodic | 25.38 | **25.37** |

are very subtle. It is only neural network an periodic covariance functions that has superior result using the EP method. It should be noted that the rational quadratic covariance function is omitted in this experiment due to the numerical infeasiblity encountered during the experiment.

#### 4.1.1.2.3 Discussion: Best Multi-Latent GPC Model

The results obtained in sections and can be completely compared with one another as shown in Table 20. It can be observed that the learning procedure cannot necessarily provide an improvement in contrast to the manual grid search strategy. Five covariance functions have their best recognition performance using the manual search strategy, and only the linear and neural network covariance functions could substantially supersede using the learning procedure. The numbers highlighted represent the lowest averaged misclassification error rate for each covariance function.

Table 20: Best Results of Multi-Latent GPC Model

| | Inference | | Learning & Inference | |
|---|---|---|---|---|
| Squared Exponential | **Laplace** | **24.32** | Laplace | 24.42 |
| Exponential | **Laplace** | **27.10** | Laplace | 27.20 |
| Matern 3.2 | **EP** | **24.40** | Laplace | 24.61 |
| Neural Network | EP | 30.99 | **EP** | **30.89** |
| Linear | Laplace | 24.62 | **Laplace** | **24.03** |
| Rational Quadratic | **Laplace** | **25.37** | NA | NA |
| Periodic | **EP** | **25.19** | EP | 25.37 |

The best result of each covariance function will be compared with the corresponding result of single-latent GPC in the section 4.1.1.3 in order to find the best GPC model configuration and parameterization on the MNIST numeral data set. This particular model will be compared with the models of RVM and SVM at the end.

#### 4.1.1.3 Discussion: Best GPC Configuration and Parameterization

Both the single-latent and multi-latent GPC models are already evaluated on the MNIST data set to this point of the experiment. The best model configuration and parameterization under different inference methods, likelihood functions and learning approaches are decided. The best results from both GPC models should be explicitly compared with one another to figure out which one can lead to a better model performance.

Table 21 illustrates the results of each covariance function under either single-latent or multi-latent GPC models. The method of acquiring the best hyperparameter values, the inference method and the averaged misclassification error rate are given for each covariance function in the table. It should be reminded that all these results are achieved using probit likelihood function.

Table 21: Comparison of Single-Latent to Multi-Latent GPC models for Various Covariance Functions

| | Single-Latent GPC | | | Multi-Latent GPC | | |
|---|---|---|---|---|---|---|
| Squared Exponential | **Search** | **EP** | **23.25** | Search | Laplace | 24.32 |
| Exponential | **Search** | **Laplace** | **26.80** | Search | Laplace | 27.10 |
| Matern 3.2 | **Search** | **Laplace** | **22.96** | Search | EP | 24.40 |
| Neural Network | **Search** | **Laplace** | **29.30** | MAP | EP | 30.89 |
| Linear | **Search** | **Laplace** | **23.05** | MAP | Laplace | 24.03 |
| Rational Quadratic | **Search** | **Laplace** | **24.58** | Search | Laplace | 25.37 |
| Periodic | **MAP** | **Laplace** | **24.50** | Search | EP | 25.19 |

In comparison of the best result of single-latent and multi-latent with each other, it can be observed that the lowest misclassification error rate for all the covariance functions is achieved for the single-latent GPC model. Even though the difference is

very subtle in most cases, it can be concluded that the OVA multi-class scheme using the underlying binary single-latent GPC model can supersede the single machine multi-latent GPC model.

Table 21 needs to be further analyzed in the search of the best covariance function arrangement. It can be observed from the table that Matern 3.2 using grid search strategy, the Laplace approximation inference method, and probit likelihood function gives the lowest averaged misclassification error rate of 22.96% among all of the other models. Linear and squared exponential are the ones that come next. This specific configuration and parameterization of GPC model will be evaluated on the test set and compared with the best RVM and SVM classifiers in section 4.1.4.

### 4.1.2 RVM

RVM is another probabilistic model examined in the experiment. Now that the best performance of GPC is achieved, the abstract idea proposed beforehand that sparsity would ruin the accuracy in uncertainty prediction needs to be supported experimentally. The experiment on RVM proceeds by selecting a kernel function and then using the same approach as GPC to cross-validate its hyperparameter. The manual search strategy will be followed and the value giving the lowest averaged misclassification error rate will be chosen. SparseBayes software [47] is used for the implementation of the RVM in all of the related experiments.

The Gaussian kernel function which is the same as squared exponential covariance function used in GPC will be examined. This kernel is the primary choice in the literature and has shown successful results in many experiments. For the sake of completeness and fair comparison with other models, this will serve as the basis for all of the three GPC, RVM, and SVM models.

As it is depicted in Figure 5, the only hyperparameter of the kernel function, length-scale, is cross-validated. The curves u-shape behavior reveals the lowest averaged misclassification error rate on the 5 cross-validation sets is achieved for $l = 22$ and

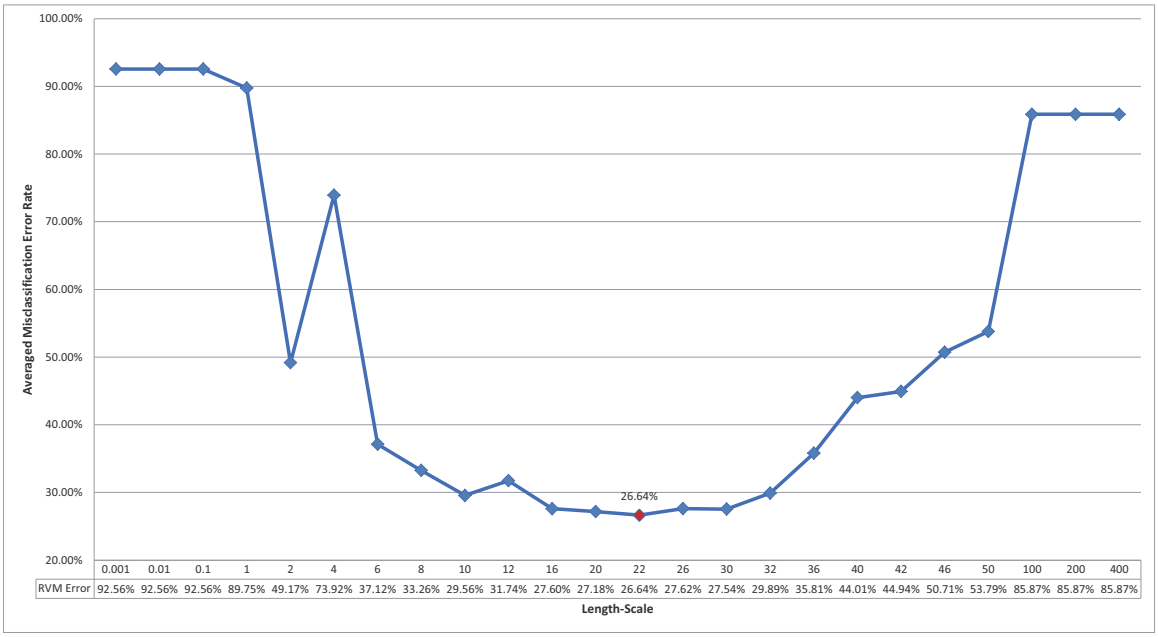


Figure 5: Averaged misclassification error rates of RVM using Gaussian Kernel on MNIST

the error rate is then 26.64%. At the first glance, it is observed that the averaged cross-validation error rate is not as low as the one achieved for GPC, but the true model performance will be measured for all the three models on the test set based on the test misclassification error rate, rejection rate and AUC of ROC curve in section 4.1.4.

### 4.1.3 SVM

SVM is considered in this part of the experiment to support the argument that non-probabilistic decision machines are not properly capable of providing true estimate of the posterior probability of the membership class. Since they are principally non-probabilistic and the output score they provide is the distance to the hyperplane, any attempt to convert them to a posterior probability estimate will not be as accurate as the probability output of a true probabilistic model such as GPC or even RVM. The other issue with SVM is that it strongly supports sparsity and follows naturally the mechanism to finalize the predicted class label decision using the sparse set of

samples. It will be concluded that in contrast to non-sparse models such as GPC, the rejection performance of the model is not comparative. LIBSVM [7] is the SVM implementation that is employed in all of the related experiments .

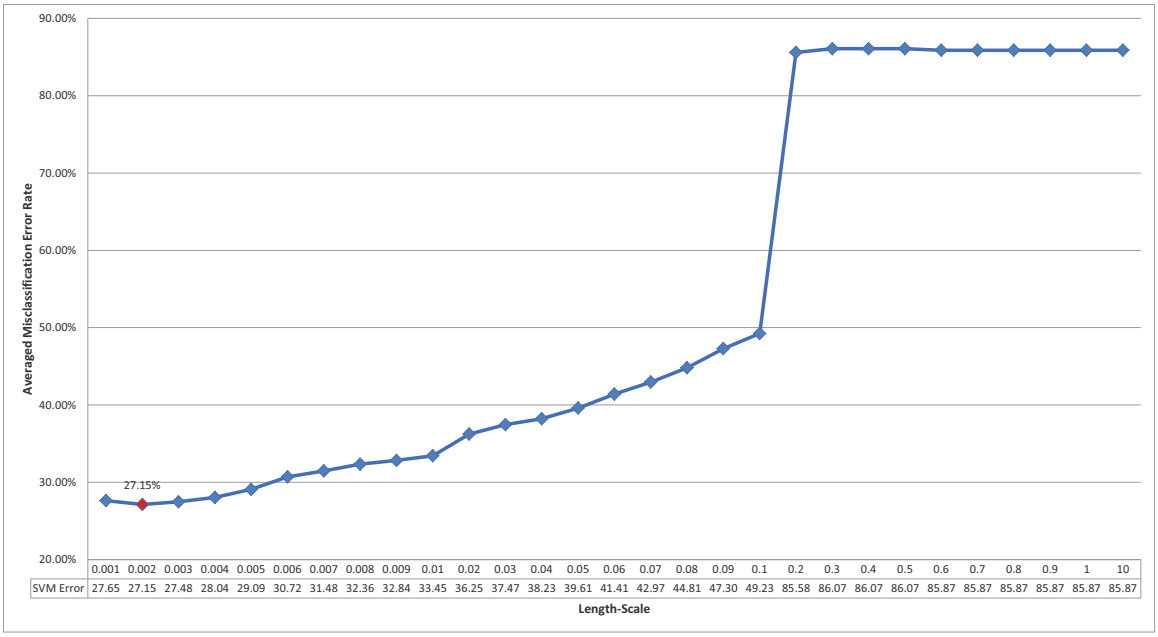


Figure 6: Averaged misclassification error rates of SVM using Gaussian Kernel on MNIST

The same as the experiment on RVM, a Gaussian kernel is chosen for SVM. The hyperparameter of length-scale will be cross-validated in the same manner. It is illustrated in figure 6 that the lowest averaged misclassification error rate of 27.15% is obtained for length-scale value of 0.002. The regularization parameter $C$ can also affect the behavior of SVM. The best value for this parameter is obtained for $C = 10$ and has been used throughout the experiment.

From the early analysis of the three models, it can be observed that GPC is the superior while RVM comes next in terms of the recognition performance on the cross-validation sets. However, the ultimate discussion on the model accuracy and reliability superiority will be given in the next section 4.1.4.

### 4.1.4 Discussion: Comparison of the three models on MNIST

The experiment on the test set proceeds as follows. First, the 100-dimension features are extracted using the trained CNN. Next, the test samples will be fed into the SVM trained to extract the most informative training samples known as support vectors form the whole training set. Having trained the GPC classifier at the third stage using the best configuration and parameterization, the best cutoff point for the rejection criterion of FTRM resulting into the highest possible reliability rate will be derived on the entire test set. It is worth mentioning again that the choice of this threshold is not the main focus of the research. It is chosen as so in order to thoroughly describe the efficiency and superiority of GPC model over SVM or even RVM and the overall idea of using principally probabilistic classifiers for the implementation of reject option. Apparently once the proposed system is proven experimentally, in practical applications any choice of threshold based on the application criticality and design criteria would result into higher recognition along with lower rejection rate. For the sake of simplicity and fair comparison, the cutoff point at 100% reliability is decided for the first stage SVM model. The rejected samples with low prediction certainty will be resulted from the third stage classifier consequently.

It should be noted that the overall performance of each model can be compared with one another free from the constraint imposed by the choice of the threshold using the concept of AUC of ROC curve described in chapter 2. In this regard, the classifier performance is examined in a way that a classification model with a higher AUC value is capable of reaching a lower rejection rate no matter what particular threshold value will be going to be decided. In other words, the model with higher AUC value would guarantee that any cutoff point can result in less rejected samples when compared to the model with lower AUC value.

The generalization performance of all three models for the best acquired configuration and parameterization are illustrated through using the number of rejected test samples reaching 100% reliability rate, AUC and test misclassification error rate in

Table 22. It is observed that Gaussian process classifier can significantly outperform the other two classifiers in terms of the number of rejected test samples and AUC value. The argument that the probabilistic classifiers can provide lower rejection rate is experimentally supported. SVM has by far the worst result of 1275 rejected samples while the RVM comes next with 893 rejected samples out of the whole 10k test samples. This implies that even though both SVM and RVM are sparse kernel machines, due to the principally probabilistic nature of RVM, estimates of uncertainty prediction are more accurate than the SVM. The GPC has the smallest number of rejected samples of 148. This achievement proves all the theoretical and experimental modeling justification for the superiority of the principally probabilistic classier of the GPC formulated through the Bayesian paradigm. It is supported that the transformation of the geometric SVM output to probability outputs is not helping to achieve a low rejection rate and consequently would impose high costs to the whole recognition system at the end.

Table 22: Comparison of the best models of GPC, RVM, and SVM based on the number of rejected samples, AUC, and test error rate on MNIST

| | Rejected Samples | AUC | Error Rate |
|---|---|---|---|
| SVM | 1275 | 0.9902 | 0.30% |
| RVM | 893 | 0.9906 | 0.37% |
| **GPC** | **148** | **0.9965** | **0.31%** |

The AUC values can further imply the superiority of GPC over the other two models. GPC has the AUC of 0.9965 whereas it is 0.9902 for SVM and 0.9906 for RVM. This implies that for any choice of the cutoff point, it is expected to have smaller number of rejected samples for GPC rather than for SVM, or even RVM. The comparison between GPC and RVM indicates that the sparsity argument is valid. It can be supported that since GPC is a nonparametric and non-sparse model, it could harness a better estimate of the prediction probability using all the training samples and therefore provide a better rejection performance. Furthermore, the superiority of RVM to SVM is explicit due to the principally probabilistic nature of RVM. For

both the rejection rate and AUC, RVM could outperform SVM. It should be noted that this rejection superiority is achieved despite the better recognition performance of SVM over RVM.

In Figure 7, some of the rejected test samples from the MNIST numeral data set using the multistage GPC system are illustrated. It is very clear that most of them are very ambiguous an prone to mistake for even human eyes. Despite the fact that the recognition system could correctly classify them, they are rejected and discarded from being predicted due to the low prediction uncertainty. Figure 8 depicts the misclassified test samples. It is apparent that most of them are very difficult to be classified. the variance in the curvature, discontinuities, and stroke thickness incurred over these samples are the main reasons behind being misclassified. However, it is strange why some of the obvious samples such as sample number 1243 are misclassified. It is evident that these misclassified samples are also rejected once the threshold reaching 100% is chosen.

Furthermore, if the test error rates are investigated, it is revealed that the model giving the lowest test error rate is not necessary capable of providing the best rejection result. SVM as a fast and accurate decision machine is capable of achieving the lowest error rate of 0.30% on the MNIST test set whereas GPC is slightly higher. Though RVM has the error rate of 0.37%, in contrast to the non-probabilistic SVM, it can reach a much smaller number of rejected samples. Therefore, it can be concluded that the characteristics of GPC has made it an appropriate choice of model to be utilized for the purpose of reject option. Additionally, SVM might be better to be used for the task of recognition rather than rejection.

Table 23: Comparison of the best model of the Emb-GPC with recent rejection results on MNIST

| | Rejected Samples |
|---|---|
| Hybrid CNN-SVM[33] | 560 |
| Com-DR[50] | 409 |
| **Emb-GPC** | **148** |


Figure 7: Some MNIST Rejected Test Samples

The embedded GPC (Emb-GPC) is defined as so to distinguish it from the previous researches and proposals in the handwriting literature. It is obvious that the naming stems from the fact that this multistage system has the GPC embedded at the core stage of the system. The Emb-GPC rejection performance can be compared with the two recent classification systems in terms of the number of rejected samples on MNIST numeral data set in Table 23. The Hybrid CNN-SVM[33] claims to achieve the lowest


Figure 8: MNIST Misclassified Test Samples

recognition rate of 0.19% on MNIST. The number of rejected test samples for this system is 560. The Com-DR[50] is recently proposed using the approach of multiple classifier systems, by combining two rejection criteria, and the extension of training set under various distortion settings. The system can achieve 409 rejected samples on the MNIST. It can be observed that the combination method is not always the solution to pattern recognition and machine learning problems. The result of 148 for rejected samples achieved by the proposed multistage system utilizing Emb-GPC obviously proves that a fundamentally appropriate classification model to the problem does truly approach better results. The Bayesian nonparametric modeling of Gaussian process model could improve the result of the best systems, to our knowledge, for more than 60%. The rejection rate 1.48% achieved using Emb-GPC on MNIST data set is known to be the lowest in the literature so far.

## 4.2 CENPARMI Numeral Data Set

CENPARMI numeral handwritten data set [45] is considered in the literature as another benchmark that many proposed models have been evaluated on. This data

set consists of a 4000 sample training set and a test set of 2000 samples. In each set, an equal number of samples for each class is included. Figure 9 illustrates some random samples of the numerals in the data set.

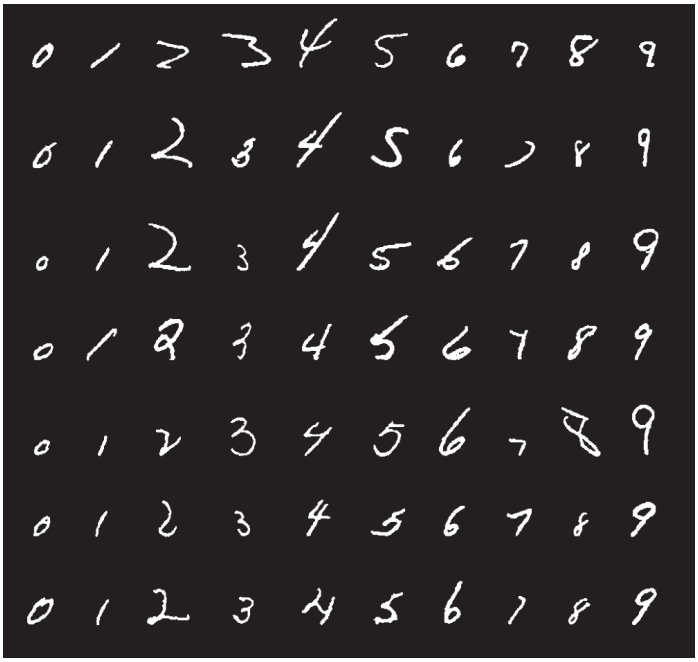

Figure 9: Samples from the CENPARMI numeral data set [45]

In the first stage, the CNN feature extractor needs to be trained. Following the same simplified CNN architecture described in the MNIST experiment in section 4.1, the training samples will be fed into the CNN and after the error rate becomes stable, the training will stop. At this point, the 100-dimension feature vector will be extracted and passed to the next stage of the experiment.

Next, an SVM model has to be trained using the samples represented in the new feature space. The role of the SVM here is to provide the most informative training samples known as support vectors to the next stage classifiers such as GPC or RVM. Following the cross-validation approach applied to the entire training set, the SVM classifier is trained using the RBF kernel function. The best hyperparameter value of $\sigma = 0.001$ and regularization model parameter of $C = 128$ is achieved. This specific SVM gives a support vector set of the size 485. These informative training samples

are extracted from the training set to form the new set for the next stage classifier.

The same as the experiment on the MNIST data set given in section 4.1, the 5-fold cross-validation will be applied to the training set. The manual search strategy will be considered for all of the classifiers to find the best hyperparameter values on the cross-validation sets. In the following sections, the experiment proceeds with first GPC, then RVM and finally SVM models evaluated on the data set. At the end, the comparison and discussion on the best model will be given.

### 4.2.1 GPC

The experiment is conducted on the CENPARMI data set using both the single-latent and multi-latent GPC classifiers. The inference method of Laplace approximation is chosen for the both classifiers. The Squared Exponential covariance function is selected for the GPC for which there are equivalent counterparts of Gaussian or RBF kernels for the two other classifiers. The hyperparameter values will be cross-validated in the manual search, finding the value that gives the lowest averaged misclassification error rate.

Table 24: Averaged misclassification error rates of Single-Latent GPC using SE covariance function

| Hyperparameter | | Magnitude | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 0.01 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 |
| | 0.1 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 | 91.17 |
| | 1 | 29.97 | 28.48 | 27.87 | 27.87 | 28.27 | 28.27 | 28.48 | 28.30 |
| | 10 | 32.23 | 22.00 | 11.69 | 7.03 | 5.78 | 5.36 | 5.57 | 5.36 |
| | 50 | 87.59 | 57.44 | 24.90 | 11.48 | 7.27 | 6.02 | 5.17 | **4.75** |
| | 100 | 87.59 | 84.85 | 39.57 | 19.69 | 10.29 | 8.64 | 7.27 | 6.02 |
| | 200 | 87.59 | 87.59 | 67.39 | 27.63 | 17.07 | 13.12 | 10.29 | 8.64 |
| | 400 | 87.59 | 87.59 | 87.59 | 52.87 | 26.54 | 21.95 | 17.07 | 13.12 |

The same 2-dimension grid search strategy like the one used in the MNIST experiment is followed in this part for the squared exponential covariance function for which two hyperparameters exist. The averaged misclassification error rate for each

pair of length-scale and magnitude is illustrated in Table 24 for the single-latent GPC model. It can be observed that the lowest rate of 4.75% is achieved for length-scale $l = 50$ with a magnitude of $\sigma_f^2 = 400$.

Table 25: Averaged misclassification error rates of Multi-Latent GPC using SE covariance function

| Hyperparameter | | Magnitude | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.01 | 0.1 | 1 | 10 | 50 | 100 | 200 | 400 |
| Length-scale | 0.01 | 88.60 | 88.44 | 89.08 | 89.99 | 91.21 | 92.46 | 91.82 | 91.61 |
| | 0.1 | 88.60 | 88.44 | 89.08 | 89.99 | 91.21 | 92.46 | 91.82 | 91.61 |
| | 1 | 88.41 | 88.44 | 88.47 | 89.57 | 91.00 | 92.46 | 91.82 | 91.61 |
| | 10 | 39.57 | 30.62 | 15.77 | 9.49 | 7.61 | 6.39 | **5.60** | 5.97 |
| | 50 | 87.59 | 80.47 | 35.92 | 16.22 | 9.47 | 8.28 | 7.09 | 6.24 |
| | 100 | 87.59 | 87.59 | 61.46 | 25.11 | 14.98 | 11.90 | 9.68 | 8.06 |
| | 200 | 87.59 | 87.59 | 87.59 | 45.69 | 23.86 | 19.48 | 14.98 | 11.90 |
| | 400 | 87.59 | 87.59 | 87.59 | 72.47 | 39.12 | 30.25 | 23.86 | 19.48 |

The results are given in Table 25 for the multi-latent GPC model following the same search strategy. The results in the table indicates that the hyperparameter values of length-scale $l = 10$ and magnitude $\sigma_f^2 = 200$ can provide the highest performance in terms of the lowest averaged misclassification error rate of 5.60%.

Comparing the best results of both GPC models with one another, it can be concluded that single-latent GPC can supersede the multi-latent GPC with a lower error rate. This model arrangement will be utilized to measure the model performance on the unseen CENPARMI test set. This result will be analyzed in contrast to the two other models to select the one that can provide higher performance at the end.

### 4.2.2 RVM

RVM classifier will be evaluated on the CENPARMI data set in this part. Following the same approach of section 4.2.1, for the sake of a fair comparison, a Gaussian kernel is selected for training the RVM. The length-scale hyperparameter needs to be cross-validated respectively.

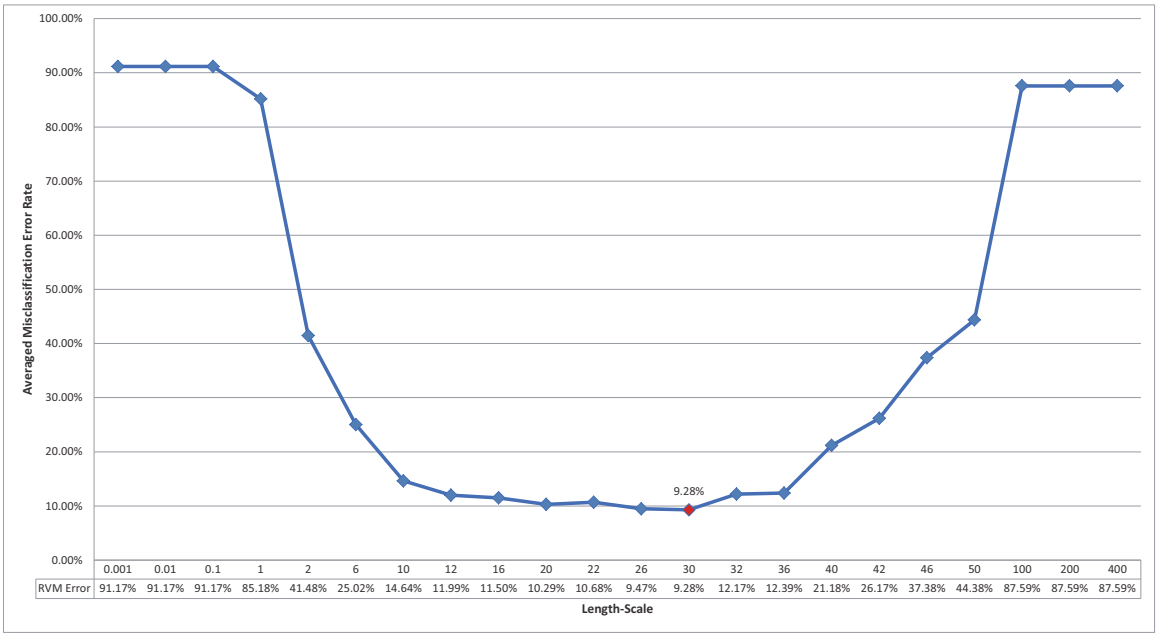


Figure 10: Averaged misclassification error rates of RVM using Gaussian Kernel on CENPARMI

Various values of the kernel functions only hyperparameter, length-scale, are cross-validated as the results are depicted in Figure 10. The lowest misclassification error rate of 9.28% is obtained for length-scale $l = 30$ based on the analysis of the u-shape error curve in the figure. This specific model with the same hyperparameter values will be examined later on the unseen test set to measure the ultimate performance and generalization capacity of the RVM.

### 4.2.3 SVM

The experiment on CENPARMI data set is further expanded to SVM classifier. It will be trained in the same manner as the two previous classifiers in this part of the experiment in order to see whether the proposed idea of GPC superiority supported empirically in the MNIST experiment can be generalized to CENPARMI data set.

Gaussian or RBF kernel is chosen similar to section 4.2.2. The kernel has one hyperparameter, length-scale that need to be cross-validated through the same manual search strategy. Various values are examined as illustrated in 11, and the averaged

misclassification error rates are measured respectively. From the figure, it can be outlined that the lowest error rate of 6.60% is achieved for the length-scale value of 0.002 after using the best regularization parameter $C = 10$ which has been preliminary experimentally verified.

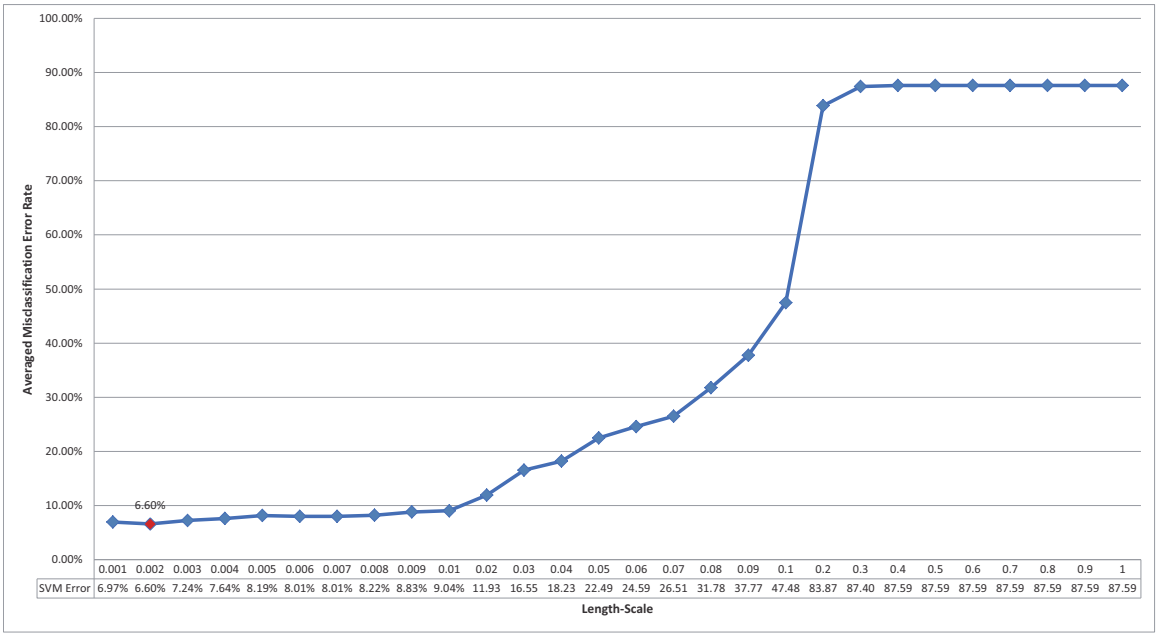


Figure 11: Averaged misclassification error rates of SVM using Gaussian Kernel on CENPARMI

### 4.2.4 Discussion: Comparison of the three models on CENPARMI

The GPC, RVM, and SVM classifiers are now completely evaluated on CENPARMI handwritten numeral data set. The best model configuration and parameterization is already decided. It is time to measure the model performance on the unseen test set in terms of the rejected sample, reliability rate, AUC, and misclassification error rate.

Following the same approach as MNIST experiment, the test set will be examined by the third stage GPC classifier already trained based on the best configuration and parametrization resulted from the previous experiments. Using the cutoff point of 100% reliability rate following the same argument given in the MNIST experiment,

some test samples are rejected based on the output of the third stage classifier such as GPC while the rest is sent to have the class label predicted. It should be noted that the argument still holds that such threshold is chosen only for the sake of representation. The rejection and recognition rates along with AUC value will be measured based on the performance of the model on the entire test samples.

The test results of the three classifiers in terms of the number of rejected samples, reliability rate and AUC are illustrated in Table 26. To achieve a fair comparison with the best results in the literature, the number of rejected samples reaching the similar reliability rate of 99.93% is measured. As it is shown in the table, the number of rejected samples is significantly lower for GPC when compared with SVM and RVM. It is 318 test samples rejected for a slightly higher reliability rate of 99.94%. This result strongly implies the generalization of this argument that the Bayesian nonparametric classification model of Gaussian process can notably outperform the two other classifiers of SVM and RVM. The argument can further be supported considering the AUC of GPC which has the highest value of 0.9654. Therefore, the generalization to any threshold value still holds on this data set.

Table 26: Comparison of the best models of GPC, RVM, and SVM based on the number of rejected samples, AUC, and test error rate on CENPARMI

| | Rejected Samples | Reliability Rate | AUC | Error Rate |
|---|---|---|---|---|
| SVM | 667 | 99.92% | 0.9570 | 2.40% |
| RVM | 881 | 99.91% | 0.9238 | 2.55% |
| **GPC** | **318** | **99.94%** | **0.9654** | **2.40%** |

Furthermore, SVM and RVM can be compared to one another. The observation is not strong enough to support the fact that the superiority of the probabilistic sparse classifier of RVM to the SVM can generalize sufficiently to CENPARMI data set. Though the number of rejected samples is close to the SVM, it does not outnumber the one for the SVM. The AUC values for both classifiers confirm this observation. Despite this subtle inferior result, the choice of the kernel and hyperparameter value

might improve the RVM classifier performance and consequently result into an outstanding result compare to the SVM. It is important to note that the test misclassification error rate of GPC is as good as SVM despite the fact that SVM has shown recognition superiority in the literature.

Some CENPARMI rejected test samples are represented in Figure 12. First, the size of the samples are variant. This makes some samples too small to carry discriminative enough information and consequently the classification prone to mistake. Most of the rejected samples are even strictly difficult to be classified by humans. Despite this difficulty, they could correctly be classified by the multistage recognition system using GPC. However, they are rejected due to the high prediction uncertainty measured by GPC. The CENPARMI misclassified samples are depicted in Figure 13. Some of them are mislabeled or mis-segmented such as the sample number 783. Though some are very easy to classify for humans, the variability in writing that particular numeral in different formats may have cause the recognition system from classifying it correctly.


Figure 12: Some CENPARMI Rejected Test Samples

The comparison is further developed with the recent outstanding results given in Table 27. Following the same naming convention, the Embedded GPC (Emb-GPC)

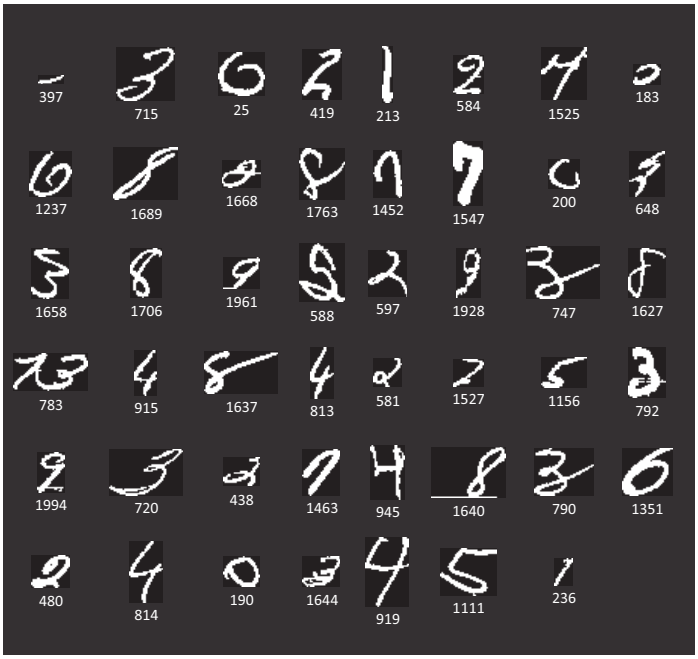


Figure 13: CENPARMI Misclassified Test Samples

is the multistage system proposed in this research utilizing the GPC at the third stage. On one hand, it is demonstrated that Emb-GPC reaches lower number of rejected samples when compared with LDAM [17] under the same reliability rate of approximately 99.67%. GPC could achieve 145 rejected test samples whereas LDAM is not capable of getting any better number than 175 for a lower reliability rate of 99.67%.

Table 27: Comparison of the best model of GPC with recent rejection results on CENPARMI

| | Rejected Samples | Reliability Rate |
|---|---|---|
| LDAM[17] | 175 | 99.67% |
| **Emb-GPC** | **145** | **99.67%** |
| Com-SM[50] | 393 | 99.94% |
| **Emb-GPC** | **318** | **99.94%** |

Moreover, the Emb-GPC, the multistage system utilizing GPC, can significantly outperform the result of the recent research using multiple classier system (MCS) approach known as Com-SM [50]. Com-SM uses a combination of Convolutional Neural Networks with different structures and use voting mechanism to improve the rejection

performance. Since the GPC is principally probabilistic and fundamentally estimate the true prediction uncertainty, the number of rejected samples can dramatically surpass the one in the Com-SM. the proposed system with GPC can obtain 318 rejected samples whereas for the same reliability rate, it is 393 for the Com-SM. This implies an improvement of around 20% in the final rejection performance. It is obvious that the result achieved using the GPC at the core of the multistage system can only be compared with the models and approaches applied to the original CENPARMI data set. The manipulation and extension of the original data set into various sets would obviously improve the results and make the comparison unfair. Therefore, such comparisons are prevented in the research.

## 4.3 Conclusion

The proposed system is experimentally evaluated on two well-known handwritten numeral data sets: MNIST and CENPARMI. For each one, the feature extraction using CNN is implemented. Then, the informative training data extraction is followed by the use of an SVM classifier. The new set is used to train the primary classifier of GPC. Different combination of inference methods, likelihood functions, multi-class classification schemes and covariance functions are attempted in the hope of finding the best configuration and parameterization of the model to achieve the highest generalization to the unseen data sets.

The work is extended to the other probabilistic classifier known as the RVM. The best hyperparameter value is cross-validated on the training set. For a fair comparison, an SVM is trained to see how it can perform compared to the previous classifiers. In the final part of the experiment, the performance of three models is evaluated on the test set. It is concluded that GPC could significantly supersede the two others in terms of the rejection rate and AUC value despite a slightly lower recognition rate when compared to SVM.

The superiority of GPC over RVM proves that the concept of sparsity does not necessary provide the appropriate means to achieve a true estimate of the posterior probability of class membership that could form the base of reject option. Despite the slow prediction process caused by considering the training samples all at once, the prediction probability accuracy could provide a much lower rejection rate in the end.

# Chapter 5

# Conclusion

The idea of modeling the pattern recognition and machine learning problems so perfectly such that the test misclassification error reaches zero has been sought so dramatically in the whole research history. Despite many strong modeling approaches leading to powerful classifiers proposed so far, the zero error rate has not been achieved yet. Consequently, there are always some misclassified test samples that impose heavy cost on practical applications especially in the field of finance and medicine. The intuition of reject option is the natural outcome of the separation of the inference and decision making procedures from one another. This separation provides proper a means on which a reliable rejection criterion can be utilized and the test samples with low prediction certainty are rejected in the decision making process. This will significantly reduce the misclassification cost of the whole system. However, it is obvious that the cost shifts to the intervention of human expert in an automatic classification process. This cost transformation is not desired at all. Therefore, a system to provide both low misclassification error rate and low rejection rate is strongly sought. Such system can be achieved by using probabilistic modeling approaches to classification.

Two probabilistic classification approaches to model specifically handwriting data sets and improve the reject performance are investigated in depth in this research. GPC as a Bayesian nonparametric model has been extensively utilized while RVM as

a probabilistic sparse kernel machine is also scrutinized. A hybrid, cascade framework is originally proposed based on the extracted CNN feature space and the utilization of SVM support vector set. These classifiers are regarded as the core of this framework to provide a reliable estimate of the posterior probability of class membership for the test samples. Then, using a promising rejection criterion, the superiority in terms of the rejection rate to other models such as SVM is achieved. All contributions of this work is presented in 5.1 while the possible countable future work that could further extend this approach will be mentioned in 5.2.

## 5.1 Contributions

This research contributes to the field of handwriting recognition system design through the proposal of the authentic intuition of the hybrid cascade integration of the probabilistic models of Gaussian process classifier or relevance vector machine with the nonparametric decision machine of SVM using the dimensionality-reduced CNN feature space. This novel classification framework proves the significant improvement in the rejection rate on two famous handwriting numeral data sets: MNIST and CENPARMI. The main contributions of this thesis can be summarized below.

Firstly, a hybrid cascade framework is proposed to cover the following two issues. First, the probabilistic model of Gaussian Process Classification (GPC) can be directly applied into practical applications and then benefited for the sake of low rejection rate. Second, the issue of its high computation complexity gets addressed in an accurate and reliable framework. CNN feature extraction method is considered as the entry block to the system that takes care of extracting spatial correlations and reducing the input space. Dealing with 2-D image space needs careful attention as misleading it would result in low performance results. At the next stage, the powerful capability of SVM in accurately extracting the most informative data samples to get around the problem of big data sets is properly utilized. At the core stage, the

probabilistic model of GPC is embedded such that using the low dimensional feature space of the reduced sized support vector set, the task of classification and rejection is addressed exclusively. This framework is extensively evaluated on the MNIST and CENPARMI numeral data sets. The acquired rejection rate of 1.48% is outstanding and superior to two previously proposed systems of Hybrid CNN-SVM[33] with 5.60%, and Com-DR[50] with 4.09% under 100% reliability rate. The 65% performance improvement shows how properly utilizing relevant classifiers with particular properties can strongly lead to a better result. GPC performance experimentally reveals that indeed a true probabilistic prediction is essential in this regard. Any attempt to derive a probabilistic interpretation from not-principally-probabilistic models such as SVM is prone to fail. It can be further concluded that a simple but principally proper classification model capable of improving the results is always needed. A unified GPC model with a particular covariance function and hyperparameter values could supersede the rejection rate of a complicated combination system.

Secondly, the Gaussian process classifier is applied to handwriting recognition problem. Different inference methods, likelihood functions, covariance functions and learning procedures are extensively experimented. Single-latent and multi-latent GPC models are specifically compared with one another. It is observed that the simple straightforward single-latent GPC approached under OVA multi-class classification scheme can easily supersede the more complicated single-machine multi-latent GPC approach. Laplace approximation method through the use of a manual search strategy for hyperparameters could mostly achieve the best results under different experiment scenarios.

Thirdly, despite the probabilistic nature of both GPC and RVM models, it is implied that sparsity has a detrimental effect on the rejection performance. The utilization of the entire extracted SV set in the GPC plays one of the main roles in giving an accurate estimate of prediction uncertainty. On the other hand, the sparse formulation of the RVM leads to a fast training and testing procedures but it does

not reach a rejection rate as low as the GPC. The error rate obtained using the GPC on the MNIST is 1.48% whereas it is 8.93% using RVM. This achievement is further proven in the generalization to CENPARMI data set. 318 rejected samples using GPC under the reliability rate of 99.94% is obviously superior to 881 using RVM under the reliability rate of around 99.91%.

Lastly, it is observed on the MNIST data set that the probabilistic sparse model of RVM could outperform non-probabilistic classifier of SVM. This evidence could further support the argument of the superiority of the true probabilistic predictions achieved through RVM. This is empirically obvious by comparing the rejection rate of 8.93% for RVM with 12.75% for SVM on MNIST. However, this observation is not completely supported in the experiment on the CENPARMI. For both models, the numbers of rejected samples are quite close to one another. This might be justified based on an inappropriate kernel choice for RVM.

## 5.2 Future Work

The design of recognition systems will never end as long as the 100% recognition rate is not achieved. Reducing the misclassification cost while lowering rejection rate and improving the recognition rate can be further studied by conducting the following lines of research.

FTRM is chosen as the rejection criterion based on its simplicity and superiority. Other new criteria have been proposed recently. Using this classification framework, other rejection criteria can be experimented in depth to see whether now that a proper probabilistic measure is available, any utilization of such criteria can further improve the rejection rate when compared to FTRM.

In the combination of multiple classifier system approach, it is crucial to have normalized probabilistic predictions from each classifier so that the combination policy could be properly applied. The GPC model with different covariance functions can

be applied in a MCS approach to further improve the rejection rate.

The other possible future work that can be done is the integration of CNN committees with different GPC models with variable properties. The data set invariance ca n be integrated into the recognition system by following the extension of the data set. Extraction of different feature types along with a combination of various GPC classifiers in a consistent manner could be studied. This data set extension could provide the same recognition rate while lowering the rejection rate.

This hybrid cascade framework, which relies on the probabilistic models of GPC or RVM, can be further studied in the field of handwriting character recognition by focusing on the recognition, rejection and practical feasibility such as training speed. Though it is fundamentally guaranteed a significant improvement can be achieved over other non-probabilistic models, it is worth to be experimentally investigated. Since the issue of a larger number of classes in character recognition system design seems burdensome and GPC has very slow training and prediction procedures, the practical feasibility should be investigated extensively. Different types of writing style cause higher variability in the data set. Moreover, the way of dealing with upper-case and lower case sets needs to be specifically addressed. Having considered these issues, it could be interesting to conduct a line of research to see the feasibility of GPC model on the character recognition problem.

# List of Acronyms

**GPC** Gaussian Process Classification

**SVM** Support Vector Machine

**RVM** Relevance Vector Machine

**KNN** K-Nearest Neighbor

**MLP** Multi-Layer Perceptron

**HMM** Hidden Markov model

**RBF** Radial Basis Function

**DBN** Deep Belief Network

**RBM** Restricted Boltzmann Machine

**CNN** Convolutional Neural Network

**MCS** Multiple Classifier System

**FRM** First Rank Measurement

**FTRM** First Two Rank Measurement

**SVMM** SVM-based Measurement

**AUCM** Area Under the Curve Measurement

**ARD** Automatic Relevance Determination

**SV** Support Vector

**ROC** Receiver Operating Characteristic

**AUC** Area Under the Curve

**GPR** Gaussian Process Regression

**IRLS** Iteratively Reweighed Least Squares

**MAP** Maximum A Posteriori

**EP** Expectation Propagation

**MCMC** Markov Chain Monte Carlo

**OVA** One-Vs-All

**AVA** All-Vs-All

**KFCV** K-Fold Cross-Validation

**Emb-GPC** Embedded Gaussian Process Classifier

# References


[1] Milton Abramowitz and Irene A Stegun. Handbook of mathematical function with formulas, graphs, and mathematical tables. *National Bureau of Standsards, Applied Mathematics Series*, 55, 1970.

[2] Nafiz Arica and Fatos T Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(2):216–233, 2001.

[3] Zhen-Long Bai and Qiang Huo. A study on the use of 8-directional features for online handwritten chinese character recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 262–266. IEEE, 2005.

[4] Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[5] Christopher M Bishop. *Neural Networks For Pattern Recognition*. Oxford University Press, 1995.

[6] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 1. springer New York, 2006.

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE, 2012.

[9] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Juergen Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition. *ArXiv Preprint arXiv:1003.0358*, 2010.

[10] Jian-xiong Dong, Adam Krzyzak, and Ching Y Suen. Fast svm training algorithm with decomposition on very large data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005.

[11] Richard O Duda, Peter E Hart, et al. *Pattern Classification And Scene Analysis*, volume 3. Wiley New York, 1973.

[12] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. Feature extraction methods for character recognition-a survey. *Pattern Recognition*, 29(4):641–662, 1996.

[13] Gregory Dzuba, Alexander Filatov, Dmitry Gershuny, Igor Kil, and Vadim Nikitin. Check amount recognition based on the cross validation of courtesy and legal amount fields. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(04):639–655, 1997.

[14] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[15] Nikolai Gorski, Valery Anisimov, Emmanuel Augustin, Olivier Baret, and Sergey Maximov. Industrial bank check processing: The a2ia checkreadertm. *International Journal on Document Analysis and Recognition (IJDAR)*, 3(4):196–206, 2001.

[16] JA Hanely and BJ McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[17] Chun Lei He, Louisa Lam, and Ching Y Suen. Rejection measurement based on linear discriminant analysis for document recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(3):263–272, 2011.

[18] Alessandro L Koerich, Robert Sabourin, and Ching Y Suen. Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis & Applications*, 6(2):97–121, 2003.

[19] Louisa Lam and Ching Y Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[20] Fabien Lauer, Ching Y Suen, and Gérard Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6):1816–1824, 2007.

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–97. IEEE, 2004.

[23] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[24] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: Investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37(2):265–279, 2004.

[25] David JC MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[26] Peter McCullagh and John A Nelder. *Generalized Linear Model*, volume 37. Chapman & Hall/CRC, 1989.

[27] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and KR Mullers. Fisher discriminant analysis with kernels. In *IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing*, pages 41–48. IEEE, 1999.

[28] Thomas P Minka. *A Family Of Algorithms For Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[29] Tom M Mitchell. Machine learning. *Burr Ridge, IL: McGraw Hill*, 45, 1997.

[30] Radford M Neal. *Bayesian Learning For Neural Networks*. PhD thesis, University of Toronto, 1995.

[31] Radford M Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*, 1997.

[32] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary gaussian process classification. *The Journal of Machine Learning Research*, 9:2035–2078, 2008.

[33] Xiao-Xiao Niu and Ching Y Suen. A novel hybrid cnn–svm classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4):1318–1325, 2012.

[34] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

[35] WM Pan, TD Bui, and CY Suen. Isolated handwritten farsi numerals recognition using sparse and over-complete representations. In *10th International Conference on Document Analysis and Recognition*, pages 586–590. IEEE, 2009.

[36] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

[37] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144, 2006.

[38] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes For Machine Learning*. the MIT Press, 2006.

[39] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[40] Volker Roth and Volker Steinhage. Nonlinear discriminant analysis using kernel functions. In *Advances in Neural Information Processing Systems*. Citeseer, 1999.

[41] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[42] Bernhard Schölkopf and Alexander J Smola. *Learning With Kernels: Support Vector Machines, Regularization, Optimization And Beyond*. the MIT Press, 2002.

[43] Patrice Simard, David Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962, 2003.

[44] Ching Y Suen and Louisa Lam. Multiple classifier combination methodologies for different output levels. In *Multiple Classifier Systems*, pages 52–66. Springer, 2000.

[45] Ching Y Suen, Christine Nadal, Raymond Legault, Tuan A Mai, and Louisa Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80(7):1162–1180, 1992.

[46] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

[47] Michael E Tipping, Anita C Faul, et al. Fast marginal likelihood maximisation for sparse bayesian models. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, volume 1, 2003.

[48] Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. Gpstuff: Bayesian modeling with gaussian processes. *Journal of Machine Learning Research*, 14:1175–1179, 2013.

[49] Vladimir Vapnik. *The Nature Of Statistical Learning Theory*. Springer, 1999.

[50] Wei-Na Wang, Xu-Yao Zhang, and Ching Y Suen. A novel pattern rejection criterion based on multiple classifiers. In *Multiple Classifier Systems*, pages 331–342. Springer, 2013.

[51] Christopher K. I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[52] Christopher KI Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.